TMS320DM816x DaVinci Digital Media Processors

Technical Reference Manual

Literature Number: SPRUGX8C
March 2015

Contents

List of Figures

List of Tables

Read This First

About This Manual

This Technical Reference Manual (TRM) details the integration, the environment, the functional description, and the programming models for each peripheral and subsystem in the device.

The TRM should not be considered a substitute for the data manual, rather a companion guide that should be used alongside the device-specific data manual to understand the details to program the device. The primary purpose of the TRM is to abstract the programming details of the device from the data manual. This allows the data manual to outline the high-level features of the device without unnecessary information about register descriptions or programming models.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers may be shown with the suffix h or the prefix 0x. For example, the following number is 40 hexadecimal (decimal 64): 40h or 0x40
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties with default reset value below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure can have one of multiple meanings:
    - Not implemented on the device
    - Reserved for future device expansion
    - Reserved for TI testing
    - Reserved configurations of the device that are not supported
  - Writing non-default values to the Reserved bits could cause unexpected behavior and should be avoided.

Glossary

SLYZ022 — *TI Glossary*.

This glossary lists and explains terms, acronyms, and definitions.

Related Documentation From Texas Instruments

For product information, visit the Texas Instruments website at http://www.ti.com.

SPRS614— *TMS320DM816x DaVinci Digital Media Processors Data Manual.*

SPRUFE8— *TMS320C674x DSP CPU and Instruction Set Reference Guide.* Describes the CPU architecture, pipeline, instruction set, and interrupts for the TMS320C674x digital signal processors (DSPs).

SPRZ329— *TMS320DM816x DaVinci Video Processors Silicon Errata (Revisions 2.1, 2.0, 1.1, and 1.0).*

Community Resources

The following links connect to TI community resources. Linked contents are provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

TI E2E™ Online Community— ***TI's Engineer-to-Engineer (E2E) Community.*** Created to foster collaboration among engineers. At e2e.ti.com, you can ask questions, share knowledge, explore ideas and help solve problems with fellow engineers.

TI Embedded Processors Wiki— ***Texas Instruments Embedded Processors Wiki.*** Established to help developers get started with Embedded Processors from Texas Instruments and to foster innovation and growth of general knowledge about the hardware and software surrounding these devices.

Chip Level Resources

Topic **Page**

1.1 Introduction

This section describes the features, supporting subsystems, and architecture of this highly integrated, programmable device. A detailed summary of the key features, required usage information, and register descriptions is provided for the following primary components and subsystems:

- Microprocessor unit (MPU) subsystem based on the ARM® Cortex®-A8 microprocessor with ARM® NEON™ technology extension
- DSP Subsystem (DSPSS), including the C674x Megamodule and associated memory
- HD Video Co-processor Subsystems (HDVICP2) System MMU
- System MMU
- SGX530 graphics subsystem for 3D graphics acceleration
- HD Video Processing Subsystem (HDVPSS) for video capture and display
- Level 3 (L3) and Level 4 (L4) interconnects
- Device Clocking and Flying Adder PLLs
- Error Location Module for the General Purpose Memory Controller
- Inter-Processor Communications with Mailbox and Spinlock components
- Control module with all chip-level control and configuration registers
- Power, Reset, and Clock Management module
- Interrupt Controller
- Boot modes and booting procedures

1.2 MPU Subsystem

1.2.1 Introduction

The Microprocessor Unit (MPU) subsystem of the device handles transactions between the ARM core (ARM Cortex-A8 Processor), the L3 interconnect, and the interrupt controller (INTC). The MPU subsystem is a hard macro that integrates the Cortex-A8 Processor with additional logic for protocol conversion, emulation, interrupt handling, and debug enhancements.

Cortex-A8 is an ARMv7 compatible, dual-issue, in-order execution engine with integrated L1 and L2 caches with NEON SIMD Media Processing Unit.

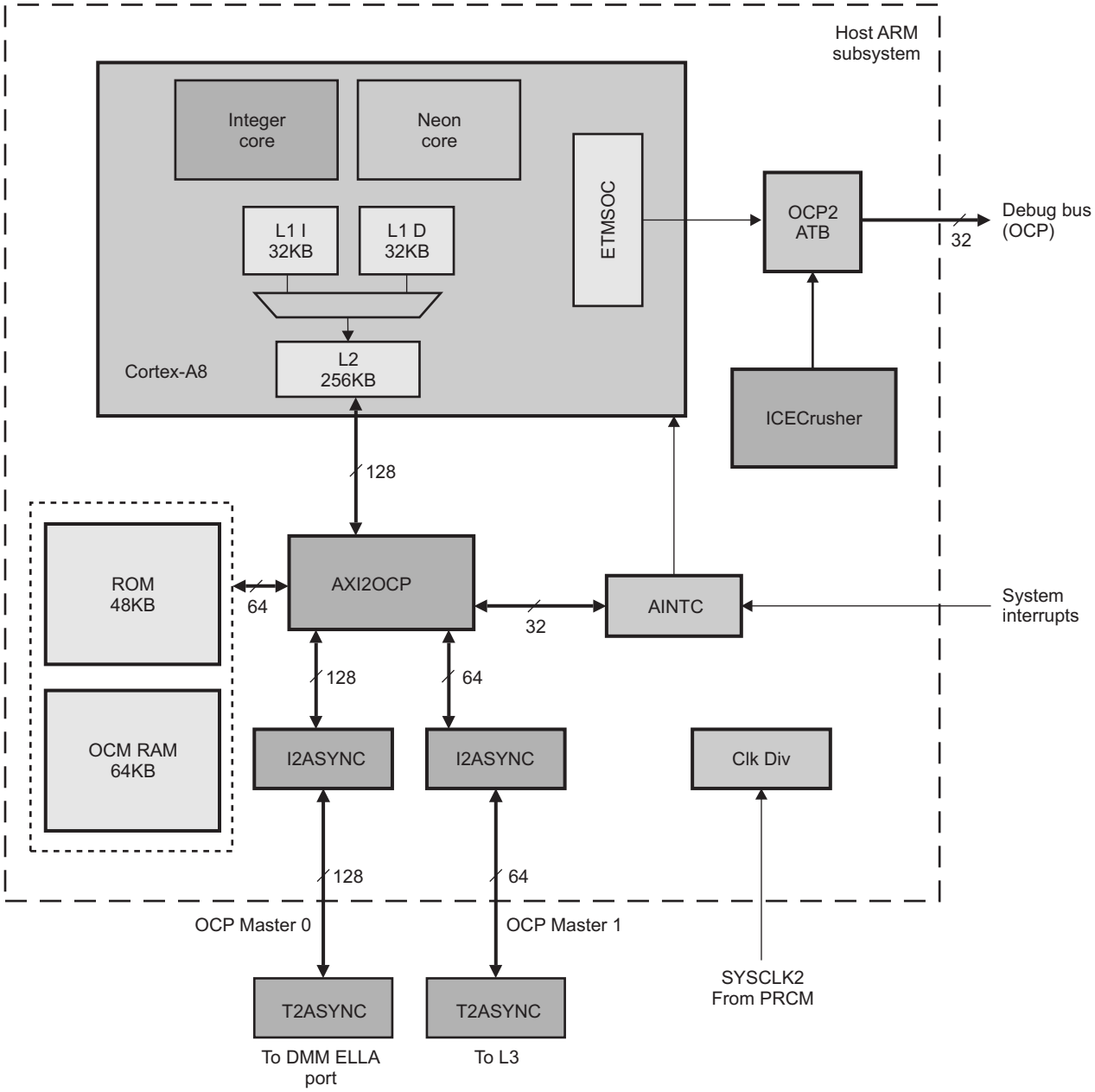
An Interrupt Controller is included in the MPU subsystem to handle host interrupt requests in the system.

The MPU subsystem includes ARM® CoreSight™ compliant logic to allow the debug subsystem access to the Cortex-A8 debug and emulation resources, including the Embedded Trace Macrocell.

The MPU subsystem has three functional clock domains, including a high-frequency clock domain used by the Cortex-A8. The high-frequency domain is isolated from the rest of the device by asynchronous bridges.

Figure 1-1 shows the high-level block diagram of the MPU subsystem.

Figure 1-1. Microprocessor Unit (MPU) Subsystem



1.2.2 Features

This section outlines the key features of the MPU subsystem:

- ARM Microprocessor
  - Cortex-A8 revision R3P2.
  - ARM Architecture version 7 ISA.
  - 2-issue, in-order execution pipeline.
  - L1 and L2 Instruction and Data Cache of 32 KB , 4-way, 16 word line with 128 bit interface.
  - Integrated L2 cache of 256 KB, 8-way, 16 word line, 128 bit interface to L1.
  - Includes the NEON media coprocessor (NEON) that implements the Advanced SIMD media processing architecture.
  - Includes the VFP coprocessor which implements the VFPv3 architecture and is fully compliant with IEEE 754 standard.
  - The external interface uses the AXI protocol configured to 128-bit data width.
  - Includes the Embedded Trace Macrocell (ETM) support for non-invasive debugging.
  - Implements the ARMv7 debug with watch-point and breakpoint registers and 32-bit Advanced Peripheral Bus (APB) slave interface to CoreSight debug systems.
- AXI2OCP Bridge
  - Support OCP 2.2.
  - Single Request Multiple Data Protocol on two ports.
  - Multiple targets, including three OCP ports (128-bit, 64-bit and 32-bit).
- Interrupt Controller
  - Support up to 128 interrupt requests
- Emulation/Debug
  - Compatible with CoreSight Architecture.
- Clock Generation
  - Through PRCM
- DFT
  - Integrated PBIST controller to test L2 tag and data ram, L1I and L1D data ram and OCM RAM.

1.2.3 MPU Subsystem Integration

The MPU subsystem integrates the following group of submodules:

Cortex-A8 Processor: Provides a high processing capability, including the NEON technology for mobile multimedia acceleration. The ARM communicates through an AXI bus with the AXI2OCP bridge and receives interrupts from the MPU subsystem interrupt controller (MPU INTC).

Interrupt controller: Handles the module interrupts (for details, see the *ARM Interrupt Controller (AINTC)* chapter).

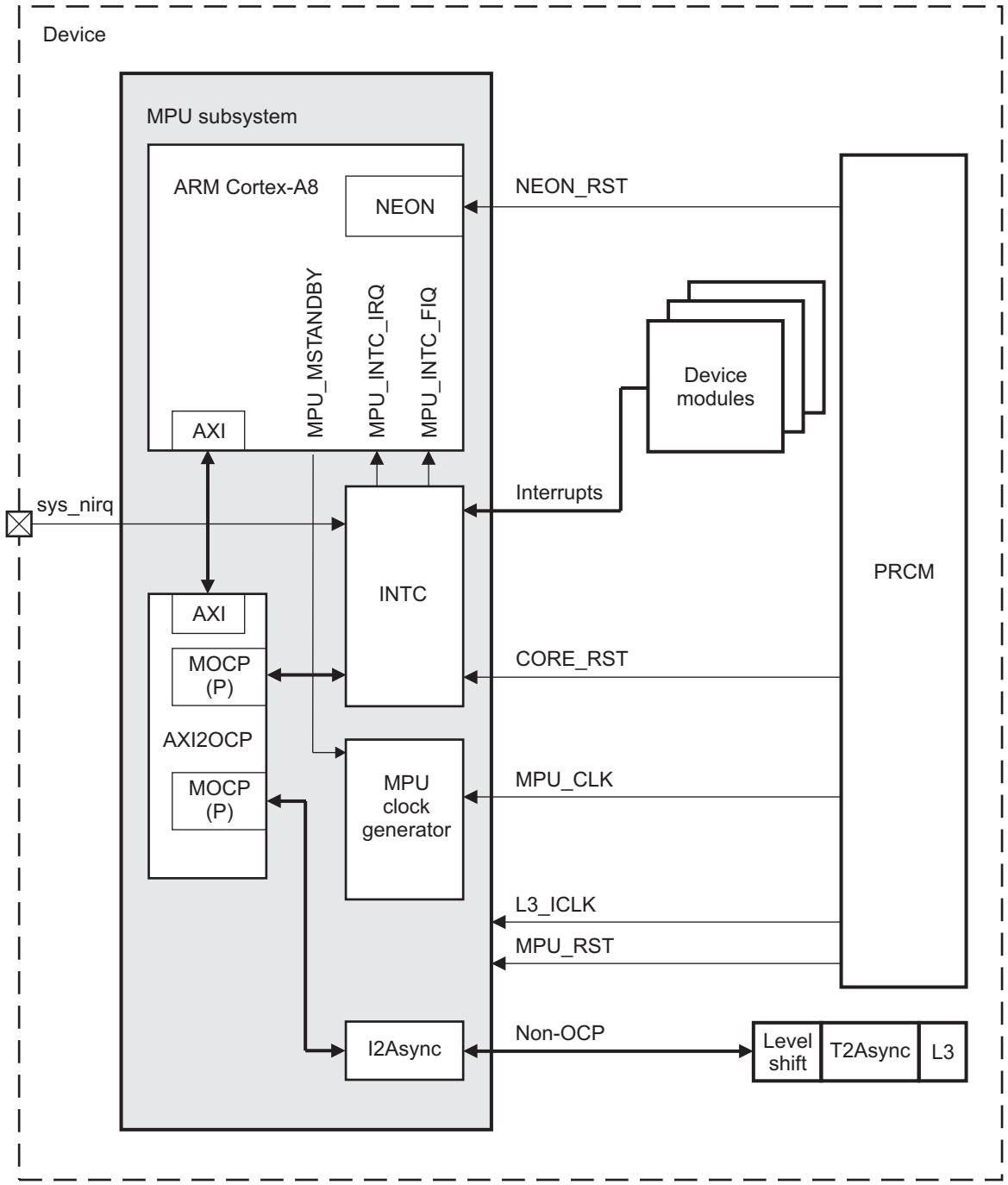
AXI2OCP bridge: Allows communication between the ARM (AXI), the INTC (OCP), and the modules (OCP L3).

I2Async bridge: This is an asynchronous bridge interface providing an asynchronous OCP to OCP interface. This interface is between the AXI2OCP bridge within the MPU subsystem and the T2Async bridge external to the MPU subsystem.

Clock Divider: Provides the required divided clocks to the internal modules of the MPU subsystem and has a clock input from SYSCLK2 that is applied by the power, reset, and clock management (PRCM) module of the device.

In-Circuit Emulator: It is fully Compatible with CoreSight Architecture and enables debugging capabilities.

Figure 1-2. MicroProcessor Unit (MPU) Subsystem Signal Interface



1.2.4 MPU Subsystem Clock and Reset Distribution

1.2.4.1 Clock Distribution

The MPU subsystem does not include an embedded DPLL. The clock is sourced from the PRCM module. A clock divider within the subsystem is used for deriving the clocks for other internal modules.

All major modules inside the MPU subsystem are clocked at half the frequency of the ARM core. The divider of the output clock can be programmed with the PRCM.CM_CLKSEL2_PLL_MPU[4:0]MPU_DPLL_CLKOUT_DIV register field, the frequency is relative to the ARM core. For details see the *Power, Reset, and Clock Management (PRCM)) Module* chapter.

The clock generator generates the following functional clocks:

ARM (ARM_FCLK): This is the core clock. It is the base fast clock that is routed internally to the ARM logic and internal RAMs, including NEON, L2 cache, the ETM core (emulation), and the ARM core.

AXI2OCP Clock (AXI_FCLK): This clock is half the frequency of the ARM clock (ARM_FCLK). The OCP interface thus performs at one half the frequency of ARM.

Interrupt Controller Functional Clock (MPU_INTC_FCLK): This clock, which is part of the INTC module, is half the frequency of the ARM clock (ARM_FCLK).

ICE-Crusher Functional Clock (ICECRUSHER_FCLK): ICE-Crusher clocking operates on the APB interface, using the ARM core clocking. This clock is half the frequency of the ARM clock (ARM_FCLK).

I2Async Clock (I2ASYNC_FCLK): This clock is half the frequency of the ARM clock (ARM_FCLK). It matches the OCP interface of the AXI2OCP bridge.

NOTE: The second half of the asynchronous bridge (T2ASYNC) is clocked directly by the PRCM module with the core clock. T2ASYNC is not part of the MPU subsystem.

Emulation Clocking: Emulation clocks are distributed by the PRCM module and are asynchronous to the ARM core clock (ARM_FCLK) and can run at a maximum of 1/3 the ARM core clock.

Table 1-1 and Table 1-2 summarize the clocks generated in the MPU subsystem by the MPU clock generator.

Figure 1-3. MPU Subsystem Clocking Scheme

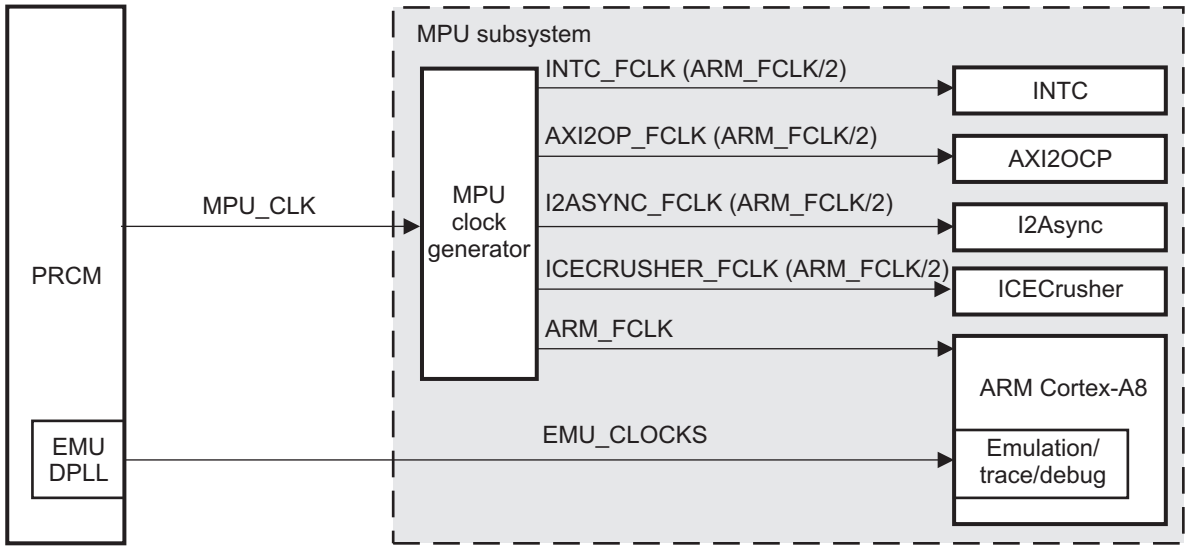


Table 1-1. MPU Subsystem Clock Frequencies

Clock Signal	Frequency	Comments
Cortex-A8 Core Functional Clock	MPUCLK	SYSCLK2 from PRCM
AXI2OCP Bridge Functional Clock	MPUCLK / 2	
IceCrusher Clock	MPUCLK / 2	
I2Async Bridge Functional Clock	MPUCLK / 2	

NOTE: The second half of the asynchronous bridges (T2ASYNC) going to EMIF and L3 are clocked directly by the PRCM module with the core clock for port. T2ASYNC half-bridges are not part of the MPU subsystem.

Table 1-2. MPU Subsystem Clock Input Signals

Clock Signal	Reference/Source Clock	Comments
MPU Clock	Non-Gated SYSCLK2 from PRCM	
Async bridge clock (core domain)	Gated SYSCLK6 from PRCM	T2ASYNC bridges only

1.2.4.2 Reset Distribution

Resets to the MPU subsystem are provided by the PRCM module and controlled by the clock generator module.

For details about clocks, resets, and power domains, see the *Power, Reset, and Clock Management (PRCM) Module* chapter.

Figure 1-4. Reset Scheme of the MPU Subsystem

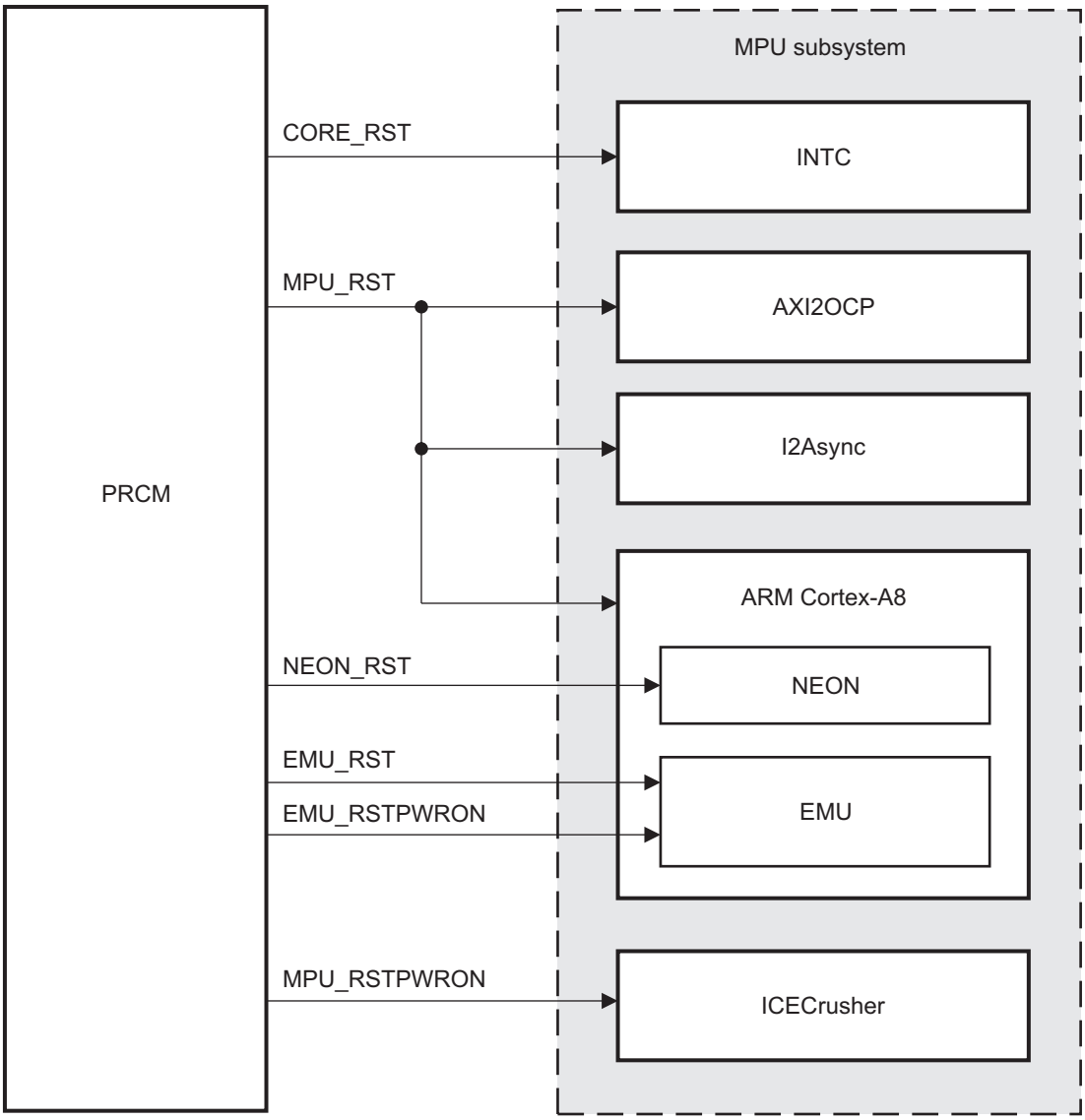


Table 1-3. Reset Scheme of the MPU Subsystem

Signal Name	I/O	Interface
MPU_RST	I	PRCM
NEON_RST	I	PRCM
CORE_RST	I	PRCM
MPU_RSTPWRON	I	PRCM
EMU_RST	I	PRCM
EMU_RSTPWRON	I	PRCM

1.2.5 ARM Subchip

1.2.5.1 ARM Overview

The ARM Cortex-A8 processor incorporates the technologies available in the ARM® ARM7™ processor architecture. These technologies include NEON for media and signal processing and ARM® Jazelle® Runtime Compilation Target (RCT) for acceleration of realtime compilers, ARM® Thumb®-2 technology for code density and the VFPv3 floating point architecture. For more information about the ARM Cortex-A8 processor, refer to the ARM® Cortex®-A8 Technical Reference Manual.

1.2.5.2 ARM Description

1.2.5.2.1 ARM Cortex-A8 Instruction, Data, and Private Peripheral Port

The AXI bus interface is the main interface to the ARM system bus. It performs L2 cache fills and non-cacheable accesses for both instructions and data. The AXI interface supports 128bit and 64-bit wide input and output data buses. It supports multiple outstanding requests on the AXI bus and a wide range of bus clock to core clock ratios. The bus clock is synchronous with the core clock.

For a complete programming model of the transaction rules (ordering, posting, and pipeline synchronization) that are applied depending on the memory region attribute associated with the transaction destination address, refer to the ARM® Cortex®-A8 Technical Reference Manual.

1.2.5.2.2 ARM Core Supported Features

Table 1-4 provides a list of main functions of the Cortex-A8 core supported inside the MPU Subsystem. For more information, refer to the ARM® Cortex®-A8 Technical Reference Manual.

Table 1-4. ARM Core Supported Features

Features	Comments
ARM version 7 ISA	Standard ARM instruction set + Thumb2, JazelleX Java accelerator, and Media extensions.
	Backward compatible with previous ARM ISA versions.
Cortex-A8 version	R3P2
L1 Icache and Dcache	32 KB , 4-way, 16-word line, 128-bit interface.
L2 Cache	256 KB, 8-way, 16-word line, 128-bit interface to L1. The L2 cache and cache controller are embedded within the Monza core.
	L2 valid bits cleared by software loop or by hardware.
TLB	Fully associative and separate ITLB with 32 entries and DTLB with 32 entries.
CoreSight ETM	The CoreSight ETM is embedded with the Monza core. The 32KB buffer (ETB) exists at the Chip Level (DebugSS)
Branch Target Address Cache	512 entries
Enhanced Memory Management Unit	Mapping sizes are 4KB, 64KB, 1MB, and 16MB. (Monza MMU adds extended physical address ranges)
NEON	Gives greatly enhanced throughput for media workloads and VFP-Lite support.
Flat Memories	176 Kbytes of ROM
	64 Kbytes of RAM
Buses	128-bit AXI internal bus from Cortex-A8 routed by an AXI2OCP bridge to the interrupt controller, ROM, RAM, and 3 asynchronous OCP bridges (128 bits and 64 bits)
Low interrupt latency	Closely coupled INTC to the Monza core with 128 interrupt lines
Vectored Interrupt Controller Port	Present.
JTAG based debug	Supported via DAP
Trace support	Supported via TPIU
External Coprocessor	Not supported

1.2.6 AXI2OCP and I2Async Bridges

1.2.6.1 Bridges Overview

The AXI2OCP Bridge is used to connect the AXI bus on the ARM A8 to the OCP native L3 interconnect (64-bit width), EMIF OCP port (128-bit width), and interrupt controller. It converts between AXI and OCP protocols and maintains a mapping of AXI tags to the OCP Tag ID. A memory region must be reserved for the interrupt handler. The bridge is required to do some minimal address decoding to decide where to forward the requests.

The AXI2OCP Bridge and the target modules (EMIF, L3) operate in different clock domains. The interface between the AXI2OCP Bridge and EMIF/L3 must go through an asynchronous bridge to properly synchronize signals to the opposite clock domain.

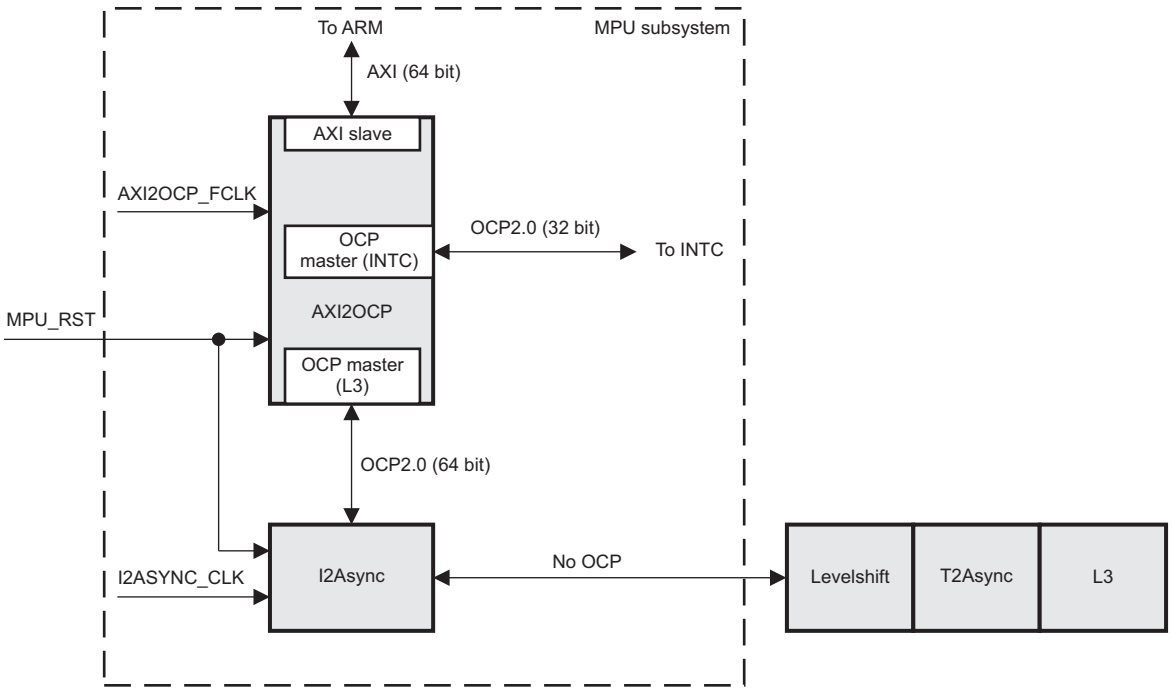
Bridging to the L3 is accomplished through an asynchronous interface involving the I2Async and T2Async modules. The I2Async module inside the MPU subsystem has an OCP port that is asynchronously transferred to the T2Async module and routed to the L3. T2Async is outside the MPU subsystem.

NOTE: The interface between I2Async and T2Async is not an OCP protocol.

1.2.6.2 Key Features

- Connects to the EMIF via a 128-bit OCP port and asynchronous bridge.
- Connects to the L3 interconnect via a 64-bit OCP port and asynchronous bridge.
- Connects to the interrupt controller via a 32-bit OCP port. (Only single transactions are supported)
- Supports Single-Request-Multiple-Data (data handshaking) burst mode to pipeline requests.
- Supports multiple outstanding requests.
- Emulation and boot-mode translation support.
- Exclusive accesses are translated to non-exclusive read/write in the bridge.

Figure 1-5. Overview of the AXI2OCP and the L3 Bridges



1.2.6.3 AXI to OCP Tag Mapping

Table 1-5 and Table 1-6 gives the Read and Write Channel AXI ID to OCP Tag Mappings

Table 1-5. Read Channel AXI ID to OCP Tag Mappings

AXI ID	Request type	OCP Thread[2]	OCP Tag	Outstanding Requests
4'b0000	NC/SO[1] data load	Thread_Mx	5'b00000	9 (one integer and up to 8 NEON)
4'b0001	Device data load	Thread_Mx	5'b00001	
4'b0011	peripheral	n/a	n/a	
4'b1110	Cacheable linefill into L1D	Thread_Mx	5'b01110	
4'b0100	NC/SO[1] instruction fetch	Thread_Mx	5'b00100	1
4'b0101	Device instruction fetch	Thread_Mx	5'b00101	
4'b1111	Linefills into L1I$ (line is L2 non-cacheable)	Thread_Mx	5'b01111	
4'b0110	NC/SO[1] Table-walk requests (instruction, data or PLE)	Thread_Mx	5'b00110	1
4'b1000	Cacheable linefills (I, D, TLB, PLE) #1, except linefill into L1D	Thread_Mx	5'b01000	1
4'b1001	Cacheable linefills (I, D, TLB, PLE) #2, except linefill into L1D	Thread_Mx	5'b01001	1
4'b1010	Cacheable linefills (I, D, TLB, PLE) #3, except linefill into L1D	Thread_Mx	5'b01010	1
4'b1011	Cacheable linefills (I, D, TLB, PLE) #4, except linefill into L1D	Thread_Mx	5'b01011	1
4'b0010	Reserved	n/a	n/a	n/a
4'b0111				
4'b1100				
4'b1101				

Table 1-6. Write Channel AXI ID to OCP Tag Mappings

AXI ID	Request type	OCP Thread[2]	OCP Tag	Outstanding Requests
4'b0000	NC/SO[1], or WT stores	Thread_Mx	5'b10000	Up to 8
4'b0001	Device writes	Thread_Mx	5'b10001	
4'b0011	Peripheral write	n/a	n/a	
4'b1000	Eviction #1 (including PLE)	Thread_Mx	5'b11000	1
4'b1001	Eviction #2 (including PLE)	Thread_Mx	5'b11001	1
4'b1010	Eviction #3 (including PLE)	Thread_Mx	5'b11010	1
4'b1011	Eviction #4 (including PLE)	Thread_Mx	5'b11011	1
4'b0010	Reserved	n/a	n/a	n/a
4'b0100				
4'b0101				
4'b0110				
4'b0111				
4'b1100				
4'b1101				
4'b1110				
4'b1111				

1.2.7 Interrupt Controller

The Host ARM Interrupt Controller (AINTC) is responsible for prioritizing all service requests from the system peripherals and generating either nIRQ or nFIQ to the host. The type of the interrupt (nIRQ or nFIQ) and the priority of the interrupt inputs are programmable. The AINTC interfaces to the Monza processor via the AXI port through an AXI2OCP bridge and runs at half the processor speed. The AINTC handles the interrupts directed to the HASS. It has the capability to handle up to 128 requests which can be steered/prioritized as A8 nFIQ or nIRQ interrupt requests.

The general features of the AINTC are:

- Up to 128 level sensitive interrupts inputs
- Individual priority for each interrupt input
- Each interrupt can be steered to nFIQ or nIRQ
- Independent priority sorting for nFIQ and nIRQ

1.2.8 Power Management

1.2.8.1 Power Domains

The MPU subsystem is divided into 5 power domains controlled by the PRCM module, as shown in Figure 1-6.

NOTE: The emulation domain and the core domain are not fully embedded in MPU subsystem.

Power management requirements at the device level govern power domains for the MPU subsystem. The device-level power domains are directly aligned with voltage domains and thus can be represented as a cross reference to the different voltage domains.

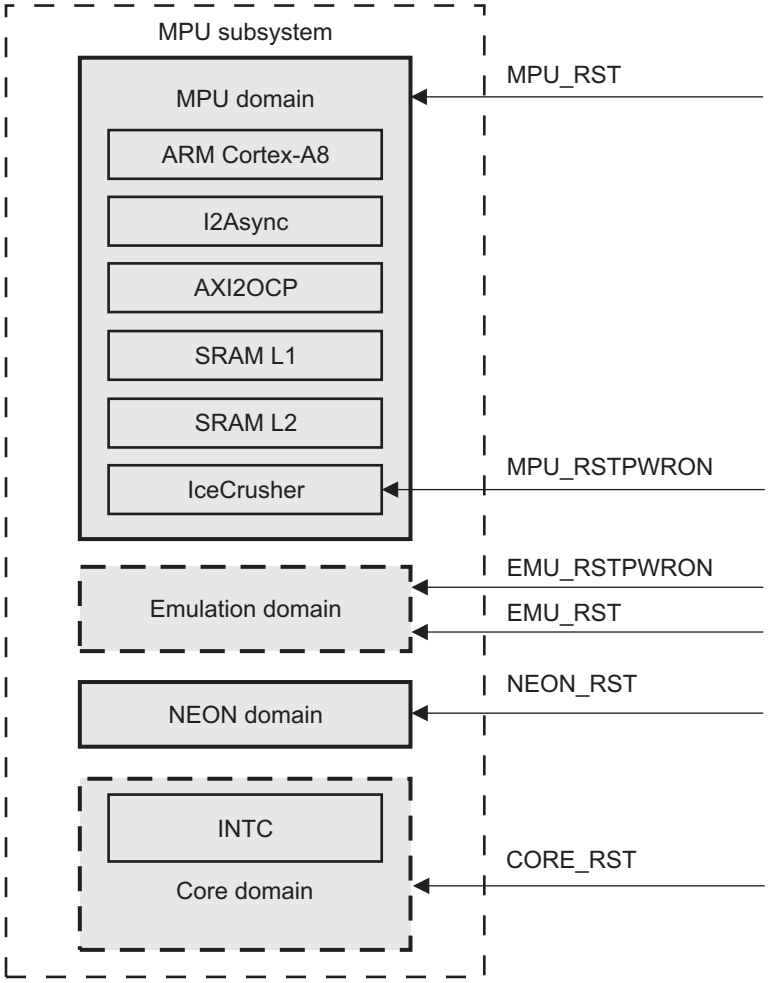
Table 1-7 shows the different power domains of the MPU subsystem and the modules inside.

Table 1-7. Overview of the MPU Subsystem Power Domain

Functional Power Domain	Physical Power Domain per System/Module
MPU subsystem domain	ARM, AXI2OCP, I2Asynch Bridge, ARM L1 and L2 periphery logic and array, ICE-Crusher, ETM, APB modules
MPU NEON domain	ARM NEON accelerator
CORE domain	MPU interrupt controller
EMU domain	EMU (ETB,DAP)

NOTE: L1 and L2 array memories have separate control signals into the in MPU Subsystem, thus directly controlled by the PRCM module. For details on the physical power domains and the voltage domains, see the *Power, Reset, and Clock Management (PRCM) Module* chapter.

Figure 1-6. MPU Subsystem Power Domain Overview



1.2.8.2 Power States

Each power domain can be driven by the PRCM module in 4 different states, depending on the functional mode required.

For each power domain, the PRCM module manages all transitions by controlling domain clocks, domain resets, domain logic power switches and memory power switches.

Table 1-8. MPU Power States

Power State	Logic Power	Memory Power	Clocks
Active	On	On or Off	On (at least one clock)
Inactive	On	On or Off	Off
Off	Off	Off	Off (all clocks)

1.2.8.3 Power Modes

The major part of the MPU subsystem belongs to the MPU power domain. The modules inside this power domain can be off at a time when the ARM processor is in an OFF or standby mode. IDLE/WAKEUP control is managed by the clock generator block but initiated by the PRCM module.

The MPU Standby status can be checked with PRCM.CM_IDLEST_MPU[0] ST_MPU bit. For the MPU to be on, the core (referred here as the device core) power must be on. Device power management does not allow INTC to go to OFF state when MPU domain is on (active or one of retention modes).

The NEON core has independent power off mode when not in use. Enabling and disabling of NEON can be controlled by software.

CAUTION

The MPU L1 cache memory does not support retention mode, and its array switch is controlled together with the MPU logic. For compliance, the L1 retention control signals exist at the PRCM module boundary, but are not used. The ARM L2 can be put into retention independently of the other domains.

Table 1-9 outlines the supported operational power modes. All other combinations are illegal. The ARM L2, NEON, and ETM/Debug can be powered up/down independently. The APB/ATB ETM/Debug column refers to all three features: ARM emulation, trace, and debug.

The MPU subsystem must be in a power mode where the MPU power domain, NEON power domain, debug power domain, and INTC power domain are in standby, or off state.

Table 1-9. MPU Subsystem Operation Power Modes

Mode	MPU and ARM Core Logic	AMR L2 RAM	NEON INTC	Device Core and ETM	APB/ATB Debug
1	Active	Active	Active	Active	Disabled or enabled
2	Active	Active	OFF	Active	Disabled or enabled
3	Active	RET	Active	Active	Disabled or enabled
4	Active	RET	OFF	Active	Disabled or enabled
5	Active	OFF	Active	Active	Disabled or enabled
6	Active	OFF	OFF	Active	Disabled or enabled
7	OFF	RET	OFF	OFF	Disabled or enabled
8	Standby	Active	Standby	Active	Disabled or enabled
9	Standby	Active	OFF	Active	Disabled or enabled
10	Standby	RET	Standby	Active	Disabled or enabled
11	Standby	RET	OFF	Active	Disabled or enabled
12	Standby	OFF	Standby	Active	Disabled or enabled
13	Standby	OFF	OFF	Active	Disabled or enabled
14	OFF	OFF	OFF	OFF	Disabled or enabled

1.2.9 Host ARM Address Map

Table 1-10. Address Map of the MPU Subsystem

Region	Address Range	Size
Internal Memory (Access not routed to external OCP ports)		
Reserved	0x4000_0000 – 0x4001_FFFF	1MB
ROM (48 KB)	0x4002_0000 – 0x4002_BFFF	
Reserved	0x4002_C000 – 0x400F_FFFF	
Reserved	0x4020_0000 – 0x402E_FFFF	1MB
SRAM (64 KB)	0x402F_0000 – 0x402F_FFFF	
Internal Reserved (Audio Back-End port not implemented on device HASS)		
Reserved	0x4010_0000 – 0x401F_FFFF	1MB
Private Peripheral Map (Access not routed to external OCP ports)		
ARM Interrupt Controller (AINTC)	0x4820_0000 – 0x4820_0FFF	4KB
Reserved	0x4820_1000 – 0x4827_FFFF	508KB
Reserved	0x4828_1000 – 0x482F_FFFF	508KB
128-bit OCP Master Port 0 (to EMIFs via DMM)		
EMIF0 / EMIF1 CS0	0x8000_0000 – 0xBFFF_FFFF	1GB
Reserved (EMIF0 / EMIF1 CS1)	0xC000_0000 – 0xFFFF_FFFF	1GB
64-bit OCP Master Port 1 (to L3)		
Boot Space [1]	0x0000_0000 – 0x00FF_FFFF	1MB
L3	0x0000_0000 – 0x5FFF_FFFF	(1.5GB – 1MB)
Tiler	0x6000_0000 – 0x7FFF_FFFF	256MB

1.2.10 ARM Programming Model

For detailed descriptions of registers used for MPU configuration, see the *Power, Reset, and Clock Management (PRCM) Module* and the *ARM Interrupt Controller (AINTC)* chapters.

1.2.10.1 Clock Control

For clock configuration settings, see the *Power, Reset, and Clock Management (PRCM) Module* chapter

1.2.10.2 MPU Power Mode Transitions

The following subsections describe transitions of different power modes for MPU power domain:

- Basic power on reset
- MPU into standby mode
- MPU out of standby mode
- MPU power on from a powered off state

1.2.10.2.1 Basic Power-On Reset

The power on reset follows the following sequence of operation and is applicable to initial power-up and wakeup from device off mode.

1. Reset the INTC (CORE_RST) and the MPU subsystem modules (MPU_RST). The clocks must be active during the MPU reset and CORE reset.

1.2.10.2.2 MPU Into Standby Mode

The MPU into standby mode follows the following sequence of operation and is applicable to initial power-up and wakeup from device Off mode.

1. The ARM core initiates entering into standby via software only (CP15 - WFI).
2. MPU modules requested internally of MPU subsystem to enter idle, after ARM core standby detected.
3. MPU is in standby output asserted for the PRCM module (all outputs guaranteed to be at reset values).
4. PRCM module can now request INTC to enter into idle mode. Acknowledge from INTC goes to the PRCM module .

NOTE: The INTC SWAKEUP output is a pure hardware signal to PRCM module for the status of its IDLE request and IDLE acknowledge handshake.

NOTE: In debug mode, ICE-Crusher could prevent MPU subsystem from entering into IDLE mode.

1.2.10.2.3 MPU Out Of Standby Mode

The MPU out of standby mode follows the following sequence of operation and is applicable to initial power-up and wakeup from device Off mode.

1. PRCM module must start clocks through DPLL programming.
2. Detect active clocking via status output of DPLL.
3. Initiate an interrupt through the INTC to wake up the ARM core from STANDBYWFI mode.

1.2.10.2.4 MPU Power-On From a Powered-Off State

1. MPU Power On, NEON Power On, Core Power On (INTC) should follow the ordered sequence per power switch daisy chain to minimize the peaking of current during power-up.

   NOTE: The core domain must be on, and reset, before the MPU can be reset.

2. Follow the reset sequence as described in the Basic Power-On Reset section.

1.2.10.3 NEON Power Mode Transition

When NEON power domain transition is configured to Automatic Hardware-supervised mode (CM_CLKSTCTRL_NEON[1:0] CLKTRCTRL_NEON bits are set to 0x3), it can not transition into idle unless MPU goes into Standby, because of the Hardware Sleep dependency between NEON and the MPU domain.

In that case, the MPU domain must also be configured in Automatic Hardware Supervised mode (CM_CLKSTCTRL_MPU[1:0] CLKTRCTRL_MPU bits must be set to 0x3) for the NEON power domain transition to happen.

For a complete programming model and more information about the ARM Cortex-A8 processor, refer to the ARM® Cortex®-A8 Technical Reference Manual.

1.3 C674x DSP Subsystem

1.3.1 Introduction

The DSP subsystem (Figure 1-7) includes TI's standard TMS320C674x megamodule and several blocks of internal memory (L1P, L1D, and L2). The DSP subsystem provides one slave port and one master port, which are connected to the L3 interconnect. It also provides three master ports for direct access to the HDVICP2 subsystem (the HDVICP2 and HDVICP2 SL2 ports). This section provides an overview of the DSP subsystem and the following considerations associated with it:

- Memory mapping
- Interrupts
- Power management

The internal architecture is an assembly of the following components:

- High-performance DSP derivative integrated in a megamodule, including local level 1 (L1) and level 2 (L2) cache and memory controllers for audio processing and general-purpose imaging and video processing
- L1 and L2 shared cache
- Dedicated enhanced data memory access (EDMA) engine to download/upload data from/to memories and peripherals external to the subchip
- Dedicated memory management unit (MMU) for accessing level 3 (L3) interconnect address space
- Local interconnect network
- Dedicated SYSC and wake-up generator (WUGEN) modules responsible for power management, clock generation, and connection to the power, reset, and clock management (PRCM) module

For more information, see the *TMS320C674x DSP Megamodule Reference Guide* (SPRUFK5), the *TMS320C674x DSP CPU and Instruction Set Reference Guide* (SPRUFE8), and the *TMS320C674x DSP Cache User's Guide* (SPRUG82).

1.3.2 C674x DSP Features and Options

The C6000 devices execute up to eight 32-bit instructions per cycle. The C674x CPU consists of 64 general-purpose 32-bit registers and eight functional units. These eight functional units contain:

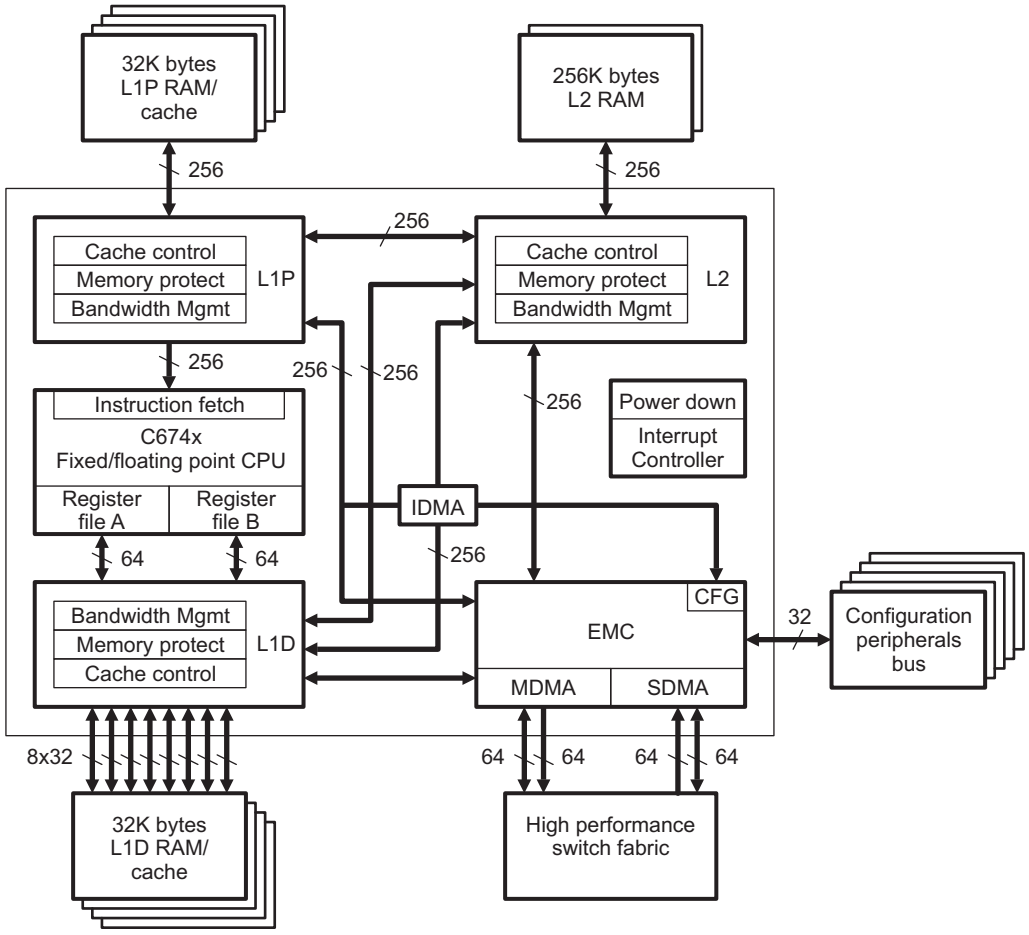
- Two multipliers
- Six ALUs

The C6000 generation has a complete set of optimized development tools, including an efficient C compiler, an assembly optimizer for simplified assembly-language programming and scheduling, and a Windows® operating system-based debugger interface for visibility into source code execution characteristics. A hardware emulation board, compatible with the TI XDS510™ and XDS560™ emulator interface, is also available. This tool complies with IEEE Standard 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture.

Features of the C6000 devices include:

- Advanced VLIW CPU with eight functional units, including two multipliers and six arithmetic units
  - Executes up to eight instructions per cycle for up to ten times the performance of typical DSPs
  - Allows designers to develop highly effective RISC-like code for fast development time
- Instruction packing
  - Gives code size equivalence for eight instructions executed serially or in parallel
  - Reduces code size, program fetches, and power consumption

Figure 1-7. TMS320C674x Megamodule Block Diagram



- Conditional execution of most instructions
  - Reduces costly branching
  - Increases parallelism for higher sustained performance
- Efficient code execution on independent functional units
  - Industry's most efficient C compiler on DSP benchmark suite
  - Industry's first assembly optimizer for fast development and improved parallelization
- 8/16/32-bit data support, providing efficient memory support for a variety of applications
- 40-bit arithmetic options add extra precision for vocoders and other computationally intensive applications
- Saturation and normalization provide support for key arithmetic operations
- Field manipulation and instruction extract, set, clear, and bit counting support common operation found in control and data manipulation applications.

The C674x devices include these additional features:

- Each multiplier can perform two 16 × 16-bit or four 8 × 8 bit multiplies every clock cycle.
- Quad 8-bit and dual 16-bit instruction set extensions with data flow support
- Support for non-aligned 32-bit (word) and 64-bit (double word) memory accesses
- Special communication-specific instructions have been added to address common operations in error-correcting codes.
- Bit count and rotate hardware extends support for bit-level algorithms.
- Compact instructions: Common instructions (AND, ADD, LD, MPY) have 16-bit versions to reduce code size.
- Protected mode operation: A two-level system of privileged program execution to support higher capability operating systems and system features such as memory protection.
- Exceptions support for error detection and program redirection to provide robust code execution
- Hardware support for modulo loop operation to reduce code size
- Each multiplier can perform 32 × 32 bit multiplies
- Additional instructions to support complex multiplies allowing up to eight 16-bit multiply/add/subtracts per clock cycle
- SPLOOP; hardware buffer for implementing hardware controlled pipelining. Resulting in smaller code size and interruptible tight loop for improved determinism.

The C674x devices are enhanced for code size improvement and floating-point performance. These additional features include:

- Hardware support for single-precision (32-bit) and double-precision (64-bit) IEEE floating-point operations.
- Execute packets can span fetch packets.
- Register file size is increased to 64 registers (32 in each datapath).
- Floating-point addition and subtraction capability in the .S unit.
- Mixed-precision multiply instructions.
- 32 × 32-bit integer multiply with 32-bit or 64-bit result.

The VelociTI architecture of the C6000 platform of devices make them the first off-the-shelf DSPs to use advanced VLIW to achieve high performance through increased instruction-level parallelism. A traditional VLIW architecture consists of multiple execution units running in parallel, performing multiple instructions during a single clock cycle. Parallelism is the key to extremely high performance, taking these DSPs well beyond the performance capabilities of traditional superscalar designs. VelociTI is a highly deterministic architecture, having few restrictions on how or when instructions are fetched, executed, or stored. It is this architectural flexibility that is key to the breakthrough efficiency levels of the TMS320C6000 Optimizing compiler. VelociTI's advanced features include:

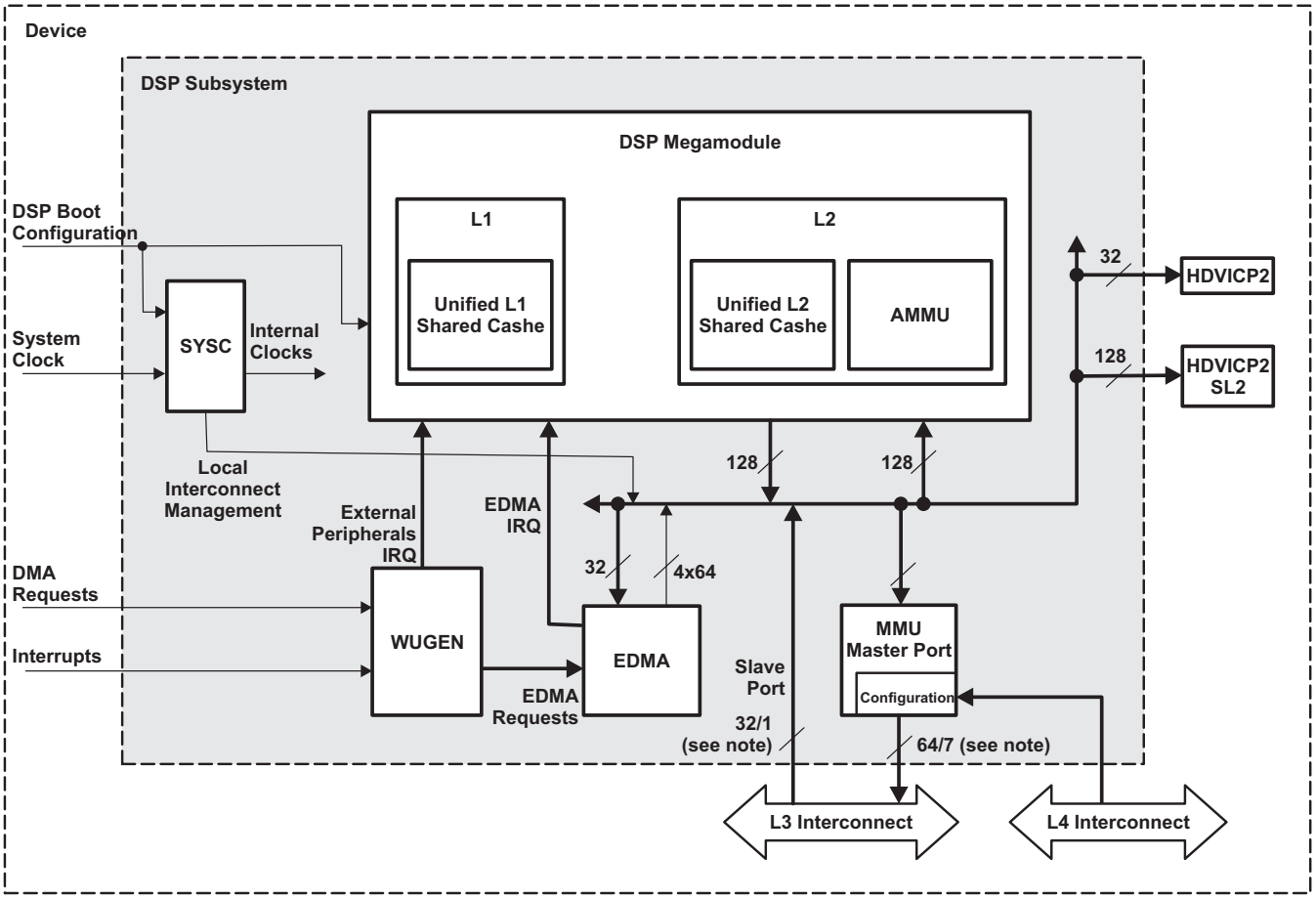
- Instruction packing: reduced code size
- All instructions can operate conditionally: flexibility of code
- Variable-width instructions: flexibility of data types
- Fully pipelined branches: zero-overhead branching.

1.3.3 DSP Subsystem Functional Description

The DSP subsystem is composed of a DSP megamodule coupled with several submodules that enable its integration in the device architecture. The DSP subsystem provides one slave port and one master port, which are connected to the L3 interconnect. It also provides three master ports for direct access to the HDVICP2 subsystem (the HDVICP2 SL2 ports).

Figure 1-8 shows the block diagram of the DSP subsystem.

Figure 1-8. DSP Subsystem Block Diagram



1.3.4 TMS320C674x Megamodule

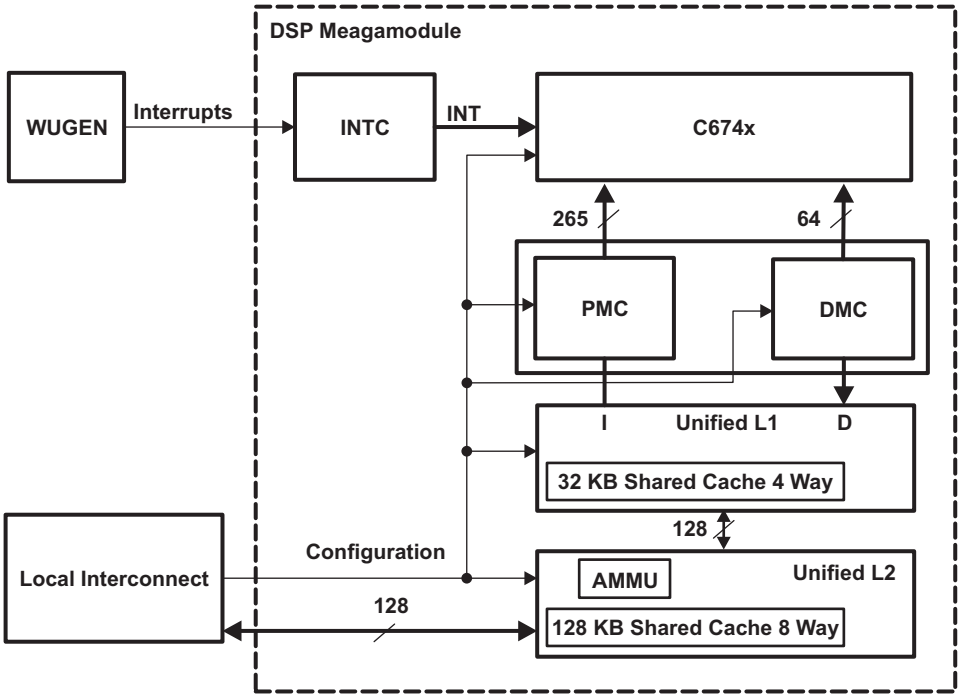
The C674x megamodule (Figure 1-7) consists of the following components:

- TMS320C674x CPU
- Internal memory controllers:
  - L1 program memory controller (PMC)
  - L1 data memory controller (DMC)
  - L1/L2 unified caches
- Internal peripherals:
  - Internal direct memory access (IDMA) controller
  - Interrupt controller (INTC)
  - Power-down controller (PDC)
  - Bandwidth manager (BWM)

- Advanced event triggering (AET)

Figure 1-9 shows a block diagram of the DSP megamodule.

Figure 1-9. TMS320C674x Megamodule Block Diagram



1.3.4.1 L1 Program Memory Controller (PMC)

The PMC delivers program fetch packets when they are requested by the DSP core. The PMC supports the following features:

- One 256-bit fetch packet per cycle (sustainable)
- DMA transfer from/to L1/L2 caches
- Fair priority-based arbitration between DSP, DMA, and cache controller for access to the SRAM
- Block and global program-initiated cache coherence support (invalidate)
- Freeze mode

1.3.4.2 L1 Data Memory Controller (DMC)

The DMC reads or writes data from/to local memories, as requested by the DSP. The DMC supports the following features:

- One 64-bit memory access per cycle (sustainable)
- A memory access pair can be any combination of read and write, with no incurred penalty
- DMA transfer from/to L1/L2 caches
- Fair priority-based arbitration between the DSP, DMA, and cache controller for access to the SRAM
- Hardware coherence maintenance with L2 (snooping)
- Block and global program-initiated cache coherence support (write-back, invalidate, and write-back-invalidate)
- Freeze and bypass mode

1.3.4.3 Unified L1/L2 Cache

The following features are supported by the shared cache design:

- 32-KB L1 unified cache implemented with high-performance bit cell to run at full speed
- 128-KB L2 unified cache implemented with high-density bit cell to optimize area at half speed
- L1 cache is 32B line size, 4-way set-associative and organized as 16-bank
- L2 cache is 32B line size and 8-way set associative
- L1 cache supports snooping
- L2 cache slave port to support snooping from EDMA accesses
- Fully pipelined/supports critical-word-first
- Write-combining
- Fill and eviction buffers, and hit in fill buffer
- Dynamic sizing, I/D allocation
- Prefetch 4 lines/preload N lines
- Background preload/clean

1.3.4.4 Internal DMA (IDMA) Controller

The IDMA controller performs fast block transfers between any two memory locations local to the C674x megamodule. Local memory locations are defined as those in Level 1 program (L1P), Level 1 data (L1D), and Level 2 (L2) memories, or in the external peripheral configuration (CFG) memory. The IDMA cannot transfer data to or from the internal MMR space. The IDMA is fully described in the *TMS320C674x DSP Megamodule Reference Guide* (SPRUFK5).

1.3.4.5 Attribute MMU

The attribute MMU (AMMU) for the shared cache provides the multi-access cache with a region-based address translation, read/write control, access type control, endiannism, and multilevel cache maintenance. The MMU is directly connected to the shared cache, and there is a dedicated interface for pipeline write policy management. L1 policies are queried at the allocation control of the cache, and L2 policies are propagated to the master interface. For flexibility, the MMU can be dedicated to an L1 or an L2 cache configuration, with or without address translation, or as an L1/L2 cache combination.

The AMMU supports different page sizes: large, medium, and small. The number of large pages, number of medium pages, and so on, is defined at design time. The maximum number of large pages is eight. This address space includes addresses for all eight regions. However, if the DSP subsystem defines only five large pages, only the first five addresses are valid.

1.3.4.6 Interrupt Controller (INTC)

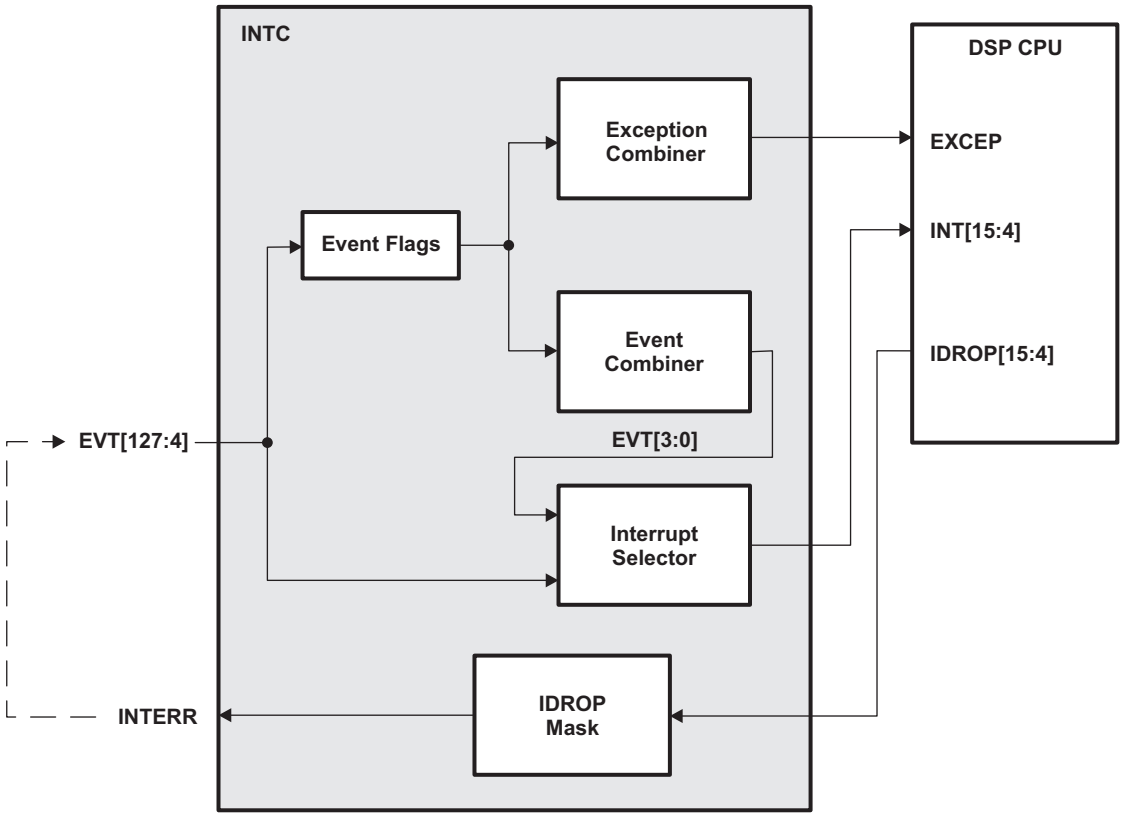
The DSP megamodule INTC detects, potentially combines, and routes up to 128 system events (internal and external) to the DSP CPU interrupt lines.

The DSP CPU has 12 maskable interrupts and one exception input. The INTC includes an interrupt selector, exception combiner, and event combiner. The interrupt selector allows the routing of any of the 128 system events (or a combination of them) to the 12 maskable interrupts of the DSP CPU, and software determines the priorities of those system events. To handle potential conflicts, the 12 CPU interrupts have fixed priorities. The exception combiner allows the combination of any of the 128 system events to the single exception input of the DSP CPU.

Figure 1-10 shows a block diagram of the DSP INTC megamodule.

For more information on the INTC, see the *TMS320C674x DSP Megamodule Reference Guide* (SPRUFK5).

Figure 1-10. DSP Megamodule INTC Block Diagram



1.3.4.6.1 NMI Interrupt

In addition to the 128 interrupts, the DSP also supports a special interrupt that behaves more like an exception, non-maskable interrupt (NMI). The NMI interrupt is controlled by two registers in the System Configuration Module, the chip signal register (CHIPSIG) and the chip signal clear register (CHIPSIG_CLR).

The NMI interrupt is asserted by writing a 1 to the CHIPSIG4 bit in CHIPSIG. The NMI interrupt is cleared by writing a 1 to the CHIPSIG4 bit in CHIPSIG_CLR.

1.3.4.7 Power-Down Controller (PDC)

The C674x megamodule includes a power-down controller (PDC). The PDC can power-down all of the following components of the C674x megamodule and internal memories of the DSP subsystem:

- C674x CPU
- Program memory controller (PMC)
- Data memory controller (DMC)
- Unified memory controller (UMC)
- Extended memory controller (EMC)
- Internal Direct Memory Access controller (IDMA)
- L1P memory
- L1D memory
- L2 memory

This device supports the static power-down feature from the C674x megamodule. The *TMS320C674x DSP Megamodule Reference Guide* (SPRUFK5) describes the power-down control in more detail.

- Static power-down: The PDC initiates power-down (clock gating) of the entire C674x megamodule and all internal memories immediately upon command from software.

Static power-down (clock gating) affects all components of the C674x megamodule and all internal memories. Software can initiate static power-down by way of a register bit in the power-down controller command register (PDCCMD) of the PDC. For more information on the PDC, see the *TMS320C674x DSP Megamodule Reference Guide* (SPRUFK5).

1.3.4.8 Bandwidth Manager (BWM)

The bandwidth manager (BWM) provides a programmable interface for optimizing bandwidth among the requesters for resources, which include the following:

- EDMA-initiated DMA transfers (and resulting coherency operations)
- IDMA-initiated transfers (and resulting coherency operations)
- Programmable cache coherency operations
  - Block based coherency operations
  - Global coherency operations
- CPU direct-initiated transfers
  - Data access (load/store)
  - Program access

The resources include the following:

- L1P memory
- L1D memory
- L2 memory
- Resources outside of the C674x megamodule: external memory, on-chip peripherals, registers

Since any given requestor could potentially block a resource for extended periods of time, the bandwidth manager is implemented to assure fairness for all requesters.

The bandwidth manager implements a weighted-priority-driven bandwidth allocation. Each requestor (EDMA, IDMA, CPU, etc.) is assigned a priority level on a per-transfer basis. The programmable priority level has a single meaning throughout the system. There are a total of nine priority levels, where priority zero is the highest priority and priority eight is the lowest priority. When requests for a single resource contend, access is granted to the highest-priority requestor. When the contention occurs for multiple successive cycles, a contention counter assures that the lower-priority requestor gets access to the resource every 1 out of *n* arbitration cycles, where *n* is programmable. A priority level of -1 represents a transfer whose priority has been increased due to expiration of the contention counter or a transfer that is fixed as the highest-priority transfer to a given resource.

1.3.5 Advanced Event Triggering (AET)

The C674x megamodule supports advanced event triggering (AET). This capability can be used to debug complex problems as well as understand performance characteristics of user applications. AET provides the following capabilities:

- Hardware Program Breakpoints: specify addresses or address ranges that can generate events such as halting the processor or triggering the trace capture.
- Data Watchpoints: specify data variable addresses, address ranges, or data values that can generate events such as halting the processor or triggering the trace capture.
- Counters: count the occurrence of an event or cycles for performance monitoring.
- State Sequencing: allows combinations of hardware program breakpoints and data watchpoints to precisely generate events for complex sequences.

1.4 HD Video Coprocessor Subsystem

1.4.1 HDVICP2 Overview

The HDVICP2 is the image and video imaging hardware accelerator subsystem.

The HDVICP2 supports resolutions up to 1080 p/i with full performance of 60 fps (or 120 fields). The HDVICP2 subsystem supports the following codec standards natively; that is, all functions of standards are accelerated (without any intervention of the digital signal processor):

- H.264: BP/MP/HP Encode and Decode
- H.264: Fast Profile/RCDO Encode and Decode
- MPEG-4: SP/ASP Encode/Decode (No support for GMC)
- DivX 5.x & higher Encode/Decode (No lower version; for example, 3.11 and 4.x)
- H.263: Profile 0 and 3 for Decode, Profile 0 for Encode
- Sorenson Spark: V0 and V1 Decode (No encode support)
- MPEG-2 SP/MP Encode/Decode
- MPEG-1 Encode/Decode
- VC1/WMV9/RTV : SP/MP/AP Encode and Decode
- ON2 VP6/VP7 Decode
- RV 8/9/10 Decode
- AVS 1.0 Encode and Decode
- JPEG (also MJPEG) Baseline Encode/Decode
- H264 Annex H (MVC)

The HDVICP2 subsystem is composed of:

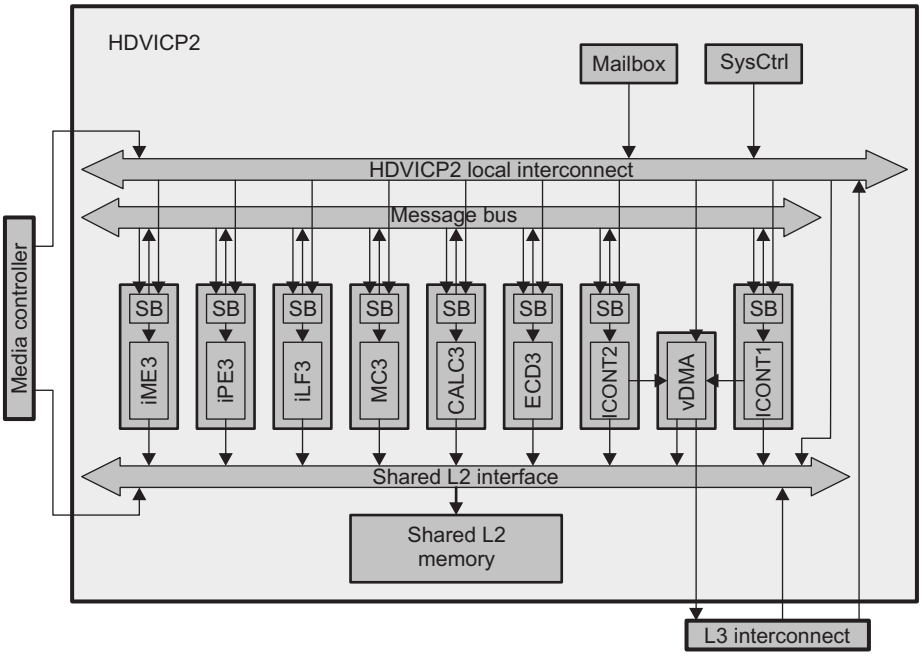
- A primary and a secondary sequencer: ICONT1 & ICONT2, which include its memories and an interrupt controller. Both these sequencers, ICONT1 and ICONT2, are identical.
- A video DMA engine: vDMA
- An entropy coder/decoder: ECD3
- A motion compensation engine: MC3
- A transform and quantization calculation engine: CALC3
- A loop filter acceleration engine: iLF3
- A motion estimation acceleration engine: iME3
- An intraprediction estimation engine: iPE3
- Shared level 2 (L2) interface and memory (of size 256 Kbytes)
- Local interconnect
- A message interface for communication between SyncBoxes
- Mailbox
- A debug module for trace event and software instrumentation: SMSET

TI has chosen eXpress DSP Digital Media (xDM) standard as the principle software interface to HDVICP2. The xDM standard defines application programming interfaces (APIs) through which an application invokes a particular class of codec, such as video, imaging, speech, and audio (aka VISA). The video and imaging classes for encode and decode apply to HDVICP2.

1.4.1.1 HDVICP2 Functional Description

Figure 1-11 shows the block diagram of the HDVICP2 subsystem.

Figure 1-11. HDVICP2 Block Diagram



1.4.1.2 SyncBox

The SyncBox is a configurable module that is in charge of scheduling all embedded hardware modules within the HDVICP2 subsystem. It handles all aspects of synchronization, data sharing, and parameters passing between accelerators. It also offers the possibilities to use asynchronous messages.

1.4.2 ICONTs

The ICONT module is an ARM968E-S based microcontroller with 32KB of instruction tightly coupled memory (TCM) and 16KB of data tightly coupled memory. It includes an interrupt controller (INTC), a local data mover, its own SyncBox module for synchronizing tasks with other modules and its associated SyncBox handler.

Two identical instances of the ICONT, ICONT1 and ICONT2, are present in the HDVICP2 subsystem. They can typically be used to perform high-level processing (at frame and slice level), control at macroblock-level bounding box computation and other vDMA processing tasks.

Software can map processing equally on either ICONT.

1.4.3 vDMA

The vDMA is a DMA engine to perform data transfer between external memories and shared L2 memory. The VDMA can also perform memory copies inside SL2 and inside external memory.

1.4.4 iME3

The iME3 accelerator is used to perform motion estimation for Video encoding. iME3 has its own embedded SyncBox module for synchronizing tasks with other modules.

The iME3 compares a current macroblock to a reference area and provides an area in the reference region (in terms of offsets) which is least different to the current macroblock. It can also interpolate a block with half or quarter pixel precision, thus producing 4 (half pixel) or 16 (quarter pixel) interpolated blocks from the original block. Additionally, the IME3 supports searching of the best matching block within the interpolated planes.

1.4.5 iPE3

The iPE3 accelerator is used to perform the intraprediction estimation for video encoding. The iPE3 has its own embedded SyncBox module for synchronizing tasks with other modules and an LSE to transfer data from internal memories to shared L2 memory.

The iPE3 supports two modes, depending on the video standard:

- Spatial intraprediction estimation for H264 and AVS. It creates intraprediction macroblocks with given intramodes from the original macroblock and provides a cost estimation between the original macroblock and each pseudo intraprediction macroblock, and then chooses a mode with the smallest cost to recommend it as an optimal intraprediction mode.
- Spatial activity for MPEG-1/2/4 and VC-1. It calculates the spatial activity of the original luminance samples with the specified block size. This mode is designed to provide information about the original Luma pixels. The values can be used to decide the coding parameters of the macroblock such as the coding mode and the quantization parameter.

1.4.6 MC3

The MC3 accelerator is used to perform the motion compensation. The MC3 has its own embedded SyncBox module for synchronizing tasks with other HWAs and an LSE to transfer data from internal memories to shared L2 memory.

It creates an interprediction macroblock with given motion vectors and modes from the reference data.

1.4.7 CALC3

The CALC3 accelerator is used to perform forward and inverse transform and quantization calculation. CALC3 has its own embedded SyncBox module for synchronizing tasks with other HWAs and an LSE to transfer data from internal memories to shared L2 memory.

It can perform transform/inverse transform, Q/iQ and DC/AC prediction.

1.4.8 iLF3

The iLF3 accelerator is used to perform deblocking filtering and boundary strength computation. The iLF3 has its own embedded SyncBox module for synchronizing tasks with other HWAs.

1.4.9 ECD3

The ECD3 accelerator is designed to encode and decode the stream. ECD3 has its own embedded SyncBox module for synchronizing tasks with other HWAs and an LSE to transfer data from internal memories to shared L2 memory.

It supports Huffman codes and arithmetic codes.

For encode, the ECD3 encodes the macroblock information and residual data into a bitstream. For decode, the ECD3 decodes the bitstream and recovers the macroblock information and residual data.

1.4.10 SL2 Interface

The shared L2 interface, SL2IF, is an arbitrator that allows 18 initiators to access to an interleaved set of eight memory banks.

The SL2IF has two sets of interfaces:

- Eighteen 128-bit interfaces for accesses from module's to shared L2 memories.
- Eight 128-bit memory interfaces for direct accesses to the memory banks.

1.4.11 Message Bus

The message bus is an arbitrator that allows eight initiators to access eight targets. It is used to dispatch messages generated by the SyncBox of the different IPs.

1.4.12 HDVICP2 Local Interconnect

The HDVICP2 local interconnect provides connectivity between two external host interconnects (Media Controller and L3), two local sequencers (ICONT1 and ICONT2) and video hardware accelerators (iME3, iLF3, ECD3, CALC3, MC3, iPE3), video DMA engine (vDMA), and local modules (mailbox and sysctrl).

1.4.13 MailBox

The function of the mailbox is to support a 2-way communication between two hosts through interrupts. It allows the software to establish a communication channel between processors through a set of registers and associated interrupt signals by sending and receiving messages.

The mailbox embedded inside the HDVICP2 subsystem implements a 2-way communication between three external users and one internal user. This communication is ensured through three pairs of mailboxes and four-message FIFO depth for each message queue.

NOTE: The internal user is one of the two ICONTs. ICONT1 and ICONT2 are connected on a shared interrupt line. The choice between ICONT1 and ICONT2 is done by masking mailbox interrupt on ICONT1 or ICONT2. ICONT1 and ICONT2 can access the mailbox through the HDVICP2 local interconnect.

1.4.14 HDVICP2 System Control

The SYSCTRL module of HDVICP2 is in charge of:

- Controlling clocks to the modules via software control and the hardware power handshaking state.
- Controlling power, reset, and clock management (PRCM) module power handshaking.
- Providing status of the above operation.
- Supporting synchronization through external event.

1.5 Memory Map Summary

The device has multiple on-chip memories associated with its processors and various subsystems. To help simplify software development a unified memory map is used where possible to maintain a consistent view of device resources across all bus masters.

1.5.1 L3 Memory Map

The L3 high-performance interconnect is based on a Network-on-Chip (NoC) interconnect infrastructure. The NoC uses an internal packet-based protocol for forward (read command, write command with data payload) and backward (read response with data payload, write response) transactions. All exposed interfaces of this NoC interconnect, both for targets and initiators, comply with the OCPIP2.2 reference standard.

Table 1-11 shows the general device level-3 (L3) memory map. The table represents the physical addresses used by the L3 infrastructure. Some processors within the device (such as Cortex-A8 ARM, C674x DSP) may re-map these targets to different virtual addresses through an internal or external MMU. Processors without MMUs and other bus masters use these physical addresses to access L3 regions. Note that not all masters have access to all L3 regions, but only those with defined connectivity. The L3 interconnect returns an address-hole error if any initiator attempts to access a target to which it has no connection.

Table 1-11. L3 Memory Map

Block Name	Start Address	End Address	Size	Description
GPMC	0000 0000h	1FFF FFFFh	512MB	GPMC
PCIe Gen2	2000 0000h	2FFF FFFFh	256MB	PCIe Gen2 Targets
Reserved	3000 0000h	3FFF FFFFh	256MB	Reserved
Reserved	4000 0000h	402F FFFFh	3MB	Reserved
L3 OCMC0	4030 0000h	4033 FFFFh	256KB	OCMC SRAM
Reserved	4034 0000h	403F FFFFh	768KB	Reserved (OCMC RAM0)
L3 OCMC1	4040 0000h	4043 FFFFh	256KB	OCMC SRAM
Reserved	4044 0000h	404F FFFFh	768KB	Reserved (OCMC RAM1)
Reserved	4050 0000h	407F FFFFh	3MB	Reserved
C674x	4080 0000h	4083 FFFFh	256KB	C674x UMAP0 (L2 RAM)
Reserved	4084 0000h	40DF FFFFh	5888KB	Reserved
C674x	40E0 0000h	40E0 7FFFh	32KB	C674x L1P Cache/RAM
Reserved	40E0 8000h	40EF FFFFh	992KB	Reserved
C674x	40F0 0000h	40F0 7FFFh	32KB	C674x L1D Cache/RAM
Reserved	40F0 8000h	40FF FFFFh	992KB	Reserved
Reserved	4100 0000h	41FF FFFFh	16MB	Reserved
Reserved	4200 0000h	43FF FFFFh	32MB	Reserved
L3 CFG Registers	4400 0000h	44BF FFFFh	12MB	L3 configuration registers
Reserved	44C0 0000h	45FF FFFFh	20MB	Reserved
McASP0	4600 0000h	463F FFFFh	4MB	McASP0
McASP1	4640 0000h	467F FFFFh	4MB	McASP1
McASP2	4680 0000h	46BF FFFFh	4MB	McASP2
HDMI	46C0 0000h	46FF FFFFh	4MB	HDMI wrapper and core registers
McBSP	4700 0000h	473F FFFFh	4MB	McBSP registers
USB2.0	4740 0000h	477F FFFFh	4MB	USB2.0 registers/CPPI
Reserved	4780 0000h	47BF FFFFh	4MB	Reserved
Reserved	47C0 0000h	47FF FFFFh	4MB	Reserved
L4 Standard Domain	4800 0000h	48FF FFFFh	16MB	L4 Standard Peripheral domain (see Table 1-12)

Table 1-11. L3 Memory Map (continued)

Block Name	Start Address	End Address	Size	Description
EDMA TPCC	4900 0000h	490F FFFFh	1MB	EDMA CC registers
Reserved	4910 0000h	497F FFFFh	7MB	Reserved
EDMA TPTC0	4980 0000h	498F FFFFh	1MB	EDMA TC0 registers
EDMA TPTC1	4990 0000h	499F FFFFh	1MB	EDMA TC1 registers
EDMA TPTC2	49A0 0000h	49AF FFFFh	1MB	EDMA TC2 registers
EDMA TPTC3	49B0 0000h	49BF FFFFh	1MB	EDMA TC3 registers
Reserved	49C0 0000h	49FF FFFFh	4MB	Reserved
L4 High-Speed Domain	4A00 0000h	4AFF FFFFh	16MB	L4 High-Speed Peripheral domain (see Table 1-13)
Instrumentation	4B00 0000h	4BFF FFFFh	16MB	EMU Subsystem region
DDR EMIF0 Registers(1)	4C00 0000h	4CFF FFFFh	16MB	Configuration registers
DDR EMIF1 Registers(1)	4D00 0000h	4DFF FFFFh	16MB	Configuration registers
DDR DMM Registers(1)	4E00 0000h	4FFF FFFFh	32MB	Configuration registers
GPMC Registers	5000 0000h	50FF FFFFh	16MB	Configuration registers
PCIe Registers	5100 0000h	51FF FFFFh	16MB	Configuration registers
Reserved	5200 0000h	52FF FFFFh	16MB	Reserved
HDVICP2-2 Configuration	5300 0000h	53FF FFFFh	16MB	HDVICP2-2 Host Port
HDVICP2-2 SL2	5400 0000h	54FF FFFFh	16MB	HDVICP2-2 SL2 Port
Reserved	5500 0000h	55FF FFFFh	16MB	Reserved
SGX530	5600 0000h	56FF FFFFh	16MB	SGX530 Slave Port
Reserved	5600 0000h	56FF FFFFh	16MB	Reserved
Reserved	5700 0000h	57FF FFFFh	16MB	Reserved
HDVICP2-0 Configuration	5800 0000h	58FF FFFFh	16MB	HDVICP2-0 Host Port
HDVICP2-0 SL2	5900 0000h	59FF FFFFh	16MB	HDVICP2-0 SL2 Port
HDVICP2-1 Configuration	5A00 0000h	5AFF FFFFh	16MB	HDVICP2-1 Host Port
HDVICP2-1 SL2	5B00 0000h	5BFF FFFFh	16MB	HDVICP2-1 SL2 Port
Reserved	5C00 0000h	5DFF FFFFh	32MB	Reserved
Reserved	5E00 0000h	5FFF FFFFh	32MB	Reserved
Tiler	6000 0000h	7FFF FFFFh	512MB	Virtual Tiled Address Space
DDR EMIF0/1 SDRAM (2)	8000 0000h	BFFF FFFFh	1GB	DDR
DDR EMIF0/1 SDRAM(2)	C000 0000h	FFFF FFFFh	1GB	DDR
DDR DMM	1 0000 0000h	1 FFFF FFFFh	4GB	DDR DMM Tiler Extended address map – Virtual Views (HDVPSS only)

(1) These accesses occur through the DDR DMM Tiler Ports. The DMM will split address ranges internally to address DDR EMIF and DDR DMM control registers.

(2) DDR EMIF0 and DDR EMIF1 addresses may be contiguous or bank interleaved depending on configuration of the DDR DMM; for more details, see the DDR DMM documentation.

1.5.2 L4 Memory Map

1.5.2.1 L4 Standard Peripheral

The L4 standard peripheral bus accesses standard peripherals and IP configuration registers. The memory map is shown in Table 1-12.

Table 1-12. L4 Standard Peripheral Memory Map

Device Name	Start Address	End Address	Size	Description
L4 Standard Configuration	4800 0000h	4800 07FFh	2KB	Address/Protection (AP)
	4800 0800h	4800 0FFFh	2KB	Link Agent (LA)
	4800 1000h	4800 13FFh	1KB	Initiator Port (IP0)
	4800 1400h	4800 17FFh	1KB	Initiator Port (IP1)
	4800 1800h	4800 1FFFh	2KB	Reserved (IP2–IP3)
Reserved	4800 2000h	4800 7FFFh	24KB	Reserved
e-Fuse Controller	4800 8000h	4800 8FFFh	4KB	Peripheral Registers
	4800 9000h	4800 9FFFh	4KB	Support Registers
Reserved	4800 A000h	4800 FFFFh	24KB	Reserved
System MMU	4801 0000h	4801 0FFFh	4KB	Peripheral Registers
	4801 1000h	4801 1FFFh	4KB	Support Registers
Reserved	4801 2000h	4801 FFFFh	56KB	Reserved
UART0	4802 0000h	4802 0FFFh	4KB	Peripheral Registers
	4802 1000h	4802 1FFFh	4KB	Support Registers
UART1	4802 2000h	4802 2FFFh	4KB	Peripheral Registers
	4802 3000h	4802 3FFFh	4KB	Support Registers
UART2	4802 4000h	4802 4FFFh	4KB	Peripheral Registers
	4802 5000h	4802 5FFFh	4KB	Support Registers
Reserved	4802 6000h	4802 7FFFh	8KB	Reserved
I2C0	4802 8000h	4802 8FFFh	4KB	Peripheral Registers
	4802 9000h	4802 9FFFh	4KB	Support Registers
I2C1	4802 A000h	4802 AFFFh	4KB	Peripheral Registers
	4802 B000h	4802 BFFFh	4KB	Support Registers
Reserved	4802 C000h	4802 DFFFh	8KB	Reserved
TIMER1	4802 E000h	4802 EFFFh	4KB	Peripheral Registers
	4802 F000h	4802 FFFFh	4KB	Support Registers
SPI	4803 0000h	4803 0FFFh	4KB	Peripheral Registers
	4803 1000h	4803 1FFFh	4KB	Support Registers
GPIO0	4803 2000h	4803 2FFFh	4KB	Peripheral Registers
	4803 3000h	4803 3FFFh	4KB	Support Registers
Reserved	4803 4000h	4803 7FFFh	16KB	Reserved
McASP0	4803 8000h	4803 9FFFh	8KB	Peripheral Registers
	4803 A000h	4803 AFFFh	4KB	Support Registers
Reserved	4803 B000h	4803 BFFFh	4KB	Reserved
McASP1	4803 C000h	4803 DFFFh	8KB	Peripheral Registers
	4803 E000h	4803 EFFFh	4KB	Support Registers
Reserved	4803 F000h	4803 FFFFh	4KB	Reserved
TIMER2	4804 0000h	4804 0FFFh	4KB	Peripheral Registers
	4804 1000h	4804 1FFFh	4KB	Support Registers
TIMER3	4804 2000h	4804 2FFFh	4KB	Peripheral Registers
	4804 3000h	4804 3FFFh	4KB	Support Registers

Table 1-12. L4 Standard Peripheral Memory Map (continued)

Device Name	Start Address	End Address	Size	Description
TIMER4	4804 4000h	4804 4FFFh	4KB	Peripheral Registers
	4804 5000h	4804 5FFFh	4KB	Support Registers
TIMER5	4804 6000h	4804 6FFFh	4KB	Peripheral Registers
	4804 7000h	4804 7FFFh	4KB	Support Registers
TIMER6	4804 8000h	4804 8FFFh	4KB	Peripheral Registers
	4804 9000h	4804 9FFFh	4KB	Support Registers
TIMER7	4804 A000h	4804 AFFFh	4KB	Peripheral Registers
	4804 B000h	4804 BFFFh	4KB	Support Registers
GPIO1	4804 C000h	4804 CFFFh	4KB	Peripheral Registers
	4804 D000h	4804 DFFFh	4KB	Support Registers
Reserved	4804 E000h	4804 FFFFh	8KB	Reserved
McASP2	4805 0000h	4805 1FFFh	8KB	Peripheral Registers
	4805 2000h	4805 2FFFh	4KB	Support Registers
Reserved	4805 3000h	4805 FFFFh	52KB	Reserved
SD/SDIO	4806 0000h	4806 FFFFh	64KB	Peripheral Registers
	4807 0000h	4807 0FFFh	4KB	Support Registers
Reserved	4807 1000h	4807 FFFFh	60KB	Reserved
ELM	4808 0000h	4808 FFFFh	64KB	Peripheral Registers
	4809 0000h	4809 0FFFh	4KB	Support Registers
Reserved	4809 1000h	480B FFFFh	188KB	Reserved
RTC	480C 0000h	480C 0FFFh	4KB	Peripheral Registers
	480C 1000h	480C 1FFFh	4KB	Support Registers
WDT1	480C 2000h	480C 2FFFh	4KB	Peripheral Registers
	480C 3000h	480C 3FFFh	4KB	Support Registers
Reserved	480C 4000h	480C 7FFFh	16KB	Reserved
Mailbox	480C 8000h	480C 8FFFh	4KB	Peripheral Registers
	480C 9000h	480C 9FFFh	4KB	Support Registers
Spinlock	480C A000h	480C AFFFh	4KB	Peripheral Registers
	480C B000h	480C BFFFh	4KB	Support Registers
Reserved	480C C000h	480F FFFFh	208KB	Reserved
HDVPSS	4810 0000h	4811 FFFFh	128KB	Peripheral Registers
	4812 0000h	4812 0FFFh	4KB	Support Registers
Reserved	4812 1000h	4812 1FFFh	4KB	Reserved
HDMI PHY	4812 2000h	4812 2FFFh	4KB	Peripheral Registers
	4812 3000h	4812 3FFFh	4KB	Support Registers
Reserved	4812 4000h	4813 FFFFh	112KB	Reserved
Control Module	4814 0000h	4815 FFFFh	128KB	Peripheral Registers
	4816 0000h	4816 0FFFh	4KB	Support Registers
Reserved	4816 1000h	4817 FFFFh	124KB	Reserved
PRCM	4818 0000h	4818 2FFFh	12KB	Peripheral Registers
	4818 3000h	4818 3FFFh	4KB	Support Registers
Reserved	4818 4000h	4818 7FFFh	16KB	Reserved
SmartReflex0	4818 8000h	4818 8FFFh	4KB	Peripheral Registers
	4818 9000h	4818 9FFFh	4KB	Support Registers
SmartReflex1	4818 A000h	4818 AFFFh	4KB	Peripheral Registers
	4818 B000h	4818 BFFFh	4KB	Support Registers

Table 1-12. L4 Standard Peripheral Memory Map (continued)

Device Name	Start Address	End Address	Size	Description
OCP Watchpoint	4818 C000h	4818 CFFFh	4KB	Peripheral Registers
	4818 D000h	4818 DFFFh	4KB	Support Registers
Reserved	4818 E000h	4818 EFFFh	4KB	Reserved
	4818 F000h	4818 FFFFh	4KB	Reserved
Reserved	4819 0000h	4819 0FFFh	4KB	Reserved
	4819 1000h	4819 1FFFh	4KB	Reserved
Reserved	4819 2000h	4819 2FFFh	4KB	Reserved
	4819 3000h	4819 3FFFh	4KB	Reserved
Reserved	4819 4000h	4819 4FFFh	4KB	Reserved
	4819 5000h	4819 5FFFh	4KB	Reserved
Reserved	4819 6000h	4819 6FFFh	4KB	Reserved
	4819 7000h	4819 7FFFh	4KB	Reserved
DDR0 PHY	4819 8000h	4819 8FFFh	4KB	Peripheral Registers
	4819 9000h	4819 9FFFh	4KB	Support Registers
DDR1 PHY	4819 A000h	4819 AFFFh	4KB	Peripheral Registers
	4819 B000h	4819 BFFFh	4KB	Support Registers
Reserved	4819 C000h	481F FFFFh	400KB	Reserved
Interrupt controller(1)	4820 0000h	4820 0FFFh	4KB	Cortex-A8 Accessible Only
Reserved(1)	4820 1000h	4823 FFFFh	252KB	Cortex-A8 Accessible Only
MPUSS Configuration register(1)	4824 0000h	4824 0FFFh	4KB	Cortex-A8 Accessible Only
Reserved(1)	4824 1000h	4827 FFFFh	252KB	Cortex-A8 Accessible Only
Reserved(1)	4828 1000h	482F FFFFh	508KB	Cortex-A8 Accessible Only
Reserved	4830 0000h	48FF FFFFh	13MB	Reserved

(1) These regions are decoded internally by the Cortex-A8 Subsystem and are not physically part of the L4 standard. They are included here only for reference when considering the Cortex-A8 memory-map. For masters other than the Cortex-A8, these regions are reserved.

1.5.2.2 L4 High-Speed Peripheral

The L4 high-speed peripheral bus accesses the IP configuration registers of high-speed peripherals in L3. The memory map is shown in Table 1-13.

Table 1-13. L4 High-Speed Peripheral Memory Map

Device Name	Start Address	End Address	Size	Description
L4 High-Speed configuration	4A00 0000h	4A00 07FFh	2KB	Address/Protection (AP)
	4A00 0800h	4A00 0FFFh	2KB	Link Agent (LA)
	4A00 1000h	4A00 13FFh	1KB	Initiator Port (IP0)
	4A00 1400h	4A00 17FFh	1KB	Initiator Port (IP1)
	4A00 1800h	4A00 1FFFh	2KB	Reserved (IP2–IP3)
Reserved	4A00 2000h	4A07 FFFFh	504KB	Reserved
Reserved	4A08_0000h	4A0A_0FFFh	132KB	Reserved
Reserved	4A0A 1000h	4A0F FFFFh	380KB	Reserved
EMAC0/MDIO	4A10 0000h	4A10 3FFFh	16KB	Peripheral Registers
	4A10 4000h	4A10 4FFFh	4KB	Support Registers
Reserved	4A10 5000h	4A11 FFFFh	108KB	Reserved
EMAC1	4A12 0000h	4A12 3FFFh	16KB	Peripheral Registers
	4A12 4000h	4A12 4FFFh	4KB	Support Registers
Reserved	4A12 5000h	4A13 FFFFh	108KB	Reserved
SATA	4A14 0000h	4A14 FFFFh	64KB	Peripheral Registers
	4A15 0000h	4A15 0FFFh	4KB	Support Registers
Reserved	4A15 1000h	4A17 FFFFh	188KB	Reserved
Reserved	4A18 0000h	4A19 FFFFh	128KB	Reserved
	4A1A 0000h	4A1A 0FFFh	4KB	Reserved
Reserved	4A1A 1000h	4AFF FFFFh	14716KB	Reserved

1.5.3 TILER Extended Addressing Map

The Tiling and Isometric Lightweight Engines for Rotation (TILER) ports are mainly used for optimized 2-D block accesses. The TILER also supports rotation of the image buffer at 0º, 90º, 180º, and 270º, with vertical and horizontal mirroring.

The top 4-GB address space is divided into eight sections of 512MB each. These eight sections correspond to the eight different orientations as shown in Table 1-14.

Table 1-14. TILER Extended Address Memory Map

Block Name	Start Address	End Address	Size	Description
Tiler View 0	1 0000 0000h	1 1FFF FFFFh	512MB	Natural 0° View
Tiler View 1	1 2000 0000h	1 3FFF FFFFh	512MB	0° with Vertical Mirror View
Tiler View 2	1 4000 0000h	1 5FFF FFFFh	512MB	0° with Horizontal Mirror View
Tiler View 3	1 6000 0000h	1 7FFF FFFFh	512MB	180° View
Tiler View 4	1 8000 0000h	1 9FFF FFFFh	512MB	90° with Vertical Mirror View
Tiler View 5	1 A000 0000h	1 BFFF FFFFh	512MB	270° View
Tiler View 6	1 C000 0000h	1 DFFF FFFFh	512MB	90° View
Tiler View 7	1 E000 0000h	1 FFFF FFFFh	512MB	90° with Horizontal Mirror View

1.5.4 Cortex-A8 Memory Map

The Cortex-A8 includes a memory management unit (MMU) to translate virtual addresses to physical addresses which are then decoded within the Host ARM Subsystem. The subsystem includes its own ROM and RAM, as well as configuration registers for its interrupt controller. These addresses are hard-coded within the subsystem. In addition, the upper 2GB of address space is routed to a special port (Master 0) intended for low-latency access to DDR memory. All other physical addresses are routed to the L3 port (Master 1) where they are decoded by the device infrastructure. The Cortex-A8 memory map is shown in Table 1-15.

Table 1-15. Cortex-A8 Memory Map

Region Name	Start Address	End Address	Size	Description
Boot Space	0000 0000h	000F FFFFh	1MB	Boot Space
L3 Target Space	0000 0000h	1FFF FFFFh	512MB	GPMC
	2000 0000h	2FFF FFFFh	256MB	PCIe Gen2 Targets
	3000 0000h	3FFF FFFFh	256MB	Reserved
ROM internal(1)	4000 0000h	4001 FFFFh	128KB	Reserved
	4002 0000h	4002 BFFFh	48KB	Public
	4002 C000h	400F FFFFh	848KB	Reserved
Reserved(1)	4010 0000h	401F FFFFh	1MB	Reserved
Reserved(1)	4020 0000h	402E FFFFh	960KB	Reserved
Reserved	402F 0000h	402F FFFFh	64KB	Reserved
L3 Target Space	4030 0000h	4033 FFFFh	256KB	OCMC SRAM
	4034 0000h	403F FFFFh	768KB	Reserved
	4040 0000h	4043 FFFFh	256KB	OCMC SRAM
	4044 0000h	404F FFFFh	768KB	Reserved
	4050 0000h	407F FFFFh	3MB	Reserved
	4080 0000h	4083 FFFFh	256KB	C674x UMAP0 (L2 RAM)
	4084 0000h	40DF FFFFh	5888KB	Reserved
	40E0 0000h	40E0 7FFFh	32KB	C674x L1P Cache/RAM
	40E0 8000h	40EF FFFFh	992KB	Reserved
	40F0 0000h	40F0 7FFFh	32KB	C674x L1D Cache/RAM
	40F0 8000h	40FF FFFFh	992KB	Reserved
	4100 0000h	41FF FFFFh	16MB	Reserved
	4200 0000h	43FF FFFFh	32MB	Reserved
	4400 0000h	44BF FFFFh	12MB	L3 configuration registers
	44C0 0000h	45FF FFFFh	20MB	Reserved
	4600 0000h	463F FFFFh	4MB	McASP0
	4640 0000h	467F FFFFh	4MB	McASP1
	4680 0000h	46BF FFFFh	4MB	McASP2
	46C0 0000h	46FF FFFFh	4MB	HDMI 1.3 Tx
	4700 0000h	473F FFFFh	4MB	McBSP
	4740 0000h	477F FFFFh	4MB	USB2.0 Registers/CPPI
	4780 0000h	47BF FFFFh	4MB	Reserved
	47C0 0000h	47FF FFFFh	4MB	Reserved
	4800 0000h	481F FFFFh	2MB	Standard Peripheral domain (see Table 1-12)
ARM Subsystem INTC(1)	4820 0000h	4820 0FFFh	4KB	Cortex-A8 Interrupt Controller
Reserved(1)	4820 1000h	4823 FFFFh	252KB	Reserved
Reserved(1)	4824 1000h	4827 FFFFh	252KB	Reserved

(1) These addresses are decoded within the Cortex-A8 subsystem.

Table 1-15. Cortex-A8 Memory Map (continued)

Region Name	Start Address	End Address	Size	Description
L3 Target Space	4830 0000h	48FF FFFFh	13MB	Standard Peripheral domain (see Table 1-12)
	4900 0000h	490F FFFFh	1MB	EDMA TPCC Registers
	4910 0000h	497F FFFFh	7MB	Reserved
	4980 0000h	498F FFFFh	1MB	EDMA TPTC0 Registers
	4990 0000h	499F FFFFh	1MB	EDMA TPTC1 Registers
	49A0 0000h	49AF FFFFh	1MB	EDMA TPTC2 Registers
	49B0 0000h	49BF FFFFh	1MB	EDMA TPTC3 Registers
	49C0 0000h	49FF FFFFh	4MB	Reserved
	4A00 0000h	4AFF FFFFh	16MB	High-Speed Peripheral domain (see Table 1-13)
	4B00 0000h	4BFF FFFFh	16MB	EMU Subsystem region
	4C00 0000h	4CFF FFFFh	16MB	DDR EMIF0[2] Configuration registers
	4D00 0000h	4DFF FFFFh	16MB	DDR EMIF1[2] Configuration registers
	4E00 0000h	4FFF FFFFh	32MB	DDR DMM[2] Configuration registers
	5000 0000h	50FF FFFFh	16MB	GPMC Configuration registers
	5100 0000h	51FF FFFFh	16MB	PCIE Configuration registers
	5200 0000h	52FF FFFFh	16MB	Reserved
	5300 0000h	53FF FFFFh	16MB	HDVICP2-2 Host Port
	5400 0000h	54FF FFFFh	16MB	HDVICP2-2 SL2 Port
	5500 0000h	55FF FFFFh	16MB	Reserved
	5600 0000h	56FF FFFFh	16MB	SGX530 Slave Port
	5600 0000h	56FF FFFFh	16MB	Reserved
	5700 0000h	57FF FFFFh	16MB	Reserved
	5800 0000h	58FF FFFFh	16MB	HDVICP2-0 Host Port
	5900 0000h	59FF FFFFh	16MB	HDVICP2-0 SL2 Port
	5A00 0000h	5AFF FFFFh	16MB	HDVICP2-1 Host Port
	5B00 0000h	5BFF FFFFh	16MB	HDVICP2-1 SL2 Port
	5C00 0000h	5FFF FFFFh	64MB	Reserved
	6000 0000h	7FFF FFFFh	512MB	TILER Window
DDR EMIF0/1 SDRAM[3][4]	8000 0000h	BFFF FFFFh	1GB	DDR
DDR EMIF0/1 SDRAM[3][4]	C000 0000h	FFFF FFFFh	1GB	DDR

(2) These accesses occur through the DDR DMM TILER ports. The DDR DMM splits address ranges internally to address DDR EMIF and DDR DMM control registers based on DDR DMM tie-offs.
(3) These addresses are routed to the Master 0 port for direct connection to the DDR DMM ELLA port.
(4) DDR EMIF0 and DDR EMIF1 addresses may be contiguous or bank interleaved, depending on configuration of the DDR DMM.

1.5.5 C674x Memory Map

Because the C674x DSP has specific hardwired address decoding built in, the C674x DSP memory map is slightly different than that of the Cortex-A8. The C674x DSP has a separate CFG bus which is used to access L4 peripherals and its UMAP1 bus has a direct connection into HDVICP2 SL2 (HDVICP2-0 and HDVICP2-1 only) memories. All C674x MDMA port accesses are routed through the System MMU for address translation.

Table 1-16. C674x Memory Map

Region Name	Start Address	End Address	Size	Description
Reserved(1)	0000 0000h	003F FFFFh	4MB	Reserved
UMAP1(1)	0040 0000h	0043 FFFFh	256KB	C674x UMAP1 (HDVICP2-0 SL2)
Reserved (UMAP1)(1)	0044 0000h	004F FFFFh	768KB	Reserved
UMAP1(1)	0050 0000h	0053 FFFFh	256KB	C674x UMAP1 (HDVICP2-1 SL2)
Reserved (UMAP1)(1)	0054 0000h	005F FFFFh	768KB	Reserved
Reserved(1)	0060 0000h	007F FFFFh	2MB	Reserved
L2 SRAM(1)	0080 0000h	0083 FFFFh	256KB	C674x UMAP0 (L2 RAM)
Reserved(1)	0084 0000h	00DF FFFFh	5888KB	Reserved
L1P SRAM(1)	00E0 0000h	00E0 7FFFh	32KB	C674x L1P Cache/RAM
Reserved(1)	00E0 8000h	00EF FFFFh	992KB	Reserved
L1D SRAM(1)	00F0 0000h	00F0 7FFFh	32KB	C674x L1D Cache/RAM
Reserved(1)	00F0 8000h	017F FFFFh	9184KB	Reserved
Internal CFG(2)(3)	0180 0000h	01BF FFFFh	4MB	C674x Internal CFG registers
Reserved(3)	01C0 0000h	07FF FFFFh	100MB	Reserved
L4 Standard Domain(3)	0800 0000h	08FF FFFFh	16MB	Peripheral Domain (see Table 1-12)
EDMA TPCC(3)	0900 0000h	090F FFFFh	1MB	EDMA CC Registers
Reserved(3)	0910 0000h	097F FFFFh	7MB	Reserved
EDMA TPTC0(3)	0980 0000h	098F FFFFh	1MB	EDMA TC0 Registers
EDMA TPTC1(3)	0990 0000h	099F FFFFh	1MB	EDMA TC1 Registers
EDMA TPTC2(3)	09A0 0000h	09AF FFFFh	1MB	EDMA TC2 Registers
EDMA TPTC3(3)	09B0 0000h	09BF FFFFh	1MB	EDMA TC3 Registers
Reserved(3)	09C0 0000h	09FF FFFFh	4MB	Reserved
L4 High-Speed Domain(3)	0A00 0000h	0AFF FFFFh	16MB	Peripheral Domain (see Table 1-13)
Reserved(3)	0B00 0000h	0FFF FFFFh	80MB	Reserved
C674x L1/L2(4)	1000 0000h	10FF FFFFh	16MB	C674x Internal Global Address
MDMA L3(5)	1100 0000h	FFFF FFFFh	3824MB	System MMU Mapped L3 Regions

(1) Addresses 0000 0000h to 017F FFFFh are internal to the C674x device.
(2) Addresses 0180 0000h to 01BF FFFFh are reserved for C674x internal CFG registers.
(3) Addresses 01C0 0000h to 0FFF FFFFh are mapped to the C674x CFG bus.
(4) Addresses 1000 0000h to 10FF FFFFh are mapped to C674x internal addresses 0000 0000h to 00FF FFFFh.
(5) These accesses are routed through the System MMU where the page tables translate to the physical L3 addresses shown in Table 1-11.

1.6 System MMU

1.6.1 MMU Overview

A memory management unit (MMU) is a hardware component responsible for handling accesses to memory requested by a processing unit. MMU functions include translation of virtual addresses to physical addresses (that is, virtual memory management).

The device contains the following MMUs:

- System MMU for C674x DSP and other requestors (please refer to the device-specific data manual for other requestors)
- L1 Shared cache MMU
- ARM® Cortex®-A8 MMU
- SGX530 MMU

This section provides a detailed description of the System MMU. For more details on the Cortex-A8 MMU, refer to the ARM® Cortex®-A8 Technical Reference Manual.

1.6.2 MMU Integration

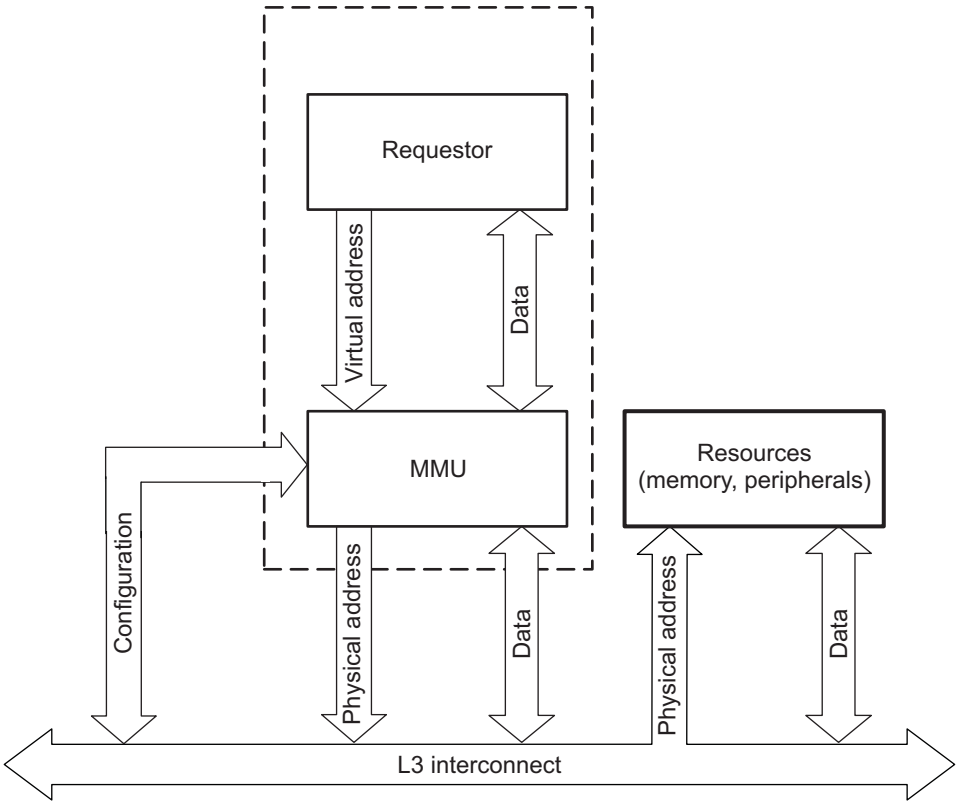
This section describes module integration in the device.

The MMU communicates accesses from the requestor (C674x DSPor Cortex-A8 MPU) to the L3 main interconnect, performing virtual to physical address translation. All MMUs are programmed (configured) through the L3 interconnect. For MMU interrupt details, refer to the device datasheet.

Figure 1-12 shows typical MMU integration.

The System MMU is dedicated to C674x DSP MDMA and other requestors. The system MMU configuration is done through MMU_CFG register (see the *Control Module* section). The MMU_CFG register is used to optionally route certain requestors through the System MMU. The C674x DSP MDMA port accesses will always be routed through the System MMU so no configuration bit is instantiated for C674x DSP.

Figure 1-12. Typical MMU Integration

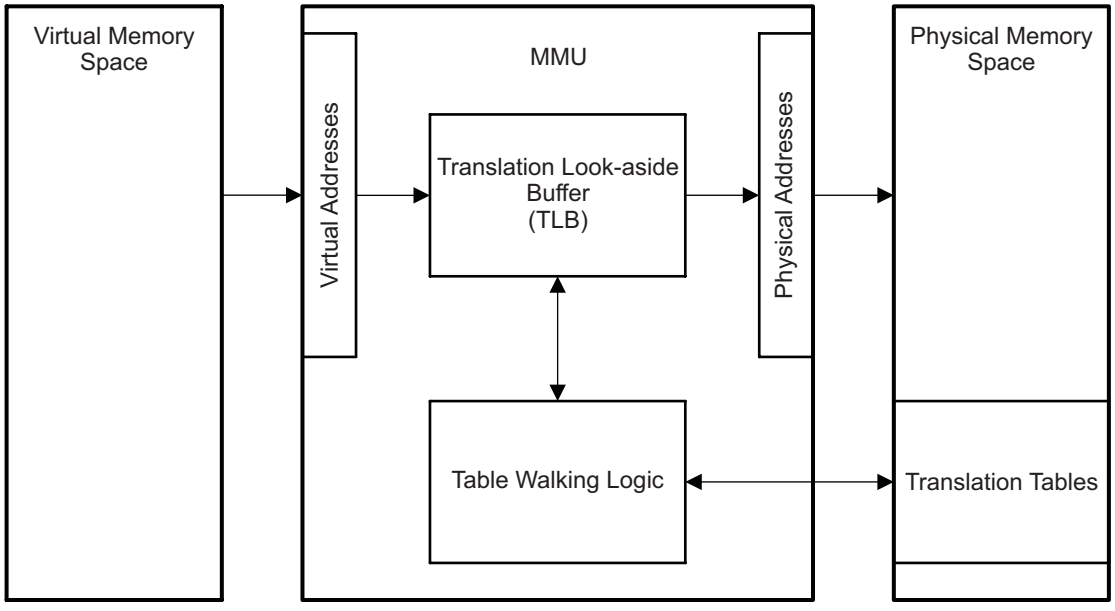


1.6.3 MMU Functional Description

1.6.3.1 MMU Block Diagram

The MMU manages the virtual to physical address translation for external addresses, as well as endianness conversion. The MMU can be programmed through the L4 interconnect.

Figure 1-13. MMU Block Diagram



Each table entry describes the translation of one contiguous memory region. For a description of the structure of these tables, see Translation Tables.

Two major functional units exist in the MMU to provide address translation automatically based on the table entries:

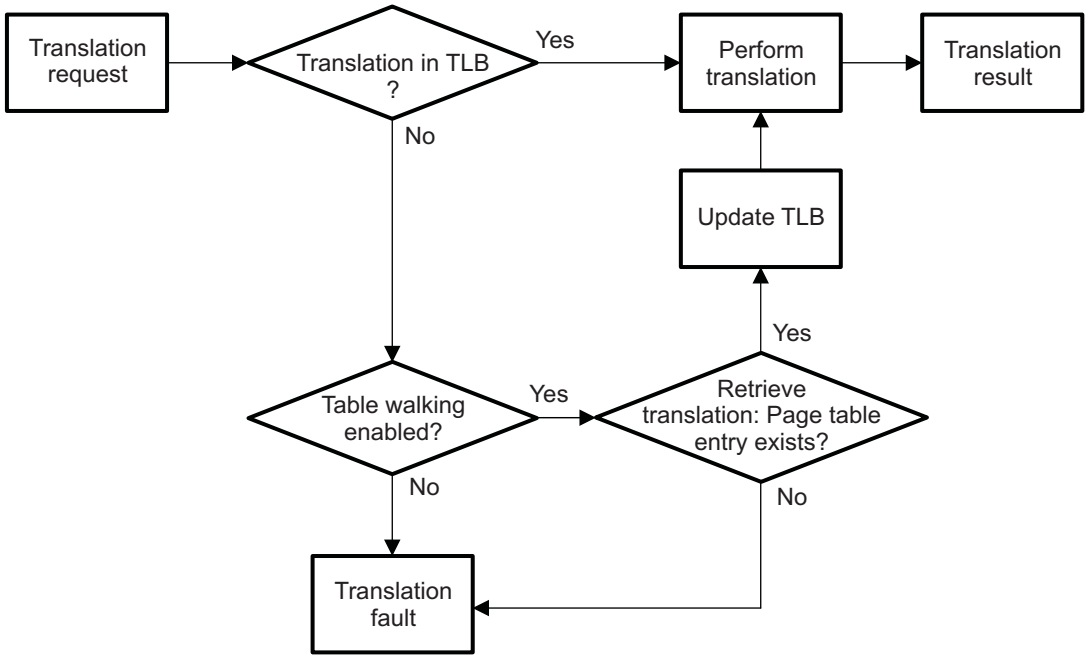
- The table walker automatically retrieves the correct translation table entry for a requested translation. If two-level translation is used (for the translation of small memory pages), the table walker also automatically reads the required second-level translation table entry. The two-level translation is described later in this section.
- The translation look-aside buffer (TLB) stores recently used translation entries, acting like a cache of the translation table.

1.6.3.1.1 MMU Address Translation Process

Whenever an address translation is requested (that is, for every access with the MMU enabled), the MMU first checks whether the translation is contained in the TLB, which acts like a cache storing recent translations. The TLB can also be programmed manually to ensure that time-critical data can be translated without delay.

If the requested translation is not in the TLB, the table-walking logic retrieves this translation from the translation table(s), and then updates the TLB. The address translation is then performed. Figure 1-14 summarizes the process.

Figure 1-14. MMU Address Translation Process



1.6.3.1.2 Translation Tables

The translation of virtual to physical addresses is based on entries in translation tables that define the following properties:

- Address translation, that is, the correspondence between virtual and physical addresses
- Size of the memory region the entry translates
- Endianness, data access size, and the mixed property of this memory region

The virtual addresses index the translation tables. Each virtual address corresponds to exactly one entry in the translation table.

1.6.3.1.2.1 Translation Table Hierarchy

When developing a table-based address translation scheme, one of the most important design parameters is the memory page size described by each translation table entry. MMU instances support 4KB and 64KB pages, a 1MB section, and a 16MB supersection. Using bigger page sizes means a smaller translation table.

Using a smaller page size greatly increases the efficiency of dynamic memory allocation and defragmentation. That is why many operating systems (OSs) can operate on memory blocks as small as 4KB; however, the smaller size implies a more complex table structure.

A quick calculation shows that using 4KB memory pages with one translation table would require one million entries to span the entire 4GB address range. The table itself would be 32MB, a size that is not feasible.

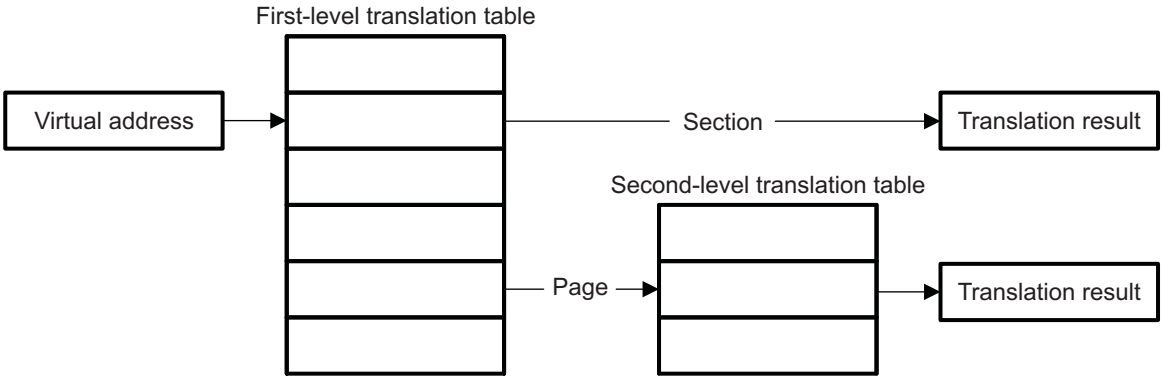
However, using bigger pages greatly reduces the functionality of the typical OS memory management. Implementing a two-level hierarchy reconciles these two requirements. Within this hierarchy, one first-level translation table describes the translation properties based on 1MB memory regions.

Each of the entries in this first-level translation table can specify the following:

- The translation properties for a big memory section. This memory section can be either 1MB (section) or 16MB (supersection). In this case, all translation parameters are specified in the first-level translation table entry.
- A pointer to a second-level translation table that specifies individual translation properties based on smaller pages within the 1MB page of memory. These pages can be either 64KB (large page) or 4KB (small page). In this case, the actual translation parameters are specified in the second-level translation table entry. The first-level translation table entry specifies only the base address of the second-level translation table.

This hierarchical approach means that additional translation information for smaller pages must be provided only when the pages are actually used.

Figure 1-15. Translation Hierarchy



The structure of the first and second-level translation tables and their entries are described in more detail in First-Level Translation Table and Two-Level Translation.

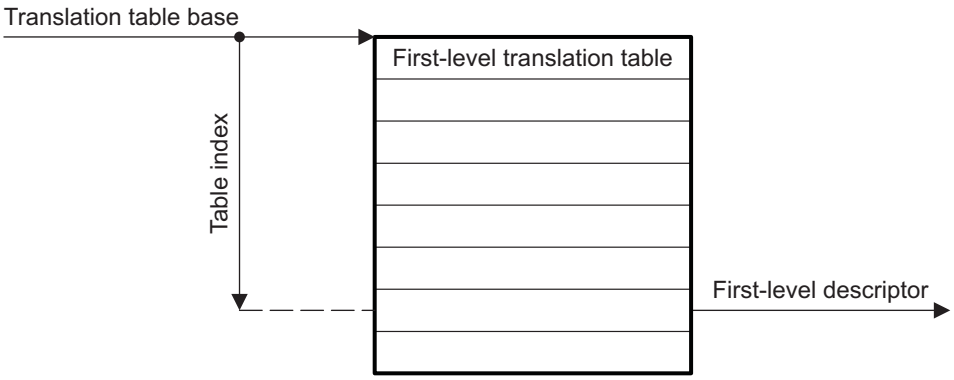
1.6.3.1.2.2 First-Level Translation Table

The first-level translation table describes the translation properties for 1MB sections. To describe a 4GB address range requires 4096 32-bit entries (so-called first-level descriptors).

The first-level translation table start address must be aligned on a multiple of the table size with a 128-byte minimum. Consequently, an alignment of at least 16K bytes is required for a complete 4096-entry table; that is, at least the last fourteen address bits must be zero.

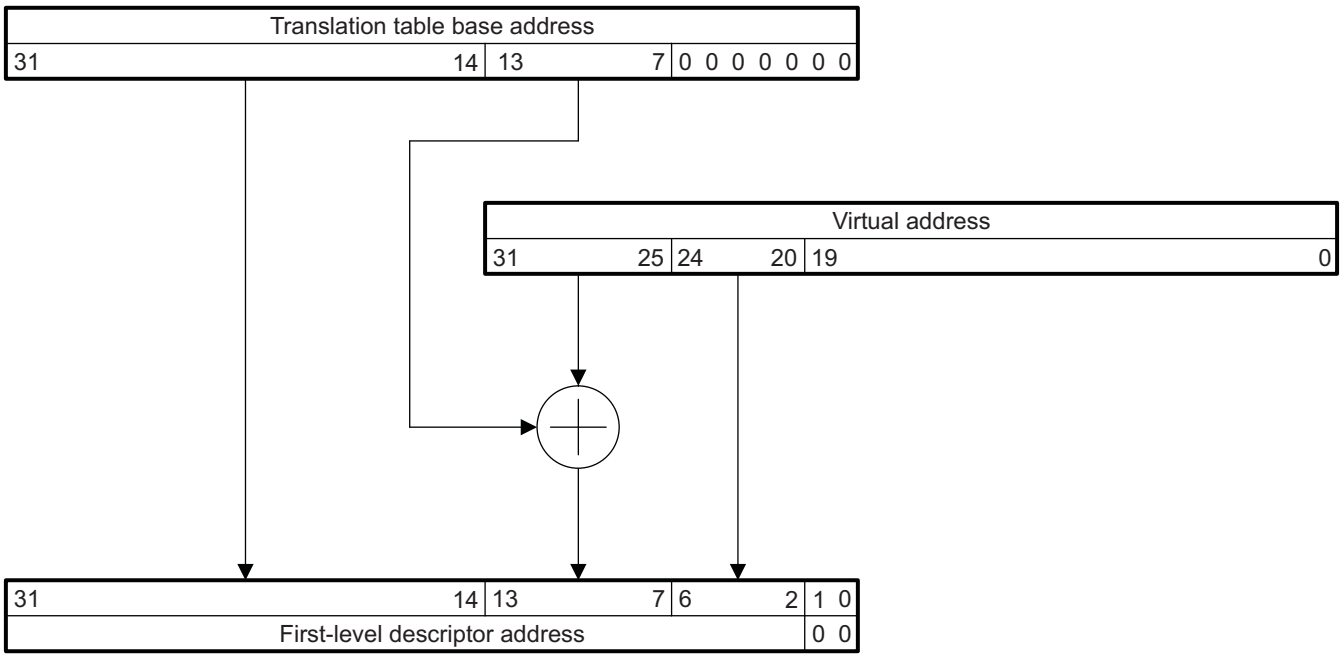
The start address of the first-level translation table is specified by the so-called translation table base. The table is indexed by the upper 12-bits of the virtual address.

Figure 1-16. First-Level Descriptor Address Calculation



To summarize, the translation table base and the translation table index together define the first-level descriptor address. The precise mechanism used to calculate this address is shown in Figure 1-17.

Figure 1-17. Detailed First-Level Descriptor Address Calculation



As an example of this mechanism, consider a translation table base address of 0x8000:0000 and a virtual address of 0x1234:5678. In this case, the first-level descriptor address is:

0x8000:0000 + (0x123 <<2) = 0x8000:048C.

1.6.3.1.2.3 First-Level Descriptor Format

Each first-level descriptor provides either the complete address translation for 1MB or 16MB sections or provides a pointer to a second-level translation table for 4KB or 64KB pages.

Table 1-17. First-Level Descriptor Format

31:24	23:20	19	18	17	16	15	14:12	11:10	9:2	1	0	
X										0	0	Fault
Second-level Translation Table Base Address									X	0	1	Page
Section Base Address		X	0	M	X	E	X	ES	X	1	0	Section
Super Section base Address		X	1	M	X	E	X	ES	X	1	0	Supersection
X										1	1	Fault

M = Mixed region: Don't care when E = 0. Set to 0 to ensure future compatibility.

E = Endianness: Set to 0 for little-endian. Big-endian mode is not supported.

ES = Element Size: Don't care when E = 0. Set to 00 to ensure future compatibility.

X = Don't care. Set to 0 to ensure future compatibility.

1.6.3.1.2.4 First-Level Page Descriptor Format

If a translation granularity smaller than 1MB is required, a two-level translation process is used. In this case, the first-level block descriptor specifies only the start address of a second-level translation table. The second-level translation table entries specify the actual translation properties.

1.6.3.1.2.5 First-Level Section Descriptor Format

Each section descriptor in the first-level translation table specifies the complete translation properties for a 1MB section or a 16MB supersection.

NOTE: Supersection descriptors must be repeated 16 times, because each descriptor in the first-level translation table describes 1MB of memory. If an access points to a descriptor that is not initialized, the MMU will behave in an unpredictable way.

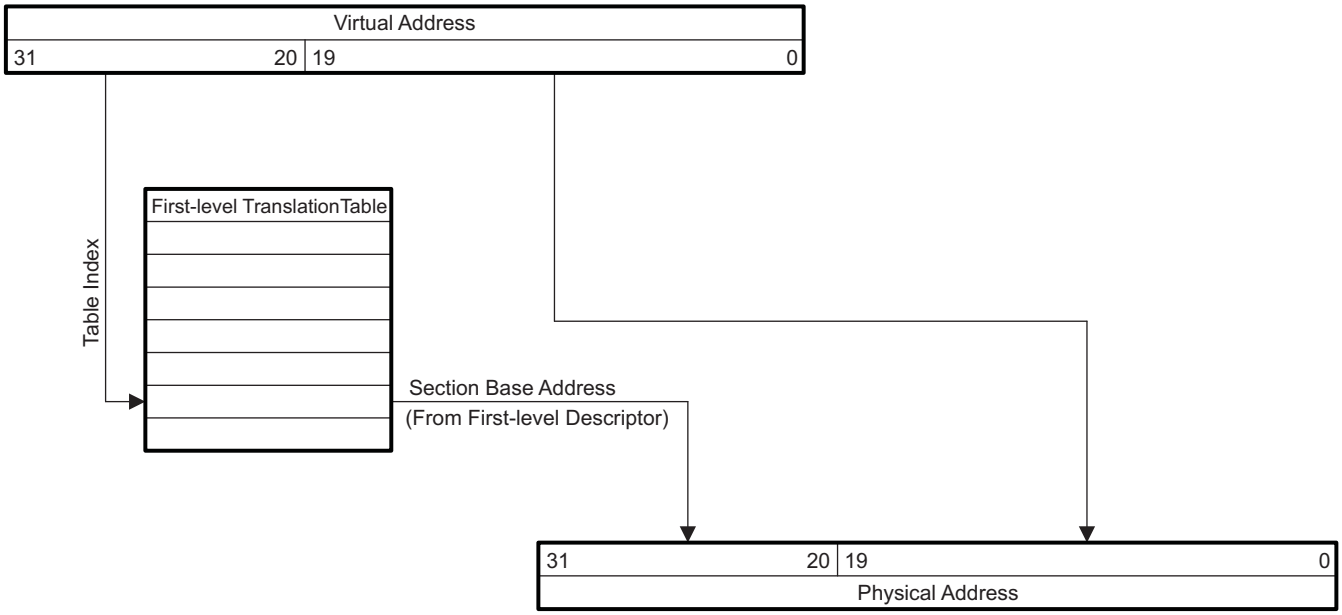
In addition to the address translation itself, three parameters are specified in the section descriptors:

- **Endianness -** The endianness parameter specifies whether the memory section uses a big- or little-endian data format. This parameter is locked to little endian. Big-endian mode is not supported.
- **Element Size -** The element size parameter can optionally specify the data access size (8, 16, or 32 bits) for all data items in the defined section.
- **Mixed Region -** The mixed region parameter specifies whether the information about the data access size is detected from the access itself (access-based detection) or if the specified element size parameter is used (page-based detection). For example, the specified element size parameter can be used when several smaller sized accesses are packed into a bigger sized access, such as two 16-bit accesses packed into one 32-bit access. In this case, with no specified data access size, 32 bits would be the access size detected, leading to an incorrect result. To avoid this problem, specify the data access size for the memory section.

1.6.3.1.2.6 Section Translation Summary

Sections and supersections can be translated based solely on the information in the first-level translation table.

Figure 1-18. Section Translation Summary



1.6.3.1.2.7 Supersection Translation Summary

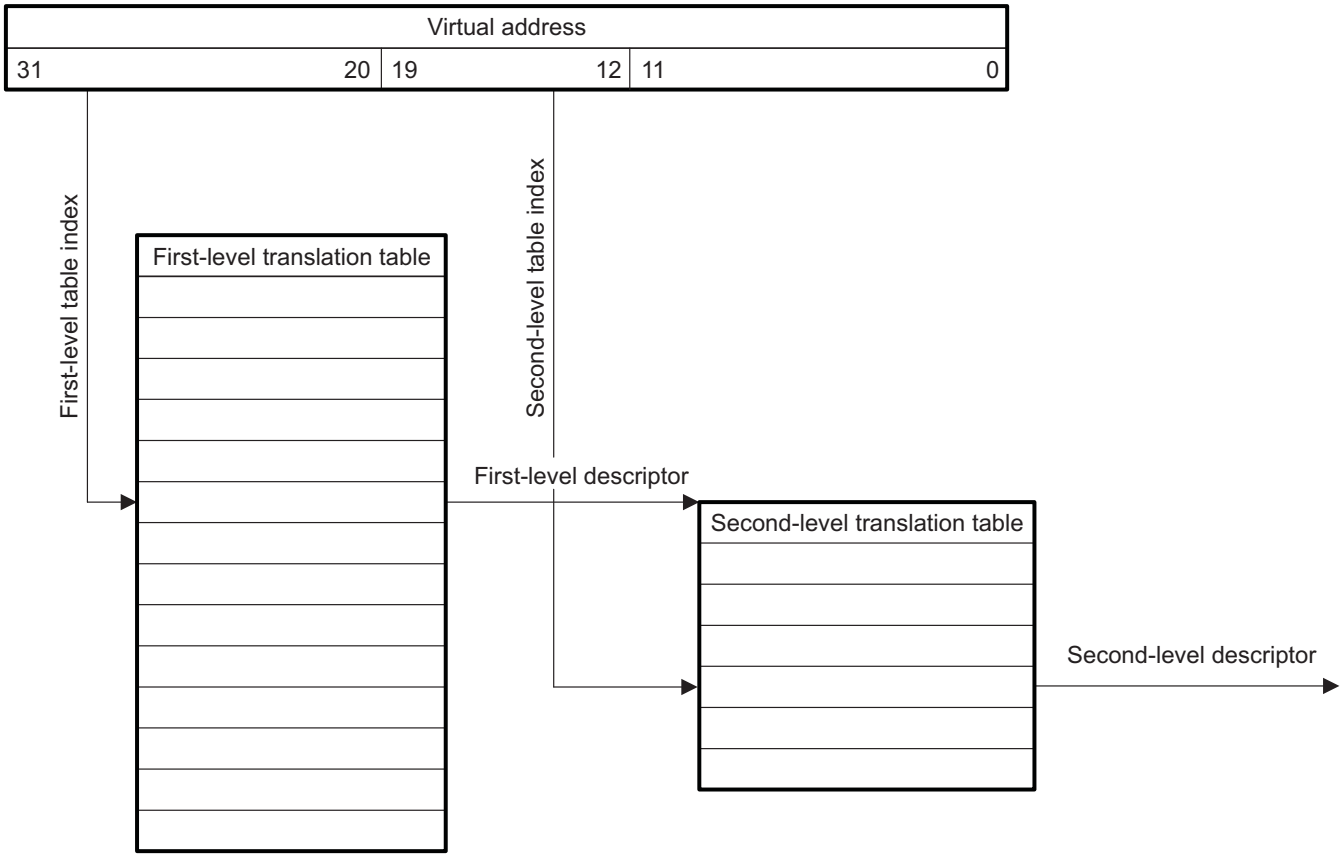
The translation of a supersection is similar to the translation of a section. The difference is that for a supersection only bits 31 to 24 index into the first-level translation table. The last four bits of the table index are implicitly assumed to be zero as there are 16 identical consecutive entries for a supersection.

Figure 1-19. Supersection Translation Summary

1.6.3.1.2.8 Two-Level Translation

Two-level translation is used when fine-grain granularity is required, that is, when memory sections smaller than 1MB are needed. In this case, the first-level descriptor provides a pointer to the base address of a second-level translation table. This second-level table is indexed by bits 19 to 12 of the virtual address.

Figure 1-20. Two-Level Translation



Each second-level translation table describes the translation of 1MB of address space in pages of 64KB (large page) or 4KB (small page). It consists of 256 second-level descriptors describing 4KB each.

NOTE: In the case of a large page, the same descriptor must be repeated 16 times. If an access points to a descriptor that is not initialized, the MMU will behave in an unpredictable way."

1.6.3.1.2.9 Second-Level Descriptor Format

Similar to first-level section descriptors, second-level descriptors provide all of the necessary information for the translation of a large or small page. The translation parameters (endianness, element size, and mixed region) have the same meaning as those for sections.

Table 1-18. Second-Level Descriptor Format

31:16	15:12	11	10	9	8:6	5:4	3:2	1	0	
X								0	0	Fault
Large Page Base Address	X	M	X	E	X	ES	X	0	1	Large Page
Small Page Base Address		M	X	E	X	ES	X	1	X	Small Page

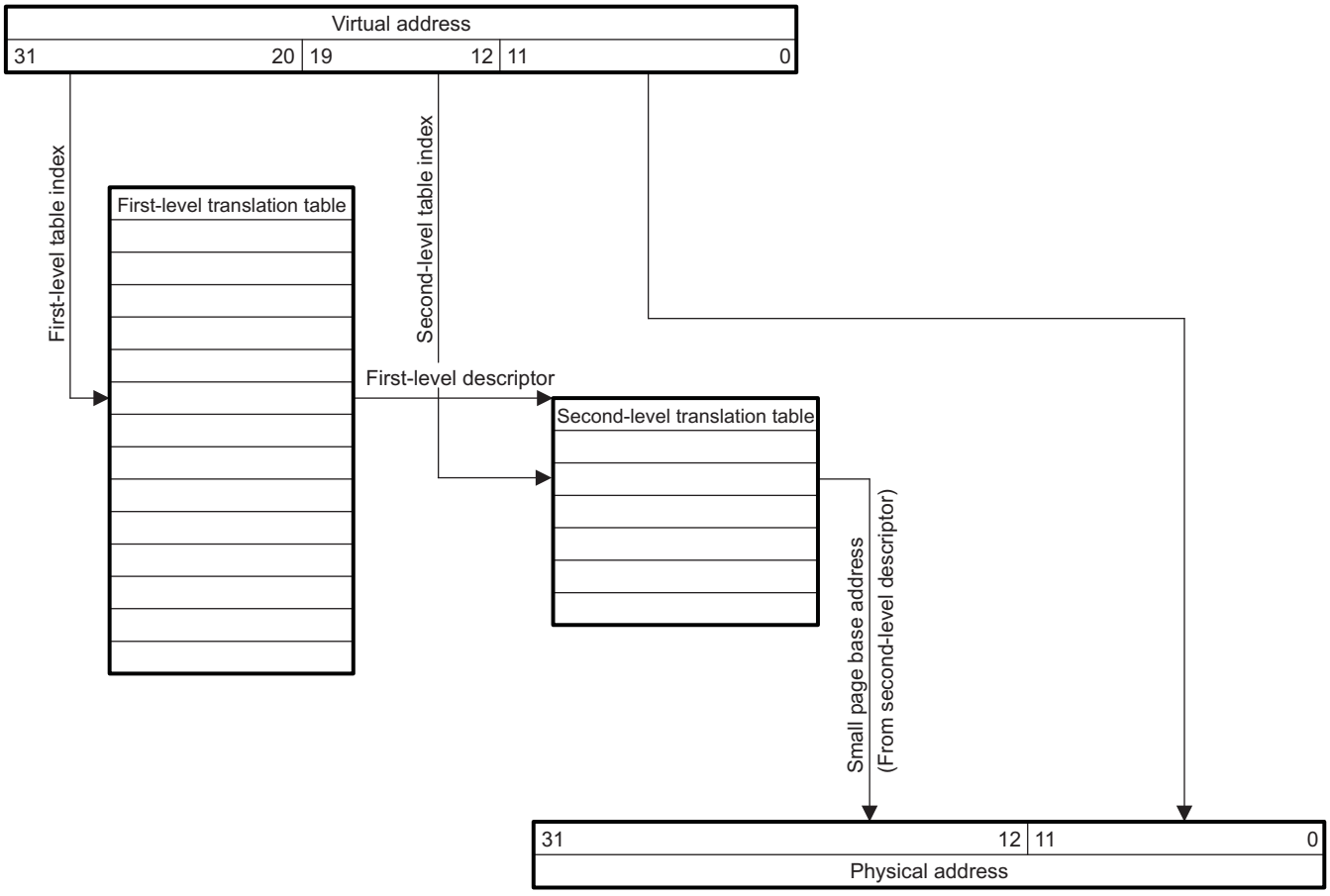
M = Mixed region: Don't care when E = 0. Set to 0 to ensure future compatibility.

E = Endianness: Set to 0 for little-endian. Big-endian mode is not supported

ES = Element Size: Don't care when E = 0. Set to 00 to ensure future compatibility

X = Don't care. Set to 0 to ensure future compatibility.

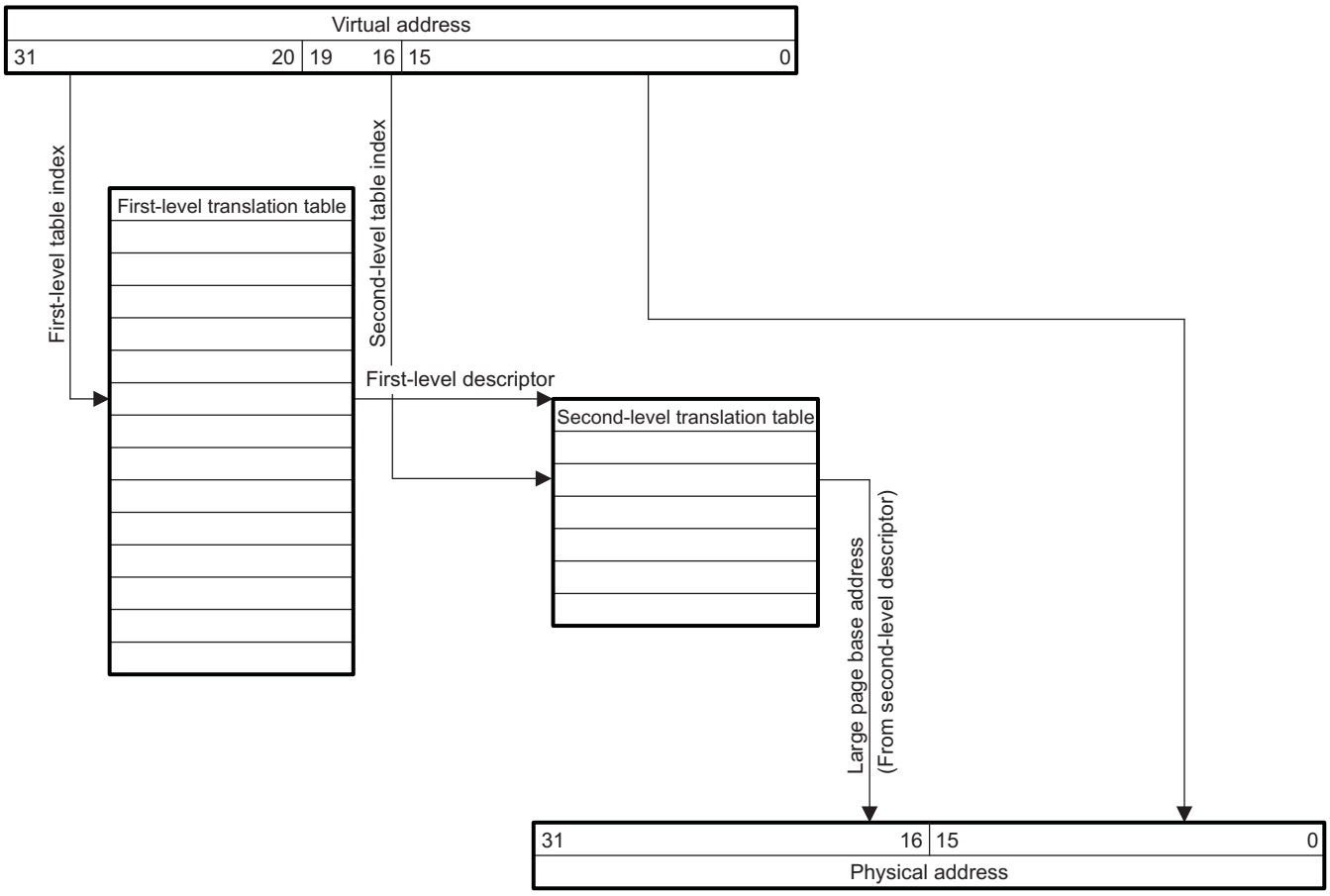
Figure 1-21. Small Page Translation Summary



1.6.3.1.2.10 Large Page Translation Summary

The translation of a large page is similar to the translation of a small page. The difference is that, for a large page, only bits 19 to 16 index into the second-level translation table. The last four bits of the table index are implicitly assumed to be zero as there are 16 identical consecutive entries for a large page.

Figure 1-22. Large Page Translation Summary



1.6.3.1.3 Translation Lookaside Buffer

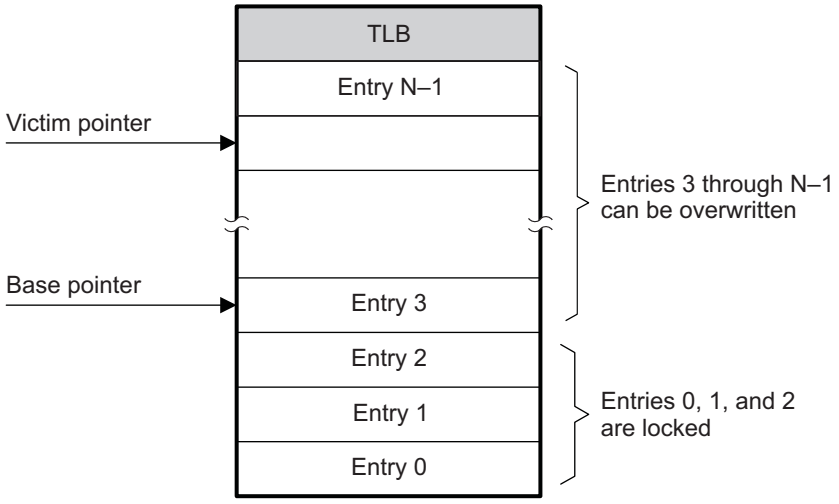
Translating virtual addresses to physical addresses is required for each memory access in systems using an MMU. To accelerate this translation process, a cache, or TLB, holds the result of recent translations.

For every translation, the MMU internal logic first checks whether the requested translation is already cached in the TLB. If the translation is cached, this translation is used; otherwise the translation is retrieved from the translation tables and the TLB is updated. If the TLB is full, one of its entries must be replaced. This entry is selected on a random basis.

The first n TLB entries, where n < Total Number N of TLB Entries , can be protected (locked) against being overwritten by setting the TLB base pointer to n . When this mechanism is used, only unprotected entries can be overwritten. The victim pointer indicates the next TLB entry to be written. Figure 1-23 shows an example of the TLB with N TLB entries (ranging from 0 to N-1). The base pointer contains the value "3" protecting Entry 0, Entry 1, and Entry 2 and the victim pointer points to the next TLB entry to be updated.

NOTE: The last TLB entry (Entry N-1) always remains unprotected.

Figure 1-23. TLB-Entry Lock Mechanism



The table walking logic automatically writes the TLB entries. The entries can also be manually written, which is done typically to ensure that the translation of time-critical data accesses is already present in the TLB so that they execute as fast as possible. The entries must be locked to prevent them from being overwritten.

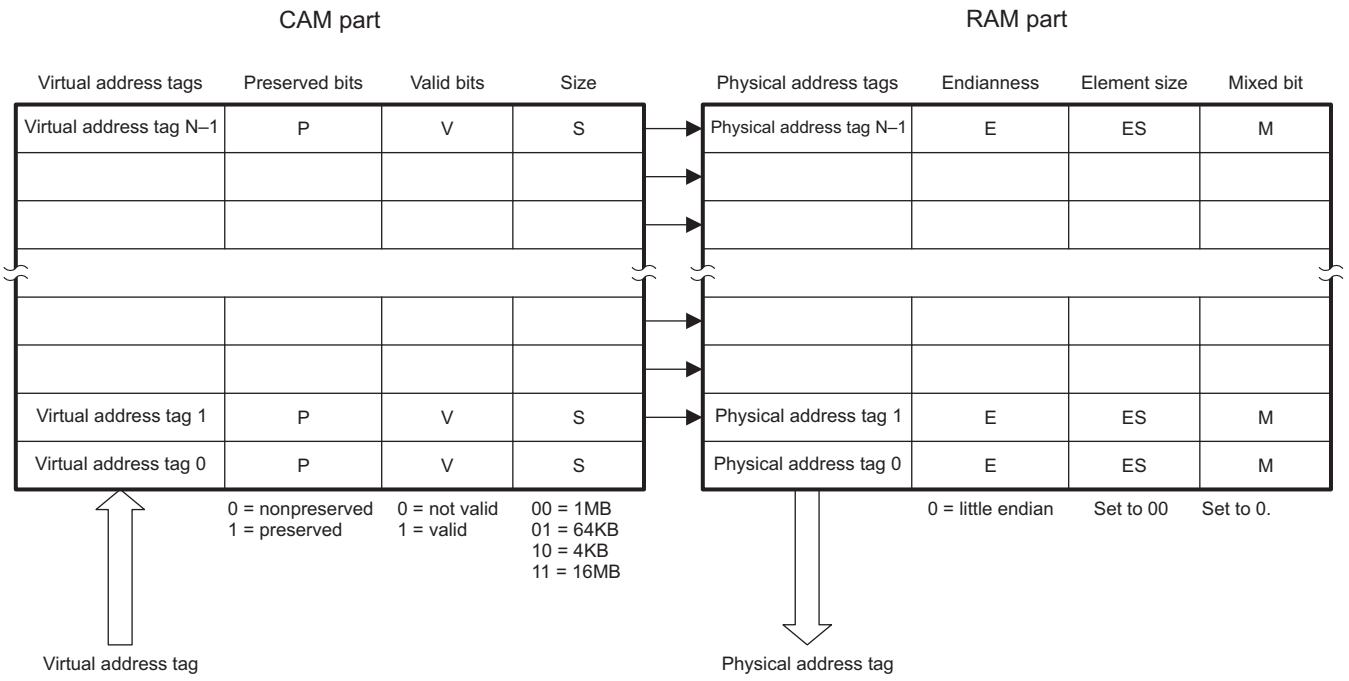
1.6.3.1.3.1 TLB Entry Format

TLB entries consist of two parts:

- The CAM part contains the virtual address tag used to determine if a virtual address translation is in the TLB. The TLB acts like a fully associative cache addressed by the virtual address tag. The CAM part also contains the section/page size, as well as the preserved and the valid parameters. See the register table for more details.
- The RAM part contains the address translation that belongs to the virtual address tag as well as the endianness, element size, and mixed parameters described in First-Level Translation Table. See the MMU_RAM register table for more details.

The valid parameter specifies whether an entry is valid or not. The preserved parameter determines the behavior of an entry in the event of a TLB flush. If an entry is set as preserved, it is not deleted when a TLB is flushed, that is, when [0] GLOBALFLUSH is set to 1. Preserved entries must be deleted manually.

Figure 1-24. TLB-Entry Structure



1.6.3.2 MMU Clock Configuration

There are two clock domains: The functional clock domain for the MMU, which is synchronous to the clock for the interconnect slave and master access ports; and the clock domain for the interconnect slave configuration port. As these clocks are matched, there is a single input clock with enables for each of the clock domains. If a clock domain should run at the same frequency as the input clock, that enable can be tied high.

Two clock enable signals exist, one to enable the interconnect data master and slave ports, and the other to enable the clock on the configuration L3 interconnect port. The clock signals are configured through the MMU_SYSCONFIG register. This is a system configuration register that controls the various parameters of the L3 interface.

1.6.3.3 MMU Software Reset

This section describes the software reset feature of the module. The MMU instances are reset together with their respective reset domains. See table Clocks and Resets for information about the reset domains of the different MMU instances.

To perform a software reset, write 1 in the MMU_SYSCONFIG[1] SOFTRESET bit. When the software reset completes, the MMU_SYSCONFIG[1] SOFTRESET bit is automatically reset. The software must ensure that the software reset completes before doing MMU operations. When an MMU instance is released from reset, its TLB is empty and the MMU is disabled.

1.6.3.4 MMU Power Management

As part of the device system-wide power management scheme, each MMU instance supports a communication protocol with the PRCM module that allows the PRCM module to request an MMU instance to enter a low-power state. When the MMU instance acknowledges a low-power mode request from the PRCM module, the clock to the instance is gated off at the PRCM clock generator. Because the clock is disabled at the source, the low-power mode offers lower power consumption than the internal clock gating method in the local power management.

1.6.3.5 MMU Local Power Management Features

Table 1-19. MMU Local Power Management Features

Feature	Register
Idle modes	MMU_SYSCONFIG[4:3] IDLEMODE
Clock activity	MMU_SYSCONFIG[9:8] CLOCKACTIVITY
Clock autogating	MMU_SYSCONFIG[0] AUTOIDLE

NOTE: The MMU_SYSCONFIG[9:8] CLOCKACTIVITY bits are read only.

1.6.3.6 MMU Interrupt Requests

Table 1-20. Events

Event Flag	Event Mask	Synchronous	Sensitivity	Map to	Description
MMU_IRQSTATUS[4] MULTIHITFAULT	MMU_IRQENABLE[4] MULTIHITFAULT	Yes	Level	M3_IRQ_0	Error in the L2 MMU due to multiple matches in the TLB
MMU_IRQSTATUS[3] TABLEWALKFAULT	MMU_IRQENABLE[3] TABLEWALKFAULT	Yes	Level	M3_IRQ_0	Error in the L2 MMU due to error response received during a Table Walk
MMU_IRQSTATUS[2] EMUMISS	MMU_IRQENABLE[2] EMUMISS				
MMU_IRQSTATUS[1] TRANSLATIONFAULT	MMU_IRQENABLE[1] TRANSLATIONFAULT	Yes	Level	M3_IRQ_0	Error in the L2 MMU due to invalid descriptor in the translation tables (translation fault)
MMU_IRQSTATUS[0] TLBMISS	MMU_IRQENABLE[0] TLBMISS	Yes	Level	M3_IRQ_0	Error in L2 MMU due to unrecoverable TLB miss (hardware TWL disabled)

1.6.3.7 MMU Error Handling

Table 1-21. Error Handling

Item	Condition	Action
1	Table-walk read has an error response.	Treat generally the same as a translation fault, but set the TableWalkFault interrupt status bit to aid in diagnosis
2	MMU is disabled during table-walk.	Not permitted; can result in loss of the current transaction but must not deadlock the MMU. Avoid this condition by first disabling the table-walk logic and then polling the TWLRunning bit to ensure that no table walk is pending.
3	MMU is disabled during an address translation.	Not permitted; can result in access to an unintended location, but must not deadlock MMU. This condition should be avoided by ensuring that no accesses are pending.
4	TLB is accessed during an address translation or a table walk.	Reading permitted; write should be done with care to ensure that the TLB is self-consistent at all times that a translation can occur.
5	TLB is flushed during address translation or a table walk.	Permitted; the flush is processed first, followed by the TWL update.
6	MMU is disabled while an interrupt is pending.	Not permitted; all pending interrupts should be processed before disabling the MMU.

L3 Interconnect configuration port: Accesses to undecoded register addresses must not give an error response.

To protect against changes to the address translation between a READEX and the corresponding write or during a burst the following configuration operations are protected against writes during these processes:

1. TLB update
2. Global flush
3. Flush entry
4. MMU disable

The protection is implemented by stalling the configuration interconnect transaction until the write can proceed safely.

1.6.4 MMU Low-level Programming Models

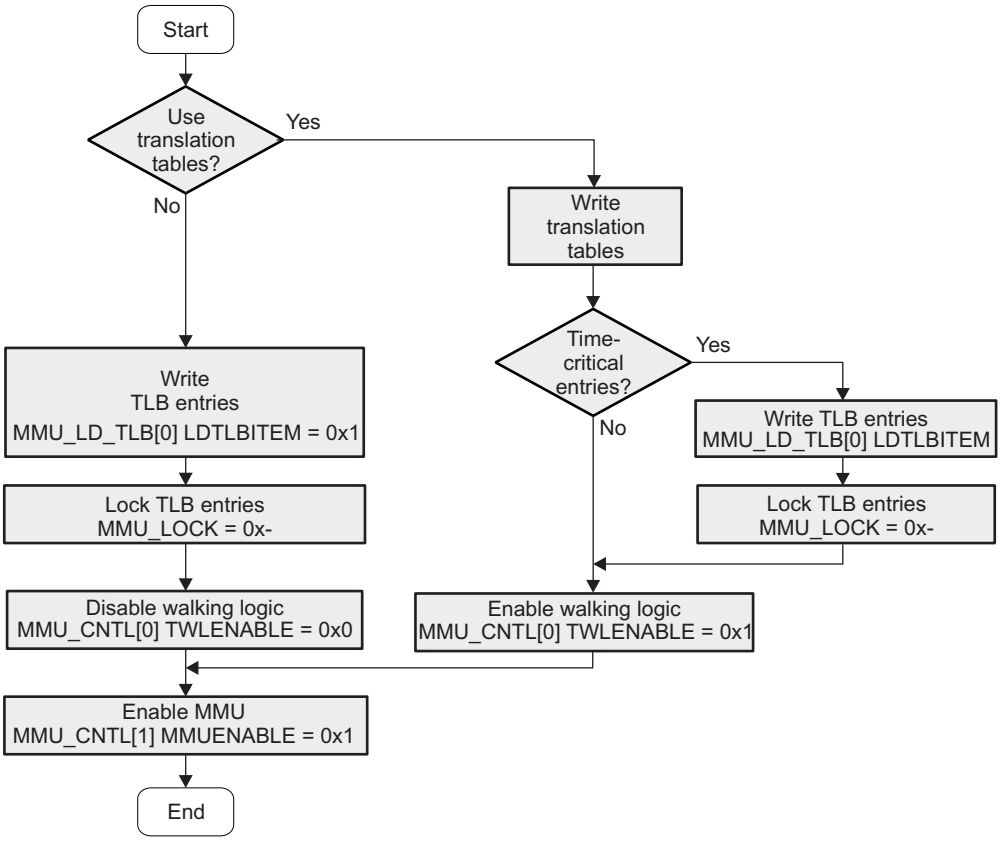
This section covers the low-level hardware programming sequences for configuration and usage of the module.

1.6.4.1 MMU Global Initialization

1.6.4.1.1 Main Sequence - MMU Global Initialization

The System MMU can be enabled or disabled in MMU_CFG register (see the *Control Module* section). Figure 1-25 shows the procedure to initialize the MMU after a power-on or software reset.

Figure 1-25. MMU Global Initialization



1.6.4.1.2 Subsequence - Configure a TLB entry

Table 1-22. Configure a TLB Entry

Step	Register / Bitfield / Programming Model	Value
Load the Virtual Address Tag	MMU_CAM[31:12] VATAG	0x-
Protect the TLB entry against flush	MMU_CAM[3] P	0x1
Validate the TLB entry	MMU_CAM[2] V	0x1
Define the page size	MMU_CAM[1:0] PAGESIZE	0x-

1.6.4.2 Operational Modes Configuration

1.6.4.2.1 Main Sequence - Writing TLB Entries Statically

Writing TLB entries statically avoids the need to write translation tables in memory and is commonly used for relatively small address spaces. This method ensures that the translation of time-critical data accesses execute as fast as possible with entries already present in the TLB. These entries must be locked to prevent them from being overwritten.

Table 1-23. MMU Writing TLB Entries Statically

Step	Register/ Bitfield / Programming Model	Value
Execute software reset	MMU_SYSCONFIG[1] SOFTRESET	0x1
Wait for reset to complete	MMU_SYSSTATUS[0] RESETDONE	0x1
Enable power saving via automatic interface clock gating	MMU_SYSCONFIG[0] AUTOIDLE	0x1
Configure TLB entries	Refer to table Configure a TLB Entry	
Load the physical Address of the page	MMU_RAM[31:12] PHYSICALADDRESS	0x-
Define the endianness of the page (little endian or big endian)	MMU_RAM[9] ENDIANNESS	0x-
Select the element size	MMU_RAM[8:7] ELEMENTSIZE	0x-
Define mixed page attribute	MMU_RAM[6] MIXED	0x-
Specify the TLB entry you want to write	MMU_LOCK[8:4] CURRENTVICTIM	0x-
Load the specified entry in the TLB	MMU_LD_TLB[0] LDTLBITEM	0x1
Enable multihit fault and TLB miss	MMU_IRQENABLE[4] MULTIHITFAULT	0x1
	MMU_IRQENABLE[0] TLBMISS	0x1
Enable memory translations	MMU_CNTL[1] MMUENABLE	0x1

1.6.4.2.2 Main Sequence - Protecting TLB Entries

The first n TLB entries (with n < total number of TLB entries) can be protected from being overwritten with new translations. This is useful to ensure that certain commonly used or time-critical translations are always in the TLB and do not require retrieval using the table walking process.

Table 1-24. Protecting TLB Entries

Step	Register / Bitfield / Programming Model	Value
Locks the TLB entries	MMU_LOCK[14:10] BASEVALUE	0x-

1.6.4.2.3 Main Sequence - Deleting TLB Entries

Two mechanisms exist to delete TLB entries. All unpreserved TLB entries, that is, TLB entries written with the preserved bit set to zero, can be deleted by invoking a TLB flush. The preserved bit should only be used on protected TLB entries, as it does not prevent replacement by the table walking logic.

Table 1-25. Deleting TLB Entries

Step	Register / Bitfield / Programming Model	Value
Flush all nonprotected TLB entries	MMU_GFLUSH[0] GLOBALFLUSH	0x1
Flush all TLB entries specified by the CAM register	MMU_FLUSH_ENTRY[0] FLUSHENTRY	0x1

1.6.4.2.4 Main Sequence - Read TLB Entries

TLB entries can be read by the programmer to determine the TLB content at runtime.

Table 1-26. Read TLB Entries

Step	Register / Bitfield / Programming Model	Value
Set the current victim pointer	MMU_LOCK[8:4] CURRENTVICTIM	0x-
Read RAM parts of the TLB entry	MMU_READ_RAM	
Read CAM parts of the TLB entry	MMU_READ_CAM	

1.6.4.3 Recovery from a TLB Miss Error

It is possible for the DMMU to hang when used in Table-Walk Mode.

The following procedure must be considered for recovery from a TLB Miss Error:

1. Read MMU_FAULT_AD[31:0] FAULTADDRESS to get the misdefined virtual address value.
2. Write the physical address (corresponding to the misdefined virtual address derived in 1.) in the MMU_RAM[31:12]PHYSICALADDRESS register
3. Fault virtual address bits [31:12] reported in MMU_FAULT_AD[31:0] are copied to MMU_CAM[31:12]VATAG bit field.
4. Write 0xC in MMU_CAM[3:0] bits. This means:
   - P is set to 1 to preserve entry
   - V is set to 1 to validate entry
   - [1:0]PAGESIZE = 0x0 for a 1 MiB sized section
5. Read MMU_IRQSTATUS register.
6. Write the value from step 5. into the same (MMU_IRQSTATUS) register to clear the error.

1.6.5 MMU Registers

Table 1-27 provides a summary of the MMU registers. For the base address of these registers, see Table 1-12.

Table 1-27. MMU Registers

Address Offset	Acronym	Type	Section
00h	MMU_REVISION	R	Section 1.6.5.1
10h	MMU_SYSCONFIG	RW	Section 1.6.5.2
14h	MMU_SYSSTATUS	R	Section 1.6.5.3
18h	MMU_IRQSTATUS	RW	Section 1.6.5.4
1Ch	MMU_IRQENABLE	RW	Section 1.6.5.5
40h	MMU_WALKING_ST	R	Section 1.6.5.6
44h	MMU_CNTL	RW	Section 1.6.5.7
48h	MMU_FAULT_AD	R	Section 1.6.5.8
4Ch	MMU_TTB	RW	Section 1.6.5.9
50h	MMU_LOCK	RW	Section 1.6.5.10
54h	MMU_LD_TLB	RW	Section 1.6.5.11
58h	MMU_CAM	RW	Section 1.6.5.12
5Ch	MMU_RAM	RW	Section 1.6.5.13
60h	MMU_GFLUSH	RW	Section 1.6.5.14
64h	MMU_FLUSH_ENTRY	RW	Section 1.6.5.15
68h	MMU_READ_CAM	R	Section 1.6.5.16
6Ch	MMU_READ_RAM	R	Section 1.6.5.17
70h	MMU_EMU_FAULT_AD	R	Section 1.6.5.18
80h	MMU_FAULT_PC	R	Section 1.6.5.19

1.6.5.1 MMU_REVISION

The MMU_REVISION register is shown in Figure 1-26 and described in Table 1-28.

Figure 1-26. MMU_REVISION

31	0
REVISION	
R-20h	

LEGEND: R = Read only; -*n* = value after reset

Table 1-28. MMU_REVISION Field Descriptions

Bit	Field	Value	Description
31-0	REVISION	20h	IP Revision.

1.6.5.2 MMU_SYSCONFIG

The MMU_SYSCONFIG register is shown in Figure 1-27 and described in Table 1-29.

Figure 1-27. MMU_SYSCONFIG

31	16
Reserved	
R-0	

15 10	9 8	7 5	4 3	2	1	0
Reserved	CLOCKACTIVITY	Reserved	IDLEMODE	Reserved	SOFTRESET	AUTOIDLE
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-29. MMU_SYSCONFIG Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Write 0's for future compatibility; reads returns 0.
9-8	CLOCKACTIVITY	0	Clock activity during wake-up mode 00 Functional and Interconnect clocks can be switched off.
7-5	Reserved	0	Write 0's for future compatibility; reads returns 0.
4-3	IDLEMODE		Idle Mode
		0	Force-idle. An idle request is acknowledged unconditionally.
		1h	No-idle. An idle request is never acknowledged.
		2h	Smart-idle. Acknowledgment to an idle request is given based on the internal activity of the module.
		3h	Reserved
2	Reserved	0	Write 0's for future compatibility; reads returns 0.
1	SOFTRESET		Software reset. This bit is automatically reset by the hardware. During reads, it always returns 0.
		0	Read: always return 0
		1	Read: never happens
		0	Write: no functional effect
		1	Write: The module is reset
0	AUTOIDLE		Internal interconnect clock gating strategy.
		0	Interconnect clock is free-running.
		1	Automatic interconnect clock gating strategy is applied, based on the interconnect interface activity.

1.6.5.3 MMU_SYSSTATUS

The MMU_SYSSTATUS register is shown in Figure 1-28 and described in Table 1-30.

Figure 1-28. MMU_SYSSTATUS

31	1	0
Reserved		RESETDONE
R-0		R-0

LEGEND: R = Read only; -*n* = value after reset

Table 1-30. MMU_SYSSTATUS Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads returns 0.
0	RESETDONE		Internal reset monitoring.
		0	Internal module reset is on-going.
		1	Reset completed.

1.6.5.4 MMU_IRQSTATUS

The MMU_IRQSTATUS register is shown in Figure 1-29 and described in Table 1-31.

Figure 1-29. MMU_IRQSTATUS

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		MULTIHITFAULT	TABLEWALKFAULT	EMUMISS	TRANSLATIONFAULT	TLBMISS
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-31. MMU_IRQSTATUS Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Write 0's for future compatibility; reads returns 0.
4	MULTIHITFAULT		Error due to multiple matches in the TLB.
		0	Read: MultiHitFault is false.
		1	Read: MultiHitFault is true ("pending").
		0	Write: MultiHitFault status bit is unchanged.
		1	Write: MultiHitFault status bit is reset.
3	TABLEWALKFAULT		Error response received during a Table Walk.
		0	Read: TableWalkFault is false.
		1	Read: TableWalkFault is true ("pending").
		0	Write: TableWalkFault status bit is unchanged.
		1	Write: TableWalkFault status bit is reset.
2	EMUMISS		Unrecoverable TLB miss during debug (hardware TWL disabled).
		0	Read: EMUMiss is false.
		1	Read: EMUMiss is true ("pending").
		0	Write: EMUMiss status bit is unchanged.
		1	Write: EMUMiss status bit is reset.
1	TRANSLATIONFAULT		Invalid descriptor in translation tables (translation fault).
		0	Read: TranslationFault is false.
		1	Read: TranslationFault is true ("pending").
		0	Write: TranslationFault status bit is unchanged.
		1	Write: TranslationFault status bit is reset.
0	TLBMISS		Unrecoverable TLB miss (hardware TWL disabled).
		0	Read: TLBMiss is false.
		1	Read: TLBMiss is true ("pending").
		0	Write: TLBMiss status bit is unchanged.
		1	Write: TLBMiss status bit is reset.

1.6.5.5 MMU_IRQENABLE

The MMU_IRQENABLE register is shown in Figure 1-30 and described in Table 1-32.

Figure 1-30. MMU_IRQENABLE

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		MULTIHITFAULT	TABLEWALKFAULT	EMUMISS	TRANSLATIONFAULT	TLBMISS
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-32. MMU_IRQENABLE Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Write 0's for future compatibility; reads returns 0.
4	MULTIHITFAULT		Error due to multiple matches in the TLB.
		0	MultiHitFault is masked.
		1	MultiHitFault event generates an interrupt if occurs.
3	TABLEWALKFAULT		Error response received during a Table Walk.
		0	TableWalkFault is masked.
		1	TableWalkFault event generates an interrupt if occurs.
2	EMUMISS		Unrecoverable TLB miss during debug (hardware TWL disabled).
		0	EMUMiss interrupt is masked.
		1	EMUMiss event generates an interrupt when it occurs.
1	TRANSLATIONFAULT		Invalid descriptor in translation tables (translation fault).
		0	TranslationFault is masked.
		1	TranslationFault event generates an interrupt if occurs.
0	TLBMISS		Unrecoverable TLB miss (hardware TWL disabled).
		0	TLBMiss interrupt is masked.
		1	TLBMiss event generates an interrupt when if occurs.

1.6.5.6 MMU_WALKING_ST

The MMU_WALKING_ST register is shown in Figure 1-31 and described in Table 1-33.

Figure 1-31. MMU_WALKING_ST

31	1	0
Reserved		TWLRUNNING
R-0		R-0

LEGEND: R = Read only; -*n* = value after reset

Table 1-33. MMU_WALKING_ST Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0.
0	TWLRUNNING		Table Walking Logic is running.
		0	TWL Completed.
		1	TWL Running.

1.6.5.7 MMU_CNTL

The MMU_CNTL register is shown in Figure 1-32 and described in Table 1-34.

Figure 1-32. MMU_CNTL

31	4	3	2	1	0
Reserved		EMUTLBUPDATE	TWLENABLE	MMUENABLE	Reserved
R-0		R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-34. MMU_CNTL Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Write 0's for future compatibility; reads returns 0.
3	EMUTLBUPDATE		Enable TLB update on emulator table walk.
		0	Emulator TLB update is disabled.
		1	Emulator TLB update is enabled.
2	TWLENABLE		Table Walking Logic enable.
		0	TWL is disabled.
		1	TWL is enabled.
1	MMUENABLE		MMU enable.
		0	MMU is disabled.
		1	MMU is enabled.
0	Reserved	0	Write 0's for future compatibility; reads returns 0.

1.6.5.8 MMU_FAULT_AD

The MMU_FAULT_AD register is shown in Figure 1-33 and described in Table 1-35.

Figure 1-33. MMU_FAULT_AD

31	0
FAULTADDRESS	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 1-35. MMU_FAULT_AD Field Descriptions

Bit	Field	Value	Description
31-0	FAULTADDRESS	0-FFFF FFFFh	Virtual address of the access that generated a fault.

1.6.5.9 MMU_TTB

The MMU_TTB register is shown in Figure 1-34 and described in Table 1-36.

Figure 1-34. MMU_TTB

31	7	6	0
TTBADDRESS		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-36. MMU_TTB Field Descriptions

Bit	Field	Value	Description
31-7	TTBADDRESS	0-3FF FFFFh	Translation Table Base Address.
6-0	Reserved	0	Write 0's for future compatibility; reads returns 0.

1.6.5.10 MMU_LOCK

The MMU_LOCK register is shown in Figure 1-35 and described in Table 1-37.

Figure 1-35. MMU_LOCK

31	15	14	10	9	8	4	3	0
Reserved		BASEVALUE		Reserved	CURRENTVICTIM		Reserved	
R-0		R/W-0		R-0	R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-37. MMU_LOCK Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Write 0's for future compatibility; reads returns 0.
14-10	BASEVALUE	0-1Fh	Locked entries base value.
9	Reserved	0	Write 0's for future compatibility; reads returns 0.
8-4	CURRENTVICTIM	0-1Fh	Current entry to be updated either by the TWL or by the SW Write value : TLB entry to be updated by software, Read value : TLB entry that will be updated by table walk logic.
3-0	Reserved	0	Write 0's for future compatibility; reads returns 0.

1.6.5.11 MMU_LD_TLB

The MMU_LD_TLB register is shown in Figure 1-36 and described in Table 1-38.

Figure 1-36. MMU_LD_TLB

31	1	0
Reserved		LDTLBITEM
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-38. MMU_LD_TLB Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Write 0's for future compatibility; reads returns 0.
0	LDTLBITEM		Write (load) data in the TLB.
		0	Read: Always returns 0.
		1	Read: Never happens.
		0	Write: No functional effect.
		1	Write: Load TLB data.

1.6.5.12 MMU_CAM

The MMU_CAM register is shown in Figure 1-37 and described in Table 1-39.

Figure 1-37. MMU_CAM

31	12	11	4	3	2	1	0
VATAG		Reserved		P	V	PAGESIZE	
R/W-0		R-0		R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-39. MMU_CAM Field Descriptions

Bit	Field	Value	Description
31-12	VATAG	0-F FFFFh	Virtual address tag.
11-4	Reserved	0	Write 0's for future compatibility; reads returns 0.
3	P		Preserved bit.
		0	TLB entry may be flushed.
		1	TLB entry is protected against flush.
2	V		Valid bit.
		0	TLB entry is invalid.
		1	TLB entry is valid.
1-0	PAGESIZE		Page size.
		0	Section (1MB)
		1h	Large page (64KB)
		2h	Small page (4KB)
		3h	Supersection (16MB)

1.6.5.13 MMU_RAM

The MMU_RAM register is shown in Figure 1-38 and described in Table 1-40.

Figure 1-38. MMU_RAM

31 12	11 10	9	8 7	6	5 0
PHYSICALADDRESS	Reserved	ENDIANNESS	ELEMENTSIZE	MIXED	Reserved
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-40. MMU_RAM Field Descriptions

Bit	Field	Value	Description
31-12	PHYSICALADDRESS	0-F FFFFh	Physical address of the page.
11-10	Reserved	0	Write 0's for future compatibility; reads returns 0.
9	ENDIANNESS		Endianness of the page.
		0	Little endian.
		1	Big endian.
8-7	ELEMENTSIZE		Element size of the page (8, 16, 32, no translation)
		0	8-bits
		1h	16-bits
		2h	32-bits
		3h	No translation
6	MIXED		Mixed page attribute (use CPU element size).
		0	Use TLB element size.
		1	Use CPU element size.
5-0	Reserved	0	Write 0's for future compatibility; reads returns 0.

1.6.5.14 MMU_GFLUSH

The MMU_GFLUSH register is shown in Figure 1-39 and described in Table 1-41.

Figure 1-39. MMU_GFLUSH

31 1	0
Reserved	GLOBALFLUSH
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-41. MMU_GFLUSH Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Write 0's for future compatibility; reads returns 0.
0	GLOBALFLUSH		Flush all the non-protected TLB entries when set.
		0	Read: always return 0.
		1	Read: never happens.
		0	Write: no functional effect.
		1	Write: flush all the non-protected TLB entries.

1.6.5.15 MMU_FLUSH_ENTRY

The MMU_FLUSH_ENTRY register is shown in Figure 1-40 and described in Table 1-42.

Figure 1-40. MMU_FLUSH_ENTRY

31	1	0
Reserved		FLUSHENTRY
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-42. MMU_FLUSH_ENTRY Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Write 0's for future compatibility; reads returns 0.
0	FLUSHENTRY		Flush the TLB entry pointed by the virtual address (VATag) in MMU_CAM register, even if this entry is set protected.
		0	Read: always return 0.
		1	Read: never happens.
		0	Write: no functional effect.
		1	Write: flush all the TLB entries specified by the CAM register.

1.6.5.16 MMU_READ_CAM

The MMU_READ_CAM register is shown in Figure 1-41 and described in Table 1-43.

Figure 1-41. MMU_READ_CAM

31	12	11	4	3	2	1	0
VATAG		Reserved		P	V	PAGESIZE	
R-0		R-0		R-0	R-0	R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 1-43. MMU_READ_CAM Field Descriptions

Bit	Field	Value	Description
31-12	VATAG	0-F FFFFh	Virtual address tag.
11-4	Reserved	0	Reads return 0.
3	P		Preserved bit.
		0	TLB entry may be flushed.
		1	TLB entry is protected against flush.
2	V		Valid bit.
		0	TLB entry is invalid.
		1	TLB entry is valid.
1-0	PAGESIZE		Page size.
		0	Section (1MB)
		1h	Large page (64KB)
		2h	Small page (4KB)
		3h	Supersection (16MB)

1.6.5.17 MMU_READ_RAM

The MMU_READ_RAM register is shown in Figure 1-42 and described in Table 1-44.

Figure 1-42. MMU_READ_RAM

31 12	11 10	9	8 7	6	5 0
PHYSICALADDRESS	Reserved	ENDIANNESS	ELEMENTSIZE	MIXED	Reserved
R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 1-44. MMU_READ_RAM Field Descriptions

Bit	Field	Value	Description
31-12	PHYSICALADDRESS	0-F FFFFh	Physical address of the page.
11-10	Reserved	0	Reads return 0.
9	ENDIANNESS		Endianness of the page.
		0	Little endian.
		1	Big endian.
8-7	ELEMENTSIZE		Element size of the page (8, 16, 32, no translation)
		0	8-bits
		1h	16-bits
		2h	32-bits
		3h	No translation
6	MIXED		Mixed page attribute (use CPU element size).
		0	Use TLB element size.
		1	Use CPU element size.
5-0	Reserved	0	Reads return 0.

1.6.5.18 MMU_EMU_FAULT_AD

The MMU_EMU_FAULT_AD register is shown in Figure 1-43 and described in Table 1-45.

Figure 1-43. MMU_EMU_FAULT_AD

31	0
EMUFAULTADDRESS	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 1-45. MMU_EMU_FAULT_AD Field Descriptions

Bit	Field	Value	Description
31-0	EMUFAULTADDRESS	0-FFFF FFFFh	Virtual address of the last emulator access that generated a fault.

1.6.5.19 MMU_FAULT_PC

The MMU_FAULT_PC register is shown in Figure 1-44 and described in Table 1-46.

Figure 1-44. MMU_FAULT_PC

31	0
PC	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 1-46. MMU_FAULT_PC Field Descriptions

Bit	Field	Value	Description
31-0	PC	0-FFFF FFFFh	CPU program counter value where cause MMU fault. The address value is captured at MMU_EMU_FAULT_AD EMUFAULTADDRESS bit field. Data-Read-access : corresponding PC. Data-write-access : not perfect accuracy due to Posted-write.

1.7 SGX530 Graphics Subsystem

This section describes the 2D/3D graphics accelerator (SGX) for the device.

NOTE: This section contains materials that are ©2003-2007 Imagination Technologies Limited.

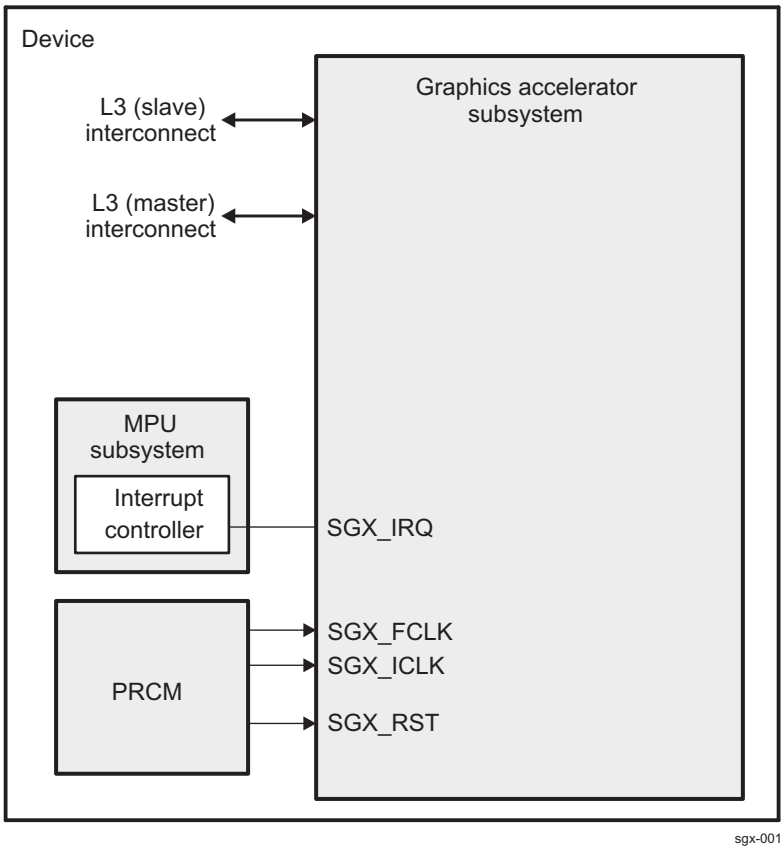
PowerVR® and USSE™ are registered trademarks or trademarks of Imagination Technologies Limited.

1.7.1 SGX Overview

The 2D/3D graphics accelerator (SGX) subsystem accelerates 2-dimensional (2D) and 3-dimensional (3D) graphics applications. The SGX subsystem is a Texas Instruments instantiation of the PowerVR® SGX530 core from Imagination Technologies Limited. SGX is a new generation of programmable PowerVR graphic cores. The PowerVR SGX530 v1.2.5 architecture is scalable and can target all market segments from mainstream mobile devices to high-end desktop graphics. Targeted applications include feature phone, PDA, and hand-held games.

Figure 1-45 shows the SGX subsystem in the device.

Figure 1-45. Graphics Accelerator Highlight



The SGX graphics accelerator can simultaneously process various multimedia data types:

- Pixel data
- Vertex data
- Video data
- General-purpose processing

This is achieved through a multithreaded architecture using two levels of scheduling and data partitioning enabling zero-overhead task switching.

The SGX subsystem is connected to the L3 interconnect by a 128-bit master and a 32-bit slave interface.

1.7.1.1 PowerVR SGX Main Features

- 2D graphics, 3D graphics, vector graphics, and programming support for GP-GPU functions
- Tile-based architecture
- Universal scalable shader engine (USSE) – multithreaded engine incorporating pixel and vertex shader functionality
- Advanced shader feature set – in excess of Microsoft VS3.0, PS3.0, and OpenGL2.0
- Industry-standard API support – Direct3D Mobile, OpenGL ES 1.1 and 2.0, OpenVG v1.1
- Fine-grained task switching, load balancing, and power management
- Advanced geometry direct memory access (DMA) driven operation for minimum CPU interaction
- Programmable high-quality image anti-aliasing
- PowerVR SGX core MMU for address translation from the core virtual address to the external physical address (up to 4GB address range)
- Fully virtualized memory addressing for OS operation in a unified memory architecture
- Advanced and standard 2D operations [for example, vector graphics, BLTs (block level transfers), ROPs (raster operations)]
- 32K stride support

1.7.1.2 SGX 3D Features

- Deferred pixel shading
- On-chip tile floating point depth buffer
- 8-bit stencil with on-chip tile stencil buffer
- 8 parallel depth/stencil tests per clock
- Scissor test
- Texture support:
  - Cube map
  - Projected textures
  - 2D textures
  - Nonsquare textures
- Texture formats:
  - RGBA 8888, 565, 1555
  - Monochromatic 8, 16, 16f, 32f, 32int
  - Dual channel, 8:8, 16:16, 16f:16f
  - Compressed textures PVR-TC1, PVR-TC2, ETC1
  - Programmable support for all YUV formats
- Resolution support:
  - Frame buffer maximum size = 2048 × 2048
  - Texture maximum size = 2048 × 2048

- Texture filtering:
  - Bilinear, trilinear, anisotropic
  - Independent minimum and maximum control
- Antialiasing:
  - 4× multisampling
  - Up to 16× full scene anti-aliasing
  - Programmable sample positions
- Indexed primitive list support
  - Bus mastered
- Programmable vertex DMA
- Render to texture:
  - Including twiddled formats
  - Auto MipMap generation
- Multiple on-chip render targets (MRT).

  Note: Performance is limited when the on-chip memory is not available.

1.7.1.3 Universal Scalable Shader Engine (USSE) – Key Features

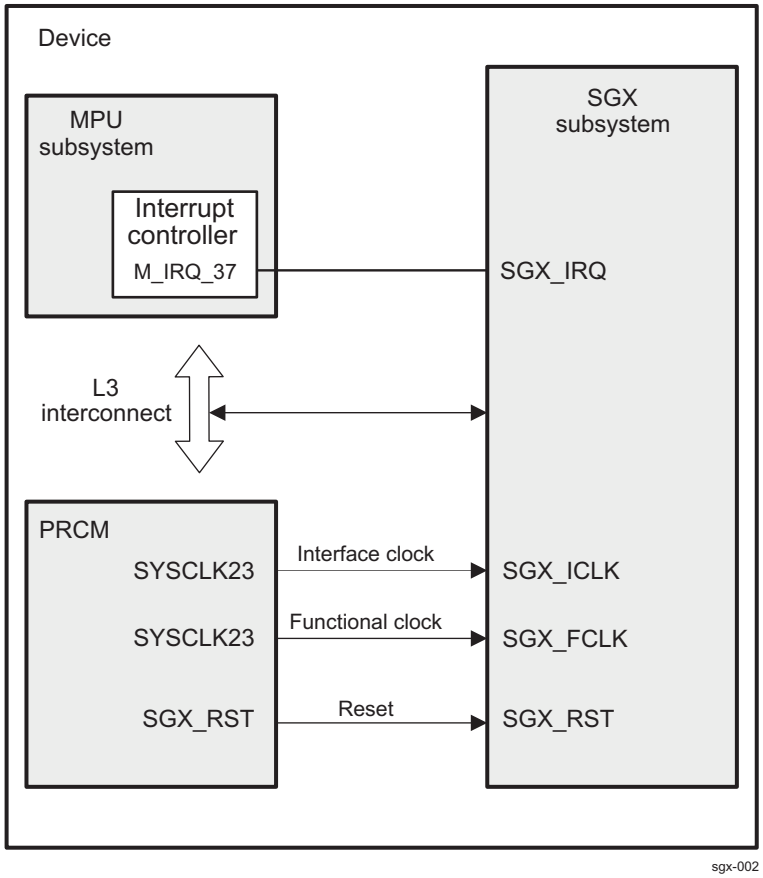
The USSE is the engine core of the PowerVR SGX architecture and supports a broad range of instructions.

- Single programming model:
  - Multithreaded with 16 simultaneous execution threads and up to 64 simultaneous data instances
  - Zero-cost swapping in, and out, of threads
  - Cached program execution model
  - Dedicated pixel processing instructions
  - Dedicated video encode/decode instructions
- SIMD execution unit supporting operations in:
  - 32-bit IEEE float
  - 2-way 16-bit fixed point
  - 4-way 8-bit integer
  - 32-bit bit-wise (logical only)
- Static and dynamic flow control:
  - Subroutine calls
  - Loops
  - Conditional branches
  - Zero-cost instruction predication
- Procedural geometry:
  - Allows generation of primitives
  - Effective geometry compression
  - High-order surface support
- External data access:
  - Permits reads from main memory using cache
  - Permits writes to main memory
  - Data fence facility
  - Dependent texture reads

1.7.2 SGX Integration

Figure 1-46 highlights the SGX subsystem integration in the device.

Figure 1-46. SGX Subsystem Integration



1.7.2.1 Clocking, Reset, and Power-Management Scheme

1.7.2.1.1 Clocks

The SGX subsystem operates from two clocks: an interface clock (SGX_ICLK) and a functional clock (SYSCLK23). The power, reset, and clock management (PRCM) module generates and distributes both clocks inside the device.

Table 1-47. Clock Descriptions

Signal Name	I/O[1]	Description
SYSCLK23	I	Functional clock → Functional clock domain
SGX_ICLK	I	Interface clock → Interface clock domain

(1) I =Input; O=Output

- SGX_ICLK interface clock manages the data transfer on the L3 master and slave ports.
- SGX_FCLK is the functional clock and is used inside the SGX subsystem to generate SGX 2D and 3D domain clock signals.

To Enable the SGX Functional and Interface Clock:

Program the PRCM.CM_SGX_CLKSTCTRL.CTLKTRCTRL to 0x2
Program the PRCM.CM_SGX_SGX_CLKCTRL.MODULEMODE to 0x2
Poll for the PRCM.CM_SGX_SGX_CLKCTRL IDLEST to be 0x0

To Disable the SGX Functional and Interface Clock:

Program the PRCM.CM_SGX_SGX_CLKCTRL.MODULEMODE to 0x0
Program the PRCM.CM_SGX_CLKSTCTRL.CTLKTRCTRL to 0x0

The SGX_ICLK and SGX_FLCK are sourced from SYSCLK23.

1.7.2.1.2 Resets

The SGX subsystem has its own reset domain. Global reset of the SGX is performed by activating the SGX_RST signal in the SGX_RST domain. Software controls the release of SGX_RST using the PRCM.RM_SGX_RSTCTRL[0] SGX_RST bit.

NOTE: The APIs delivered with the SGX provide a software reset functionally equivalent to a hardware reset.

1.7.2.1.3 Power Management

The SGX subsystem has its own power domain (SGX power domain). See the *Power, Reset, and Clock Management (PRCM) Module* chapter for additional information about the SGX power domain.

The SGX subsystem receives two clock signals from the PRCM module. The functional clock is used inside the SGX to generate clock signals to the multiple internal module SGX clock domains. The division ratio depends on the PRCM registers setting. For more information, see the *Power, Reset, and Clock Management (PRCM) Module* chapter.

Three power-management modes are defined:

- Deep power sleep (All clocks are gated.)
- Idle (2D and 3D clocks are gated.)
- 3D (No clock is gated.)

These modes are fully managed inside the SGX and are handled by the related API delivered with the module.

The SGX handles the automatic clock gating performed on the multiple internal module clock domains.

1.7.2.2 Hardware Requests

1.7.2.2.1 Interrupt Request

The SGX subsystem can generate one interrupt (SGX_IRQ) to the MPU subsystem interrupt controller mapped on M_IRQ_37.

1.7.3 SGX Functional Description

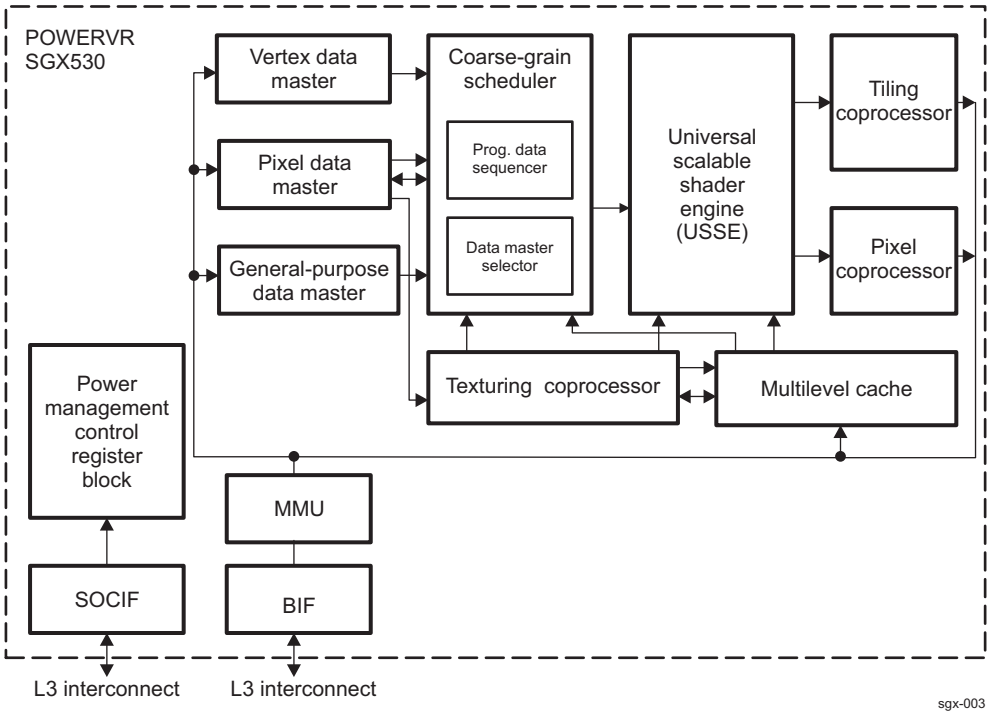
1.7.3.1 SGX Block Diagram

The architecture uses programmable and hard coded pipelines to perform various processing tasks required in 2D, 3D, and video processing. The SGX architecture comprises the following elements:

- Coarse grain scheduler
  - Programmable data sequencer (PDS)
  - Data master selector (DMS)
- Vertex data master (VDM)
- Pixel data master (PDM)
- General-purpose data master
- USSE
- Tiling coprocessor
- Pixel coprocessor
- Texturing coprocessor
- Multilevel cache

Figure 1-47 shows a block diagram of the SGX cores.

Figure 1-47. SGX Block Diagram



1.7.3.2 SGX Elements Description

The coarse grain scheduler (CGS) is the main system controller for the PowerVR SGX architecture. It consists of two stages, the DMS and the PDS. The DMS processes requests from the data masters and determines which tasks can be executed given the resource requirements. The PDS then controls the loading and processing of data on the USSE.

There are three data masters in the SGX core:

- The VDM is the initiator of transform and lighting processing within the system. The VDM reads an input control stream, which contains triangle index data and state data. The state data indicates the PDS program, size of the vertices, and the amount of USSE output buffer resource available to the VDM. The triangle data is parsed to determine unique indices that must be processed by the USSE. These are grouped together according to the configuration provided by the driver and presented to the DMS.
- The PDM is the initiator of rasterization processing within the system. Each pixel pipeline processes pixels for a different half of a given tile, which allows for optimum efficiency within each pipe due to locality of data. It determines the amount of resource required within the USSE for each task. It merges this with the state address and issues a request to the DMS for execution on the USSE.
- The general-purpose data master responds to events within the system (such as end of a pass of triangles from the ISP, end of a tile from the ISP, end of render, or parameter stream breakpoint event). Each event causes either an interrupt to the host or synchronized execution of a program on the PDS. The program may, or may not cause a subsequent task to be executed on the USSE.

The USSE is a user-programmable processing unit. Although general in nature, its instructions and features are optimized for three types of task: processing vertices (vertex shading), processing pixels (pixel shading), and video/imaging processing.

The multilevel cache is a 2-level cache consisting of two modules: the main cache and the mux/arbiter/demux/decompression unit (MADD). The MADD is a wrapper around the main cache module designed to manage and format requests to and from the cache, as well as providing Level 0 caching for texture and USSE requests. The MADD can accept requests from the PDS, USSE, and texture address generator modules. Arbitration, as well as any required texture decompression, are performed between the three data streams.

The texturing coprocessor performs texture address generation and formatting of texture data. It receives requests from either the iterators or USSE modules and translates these into requests in the multilevel cache. Data returned from the cache are then formatted according to the texture format selected, and sent to the USSE for pixel-shading operations.

To process pixels in a tiled manner, the screen is divided into tiles and arranged as groups of tiles by the tiling coprocessor. An inherent advantage of tiling architecture is that a large amount of vertex data can be rejected at this stage, thus reducing the memory storage requirements and the amount of pixel processing to be performed.

The pixel coprocessor is the final stage of the pixel-processing pipeline and controls the format of the final pixel data sent to the memory. It supplies the USSE with an address into the output buffer and then USSE returns the relevant pixel data. The address order is determined by the frame buffer mode. The pixel coprocessor contains a dithering and packing function.

1.7.4 SGX Registers

CAUTION

All SGX registers are limited to 32-bit data accesses, 8- and 16-bit accesses are not allowed because they can corrupt register content.

Table 1-48 provides a summary of the SGX registers. For the base address of these registers. see Table 1-11.

Table 1-48. SGX Registers

Offset	Acronym	Register Description	Section
FE00h	OCP_REVISION	OCP Revision Register	Section 1.7.4.1
FE04h	OCP_HWINFO	Hardware Implementation Information Register	Section 1.7.4.2
FE10h	OCP_SYSCONFIG	System Configuration Register	Section 1.7.4.3
FE24h	OCP_IRQSTATUS_RAW_0	Raw IRQ 0 Status Register	Section 1.7.4.4
FE28h	OCP_IRQSTATUS_RAW_1	Raw IRQ 1 Status Register	Section 1.7.4.5
FE2Ch	OCP_IRQSTATUS_RAW_2	Raw IRQ 2 Status Register	Section 1.7.4.6
FE30h	OCP_IRQSTATUS_0	Interrupt 0 Status Event Register	Section 1.7.4.7
FE34h	OCP_IRQSTATUS_1	Interrupt 1 Status Event Register	Section 1.7.4.8
FE38h	OCP_IRQSTATUS_2	Interrupt 2 Status Event Register	Section 1.7.4.9
FE3Ch	OCP_IRQENABLE_SET_0	Enable Interrupt 0 Register	Section 1.7.4.10
FE40h	OCP_IRQENABLE_SET_1	Enable Interrupt 1 Register	Section 1.7.4.11
FE44h	OCP_IRQENABLE_SET_2	Enable Interrupt 2 Register	Section 1.7.4.12
FE48h	OCP_IRQENABLE_CLR_0	Disable Interrupt 0 Register	Section 1.7.4.13
FE4Ch	OCP_IRQENABLE_CLR_1	Disable Interrupt 1 Register	Section 1.7.4.14
FE50h	OCP_IRQENABLE_CLR_2	Disable Interrupt 2 Register	Section 1.7.4.15
FF00h	OCP_PAGE_CONFIG	Configure Memory Page Register	Section 1.7.4.16
FF04h	OCP_INTERRUPT_EVENT	Interrupt Events Register	Section 1.7.4.17
FF08h	OCP_DEBUG_CONFIG	Configuration of Debug Modes Register	Section 1.7.4.18
FF0Ch	OCP_DEBUG_STATUS	Debug Status Register	Section 1.7.4.19

1.7.4.1 OCP Revision Register (OCP_REVISION)

The OCP Revision Register (OCP_REVISION) is shown in Figure 1-48 and described in Table 1-49.

Figure 1-48. OCP Revision Register (OCP_REVISION)

31	0
REVISIONID	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 1-49. OCP Revision Register (OCP_REVISION) Field Descriptions

Bit	Field	Value	Description
31-0	REVISIONID	0-FFFF FFFFh	Revision value.

1.7.4.2 Hardware Implementation Information Register (OCP_HWINFO)

The Hardware Implementation Information Register (OCP_HWINFO) is shown in Figure 1-49 and described inTable 1-50.

Figure 1-49. Hardware Implementation Information Register (OCP_HWINFO)

31	3	2	1	0
Reserved		MEM_BUS_WIDTH	SYS_BUS_WIDTH	
R-0		R-0	R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 1-50. Hardware Implementation Information Register (OCP_HWINFO) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved.
2	MEM_BUS_WIDTH		Memory bus width.
		0	Memory bus width is 64 bits.
		1	Memory bus width is 128 bits
1-0	SYS_BUS_WIDTH		System bus width.
		0	System bus width is 32 bits.
		1h	System bus width is 64 bits.
		2h	System bus width is 128 bits.
		3h	Reserved.

1.7.4.3 System Configuration Register (OCP_SYSCONFIG)

The System Configuration Register (OCP_SYSCONFIG) is shown in Figure 1-50 and described in Table 1-51.

Figure 1-50. System Configuration Register (OCP_SYSCONFIG)

31 6	5 4	3 2	1 0
Reserved	STANDBY_MODE	IDLE_MODE	Reserved
R-0	R/W-2h	R/W-2h	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-51. System Configuration Register (OCP_SYSCONFIG) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved.
5-4	STANDBY_MODE		Clock standby mode.
		0	Force Standby mode.
		1h	No Standby mode.
		2h	Smart Standby mode.
		3h	Smart Standby mode.
3-2	IDLE_MODE		Clock Idle mode.
		0	Force Idle mode.
		1h	No idle mode.
		2h	Smart Idle mode.
		3h	Smart Idle mode.
1-0	Reserved	0	Reserved.

1.7.4.4 Raw IRQ 0 Status Register (OCP_IRQSTATUS_RAW_0)

The Raw IRQ 0 Status Register (OCP_IRQSTATUS_RAW_0) is shown in Figure 1-51 and described in Table 1-52.

Figure 1-51. Raw IRQ 0 Status Register (OCP_IRQSTATUS_RAW_0)

31	1	0
Reserved		INIT_MINTERRUPT_RAW
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-52. Raw IRQ 0 Status Register (OCP_IRQSTATUS_RAW_0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	INIT_MINTERRUPT_RAW		Interrupt 0 - master port raw event.
		0	Read: No event pending.
		1	Read: Event pending.
		0	Write: No action.
		1	Write: Set event (Used for debug).

1.7.4.5 Raw IRQ 1 Status Register (OCP_IRQSTATUS_RAW_1)

The Raw IRQ 1 Status Register (OCP_IRQSTATUS_RAW_1) is shown in Figure 1-52 and described in Table 1-53.

Figure 1-52. Raw IRQ 1 Status Register (OCP_IRQSTATUS_RAW_1)

31	1	0
Reserved		TARGET_SINTERRUPT_RAW
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-53. Raw IRQ 1 Status Register (OCP_IRQSTATUS_RAW_1) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	TARGET_SINTERRUPT_RAW		Interrupt 1- slave port raw event.
		0	Read: No event pending.
		1	Read: Event pending.
		0	Write: No action.
		1	Write: Set event (Used for debug).

1.7.4.6 Raw IRQ 2 Status Register (OCP_IRQSTATUS_RAW_2)

The Raw IRQ 2 Status Register (OCP_IRQSTATUS_RAW_2) is shown in Figure 1-53 and described in Table 1-54.

Figure 1-53. Raw IRQ 2 Status Register (OCP_IRQSTATUS_RAW_2)

31	1	0
Reserved		THALIA_IRQ_RAW
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-54. Raw IRQ 2 Status Register (OCP_IRQSTATUS_RAW_2) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	THALIA_IRQ_RAW		Interrupt 2 - Thalia raw event.
		0	Read: No event pending.
		1	Read: Event pending.
		0	Write: No action.
		1	Write: Set event (Used for debug).

1.7.4.7 Interrupt 0 Status Event Register (OCP_IRQSTATUS_0)

The Interrupt 0 Status Event Register (OCP_IRQSTATUS_0) is shown in Figure 1-54 and described in Table 1-55.

Figure 1-54. Interrupt 0 Status Event Register (OCP_IRQSTATUS_0)

31	1	0
Reserved		INIT_MINTERRUPT_STATUS
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-55. Interrupt 0 Status Event Register (OCP_IRQSTATUS_0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	INIT_MINTERRUPT_STATUS		Interrupt 0 - Master port status event.
		0	Read: No event pending.
		1	Read: Event pending and interrupt enabled.
		0	Write: No action.
		1	Write: Clear event.

1.7.4.8 Interrupt 1 Status Event Register (OCP_IRQSTATUS_1)

The Interrupt 1 Status Event Register (OCP_IRQSTATUS_1) is shown in Figure 1-55 and described in Table 1-56.

Figure 1-55. Interrupt 1 Status Event Register (OCP_IRQSTATUS_1)

31	1	0
Reserved		TARGET_SINTERRUPT_ STATUS
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-56. Interrupt 1 Status Event Register (OCP_IRQSTATUS_1) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	TARGET_SINTERRUPT_STATUS		Interrupt 1 - slave port status event.
		0	Read: No event pending.
		1	Read: Event pending and interrupt enabled.
		0	Write: No action.
		1	Write: Clear event.

1.7.4.9 Interrupt 2 Status Event Register (OCP_IRQSTATUS_2)

The Interrupt 2 Status Event Register (OCP_IRQSTATUS_2) is shown in Figure 1-56 and described in Table 1-57.

Figure 1-56. Interrupt 2 Status Event Register (OCP_IRQSTATUS_2)

31	1	0
Reserved		TARGET_SINTERRUPT_ STATUS
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-57. Interrupt 2 Status Event Register (OCP_IRQSTATUS_2) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	TARGET_SINTERRUPT_STATUS		Interrupt 1 - slave port status event.
		0	Read: No event pending.
		1	Read: Event pending and interrupt enabled.
		0	Write: No action.
		1	Write: Clear event.

1.7.4.10 Enable Interrupt 0 Register (OCP_IRQENABLE_SET_0)

The Enable Interrupt 0 Register (OCP_IRQENABLE_SET_0) is shown in Figure 1-57 and described in Table 1-58.

Figure 1-57. Enable Interrupt 0 Register (OCP_IRQENABLE_SET_0)

31	1	0
Reserved		INIT_MINTERRUPT_ENABLE
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-58. Enable Interrupt 0 Register (OCP_IRQENABLE_SET_0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	INIT_MINTERRUPT_ENABLE		Enable interrupt 0 - master port.
		0	Read: Interrupt is enabled.
		1	Read: Interrupt is disabled.
		0	Write: No action.
		1	Write: Enable interrupt.

1.7.4.11 Enable Interrupt 1 Register (OCP_IRQENABLE_SET_1)

The Enable Interrupt 1 Register (OCP_IRQENABLE_SET_1) is shown in Figure 1-58 and described in Table 1-59.

Figure 1-58. Enable Interrupt 1 Register (OCP_IRQENABLE_SET_1)

31	1	0
Reserved		TARGET_SINTERRUPT_ ENABLE
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-59. Enable Interrupt 1 Register (OCP_IRQENABLE_SET_1) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	TARGET_SINTERRUPT_ENABLE		Enable interrupt 1 - slave port interrupt.
		0	Read: Interrupt is enabled.
		1	Read: Interrupt is disabled.
		0	Write: No action.
		1	Write: Enable interrupt.

1.7.4.12 Enable Interrupt 2 Register (OCP_IRQENABLE_SET_2)

The Enable Interrupt 2 Register (OCP_IRQENABLE_SET_2) is shown in Figure 1-59 and described in Table 1-60.

Figure 1-59. Enable Interrupt 2 Register (OCP_IRQENABLE_SET_2)

31	1	0
Reserved		THALIA_IRQ_ENABLE
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-60. Enable Interrupt 2 Register (OCP_IRQENABLE_SET_2) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	THALIA_IRQ_ENABLE		Enable interrupt 2 - Thalia (core) interrupt.
		0	Read: Interrupt is enabled.
		1	Read: Interrupt is disabled.
		0	Write: No action.
		1	Write: Enable interrupt.

1.7.4.13 Disable Interrupt 0 Register (OCP_IRQENABLE_CLR_0)

The Disable Interrupt 0 Register (OCP_IRQENABLE_CLR_0) is shown in Figure 1-60 and described in Table 1-61.

Figure 1-60. Disable Interrupt 0 Register (OCP_IRQENABLE_CLR_0)

31	1	0
Reserved		INIT_MINTERRUPT_ DISABLE
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-61. Disable Interrupt 0 Register (OCP_IRQENABLE_CLR_0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	INIT_MINTERRUPT_DISABLE		Disable interrupt 0 - master port.
		0	Read: Interrupt is enabled.
		1	Read: Interrupt is disabled.
		0	Write: No action.
		1	Write: Disable interrupt.

1.7.4.14 Disable Interrupt 1 Register (OCP_IRQENABLE_CLR_1)

The Disable Interrupt 1 Register (OCP_IRQENABLE_CLR_1) is shown in Figure 1-61 and described in Table 1-62.

Figure 1-61. Disable Interrupt 1 Register (OCP_IRQENABLE_CLR_1)

31	1	0
Reserved		TARGET_SINTERRUPT_DISABLE
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-62. Disable Interrupt 1 Register (OCP_IRQENABLE_CLR_1) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	TARGET_SINTERRUPT_DISABLE		Disable interrupt 1 - slave port.
		0	Read: Interrupt is enabled.
		1	Read: Interrupt is disabled.
		0	Write: No action.
		1	Write: Disable interrupt.

1.7.4.15 Disable Interrupt 2 Register (OCP_IRQENABLE_CLR_2)

The Disable Interrupt 2 Register (OCP_IRQENABLE_CLR_2) is shown in Figure 1-62 and described in Table 1-63.

Figure 1-62. Disable Interrupt 2 Register (OCP_IRQENABLE_CLR_2)

31	1	0
Reserved		THALIA_IRQ_DISABLE
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-63. Disable Interrupt 2 Register (OCP_IRQENABLE_CLR_2) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	THALIA_IRQ_DISABLE		Disable interrupt 2 - Thalia (core) interrupt.
		0	Read: Interrupt is enabled.
		1	Read: Interrupt is disabled.
		0	Write: No action.
		1	Write: Disable interrupt.

1.7.4.16 Configure Memory Page Register (OCP_PAGE_CONFIG)

The Configure Memory Page Register (OCP_PAGE_CONFIG) is shown in Figure 1-62 and described in Table 1-63.

Figure 1-63. Configure Memory Page Register (OCP_PAGE_CONFIG)

31	5	4	3	2	1	0
Reserved		OCP_PAGE_SIZE		MEM_PAGE_CHECK_EN	MEM_PAGE_SIZE	
R-0		R/W-2h		R/W-1h	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-64. Configure Memory Page Register (OCP_PAGE_CONFIG) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved.
4-3	OCP_PAGE_SIZE		Defines the page size on OCP memory interface.
		0	Page size is 4 KB.
		1h	Page size is 2 KB.
		2h	Page size is 1 KB.
		3h	Page size is 520 B.
2	MEM_PAGE_CHECK_EN		Enable page boundary checking.
		0	Page boundary checking is disabled.
		1	Page boundary checking is enabled.
1-0	MEM_PAGE_SIZE		Defines the page size on internal memory interface.
		0	Page size is 4 KB.
		1h	Page size is 2 KB.
		2h	Page size is 1 KB.
		3h	Page size is 520 B.

1.7.4.17 Interrupt Events Register (OCP_INTERRUPT_EVENT)

The Interrupt Events Register (OCP_INTERRUPT_EVENT) is shown in Figure 1-64 and described in Table 1-65.

Figure 1-64. Interrupt Events Register (OCP_INTERRUPT_EVENT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved					TARGET_INVALID_OCP_CMD	TARGET_CMD_FIFO_FULL	TARGET_RESP_FIFO_FULL
R-0					R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved		INIT_MEM_REQ_FIFO_OVERRUN	INIT_READ_TAG_FIFO_OVERRUN	INIT_PAGE_CROSS_ERROR	INIT_RESP_ERROR	INIT_RESP_UNUSED_TAG	INIT_RESP_UNEXPECTED
		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-65. Interrupt Events Register (OCP_INTERRUPT_EVENT) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved.
10	TARGET_INVALID_OCP_CMD		Invalid command from OCP
		0	Read: No event pending.
		1	Read: Event pending.
		0	Write: Clear the event.
		1	Write: Set event and interrupt if enabled (debug only).
9	TARGET_CMD_FIFO_FULL		Command FIFO full.
		0	Read: No event pending.
		1	Read: Event pending.
		0	Write: Clear the event.
		1	Write: Set event and interrupt if enabled (debug only).
7-6	Reserved	0	Reserved.
5	INIT_MEM_REQ_FIFO_OVERRUN		Memory request FIFO overrun.
		0	Read: No event pending.
		1	Read: Event pending.
		0	Write: Clear the event.
		1	Write: Set event and interrupt if enabled (debug only).
4	INIT_READ_TAG_FIFO_OVERRUN		Read tag FIFO overrun.
		0	Read: No event pending.
		1	Read: Event pending.
		0	Write: Clear the event.
		1	Write: Set event and interrupt if enabled (debug only).
3	INIT_PAGE_CROSS_ERROR		Memory page had been crossed during a burst.
		0	Read: No event pending.
		1	Read: Event pending.
		0	Write: Clear the event.
		1	Write: Set event and interrupt if enabled (debug only).

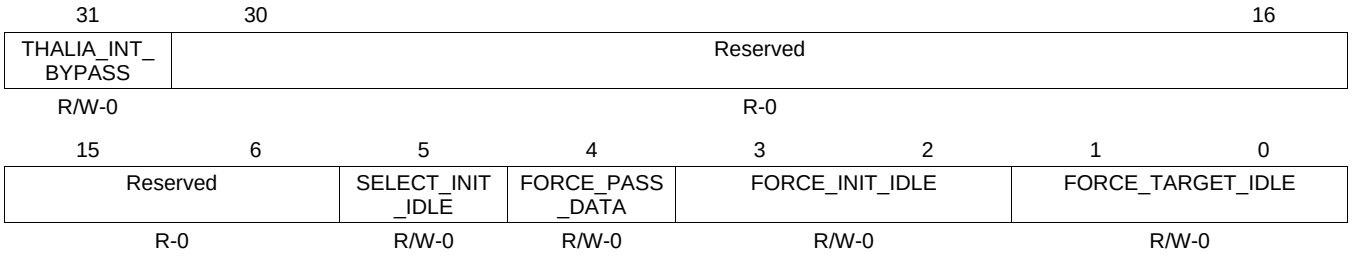
Table 1-65. Interrupt Events Register (OCP_INTERRUPT_EVENT) Field Descriptions (continued)

Bit	Field	Value	Description
2	INIT_RESP_ERROR		Receiving error response.
		0	Read: No event pending.
		1	Read: Event pending.
		0	Write: Clear the event.
		1	Write: Set event and interrupt if enabled (debug only).
1	INIT_RESP_UNUSED_TAG		Receiving response on an unused tag.
		0	Read: No event pending.
		1	Read: Event pending.
		0	Write: Clear the event.
		1	Write: Set event and interrupt if enabled (debug only).
0	INIT_RESP_UNEXPECTED		Receiving response when not expected.
		0	Read: No event pending.
		1	Read: Event pending.
		0	Write: Clear the event.
		1	Write: Set event and interrupt if enabled (debug only).

1.7.4.18 Configuration of Debug Modes Register (OCP_DEBUG_CONFIG)

The Configuration of Debug Modes Register (OCP_DEBUG_CONFIG) is shown in Figure 1-65 and described in Table 1-66.

Figure 1-65. Configuration of Debug Modes Register (OCP_DEBUG_CONFIG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-66. Configuration of Debug Modes Register (OCP_DEBUG_CONFIG) Field Descriptions

Bit	Field	Value	Description
31	THALIA_INT_BYPASS		Bypass OCP IPG interrupt logic.
		0	Do not bypass.
		1	Bypass core interrupt to IO pin, ie disregard the interrupt enable setting in IPG register.
30-6	Reserved	0	Reserved.
5	SELECT_INIT_IDLE		To select which idle the disconnect protocol should act on 0.
		0	Whole SGX Idle.
		1	OCP initiator idle only
4	FORCE_PASS_DATA		Forces the initiator to pass data independent of disconnect protocol.
		0	Normal mode. Do not force.
		1	Never fence request to OCP.
3-2	FORCE_INIT_IDLE		Forces the OCP master port to Idle.
		0	Normal mode. No force.
		1h	Force port to be always Idle.
		2h	Forces target port to never be in Idle mode.
		3h	Normal mode. No force.
1-0	FORCE_TARGET_IDLE		Forces the OCP target port to Idle.
		0	Normal mode. No force.
		1h	Force port to be always Idle.
		2h	Forces target port to never be in Idle mode.
		3h	Normal mode. No force.

1.7.4.19 Debug Status Register (OCP_DEBUG_STATUS)

The Debug Status Register (OCP_DEBUG_STATUS) is shown in Figure 1-66 and described in Table 1-67.

Figure 1-66. Debug Status Register (OCP_DEBUG_STATUS)

31	30	29	28	27	26	25	24
CMD_DEBUG_STATE	CMD_RESP_DEBUG_STATE	TARGET_IDLE	RESP_FIFO_FULL	CMD_FIFO_FULL	RESP_ERROR	WHICH_TARGET_REGISTER	
R/W-0	R/W-0	R-0	R-0	R-0	R-0	R/W-0	

23	21	20	18	17	16
WHICH_TARGET_REGISTER		TARGET_CMD_OUT		INIT_MSTANDBY	INIT_MWAIT
R/W-0		R-0		R-0	R-0

15	14	13	12	11	10	9	8
INIT_MDISCREQ		INIT_MDISCACK	INIT_SCONNECT2	INIT_SCONNECT1	INIT_SCONNECT0	INIT_MCONNECT	
R-0		R/W-0	R-0	R-0	R-0	R-0	

7	6	5	4	3	2	1	0
TARGET_SIDLEACK		TARGET_SDISCACK		TARGET_SIDLEREQ	TARGET_SCONNECT	TARGET_MCONNECT	
R-0		R-0		R-0	R-0		

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-67. Debug Status Register (OCP_DEBUG_STATUS) Field Descriptions

Bit	Field	Value	Description
31	CMD_DEBUG_STATE		Target command state machine
		0	Idle.
		1	Accept command.
30	CMD_RESP_DEBUG_STATE		Target response state machine.
		0	Send accept.
		1	Wait accept.
29	TARGET_IDLE	0-1	Target idle.
28	RESP_FIFO_FULL	0-1	Target response FIFO full.
27	CMD_FIFO_FULL	0-1	Target command FIFO full.
26	RESP_ERROR	0-1	Respond to OCP with error, which could be caused by either address misalignment or invalid byte enable.
25-21	WHICH_TARGET_REGISTER	0-1Fh	Indicates which OCP target registers to read.
20-18	TARGET_CMD_OUT		Command received from OCP.
		0	Command WRSYS received.
		1h	Command RDSYS received.
		2h	Command WR_ERROR received.
		3h	Command RD_ERROR received.
		4h	Command CHK_WRADDR_PAGE received. Not used.
		5h	Command CHK_RDADDR_PAGE received. Not used.
		6h	Command TARGET_REG_WRITE received.
		7h	Command TARGET_REG_READ received.
17	INIT_MSTANDBY	1-0	Status of init_MStandby signal.
16	INIT_MWAIT	1-0	Status of init_MWait signal.

Table 1-67. Debug Status Register (OCP_DEBUG_STATUS) Field Descriptions (continued)

Bit	Field	Value	Description
15-14	INIT_MDISCREQ		Disconnect status of the OCP interface.
		0	State is FUNCT.
		1h	State is SLEEP TRANS
		2h	Reserved.
		3h	State is IDLE.
13	INIT_MDISCACK		Memory request FIFO full.
		0	Read: No event pending.
		1	Read: Event pending.
		0	Write: Clear the event.
		1	Write: Set the event and interrupt if enabled (debug only)
12	INIT_SCONNECT2		Defines whether to wait in M_WAIT state for MConnect FSM.
		0	Skip M_WAIT state.
		1	Wait in M_WAIT state.
11	INIT_SCONNECT1		Defines the busy-ness state of the slave.
		0	Slave is drained.
		1	Slave is loaded.
10	INIT_SCONNECT0		Disconnect from slave.
		0	Disconnect request from slave.
		1	Connect request from slave.
9-8	INIT_MCONNECT		Initiator MConnect state.
		0	State is M_OFF.
		1h	State is M_WAIT.
		2h	State is M_DISC.
		3h	State is M_CON.
7-6	TARGET_SIDLEACK		Acknowledge the SIdleAck state machine.
		0	State is FUNCT.
		1h	State is SLEEP TRANS.
		2h	Reserved.
		3h	State is IDLE.
5-4	TARGET_SDISCACK		Acknowledge the SDiscAck state machine.
		0	State is FUNCT.
		1h	State is SLEEP TRANS.
		2h	Reserved.
		3h	State is IDLE.
3	TARGET_SIDLEREQ		Request the target to go idle.
		0	Do not go idle, or go active.
		1	Go idle.
2	TARGET_SCONNECT		Target SConnect state.
		0	Disconnect interface.
		1	Connect OCP interface.
1-0	TARGET_MCONNECT		Target MConnect state.
		0	Target is in M_OFF state.
		1h	Target is in M_WAIT disconnect state.
		2h	Target is in M_DISC state.
		3h	Target is in M_CON state.

1.8 Device Interrupts

1.8.1 Interrupt Requests to Cortex-A8 MPU INTC

Table 1-68 shows the Cortex-A8 MPU INTC interrupt mapping.

Table 1-68. Cortex-A8 MPU INTC Interrupt Mapping

MPUSS Interrupt Number	Event	Interrupt Source	Description
0	EMUINT	MPUSS Internal	MPU Emulation(1)
1	COMMTX	MPUSS Internal	MPU Emulation(1)
2	COMMRX	MPUSS Internal	MPU Emulation(1)
3	BENCH	MPUSS Internal	MPU Emulation(1)
4	ELM_IRQ	ELM	Error Location process completion
5	Reserved	Reserved	
6	Reserved	Reserved	
7	NMI	External Pin	NMI Interrupt
8	Reserved	Reserved	
9	L3DEBUG	L3	L3 Debug Error Interrupt
10	L3APPINT	L3	L3 Application Error Interrupt
11	Reserved	Reserved	
12	EDMA3CC_INT0	EDMA TPCC (EDMA3CC)	EDMA3CC Region 0 DMA completion Interrupt
13	EDMA3CC_MPINT	EDMA3CC	EDMA3CC memory protection error Interrupt
14	EDMA3CC_ERRINT	EDMA3CC	EDMA3CC error Interrupt
15	Reserved	Reserved	
16	SATAINT	SATA	SATA Module interrupt
17	USBSSINT	USBSS	Queue MGR or CPPI Completion interrupt
18	USBINT0	USBSS	RX/TX DMA, Endpoint ready/error, or USB0 interrupt
19	USBINT1	USBSS	RX/TX DMA, Endpoint ready/error, or USB1 interrupt
20	Reserved	Reserved	
21	Reserved	Reserved	
22	Reserved	Reserved	
24	Reserved	Reserved	
25	Reserved	Reserved	
27	Reserved	Reserved	
28	Reserved	Reserved	
29	Reserved	Reserved	
30	Reserved	Reserved	
31	Reserved	Reserved	
32	Reserved	Reserved	
33	Reserved	Reserved	
34	USBWAKEUP	USBSS	USB wakeup Interrupt
35	PCIeWAKEUP	PCIe	PCIe Wakeup Interrupt
36	HDVPSSINT0	HD-VPSS	HD-VPSS Interrupt 0
37	GFXINT	SGX530	Error in the IMG bus interrupt
38	HDMIINT	HDMI	HDMI Interrupt
39	Reserved	Reserved	
40	MACRXTHR0	CPGMAC0	CPGMAC0 Receive threshold interrupt
41	MACRXINT0	CPGMAC0	CPGMAC0 Receive pending interrupt
42	MACTXINT0	CPGMAC0	CPGMAC0 Transmit pending interrupt

Table 1-68. Cortex-A8 MPU INTC Interrupt Mapping (continued)

MPUSS Interrupt Number	Event	Interrupt Source	Description
43	MACMISC0	CPGMAC0	CPGMAC0 Stat, Host, MDIO LINKINT or MDIO USERINT
44	MACRXTHR1	CPGMAC1	CPGMAC1 Receive threshold interrupt
45	MACRXINT1	CPGMAC1	CPGMAC1 Receive pending interrupt
46	MACTXINT1	CPGMAC1	CPGMAC1 Transmit pending interrupt
47	MACMISC1	CPGMAC1	CPGMAC1 Stat, Host, MDIO LINKINT or MDIO USERINT
48	PCIINT0	PCIe	Legacy Interrupts (RC mode only)
49	PCIINT1	PCIe	MSI Interrupts (RC mode only)
50	PCIINT2	PCIe	Error Interrupts
51	PCIINT3	PCIe	Power Management Interrupt
52	Reserved	Reserved	
53	Reserved	Reserved	
54	Reserved	Reserved	
55	Reserved	Reserved	
56	Reserved	Reserved	
57	Reserved	Reserved	
58	Reserved	Reserved	
59	Reserved	Reserved	
60	Reserved	Reserved	
61	Reserved	Reserved	
62	Reserved	Reserved	
63	Reserved	Reserved	
64	SDINT	SD/SDIO	SDIO interrupt
65	SPIINT	SPI	SPI Interrupt
66	Reserved	Reserved	
67	TINT1	Timer1	Timer 1 interrupt
68	TINT2	Timer2	Timer 2 interrupt
69	TINT3	Timer3	Timer 3 interrupt
70	I2CINT0	I2C0	I2C0 interrupt
71	I2CINT1	I2C1	I2C0 interrupt
72	UARTINT0	UART0	UART/IrDA 0 interrupt
73	UARTINT1	UART1	UART/IrDA 1 interrupt
74	UARTINT2	UART2	UART/IrDA 2 interrupt
75	RTCINT	RTC	Timer Interrupt
76	RTCALARMINT	RTC	Alarm interrupt
77	MBINT	Mailbox	Mailbox interrupt
78	Reserved	Reserved	
79	Reserved	Reserved	
80	MCATXINT0	McASP0	McASP 0 Transmit interrupt
81	MCARXINT0	McASP0	McASP 0 Receive interrupt
82	MCATXINT1	McASP1	McASP 1 Transmit interrupt
83	MCARXINT1	McASP1	McASP 1 Receive interrupt
84	MCATXINT2	McASP2	McASP 2 Transmit interrupt
85	MCARXINT2	McASP2	McASP 2 Receive interrupt
86	MCBSPINT	McBSP	McBSP Common(TX/RX) Interrupt
87	Reserved	Reserved	

Table 1-68. Cortex-A8 MPU INTC Interrupt Mapping (continued)

MPUSS Interrupt Number	Event	Interrupt Source	Description
88	Reserved	Reserved	
89	Reserved	Reserved	
90	Reserved	Reserved	
91	Reserved	Reserved	
92	TINT4	Timer4	Timer 4 interrupt
93	TINT5	Timer5	Timer 5 interrupt
94	TINT6	Timer6	Timer 6 interrupt
95	TINT7	Timer7	Timer 7 interrupt
96	GPIOINT0A	GPIO 0	GPIO 0 interrupt 1
97	GPIOINT0B	GPIO 0	GPIO 0 interrupt 2
98	GPIOINT1A	GPIO 1	GPIO 1 interrupt 1
99	GPIOINT1B	GPIO 1	GPIO 1 interrupt 2
100	GPMCINT	GPMC	GPMC interrupt
101	DDRERR0	DDR EMIF0	DDR2/3 0 Error interrupt
102	DDRERR1	DDR EMIF1	DDR2/3 1 Error interrupt
103	HDVICP0CONT1SYNC	HDVICP2-0	iCONT1 sync interrupt
104	HDVICP0CONT2SYNC	HDVICP2-0	iCONT2 sync interrupt
105	HDVICP1CONT1SYNC	HDVICP2-1	iCONT1 sync interrupt
106	HDVICP1CONT2SYNC	HDVICP2-1	iCONT2 sync interrupt
107	HDVICP0MBOXINT	HDVICP2-0	Mailbox interrupt
108	HDVICP1MBOXINT	HDVICP2-1	Mailbox interrupt
109	HDVICP2MBOXINT	HDVICP2-2	Mailbox interrupt
110	HDVICP2CONT1SYNC	HDVICP2-0	iCONT1 sync interrupt
111	HDVICP2CONT2SYNC	HDVICP2-0	iCONT2 sync interrupt
112	EDMA3TC0_ERRINT	EDMA TPTC0 (EDMA3TC0)	EDMA3TC0 error interrupt
113	EDMA3TC1_ERRINT	EDMA TPTC1 (EDMA3TC1)	EDMA3TC1 error interrupt
114	EDMA3TC2_ERRINT	EDMA TPTC2 (EDMA3TC2)	EDMA3TC2 error interrupt
115	EDMA3TC3_ERRINT	EDMA TPTC3 (EDMA3TC3)	EDMA3C3 error interrupt
116	Reserved	Reserved	
117	Reserved	Reserved	
118	Reserved	Reserved	
119	Reserved	Reserved	
120	SMRFLX0	Smart Reflex0	SVT SmartReflex interrupt
121	SMRFLX1	Smart Reflex1	HVT SmartReflex interrupt
122	SYSMMUINT	System MMU	Table walk abort interrupt
123	Reserved	Reserved	
124	DMMINT	DMM	PAT Fault interrupt
125	Reserved	Reserved	
126	Reserved	Reserved	
127	Reserved	Reserved	

1.8.2 Interrupt Requests to DSP INTC

Table 1-69 shows the DSP INTC interrupt mapping.

Table 1-69. DSP INTC Interrupt Mapping

DSP Interrupt Number	Event	Interrupt Source	Description
0	EVT0	INTC	Output of event combiner 0, for events 1–31
1	EVT1	INTC	Output of event combiner 1, for events 32–63
2	EVT2	INTC	Output of event combiner 2, for events 64–95
3	EVT3	INTC	Output of event combiner 3, for events 96–127
4	Reserved	Reserved	
5	Reserved	Reserved	
6	Reserved	Reserved	
7	Reserved	Reserved	
8	Reserved	Reserved	
9	EMU_DTDMA	EMU	ECM interrupt for: 1. Host scan access event 2. DTDMA transfer complete event 3. AET interrupt event
10	Reserved	Reserved	
11	EMU_RTDXRX	EMU	RTDX receive complete event
12	EMU_RTDXTX	EMU	RTDX transmit complete event
13	IDMAINT0	IDMA	C674x 0 event
14	IDMAINT1	IDMA	C674x 1 event
15	SDINT	SD/SDIO	SD/SDIO interrupt
16	SPIINT	SPI	SPI Interrupt
17	Reserved	Reserved	
18	Reserved	Reserved	
19	Reserved	Reserved	
20	EDA3CC_INT1	EDMA TPCC (EDMA3CC)	EDMA3CC Region 1 DMA completion Interrupt
21	EDMA3CC_ERRINT	EDMA3CCCC	EDMA3CC error Interrupt
22	EDMA3TC0_ERRINT	EDMA3TC0 (TPTC0)	TPTC0 error interrupt
23	Reserved	Reserved	
24	Reserved	Reserved	
25	Reserved	Reserved	
26	Reserved	Reserved	
27	Reserved	Reserved	
28	Reserved	Reserved	
29	Reserved	Reserved	
30	Reserved	Reserved	
31	Reserved	Reserved	
32	MACRXTHR0	CPGMAC0	CPGMAC0 Receive threshold interrupt
33	MACRXINT0	CPGMAC0	CPGMAC0 Receive pending interrupt
34	MACTXINT0	CPGMAC0	CPGMAC0 Transmit pending interrupt
35	MACMISC0	CPGMAC0	CPGMAC0 Stat, Host, MDIO LINKINT or MDIO USERINT
36	MACRXTHR1	CPGMAC1	CPGMAC1 Receive threshold interrupt
37	MACRXINT1	CPGMAC1	CPGMAC1 Receive pending interrupt
38	MACTXINT1	CPGMAC1	CPGMAC1 Transmit pending interrupt

Table 1-69. DSP INTC Interrupt Mapping (continued)

DSP Interrupt Number	Event	Interrupt Source	Description
39	MACMISC1	CPGMAC1	CPGMAC1 Stat, Host, MDIO LINKINT or MDIO USERINT
40	HDVPSSINT1	HD-VPSS	HD-VPSS Interrupt 1
41	HDMIINT	HDMI	HDMI Interrupt
42	Reserved	Reserved	
43	Reserved	Reserved	
44	Reserved	Reserved	
45	Reserved	Reserved	
46	Reserved	Reserved	
47	Reserved	Reserved	
48	Reserved	Reserved	
49	TINT1	Timer1	Timer 1 interrupt
50	TINT2	Timer2	Timer 2 interrupt
51	TINT3	Timer3	Timer 3 interrupt
52	TINT4	Timer4	Timer 4 interrupt
53	TINT5	Timer5	Timer 5 interrupt
54	TINT6	Timer6	Timer 6 interrupt
55	TINT7	Timer7	Timer 7 interrupt
56	MBINT	Mailbox	Mail Box Interrupt
57	Reserved	Reserved	
58	I2CINT0	I2C0	I2C0 interrupt
59	I2CINT1	I2C1	I2C0 interrupt
60	UARTINT0	UART0	UART/IrDA 0 interrupt
61	UARTINT1	UART1	UART/IrDA 1 interrupt
62	UARTINT2	UART2	UART/IrDA 2 interrupt
63	Reserved	Reserved	
64	GPIOINT0A	GPIO 0	GPIO 0 interrupt 1
65	GPIOINT0B	GPIO 0	GPIO 0 interrupt 2
66	GPIOINT1A	GPIO 1	GPIO 1 interrupt 1
67	GPIOINT1B	GPIO 1	GPIO 1 interrupt 2
68	Reserved	Reserved	
69	Reserved	Reserved	
70	MCATXINT0	McASP0	McASP 0 Transmit interrupt
71	MCARXINT0	McASP0	McASP 0 Receive interrupt
72	MCATXINT1	McASP1	McASP 1 Transmit interrupt
73	MCARXINT1	McASP1	McASP 1 Receive interrupt
74	MCATXINT2	McASP2	McASP 2 Transmit interrupt
75	MCARXINT2	McASP2	McASP 2 Receive interrupt
76	MCBSPINT	McBSP	McBSP Common(TX/RX) Interrupt
77	Reserved	Reserved	
78	Reserved	Reserved	
79	Reserved	Reserved	
80	Reserved	Reserved	
81	Reserved	Reserved	
82	Reserved	Reserved	
83	Reserved	Reserved	
84	Reserved	Reserved	

Table 1-69. DSP INTC Interrupt Mapping (continued)

DSP Interrupt Number	Event	Interrupt Source	Description
85	Reserved	Reserved	
86	Reserved	Reserved	
87	HDVICP2CONT1SYNC	HDVICP2-2	iCONT1 sync interrupt
88	HDVICP2CONT2SYNC	HDVICP2-2	iCONT2 sync interrupt
89	HDVICP2MBOXINT	HDVICP2-2	Mailbox interrupt
90	HDVICP0CONT1SYNC	HDVICP2-0	iCONT1 sync interrupt
91	HDVICP0CONT2SYNC	HDVICP2-0	iCONT2 sync interrupt
92	HDVICP1CONT1SYNC	HDVICP2-1	Mailbox interrupt
93	HDVICP1CONT2SYNC	HDVICP2-1	iCONT1 sync interrupt
94	HDVICP0MBOXINT	HDVICP2-0	iCONT2 sync interrupt
95	HDVICP1MBOXINT	HDVICP2-1	Mailbox interrupt
96	INTERR	INTC	C674x Interrupt Controller Dropped CPU Interrupt Event
97	EMC_IDMAERR	EMC	C674x EMC Invalid IDMA Parameters event
98	PBISTINT	PBIST	PBIST event
99	Reserved	Reserved	
100	EFIINTA	EFI	EFI Interrupt from side A event
101	EFIINTB	EFI	EFI Interrupt from side B event
102	Reserved	Reserved	
103	Reserved	Reserved	
104	Reserved	Reserved	
105	Reserved	Reserved	
106	Reserved	Reserved	
107	Reserved	Reserved	
108	Reserved	Reserved	
109	Reserved	Reserved	
110	Reserved	Reserved	
111	Reserved	Reserved	
112	Reserved	Reserved	
113	PMC_ED	PMC	PMC event
114	Reserved	Reserved	
115	Reserved	Reserved	
116	UMC_ED1	UMC	L2 single bit error detected event
117	UMC_ED2	UMC	L2 two bit error detected event
118	PDC_INT	PDC	Power Down sleep interrupt event
119	SYS_CMPA	SYS	SYS event
120	PMC_CMPA	PMC	L1P CPU memory protection fault event
121	PMC_DMPA	PMC	L1P DMA memory protection fault event
122	DMC_CMPA	DMC	L1D CPU memory protection fault event
123	DMC_DMPA	DMC	L1D DMA memory protection fault event
124	UMC_CMPA	UMC	L2 CPU memory protection fault event
125	UMC_DMPA	UMC	L2 DMA memory protection fault event
126	EMC_CMPA	EMC	IDMA CPU memory protection fault event
127	EMC_BUSERR	EMC	IDMA Bus error interrupt event

1.9 EDMA and EDMA Events

1.9.1 Overview

The device uses L3 crossbar architecture to control access between device processors, subsystems, and peripherals. It includes four third-party DMA transfer controllers (TPTCs) that provide four EDMA channels for transfer between L3 slaves. The device also makes use of the third-party channel controller (TPCC). The TPCC provides a user and event interface to the EDMA system. It includes up to 64 event channels to which all system synchronization events can be mapped and 8 auto-submit channels (QDMA).

Parameter RAM: Each EDMA is specified by an 8-word (32-byte) parameter table contained in Parameter RAM (PaRAM) within the TPCC. The number of PaRAM entries supported is 512.

QDMA: The quick DMA (QDMA) function is contained within the TPCC. The device implements all 8 possible QDMA channels. Each QDMA channel has a selectable PaRAM entry used to specify the transfer. A QDMA transfer is submitted immediately upon writing of the “trigger” parameter (as opposed to the occurrence of an event as with EDMA). The QDMA parameter RAM may be written by any bus master with an L3 connection to the TPCC CFG port.

1.9.2 EDMA Regions

In order to support multiple processors, the TPCC is configured with all eight interrupt regions implemented. This allows each processing element (Cortex-A8, DSP) to be interrupted upon completion of any EDMAs associated with its program flow independent of other EDMAs in the system. Assignment of the region interrupts is shown in Table 1-70.

Table 1-70. EDMA Regions

TPCC Region	Processor	Interrupt Name (Processor)
0	Cortex-A8	CCINT0
1	DSP	CCINT1
2	–	–
3	–	–
4	–	–
5	–	–
6	–	–
7	–	–

1.9.3 Synchronization Events

The EDMA supports up to 64 EDMA channels that service peripheral devices and external memory.Table 1-71 lists the source of EDMA synchronization events associated with each of the programmable EDMA channels. The association of an event to a channel is fixed; each of the EDMA channels has one specific event associated with it. These specific events are captured in the EDMA event registers (ER, ERH) even if the events are disabled by the EDMA event enable registers (EER, EERH). For more detailed information on the EDMA module and how EDMA events are enabled, captured, processed, linked, chained, and cleared, see the *Enhanced DMA (EDMA) Controller* chapter.

Table 1-71. EDMA Channel Synchronization Events

TPCC Channel	Default Event #	Binary	Default Event	Source
0	0	00 0000	Unused	–
1	1	00 0001	Unused	–
2	2	00 0010	Unused	–
3	3	00 0011	Unused	–
4	4	00 0100	Unused	–
5	5	00 0101	Unused	–
6	6	00 0110	Unused	–
7	7	00 0111	Unused	–
8	8	00 1000	AXEVT0	McASP0
9	9	00 1001	AREVT0	McASP0
10	10	00 1010	AXEVT1	McASP1
11	11	00 1011	AREVT1	McASP1
12	12	00 1100	AXEVT2	McASP2
13	13	00 1101	AREVT2	McASP2
14	14	00 1110	BXEVT	McBSP
15	15	00 1111	BREVT	McBSP
16	16	01 0000	SPIXEVT0	SPI
17	17	01 0001	SPIREVT0	SPI
18	18	01 0010	SPIXEVT1	SPI
19	19	01 0011	SPIREVT1	SPI
20	20	01 0100	SPIXEVT2	SPI
21	21	01 0101	SPIREVT2	SPI
22	22	01 0110	SPIXEVT3	SPI
23	23	01 0111	SPIREVT4	SPI
24	24	01 1000	SDTXEVT	SD
25	25	01 1001	SDRXEVT	SD
26	26	01 1010	UTXEVT0	UART0
27	27	01 1011	URXEVT0	UART0
28	28	01 1100	UTXEVT1	UART1
29	29	01 1101	URXEVT1	UART1
30	30	01 1110	UTXEVT2	UART2
31	31	01 1111	URXEVT2	UART2
32	32	10 0000	Unused	–
33	33	10 0001	Unused	–
34	34	10 0010	Unused	–
35	35	10 0011	Unused	–
36	36	10 0100	Unused	–
37	37	10 0101	Unused	–
38	38	10 0110	Unused	–
39	39	10 0111	Unused	–
40	40	10 1000	Unused	–
41	41	10 1001	Unused	–
42	42	10 1010	Unused	–
43	43	10 1011	Unused	–
44	44	10 1100	Unused	–
45	45	10 1101	Unused	–
46	46	10 1110	Unused	–
47	47	10 1111	Unused	–

Table 1-71. EDMA Channel Synchronization Events (continued)

TPCC Channel	Default Event #	Binary	Default Event	Source
48	48	11 0000	TINTEVT4	Timer 4
49	49	11 0001	TINTEVT5	Timer 5
50	50	11 0010	TINTEVT6	Timer 6
51	51	11 0011	TINTEVT7	Timer 7
52	52	11 0100	GPMCEVT	GPMC
53	53	11 0101	HDMIEVT	HDMI
54	54	11 0110	Unused	–
55	55	11 0111	Unused	–
56	56	11 1000	Unused	–
57	57	11 1001	Unused	–
58	58	11 1010	I2CTXEVT0	I2C0
59	59	11 1011	I2CRXEVT0	I2C0
60	60	11 1100	I2CTXEVT1	I2C1
61	61	11 1101	I2CRXEVT1	I2C1
62	62	11 1110	Unused	–
63	63	11 1111	Unused	–

1.10 Device Clocking and Flying Adder PLL

1.10.1 Overview

This device has 4 on-chip PLLs and requires a 27-MHz reference clock that is generated by an on-chip oscillator. Figure 1-67 shows a high-level overview of the device clocking structure.

For detailed specifications on clock frequency and voltage requirements, see the device-specific data manual.

In addition to the reference clock, there is a 100-MHz differential clock input for the SATA and PCIe. A third clock input is an optional 32768-Hz clock input (no on-chip oscillator) for the RTC. All possible input clocks are described in Table 1-72 and System Clock Domains are described in Table 1-73. Figure 1-68 shows detailed clocking architecture on this device.

Figure 1-67. Top Level Clock Architecture

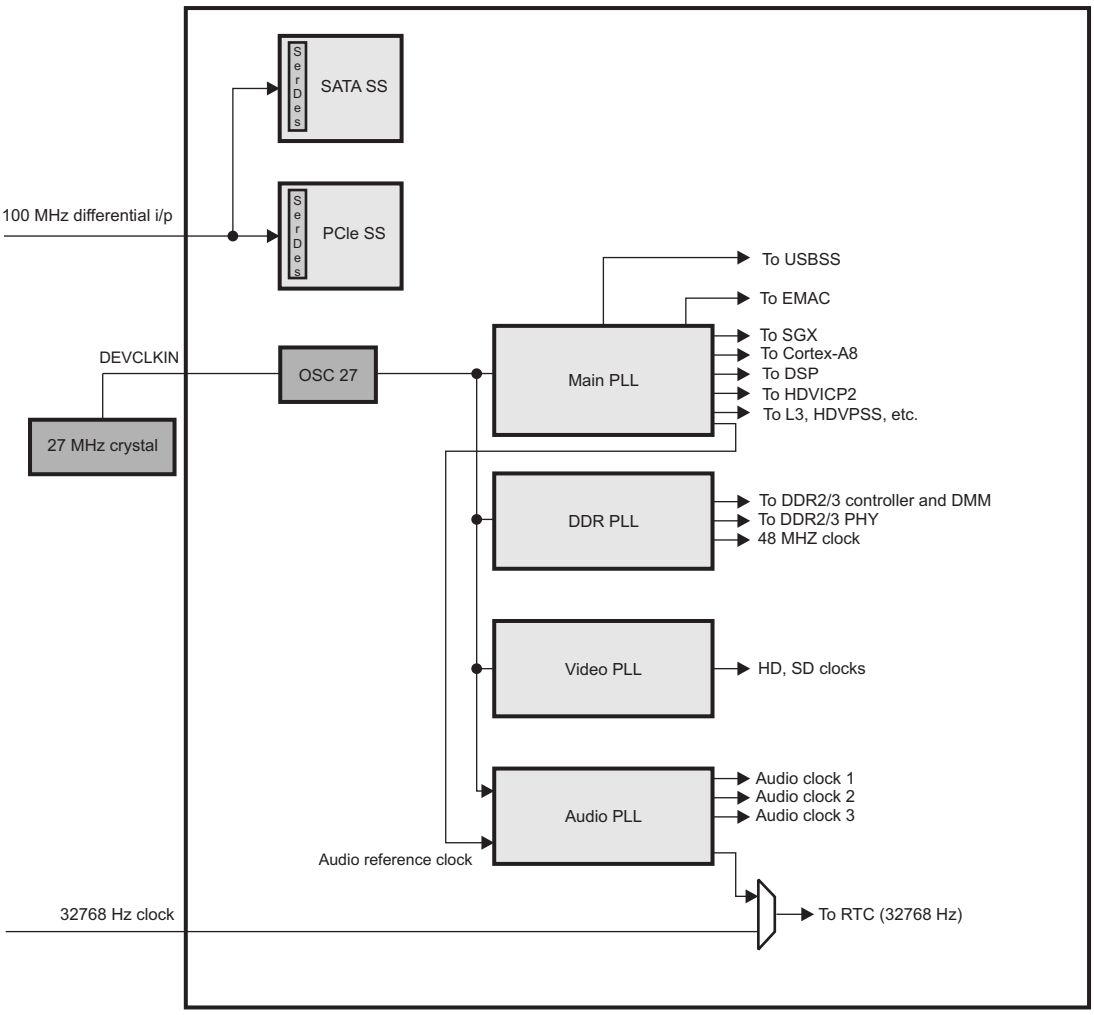


Table 1-72. Device Clock Inputs

Peripheral	Input Clock Signal Name
PLL	CLKIN1, CLKIN2
RTC	CLKIN32
JTAG	TCLK, RTCLK
EMAC	GMII0_RXCLK, GMII0_TXCLK, GMII1_RXCLK, GMII1_TXCLK
HDVPSS	VIN0_CLK0, VIN0_CLK1, VIN1_CLK0, VIN1_CLK1
McASP	MCA0_ACLKR, MCA0_AHCLKR, MCA0_ACLKX, MCA0_AHCLKX, MCA1_ACLKR, MCA1_AHCLKR, MCA1_ACLKX, MCA1_AHCLKX, MCA2_ACLKR, MCA2_AHCLKR, MCA2_ACLKX, MCA2_AHCLKX
McBSP	MCB_CLKS
PCIe, SATA	SERDES_CLKP, SERDES_CLKP
SPI	SPI_SCLK
TIMER	TCLKIN
I2C	I2C0_SCL, I2C1_SCL

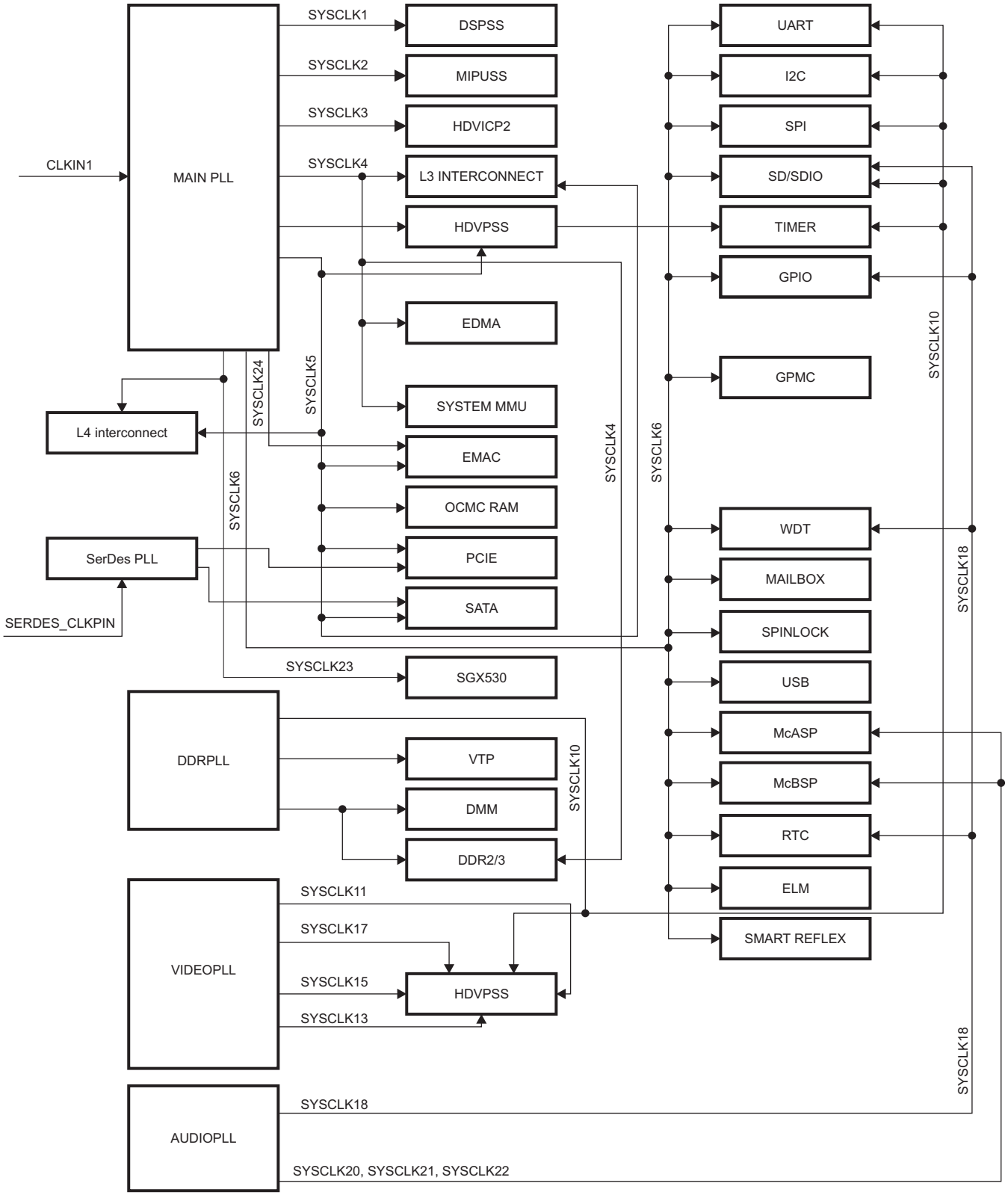
Table 1-73. System Clock Domains

CPU/Device Peripherals	System Clock Domain	Description	PLL Type
DSPSS	SYSCLK1	DSP Functional Clock	MAINPLL
MPUSS	SYSCLK2	Cortex-A8 Functional Clock	MAINPLL
HDVICP2	SYSCLK3	Interface and Functional Clock	MAINPLL
L3 Interconnect	SYSCLK4	L3 Interconnect Clock	MAINPLL
	SYSCLK5	L3 Interconnect Clock	MAINPLL
	SYSCLK6	L3 Interconnect Clock	MAINPLL
HDVPSS	SYSCLK4	VPDMA Functional Clock	MAINPLL
	SYSCLK5	Functional Clock	MAINPLL
	SYSCLK6	Interface Clock	MAINPLL
	SYSCLK10	HDMI CEC Interface Clock	DDRPLL
	SYSCLK13	HDMI/DVO1/DVO2 Pixel Clock	VIDEOPLL
	SYSCLK15	HD Comp/DVO2 Pixel Clock	VIDEOPLL
	SYSCLK17	SD VENC Pixel Clock	VIDEOPLL
SYSTEM MMU	SYSCLK4	Interface Clock	MAINPLL
EDMA	SYSCLK4	Interface and Functional Clock	MAINPLL
DMM	SYSCLK8	Functional Clock	DDRPLL
L4 Interconnect	SYSCLK5	High Speed Interconnect Clock	MAINPLL
	SYSCLK6	Standard Speed Interconnect Clock	MAINPLL
EMAC	SYSCLK5	Interface and Functional Clock	MAINPLL
	SYSCLK24	EMAC Reference Clock	MAINPLL
SATA	SYSCLK5	Interface and Functional Clock	MAINPLL
PCIe	SYSCLK5	Interface and Functional Clock	MAINPLL
OCMCRAM	SYSCLK5	Interface and Functional Clock	MAINPLL
UART	SYSCLK6	Interface Clock	MAINPLL
	SYSCLK10	Functional Clock	DDRPLL
I2C	SYSCLK6	Interface Clock	MAINPLL
	SYSCLK10	Functional Clock	DDRPLL
SPI	SYSCLK6	Interface Clock	MAINPLL
	SYSCLK10	Functional Clock	DDRPLL

Table 1-73. System Clock Domains (continued)

CPU/Device Peripherals	System Clock Domain	Description	PLL Type
SD/SDIO	SYSCLK6	Interface Clock	MAINPLL
	SYSCLK10	Functional Clock	DDRPLL
	SYSCLK18	Debounce Clock	AUDIOPLL
TIMER	SYSCLK18	Functional Clock	AUDIOPLL
	SYSCLK6	Interface Clock	MAINPLL
GPIO	SYSCLK6	Interface and Functional Clock	MAINPLL
	SYSCLK18	Debounce Functional Clock	MAINPLL
GPMC	SYSCLK6	Interface and Functional Clock	MAINPLL
HDMI	SYSCLK6	Interface Clock	MAINPLL
Watch Dog Timer	SYSCLK6	Interface Clock	MAINPLL
	SYSCLK18	Functional Clock	AUDIOPLL
Mailbox	SYSCLK6	Interface and Functional Clock	MAINPLL
Spinlock	SYSCLK6	Interface and Functional Clock	MAINPLL
Smart Reflex	SYSCLK6	Interface and Functional Clock	MAINPLL
USB	SYSCLK6	Interface and Functional Clock	MAINPLL
DDR2/DDR3	SYSCLK8	Functional Clock	DDRPLL
	SYSCLK4	Interface Clock	MAINPLL
VTP	SYSCLK9	Interface and Functional Clock	DDRPLL
RTC	SYSCLK6	Interface Clock	MAINPLL
	SYSCLK18	Functional Clock	AUDIOPLL
McASP, McBSP	SYSCLK6	Interface Clock	MAINPLL
	SYSCLK20	Functional Clock	AUDIOPLL
	SYSCLK21	Functional Clock	AUDIOPLL
	SYSCLK22	Functional Clock	AUDIOPLL
SGX530	SYSCLK23	Interface and Functional Clock	MAINPLL
ELM	SYSCLK6	Interface and Functional Clock	MAINPLL

Figure 1-68. Detailed Clock Architecture



1.10.2 I/O Domains

The I/O domains refer to the frequencies of the peripherals that communicate through device pins. In many cases, there are frequency requirements for a peripheral pin interface that are set by an outside standard and must be met. It is not necessarily possible to obtain these frequencies from the on-chip clock generation circuitry, so the frequencies must be obtained from external sources and are asynchronous to the CPU frequency by definition. Table 1-74 shows the external Peripheral Clock Sources.

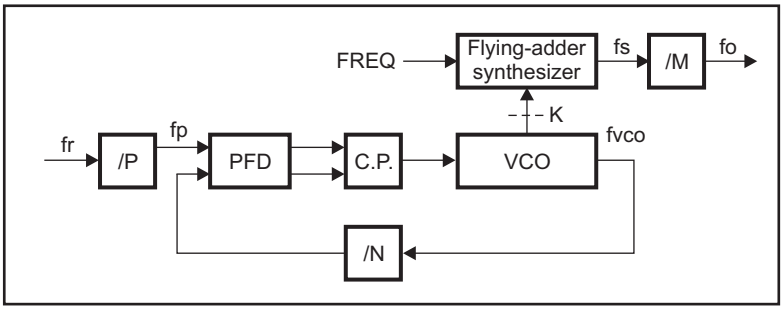
Table 1-74. External Peripheral Clock Sources

I/O (External) Domain Clock Source Options			
Peripheral	**I/O Domain Clock Frequency**	**Internal Clock Source**	**External Clock Source**
DDR2	333 to 533 MHz	SYSCLK8	
DDR3	333 to 800 MHz	SYSCLK8	
MTS	27 MHz	SYSCLK14, SYSCLK16	MTSI_DLCK
EMAC	125 MHz	SYSCLK24	EMAC[0]_TXCLK, EMAC[0]_RXCLK, EMAC[1]_TXCLK, EMAC[1]_RXCLK
MDIO	25 MHz	SYSCLK6	
HDVPSS	162 MHz	SYSCLK13, SYSCLK15, SYSCLK17	VIN[0]A_CLK, VIN[0]B_CLK,VIN[1]A_CLK, VIN[1]B_CLK
McASP	50 MHz	SYSCLK20, SYSCLK21, SYSCLK22	MCA[0]_ACLKX, MCA[0]_AHCLKX, MCA[0]_ACLKR, MCA[0]_AHCLKR, MCA[1]_ACLKX, MCA[1]_AHCLKX, MCA[1]_ACLKR, MCA[1]_AHCLKR, MCA[2]_ACLKX, MCA[2]_AHCLKX, MCA[2]_ACLKR, MCA[2]_AHCLKR
McBSP	50 MHz	SYSCLK20, SYSCLK21, SYSCLK22	MCB_CLKS, MCB_CLKX, MCB_CLKR
SPI	48 MHz	SYSCLK6	SPI_SCLK
GPMC	125 MHz	SYSCLK6	
I2C	up to 400 KHz	SYSCLK6	I2C[0]_SCL, I2C[1]_SCL
RTC	32.768 KHz		CLKIN32
TIMER	50 MHz		TCLKIN
HDMI	48 MHz	SYSCLK13, SYSCLK15, SYSCLK17	
PCIe/SATA	100 MHz		SERDES_CLKP, SERDES_CLKN
SD/SDIO	48 MHz	SYSCLK6	
MAINPLL, DDRPLL, VIDEOPLL, AUDIOPLL	27 MHz		CLKIN1

1.10.3 Flying Adder PLL

The device uses Flying Adder PLLs for all on-chip PLLs. Figure 1-69 shows basic structure of the Flying Adder PLL.

Figure 1-69. Flying Adder PLL



Flying Adder PLL (FAPLL) has two main components:

1. Multiphase PLL
2. Flying Adder Synthesizer.

The multiphase PLL takes input reference clock (f_r), mutliplies it with factor N and provides a K phase output to the flying adder synthesizer. The flying adder synthesizer takes this multi phase clock input and produces a variable frequency clock (f_s). There is post divider on this clock that takes in clock f_s and drives out clock f_o.

The f_{vco} frequency of the PLL is given by $f_{vco} = (N/P) \times f_r$

The f_s frequency of the FAPLL is given by $f_s = (f_{vco} \times K)/FREQ$

The f_o frequency of the Post divider is given by $f_o = f_s/M$

Where:

f_{vco} = VCO frequency. Range is 800 to 1600 MHz.

f_r = Input reference frequency

N = Multiplier is 9 bits and supported range is 1 to 511.

P = Pre-divider

K = Phase Output. 8 is the value for this device

f_s = FAPLL output

f_o = Post Divider output

FREQ = Supported integer range is 8 to 15 and fraction range is 0 to FF FFFFh.

M = Post Divider. Value ranges from 1 to 255.

There can be multiple flying adder synthesizers attached to one multiphase PLL to generate different frequencies. FREQ and M values can be adjusted for each clock separately based on the frequency needed.

NOTE: Maximum frequency of PLL out clock must meet minimum cycle limits requirements. Refer to the device-specific data manual for PLL Programming Limits.

1.10.3.1 PLL Types

This device has 4 on-chip PLLs:

1. MAIN PLL
2. DDR PLL
3. VIDEO PLL
4. AUDIO PLL

1.10.3.1.1 Main PLL

Figure 1-70 shows structure of Main PLL. The Main PLL has 5 flying adder synthesizers connected to the multi-phase PLL. The outputs of these synthesizers are muxed with the 27-MHz reference clock to allow its selection during PLL bypass mode. Table 1-76 lists the IPs that are clocked from each of these flying adder synthesizers.

As shown in Figure 1-70, there are five independently controllable clock outputs (asynchronous to each other) driven by the main PLL. The FREQ and post divider can be tuned independently to control each clock. This makes it possible to run DSP, Cortex-A8, HDVICP2 and interconnect independent of each other. However as the same multiphase PLL is used to generate all these clocks, the choice of possible frequencies will be limited. The set of available frequencies will depend upon choice of P and N for the multi phase PLL.

The Main PLL also provides a reference clock for the Audio PLL and the USB subsystem using programmable dividers (for audio PLL divider factor, see Section 1.16.1.2.14; for USB PLL divider factor, see Section 1.16.1.2.13). Table 1-77, Table 1-78, and Table 1-79 show examples for Main PLL System Clock generation for speed grade blank, grade 2, and grade 4, respectively. In these examples, Multiplier N is carefully set to 56 to allow 24 MHz USB clock generation and keep the VCO frequency less than 1600 MHz. Note that the SYSCLK frequencies listed in the examples are less than the maximum frequency listed in Table 1-76. This is because the minimum cycle limit has been applied by selecting a FREQ × M value that does not cause the instantaneous frequency to go over the maximum frequency.

Figure 1-70. Main PLL Structure

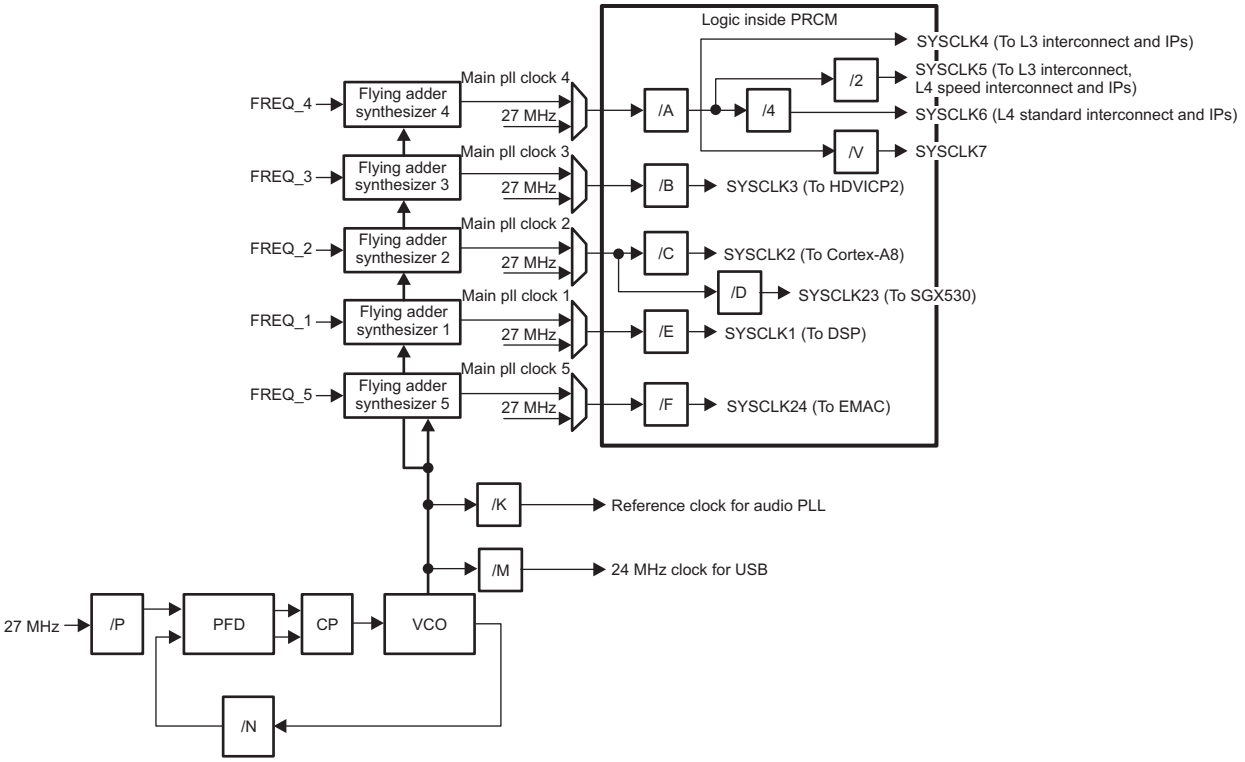


Table 1-75. MAIN PLL Dividers

Divider	Supported Divide Ratios	Default Value
A	1/1, 1/2	1/1
B	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
C	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
D	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/4
E	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
V	1/5, 1/6, 1/8, 1/16	1/5
F	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
K	See MAINPLL_DIV7[7:0] MAIND_DIV7 (Table 1-182)	1/4
M	See MAIN_PLL_DIV6[7:0] MAIND_DIV6 (Table 1-181)	1/72

Table 1-76. Main PLL SYSCLK Frequencies and Destination

SYSCLK	Device Speed Range[1]	Maximum Frequency (MHz)[2]	Destination
SYSCLK1	Blank	750	To C674x DSP (DSPSS)
	2	930	
	4	1000	
SYSCLK2	Blank	930	To ARM Cortex-A8 (MPUSS)
	2	1100	
	4	1200	
SYSCLK3	Blank	500	To HDVICP2
	2	550	
	4	630	
SYSCLK4	Blank	460	L3, OCP clock for HDVPSS, TPTCs, TPCC, DMM, Unicache clock for Media Controller, EDMA
	2	550	
	4	570	
SYSCLK5	Blank	230	L3, L4_HS, OCP clock for EMAC, SATA, PCIe, Media Controller, OCMC RAM
	2	275	
	4	285	
SYSCLK6	Blank	115	L3, L4_STD, UART, I2C, SPI, SD, SDIO, TIMER, GPIO, PRCM, McASP, McBSP, GPMC, ELM, HDMI, WDT, Mailbox, RTC, Spinlock, SmartReflex and USB
	2	137	
	4	143	
SYSCLK7	Blank	90	Reserved
	2	110	
	4	115	
SYSCLK8	Blank	364	DMM, DDR OCP clock
	2	364	
	4	425	
SYSCLK23	Blank	310	SGX530 OCP clock
	2	275	
	4	300	
SYSCLK24	Blank	125	EMAC clock
	2		
	4		

(1) For more information on the available device speed ranges for each part number, see the device-specific data manual

(2) Maximum frequency of PLL out clocks should meet minimum cycle limits. Refer to the device-specific data manual for PLL Programming Limits.

Table 1-77. Example for Main PLL Frequencies for Speed Grade Blank

Input Ref Freq	Pre-Divider	Mult	VCO Output Freq	4 Bits Integer	24 Bits Fract	FAPLL Output	Post Divider	Post Divider Output		PRCM Divider		System Clock Domain	SYSCLK Freq (MHz)
								Mean[1]	Max[2]				
f_r (MHz)	P[3]	N[3]	f_{vco} (MHz)	FREQ[3]		f_s (MHz)	M[3]	f_o (MHz)					
27	1	56	1512	8	0	1512	2	756	767	1		SYSCLK1	756
				13	0	930	1	930	954	1		SYSCLK2	930
										3		SYSCLK23	310
				12	0	1008	2	504	510	1		SYSCLK3	504
				13	0	930	2	465	473	1		SYSCLK4	465
										1	2	SYSCLK5	233
										1	4	SYSCLK6	116
										1	5	SYSCLK7	93
				12	0.096	1000	8	125	127	1		SYSCLK24	125

(1) Mean frequency is an average FAPLL output frequency over time when P, N, M, and FREQ are tuned.
(2) Maximum frequency is an instantaneous frequency when FAPLL generates clock at minimum clock cycle.
(3) The values of Pre-Divider (P), Multiplier (N), FREQ, and Post Divider (M) can be tuned in order to obtain different frequencies; however, the software must respect the limitations (and requirements) described in the device-specific data manual for PLL Programming Limits.

Table 1-78. Example for Main PLL Frequencies for Speed Grade 2

Input Ref Freq	Pre-Divider	Mult	VCO Output Freq	4 Bits Integer	24 Bits Fract	FAPLL Output	Post Divider	Post Divider Output		PRCM Divider		System Clock Domain	SYSCLK Freq (MHz)
								Mean[1]	Max[2]				
f_r (MHz)	P[3]	N[3]	f_{vco} (MHz)	FREQ[3]		f_s (MHz)	M[3]	f_o (MHz)					
27	1	56	1512	13	0	930	1	930	954	1		SYSCLK1	930
				11	0	1100	1	1100	1131	1		SYSCLK2	1100
										4		SYSCLK23	275
				11	0	1100	2	550	560	1		SYSCLK3	550
				11	0	1100	2	550	560	1		SYSCLK4	550
										1	2	SYSCLK5	275
										1	4	SYSCLK6	138
										1	5	SYSCLK7	110
				12	0.096	1000	8	125	127	1		SYSCLK24	125

(1) Mean frequency is an average FAPLL output frequency over time when P, N, M, and FREQ are tuned.
(2) Maximum frequency is an instantaneous frequency when FAPLL generates clock at minimum clock cycle.
(3) The values of Pre-Divider (P), Multiplier (N), FREQ, and Post Divider (M) can be tuned in order to obtain different frequencies; however, the software must respect the limitations (and requirements) described in the device-specific data manual for PLL Programming Limits.

Table 1-79. Example for Main PLL Frequencies for Speed Grade 4

Input Ref Freq	Pre-Divider	Mult	VCO Output Freq	4 Bits Integer	24 Bits Fract	FAPLL Output	Post Divider	Post Divider Output		PRCM Divider		System Clock Domain	SYSCLK Freq (MHz)
								Mean[1]	Max[2]				
f_r (MHz)	P[3]	N[3]	f_{vco} (MHz)	FREQ[3]		f_s (MHz)	M[3]	f_o (MHz)					
27	1	56	1512	12	0	1008	1	1008	1035	1		SYSCLK1	1008
				10	0	1210	1	1210	1247	1		SYSCLK2	1210
										4		SYSCLK23	303
				9	0.5	1273	2	637	649	1		SYSCLK3	637
				10	0.5	1152	2	576	586	1		SYSCLK4	576
										1	2	SYSCLK5	288
										1	4	SYSCLK6	144
										1	5	SYSCLK7	115
				12	0.096	1000	8	125	127	1		SYSCLK24	125

(1) Mean frequency is an average FAPLL output frequency over time when P, N, M, and FREQ are tuned.
(2) Maximum frequency is an instantaneous frequency when FAPLL generates clock at minimum clock cycle.
(3) The values of Pre-Divider (P), Multiplier (N), FREQ, and Post Divider (M) can be tuned in order to obtain different frequencies; however, the software must respect the limitations (and requirements) described in the device-specific data manual for PLL Programming Limits.

Table 1-80 is an example that shows the effect of a fractional FREQ causing the SYSCLK5 frequency to break the maximum frequency limit. The maximum frequency of SYSCLK5 for speed grade 2 is 280 MHz. When FREQ is set to integer 11, SYSCLK5 output mean frequency is 275 MHz and its maximum instantaneous frequency is 280 MHz, which is not over the maximum frequency limit. When FREQ is set to fractional 10.8, SYSCLK5 output maximum instantaneous frequency is 293 MHz, which is over the maximum frequency limit.

Table 1-80. Example of SYSCLK5 Frequency with Integer FREQ vs Fractional FREQ

Input Ref Freq	Pre-Divider	Mult	VCO Output Freq	4 Bits Integer	24 Bits Fract	FAPLL Output	Post Divider	Post Divider Output		PRCM Divider		System Clock Domain	SYSCLK Mean Freq (MHz)
								Mean[1]	Max[2]				
f_r (MHz)	P[3]	N[3]	f_{vco} (MHz)	FREQ[3]		f_s (MHz)	M[3]	f_o (MHz)					
27	1	56	1512	11	0	1100	2	550	560	1	2	SYSCLK5	275
				10	0.8	1120	2	560	587	1	2	SYSCLK5	280

(1) Mean frequency is an average FAPLL output frequency over time when P, N, M, and FREQ are tuned.
(2) Maximum frequency is an instantaneous frequency when FAPLL generates clock at minimum clock cycle.
(3) The values of Pre-Divider (P), Multiplier (N), FREQ, and Post Divider (M) can be tuned in order to obtain different frequencies; however, the software must respect the limitations (and requirements) described in the device-specific data manual for PLL Programming Limits.

1.10.3.1.1.1 Steps for Changing Main PLL Frequency

Refer to the appropriate subsection on how to program the MAINPLL clocks:

- If the MAINPLL is powered down (MAIN_PLLEN bit in MAINPLL_CTRL is cleared to 0), follow the procedure in Section 1.10.3.1.1.1.1 to initialize the MAINPLL.
- If the MAINPLL is not powered down (MAIN_PLLEN bit in MAINPLL_CTRL is set to 1), follow the procedure in Section 1.10.3.1.1.1.2 to initialize the MAINPLL multiplier.
- If the MAINPLL is already running at a desired frequency and you only want to change the SYSCLK FREQ and post dividers, follow the procedure in Section 1.10.3.1.1.1.3 to change the SYSCLK dividers.

Note that the MAINPLL is powered down after the following device-level global resets:

- Power-on Reset (POR)
- Warm Reset (RESET)
- Max Reset

1.10.3.1.1.1.1 Initializing and Enabling MAINPLL from Power Down Mode

If the MAINPLL is powered down (MAIN_PLLEN bit in MAINPLL_CTRL is cleared to 0), perform the following procedure to initialize the MAINPLL:

1. Set MAIN_BP bit in MAINPLL_CTRL to 1 to put MAINPLL in bypass mode.
2. Set MAIN_PLLEN bit in MAINPLL_CTRL to 1 to bring MAINPLL out of power-down mode.
3. Clear PWD_CLK1 to PWD_CLK7 bits in MAINPLL_PWD to 0 to bring individual output clocks of MAINPLL out of power-down mode.
4. Set or clear MAIN_LOC_CTL bit in MAINPLL_CTRL to select MAINPLL lock output polarity (MAIN_LOCK bit in MAINPLL_CTRL).
5. Program the required pre-divider and multiplier values in MAIN_P and MAIN_N bit fields of MAINPLL_CTRL.
6. If necessary, program MAIN_INTFREQx and MAIN_FRACFREQx bit fields and set MAIN_LDFREQx bit to 1 in MAINPLL_FREQx to load integer and fraction values into Main Synthesizer x.
7. If necessary, program MAIN_MDIVx bit field and set MAIN_LDMDIVx bit to 1 in MAINPLL_DIVx to load the Post Divider into Main Synthesizer x.
8. Wait for PLL to Lock: Poll for MAIN_LOCK bit in MAINPLL_CTRL to become 1, if MAIN_LOC_CTL bit in MAINPLL_CTRL was cleared to 0; else Poll for MAIN_LOCK bit in MAINPLL_CTRL to become 0, if MAIN_LOC_CTL bit in MAINPLL_CTRL was set to 1.
9. If MAINPLL is locked in step 8, then clear the MAIN_BP bit in MAINPLL_CTRL to 0 to bring MAINPLL out of bypass mode.

Where x = 1, 2, 3, 4, 5 Flying Adder Synthesizer.

1.10.3.1.1.1.2 Changing MAINPLL Multiplier

If the MAINPLL is not powered down (MAIN_PLLEN bit in MAINPLL_CTRL is set to 1), perform the following procedure to initialize the MAINPLL multiplier:

1. Set MAIN_BP bit in MAINPLL_CTRL to 1 to put MAINPLL in bypass mode.
2. Clear PWD_CLK1 to PWD_CLK7 bits in MAINPLL_PWD to 0 to bring individual output clocks of MAINPLL out of power-down mode.
3. Set or clear MAIN_LOC_CTL bit in MAINPLL_CTRL to select MAINPLL lock output polarity (MAIN_LOCK bit in MAINPLL_CTRL).
4. Program the required pre-divider and multiplier values in MAIN_P and MAIN_N bit fields of MAINPLL_CTRL.
5. If necessary, program MAIN_INTFREQx and MAIN_FRACFREQx bit fields and set MAIN_LDFREQx bit to 1 in MAINPLL_FREQx to load integer and fraction values into Main Synthesizer x.
6. If necessary, program MAIN_MDIVx bit field and set MAIN_LDMDIVx bit to 1 in MAINPLL_DIVx to load the Post Divider into Main Synthesizer x.
7. Wait for PLL to Lock: Poll for MAIN_LOCK bit in MAINPLL_CTRL to become 1, if MAIN_LOC_CTL bit in MAINPLL_CTRL was cleared to 0; else Poll for MAIN_LOCK bit in MAINPLL_CTRL to become 0, if MAIN_LOC_CTL bit in MAINPLL_CTRL was set to 1.
8. If MAINPLL is locked in step 7, then clear the MAIN_BP bit in MAINPLL_CTRL to 0 to bring MAINPLL out of bypass mode.

Where x = 1, 2, 3, 4, 5 Flying Adder Synthesizer.

1.10.3.1.1.1.3 Changing MAINPLL SYSCLK Dividers

This section discusses the software sequence to change the SYSCLK dividers.

1. If necessary, program MAIN_INTFREQx and MAIN_FRACFREQx bit fields and set MAIN_LDFREQx bit to 1 in MAINPLL_FREQx to load integer and fraction values into Main Synthesizer x.
2. If necessary, program MAIN_MDIVx bit field and set MAIN_LDMDIVx bit to 1 in MAINPLL_DIVx to load the Post Divider into Main Synthesizer x.
3. If necessary, further divide down the SYSCLKs by programming the respective PRCM registers to get the desired SYSCLK frequencies (see Table 1-75 for supported PRCM divider values).

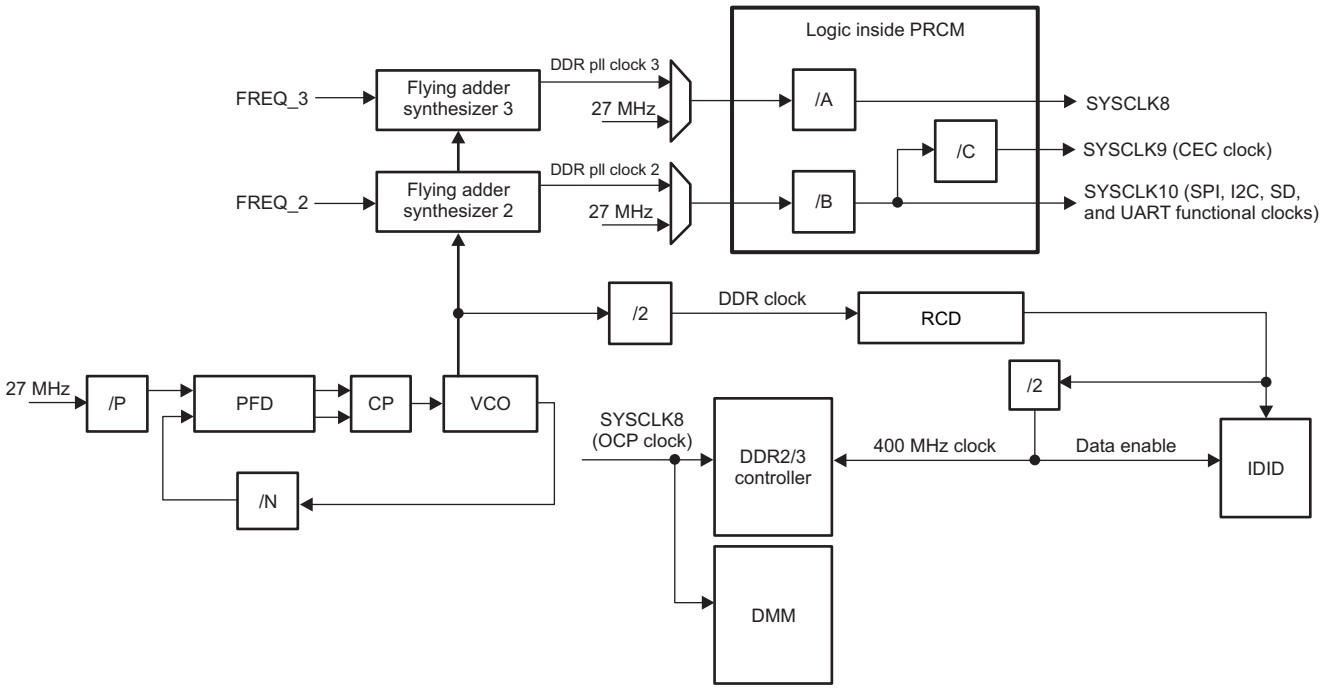
Where x = 1, 2, 3, 4, 5, 6, 7 Flying Adder Synthesizer.

1.10.3.1.2 DDR PLL

Figure 1-71 shows the structure of the DDR PLL. An 800-MHz DDR clock is derived directly from the PLL's VCO output (divided by 2, Post Divider). This clock passes through the RCD(Reset Clock Distribution) to the IDID (DDR Macro). The IDID clock is also divided by 2 to create DDR2/3 controller clock which connects to the IDID to serve as a data enable. The DDR PLL has 2 flying adder synthesizers. One is to generate the 400-MHz OCP clock for the DDR2/3 controller and 364 MHz for DMM and the other is the 48-MHz SYSCLK10 for SPI, I2C, UART, HDMI CEC, and SD/SDIO functional clocks. SYSCLK9 is dividable clock derived from SYSCLK10 which goes to the VTP controller as the VTP clock. The outputs of both synthesizers are muxed with the 27-MHz reference clock to allow its selection during PLL bypass mode.

The L3 interconnect probe also requires a 400-MHz clock synchronous to the DDR2/3 controller OCP interface clock for capturing DDR2/3 probe data. This clock must be free running to keep alive the host service network within the L3. The DDR PLL Clock 3 output directly from the FA-PLL rather than the SYSCLK8 gated clock is used for this purpose. Additionally the SYSCLK8 divider (/A) is tied to 1/1 to ensure that the DDR2/3 controller OCP and L3 interconnect probe remain frequency locked.

Figure 1-71. DDR PLL Structure



A good set of frequencies for DDR2/3 can be generated by playing with FREQ and post divider values. Table 1-81 shows supported divide ratios in PRCM module for dividers.

Table 1-81. DDR PLL Dividers

Divider	Supported Divide Ratio	Default Value
A	1/1	1/1
B	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
C	1/3	1/3

Table 1-82 lists the IPs that are clocked from DDR PLL flying adder synthesizers.

Table 1-82. DDR PLL SYSCLK Frequencies and Destination

SYSCLK	Device Speed Range[1]	Maximum Frequency (MHz)[2]	Destination
SYSCLK8[3]	Blank	364	DMM, DDR OCP clock
	2	364	
	4	425	
SYSCLK9	Blank	16	CEC clock and VTP
	2		
	4		
SYSCLK10	Blank	48	SPI, I2C, SD/SDIO, and UART
	2		
	4		
DDR2/DDR3	Blank	400	DDR2/DDR3
	2		
	4		

(1) For more information on the available device speed ranges for each part number, see the device-specific data manual

(2) Maximum frequency of PLL out clocks should meet minimum cycle limits. Refer to the device-specific data manual for PLL Programming Limits.

(3) SYSCLK8 is applied to the 128-bit interface and the EMIF DDR3 is a 32-bit interface; therefore, the SYSCLK8 364 MHz can work with an 800 MHz DDR3 clock speed without any system issue.

Table 1-83 and Table 1-84 show examples for DDR PLL System Clock generation for speed grade blank and grade 2, and grade 4, respectively. Table 1-85 shows an example for DDR PLL System Clock generation for all speed grades at lower DDR clock speed.

NOTE: Software must configure the desired FREQ values in DDRPLL_FREQ3 register (see Section 1.16.1.2.20) to obtain 364 MHz clock for DMM or 400 MHz clock for DDR2/3.

Table 1-83. Example for DDR PLL Frequencies for Speed Grade Blank and Grade 2

Input Ref Freq	Pre-Divider	Mult	VCO Output Freq	4 Bits Integer	24 Bits Fract	FAPLL Output	Post Divider	Post Divider Output		PRCM Divider	System Clock Domain	SYSCLK Freq (MHz)
								Mean[1]	Max[2]			
f_r (MHz)	P[3]	N[3]	f_{vco} (MHz)	FREQ[3]		f_s (MHz)	M[3]	f_o (MHz)				
27	1	59	1593	11	0.666667	1092	3	364	369	1	SYSCLK8	364
				8	0.85	1440	30	48	48	3	SYSCLK9	16
				8	0.85	1440	30	48	48	1	SYSCLK10	48
				NA	NA	NA	2	797	797	NA	DDR External Clock	797
				NA	NA	NA	4	398	398	NA	DDR Clock	398

(1) Mean frequency is an average FAPLL output frequency over time when P, N, M, and FREQ are tuned.

(2) Maximum frequency is an instantaneous frequency when FAPLL generates clock at minimum clock cycle.

(3) The values of Pre-Divider (P), Multiplier (N), FREQ, and Post Divider (M) can be tuned in order to obtain different frequencies; however, the software must respect the limitations (and requirements) described in the device-specific data manual for PLL Programming Limits.

Table 1-84. Example for DDR PLL Frequencies for Speed Grade 4

Input Ref Freq	Pre-Divider	Mult	VCO Output Freq	4 Bits Integer	24 Bits Fract	FAPLL Output	Post Divider	Post Divider Output		PRCM Divider	System Clock Domain	SYSCLK Freq (MHz)
								Mean[1]	Max[2]			
f_r (MHz)	P[3]	N[3]	f_{vco} (MHz)	FREQ[3]		f_s (MHz)	M[3]	f_o (MHz)				
27	1	59	1593	10	0	1274	3	425	431	1	SYSCLK8	425
				8	0.85	1440	30	48	48	3	SYSCLK9	16
				8	0.85	1440	30	48	48	1	SYSCLK10	48
				NA	NA	NA	2	797	797	NA	DDR External Clock	797
				NA	NA	NA	4	398	398	NA	DDR Clock	398

(1) Mean frequency is an average FAPLL output frequency over time when P, N, M, and FREQ are tuned.
(2) Maximum frequency is an instantaneous frequency when FAPLL generates clock at minimum clock cycle.
(3) The values of Pre-Divider (P), Multiplier (N), FREQ, and Post Divider (M) can be tuned in order to obtain different frequencies; however, the software must respect the limitations (and requirements) described in the device-specific data manual for PLL Programming Limits.

Table 1-85. Example for DDR PLL Frequencies for All Speed Grades (Lower DDR Clock Speed)

Input Ref Freq	Pre-Divider	Mult	VCO Output Freq	4 Bits Integer	24 Bits Fract	FAPLL Output	Post Divider	Post Divider Output		PRCM Divider	System Clock Domain	SYSCLK Freq (MHz)
								Mean[1]	Max[2]			
f_r (MHz)	P[3]	N[3]	f_{vco} (MHz)	FREQ[3]		f_s (MHz)	M[3]	f_o (MHz)				
27	1	50	1350	11	0	1080	3	360	365	1	SYSCLK8	360
				9	0	1200	25	48	48	3	SYSCLK9	16
				9	0	1200	25	48	48	1	SYSCLK10	48
				NA	NA	NA	2	675	675	NA	DDR External Clock	675
				NA	NA	NA	4	338	338	NA	DDR Clock	338

(1) Mean frequency is an average FAPLL output frequency over time when P, N, M, and FREQ are tuned.
(2) Maximum frequency is an instantaneous frequency when FAPLL generates clock at minimum clock cycle.
(3) The values of Pre-Divider (P), Multiplier (N), FREQ, and Post Divider (M) can be tuned in order to obtain different frequencies; however, the software must respect the limitations (and requirements) described in the device-specific data manual for PLL Programming Limits.

1.10.3.1.2.1 Steps for Changing DDRPLL Frequency

Refer to the appropriate subsection on how to program the DDRPLL clocks:

- If the DDRPLL is powered down (DDR_PLLEN bit in DDRPLL_CTRL is cleared to 0), follow the procedure in Section 1.10.3.1.2.1.1 to initialize the DDRPLL.
- If the DDRPLL is not powered down (DDR_PLLEN bit in VIDEOPLL_CTRL is set to 1), follow the procedure in Section 1.10.3.1.2.1.2 to initialize the DDRPLL multiplier.
- If the DDRPLL is already running at a desired frequency and you only want to change the SYSCLK FREQ and post dividers, follow the procedure in Section 1.10.3.1.2.1.3 to change the SYSCLK dividers.

1.10.3.1.2.1.1 Initializing and Enabling DDRPLL from Power Down Mode

If the DDRPLL is powered down (DDR_PLLEN bit in DDRPLL_CTRL is cleared to 0), perform the following procedure to initialize the DDRPLL:

1. Clear DDR_BP bit in DDRPLL_CTRL to 0 to put DDRPLL in bypass mode.
2. Set DDR_PLLEN bit in DDRPLL_CTRL to 1 to bring DDRPLL out of power-down mode.
3. Clear PWD_CLK1 to PWD_CLK5 bits in DDRPLL_PWD to 0 to bring individual output clocks of DDRPLL out of power-down mode.
4. Set or clear DDR_LOC_CTL bit in DDRPLL_CTRL to select Digital lock or Analog lock detector.
5. Program the required pre-divider and multiplier values in DDR_P and DDR_N bit fields of DDRPLL_CTRL.
6. If necessary, program DDR_INTFREQx and DDR_FRACFREQx bit fields and set DDR_LDFREQx bit to 1 in DDRPLL_FREQx to load integer and fraction values into DDR Synthesizer x.
7. If necessary, program DDR_MDIVx bit field and set DDR_LDMDIVx bit to 1 in DDRPLL_DIVx to load the Post Divider into DDR Synthesizer x.
8. Wait for PLL to Lock: Poll for DDR_LOCK bit in DDRPLL_CTRL to become 1.
9. If DDRPLL is locked in step 8, then set the DDR_BP bit in DDRPLL_CTRL to 1 to bring DDRPLL out of bypass mode.
10. Set PWRDN bit in DDR_RCD to 1 to bring the Reset Clock Distribution module of DDR out of power-down mode.

Where x = 2, 3 Flying Adder Synthesizer.

1.10.3.1.2.1.2 Changing DDRPLL Multiplier

If the DDRPLL is not powered down (DDR_PLLEN bit in DDRPLL_CTRL is set to 1), perform the following procedure to initialize the DDRPLL:

1. Clear DDR_BP bit in DDRPLL_CTRL to 0 to put DDRPLL in bypass mode.
2. Clear PWD_CLK1 to PWD_CLK5 bits in DDRPLL_PWD to 0 to bring individual output clocks of DDRPLL out of power-down mode.
3. Set or clear DDR_LOC_CTL bit in DDRPLL_CTRL to select Digital lock or Analog lock detector.
4. Program the required pre-divider and multiplier values in DDR_P and DDR_N bit fields of DDRPLL_CTRL.
5. If necessary, program DDR_INTFREQx and DDR_FRACFREQx bit fields and set DDR_LDFREQx bit to 1 in DDRPLL_FREQx to load integer and fraction values into DDR Synthesizer x.
6. If necessary, program DDR_MDIVx bit field and set DDR_LDMDIVx bit to 1 in DDRPLL_DIVx to load the Post Divider into DDR Synthesizer x.
7. Wait for PLL to Lock: Poll for DDR_LOCK bit in DDRPLL_CTRL to become 1.
8. If DDRPLL is locked in step 7, then set the DDR_BP bit in DDRPLL_CTRL to 1 to bring DDRPLL out of bypass mode.
9. Set PWRDN bit in DDR_RCD to 1 to bring the Reset Clock Distribution module of DDR out of power-down mode.

Where x = 2, 3 Flying Adder Synthesizer.

1.10.3.1.2.1.3 Changing DDRPLL SYSCLK Dividers

This section discusses the software sequence to change the SYSCLK dividers.

1. If necessary, program DDR_INTFREQx and DDR_FRACFREQx bit fields and set DDR_LDFREQx bit to 1 in DDRPLL_FREQx to load integer and fraction values into DDR Synthesizer x.
2. If necessary, program DDR_MDIVx bit field and set DDR_LDMDIVx bit to 1 in DDRPLL_DIVx to load the Post Divider into DDR Synthesizer x.
3. If necessary, further divide down the SYSCLKs by programming the respective PRCM registers to get the desired SYSCLK frequencies (see Table 1-81 for supported PRCM divider values).

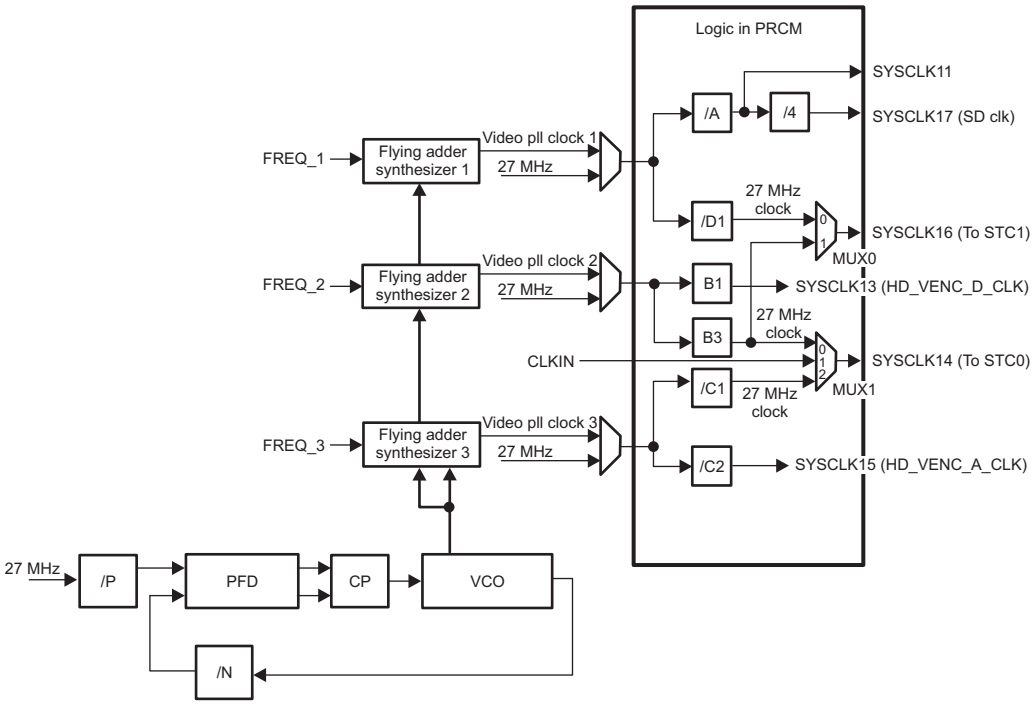
Where x = 2, 3 Flying Adder Synthesizer.

1.10.3.1.3 Video PLL

Figure 1-72 shows the Video PLL structure. There are 3 flying adder synthesizers. The outputs of these synthesizers are muxed with the 27-MHz reference clock to allow its selection during PLL bypass mode. The first synthesizer provides the SYSCLK17 (VENC SD clock) to HDVPSS; this clock is always 54 MHz. The second one provides SYSCLK13 (HD_VENC_D_CLK) for the display subsystem. HD_VENC_D_CLK needs to support 13.5 MHz, 27 MHz, 54 MHz, 74.25 MHz, 148.5 MHz, and 161 MHz frequencies. The third flying adder synthesizer supplies SYSCLK15 (HD_VENC_A_CLK). This clock has similar frequency requirements as HD_VENC_D_CLK.

This devices uses on-chip DCXO using the FA-PLL. Flying adder synthesizer 1, 2, and 3 will be used for this. Software will control the FREQ_1, FREQ_2, and FREQ_3 to track the frequency.

Figure 1-72. Video PLL Structure



Mux0 and Mux1 selects will default to 0.

Table 1-86 shows the supported divide ratios in PRCM module for all dividers in the Video PLL.

Table 1-86. Video PLL Dividers

Divider	Supported Divide Ratios	Default Values
A	1/1, 1/2	1/1
D1	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/8
B1	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/8
B3	1/1,1/2, 1/22	1/22
C1	1/1,1/2, 1/22	1/22
C2	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/4

Table 1-87 lists the IPs that are clocked from Video PLL flying adder synthesizers.

Table 1-87. Video PLL SYSCLK Frequencies and Destination

SYSCLK	Device Speed Range[1]	Maximum Frequency (MHz)[2]	Destination
SYSCLK13	Blank	165	HDVPSS
	2		
	4		
SYSCLK15	Blank	165	HDVPSS
	2		
	4		
SYSCLK17	Blank	54	HDVPSS
	2		
	4		

(1) For more information on the available device speed ranges for each part number, see the device-specific data manual

(2) Maximum frequency of PLL out clocks should meet minimum cycle limits. Refer to the device-specific data manual for PLL Programming Limits.

Table 1-88 shows an example for Video PLL System Clock generation.

Table 1-88. Example for Video PLL Frequencies for All Speed Grades

Input Ref Freq	Pre-Divider	Mult	VCO Output Freq	4 Bits Integer	24 Bits Fract	FAPLL Output	Post Divider	Post Divider Output	PRCM Divider	System Clock Domain	SYSCLK Freq (MHz)
f_r (MHz)	P	N	f_{vco} (MHz)	FREQ		f_s (MHz)	M	f_o (MHz)			
27	2	110	1485	11	0	1080	5	216	4	SYSCLK17	54
									8	SYSCLK16	27
				10	0	1188	2	594	8	SYSCLK13	74.25
									22	SYSCLK14	27
				10	0	1188	2	594	22	SYSCLK14	27
									4	SYSCLK15	148.5

1.10.3.1.3.1 Steps for Changing VIDEOPLL Frequency

Refer to the appropriate subsection on how to program the VIDEOPLL clocks:

- If the VIDEOPLL is powered down (VIDEO_PLLEN bit in VIDEOPLL_CTRL is cleared to 0), follow the procedure in Section 1.10.3.1.3.1.1 to initialize the VIDEOPLL.
- If the VIDEOPLL is not powered down (VIDEO_PLLEN bit in VIDEOPLL_CTRL is set to 1), follow the procedure in Section 1.10.3.1.3.1.2 to initialize the VIDEOPLL multiplier.
- If the VIDEOPLL is already running at a desired frequency and you only want to change the SYSCLK FREQ and post dividers, follow the procedure in Section 1.10.3.1.3.1.3 to change the SYSCLK dividers.

1.10.3.1.3.1.1 Initializing and Enabling VIDEOPLL from Power Down Mode

If the VIDEOPLL is powered down (VIDEO_PLLEN bit in VIDEOPLL_CTRL is cleared to 0), perform the following procedure to initialize the VIDEOPLL:

1. Set VIDEO_BP bit in VIDEOPLL_CTRL to 1 to put VIDEOPLL in bypass mode.
2. Set VIDEO_PLLEN bit in VIDEOPLL_CTRL to 1 to bring VIDEOPLL out of power-down mode.
3. Clear PWD_CLK1 to PWD_CLK3 bits in VIDEOPLL_PWD to 0 to bring individual output clocks of VIDEOPLL out of power-down mode.
4. Set or clear VIDEO_LOC_CTL bit in VIDEOPLL_CTRL to select Digital lock or Analog lock detector.
5. Program the required pre-divider and multiplier values in VIDEO_P and VIDEO_N bit fields of VIDEOPLL_CTRL.
6. If necessary, program VIDEO_INTFREQx and VIDEO_FRACFREQx bit fields and set VIDEO_LDFREQx bit to 1 in VIDEOPLL_FREQx to load integer and fraction values into VIDEO Synthesizer x.
7. If necessary, program VIDEO_MDIVx bit field and set VIDEO_LDMDIVx bit to 1 in VIDEOPLL_DIVx to load the Post Divider into VIDEO Synthesizer x.
8. Wait for PLL to Lock: Poll for VIDEO_LOCK bit in VIDEOPLL_CTRL to become 1.
9. If VIDEOPLL is locked in step 8, then clear the VIDEO_BP bit in VIDEOPLL_CTRL to 0 to bring VIDEOPLL out of bypass mode.

Where x = 1, 2, 3 Flying Adder Synthesizer.

1.10.3.1.3.1.2 Changing VIDEOPLL Multiplier

If the VIDEOPLL is not powered down (VIDEO_PLLEN bit in VIDEOPLL_CTRL is set to 1), perform the following procedure to initialize the VIDEOPLL:

1. Set VIDEO_BP bit in VIDEOPLL_CTRL to 1 to put VIDEOPLL in bypass mode.
2. Clear PWD_CLK1 to PWD_CLK3 bits in VIDEOPLL_PWD to 0 to bring individual output clocks of VIDEOPLL out of power-down mode.
3. Set or clear VIDEO_LOC_CTL bit in VIDEOPLL_CTRL to select Digital lock or Analog lock detector.
4. Program the required pre-divider and multiplier values in VIDEO_P and VIDEO_N bit fields of VIDEOPLL_CTRL.
5. If necessary, program VIDEO_INTFREQx and VIDEO_FRACFREQx bit fields and set VIDEO_LDFREQx bit to 1 in VIDEOPLL_FREQx to load integer and fraction values into VIDEO Synthesizer x.
6. If necessary, program VIDEO_MDIVx bit field and set VIDEO_LDMDIVx bit to 1 in VIDEOPLL_DIVx to load the Post Divider into VIDEO Synthesizer x.
7. Wait for PLL to Lock: Poll for VIDEO_LOCK bit in VIDEOPLL_CTRL to become 1.
8. If VIDEOPLL is locked in step 7, then clear the VIDEO_BP bit in VIDEOPLL_CTRL to 0 to bring VIDEOPLL out of bypass mode.

Where x = 1, 2, 3 Flying Adder Synthesizer.

1.10.3.1.3.1.3 Changing VIDEOPLL SYSCLK Dividers

This section discusses the software sequence to change the SYSCLK dividers.

1. If necessary, program VIDEO_INTFREQx and VIDEO_FRACFREQx bit fields and set VIDEO_LDFREQx bit to 1 in VIDEOPLL_FREQx to load integer and fraction values into VIDEO Synthesizer x.
2. If necessary, program VIDEO_MDIVx bit field and set VIDEO_LDMDIVx bit to 1 in VIDEOPLL_DIVx to load the Post Divider into VIDEO Synthesizer x.
3. If necessary, further divide down the SYSCLKs by programming the respective PRCM registers to get the desired SYSCLK frequencies (see Table 1-86 for supported PRCM divider values).

Where x = 1, 2, 3 Flying Adder Synthesizer.

1.10.3.1.4 Audio PLL

Figure 1-73 shows the structure of the Audio PLL. The Audio PLL has 5 flying adder synthesizers. The outputs of these synthesizers are muxed with the 27-MHz reference clock to allow its selection during PLL bypass mode. The first synthesizer uses the 27-MHz reference clock as a source and provides a 32.768 kHz clock (SYSCLK18) for the SD/SDIO peripheral. A 32.768 kHz clock for the RTC is sourced from the Control Module. This clock has two sources, SYSCLK18 and the 32768 Hz clock from the primary input pin. The Mux2 select defaults to 0 which selects clock generated from FA synthesizer. This clock is also supplied to the watchdog timer. The DMTIMERS (1-7) have the following muxing for the input clock. Select for this mux will default to 1.

Figure 1-73 shows the connection of audio clocks to McASP, McBSP, and HDMI modules. The default select value for these muxes is “0x0”, which selects SYSCLK20.

Figure 1-73. Audio PLL Structure

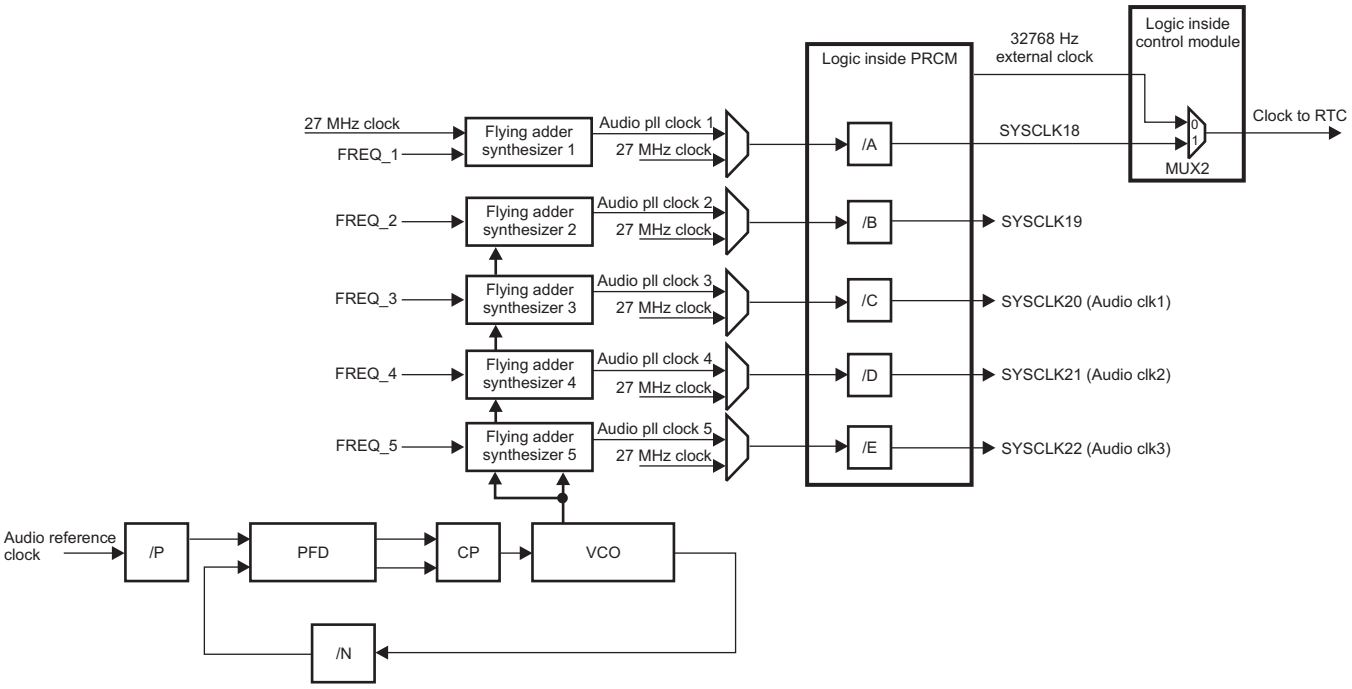


Table 1-89 shows the supported divide ratios in PRCM module for all dividers in the Audio PLL.

Table 1-89. Audio PLL Dividers

Divider	Supported Divider Ratios	Default Value
A	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
B	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
C	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
D	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
E	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1

Table 1-90 lists the IPs that are clocked from Audio PLL flying adder synthesizers.

Table 1-90. Audio PLL SYSCLK Frequencies and Destination

SYSCLK	Device Speed Range[1]	Maximum Frequency[2]	Destination
SYSCLK18	Blank	32 KHz	RTC
	2		
	4		
SYSCLK20	Blank	196 MHz	Audio clock 1
	2		
	4		
SYSCLK21	Blank	196 MHz	Audio clock 2
	2		
	4		
SYSCLK22	Blank	196 MHz	Audio clock 3
	2		
	4		

(1) For more information on the available device speed ranges for each part number, see the device-specific data manual

(2) Maximum frequency of PLL out clocks should meet minimum cycle limits. Refer to the device-specific data manual for PLL Programming Limits.

Table 1-91 shows an example for Audio PLL System Clock generation.

Table 1-91. Example for Audio PLL Frequencies for All Speed Grades

Input Ref Freq	Pre-Divider	Mult	VCO Output Freq	4 Bits Integer	24 Bits Fract	FAPLL Output	Post Divider	Post Divider Output	PRCM Divider	System Clock Domain	SYSCLK Freq (MHz)
f_r (MHz)	P	N	f_{vco} (MHz)	FREQ		f_s (MHz)	M	f_o (MHz)			
378	25	64	967	12	0.096	640	4	160	1	SYSCLK19	160
				9	0.8742857	784	4	196	1	SYSCLK20	196
				8	0.5714282	903	20	45	1	SYSCLK21	45
				11	0	704	20	35	1	SYSCLK22	35

1.10.3.1.4.1 Steps for Changing AUDIOPLL Frequency

Refer to the appropriate subsection on how to program the AUDIOPLL clocks:

- If the AUDIOPLL is powered down (AUDIO_PLLEN bit in AUDIOPLL_CTRL is cleared to 0), follow the procedure in Section 1.10.3.1.4.1.1 to initialize the AUDIOPLL.
- If the AUDIOPLL is not powered down (AUDIO_PLLEN bit in AUDIOPLL_CTRL is set to 1), follow the procedure in Section 1.10.3.1.4.1.2 to initialize the AUDIOPLL multiplier.
- If the AUDIOPLL is already running at a desired frequency and you only want to change the SYSCLK FREQ and post dividers, follow the procedure in Section 1.10.3.1.4.1.3 to change the SYSCLK dividers.

1.10.3.1.4.1.1 Initializing and Enabling AUDIOPLL from Power Down Mode

If the AUDIOPLL is powered down (AUDIO_PLLEN bit in AUDIOPLL_CTRL is cleared to 0), perform the following procedure to initialize the AUDIOPLL:

1. Set AUDIO_BP bit in AUDIOPLL_CTRL to 1 to put AUDIOPLL in bypass mode.
2. Set AUDIO_PLLEN bit in AUDIOPLL_CTRL to 1 to bring AUDIOPLL out of power-down mode.
3. Clear PWD_CLK2 to PWD_CLK5 bits in AUDIOPLL_PWD to 0 to bring individual output clocks of AUDIOPLL out of power-down mode.
4. Set or clear AUDIO_LOC_CTL bit in AUDIOPLL_CTRL to select Digital lock or Analog lock detector.
5. Program the required pre-divider and multiplier values in AUDIO_P and AUDIO_N bit fields of AUDIOPLL_CTRL.
6. If necessary, program AUDIO_INTFREQx and AUDIO_FRACFREQx bit fields and set AUDIO_LDFREQx bit to 1 in AUDIOPLL_FREQx to load integer and fraction values into AUDIO Synthesizer x.
7. If necessary, program AUDIO_MDIVx bit field and set AUDIO_LDMDIVx bit to 1 in AUDIOPLL_DIVx to load the Post Divider into AUDIO Synthesizer x.
8. Wait for PLL to Lock: Poll for AUDIO_LOCK bit in AUDIOPLL_CTRL to become 1.
9. If AUDIOPLL is locked in step 8, then clear the AUDIO_BP bit in AUDIOPLL_CTRL to 0 to bring AUDIOPLL out of bypass mode.

Where x = 1, 2, 3, 4, 5 Flying Adder Synthesizer.

1.10.3.1.4.1.2 Changing AUDIOPLL Multiplier

If the AUDIOPLL is not powered down (AUDIO_PLLEN bit in AUDIOPLL_CTRL is set to 1), perform the following procedure to initialize the AUDIOPLL:

1. Set AUDIO_BP bit in AUDIOPLL_CTRL to 1 to put AUDIOPLL in bypass mode.
2. Clear PWD_CLK2 to PWD_CLK5 bits in AUDIOPLL_PWD to 0 to bring individual output clocks of AUDIOPLL out of power-down mode.
3. Set or clear AUDIO_LOC_CTL bit in AUDIOPLL_CTRL to select Digital lock or Analog lock detector.
4. Program the required pre-divider and multiplier values in AUDIO_P and AUDIO_N bit fields of AUDIOPLL_CTRL.
5. If necessary, program AUDIO_INTFREQx and AUDIO_FRACFREQx bit fields and set AUDIO_LDFREQx bit to 1 in AUDIOPLL_FREQx to load integer and fraction values into AUDIO Synthesizer x.
6. If necessary, program AUDIO_MDIVx bit field and set AUDIO_LDMDIVx bit to 1 in AUDIOPLL_DIVx to load the Post Divider into AUDIO Synthesizer x.
7. Wait for PLL to Lock: Poll for AUDIO_LOCK bit in AUDIOPLL_CTRL to become 1.
8. If AUDIOPLL is locked in step 7, then clear the AUDIO_BP bit in AUDIOPLL_CTRL to 0 to bring AUDIOPLL out of bypass mode.

Where x = 1, 2, 3, 4, 5 Flying Adder Synthesizer.

1.10.3.1.4.1.3 Changing AUDIOPLL SYSCLK Dividers

This section discusses the software sequence to change the SYSCLK dividers.

1. If necessary, program AUDIO_INTFREQx and AUDIO_FRACFREQx bit fields and set AUDIO_LDFREQx bit to 1 in AUDIOPLL_FREQx to load integer and fraction values into AUDIO Synthesizer x.
2. If necessary, program AUDIO_MDIVx bit field and set AUDIO_LDMDIVx bit to 1 in AUDIOPLL_DIVx to load the Post Divider into AUDIO Synthesizer x.
3. If necessary, further divide down the SYSCLKs by programming the respective PRCM registers to get the desired SYSCLK frequencies (see Table 1-89 for supported PRCM divider values).

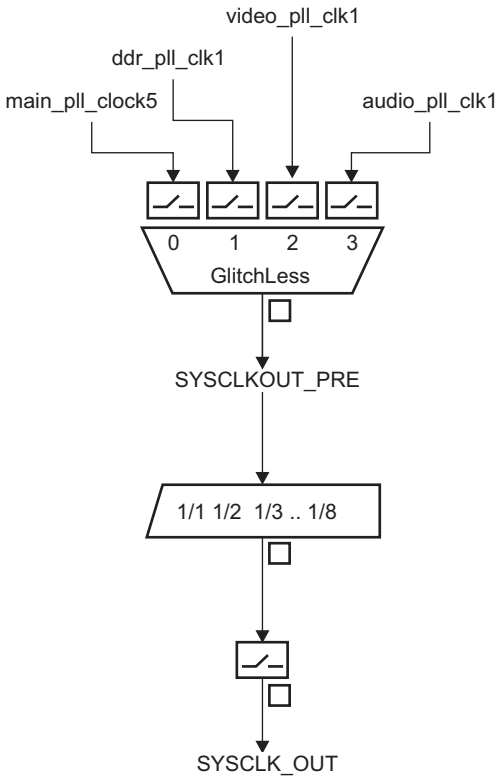
Where x = 1, 2, 3, 4, 5 Flying Adder Synthesizer.

1.10.4 Clock Out

This device has one clockout output pin. Figure 1-74 shows which clocks can be exported out on this pin. The clock sources are muxed, divided and then passed through a clock gate.

As shown in Figure 1-74, there are 4 possible sources for clkout, one clock from each of the 4 PLLs. The selected clock can be further divided by any ratio from 1 to 1/8 before going out on clkout pin. The default selection is to select Main pll clock5, divider set to 1/1 and clock disabled. Refer to the CM_CLKOUT_CTRL register in the *Power, Reset, and Clock Management (PRCM) Module* chapter for more details.

Figure 1-74. Clocks



CAUTION

Glitch-free muxes have an output which is free from any glitches. It requires the previously selected clock and the newly selected clock to both be free-running for switch-over to occur. Switch-over could take 1-2 clock cycles of previously selected and newly selected clock to complete. If newly selected clock is idle a switchover never occurs (previously selected clock continues to pass through the mux). Do not switch to a non existent clock.

1.11 Bus Interconnect

The SoC interconnect is based on a 2-level hierarchical architecture (L3, L4) driven by system performance.

L4 interconnect is based on a fully native OCP infrastructure, directly complying with the OCPIP2.2 reference standard.

1.11.1 Terminology

The following is a brief explanation of some terms used in this document:

Initiator — Module able to initiate read and write requests to the chip interconnect (typically: processors, DMA, etc.).

Target— Unlike an initiator, a target module cannot generate read/write requests to the chip interconnect, but it can respond to these requests. However, it may generate interrupts or a DMA request to the system (typically: peripherals, memory controllers).

Note: A module can have several separate ports; therefore, a module can be an initiator and a target.

Agent— Each connection of one module to one interconnect is done using an agent, which is an adaptation (sometimes configurable) between the module and the interconnect. A target module is connected by a target agent (TA), and an initiator module is connected by an initiator agent (IA).

Interconnect— The decoding, routing, and arbitration logic that enables the connection between multiple initiator modules and multiple target modules connected on it.

Register target (RT)— Special TA used to access the interconnect internal configuration registers.

Data-flow signal— Any signal that is part of a clearly identified transfer or data flow (typically: command, address, byte enables, etc.). Signal behavior is defined by the protocol semantics.

Sideband signal— Any signal whose behavior is not associated to a precise transaction or data flow.

Command Slot— A command slot is a subset of the command list. It is the memory buffer for a single command. A total of 32 command slots exist.

Out-of-band error— Any signal whose behavior is associated to a device error-reporting scheme, as opposed to in-band errors.

Note: Interrupt requests and DMA requests are not routed by the interconnect in the device.

ConnID— Any transaction in the system interconnect is tagged by an in-band qualifier ConnID, which uniquely identifies the initiator at a given interconnect point. A ConnID is transmitted in band with the request and is used for error-logging mechanism.

1.11.2 L3 Interconnect

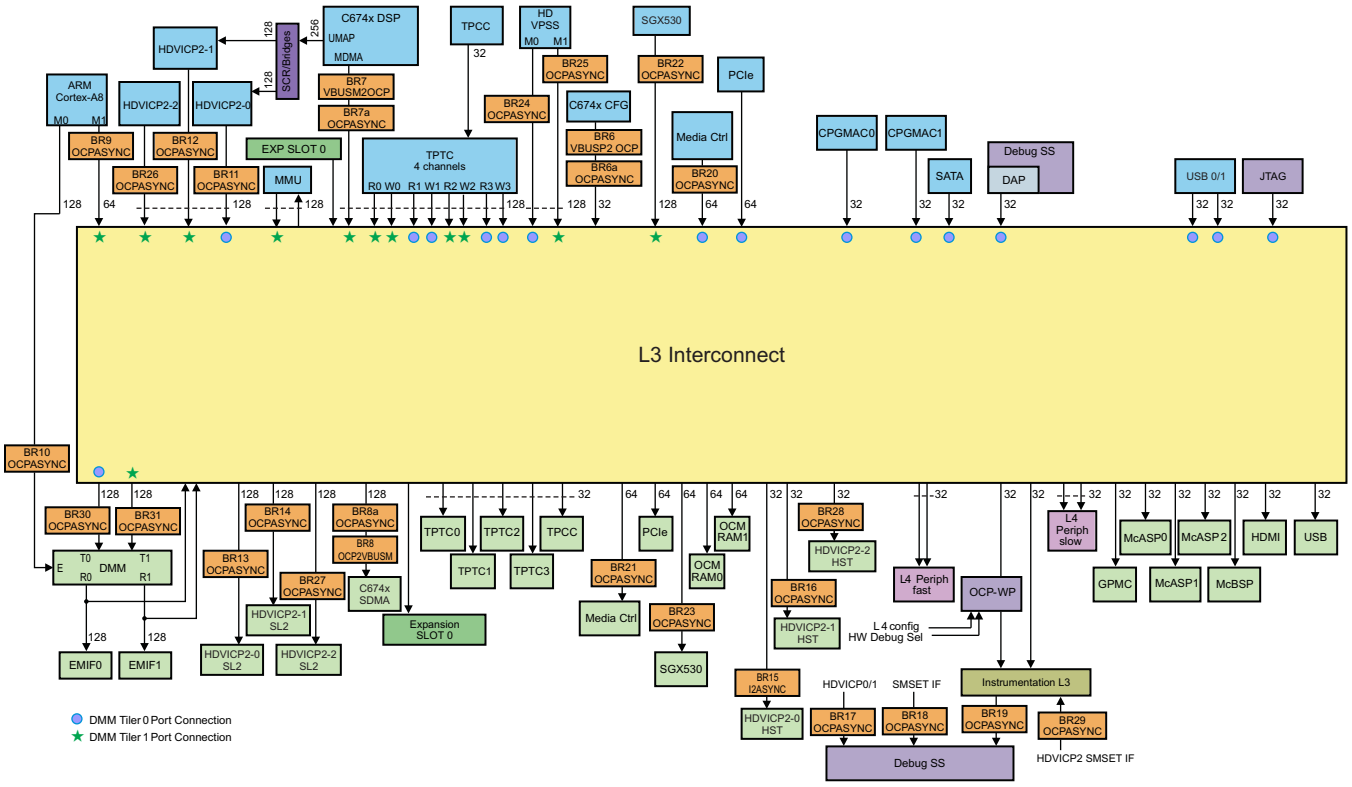
The L3 high-performance interconnect is based on a Network On Chip (NoC) interconnect infrastructure. The NoC uses an internal packet-based protocol for forward (read command, write command with data payload) and backward (read response with data payload, write response) transactions. All exposed interfaces of this NoC interconnect, both for Targets and Initiators; comply with the OCPIP2.2 reference standard.

1.11.2.1 Topology

The L3 topology is driven by performance requirements, bus types, and clocking structure. The main L3 paths are shown in Figure 1-75. Arrows indicate the master/slave relationship not data flow.

As Figure 1-75 shows, there are four clock domains in the NOC corresponding to fast (L3F), median (L3M), and slow (L3S) initiator and target port speeds and a special domain (L3P) for EMIF probes.

Figure 1-75. L3 Interconnect Block Diagram



1.11.2.2 L3 Port Mapping

Each initiator and target core is connected to the L3 interconnect through an NIU. The NIUs act as entry and exit points to the L3 Network on Chip – converting between the IP's OCP protocol and the NOC's internal protocol, and also include various programming registers. All ports are single threaded with tags used to enable pipelined transactions. The SoC interconnect includes:

Initiator Ports:

- Host ARM Subsystem (HASS) 64-bit initiator Port 1 (HASS Port 0 is a special case 128-bit initiator which bypasses the L3.)
- C674x DSP Subsystem MDMA 128-bit initiator port.
- C674x DSP Subsystem CFG 32-bit initiator port.
- HD Video Processing Subsystem (HD-VPSS) 128-bit initiator Port 0 and Port 1.
- HDVICP2-0 128-bit initiator port.
- HDVICP2-1 128-bit initiator port.
- HDVICP2-2 128-bit initiator port.
- MMU 128-bit initiator port.
- 4 TPTC 128-bit read initiator ports.
- 4 TPTC 128-bit write initiator ports.
- Expansion Slot 0 128-bit initiator port.
- CPGMAC0 32-bit initiator port.
- CPGMAC1 32-bit initiator port.
- SATA 32-bit initiator port.
- PCIe 64-bit initiator port.
- Debug Subsystem (DAP) 32-bit initiator port.
- Graphics accelerator (SGX530) 128-bit initiator port.
- USB 32-bit CPPI DMA initiator port.
- USB 32-bit Queue Manager initiator port.
- JTAG Interface 32-bit initiator port.

Target Ports:

- DMM Tiler0 128-bit target port.
- DMM Tiler1 128-bit target port.
- HDVICP2-0 SL2 128-bit target port.
- HDVICP2-1 SL2 128-bit target port.
- HDVICP2-2 SL2 128-bit target port.
- C674x DSP Subsystem SDMA 128-bit target port.
- 4 TPTC CFG 32-bit target ports.
- TPCC CFG 32-bit target port.
- Expansion Slot 0 128-bit target port.
- System MMU 128-bit target port.
- Graphics accelerator (SGX530) 64-bit target port.
- PCIe 64-bit target port.
- OCM RAM0 64-bit target port.
- OCM RAM1 64-bit target port.
- 2 L4_Fast peripheral 32-bit target ports.
- HDVICP2-0 Host1 Control 32-bit target port.
- HDVICP2-1 Host1 Control 32-bit target port.
- HDVICP2-2 Host1 Control 32-bit target port.

- Instrumentation L3 32-bit target port.
- 2 L4_Slow peripheral 32-bit target ports.
- GPMC 32-bit target port.
- McASP0 32-bit target port.
- McASP1 32-bit target port.
- McASP2 32-bit target port.
- McBSP 32-bit target port.
- HDMI 32-bit target port.
- USB 32-bit target port.

1.11.2.3 Connection Identification

Each L3 initiator includes a unique 6-bit master connection identifier (MConnID) that is used to identify the source of a transfer request. The full 6-bit MConnID is used for debug purposes, otherwise only the 4 MSBs are used to distinguish unique masters.

Table 1-92. Device MConnID Assignment

Initiator	4-bit MConnID	6-bit MConnID	Instrumentation	Comment
Host ARM	0	0	Software	
HDVICP2-0 SWINST IC0		2h	Software	Direct connect to DebugSS
HDVICP2-0 SWINST IC1		3h	Software	Direct connect to DebugSS
DAP	1h	4h	Software	
JTAG Interface	1h	5h	Software	
HDVICP2-1 SWINST IC0		6h	Software	Direct connect to DebugSS
HDVICP2-1 SWINST IC1		7h	Software	Direct connect to DebugSS
C674x DSP MDMA	2h	8h	Software	
C674x DSP CFG	2h	9h	Software	
System MMU	2h	Ah	Software	
HDVICP2-2 SWINST IC0		Ch	Software	Direct connect to DebugSS
HDVICP2-2 SWINST IC1		Dh	Software	Direct connect to DebugSS
TPTC0 Read	6h	18h	Software	
TPTC1 Read	6h	19h	Software	
TPTC2 Read	6h	1Ah	Software	
TPTC3 Read	6h	1Bh	Software	
TPTC0 Write	7h	1Ch	Software	
TPTC1 Write	7h	1Dh	Software	
TPTC2 Write	7h	1Eh	Software	
TPTC3 Write	7h	1Fh	Software	
SGX530	8h	20h	0	
Watchpoint TR		21h	Hardware	Direct connect to DebugSS
Watchpoint DMA Prof		22h	Hardware	Direct connect to DebugSS
HD VPSS 0	9h	24h	0	
HD VPSS 1	9h	25h	0	
HDVICP2-0	Ah	28h	0	

Table 1-92. Device MConnID Assignment (continued)

Initiator	4-bit MConnID	6-bit MConnID	Instrumentation	Comment
HDVICP2-0 SMSET		29h	Hardware	Direct connect to DebugSS
HDVICP2-1 SMSET		2Ah	Hardware	Direct connect to DebugSS
HDVICP2-2 SMSET		2Bh	Hardware	Direct connect to DebugSS
HDVICP2-1	Bh	2Ch	0	
HDVICP2-2	Bh	2Dh	0	
CPGMAC0	Ch	30h	0	
CPGMAC1	Ch	31h	0	
USB DMA	Dh	34h	0	
USB QMGR	Dh	35h	0	
SATA	Eh	39h	0	
PCIe	Eh	3Ah	0	
Expansion Port	Eh	3Bh	0	
Stat Collctr 0	Fh	3Ch	Hardware	
Stat Collctr 1	Fh	3Dh	Hardware	
Stat Collctr 2	Fh	3Eh	Hardware	
Stat Collctr 3	Fh	3Fh	Hardware	
DMM Page Table	Fh		0	Connects only to EMIFs

1.11.2.4 Interconnect Requirements

The required L3 connections between SoC bus masters and slave ports are shown in the tables below. The L3 interconnect will return an address-hole error if any initiator attempts to access a target to which it has no connection.

Table 1-93. L3 Master/Slave Connectivity (Table 1 of 3)[1]

Masters	Slaves												
	Master ID	System MMU	DMM Tiler/ Lisa0	DMM Tiler/ Lisa1	DMM ELLA	GPMC	SGX530	C674x DSP_ SDMA	PCIe Slave	McASPs	McBSP	Target	HDMI Audio
M1 (128-bit)	0				R								
M2 (64-bit)	0			R		R	R	R	R	R	R	R	R
C674x DSP MDMA	8h	R											
System MMU	Ah			R		R				R	R	R	R
CX674x DSP CFG	9h												
HDVICP2-0 VDMA	28h		R										
HDVICP2-1 VDMA	2Ch			R									
HDVICP2-2 VDMA	2Dh			R									
DSS Mstr0	24h		R					R					
DSS Mstr1	25h			R				R					
SGX530 BIF	20h			R		R		R					
SATA	39h		R			R		R					
CPGMAC0 Rx/Tx	30h		R					R					
CPGMAC1 Rx/Tx	31h		R					R					
USB DMA	34h		R					R					
USB Queue Mgr	35h		R			R		R					
PCIe	3Ah		R			R		R					
EMU (DAP)	4h		R			R	R	R	R	R	R	R	R
IEEE1500	5h		R			R	R	R	R	R	R	R	R
Expansion Port	3Bh	R1		R		R		R					
TPTC0 RD	18h	R1		R		R	R	R	R	R	R	R	R
TPTC0 WR	19h	R1		R		R	R	R	R	R	R	R	R
TPTC1 RD	1Ah		R			R	R	R	R	R	R	R	R
TPTC1 WR	1Bh		R			R	R	R	R	R	R	R	R
TPTC2 RD	1Ch			R		R	R	R	R	R	R	R	R
TPTC2 WR	1Dh			R		R	R	R	R	R	R	R	R
TPTC3 RD	1Eh		R			R	R	R	R	R	R	R	R
TPTC3 WR	1Fh		R			R	R	R	R	R	R	R	R

(1) R : Required path.
R1 : Selectable path based on 33rd address bit from Control Module register for System MMU accessible targets. Non-MMU Accessible targets (such as C674x DSP SDMA) will always be direct mapped.

Table 1-94. L3 Master/Slave Connectivity (Table 2 of 3)

Masters	Slaves											
	Master ID	L4 HS Periph Port 0	L4 HS Periph Port 1	L4 STD Periph Port 0	L4 STD Periph Port 1	TPTC0 - 3 CFG	TPCC	OCM RAM0/ RAM1	INSTR Port	Expansion Port	USB CFG	NOC Regs
M1 (128-bit)	0											
M2 (64-bit)	0	R		R		R	R	R	R	R	R	R
C674x DSP MDMA	8h											
System MMU	Ah			R			R	R		R		
C674x DSP CFG	9h			R		R	R					
HDVICP2-0 VDMA	28h							R				
HDVICP2-1 VDMA	2Ch							R				
HDVICP2-2 VDMA	2Dh							R				
DSS Mstr0	24h							R				
DSS Mstr1	25h							R				
SGX530 BIF	20h							U				
SATA	39h							R				
CPGMAC0 Rx/Tx	30h							R				
CPGMAC1 Rx/Tx	31h							R				
USB DMA	34h											
USB Queue Mgr	35h							R				
PCIe	3Ah							R				
EMU (DAP)	4h		R		R	R	R	R	R	R	R	R
IEEE1500	5h		R		R	R	R	R	R	R	R	R
Expansion Port	3Bh							R		R		
TPTC0 RD	18h		R		R		R	R		R	R	
TPTC0 WR	19h		R		R		R	R	R	R	R	
TPTC1 RD	1Ah	R		R			R	R		R	R	
TPTC1 WR	1Bh	R		R			R	R	R	R	R	
TPTC2 RD	1Ch		R		R		R	R		R	R	
TPTC2 WR	1Dh		R		R		R	R		R	R	
TPTC3 RD	1Eh	R		R			R	R		R	R	
TPTC3 WR	1Fh	R		R			R	R		R	R	

Table 1-95. L3 Master/Slave Connectivity (Table 3 of 3)[1]

Masters	Slaves						
	Master ID	HDVICP2-0 SL2	HDVICP2-1 SL2	HDVICP2-2 SL2	HDVICP2-0 Host	HDVICP2-1 Host	HDVICP2-2 Host
M1 (128-bit)	0						
M2 (64-bit)	0	R	R	R	R	R	R
C674x DSP MDMA	8h						
System MMU	Ah	R	R	R	R	R	R
CX674x DSP CFG	9h						
HDVICP2-0 VDMA	28h						
HDVICP2-1 VDMA	2Ch						
HDVICP2-2 VDMA	2Dh						
DSS Mstr0	24h	R	R	R			
DSS Mstr1	25h	R	R	R			
SGX530 BIF	20h						
SATA	39h						
CPGMAC0 Rx/Tx	30h						
CPGMAC1 Rx/Tx	31h						
USB DMA	34h						
USB Queue Mgr	35h						
PCIe	3Ah						
EMU (DAP)	4h	R	R	R	R	R	R
IEEE1500	5h	R	R	R	R	R	R
Expansion Port	3Bh	R	R	R			
TPTC0 RD	18h	R	R	R	R	R	R
TPTC0 WR	19h	R	R	R	R	R	R
TPTC1 RD	1Ah	R	R	R	R	R	R
TPTC1 WR	1Bh	R	R	R	R	R	R
TPTC2 RD	1Ch	R	R	R	R	R	R
TPTC2 WR	1Dh	R	R	R	R	R	R
TPTC3 RD	1Eh	R	R	R	R	R	R
TPTC3 WR	1Fh	R	R	R	R	R	R

(1) R : Required path.

1.12 Inter-Processor Communication

This SoC is a heterogeneous multi-core device, which requires software to efficiently manage and communicate between the cores. The following are the main features that need to be implemented by any such software:

1. Device management of the slave processors from the host processor.
2. Inter-processor communication between the cores for transfer and exchange of information between them.

In this SoC, the host processor is usually the Cortex-A8. This processor is responsible for boot-loading the slave processors (Video-Media Controller, VPSS-Media Controller, C674x DSP).

The HDVICP2 cores are managed by the Video-Media Controller.

Boot-loading includes power management of the slaves (power-up/down and other power management), reset control (reset/release of the slave processor) and setting the entry point of the slave executable into the appropriate register.

For implementing efficient Inter-Processor Communication between the multiple cores on the device, certain hardware features are provided:

- Mailbox interrupts
- Hardware Spinlocks

1.12.1 Reset Requirements

This SoC has a power on reset (POR) and warm reset. For the POR reset, the A8 is taken out of reset and it boots from its boot ROM. Once booted, the A8 will boot load the C674x DSP, VPSS-Media Controller and Video-Media Controller. The Video-Media Controller will be responsible to boot the 3 HDVICP2s when needed.

1.12.2 Features

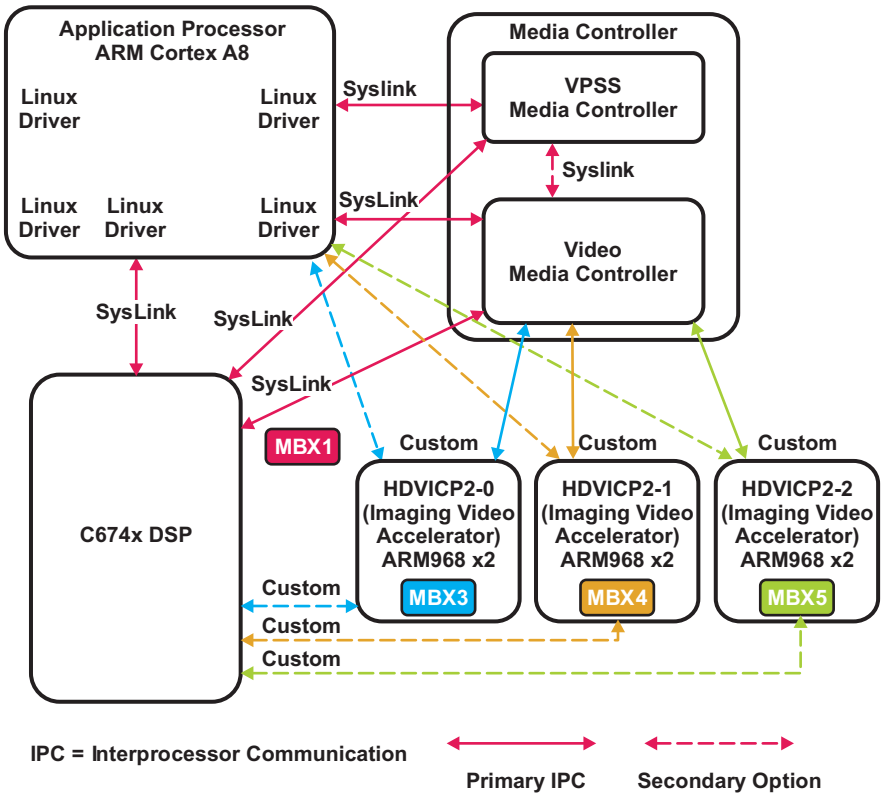
Different methods of IPC are used between the various processing elements in the SoC:

- Custom IPC – This is an IPC that doesn't use any standard IPC software.
- SysLink – This is a new implementation of DSP/BIOS Link that allows communication between the processors in the SoC. SysLink includes ProcMgr module for device management of the slave cores (Video-Media Controller, VPSS-Media Controller, C674x DSP). The Notify method is the API for responding to interrupts. SysLink also incorporates full IPC features including MessageQ, RingIO, and FrameQ.

1.12.3 Overview and Strategy

Figure 1-76 shows the processing elements that are part of the SoC.

Figure 1-76. IPC Overview Diagram



The SoC contains Mailbox and Spinlock hardware to facilitate the IPC mechanism.

Mailboxes provide a mechanism for one processor to write a value to a register and send an interrupt to another processor. One system level mailbox is provided for the IPCs between the A8, the C674x DSP, and the media controllers. The HDVICP2s each have their own mailbox within their IP module.

The SoC shall provide Spinlocks to facilitate Mutexes for access to shared resources in the system.

Mailboxes and Spinlocks will be discussed in more detail later in this document.

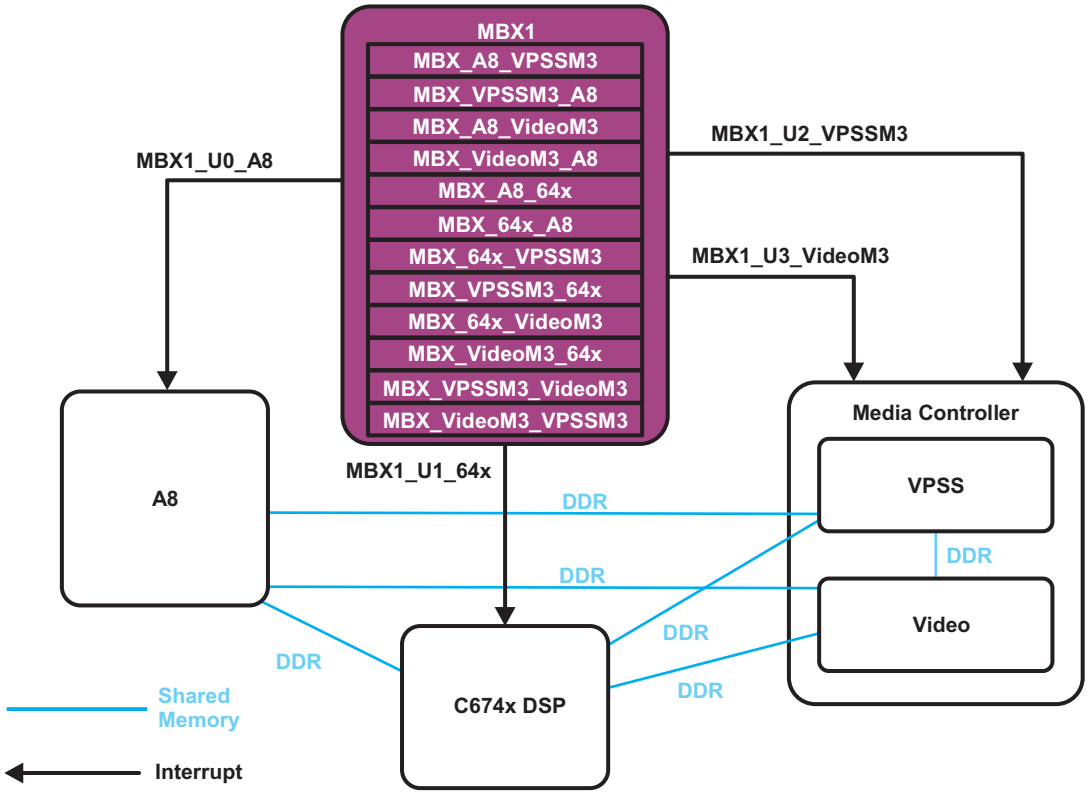
It is intended that the following IPC mechanisms shall be used between various processors:

- SysLink / IPC
  - A8 to C674x DSP
  - A8 to Video-Media Controller
  - A8 to VPSS-Media Controller
  - Video-Media Controller to VPSS-Media Controller
  - C674x DSP to Video-Media Controller
  - C674x DSP to VPSS-Media Controller
- Custom
  - Video-Media Controller to HDVICP2s

1.12.3.1 System IPCs

Figure 1-77 depicts how Mailbox 1 (MBX1) will be used for system IPCs. 4 interrupts are generated from the mailbox that allows the A8, C674x DSP, VPSS-Media Controller, and Video-Media Controller to communicate. DDR is used as the shared memory interface. An IPC can be created between any of the processors listed. The A8 will communicate to the C674x DSP, Video-Media Controller and VPSS-Media Controller using SysLink.

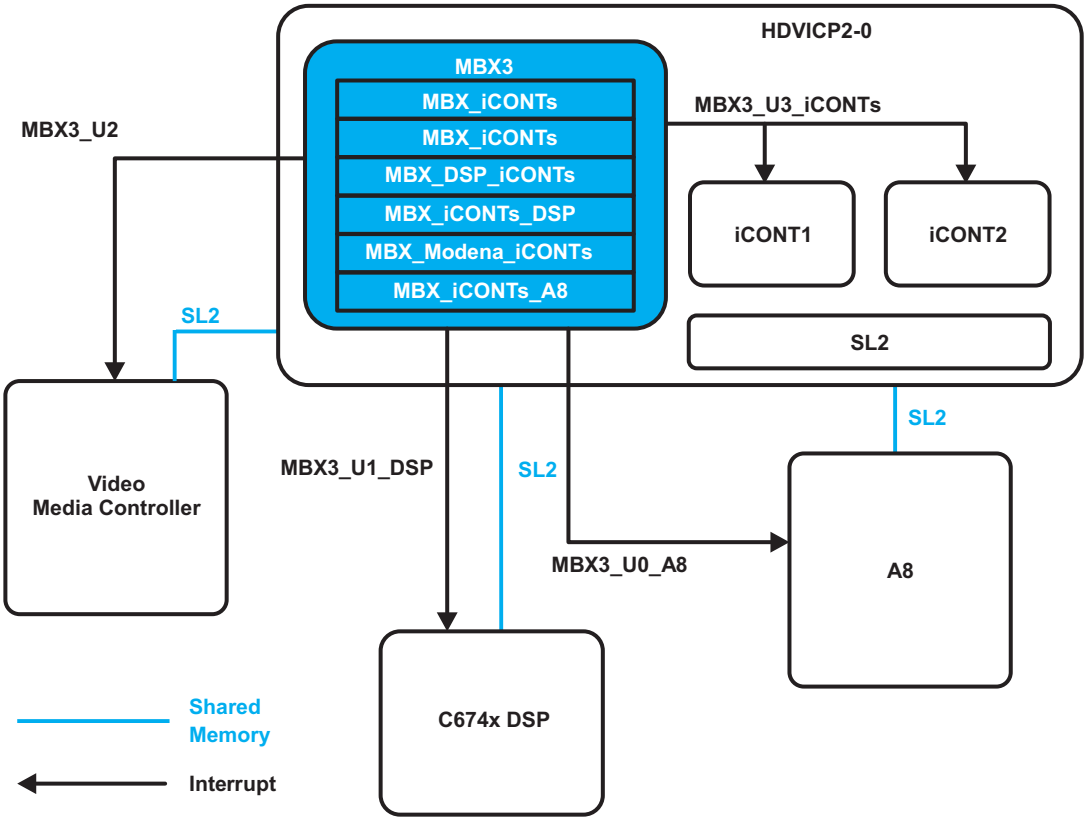
Figure 1-77. System IPCs



1.12.3.2 HDVICP2-0 IPC

Figure 1-78 shows how Mailbox 3 (MBX3) is used for the IPC between HDVICP2-0 and the other processors that may control HDVICP2-0. Only one processor is chosen to control the HDVICP2 at a time. The Video-Media Controller is the primary processor that controls the HDVICP2s. Interrupts from the mailbox are provided as shown in Figure 1-78. IPCs use the Shared Level 2 (SL2) memory that is part of the HDVICP2-0 module. This is necessary since the HDVICP2 processors cannot access DDR or the L3/L4 switch fabric. The HDVICP2 software resides beneath the xDM API.

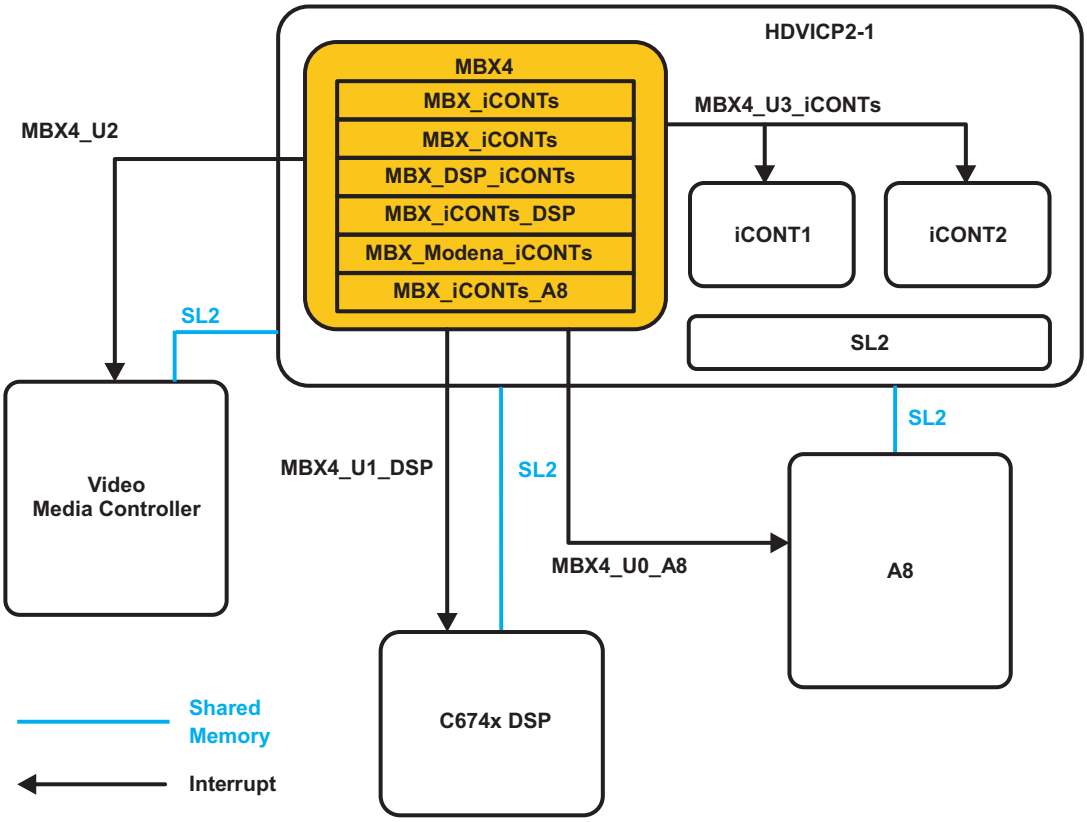
Figure 1-78. HDVICP2-0 IPC



1.12.3.3 HDVICP2-1 IPC

Figure 1-79 shows how Mailbox 4 (MBX4) is used for the IPC between HDVICP2-1 and the other processors that may control HDVICP2-1. HDVICP2-1 is controlled in the same way as HDVICP2.-0

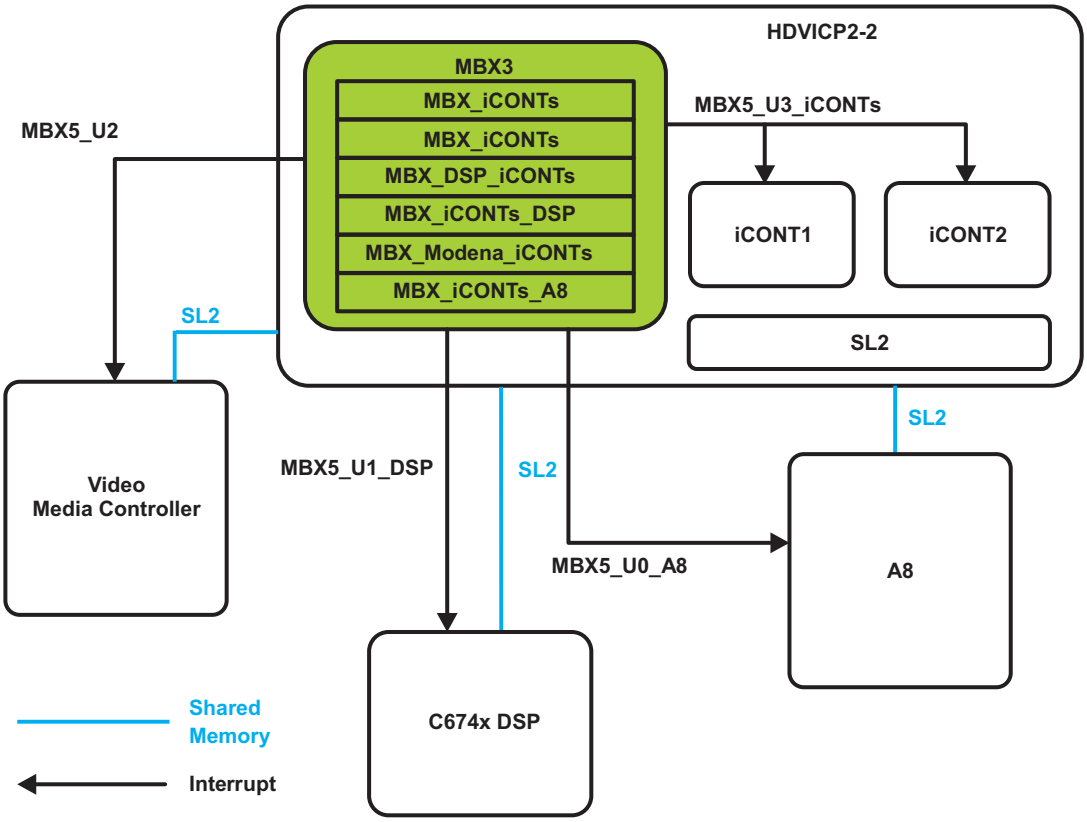
Figure 1-79. HDVICP2-1 IPC



1.12.3.4 HDVICP2-2 IPC

Figure 1-80 shows how Mailbox 5 (MBX5) is used for the IPC between HDVICP2-2 and the other processors that may control HDVICP2-2. HDVICP2-2 is controlled in the same way as HDVICP2-0.

Figure 1-80. HDVICP2-2 IPC



1.12.4 IPC Component Configuration

1.12.4.1 Shared Memory

Shared memory will be used as necessary for the IPCs of the system. Care must be taken to ensure that cache coherency is maintained.

1.12.4.2 IPC Interrupt Module - Mailboxes

Interrupts used by IPCs are generated from Mailbox modules.

All mailboxes use interrupts with both an active high level and active low level. The appropriate type of interrupt is connected to each processor based on the characteristics of that processor.

Each mailbox shall provide 2 messages per mailbox submodule.

1.12.4.3 Hardware Spinlocks

Spinlocks are pared down semaphores designed to provide direct access mutex support without the need for interrupt generator or queues. Spinlocks can be used very effectively as a mechanism to protect data structures in shared memory space that might be accessed by multiple processing cores simultaneously. They can be utilized if the following conditions apply:

1. The time to hold the lock is predictable and small. (How small is a hardware/software system design consideration. For example a maximum hold time of less than 200 CPU cycles may be acceptable.)
2. The locking task cannot be preempted or suspended or interrupted while holding the lock. (This would make the hold time large and unpredictable.)
3. The lock is lightly contended, that is, the chance of any other process (or processor) trying to acquire the lock while it is held is small.

If these conditions hold, then the locking code can retry a failed attempt to acquire the lock until success.

The spin lock unit is responsible for providing hardware assistance for synchronizing the processes running on multiple processors in the device:

- The application processor (ARM Cortex-A8)
- C674x DSP

Note that the HDVICP2s can not access the L3/L4 switch fabric, so IPCs to the HDVICP2s can not use spinlocks.

Hardware spinlocks are used by SysLink and IPC products to provide multi-core mutual-exclusion between the processors on this SoC. This is used to protect access to control structures in shared memory. In addition spinlocks can provide a mutual exclusion mechanism that could be used in a system level resource manager.

This SoC has 64 spinlocks for the system.

Table 1-96. Hardware Spinlock Configuration

Hardware Spinlocks	Name	Configuration
0..63	SPINLOCK_0 .. SPINLOCK_63	No configurable options

1.13 Mailbox

1.13.1 Overview

Communication between the on-chip processors of the device uses a queued mailbox-interrupt mechanism.

The queued mailbox-interrupt mechanism allows the software to establish a communication channel between two processors through a set of registers and associated interrupt signals by sending and receiving messages (mailboxes).

There are two mailbox module instances in the device:

- System mailbox - used for Cortex-A8 microprocessor unit (Cortex-A8 MPU) and digital signal processor (DSP) communications.
- HDVICP2 mailbox - used for communication between one internal to the HDVICP2 subsystem user (imaging controller 1 - iCont1, or imaging controller 2 - iCont2) and three external to the HDVICP2 subsystem users (Cortex-A8 MPU, DSP, and Video-Media Controller). This communication is insured through three pairs of mailboxes.

The mailbox module includes the following features:

- Four users for the system mailbox instance (Cortex-A8 MPU, DSP and Media Controller Subsystem)/four users for the HDVICP2 mailbox instance (iCont1/iCont2, Cortex-A8 MPU, DSP, and Video-Media Controller MPU)
- Four mailbox message queues for the system mailbox instance/six mailbox message queues for the HDVICP2 mailbox instance
- Flexible assignment of receiver and sender for each mailbox through interrupt configuration
- Four interrupts (one per user) for the system mailbox instance/four interrupts (one per user) for the HDVICP2 mailbox instance
- 32-bit message width
- Four-message FIFO depth for each message queue
- Message reception and queue-not-full notification using interrupts
- Support of 16-/32-bit addressing scheme
- Power management support

1.13.2 System Mailbox Integration

This section describes the system mailbox integration in the device, including information about clocks, resets, and hardware requests. Figure 1-81 shows the system mailbox integration.

Table 1-97 through Table 1-99 summarize the system mailbox integration in the device.

Figure 1-81. System Mailbox Integration

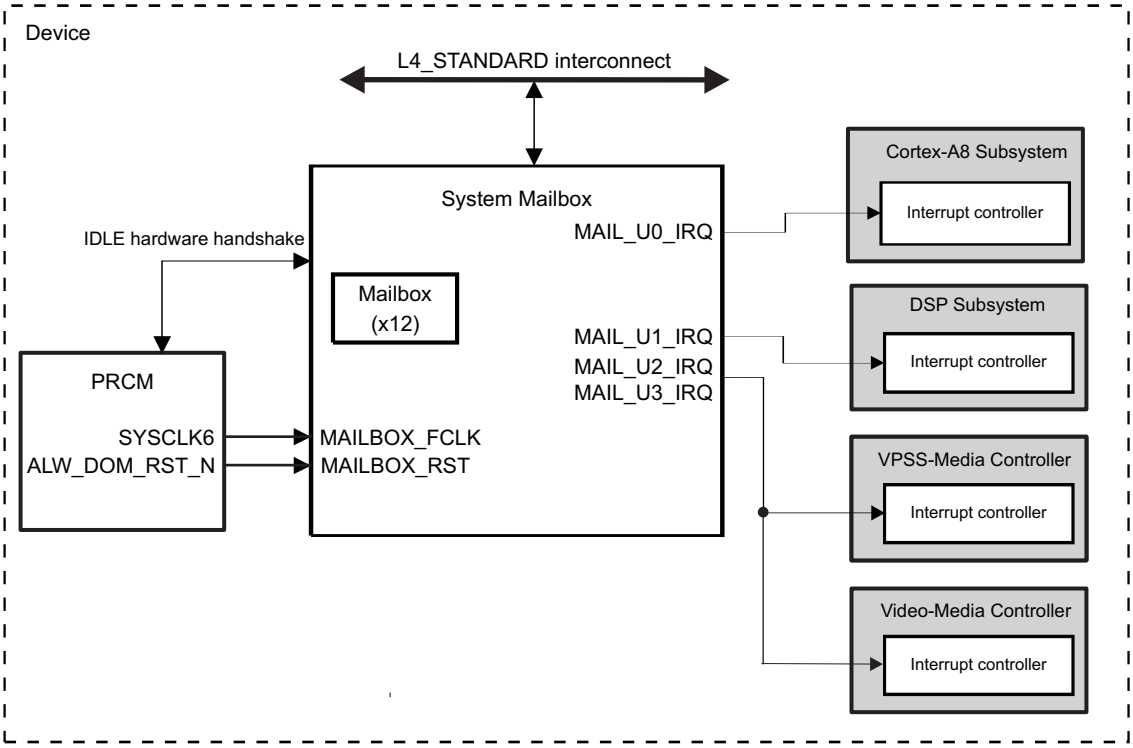


Table 1-97. Integration Attributes

Module Instance	Attributes	
	Power Domain	Interconnect
SYSTEM_MAILBOX	PD_ALWAYS_ON	L4_STANDARD

Table 1-98. Clocks and Resets

Clocks				
Module Instance	**Destination Signal Name**	**Source Signal**	**Source**	**Description**
SYSTEM_MAILBOX	MAILBOX_FCLK	SYSCLK6	PRCM	Mailbox interface clock. This clock is used for all interface and functional operations.
Resets				
SYTEM_MAILBOX	MAILBOX_RST	ALW_DOM_RST_N	PRCM	Mailbox hardware reset. This reset is asynchronously applied to the Mailbox registers

Table 1-99. Hardware Requests

Module Instance	Source Signal Name	Destination Signal	Destination	Description
SYSTEM_MAILBOX	MAIL_U0_IRQ	A_IRQ_77	Cortex-A8	System Mailbox user 0 interrupt
	MAIL_U1_IRQ	D_IRQ_56	DSP	System Mailbox user 1 interrupt

1.13.3 Functional Description

This device has the following Mailbox instances:

- System Mailbox
- HDVICP2-0 Mailbox
- HDVICP2-1 Mailbox
- HDVICP2-2 Mailbox

Table 1-100 shows Mailbox Implementation in this device, where u is the user number and m is the mailbox number.

Table 1-100. Mailbox Implementation

Mailbox Type	User Number(u)	Mailbox Number(m)	Messages per Mailbox
System Mailbox	0 to 3	0 to 11	4
HDVICP2-0 Mailbox	0 to 3	0 to 5	4
HDVICP2-1Mailbox	0 to 3	0 to 5	4
HDVICP2-2 Mailbox	0 to 3	0 to 5	4

The mailbox module provides a means of communication through message queues among the users (depending on the mailbox module instance). The individual mailbox modules (12 for the system mailbox instance, 6 for each HDVICP2 mailbox instance), or FIFOs, can associate (or de-associate) with any of the processors using the MAILBOX_IRQENABLE_SET_u (or MAILBOX_IRQENABLE_CLR_u) register.

CAUTION

For each HDVICP2 mailbox instance, communication is possible only if one of the users is iCont1 or iCont2.

The system mailbox module includes the following user subsystems:

- User 0: Cortex-A8 MPU subsystem (u = 0)
- User 1: DSP subsystem (u = 1)

Each HDVICP2 mailbox module includes the following user subsystems, where x=0,1,2.:

- User 0: Cortex-A8 MPU subsystem (u = 0)
- User 1: DSP subsystem (u = 1)
- User 3: HDVICP2 subsystem - only available to the HDVICP2 mailbox instance (u = 3)

Each user has a dedicated interrupt signal from the corresponding mailbox module instance and dedicated interrupt enabling and status registers. Each MAILBOX_IRQSTATUS_RAW_u/MAILBOX_IRQSTATUS_CLR_u interrupt status register corresponds to a particular user.

For the system mailbox instance, a user can query its interrupt status register through the L4_STANDARD (L4 STD) interconnect.

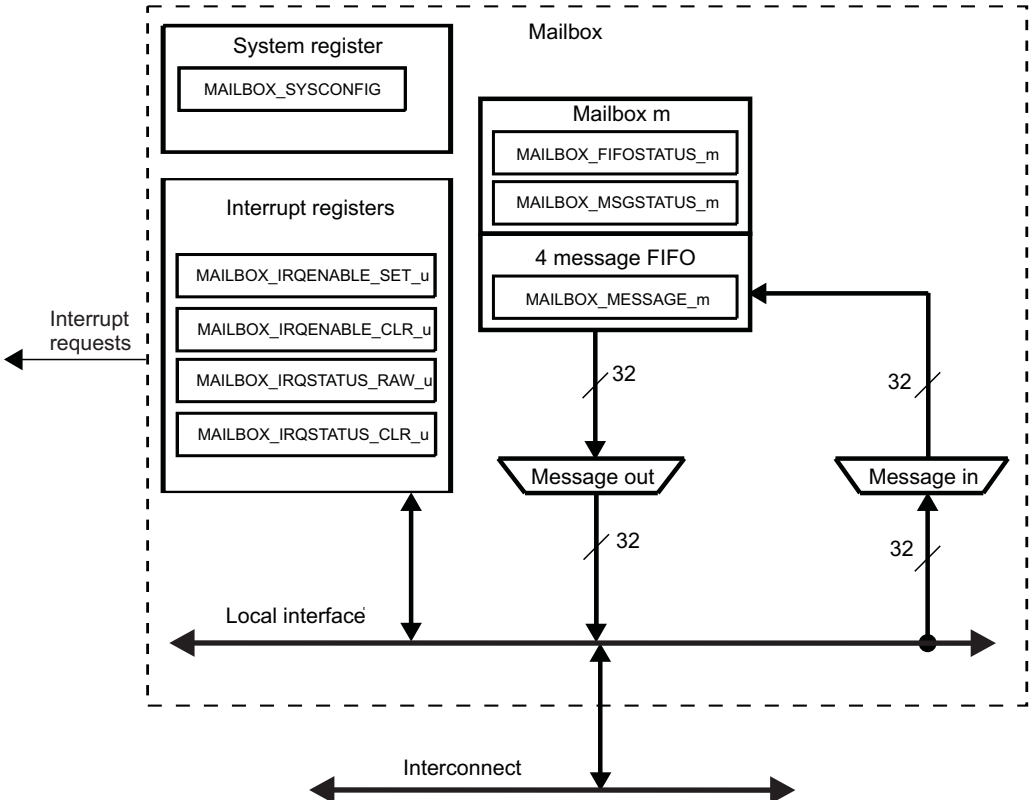
For each HDVICP2 mailbox instance, a user can query its interrupt status register as follows:

- Cortex-A8 MPU and DSP - through the L3 interconnect
- iCont1/iCont2 and DSP - private access (directly through the HDVICP2 and DSP local interconnect respectively)

1.13.3.1 Block Diagram

Figure 1-82 shows the mailbox block diagram.

Figure 1-82. Mailbox Block Diagram



1.13.3.2 Software Reset

The mailbox module supports a software reset through the MAILBOX_SYSCONFIG[0].SOFTRESET bit. Setting this bit to 1 enables an active software reset that is functionally equivalent to a hardware reset. Reading the MAILBOX_SYSCONFIG[0] SOFTRESET bit gives the status of the software reset:

- Read 1: the software reset is on-going.
- Read 0: the software reset is complete.

The software must ensure that the software reset completes before doing mailbox operations.

1.13.3.3 Power Management

Table 1-101 describes power-management features available for the mailbox module.

Table 1-101. Local Power Management Features

Feature	Registers	Description
Clock autogating	NA	Feature not available
Slave idle modes	MAILBOX_SYSCONFIG[3:2].SIDLEMODE	Force-idle, no-idle and smart-idle modes are available
Clock activity	NA	Feature not available
Master standby modes	NA	Feature not available
Global wake-up enable	NA	Feature not available
Wake-up sources enable	NA	Feature not available

The mailbox module can be configured using the MAILBOX_SYSCONFIG[3:2] SIDLEMODE bit field to one of the following acknowledgment modes:

- Force-idle mode (SIDLEMODE = 0x0): The mailbox module immediately enters the idle state on receiving a low-power-mode request from the PRCM module. In this mode, the software must ensure that there are no asserted output interrupts before requesting this mode to go into the idle state.
- No-idle mode (SIDLEMODE = 0x1): The mailbox module never enters the idle state.
- Smart-idle mode (SIDLEMODE = 0x2): After receiving a low-power-mode request from the PRCM module, the mailbox module enters the idle state only after all asserted output interrupts are acknowledged.

1.13.3.4 Interrupt Requests

An interrupt request allows the user of the mailbox to be notified when a message is received or when the message queue is not full. There is one interrupt per user. Table 1-102 lists the event flags, and their mask, that can cause module interrupts.

Table 1-102. Interrupt Events

Non-Maskable Event Flag[1]	Maskable Event Flag	Event Mask Bit	Event Unmask Bit	Description
MAILBOX_IRQSTATUS_RAW_u[0+m*2].NEWMSGSTATUSUUMBm	MAILBOX_IRQSTATUS_CLR_u[0+m*2].NEWMSGSTATUSUUMBm	MAILBOX_IRQENABLE_CLR_u[0+m*2].		
MAILBOX_IRQSTATUS_RAW_u[0+m*2].NEWMSGSTATUSUUMBm	MAILBOX_IRQSTATUS_CLR_u[0+m*2].NEWMSGSTATUSUUMBm	MAILBOX_IRQENABLE_CLR_u[0+m*2].NEWMSGSTATUSUUMBm	MAILBOX_IRQENABLE_SET_u[0+m*2].NEWMSGSTATUSUUMBm	Mailbox m receives a new message
MAILBOX_IRQSTATUS_RAW_u[1+m*2].NOTFULLSTATUSUMBm	MAILBOX_IRQSTATUS_CLR_u[1+m*2].NOTFULLSTATUSUMBm	MAILBOX_IRQENABLE_CLR_u[1+m*2].NOTFULLSTATUSUMBm	MAILBOX_IRQENABLE_SET_u[1+m*2].NOTFULLSTATUSUMBm	Mailbox m message queue is not full

(1) MAILBOX.MAILBOX_IRQSTATUS_RAW_u register is mostly used for debug purposes.

CAUTION

Once an event generating the interrupt request has been processed by the software, it must be cleared by writing a logical 1 in the corresponding bit of the MAILBOX_IRQSTATUS_CLR_u register. Writing a logical 1 in a bit of the MAILBOX_IRQSTATUS_CLR_u register will also clear to 0 the corresponding bit in the appropriate MAILBOX_IRQSTATUS_RAW_u register.

An event can generate an interrupt request when a logical 1 is written to the corresponding unmask bit in the MAILBOX_IRQENABLE_SET_u register. Events are reported in the appropriate MAILBOX_IRQSTATUS_CLR_u and MAILBOX_IRQSTATUS_RAW_u registers.

An event stops generating interrupt requests when a logical 1 is written to the corresponding mask bit in the MAILBOX_IRQENABLE_CLR_u register. Events are only reported in the appropriate MAILBOX_IRQSTATUS_RAW_u register.

In case of the MAILBOX_IRQSTATUS_RAW_u register, the event is reported in the corresponding bit even if the interrupt request generation is disabled for this event.

1.13.3.5 Assignment

1.13.3.5.1 Description

To assign a receiver to a mailbox, set the new message interrupt enable bit corresponding to the desired mailbox in the MAILBOX_IRQENABLE_SET_u register. The receiver reads the MAILBOX_MESSAGE_m register to retrieve a message from the mailbox.

An alternate method for the receiver that does not use the interrupts is to poll the MAILBOX_FIFOSTATUS_m and/or MAILBOX_MSGSTATUS_m registers to know when to send or retrieve a message to or from the mailbox. This method does not require assigning a receiver to a mailbox. Because this method does not include the explicit assignment of the mailbox, the software must avoid having multiple receivers use the same mailbox, which can result in incoherency.

To assign a sender to a mailbox, set the queue-not-full interrupt enable bit of the desired mailbox in the MAILBOX_IRQENABLE_SET_u register, where u is the number of the sending user. However, direct allocation of a mailbox to a sender is not recommended because it can cause the sending processor to be constantly interrupted.

It is recommended that register polling be used to:

- Check the status of either the MAILBOX_FIFOSTATUS_m or MAILBOX_MSGSTATUS_m registers
- Write the message to the corresponding MAILBOX_MESSAGE_m register, if space is available

The sender might use the queue-not-full interrupt when the initial mailbox status check indicates the mailbox is full. In this case, the sender can enable the queue-not-full interrupt for its mailbox in the appropriate MAILBOX_IRQENABLE_SET_u register. This allows the sender to be notified by interrupt only when a FIFO queue has at least one available entry.

Reading the MAILBOX_IRQSTATUS_CLR_u register determines the status of the new message and the queue-not-full interrupts for a particular user. Writing 1 to the corresponding bit in the MAILBOX_IRQSTATUS_CLR_u register acknowledges, and subsequently clears, an interrupt.

CAUTION

Assigning multiple senders or multiple receivers to the same mailbox is not recommended.

1.13.3.6 Sending and Receiving Messages

1.13.3.6.1 Description

When a 32-bit message is written to the MAILBOX_MESSAGE_m register, the message is appended into the FIFO queue. This queue holds four messages. If the queue is full, the message is discarded. Queue overflow can be avoided by first reading the MAILBOX_FIFOSTATUS_m register to check that the mailbox message queue is not full before writing a new message to it. Reading the MAILBOX_MESSAGE_m register returns the message at the beginning of the FIFO queue and removes it from the queue. If the FIFO queue is empty when the MAILBOX_MESSAGE_m register is read, the value 0 is returned. The new message interrupt is asserted when at least one message is in the mailbox message FIFO queue. To determine the number of messages in the mailbox message FIFO queue, read the MAILBOX_MSGSTATUS_m register.

1.13.3.7 16-Bit Register Access

1.13.3.7.1 Description

So that 16-bit processors can access the mailbox module, the module allows 16-bit register read and write access, with restrictions for the MAILBOX_MESSAGE_m registers. The 16-bit half-words are organized in little endian fashion; that is, the least-significant 16 bits are at the low address and the most-significant 16 bits are at the high address (low address + 0x02). All mailbox module registers can be read or written to directly using individual 16-bit accesses with no restriction on interleaving, except the MAILBOX_MESSAGE_m registers, which must always be accessed by either single 32-bit accesses or two consecutive 16-bit accesses.

CAUTION

When using 16-bit accesses to the MAILBOX_MESSAGE_m registers, the order of access must be the least-significant half-word first (low address) and the most-significant half-word last (high address). This requirement is because of the update operation by the message FIFO of the MAILBOX_MSGSTATUS_m registers. The update of the FIFO queue contents and the associated status registers and possible interrupt generation occurs only when the most-significant 16 bits of a MAILBOX_MESSAGE_m are accessed.

1.13.4 Programming Guide

1.13.4.1 Low-level Programming Models

This section covers the low-level hardware programming sequences for configuration and usage of the mailbox module.

1.13.4.1.1 Global Initialization

1.13.4.1.1.1 Surrounding Modules Global Initialization

This section identifies the requirements of initializing the surrounding modules when the mailbox module is to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration of the mailbox.

See Section 1.13.2 for further information.

Table 1-103. Global Initialization of Surrounding Modules for System Mailbox

Surrounding Modules	Comments
PRCM	Mailbox functional/interface clock must be enabled.
Interrupt Controllers	Cortex-A8 MPU or DSP interrupt controller must be configured to enable the interrupt request generation to the Cortex-A8 MPU or DSP subsystem.

Table 1-104. Global Initialization of Surrounding Modules for HDVICP2-0 Mailbox

Surrounding Modules	Comments
PRCM	Mailbox functional/interface clock must be enabled.
Interrupt Controllers	Cortex-A8 MPU or DSP or iCont1/iCont2 of HDVICP2-0 Subsystem interrupt controller must be configured to enable the interrupt request generation to the Cortex-A8 MPU or DSP subsystem or HDVICP2-0 Subsystem

Table 1-105. Global Initialization of Surrounding Modules for HDVICP2-1 Mailbox

Surrounding Modules	Comments
PRCM	Mailbox functional/interface clock must be enabled.
Interrupt Controllers	Cortex-A8 MPU or DSP or iCont1/iCont2 of HDVICP2-1 Subsystem interrupt controller must be configured to enable the interrupt request generation to the Cortex-A8 MPU or DSP subsystem or HDVICP2-1 Subsystem

Table 1-106. Global Initialization of Surrounding Modules for HDVICP2-2 Mailbox

Surrounding Modules	Comments
PRCM	Mailbox functional/interface clock must be enabled.
Interrupt Controllers	Cortex-A8 MPU or DSP or iCont1/iCont2 of HDVICP2-2 Subsystem interrupt controller must be configured to enable the interrupt request generation to the Cortex-A8 MPU or DSP subsystem or HDVICP2-2 Subsystem

1.13.4.1.1.2 Mailbox Global Initialization

1.13.4.1.1.2.1 Main Sequence - Mailbox Global Initialization

This procedure initializes the mailbox module after a power-on or software reset.

Table 1-107. Mailbox Global Initialization

	Register/Bitfield/Programming Model	Value
Perform a software reset	MAILBOX_SYSCONFIG[0].SOFTRESET	1
Wait until reset is complete	MAILBOX_SYSCONFIG[0].SOFTRESET	0
Set idle mode configuration	MAILBOX_SYSCONFIG[3:2].SIDLEMODE	0x-

1.13.4.1.2 Operational Modes Configuration

1.13.4.1.2.1 Main Sequence - Sending a Message (Polling Method)

Table 1-108. Sending a Message (Polling Method)

Step	Register/Bitfield/Programming Model	Value
IF : Is FIFO full ?	MAILBOX_FIFOSTATUS_m[0].FIFOFULLMB	=1h
Wait until at least one message slot is available	MAILBOX_FIFOSTATUS_m[0].FIFOFULLMB	=0h
ELSE		
Write message	MAILBOX_MESSAGE_m[31:0].MESSAGEVALUE MBM	----h
ENDIF		

1.13.4.1.2.2 Main Sequence - Sending a Message (Interrupt Method)

Table 1-109. Sending a Message (Interrupt Method)

Step	Register/Bitfield/Programming Model	Value
IF : Is FIFO full ?	MAILBOX_FIFOSTATUS_m[0].FIFOFULLMB	=1h
Enable interrupt event	MAILBOX_IRQENABLE_SET_u[1+ m*2]	1h
User(processor) can perform another task until interrupt occurs		
ELSE		
Write message	MAILBOX_MESSAGE_m[31:0].MESSAGEVALUE MBM	----h
ENDIF		

1.13.4.1.2.3 Main Sequence - Receiving a Message (Polling Method)

Table 1-110. Receiving a Message (Polling Method)

Step	Register/Bitfield/Programming Model	Value
IF : Number of messages is not equal to 0	MAILBOX_MSGSTATUS_m[2:0].NBOFMSGMB	!=0h
Read message	MAILBOX_MESSAGE_m[31:0].MESSAGEVALUE MBM	----h
ENDIF		

1.13.4.1.2.4 Main Sequence - Receiving a Message (Interrupt Method)

Table 1-111. Receiving a Message (Interrupt Method)

Step	Register/Bitfield/Programming Model	Value
Enable interrupt event	MAILBOX_IRQENABLE_SET_u[0 + m*2]	1h
User(processor) can perform another task until interrupt occurs		

1.13.4.1.3 Events Servicing

1.13.4.1.3.1 Sending Mode

Table 1-112 describes the events servicing in sending mode.

Table 1-112. Events Servicing in Sending Mode

Step	Register/Bitfield/Programming Model	Value
Read interrupt status bit	MAILBOX_IRQSTATUS_CLR_u[1 + m*2]	1
Write message	MAILBOX_MESSAGE_m[31:0].MESSAGEVALUE MBM	----h
Write 1 to acknowledge interrupt	MAILBOX_IRQSTATUS_CLR_u[1 + m*2]	1

1.13.4.1.3.2 Receiving Mode

Table 1-113 describes the events servicing in receiving mode.

Table 1-113. Events Servicing in Receiving Mode

Step	Register/Bitfield/Programming Model	Value
Read interrupt status bit	MAILBOX_IRQSTATUS_CLR_u[0 + m*2]	1
IF : Number of messages is not equal to 0 ?	MAILBOX_MSGSTATUS_m[2:0].NBOFMSGMB	!=0h
Read message	MAILBOX_MESSAGE_m[31:0].MESSAGEVALUE MBM	----h
ELSE		
Write 1 to acknowledge interrupt	MAILBOX_IRQSTATUS_CLR_u[0 + m*2]	1
ENDIF		

1.13.5 Mailbox Registers

Table 1-114 lists the mailboxes. The previous register set is applicable to these mailboxes. See the device-specific data manual for the memory address of these mailboxes.

Table 1-114. Mailboxes

Mailbox Type	User Number(u)	Mailbox Number(m)	Messages per Mailbox
System Mailbox	0 to 3	0 to 11	4
HDVICP2-0 Mailbox	0 to 3	0 to 5	4
HDVICP2-1Mailbox	0 to 3	0 to 5	4
HDVICP2-2 Mailbox	0 to 3	0 to 5	4

Table 1-115 lists the mailbox registers. For the base address of these registers, see Table 1-12.

Table 1-115. Mailbox Registers

Offset	Acronym	Register Description	Section
0h	MAILBOX_REVISION	Mailbox Revision Register	Section 1.13.5.1
10h	MAILBOX_SYSCONFIG	Mailbox System Configuration Register	Section 1.13.5.2
40h + (4h × m)	MAILBOX_MESSAGE_m	Mailbox Message Register	Section 1.13.5.3
80h + (4h × m)	MAILBOX_FIFOSTATUS_m	Mailbox FIFO Status Register	Section 1.13.5.4
C0h + (4h × m)	MAILBOX_MSGSTATUS_m	Mailbox Message Status Register	Section 1.13.5.5
100h + (10h × u)	MAILBOX_IRQSTATUS_RAW_u	Mailbox IRQ RAW Status Register	Section 1.13.5.6
104h + (10h × u)	MAILBOX_IRQSTATUS_CLR_u	Mailbox IRQ Clear Status Register	Section 1.13.5.7
108h + (10h × u)	MAILBOX_IRQENABLE_SET_u	Mailbox IRQ Enable Set Register	Section 1.13.5.8
10Ch + (10h × u)	MAILBOX_IRQENABLE_CLR_u	Mailbox IRQ Enable Clear Register	Section 1.13.5.9

1.13.5.1 Revision Register (MAILBOX_REVISION)

This register contains the IP revision code. The Mailbox Revision Register (MAILBOX_REVISION) is shown in Figure 1-83 described in Table 1-116.

Figure 1-83. Revision Register (MAILBOX_REVISION)

31	4	3 2	1	0
REVISION		SIDLEMODE	Reserved	SOFTRESET
R-0		R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-116. Revision Register (MAILBOX_REVISION) Field Descriptions

Bit	Field	Value	Description
31-4	REVISION	0	Reserved
3-2	SIDLEMODE	0	Force-idle. An idle request is acknowledged unconditionally
		1	No-idle. An idle request is never acknowledged
		2	Smart-idle. Acknowledgment to an idle request is given based on the internal activity of the module based on the internal activity of the module
		3	Reserved
1	Reserved	0	Reserved
0	SOFTRESET	0	Soft/Hard reset is done.
		1	Reset is ongoing.
		0	No action.
		1	Start the soft reset sequence.

1.13.5.2 System Configuration Register (MAILBOX_SYSCONFIG)

This register controls the various parameters of the communication interface. The Mailbox System Configuration Register (MAILBOX_SYSCONFIG) is shown in Figure 1-84 and described in Table 1-117.

Figure 1-84. System Configuration Register (MAILBOX_SYSCONFIG)

31	0
REVISION	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 1-117. System Configuration Register (MAILBOX_SYSCONFIG) Field Descriptions

Bit	Field	Value	Description
31-0	REVISION	0-FFFF FFFFh	IP Revision

1.13.5.3 Message Register (MAILBOX_MESSAGE_m)

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue. The Mailbox Message Register (MAILBOX_MESSAGE_m) is shown in Figure 1-85 and described in Table 1-118.

Figure 1-85. Message Register (MAILBOX_MESSAGE_m)

31	0
MESSAGEVALUEMBM	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 1-118. Message Register (MAILBOX_MESSAGE_m) Field Descriptions

Bit	Field	Value	Description
31-0	MESSAGEVALUEMBM	0-FFFF FFFFh	Message in Mailbox. The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

1.13.5.4 FIFO Status Register (MAILBOX_FIFOSTATUS_m)

The FIFO status register has the status related to the mailbox internal FIFO. The Mailbox FIFO Status Register (MAILBOX_FIFOSTATUS_m) is shown in Figure 1-86 and described in Table 1-119.

Figure 1-86. FIFO Status Register (MAILBOX_FIFOSTATUS_m)

31	1	0
MESSAGEVALUEMBM		FIFOFULLMBM
R/W-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

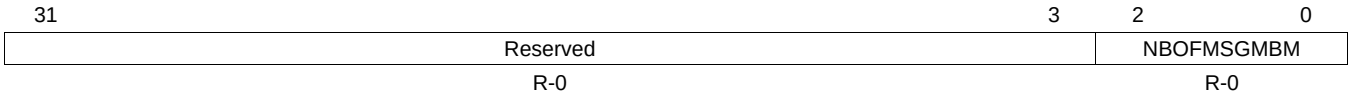
Table 1-119. FIFO Status Register (MAILBOX_FIFOSTATUS_m) Field Descriptions

Bit	Field	Value	Description
31-1	MESSAGEVALUEMBM	0-7FFFF FFFFh	Message in Mailbox. The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.
0	FIFOFULLMBM		Full flag for Mailbox
		0	Mailbox FIFO is not full
		1	Mailbox FIFO is full

1.13.5.5 Message Status Register (MAILBOX_MSGSTATUS_m)

The message status register has the status of the messages in the mailbox. The Mailbox Message Status Register (MAILBOX_MSGSTATUS_m) is shown in Figure 1-87 and described in Table 1-120.

Figure 1-87. Message Status Register (MAILBOX_MSGSTATUS_m)



LEGEND: R = Read only; -*n* = value after reset

Table 1-120. Message Status Register (MAILBOX_MSGSTATUS_m) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved.
2-0	NBOFMSGMBM	0-7h	Number of unread messages in Mailbox. Limited to four messages per mailbox.

1.13.5.6 IRQ RAW Status Register (MAILBOX_IRQSTATUS_RAW_u)

The interrupt status register has the raw status for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit sets this bit. This register is mainly used for debug purpose. The Mailbox IRQ RAW Status Register (MAILBOX_IRQSTATUS_RAW_u) is shown in Figure 1-88 and described in Table 1-121.

Figure 1-88. IRQ RAW Status Register (MAILBOX_IRQSTATUS_RAW_u)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
NOTFULLSTATUSUUMB11	NEWMSGSTATUSUUMB11	NOTFULLSTATUSUUMB10	NEWMSGSTATUSUUMB10	NOTFULLSTATUSUUMB9	NEWMSGSTATUSUUMB9	NOTFULLSTATUSUUMB8	NEWMSGSTATUSUUMB8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8
NOTFULLSTATUSUUMB7	NEWMSGSTATUSUUMB7	NOTFULLSTATUSUUMB6	NEWMSGSTATUSUUMB6	NOTFULLSTATUSUUMB5	NEWMSGSTATUSUUMB5	NOTFULLSTATUSUUMB4	NEWMSGSTATUSUUMB4
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
NOTFULLSTATUSUUMB3	NEWMSGSTATUSUUMB3	NOTFULLSTATUSUUMB2	NEWMSGSTATUSUUMB2	NOTFULLSTATUSUUMB1	NEWMSGSTATUSUUMB1	NOTFULLSTATUSUUMB0	NEWMSGSTATUSUUMB0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-121. IRQ RAW Status Register (MAILBOX_IRQSTATUS_RAW_u) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23	NOTFULLSTATUSUUMB11		Not Full Status bit for User u, Mailbox 11
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
22	NEWMSGSTATUSUUMB11		New Message Status bit for User u, Mailbox 11
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
21	NOTFULLSTATUSUUMB10		Not Full Status bit for User u, Mailbox 10
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
20	NEWMSGSTATUSUUMB10		New Message Status bit for User u, Mailbox 10
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)

Table 1-121. IRQ RAW Status Register (MAILBOX_IRQSTATUS_RAW_u) Field Descriptions (continued)

Bit	Field	Value	Description
19	NOTFULLSTATUSUUMB9		Not Full Status bit for User u, Mailbox 9
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
18	NEWMSGSTATUSUUMB9		New Message Status bit for User u, Mailbox 9
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
17	NOTFULLSTATUSUUMB8		Not Full Status bit for User u, Mailbox 8
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
16	NEWMSGSTATUSUUMB8		New Message Status bit for User u, Mailbox 8
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
15	NOTFULLSTATUSUUMB7		Not Full Status bit for User u, Mailbox 7
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
14	NEWMSGSTATUSUUMB7		New Message Status bit for User u, Mailbox 7
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
13	NOTFULLSTATUSUUMB6		Not Full Status bit for User u, Mailbox 6
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
12	NEWMSGSTATUSUUMB6		New Message Status bit for User u, Mailbox 6
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
11	NOTFULLSTATUSUUMB5		Not Full Status bit for User u, Mailbox 5
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)

Table 1-121. IRQ RAW Status Register (MAILBOX_IRQSTATUS_RAW_u) Field Descriptions (continued)

Bit	Field	Value	Description
10	NEWMSGSTATUSUUMB5		New Message Status bit for User u, Mailbox 5
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
9	NOTFULLSTATUSUUMB4		Not Full Status bit for User u, Mailbox 4
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
8	NEWMSGSTATUSUUMB4		New Message Status bit for User u, Mailbox 4
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
7	NOTFULLSTATUSUUMB3		Not Full Status bit for User u, Mailbox 3
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
6	NEWMSGSTATUSUUMB3		New Message Status bit for User u, Mailbox 3
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
5	NOTFULLSTATUSUUMB2		Not Full Status bit for User u, Mailbox 2
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
4	NEWMSGSTATUSUUMB2		New Message Status bit for User u, Mailbox 2
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
3	NOTFULLSTATUSUUMB1		Not Full Status bit for User u, Mailbox 1
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
2	NEWMSGSTATUSUUMB1		New Message Status bit for User u, Mailbox 1
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)

Table 1-121. IRQ RAW Status Register (MAILBOX_IRQSTATUS_RAW_u) Field Descriptions (continued)

Bit	Field	Value	Description
1	NOTFULLSTATUSUUMB0		Not Full Status bit for User u, Mailbox 0
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
0	NEWMSGSTATUSUUMB0		New Message Status bit for User u, Mailbox 0
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)

1.13.5.7 IRQ Clear Status Register (MAILBOX_IRQSTATUS_CLR_u)

The interrupt status register has the status combined with irq-enable for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit. The Mailbox IRQ Clear Status Register (MAILBOX_IRQSTATUS_CLR_u) is shown in Figure 1-89 and described in Table 1-122.

Figure 1-89. IRQ Clear Status Register (MAILBOX_IRQSTATUS_CLR_u)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
NOTFULLSTATUSUUMB11	NEWMSGSTATUSUUMB11	NOTFULLSTATUSUUMB10	NEWMSGSTATUSUUMB10	NOTFULLSTATUSUUMB9	NEWMSGSTATUSUUMB9	NOTFULLSTATUSUUMB8	NEWMSGSTATUSUUMB8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8
NOTFULLSTATUSUUMB7	NEWMSGSTATUSUUMB7	NOTFULLSTATUSUUMB6	NEWMSGSTATUSUUMB6	NOTFULLSTATUSUUMB5	NEWMSGSTATUSUUMB5	NOTFULLSTATUSUUMB4	NEWMSGSTATUSUUMB4
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
NOTFULLSTATUSUUMB3	NEWMSGSTATUSUUMB3	NOTFULLSTATUSUUMB2	NEWMSGSTATUSUUMB2	NOTFULLSTATUSUUMB1	NEWMSGSTATUSUUMB1	NOTFULLSTATUSUUMB0	NEWMSGSTATUSUUMB0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-122. IRQ Clear Status Register (MAILBOX_IRQSTATUS_CLR_u) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved.
23	NOTFULLSTATUSUUMB11		Not Full Status bit for User u, Mailbox 11
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
22	NEWMSGSTATUSUUMB11		New Message Status bit for User u, Mailbox 11
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
21	NOTFULLSTATUSUUMB10		Not Full Status bit for User u, Mailbox 10
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
20	NEWMSGSTATUSUUMB10		New Message Status bit for User u, Mailbox 10
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)

Table 1-122. IRQ Clear Status Register (MAILBOX_IRQSTATUS_CLR_u) Field Descriptions (continued)

Bit	Field	Value	Description
19	NOTFULLSTATUSUUMB9		Not Full Status bit for User u, Mailbox 9
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
18	NEWMSGSTATUSUUMB9		New Message Status bit for User u, Mailbox 9
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
17	NOTFULLSTATUSUUMB8		Not Full Status bit for User u, Mailbox 8
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
16	NEWMSGSTATUSUUMB8		New Message Status bit for User u, Mailbox 8
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
15	NOTFULLSTATUSUUMB7		Not Full Status bit for User u, Mailbox 7
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
14	NEWMSGSTATUSUUMB7		New Message Status bit for User u, Mailbox 7
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
13	NOTFULLSTATUSUUMB6		Not Full Status bit for User u, Mailbox 6
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
12	NEWMSGSTATUSUUMB6		New Message Status bit for User u, Mailbox 6
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
11	NOTFULLSTATUSUUMB5		Not Full Status bit for User u, Mailbox 5
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)

Table 1-122. IRQ Clear Status Register (MAILBOX_IRQSTATUS_CLR_u) Field Descriptions (continued)

Bit	Field	Value	Description
10	NEWMSGSTATUSUUMB5		New Message Status bit for User u, Mailbox 5
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
9	NOTFULLSTATUSUUMB4		Not Full Status bit for User u, Mailbox 4
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
8	NEWMSGSTATUSUUMB4		New Message Status bit for User u, Mailbox 4
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
7	NOTFULLSTATUSUUMB3		Not Full Status bit for User u, Mailbox 3
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
6	NEWMSGSTATUSUUMB3		New Message Status bit for User u, Mailbox 3
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
5	NOTFULLSTATUSUUMB2		Not Full Status bit for User u, Mailbox 2
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
4	NEWMSGSTATUSUUMB2		New Message Status bit for User u, Mailbox 2
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
3	NOTFULLSTATUSUUMB1		Not Full Status bit for User u, Mailbox 1
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
2	NEWMSGSTATUSUUMB1		New Message Status bit for User u, Mailbox 1
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)

Table 1-122. IRQ Clear Status Register (MAILBOX_IRQSTATUS_CLR_u) Field Descriptions (continued)

Bit	Field	Value	Description
1	NOTFULLSTATUSUUMB0		Not Full Status bit for User u, Mailbox 0
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
0	NEWMSGSTATUSUUMB0		New Message Status bit for User u, Mailbox 0
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)

1.13.5.8 IRQ Enable Set Register (MAILBOX_IRQENABLE_SET_u)

The interrupt enable register enables to unmask the module internal source of interrupt to the corresponding user. This register is write 1 to set. The Mailbox IRQ Enable Set Register (MAILBOX_IRQENABLE_SET_u) is shown in Figure 1-90 and described in Table 1-123.

Figure 1-90. IRQ Enable Set Register (MAILBOX_IRQENABLE_SET_u)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
NOTFULLSTATUSUUMB11	NEWMSGSTATUSUUMB11	NOTFULLSTATUSUUMB10	NEWFULLSTATUSUUMB10	NOTFULLSTATUSUUMB9	NEWMSGSTATUSUUMB9	NOTFULLSTATUSUUMB8	NEWMSGSTATUSUUMB8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8
NOTFULLSTATUSUUMB7	NEWMSGSTATUSUUMB7	NOTFULLSTATUSUUMB6	NEWMSGSTATUSUUMB6	NOTFULLSTATUSUUMB5	NEWMSGSTATUSUUMB5	NOTFULLSTATUSUUMB4	NEWMSGSTATUSUUMB4
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
NOTFULLSTATUSUUMB3	NEWMSGSTATUSUUMB3	NOTFULLSTATUSUUMB2	NEWMSGSTATUSUUMB2	NOTFULLSTATUSUUMB1	NEWMSGSTATUSUUMB1	NOTFULLSTATUSUUMB0	NEWMSGSTATUSUUMB0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-123. IRQ Enable Set Register (MAILBOX_IRQENABLE_SET_u) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23	NOTFULLSTATUSUUMB11		Not Full Status bit for User u, Mailbox 11
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
22	NEWMSGSTATUSUUMB11		New Message Status bit for User u, Mailbox 11
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
21	NOTFULLSTATUSUUMB10		Not Full Status bit for User u, Mailbox 10
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
20	NEWFULLSTATUSUUMB10		New Message Status bit for User u, Mailbox 10
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
19	NOTFULLSTATUSUUMB9		Not Full Status bit for User u, Mailbox 9
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)

Table 1-123. IRQ Enable Set Register (MAILBOX_IRQENABLE_SET_u) Field Descriptions (continued)

Bit	Field	Value	Description
18	NEWMSGSTATUSUUMB9		New Message Status bit for User u, Mailbox 9
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
17	NOTFULLSTATUSUUMB8		Not Full Status bit for User u, Mailbox 8
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
16	NEWMSGSTATUSUUMB8		New Message Status bit for User u, Mailbox 8
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
15	NOTFULLSTATUSUUMB7		Not Full Status bit for User u, Mailbox 7
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
14	NEWMSGSTATUSUUMB7		New Message Status bit for User u, Mailbox 7
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
13	NOTFULLSTATUSUUMB6		Not Full Status bit for User u, Mailbox 6
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
12	NEWMSGSTATUSUUMB6		New Message Status bit for User u, Mailbox 6
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
11	NOTFULLSTATUSUUMB5		Not Full Status bit for User u, Mailbox 5
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
10	NEWMSGSTATUSUUMB5		New Message Status bit for User u, Mailbox 5
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)

Table 1-123. IRQ Enable Set Register (MAILBOX_IRQENABLE_SET_u) Field Descriptions (continued)

Bit	Field	Value	Description
9	NOTFULLSTATUSUUMB4		Not Full Status bit for User u, Mailbox 4
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
8	NEWMSGSTATUSUUMB4		New Message Status bit for User u, Mailbox 4
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
7	NOTFULLSTATUSUUMB3		Not Full Status bit for User u, Mailbox 3
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
6	NEWMSGSTATUSUUMB3		New Message Status bit for User u, Mailbox 3
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
5	NOTFULLSTATUSUUMB2		Not Full Status bit for User u, Mailbox 2
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
4	NEWMSGSTATUSUUMB2		New Message Status bit for User u, Mailbox 2
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
3	NOTFULLSTATUSUUMB1		Not Full Status bit for User u, Mailbox 1
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
2	NEWMSGSTATUSUUMB1		New Message Status bit for User u, Mailbox 1
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
1	NOTFULLSTATUSUUMB0		Not Full Status bit for User u, Mailbox 0
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)

Table 1-123. IRQ Enable Set Register (MAILBOX_IRQENABLE_SET_u) Field Descriptions (continued)

Bit	Field	Value	Description
0	NEWMSGSTATUSUUMB0		New Message Status bit for User u, Mailbox 0
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)

1.13.5.9 IRQ Enable Clear Register (MAILBOX_IRQENABLE_CLR_u)

The interrupt enable register enables to mask the module internal source of interrupt to the corresponding user. This register is write 1 to clear. The IRQ Enable Clear Register (MAILBOX_IRQENABLE_CLR_u) is shown in Figure 1-91 and described in Table 1-124.

Figure 1-91. IRQ Enable Clear Register (MAILBOX_IRQENABLE_CLR_u)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
NOTFULLSTATUSUUMB11	NEWMSGSTATUSUUMB11	NOTFULLSTATUSUUMB10	NEWMSGSTATUSUUMB10	NOTFULLSTATUSUUMB9	NEWMSGSTATUSUUMB9	NOTFULLSTATUSUUMB8	NEWMSGSTATUSUUMB8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8
NOTFULLSTATUSUUMB7	NEWMSGSTATUSUUMB7	NOTFULLSTATUSUUMB6	NEWMSGSTATUSUUMB6	NOTFULLSTATUSUUMB5	NEWMSGSTATUSUUMB5	NOTFULLSTATUSUUMB4	NEWMSGSTATUSUUMB4
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
NOTFULLSTATUSUUMB3	NEWMSGSTATUSUUMB3	NOTFULLSTATUSUUMB2	NEWMSGSTATUSUUMB2	NOTFULLSTATUSUUMB1	NEWMSGSTATUSUUMB1	NOTFULLSTATUSUUMB0	NEWMSGSTATUSUUMB0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-124. IRQ Enable Clear Register (MAILBOX_IRQENABLE_CLR_u) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23	NOTFULLSTATUSUUMB11		Not Full Status bit for User u, Mailbox 11
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
22	NEWMSGSTATUSUUMB11		New Message Status bit for User u, Mailbox 11
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
21	NOTFULLSTATUSUUMB10		Not Full Status bit for User u, Mailbox 10
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
20	NEWMSGSTATUSUUMB10		New Message Status bit for User u, Mailbox 10
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
19	NOTFULLSTATUSUUMB9		Not Full Status bit for User u, Mailbox 9
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)

Table 1-124. IRQ Enable Clear Register (MAILBOX_IRQENABLE_CLR_u) Field Descriptions (continued)

Bit	Field	Value	Description
18	NEWMSGSTATUSUUMB9		New Message Status bit for User u, Mailbox 9
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
17	NOTFULLSTATUSUUMB8		Not Full Status bit for User u, Mailbox 8
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
16	NEWMSGSTATUSUUMB8		New Message Status bit for User u, Mailbox 8
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
15	NOTFULLSTATUSUUMB7		Not Full Status bit for User u, Mailbox 7
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
14	NEWMSGSTATUSUUMB7		New Message Status bit for User u, Mailbox 7
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
13	NOTFULLSTATUSUUMB6		Not Full Status bit for User u, Mailbox 6
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
12	NEWMSGSTATUSUUMB6		New Message Status bit for User u, Mailbox 6
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
11	NOTFULLSTATUSUUMB5		Not Full Status bit for User u, Mailbox 5
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
10	NEWMSGSTATUSUUMB5		New Message Status bit for User u, Mailbox 5
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)

Table 1-124. IRQ Enable Clear Register (MAILBOX_IRQENABLE_CLR_u) Field Descriptions (continued)

Bit	Field	Value	Description
9	NOTFULLSTATUSUUMB4		Not Full Status bit for User u, Mailbox 4
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
8	NEWMSGSTATUSUUMB4		New Message Status bit for User u, Mailbox 4
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
7	NOTFULLSTATUSUUMB3		Not Full Status bit for User u, Mailbox 3
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
6	NEWMSGSTATUSUUMB3		New Message Status bit for User u, Mailbox 3
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
5	NOTFULLSTATUSUUMB2		Not Full Status bit for User u, Mailbox 2
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
4	NEWMSGSTATUSUUMB2		New Message Status bit for User u, Mailbox 2
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
3	NOTFULLSTATUSUUMB1		Not Full Status bit for User u, Mailbox 1
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)
2	NEWMSGSTATUSUUMB1		New Message Status bit for User u, Mailbox 1
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)
1	NOTFULLSTATUSUUMB0		Not Full Status bit for User u, Mailbox 0
		0	Read: No event pending (message queue full)
		1	Read: Event pending (message queue not full)
		0	Write: No action
		1	Write: Set the event (for debug)

Table 1-124. IRQ Enable Clear Register (MAILBOX_IRQENABLE_CLR_u) Field Descriptions (continued)

Bit	Field	Value	Description
0	NEWMSGSTATUSUUMB0		New Message Status bit for User u, Mailbox 0
		0	Read: No event (message) pending
		1	Read: Event (message) pending
		0	Write: No action
		1	Write: Set the event (for debug)

1.14 Spinlock

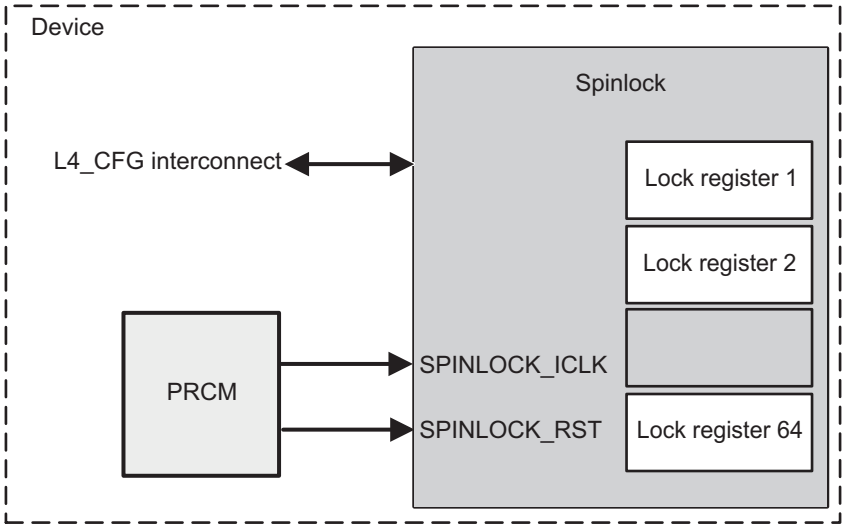
1.14.1 Overview

The Spinlock module provides hardware assistance for synchronizing the processes running on multiple processors in the device:

- Cortex-A8 microprocessor unit (MPU) subsystem
- Digital signal processor (DSP) subsystem
- Media Controller subsystem

The Spinlock module implements 64 spinlocks (or hardware semaphores), which provide an efficient way to perform a lock operation of a device resource using a single read-access, avoiding the need of a read-modify-write bus transfer that the programmable cores are not capable of.

Figure 1-92. Spinlock Module



1.14.2 Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

Figure 1-93 shows the Spinlock integration.

Figure 1-93. Spinlock Integration

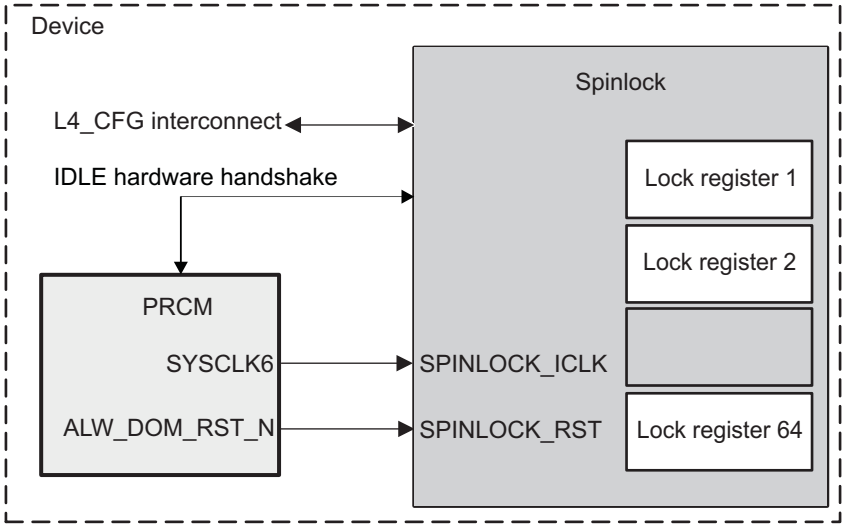


Table 1-125 and Table 1-126 summarize the integration of the module in the device.

Table 1-125. Integration Attributes

Module Instance	Attributes	
	Power Domain	Interconnect
SPINLOCK	PD_ALWAYS_ON	L4_STANDARD

Table 1-126. Clocks and Resets

Clocks				
Module Instance	**Destination Signal Name**	**Source Signal Name**	**Source**	**Description**
SPINLOCK	SPINLOCK_ICLK	SYSCLK6	PRCM	Spinlock interface clock. This clock is used for all interface and functional operations.
Resets				
SPINLOCK	SPINLOCK_RST	ALW_DOM_RST_N	PRCM	Spinlock hardware reset. This reset is asynchronously applied to the Spinlock registers.

The Spinlock module does not support any interrupt and DMA requests.

1.14.3 Functional Description

1.14.3.1 Software Reset

The Spinlock module can be reset by software through the SPINLOCK_SYSCONFIG[1] SOFTRESET bit. Setting this bit to 1 enables an active software reset that is functionally equivalent to a hardware reset. This bit also indicates that the software reset is complete when its value is reset to 0. The software must ensure that the software reset completes before doing Spinlock operations.

1.14.3.2 Power Management

Table 1-127 describes power-management features available to the Spinlock module.

Table 1-127. Local Power Management Features

Feature	Registers	Description
Clock auto gating	[0] AUTOGATING bit	This bit indicates that the module uses an automatic internal interface clock gating strategy, based on interface activity.
Slave idle modes	[4:3] SIDLEMODE bit field	This bit field indicates that the module uses smart-idle mode.
Clock activity	[8] CLOCKACTIVITY bit	This bit indicates that the interface clock is not required by the module during idle mode and may be switched off.
Global wake-up enable	[2] ENAWAKEUP bit	This bit indicates that the wake-up generation feature (at module level) is disabled.

NOTE: All the local power management features are non-configurable (that is, their respective bits or bit fields are read-only).

CAUTION

The PRCM module has no hardware means of reading CLOCKACTIVITY settings. Thus, software must ensure consistent programming between the CLOCKACTIVITY bit and Spinlock clock PRCM control bits.

The Spinlock module is normally idle, except when processing a request from its slave interface port. The smart-idle mode acknowledges idle requests from the PRCM module only when the module is prepared to go idle. The Spinlock module is always ready to go idle if it does not have any request that it is processing.

The Spinlock module uses retention flops to retain state including the Taken state of each lock register. This means that the module can be placed in retention at any time when it is not processing a request and it is known that the system will not need to access the module.

Software must ensure to only power off the Spinlock module when no locks would be lost. In general, the steps to powering down the Spinlock module are:

- Check that all masters which might use the Spinlock module are either:
  - Already powered off
  - Notified that Spinlock is not available and the notification is acknowledged
- If desired, check that no locks are currently held in the Spinlock module. The status of each bank of 64 locks can be read from the register. If any locks are held, they are orphaned because they are not held by any master that is still active. Alternatively, you may decide to wait a timeout period to allow any active master to clean up its locks before powering down.
- The Spinlock module can now be powered off by writing the appropriate status to the PRCM module.

In the case of powering off the whole system, these steps are unnecessary.

1.14.3.3 About Spinlocks

Spinlocks are present to solve the need for synchronization and mutual exclusion between heterogeneous processors and those not operating under a single, shared operating system. There is no alternative mechanism to accomplish these operations between processors in separate subsystems.

Spinlocks are not the best way to synchronize between tasks or threads on one CPU. Instead, spinlocks are for use in synchronization between different subsystems in the device that don't have any other means of hardware-based synchronization.

Spinlocks do not solve all system synchronization issues. They have limited applicability and should be used with care to implement higher level synchronization protocols.

A spinlock is appropriate for mutual exclusion for access to a shared data structure. It should be used only when:

- The time to hold the lock is predictable and small (for example, a maximum hold time of less than 200 CPU cycles may be acceptable).
- The locking task cannot be preempted, suspended, or interrupted while holding the lock (this would make the hold time large and unpredictable).
- The lock is lightly contended, that is the chance of any other process (or processor) trying to acquire the lock while it is held is small.

If these conditions are met, then the locking code can retry a failed attempt to acquire the lock until success.

If the conditions are not met, then a spinlock is not a good candidate. One alternative is to use a spinlock for critical section control (engineered to meet the conditions) to implement a higher level semaphore that can support preemption, notification, timeout or other higher level properties.

1.14.3.4 Functional Operation

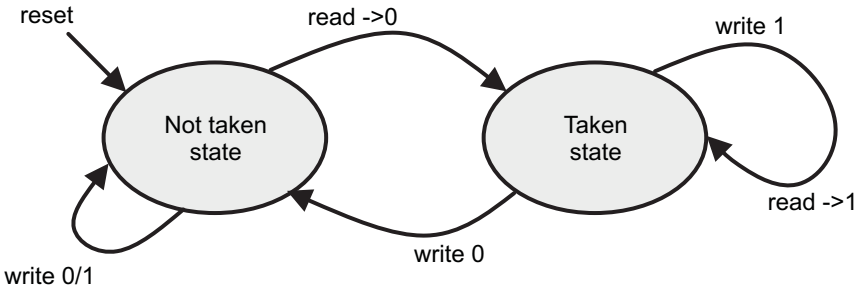
The Spinlock module supports 64 spinlocks. It accepts only a single command at a time and processes the command fully before accepting the next command. A lock is requested by reading the SPINLOCK_LOCK_REG_i[0] TAKEN bit. There are two states: Taken (SPINLOCK_LOCK_REG_i[0] TAKEN = 1) or Not Taken (SPINLOCK_LOCK_REG_i[0] TAKEN = 0).

When the status of lock i (where i = 0 to 63) is Not Taken (free), a read from the SPINLOCK_LOCK_REG_i register returns 0 and sets the lock to Taken (locked). When the status of lock i is Taken, a read returns 1 and does not change the state of the lock. A write to the SPINLOCK_LOCK_REG_i register does not change the state of lock, unless when writing 0 when the lock is in Taken state. By doing this, the requester frees the lock.

CAUTION

Only 32-bit reads and writes are supported.

Figure 1-94. SPINLOCK_LOCK_REG_i Register State Diagram



1.14.4 Programming Guide

1.14.4.1 Low-level Programming Models

This section covers the low-level hardware programming sequences for configuration and usage of the module.

1.14.4.1.1 Surrounding Modules Global Initialization

This procedure initializes the surrounding modules when the Spinlock module is used for the first time after a device reset.

1.14.4.1.2 Global Initialization of Surrounding Modules

Table 1-128. Global Initialization of Surrounding Modules

Surrounding Modules	Comments
PRCM	Spinlock interface clock must be enabled. For more information, Refer to the *Power, Reset, and Clock Management (PRCM) Module* chapter.
Interconnect	For more information about the L4-Standard interconnect configuration, Refer to the *Bus Interconnect* section.

1.14.4.2 Basic Spinlock Operations

The main spinlock operations are:

- Clear all the Taken spinlocks (only after a system bug recovery)
- Take a spinlock
- Release spinlock

1.14.4.2.1 Spinlocks Clearing After a System Bug Recovery

Module initialization (after reset) is not needed, except after system bug recovery. The following table presents the Spinlock initialization after a system bug recovery. Software should store 0 into each of the SPINLOCK_LOCK_REG_i registers at system startup to insure that all locks are initialized to Not Taken.

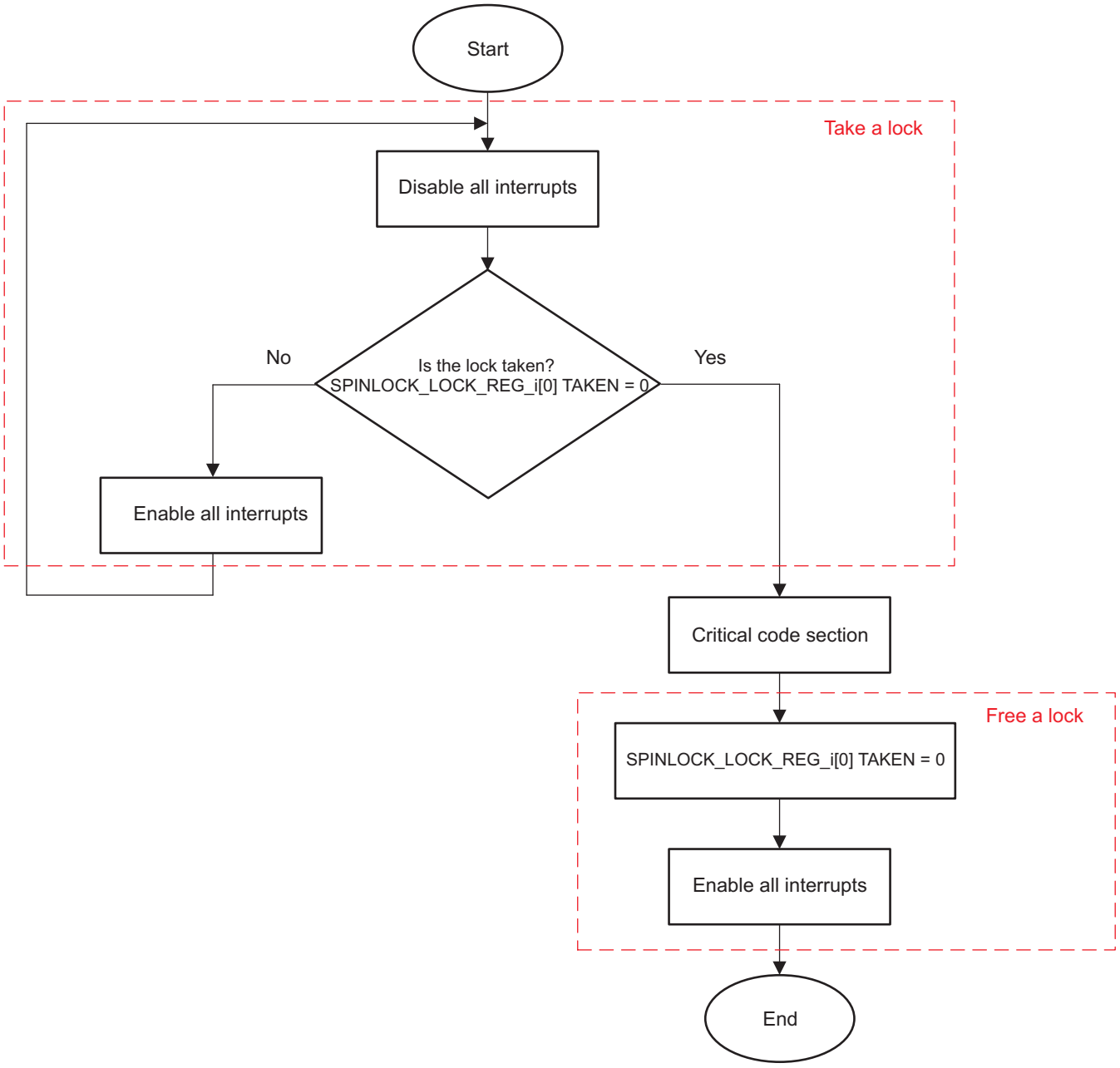
Table 1-129. Spinlock System Bug Recovery

Step	Register	Value
IF: SPINLOCK_SYSTATUS[0] IU0 = = 1?	SPINLOCK_SYSTATUS[0] IU0	
Free the 64 locks	SPINLOCK_LOCK_REG_i[0] TAKEN (i=0 to 63)	0
END		

1.14.4.2.2 Take and Release Spinlock

This procedure configures the take and release (free) operations for the Spinlock module. A spinlock should only be held with interrupts disabled. So, before attempting to obtain the spinlock, software should disable interrupts. Then it should read the SPINLOCK_LOCK_REG_i[0] TAKEN bit to attempt to obtain the lock. If it succeeds, it should proceed directly through the critical section then unlock and re-enable interrupts. If the acquisition attempt fails, the acquisition should be re-attempted. To prevent unknown interrupt disabled time, interrupts should be re-enabled and then disabled before re-attempting to acquire the lock. Figure 1-95 shows the described above procedure.

Figure 1-95. Take and Release Spinlock



1.14.5 Spinlock Registers

Table 1-130 lists the spinlock module registers. For the base address of these registers, see Table 1-12.

Table 1-130. Spinlock Module Registers

Address Offset	Acronym	Description	Section
00h	SPINLOCK_REV	Revision Register	Section 1.14.5.1
010h	SPINLOCK_SYSCFG	System Configuration Register	Section 1.14.5.2
014h	SPINLOCK_SYSSTAT	System Status Register	Section 1.14.5.3
800h + (4h × i)	SPINLOCK_LOCK_REG_i	Lock Register	Section 1.14.5.4

1.14.5.1 Revision Register (SPINLOCK_REV)

The Spin Lock Revision Register (SPINLOCK_REV) is shown in Figure 1-96 and described in Table 1-131.

Figure 1-96. Revision Register (SPINLOCK_REV)

31	0
REV	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 1-131. Revision Register (SPINLOCK_REV) Field Descriptions

Bit	Field	Value	Description
31-0	REV	0	IP Revision Code.

1.14.5.2 System Configuration Register (SPINLOCK_SYS_CFG)

The Spin Lock System Configuration Register (SPINLOCK_SYSCFG) is shown in Figure 1-97 described in Table 1-132.

Figure 1-97. System Configuration Register (SPINLOCK_SYS_CFG)

31	16
Reserved	
R-0	

15	9	8	7	5	4	3	2	1	0
Reserved		CLOCKACTIVITY	Reserved		SIDLEMODE		ENWAKEUP	SOFTRESET	AUTOGATING
R-0		R-0	R-0		R-0		R-0	W-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 1-132. System Configuration Register (SPINLOCK_SYS_CFG) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLOCKACTIVITY		Indicates whether the module requires the interface clock when in IDLE mode.
		0	Interface clock is not required by the module during IDLE mode and may be switched off.
		1	Interface clock is required by the module, even during idle mode.
7-5	Reserved	0	Reserved
4-3	SIDLEMODE		Slave interface power management (IDLE request/acknowledgment control)
		0	Force-idle. IDLE request is acknowledged unconditionally and immediately.
		1h	No-idle. IDLE request is never acknowledged.
		2h	Smart-idle. IDLE request acknowledgment is based on the internal module activity.
		3h	Reserved
2	ENWAKEUP		Asynchronous wakeup generation
		0	Wakeup generation is disabled.
		1	Wakeup generation is enabled.
1	SOFTRESET		Module software reset
		0	No action
		1	Start soft reset sequence
0	AUTOGATING		Internal interface clock gating strategy.
		0	Interface clock is not gated when the L4-STANDARD interface is idle.
		1	Automatic internal interface clock gating strategy is applied, based on the L4-CFG interface activity.

1.14.5.3 System Status Register (SPINLOCK_SYSSTAT)

The Spin Lock System Status Register (SPINLOCK_SYSSTAT) is shown in Figure 1-98 and described in Table 1-133.

Figure 1-98. System Status Register (SPINLOCK_SYSSTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMLOCKS								Reserved							
R-0								R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IU7	IU6	IU5	IU4	IU3	IU2	IU1	IU0	Reserved							RESETDONE
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0							R-0

LEGEND: R = Read only; -*n* = value after reset

Table 1-133. System Status Register (SPINLOCK_SYSSTAT) Field Descriptions

Bit	Field	Value	Description
31-24	NUMLOCKS		Number of lock registers implemented.
		1h	This instance has 32 lock registers.
		2h	This instance has 64 lock registers.
		4h	This instance has 128 lock registers.
		8h	This instance has 256 lock registers.
23-16	Reserved	0	Reserved
15	IU7	0	In-Use flag 7. Reads always return 0.
14	IU6	0	In-Use flag 6. Reads always return 0.
13	IU5	0	In-Use flag 5. Reads always return 0.
12	IU4	0	In-Use flag 4. Reads always return 0.
11	IU3	0	In-Use flag 3. Reads always return 0.
10	IU2	0	In-Use flag 2. Reads always return 0.
9	IU1		In-Use flag 1, covering lock registers 0–31.
		0	All lock registers 0–31 are in the Not Taken state.
		1	At least one of the lock registers 0–31 is in the Taken state.
8	IU0		In-Use flag 0, covering lock registers 32–63.
		0	All lock registers 32–63 are in the Not Taken state.
		1	At least one of the lock registers 32–63 is in the Taken state.
7-1	Reserved	0	Reserved
0	RESETDONE		Reset done status.
		0	Reset is in progress.
		1	Reset is completed.

1.14.5.4 Lock Register (SPINLOCK_LOCK_REG_i)

The Spin Lock Lock Register (SPINLOCK_LOCK_REG_i) is shown in Figure 1-99 and described in Table 1-134.

NOTE: i = 0 to 63

Figure 1-99. Lock Register (SPINLOCK_LOCK_REG_i)

31	1	0
Reserved		TAKEN
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-134. Lock Register (SPINLOCK_LOCK_REG_i) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	TAKEN		Lock State
		0	Read: Lock was previously Not Taken (free). The requester is granted the lock.
		1	Read: Lock was previously Taken. The requester is not granted the lock and must retry.
		0	Write: Set the lock to Not Taken (free).
		1	Write: No update to the lock value.

1.15 Error Location Module

1.15.1 Error Location Module Overview

Non-managed NAND flash memories can be dense and nonvolatile in their own nature, but error-prone. When reading from NAND flash memories, some level of error-correction is required. In the case of NAND modules with no internal correction capability, sometimes referred to as *bare NANDs*, the correction process is delegated to the memory controller.

The general-purpose memory controller (GPMC) probes data read from an external NAND flash and uses this to compute checksum-like information, called syndrome polynomials, on a per-block basis. Each syndrome polynomial gives a status of the read operations for a full block, including 512 bytes of data, parity bits, and an optional spare-area data field, with a maximum block size of 1023 bytes. Computation is based on a Bose-ChaudhurI-Hocquenghem (BCH) algorithm. The error-location module (ELM) extracts error addresses from these syndrome polynomials.

Based on the syndrome polynomial value, the ELM can detect errors, compute the number of errors, and give the location of each error bit. The actual data is not required to complete the error-correction algorithm. Errors can be reported anywhere in the NAND flash block, including in the parity bits.

The maximum acceptable number of errors that can be corrected depends on a programmable configuration parameter. 4-, 8-, and 16-bit error-correction levels are supported. The ELM relies on a static and fixed definition of the generator polynomial for each error-correction level that corresponds to the generator polynomials defined in the GPMC (there are three fixed polynomial for the three correction error levels). A larger number of errors than the programmed error-correction level may be detected, but the ELM cannot correct them all. The offending block is then tagged as *uncorrectable* in the associated computation exit status register. If the computation is successful, that is, if the number of errors detected does not exceed the maximum value authorized for the chosen correction capability, the exit status register contains the information on the number of detected errors.

When the error-location process completes, an interrupt is triggered to inform the central processing unit (CPU) that its status can be checked. The number of detected errors and their locations in the NAND block can be retrieved from the module through register accesses.

1.15.1.1 ELM Features

The ELM has the following features:

- 4, 8, and 16 bits per 512-byte block error-location based on BCH algorithms
- Eight simultaneous processing contexts
- Page-based and continuous modes
- Interrupt generation on error-location process completion:
  - When the full page has been processed in page mode
  - For each syndrome polynomial in continuous mode

1.15.2 ELM Integration

The ELM extracts error addresses from generated syndrome polynomials.

The ELM is used with the GPMC. Syndrome polynomials generated on-the-fly when reading a NAND flash page and stored in GPMC registers are passed to the ELM. The microprocessor unit (MPU) can then correct the data block by flipping the bits to which the ELM error-location outputs point.

Figure 1-100 shows the integration of the ELM subsystem in the device.

Figure 1-100. ELM Integration

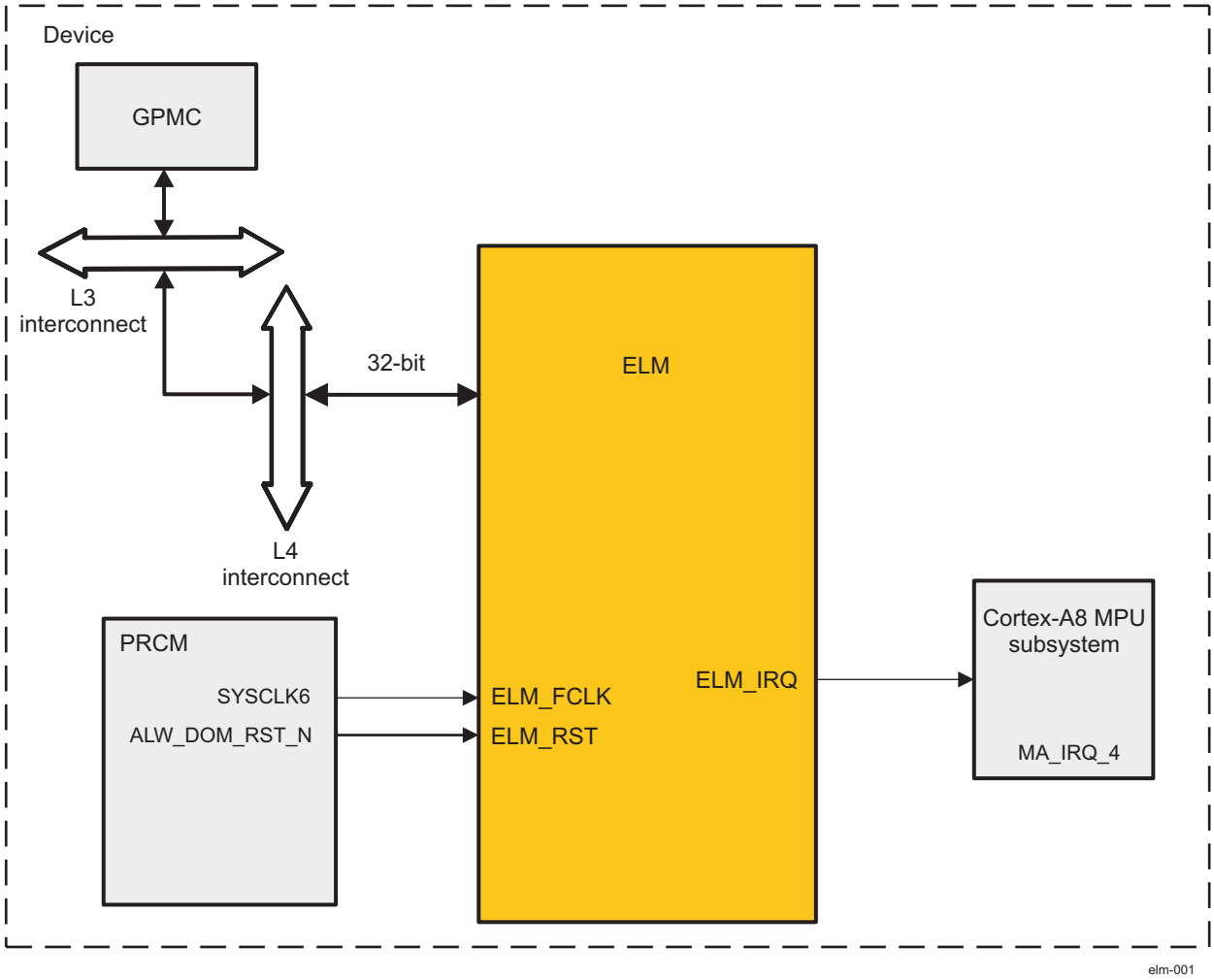


Table 1-135. Integration Attributes

Module Instance	Attributes		
	Power Domain	Wake-Up Capability	Interconnect
ELM	PD_ALWAYS_ON	No	NA

Table 1-136. Clocks and Resets

Clocks				
Module Instance	**Destination Signal Name**	**Source Signal Name**	**Source**	**Description**
ELM	ELM_FCLK	SYSCLK6	PRCM	Functional clock
Resets				
ELM	ELM_RST	ALW_DOM_RST_N	PRCM	Power domain hardware reset

Table 1-137. Hardware Requests

Interrupt Requests				
Module Instance	**Source Signal Name**	**Destination Signal Name**	**Destination**	**Description**
ELM	ELM_IRQ	MA_IRQ_4	Cortex-A8	BCH error-location module interrupt

1.15.3 ELM Functional Description

The ELM is designed around the error-location engine, which handles the computation based on the input syndrome polynomials.

The ELM maps the error-location engine to a standard interconnect interface by using a set of registers to control inputs and outputs.

1.15.3.1 ELM Software Reset

To perform a software reset, write a 1 to the ELM_SYSCONFIG[1] SOFTRESET bit. The ELM_SYSSTATUS[0] RESETDONE bit indicates that the software reset is complete when its value is 1. When the software reset completes, the ELM_SYSCONFIG[1] SOFTRESET bit is automatically reset.

1.15.3.2 ELM Power Management

Table 1-138 describes the power-management features available to the ELM module.

Table 1-138. Local Power Management Features

Feature	Registers	Description
Clock autogating	ELM_SYSCONFIG[0] AUTOGATING bit	This bit allows a local power optimization inside the module by gating the ELM_FCLK clock upon the interface activity.
Slave idle modes	ELM_SYSCONFIG[4:3] SIDLEMODE bit field	Force-idle, No-idle, and Smart-idle modes are available.
Clock activity	ELM_SYSCONFIG[8] CLOCKACTIVITY bit	The clock can be switched-off or maintained during the wake-up period.
Master Standby modes	N/A	
Global Wake-up Enable	N/A	
Wake-up Sources Enable	N/A	

CAUTION

The PRCM module has no hardware means of reading CLOCKACTIVITY settings. Thus, software must ensure consistent programming between the ELM CLOCKACTIVITY and ELM clock PRCM control bits.

1.15.3.3 ELM Interrupt Requests

Table 1-139 lists the event flags, and their masks, that can cause module interrupts.

Table 1-139. Events

Event Flag	Event Mask	Map to	Description
ELM_IRQSTATUS[8] PAGE_VALID	ELM_IRQENABLE[8] PAGE_MASK	ELM_IRQ	Page interrupt
ELM_IRQSTATUS[7] LOC_VALID_7	ELM_IRQENABLE[7] LOCATION_MASK_7	ELM_IRQ	Error-location interrupt for syndrome polynomial 7
ELM_IRQSTATUS[6] LOC_VALID_6	ELM_IRQENABLE[6] LOCATION_MASK_6	ELM_IRQ	Error-location interrupt for syndrome polynomial 6
ELM_IRQSTATUS[5] LOC_VALID_5	ELM_IRQENABLE[5] LOCATION_MASK_5	ELM_IRQ	Error-location interrupt for syndrome polynomial 5
ELM_IRQSTATUS[4] LOC_VALID_4	ELM_IRQENABLE[4] LOCATION_MASK_4	ELM_IRQ	Error-location interrupt for syndrome polynomial 4
ELM_IRQSTATUS[3] LOC_VALID_3	ELM_IRQENABLE[3] LOCATION_MASK_3	ELM_IRQ	Error-location interrupt for syndrome polynomial 3
ELM_IRQSTATUS[2] LOC_VALID_2	ELM_IRQENABLE[2] LOCATION_MASK_2	ELM_IRQ	Error-location interrupt for syndrome polynomial 2
ELM_IRQSTATUS[1] LOC_VALID_1	ELM_IRQENABLE[1] LOCATION_MASK_1	ELM_IRQ	Error-location interrupt for syndrome polynomial 1
ELM_IRQSTATUS[0] LOC_VALID_0	ELM_IRQENABLE[0] LOCATION_MASK_0	ELM_IRQ	Error-location interrupt for syndrome polynomial 0

1.15.3.4 Processing Initialization

ELM_LOCATION_CONFIG global setting parameters must be set before using the error-location engine. The ELM_LOCATION_CONFIG[1:0] ECC_BCH_LEVEL bit defines the error-correction level used (4-,8-, or 16-bit error-correction). The ELM_LOCATION_CONFIG[26:16] ECC_SIZE bit field defines the maximum buffer length beyond which the engine processing no longer looks for errors.

The CPU can choose to use the ELM in continuous mode or page mode. If all ELM_PAGE_CTRL[i] SECTOR_i bits are reset (i is the syndrome polynomial number, i = 0 to 7), continuous mode is used. In any other case, page mode is implicitly selected.

- Continuous mode: Each syndrome polynomial is processed independently – results for a syndrome can be retrieved and acknowledged at any time, whatever the status of the other seven processing contexts.
- Page mode: Syndrome polynomials are grouped into atomic entities – only one page can be processed at any given time, even if all eight contexts are not used for this page. Unused contexts are lost and cannot be affected to any other processing. The full page must be acknowledged and cleared before moving to the next page.

For completion interrupts to be generated correctly, all ELM_IRQENABLE[i] LOCATION_MASK_i bits (i = 0 to 7) must be forced to 0 when in page mode, and set to 1 in continuous mode. Additionally, the ELM_IRQENABLE[8] PAGE_MASK bit must be set to 1 when in page mode.

The CPU initiates error-location processing by writing a syndrome polynomial into one of the eight possible register sets. Each of these register sets includes seven registers: ELM_SYNDROME_FRAGMENT_0_i to ELM_SYNDROME_FRAGMENT_6_i. The first six registers can be written in any order, but ELM_SYNDROME_FRAGMENT_6_i must be written last because it includes the validity bit, which instructs the ELM that this syndrome polynomial must be processed (the ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID bit).

As soon as one validity bit is asserted (ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID = 0x1, with i = 0 to 7), error-location processing can start for the corresponding syndrome polynomial. The associated ELM_LOCATION_STATUS_i and ELM_ERROR_LOCATION_0_i to ELM_ERROR_LOCATION_15_i registers are not reset (i = 0 to 7). The software must not consider them until the corresponding ELM_IRQSTATUS[i] LOC_VALID_i bit is set.

1.15.3.5 Processing Sequence

While the error-location engine is busy processing one syndrome polynomial, further syndrome polynomials can be written. They are processed when the current processing completes.

The engine completes early when:

- No error is detected; that is, when the ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE bit is set to 1 and the ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS bit field is set to 0x0.
- Too many errors are detected; that is, when the ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE bit is set to 0 while the ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS bit field is set with the value output by the error-location engine. The reported number of errors is not ensured if ECC_CORRECTABLE is 0.

If the engine completes early, the associated error-location registers ELM_ERROR_LOCATION_0_i to ELM_ERROR_LOCATION_15_i are not updated (i = 0 to 7).

In all other cases, the engine goes through the entire error-location process. Each time an error-location is found, it is logged in the associated ECC_ERROR_LOCATION bit field. The first error detected is logged in the ELM_ERROR_LOCATION_0_i[12:0] ECC_ERROR_LOCATION bit field; the second in the ELM_ERROR_LOCATION_1_i[12:0] ECC_ERROR_LOCATION bit field, and so on.

Table 1-140. ELM_LOCATION_STATUS_i Value Decoding Table

ECC_CORRECTABLE Value	ECC_NB_ERRORS Value	Status	Number of Errors Detected	Action Required
1	0	OK	0	None
1	≠ 0	OK	ECC_NB_ERRORS	Correct the data buffer read based on the ELM_ERROR_LOCATION_0_i to ELM_ERROR_LOCATION_15_i results.
0	Any	Failed	Unknown	Software-dependant

1.15.3.6 Processing Completion

When the processing for a given syndrome polynomial completes, its ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID bit is reset. It must not be set again until the exit status registers, ELM_LOCATION_STATUS_i (i = 0 to 7), for this processing are checked. Failure to comply with this rule leads to potential loss of the first polynomial process data output.

The error-location engine signals the process completion to the ELM. When this event is detected, the corresponding ELM_IRQSTATUS[i] LOC_VALID_i bit is set (i = 0 to 7). The processing-exit status is available from the associated ELM_LOCATION_STATUS_i register, and error locations are stored in order in the ECC_ERROR_LOCATION fields. The software must only read valid error-location registers based on the number of errors detected and located.

Immediately after the error-location engine completes, a new syndrome polynomial can be processed, if any is available, as reported by the ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID validity bit, depending on the configured error-correction level. If several syndrome polynomials are available, a round-robin arbitration is used to select one for processing.

In continuous mode (that is, all bits in ELM_PAGE_CTRL are reset), an interrupt is triggered whenever a ELM_IRQSTATUS[i] LOC_VALID_i bit is asserted. The CPU must read the ELM_IRQSTATUS register to determine which polynomial is processed and retrieve the exit status and error locations (ELM_LOCATION_STATUS_i and ELM_ERROR_LOCATION_0_i to ELM_ERROR_LOCATION_15_i). When done, the CPU must clear the corresponding ELM_IRQSTATUS[i] LOC_VALID_i bit by writing it to 1. Other status bits must be written to 0 so that other interrupts are not unintentionally cleared. When using this mode, the ELM_IRQSTATUS[8] PAGE_VALID interrupt is never triggered.

In page mode, the module does not trigger interrupts for the processing completion of each polynomial because the ELM_IRQENABLE[i] LOCATION_MASK_i bits are cleared. A page is defined using the ELM_PAGE_CTRL register. Each SECTOR_i bit set means the corresponding polynomial i is part of the page processing. A page is fully processed when all tagged polynomials have been processed, as logged in the ELM_IRQSTATUS[i] LOC_VALID_i bit fields. The module triggers an ELM_IRQSTATUS[8] PAGE_VALID interrupt whenever it detects that the full page has been processed. To make sure the next page can be correctly processed, all status bits in the ELM_IRQSTATUS register must be cleared by using a single atomic-write access.

NOTE: Do not modify page setting parameters in the ELM_PAGE_CTRL register unless the engine is idle, no polynomial input is valid, and all interrupts have been cleared.

Because no polynomial-level interrupt is triggered in page mode, polynomials cleared in the ELM_PAGE_CTRL[i] SECTOR_i bit fields (i = 0 to 7) are processed as usual, but are essentially ignored. The CPU must manually poll the ELM_IRQSTATUS bits to check for their status.

1.15.4 ELM Basic Programming Model

1.15.4.1 ELM Low Level Programming Model

1.15.4.1.1 Processing Initialization

Table 1-141. ELM Processing Initialization

Step	Register/ Bit Field / Programming Model	Value
Resets the module	ELM_SYSCONFIG[1] SOFTRESET	0x1
Wait until reset is done.	ELM_SYSSTATUS[0] RESETDONE	0x1
Configure the slave interface power management.	ELM_SYSCONFIG[4:3] SIDLEMODE	Set value
Defines the error-correction level used	ELM_LOCATION_CONFIG[1:0] ECC_BCH_LEVEL	Set value
Defines the maximum buffer length	ELM_LOCATION_CONFIG[26:16] ECC_SIZE	Set value
Sets the ELM in continuous mode or page mode	ELM_PAGE_CTRL	Set value
If continuous mode is used	All ELM_PAGE_CTRL[i] SECTOR_i (i = 0 to 7)	0x0
Enables interrupt for syndrome polynomial i	ELM_IRQENABLE[i] LOCATION_MASK_i	0x1
else (page mode is used)	One syndrome polynomial i is set ELM_PAGE_CTRL[i] SECTOR_i (i = 0 to 7)	0x1
Disable all interrupts for syndrome polynomial and enable PAGE_MASK interrupt.	All ELM_IRQENABLE[i] LOCATION_MASK_i = 0x0 and ELM_IRQENABLE[8] PAGE_MASK = 0x1	Set value
endif		Set value
Set the input syndrome polynomial i.	ELM_SYNDROME_FRAGMENT_0_i	Set value
	ELM_SYNDROME_FRAGMENT_1_i	Set value
	ELM_SYNDROME_FRAGMENT_5_i	Set value
	ELM_SYNDROME_FRAGMENT_6_i	Set value
Initiates the computation process	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID	0x1

1.15.4.1.2 Read Results

The engine goes through the entire error-location process and results can be read. Table 1-142 and Table 1-143 describe the processing completion for continuous and page modes, respectively.

Table 1-142. ELM Processing Completion for Continuous Mode

Step	Register/ Bit Field / Programming Model	Value
Wait until process is complete for syndrome polynomial i: Wait until the ELM_IRQ interrupt is generated, or poll the status register.		
Read for which i the error-location process is complete.	ELM_IRQSTATUS[i] LOC_VALID_i	0x1
if the process fails (too many errors)	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE	0x0
It is software dependant.		
else (process successful, the engine completes)	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE	0x1
Read the number of errors.	ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS	
Read the error-location bit addresses for syndrome polynomial i of the ECC_NB_ERRORS first registers. It is the software responsibility to correct errors in the data buffer.	ELM_ERROR_LOCATION_0_i[12:0] ECC_ERROR_LOCATION	
	ELM_ERROR_LOCATION_1_i[12:0] ECC_ERROR_LOCATION	
	...	
	ELM_ERROR_LOCATION_15_i[12:0] ECC_ERROR_LOCATION	
endif		
Clear the corresponding i interrupt.	ELM_IRQSTATUS[i] LOC_VALID_i	0x1

A new syndrome polynomial can be processed after the end of processing (ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID = 0x0) and after the exit status register check (ELM_LOCATION_STATUS_i).

Table 1-143. ELM Processing Completion for Page Mode

Step	Register/ Bit Field / Programming Model	Value
Wait until process is complete for syndrome polynomial i: Wait until the ELM_IRQ interrupt is generated, or poll the status register.		
Wait for page completed interrupt: All error locations are valid.	ELM_IRQSTATUS[8] PAGE_VALID	0x1
Repeat the following actions the necessary number of times. That is, once for each valid defined block in the page.		
Read the process exit status.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE	
if the process fails (too many errors)	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE	0x0
It is software dependant.		
else (process successful, the engine completes)	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE	0x1
Read the number of errors.	ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS	
Read the error-location bit addresses for syndrome polynomial i of the ECC_NB_ERRORS first registers.	ELM_ERROR_LOCATION_0_i[12:0] ECC_ERROR_LOCATION	
	ELM_ERROR_LOCATION_1_i[12:0] ECC_ERROR_LOCATION	
	...	
	ELM_ERROR_LOCATION_15_i[12:0] ECC_ERROR_LOCATION	
endif		
End Repeat		
Clear the ELM_IRQSTATUS register.	ELM_IRQSTATUS	0x1FF

Next page can be correctly processed after a page is fully processed, when all tagged polynomials have been processed (ELM_IRQSTATUS[i] LOC_VALID_i = 0x1 for all syndrome polynomials i used in the page).

1.15.4.2 Use Case: ELM Used in Continuous Mode

In this example, the ELM module is programmed for an 8-bit error-correction capability in continuous mode. After reading a 528-byte NAND flash sector (512B data plus 16B spare area) with a 16-bit interface, a non-zero polynomial syndrome is reported from the GPMC (Polynomial syndrome 0 is used in the ELM):

- P = 0x0A16ABE115E44F767BFB0D0980

Table 1-144. Use Case: Continuous Mode

Step	Register/ Bit Field / Programming Model	Value
Resets the module	ELM_SYSCONFIG[1] SOFTRESET	0x1
Wait until reset is done.	ELM_SYSSTATUS[0] RESETDONE	0x1
Configure the slave interface power management: Smart idle is used.	ELM_SYSCONFIG[4:3] SIDLEMODE	0x2
Defines the error-correction level used: 8 bits	ELM_LOCATION_CONFIG[1:0] ECC_BCH_LEVEL	0x1
Defines the maximum buffer length: 528 bytes (2x528 = 1056)	ELM_LOCATION_CONFIG[26:16] ECC_SIZE	0x420
Sets the ELM in continuous mode	ELM_PAGE_CTRL	0
Enables interrupt for syndrome polynomial 0	ELM_IRQENABLE[0] LOCATION_MASK_0	0x1
Set the input syndrome polynomial 0.	ELM_SYNDROME_FRAGMENT_0_i (i=0)	0xFB0D0980
	ELM_SYNDROME_FRAGMENT_1_i (i=0)	0xE44F767B
	ELM_SYNDROME_FRAGMENT_2_i (i=0)	0x16ABE115
	ELM_SYNDROME_FRAGMENT_3_i (i=0)	0x0000000A
Initiates the computation process	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID (i=0)	0x1
Wait until process is complete for syndrome polynomial 0: IRQ_ELM is generated or poll the status register.		
Read that error-location process is complete for syndrome polynomial 0.	ELM_IRQSTATUS[0] LOC_VALID_0	0x1
Read the process exit status: All errors were successfully located.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE (i=0)	0x1
Read the number of errors: Four errors detected.	ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS (i=0)	0x4
Read the error-location bit addresses for syndrome polynomial 0 of the 4 first registers: Errors are located in the data buffer at decimal addresses 431, 1062, 1909, 3452.	ELM_ERROR_LOCATION_0_i (i=0)	0x1AF
	ELM_ERROR_LOCATION_1_i (i=0)	0x426
	ELM_ERROR_LOCATION_2_i (i=0)	0x775
	ELM_ERROR_LOCATION_3_i (i=0)	0xD7C
Clear the corresponding interrupt for polynomial 0.	ELM_IRQSTATUS[0] LOC_VALID_0	0x1

The NAND flash data in the sector are seen as a polynomial of degree 4223 (number of bits in a 528 byte buffer minus 1), with each data bit being a coefficient in the polynomial. When reading from a NAND flash using the GPMC module, computation of the polynomial syndrome assumes that the first NAND word read at address 0x0 contains the highest-order coefficient in the message. Furthermore, in the 16-bit NAND word, bits are ordered from bit 7 to bit 0, then from bit 15 to bit 8. Based on this convention, an address table of the data buffer can be built. NAND memory addresses in Table 1-145 are given in decimal format.

Table 1-145. 16-bit NAND Sector Buffer Address Map

NAND Memory Address	Message bit addresses in the memory word															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	4215	4214	4213	4212	4211	4210	4209	4208	4223	4222	4221	4220	4219	4218	4217	4216
1	4175	4174	4173	4172	4171	4170	4169	4168	4183	4182	4181	4180	4179	4178	4177	4176
...																
47	3463	3462	3461	3460	3459	3458	3457	3456	3471	3470	3469	3468	3467	3466	3465	3464
48	3447	3446	3445	3444	3443	3442	3441	3440	3455	3454	3453	3452	3451	3450	3449	3448
49	3431	3430	3429	3428	3427	3426	3425	3424	3439	3438	3437	3436	3435	3434	3433	3432
50	3415	3414	3413	3412	3411	3410	3409	3408	3423	3422	3421	3420	3419	3418	3417	3416
...																
255	135	134	133	132	131	130	129	128	143	142	141	140	139	138	137	136
256	119	118	117	116	115	114	113	112	127	126	125	124	123	122	121	120
257	103	102	101	100	99	98	97	96	111	110	109	108	107	106	105	104
258	87	86	85	84	83	82	81	80	95	94	93	92	91	90	89	88
259	71	70	69	68	67	66	65	64	79	78	77	76	75	74	73	72
260	55	54	53	52	51	50	49	48	63	62	61	60	59	58	57	56
261	39	38	37	36	35	34	33	32	47	46	45	44	43	42	41	40
262	23	22	21	20	19	18	17	16	31	30	29	28	27	26	25	24
263	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8

The table can now be used to determine which bits in the buffer were incorrect and must be flipped. In this example, the first bit to be flipped is bit 4 from the 49th byte read from memory. It is up to the processor to correctly map this word to the copied buffer and to flip this bit. The same process must be repeated for all detected errors.

1.15.4.3 Use Case: ELM Used in Page Mode

In this example, the ELM module is programmed for an 16-bit error-correction capability in page mode. After reading a 528-byte NAND flash sector (512B data plus 16B spare area) with a 16-bit interface, four non-zero polynomial syndromes are reported from the GPMC (Polynomial syndrome 0, 1, 2, and 3 are used in the ELM):

- P0 = 0xE8B0 12ADDB5A318E05BE B0693DB28330B5CC A329AA05E0B718EF
- P1 = 0xBAD0 49A0D932C22E6669 0948DF08BE093336 79C6BA10E5F935EB
- P2 = 0x69D9 B86ABCD5EC3697FA A6498FEE54556EA0 1579EF7D60BA3189
- P3 = 0x0

Table 1-146. Use Case: Page Mode

Step	Register/ Bit Field / Programming Model	Value
Resets the module	ELM_SYSCONFIG[1] SOFTRESET	0x1
Wait until reset is done.	ELM_SYSSTATUS[0] RESETDONE	0x1
Configure the slave interface power management: Smart idle is used.	ELM_SYSCONFIG[4:3] SIDLEMODE	0x2
Defines the error-correction level used: 16 bits	ELM_LOCATION_CONFIG[1:0] ECC_BCH_LEVEL	0x2
Defines the maximum buffer length: 528 bytes	ELM_LOCATION_CONFIG[26:16] ECC_SIZE	0x420
Sets the ELM in page mode (4 blocks in a page)	ELM_PAGE_CTRL[0] SECTOR_0	0x1
	ELM_PAGE_CTRL[1] SECTOR_1	0x1
	ELM_PAGE_CTRL[2] SECTOR_2	0x1
	ELM_PAGE_CTRL[3] SECTOR_3	0x1
Disable all interrupts for syndrome polynomial and enable PAGE_MASK interrupt.	ELM_IRQENABLE	0x100
Set the input syndrome polynomial 0.	ELM_SYNDROME_FRAGMENT_0_i (i=0)	0xE0B718EF
	ELM_SYNDROME_FRAGMENT_1_i (i=0)	0xA329AA05
	ELM_SYNDROME_FRAGMENT_2_i (i=0)	0x8330B5CC
	ELM_SYNDROME_FRAGMENT_3_i (i=0)	0xB0693DB2
	ELM_SYNDROME_FRAGMENT_4_i (i=0)	0x318E05BE
	ELM_SYNDROME_FRAGMENT_5_i (i=0)	0x12ADDB5A
	ELM_SYNDROME_FRAGMENT_6_i (i=0)	0xE8B0
Set the input syndrome polynomial 1.	ELM_SYNDROME_FRAGMENT_0_i (i=1)	0xE5F935EB
	ELM_SYNDROME_FRAGMENT_1_i (i=1)	0x79C6BA10
	ELM_SYNDROME_FRAGMENT_2_i (i=1)	0xBE093336
	ELM_SYNDROME_FRAGMENT_3_i (i=1)	0x0948DF08
	ELM_SYNDROME_FRAGMENT_4_i (i=1)	0xC22E6669
	ELM_SYNDROME_FRAGMENT_5_i (i=1)	0x49A0D932
	ELM_SYNDROME_FRAGMENT_6_i (i=1)	0xBAD0
Set the input syndrome polynomial 2.	ELM_SYNDROME_FRAGMENT_0_i (i=2)	0x60BA3189
	ELM_SYNDROME_FRAGMENT_1_i (i=2)	0x1579EF7D
	ELM_SYNDROME_FRAGMENT_2_i (i=2)	0x54556EA0
	ELM_SYNDROME_FRAGMENT_3_i (i=2)	0xA6498FEE
	ELM_SYNDROME_FRAGMENT_4_i (i=2)	0xEC3697FA
	ELM_SYNDROME_FRAGMENT_5_i (i=2)	0xB86ABCD5
	ELM_SYNDROME_FRAGMENT_6_i (i=2)	0x69D9

Table 1-146. Use Case: Page Mode (continued)

Step	Register/ Bit Field / Programming Model	Value
Set the input syndrome polynomial 3.	ELM_SYNDROME_FRAGMENT_0_i (i=3)	0x0
	ELM_SYNDROME_FRAGMENT_1_i (i=3)	0x0
	ELM_SYNDROME_FRAGMENT_2_i (i=3)	0x0
	ELM_SYNDROME_FRAGMENT_3_i (i=3)	0x0
	ELM_SYNDROME_FRAGMENT_4_i (i=3)	0x0
	ELM_SYNDROME_FRAGMENT_5_i (i=3)	0x0
	ELM_SYNDROME_FRAGMENT_6_i (i=3)	0x0
Initiates the computation process for syndrome polynomial 0	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID (i=0)	0x1
Initiates the computation process for syndrome polynomial 1	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID (i=1)	0x1
Initiates the computation process for syndrome polynomial 2	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID (i=2)	0x1
Initiates the computation process for syndrome polynomial 3	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID (i=3)	0x1
Wait until process is complete for syndrome polynomial 0, 1, 2, and 3: Wait until the ELM_IRQ interrupt is generated or poll the status register.		
Wait for page completed interrupt: All error locations are valid.	ELM_IRQSTATUS[8] PAGE_VALID	0x1
Read the process exit status for syndrome polynomial 0: All errors were successfully located.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE (i=0)	0x1
Read the process exit status for syndrome polynomial 1: All errors were successfully located.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE (i=1)	0x1
Read the process exit status for syndrome polynomial 2: All errors were successfully located.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE (i=2)	0x1
Read the process exit status for syndrome polynomial 3: All errors were successfully located.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE (i=3)	0x1
Read the number of errors for syndrome polynomial 0: 4 errors detected.	ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS (i=0)	0x4
Read the number of errors for syndrome polynomial 1: 2 errors detected.	ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS (i=1)	0x2
Read the number of errors for syndrome polynomial 2: 1 error detected.	ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS (i=2)	0x1
Read the number of errors for syndrome polynomial 3: 0 errors detected.	ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS (i=3)	0x0
Read the error-location bit addresses for syndrome polynomial 0 of the 4 first registers:	ELM_ERROR_LOCATION_0_i (i=0)	0x1FE
	ELM_ERROR_LOCATION_1_i (i=0)	0x617
	ELM_ERROR_LOCATION_2_i (i=0)	0x650
	ELM_ERROR_LOCATION_3_i (i=0)	0xA83
Read the error-location bit addresses for syndrome polynomial 1 of the 2 first registers:	ELM_ERROR_LOCATION_0_i (i=1)	0x4
	ELM_ERROR_LOCATION_1_i (i=1)	0x1036
Read the errors location bit addresses for syndrome polynomial 2 of the first registers:	ELM_ERROR_LOCATION_0_i (i=1)	0x3E8
Clear the ELM_IRQSTATUS register.	ELM_IRQSTATUS	0x1FF

1.15.5 ELM Registers

Table 1-147. ELM Registers

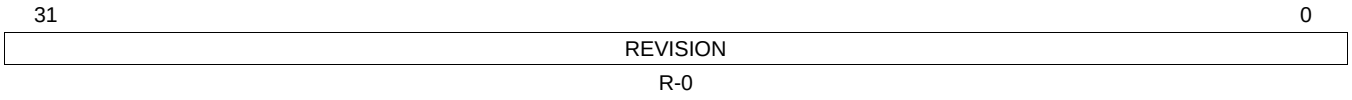
Address Offset	Acronym	Description	Section
0h	ELM_REVISION	ELM Revision Register	Section 1.15.5.1
10h	ELM_SYSCONFIG	ELM System Configuration Register	Section 1.15.5.2
14h	ELM_SYSSTATUS	ELM System Status Register	Section 1.15.5.3
18h	ELM_IRQSTATUS	ELM Interrupt Status Register	Section 1.15.5.4
1Ch	ELM_IRQENABLE	ELM Interrupt Enable Register	Section 1.15.5.5
20h	ELM_LOCATION_CONFIG	ELM Location Configuration Register	Section 1.15.5.6
80h	ELM_PAGE_CTRL	ELM Page Definition Register	Section 1.15.5.7
400h + (40h × i)	ELM_SYNDROME_FRAGMENT_0_i[(1)]	ELM_SYNDROME_FRAGMENT_0_i Register	Section 1.15.5.8
404h+ (40h × i)	ELM_SYNDROME_FRAGMENT_1_i	ELM_SYNDROME_FRAGMENT_1_i Register	Section 1.15.5.9
408h + (40h × i)	ELM_SYNDROME_FRAGMENT_2_i	ELM_SYNDROME_FRAGMENT_2_i Register	Section 1.15.5.10
40Ch + (40h × i)	ELM_SYNDROME_FRAGMENT_3_i	ELM_SYNDROME_FRAGMENT_3_i Register	Section 1.15.5.11
410h + (40h × i)	ELM_SYNDROME_FRAGMENT_4_i	ELM_SYNDROME_FRAGMENT_4_i Register	Section 1.15.5.12
414h + (40h × i)	ELM_SYNDROME_FRAGMENT_5_i	ELM_SYNDROME_FRAGMENT_5_i Register	Section 1.15.5.13
418h + (40h × i)	ELM_SYNDROME_FRAGMENT_6_1	ELM_SYNDROME_FRAGMENT_6_1 Register	Section 1.15.5.14
800h + (100h × i)	ELM_LOCATION_STATUS_i	ELM_LOCATION_STATUS_i Register	Section 1.15.5.15
880h-8BCh + (100h × i)	ELM_ERROR_LOCATION_0-15_i	ELM_ERROR_LOCATION_0-15_i Registers	Section 1.15.5.16

(1) i = 0 to 8

1.15.5.1 ELM Revision Register (ELM_REVISION)

This register contains the IP revision code. The ELM Revision Register (ELM_REVISION) is shown in Figure 1-101 and described in Table 1-148.

Figure 1-101. ELM Revision Register (ELM_REVISION)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

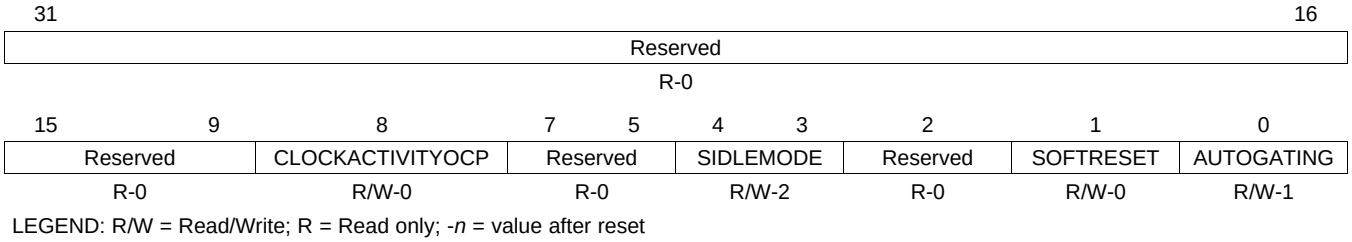
Table 1-148. ELM Revision Register (ELM_REVISION) Field Descriptions

Bit	Field	Value	Description
31-0	REVISION	0-FFFF FFFFh	IP Revision

1.15.5.2 ELM System Configuration Register (ELM_SYSCONFIG)

This register allows controlling various parameters of the OCP interface. The ELM System Configuration Register (ELM_SYSCONFIG) is shown in Figure 1-102 and described in Table 1-149.

Figure 1-102. ELM System Configuration Register (ELM_SYSCONFIG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-149. ELM System Configuration Register (ELM_SYSCONFIG) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLOCKACTIVITY OCP		OCP Clock activity when module is in IDLE mode (during wake up mode period).
		0	OCP Clock can be switch-off
		1	OCP Clock is maintained during wake up period
7-5	Reserved	0	Reserved
4-3	SIDLEMODE		Slave interface power management (IDLE req/ack control).
		0	FORCE Idle. IDLE request is acknowledged unconditionally and immediately. (Default *Dumb* mode for safety)
		1h	NO idle. IDLE request is never acknowledged.
		2h	SMART Idle. The acknowledgment to an IDLE request is given based on the internal activity.
		3h	Reserved - do not use
2	Reserved	0	Reserved
1	SOFTRESET		Module software reset. This bit is automatically reset by hardware (During reads, it always returns 0.). It has same effect as the OCP hardware reset.
		0	Normal mode.
		1	Start soft reset sequence.
0	AUTOGATING		Internal OCP clock gating strategy. (No module visible impact other than saving power.)
		0	OCP clock is free-running.
		1	Automatic internal OCP clock gating strategy is applied based on the OCP interface activity.

1.15.5.3 ELM System Status Register (ELM_SYSSTATUS)

The ELM System Status Register (ELM_SYSSTATUS) is shown in Figure 1-103 and described in Table 1-150.

Figure 1-103. ELM System Status Register (ELM_SYSSTATUS)

31	1	0
Reserved		RESETDONE
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-150. ELM System Status Register (ELM_SYSSTATUS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	RESETDONE		Internal reset monitoring (OCP domain). Undefined since: From hardware perspective, the reset state is 0. From software user perspective, when the accessible module is 1.
		0	Reset is on-going
		1	Reset is done (completed)

1.15.5.4 ELM Interrupt Status Register (ELM_IRQSTATUS)

This register doubles as a status register for the error-location processes. The ELM Interrupt Status Register (ELM_IRQSTATUS) is shown in Figure 1-104 and described in Table 1-151.

Figure 1-104. ELM Interrupt Status Register (ELM_IRQSTATUS)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved							PAGE_VALID
R-0							R/W-0

7	6	5	4	3	2	1	0
LOC_VALID_7	LOC_VALID_6	LOC_VALID_5	LOC_VALID_4	LOC_VALID_3	LOC_VALID_2	LOC_VALID_1	LOC_VALID_0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-151. ELM Interrupt Status Register (ELM_IRQSTATUS) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	PAGE_VALID		Error-location status for a full page, based on the mask definition.
		0	Read: Error locations invalid for all polynomials enabled in the ECC_INTERRUPT_MASK register.
		1	Read: All error locations valid.
		0	Write: No effect.
		1	Write: Clear interrupt.
7	LOC_VALID_7		Error-location status for syndrome polynomial 7.
		0	Read: No syndrome processed or process in progress.
		1	Read: Error-location process completed.
		0	Write: No effect.
		1	Write: Clear interrupt.
6	LOC_VALID_6		Error-location status for syndrome polynomial 6.
		0	Read: No syndrome processed or process in progress.
		1	Read: Error-location process completed.
		0	Write: No effect.
		1	Write: Clear interrupt.
5	LOC_VALID_5		Error-location status for syndrome polynomial 5.
		0	Read: No syndrome processed or process in progress.
		1	Read: Error-location process completed.
		0	Write: No effect.
		1	Write: Clear interrupt.
4	LOC_VALID_4		Error-location status for syndrome polynomial 4.
		0	Read: No syndrome processed or process in progress.
		1	Read: Error-location process completed.
		0	Write: No effect.
		1	Write: Clear interrupt.
3	LOC_VALID_3		Error-location status for syndrome polynomial 3.
		0	Read: No syndrome processed or process in progress.
		1	Read: Error-location process completed.
		0	Write: No effect.
		1	Write: Clear interrupt.

Table 1-151. ELM Interrupt Status Register (ELM_IRQSTATUS) Field Descriptions (continued)

Bit	Field	Value	Description
2	LOC_VALID_2		Error-location status for syndrome polynomial 2.
		0	Read: No syndrome processed or process in progress.
		1	Read: Error-location process completed.
		0	Write: No effect.
		1	Write: Clear interrupt.
1	LOC_VALID_1		Error-location status for syndrome polynomial 1.
		0	Read: No syndrome processed or process in progress.
		1	Read: Error-location process completed.
		0	Write: No effect.
		1	Write: Clear interrupt.
0	LOC_VALID_0		Error-location status for syndrome polynomial 0.
		0	Read: No syndrome processed or process in progress.
		1	Read: Error-location process completed.
		0	Write: No effect.
		1	Write: Clear interrupt.

1.15.5.5 ELM Interrupt Enable Register (ELM_IRQENABLE)

The ELM Interrupt Enable Register (ELM_IRQENABLE) is shown in Figure 1-105 and described in Table 1-152.

Figure 1-105. ELM Interrupt Enable Register (ELM_IRQENABLE)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved							PAGE_MASK
R-0							R/W-0

7	6	5	4	3	2	1	0
LOCATION_MASK_7	LOCATION_MASK_6	LOCATION_MASK_5	LOCATION_MASK_4	LOCATION_MASK_3	LOCATION_MASK_2	LOCATION_MASK_1	LOCATION_MASK_0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-152. ELM Interrupt Enable Register (ELM_IRQENABLE) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	PAGE_MASK		Page interrupt mask bit
		0	Disable interrupt.
		1	Enable interrupt.
7	LOCATION_MASK_7		Error-location interrupt mask bit for syndrome polynomial 7.
		0	Disable interrupt.
		1	Enable interrupt.
6	LOCATION_MASK_6		Error-location interrupt mask bit for syndrome polynomial 6.
		0	Disable interrupt.
		1	Enable interrupt.
5	LOCATION_MASK_5		Error-location interrupt mask bit for syndrome polynomial 5.
		0	Disable interrupt.
		1	Enable interrupt.
4	LOCATION_MASK_4		Error-location interrupt mask bit for syndrome polynomial 4.
		0	Disable interrupt.
		1	Enable interrupt.
3	LOCATION_MASK_3		Error-location interrupt mask bit for syndrome polynomial 3.
		0	Disable interrupt.
		1	Enable interrupt.
2	LOCATION_MASK_2		Error-location interrupt mask bit for syndrome polynomial 2.
		0	Disable interrupt.
		1	Enable interrupt.
1	LOCATION_MASK_1		Error-location interrupt mask bit for syndrome polynomial 1.
		0	Disable interrupt.
		1	Enable interrupt.
0	LOCATION_MASK_0		Error-location interrupt mask bit for syndrome polynomial 0.
		0	Disable interrupt.
		1	Enable interrupt.

1.15.5.6 ELM Location Configuration Register (ELM_LOCATION_CONFIG)

The ELM Location Configuration Register (ELM_LOCATION_CONFIG) is shown in Figure 1-106 and described in Table 1-153.

Figure 1-106. ELM Location Configuration Register (ELM_LOCATION_CONFIG)

31 27	16 16	15 2	1 0
Reserved	ECC_SIZE	Reserved	ECC_BCH_LEVEL
R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-153. ELM Location Configuration Register (ELM_LOCATION_CONFIG) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	ECC_SIZE	0-7FFh	Maximum size of the buffers for which the error-location engine is used, in number of nibbles (4-bits entities)
15-2	Reserved	0	Reserved
1-0	ECC_BCH_LEVEL		Error correction level.
		0	4 bits.
		1h	8 bits.
		2h	16 bits.
		3h	Reserved.

1.15.5.7 ELM Page Definition Register (ELM_PAGE_CTRL)

The ELM Page Definition Register (ELM_PAGE_CTRL) is shown in Figure 1-107 and described in Table 1-154.

Figure 1-107. ELM Page Definition Register (ELM_PAGE_CTRL)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
SECTOR_7	SECTOR_6	SECTOR_5	SECTOR_4	SECTOR_3	SECTOR_2	SECTOR_1	SECTOR_0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

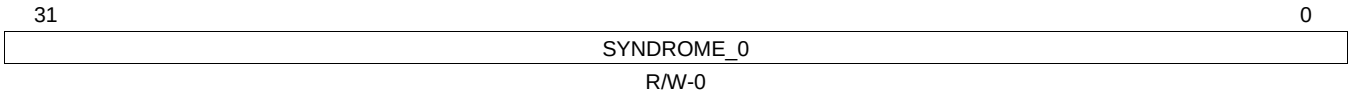
Table 1-154. ELM Page Definition Register (ELM_PAGE_CTRL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7	SECTOR_7	0-1	Set to 1 if syndrome polynomial 7 is part of the page in page mode. Must be 0 in continuous mode.
6	SECTOR_6	0-1	Set to 1 if syndrome polynomial 6 is part of the page in page mode. Must be 0 in continuous mode.
5	SECTOR_5	0-1	Set to 1 if syndrome polynomial 5 is part of the page in page mode. Must be 0 in continuous mode.
4	SECTOR_4	0-1	Set to 1 if syndrome polynomial 4 is part of the page in page mode. Must be 0 in continuous mode.
3	SECTOR_3	0-1	Set to 1 if syndrome polynomial 3 is part of the page in page mode. Must be 0 in continuous mode.
2	SECTOR_2	0-1	Set to 1 if syndrome polynomial 2 is part of the page in page mode. Must be 0 in continuous mode.
1	SECTOR_1	0-1	Set to 1 if syndrome polynomial 1 is part of the page in page mode. Must be 0 in continuous mode.
0	SECTOR_0	0-1	Set to 1 if syndrome polynomial 0 is part of the page in page mode. Must be 0 in continuous mode.

1.15.5.8 ELM_SYNDROME_FRAGMENT_0_i Register

The ELM_SYNDROME_FRAGMENT_0_i Register is shown in Figure 1-108 and described in Table 1-155.

Figure 1-108. ELM_SYNDROME_FRAGMENT_0_i Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

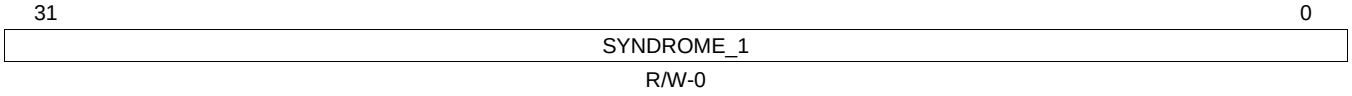
Table 1-155. ELM_SYNDROME_FRAGMENT_0_i Register Field Descriptions

Bit	Field	Value	Description
31-0	SYNDROME_0	0-FFFF FFFFh	Syndrome bits 0 to 31.

1.15.5.9 ELM_SYNDROME_FRAGMENT_1_i Register

The ELM_SYNDROME_FRAGMENT_1_i Register is shown in Figure 1-109 and described in Table 1-156.

Figure 1-109. ELM_SYNDROME_FRAGMENT_1_i Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

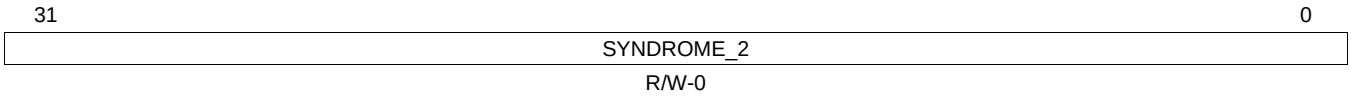
Table 1-156. ELM_SYNDROME_FRAGMENT_1_i Register Field Descriptions

Bit	Field	Value	Description
31-0	SYNDROME_1	0-FFFF FFFFh	Syndrome bits 32 to 63.

1.15.5.10 ELM_SYNDROME_FRAGMENT_2_i Register

The ELM_SYNDROME_FRAGMENT_2_i Register is shown in Figure 1-110 and described in Table 1-157.

Figure 1-110. ELM_SYNDROME_FRAGMENT_2_i Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-157. ELM_SYNDROME_FRAGMENT_2_i Register Field Descriptions

Bit	Field	Value	Description
31-0	SYNDROME_2	0-FFFF FFFFh	Syndrome bits 64 to 95.

1.15.5.11 ELM_SYNDROME_FRAGMENT_3_i Register

The ELM_SYNDROME_FRAGMENT_3_i Register is shown in Figure 1-111 and described in Table 1-158.

Figure 1-111. ELM_SYNDROME_FRAGMENT_3_i Register

31	0
SYNDROME_3	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-158. ELM_SYNDROME_FRAGMENT_3_i Register Field Descriptions

Bit	Field	Value	Description
31-0	SYNDROME_3	0-FFFF FFFFh	Syndrome bits 96 to 127.

1.15.5.12 ELM_SYNDROME_FRAGMENT_4_i Register

The ELM_SYNDROME_FRAGMENT_4_i Register is shown in Figure 1-112 and described in Table 1-159.

Figure 1-112. ELM_SYNDROME_FRAGMENT_4_i Register

31	0
SYNDROME_4	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-159. ELM_SYNDROME_FRAGMENT_4_i Register Field Descriptions

Bit	Field	Value	Description
31-0	SYNDROME_4	0-FFFF FFFFh	Syndrome bits 128 to 159.

1.15.5.13 ELM_SYNDROME_FRAGMENT_5_i Register

The ELM_SYNDROME_FRAGMENT_5_i Register is shown in Figure 1-113 and described in Table 1-160.

Figure 1-113. ELM_SYNDROME_FRAGMENT_5_i Register

31	0
SYNDROME_5	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-160. ELM_SYNDROME_FRAGMENT_5_i Register Field Descriptions

Bit	Field	Value	Description
31-0	SYNDROME_5	0-FFFF FFFFh	Syndrome bits 160 to 191.

1.15.5.14 ELM_SYNDROME_FRAGMENT_6_i Register

The ELM_SYNDROME_FRAGMENT_6_i Register is shown in Figure 1-114 and described in Table 1-161.

Figure 1-114. ELM_SYNDROME_FRAGMENT_6_i Register

31	17	16	15	0
Reserved		SYNDROME_VALID	SYNDROME_6	
R-0		R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-161. ELM_SYNDROME_FRAGMENT_6_i Register Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	SYNDROME_VALID		Syndrome valid bit.
		0	This syndrome polynomial should not be processed.
		1	This syndrome polynomial must be processed.
15-0	SYNDROME_6	0-FFFFh	Syndrome bits 192 to 207.

1.15.5.15 ELM_LOCATION_STATUS_i Register

The ELM_LOCATION_STATUS_i Register is shown in Figure 1-115 and described in Table 1-162.

Figure 1-115. ELM_LOCATION_STATUS_i Register

31	9	8	7	5	4	0
Reserved		ECC_CORRECTABL	Reserved		ECC_NB_ERRORS	
R-0		R-0	R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-162. ELM_LOCATION_STATUS_i Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	ECC_CORRECTABL		Error-location process exit status.
		0	ECC error-location process failed. Number of errors and error locations are invalid.
		1	All errors were successfully located. Number of errors and error locations are valid.
7-5	Reserved	0	Reserved
4-0	ECC_NB_ERRORS	0-1Fh	Number of errors detected and located.

1.15.5.16 ELM_ERROR_LOCATION_0-15_i Registers

The ELM_ERROR_LOCATION_0-15_i Registers is shown in Figure 1-116 and described in Table 1-163.

Figure 1-116. ELM_ERROR_LOCATION_0-15_i Registers

31	13	12	0
Reserved		ECC_ERROR_LOCATION	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-163. ELM_ERROR_LOCATION_0-15_i Registers Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved
12-0	ECC_ERROR_LOCATION	0-1FFFh	Error-location bit address.

1.16 Control Module

The Control module complies to the industry standard - OCP Specification Revision 2.2. It includes within it all chip-level registers, which are common to all interfaces whose functionality is general purpose configuration, control and status.

NOTE: Be advised that DSP, GEM, and C674x are used interchangeably throughout this section.

The registers can be classified into the following groups:

Device BOOT Registers
These include the boot control, status and boot address registers.

PLL Control Registers
These include the control and configuration registers for main PLL, Audio PLL, video PLL and DDR PLLs.

Device Configuration Registers
These registers are used to for general purpose configuration, control and status information. Some of the functionality of these registers include - clock and oscillator control, initiator priorities, system MMU configuration, DDR control register, C674x DSP state management, USB control and USB PHY control, Ethernet MAC 48-bit ID, PCIe configuration, RTC control, Audio mute control, HD DACs control, SD DAC control, and HW Event select registers.

PAD Configuration Registers
These set of registers control pin multiplexing and pull up/pull down control per pin of the device.

1.16.1 Control Module Registers

1.16.1.1 Device BOOT Registers

Table 1-164 lists registers that control various boot aspects of the device. For the base address of these registers, see Table 1-12.

Table 1-164. PLL BOOT Registers

Offset	Acronym	Register Description	Section
40h	CONTROL_STATUS	Control Status Register	Section 1.16.1.1.1
44h	BOOTSTAT	Boot Status Register	Section 1.16.1.1.2
48h	DSPBOOTADDR	DSP Boot Address Vector	Section 1.16.1.1.3

1.16.1.1.1 Control Status Register (CONTROL_STATUS)

The CONTROL_STATUS register reflects the system boot and the device type configuration values as sampled when the power-on reset (PORz) or warm reset (RESETz) signal goes high.

The Control Status Register (CONTROL_STATUS) is shown in Figure 1-117 and described in Table 1-165.

Figure 1-117. Control Status Register (CONTROL_STATUS)

31	20	19	18	17	16
Reserved		ADMUX		WAITEN	BW
R-0		R-L		R-L	R-L

15	5	4	0
Reserved		SYSBOOT	
R-0		R-L	

LEGEND: R = Read only; L = Latched pin value; -*n* = value after reset

Table 1-165. Control Status Register (CONTROL_STATUS) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reserved.
19-18	ADMUX		GPMC CS0 Default Address Muxing, from CS0MUX[0] and CS0MUX[1] pins.
		0	No Addr/Data Muxing
		1h	Addr/Data Muxing
		2h	Addr/Addr/Data Muxing
		3h	Reserved
17	WAITEN		GPMC CS0 Default Wait Enable, from CS0WAIT pin.
		0	Ignore WAIT input.
		1	Use WAIT input.
16	BW		GPMC CS0 Default Bus Width, from CS0BW pin.
		0	8-bit data bus.
		1	16-bit data bus.
15-5	Reserved	0	Reserved
4-0	SYSBOOT		System Boot Type, from BTMODE pins. These pins determine the primary bootmode.

1.16.1.1.2 Boot Status Register (BOOTSTAT)

The Boot Status (BOOTSTAT) register indicates the status of the device boot process.

The Boot Status Register (BOOTSTAT) is shown in Figure 1-118 and described in Table 1-166.

Figure 1-118. Boot Status Register (BOOTSTAT)

31	20	19	16	15	1	0
Reserved		BOOTERR		Reserved		BC
R-0		R/W-0		R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-166. Boot Status Register (BOOTSTAT) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reserved.
19-16	BOOTERR		Boot Error. The exact meaning of the various error codes will be determined by the bootloader software.
		0	No Boot Error
		Other	Bootloader detected boot error.
15-1	Reserved	0	Reserved.
0	BC		Boot Complete bit. This bit may be optionally set by a host boot device to indicate that it has finished loading code. The Host ARM can poll this bit to determine whether to continue the boot process
		0	Host has not complete boot sequence.
		1	Host boot sequence is complete.

1.16.1.1.3 DSP Boot Address Register (DSPBOOTADDR)

The DSP boot address defaults to 0800 0000h (C674x DSP L2 space). The Host ARM must place DSP boot code into C674x DSP L2 prior to releasing GEM reset or change the DSPBOOTADDR to point to the code location. The DSPBOOTADDR register is reset to its default value by a hard reset only and retains its current value over a soft reset.

The DSP Boot Address Register (DSPBOOTADDR) is shown in Figure 1-119 and described in Table 1-167.

Figure 1-119. DSP Boot Address Register (DSPBOOTADDR)

31	10	9	1	0
BOOTADDR		Reserved		RSTDONE
R/W-0		R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-167. DSP Boot Address Register (DSPBOOTADDR) Field Descriptions

Bit	Field	Value	Description
31-10	BOOTADDR	0-3F FFFFh	DSP Boot Address (upper 22 bits)
9-1	Reserved	0	Reserved.
0	RSTDONE	0-1	DSP Reset Done.

1.16.1.2 PLL Control Registers

Table 1-168 lists the registers of the PLL control registers. For the base address of these registers, see Table 1-12.

Table 1-168. PLL Control Registers

Offset	Acronym	Register Description	Section
400h	MAINPLL_CTRL	Main PLL Control Register	Section 1.16.1.2.1
404h	MAINPLL_PWD	Main PLL Powerdown Register	Section 1.16.1.2.2
408h	MAINPLL_FREQ1	Main PLL Frequency 1 Register	Section 1.16.1.2.3
40Ch	MAINPLL_DIV1	Main PLL Divider 1 Register	Section 1.16.1.2.4
410h	MAINPLL_FREQ2	Main PLL Frequency 2 Register	Section 1.16.1.2.5
414h	MAINPLL_DIV2	Main PLL Divider 2 Register	Section 1.16.1.2.6
2748h	MAINPLL_FREQ3	Main PLL Frequency 3 Register	Section 1.16.1.2.7
41Ch	MAINPLL_DIV3	Main PLL Divider 3 Register	Section 1.16.1.2.8
420h	MAINPLL_FREQ4	Main PLL Frequency 4 Register	Section 1.16.1.2.9
424h	MAINPLL_DIV4	Main PLL Divider 4 Register	Section 1.16.1.2.10
428h	MAINPLL_FREQ5	Main PLL Frequency 5 Register	Section 1.16.1.2.11
42Ch	MAINPLL_DIV5	Main PLL Divider 5 Register	Section 1.16.1.2.12
434h	MAINPLL_DIV6	Main PLL Divider 6 Register	Section 1.16.1.2.13
43Ch	MAINPLL_DIV7	Main PLL Divider 7 Register	Section 1.16.1.2.14
440h	DDRPLL_CTRL	DDR PLL Control Register	Section 1.16.1.2.15
444h	DDRPLL_PWD	DDR PLL Powerdown Register	Section 1.16.1.2.16
44Ch	DDRPLL_DIV1	DDR PLL Divider 1 Register	Section 1.16.1.2.17
450h	DDRPLL_FREQ2	DDR PLL Frequency 2 Register	Section 1.16.1.2.18
454h	DDRPLL_DIV2	DDR PLL Divider 2 Register	Section 1.16.1.2.19
458h	DDRPLL_FREQ3	DDR PLL Frequency 3 Register	Section 1.16.1.2.20
45Ch	DDRPLL_DIV3	DDR PLL Divider 3 Register	Section 1.16.1.2.21
460h	DDRPLL_FREQ4	DDR PLL Frequency 4 Register	Section 1.16.1.2.22
464h	DDRPLL_DIV4	DDR PLL Divider 4 Register	Section 1.16.1.2.23
468h	DDRPLL_FREQ5	DDR PLL Frequency 5 Register	Section 1.16.1.2.24
46Ch	DDRPLL_DIV5	DDR PLL Divider 5 Register	Section 1.16.1.2.25
470h	VIDEOPLL_CTRL	Video PLL Control Register	Section 1.16.1.2.26
474h	VIDEOPLL_PWD	Video PLL Powerdown Register	Section 1.16.1.2.27
478h	VIDEOPLL_FREQ1	Video PLL Frequency 1 Register	Section 1.16.1.2.28
47Ch	VIDEOPLL_DIV1	Video PLL Divider 1 Register	Section 1.16.1.2.29
480h	VIDEOPLL_FREQ2	Video PLL Frequency 2 Register	Section 1.16.1.2.30
484h	VIDEOPLL_DIV2	Video PLL Divider 2 Register	Section 1.16.1.2.31
488h	VIDEOPLL_FREQ3	Video PLL Frequency 3 Register	Section 1.16.1.2.32
48Ch	VIDEOPLL_DIV3	Video PLL Divider 3 Register	Section 1.16.1.2.33
4A0h	AUDIOPLL_CTRL	Audio PLL Control Register	Section 1.16.1.2.34
4A4h	AUDIOPLL_PWD	Audio PLL Powerdown Register	Section 1.16.1.2.35
4B0h	AUDIOPLL_FREQ2	Audio PLL Frequency 2 Register	Section 1.16.1.2.36
4B4h	AUDIOPLL_DIV2	Audio PLL Divider 2 Register	Section 1.16.1.2.37
4B8h	AUDIOPLL_FREQ3	Audio PLL Frequency 3 Register	Section 1.16.1.2.38
4BCh	AUDIOPLL_DIV3	Audio PLL Divider 3 Register	Section 1.16.1.2.39
4C0h	AUDIOPLL_FREQ4	Audio PLL Frequency 4 Register	Section 1.16.1.2.40
4C4h	AUDIOPLL_DIV4	Audio PLL Divider 4 Register	Section 1.16.1.2.41
4C8h	AUDIOPLL_FREQ5	Audio PLL Frequency 5 Register	Section 1.16.1.2.42
4CCh	AUDIOPLL_DIV5	Audio PLL 5 Divider Register	Section 1.16.1.2.43

1.16.1.2.1 Main PLL Control Register (MAINPLL_CTRL)

The MAINPLL_CTRL register is used to enable and control the base frequency of the main PLL. The default values set the frequency at 1728 MHz.

The Main PLL Control Register (MAINPLL_CTRL) is shown in Figure 1-120 and described in Table 1-169.

Figure 1-120. Main PLL Control Register (MAINPLL_CTRL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAIN_N															
R/W-40h															

15	14	13	12	11	10	9	8
MAIN_P							
R/W-1							

7	6	5	4	3	2	1	0
MAIN_LOCK	Reserved			MAIN_PLLEN	MAIN_BP	Reserved	MAIN_LOC_CTL
R-0	R-0			R/W-1	R/W-1	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-169. Main PLL Control Register (MAINPLL_CTRL) Field Descriptions

Bit	Field	Value	Description
31-16	MAIN_N	0-FFFFh	Main PLL N multiplier value.
15-8	MAIN_P	0-FFh	Main PLL P divider value.
7	MAIN_LOCK	0-1	Main PLL lock status bit.
6-4	Reserved	0	Reserved. Reads returns 0.
3	MAIN_PLLEN		Main PLL enable.
		0	PLL is disabled (power down state).
		1	PLL is enabled (normal operation).
2	MAIN_BP		Main PLL bypass enable.
		0	Normal operation.
		1	PLL in bypass mode, reference clock driven at output.
1	Reserved	0	Reserved. Reads returns 0.
0	MAIN_LOC_CTL		Select the source to detect PLL lock. Value of 0 is recommended to use the more reliable Analog circuit.
		0	Analog lock
		1	Digital lock

1.16.1.2.2 Main PLL Powerdown Register (MAINPLL_PWD)

The MAINPLL_PWD register is used to power down the individual output clocks of the main PLL.

The Main PLL Powerdown Register (MAINPLL_PWD) is shown in Figure 1-121 and described in Table 1-170.

Figure 1-121. Main PLL Powerdown Register (MAINPLL_PWD)

31							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
PWD_CLK7	PWD_CLK6	PWD_CLK5	PWD_CLK4	PWD_CLK3	PWD_CLK2	PWD_CLK1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-170. Main PLL Powerdown Register (MAINPLL_PWD) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Reads returns 0.
7	PWD_CLK7	1-0	Main PLL Clock7 Powerdown. Setting this bit powers down clock 7 (Audio PLL Reference clock).
6	PWD_CLK6	1-0	Main PLL Clock6 Powerdown. Setting this bit powers down clock 6 (USB Reference clock).
5	PWD_CLK5	1-0	Main PLL Clock5 Powerdown. Setting this bit powers down clock 5 (SYSCLK24).
4	PWD_CLK4	1-0	Main PLL Clock4 Powerdown. Setting this bit powers down clock 4 (SYSCLK4).
3	PWD_CLK3	1-0	Main PLL Clock3 Powerdown. Setting this bit powers down clock 3 (SYSCLK3).
2	PWD_CLK2	1-0	Main PLL Clock2 Powerdown. Setting this bit powers down clock 2 (SYSCLK2).
1	PWD_CLK1	1-0	Main PLL Clock1 Powerdown. Setting this bit powers down clock 1 (SYSCLK1).
0	Reserved	0	Reserved. Reads returns 0.

1.16.1.2.3 Main PLL Frequency 1 Register (MAINPLL_FREQ1)

The MAINPLL_FREQ1 register is used to control the Main PLL Clock 1 pre-divider frequency of the SYSCLK1 (C674x DSP) clock. The default FREQ1 value is 8.5 to generate an 800 MHz (813.1765 MHz) SYSCLK1.

The Main PLL Frequency 1 Register (MAINPLL_FREQ1) is shown in Figure 1-122 and described in Table 1-171.

Figure 1-122. Main PLL Frequency 1 Register (MAINPLL_FREQ1)

31	30	29	28	27	24	23	0
MAIN_LDFREQ1	Reserved		MAIN_TRUNC1	MAIN_INTFREQ1		MAIN_FRACFREQ1	
R/W-1	R-0		R/W-0	R/W-8h		R/W-800000h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-171. Main PLL Frequency 1 Register (MAINPLL_FREQ1) Field Descriptions

Bit	Field	Value	Description
31	MAIN_LDFREQ1	1-0	Load Synth1 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into Main Synthesizer1.
30-29	Reserved	0	Reserved. Reads returns 0.
28	MAIN_TRUNC1	1-0	Synth1 Enable Truncate Correction.
27-24	MAIN_INTFREQ1	8h-Fh	Synth1 Frequency integer divider.
23-0	MAIN_FRACFREQ1	0-FF FFFFh	Synth1 Frequency fractional divider.

1.16.1.2.4 Main PLL Divider 1 Register (MAINPLL_DIV1)

The MAINPLL_DIV1 register is used to control the Main PLL Clock 1 post-divider frequency of the SYSCLK1 (GEM is changed to C674x DSP) clock. The default DIV1 value is 2 to generate an 800 MHz SYSCLK1.

The Main PLL Divider 1 Register (MAINPLL_DIV1) is shown in Figure 1-123 and described in Table 1-172.

Figure 1-123. Main PLL Divider 1 Register (MAINPLL_DIV1)

31	9	8	7	0
Reserved		MAIN_LDMDIV1	MAIN_MDIV1	
R-0		R/W-1	R/W-2h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-172. Main PLL Divider 1 Register (MAINPLL_DIV1) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	MAIN_LDMDIV1	1-0	Load Synth1 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into Main Synthesizer1.
7-0	MAIN_MDIV1	0-FFh	Synth1 Frequency M Post Divider.

1.16.1.2.5 Main PLL Frequency 2 Register (MAINPLL_FREQ2)

The MAINPLL_FREQ2 register is used to control the Main PLL Clock 2 pre-divider frequency of the SYSCLK2 (Cortex-A8) clock. The default FREQ2 value is 14 to generate a 1GHz (987.42) clock.

The Main PLL Frequency 2 Register (MAINPLL_FREQ2) is shown in Figure 1-124 and described in Table 1-173.

Figure 1-124. Main PLL Frequency 2 Register (MAINPLL_FREQ2)

31	30 29	28	27 24	23 0
MAIN_LDFREQ2	Reserved	MAIN_TRUNC2	MAIN_INTFREQ2	MAIN_FRACFREQ2
R/W-1	R-0	R/W-0	R/W-Eh	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-173. Main PLL Frequency 2 Register (MAINPLL_FREQ2) Field Descriptions

Bit	Field	Value	Description
31	MAIN_LDFREQ2	1-0	Load Synth2 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into Main Synthesizer2.
30-29	Reserved	0	Reserved. Reads returns 0.
28	MAIN_TRUNC2	1-0	Synth2 Enable Truncate Correction.
27-24	MAIN_INTFREQ2	8h-Fh	Synth2 Frequency integer divider.
23-0	MAIN_FRACFREQ2	0-FF FFFFh	Synth2 Frequency fractional divider.

1.16.1.2.6 Main PLL Divider 2 Register (MAINPLL_DIV2)

The MAINPLL_DIV2 register is used to control the Main PLL Clock 2 post-divider frequency of the SYSCLK2 (Cortex-A8) clock. The default DIV2 value is 1 for a ~1 GHz SYSCLK2.

The Main PLL Divider 2 Register (MAINPLL_DIV2) is shown in Figure 1-125 and described in Table 1-174.

Figure 1-125. Main PLL Divider 2 Register (MAINPLL_DIV2)

31 9	8	7 0
Reserved	MAIN_LDMDIV2	MAIN_MDIV2
R-0	R/W-1	R/W-1h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-174. Main PLL Divider 2 Register (MAINPLL_DIV2) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	MAIN_LDMDIV2	1-0	Load Synth2 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into Main Synthesizer2.
7-0	MAIN_MDIV2	0-FFh	Synth2 Frequency M Post Divider.

1.16.1.2.7 Main PLL Frequency 3 Register (MAINPLL_FREQ3)

The MAINPLL_FREQ3 register is used to control the Main PLL Clock 3 pre-divider frequency of the SYSCLK3 (IVA HD is changed to HDVICP2) clock. The default FREQ3 value is 8.666667 to generate a 533 MHz SYSCLK3.

The Main PLL Frequency 3 Register (MAINPLL_FREQ3) is shown in Figure 1-126 and described in Table 1-175.

Figure 1-126. Main PLL Frequency 3 Register (MAINPLL_FREQ3)

31	30 29	28	27 24	23 0
MAIN_LDFREQ3	Reserved	MAIN_TRUNC3	MAIN_INTFREQ3	MAIN_FRACFREQ3
R/W-1	R-0	R/W-1	R/W-8h	R/W-AAAAB0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-175. Main PLL Frequency 3 Register (MAINPLL_FREQ3) Field Descriptions

Bit	Field	Value	Description
31	MAIN_LDFREQ3	1-0	Load Synth3 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into Main Synthesizer3.
30-29	Reserved	0	Reserved. Reads returns 0.
28	MAIN_TRUNC3	1-0	Synth3 Enable Truncate Correction.
27-24	MAIN_INTFREQ3	0-Fh	Synth3 Frequency integer divider.
23-0	MAIN_FRACFREQ3	0-FF FFFFh	Synth3 Frequency fractional divider.

1.16.1.2.8 Main PLL Divider 3 Register (MAINPLL_DIV3)

The MAINPLL_DIV3 register is used to control the Main PLL Clock 3 post-divider frequency of the SYSCLK3 (IVA HD is changed to HDVICP2) clock. The default DIV3 value is 3 to generate a 533 MHz SYSCLK3.

The Main PLL Divider 3 Register (MAINPLL_DIV3) is shown in Figure 1-127 and described in Table 1-176.

Figure 1-127. Main PLL Divider 3 Register (MAINPLL_DIV3)

31 9	8	7 0
Reserved	MAIN_LDMDIV3	MAIN_MDIV3
R-0	R/W-1	R/W-3h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-176. Main PLL Divider 3 Register (MAINPLL_DIV3) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	MAIN_LDMDIV3	1-0	Load Synth3 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into Main Synthesizer3. Bit is cleared by hardware after load is complete.
7-0	MAIN_MDIV3	0-FFh	Synth3 Frequency M Post Divider.

1.16.1.2.9 Main PLL Frequency 4 Register (MAINPLL_FREQ4)

The MAINPLL_FREQ4 register is used to control the Main PLL Clock 4 pre-divider frequency of the SYSCLK4, SYSCLK5, SYSCLK6, and SYSCLK7 clocks. The default FREQ4 value is 9.333333 to generate a 500 MHz SYSCLK4.

The Main PLL Frequency 4 Register (MAINPLL_FREQ4) is shown in Figure 1-128 and described in Table 1-177.

Figure 1-128. Main PLL Frequency 4 Register (MAINPLL_FREQ4)

31	30 29	28	27 24	23 0
MAIN_LDFREQ4	Reserved	MAIN_TRUNC4	MAIN_INTFREQ4	MAIN_FRACFREQ4
R/W-1	R-0	R/W-1	R/W-9h	R/W-55554Fh

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-177. Main PLL Frequency 4 Register (MAINPLL_FREQ4) Field Descriptions

Bit	Field	Value	Description
31	MAIN_LDFREQ4	1-0	Load Synth4 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into Main Synthesizer4.
30-29	Reserved	0	Reserved. Reads returns 0.
28	MAIN_TRUNC4	1-0	Synth4 Enable Truncate Correction.
27-24	MAIN_INTFREQ4	8h-Fh	Synth4 Frequency integer divider.
23-0	MAIN_FRACFREQ4	0-FF FFFFh	Synth4 Frequency fractional divider.

1.16.1.2.10 Main PLL Divider 4 Register (MAINPLL_DIV4)

The MAINPLL_DIV4 register is used to control the Main PLL Clock 4 post-divider frequency of the SYSCLK4, SYSCLK5, SYSCLK6, and SYSCLK7 clocks. The default DIV4 value is 3 to generate a 500 MHz SYSCLK4.

The Main PLL Divider 4 Register (MAINPLL_DIV4) is shown in Figure 1-129 and described in Table 1-178.

Figure 1-129. Main PLL Divider 4 Register (MAINPLL_DIV4)

31 9	8	7 0
Reserved	MAIN_LDMDIV4	MAIN_MDIV4
R-0	R/W-1	R/W-3h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-178. Main PLL Divider 4 Register (MAINPLL_DIV4) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	MAIN_LDMDIV4	1-0	Load Synth4 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into Main Synthesizer4.
7-0	MAIN_MDIV4	0-FFh	Synth4 Frequency M Post Divider.

1.16.1.2.11 Main PLL Frequency 5 Register (MAINPLL_FREQ5)

The MAINPLL_FREQ5 register is used to control the Main PLL Clock 5 pre-divider frequency of the CPGMAC rft_clk (SYSCLK24) clock. The default FREQ5 value is 9.216 to generate a 125 MHz SYSCLK24.

The Main PLL Frequency 5 Register (MAINPLL_FREQ5) is shown in Figure 1-130 and described in Table 1-179.

Figure 1-130. Main PLL Frequency 5 Register (MAINPLL_FREQ5)

31	30 29	28	27 24	23 0
MAIN_LDFREQ5	Reserved	MAIN_TRUNC5	MAIN_INTFREQ5	MAIN_FRACFREQ5
R/W-1	R-0	R/W-0	R/W-9h	R/W-374BC6h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-179. Main PLL Frequency 5 Register (MAINPLL_FREQ5) Field Descriptions

Bit	Field	Value	Description
31	MAIN_LDFREQ5	1-0	Load Synth5 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into Main Synthesizer5.
30-29	Reserved	0	Reserved. Reads returns 0.
28	MAIN_TRUNC5	1-0	Synth5 Enable Truncate Correction.
27-24	MAIN_INTFREQ5	8h-Fh	Synth5 Frequency integer divider.
23-0	MAIN_FRACFREQ5	0-FF FFFFh	Synth5 Frequency fractional divider.

1.16.1.2.12 Main PLL Divider 5 Register (MAINPLL_DIV5)

The MAINPLL_DIV5 register is used to control the Main PLL Clock 5 post-divider frequency of the CPGMAC rft_clk clock. The default DIV5 value is 12.

The Main PLL Divider 5 Register (MAINPLL_DIV5) is shown in Figure 1-131 and described in Table 1-180.

Figure 1-131. Main PLL Divider 5 Register (MAINPLL_DIV5)

31 9	8	7 0
Reserved	MAIN_LDMDIV5	MAIN_MDIV5
R-0	R/W-1	R/W-Ch

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-180. Main PLL Divider 5 Register (MAINPLL_DIV5) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	MAIN_LDMDIV5	1-0	Load Synth5 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into Main Synthesizer5.
7-0	MAIN_MDIV5	0-FFh	Synth5 Frequency M Post Divider.

1.16.1.2.13 Main PLL Divider 6 Register (MAINPLL_DIV6)

The MAINPLL_DIV6 register is used to control the Main PLL Clock 6 post-divider frequency of the USB Reference clock. The default DIV6 value is 72 to select 24 MHz.

The Main PLL Divider 6 Register (MAINPLL_DIV6) is shown in Figure 1-132 and described in Table 1-181.

Figure 1-132. Main PLL Divider 6 Register (MAINPLL_DIV6)

31	9	8	7	0
Reserved		MAIN_LDMDIV6	MAIN_MDIV6	
R-0		R/W-1	R/W-48h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-181. Main PLL Divider 6 Register (MAINPLL_DIV6) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	MAIN_LDMDIV6	1-0	Load M Divider6 value. Setting this bit to 1 causes the M Divider6 value to be loaded.
7-0	MAIN_MDIV6	0-FFh	Frequency M Post Divider6.

1.16.1.2.14 Main PLL Divider 7 Register (MAINPLL_DIV7)

The MAINPLL_DIV7 register is used to control the Main PLL Clock 7 post-divider frequency of the Audio PLL reference clock. The default DIV7 value is 4 to select 432 MHz.

The Main PLL Divider 7 Register (MAINPLL_DIV7) is shown in Figure 1-133 and described in Table 1-182.

Figure 1-133. Main PLL Divider 7 Register (MAINPLL_DIV7)

31	9	8	7	0
Reserved		MAIN_LDMDIV7	MAIN_MDIV7	
R-0		R/W-1	R/W-4h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-182. Main PLL Divider 7 Register (MAINPLL_DIV7) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	MAIN_LDMDIV7	1-0	Load M Divider7 value. Setting this bit to 1 causes the M Divider7 value to be loaded.
7-0	MAIN_MDIV7	0-FFh	Frequency M Post Divider7.

1.16.1.2.15 DDR PLL Control Register (DDRPLL_CTRL)

The DDRPLL_CTRL register is used to control the base frequency of the DDR PLL. The default values set the base frequency at 800 MHz (796.5 MHz).

The DDR PLL Control Register (DDRPLL_CTRL) is shown in Figure 1-134 and described in Table 1-183.

Figure 1-134. DDR PLL Control Register (DDRPLL_CTRL)

31															16
DDR_N															
R/W-3Bh															

15							8
DDR_P							
R/W-1							

7	6	4	3	2	1	0
DDR_LOCK	Reserved		DDR_PLLEN	DDR_BP	Reserved	DDR_LOC_CTL
R-0	R-0		R/W-1	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-183. DDR PLL Control Register (DDRPLL_CTRL) Field Descriptions

Bit	Field	Value	Description
31-16	DDR_N	0-FFFFh	DDR PLL N multiplier value.
15-8	DDR_P	0-FFh	DDR PLL P divider value.
7	DDR_LOCK	1-0	DDR PLL lock status bit.
6-4	Reserved	0	Reserved. Reads returns 0.
3	DDR_PLLEN		DDR PLL enable.
		0	PLL is disabled (power down state).
		1	PLL is enabled (normal operation).
2	DDR_BP		DDR PLL bypass enable.
		0	PLL in bypass mode, reference clock driven at output.
		1	Normal operation.
1	Reserved	0	Reserved. Reads returns 0.
0	DDR_LOC_CTL		DDR PLL lock output select.
		0	Analog lock
		1	Digital lock

1.16.1.2.16 DDR PLL Powerdown Register (DDRPLL_PWD)

The DDRPLL_PWD register is used to powerdown the individual output clocks of the DDR PLL.

The DDR PLL Powerdown Register (DDRPLL_PWD) is shown in Figure 1-135 and described in Table 1-184.

Figure 1-135. DDR PLL Powerdown Register (DDRPLL_PWD)

31	16
Reserved	
R-0	

15	6	5	4	3	2	1	0
Reserved		PWD_CLK5	PWD_CLK4	PWD_CLK3	PWD_CLK2	PWD_CLK1	Reserved
R-0		R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-184. DDR PLL Powerdown Register (DDRPLL_PWD) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved. Reads returns 0.
5	PWD_CLK5	1-0	DDR PLL Clock5 Powerdown. Setting this bit powers down clock 5 (spare).
4	PWD_CLK4	1-0	DDR PLL Clock4 Powerdown. Setting this bit powers down clock 4 (spare).
3	PWD_CLK3	1-0	DDR PLL Clock3 Powerdown. Setting this bit powers down clock 3 (SYSCLK8)
2	PWD_CLK2	1-0	DDR PLL Clock2 Powerdown. Setting this bit powers down clock 2 (SYSCLK9/10).
1	PWD_CLK1	1-0	DDR PLL Clock1 Powerdown. Setting this bit powers down clock 1 (DDR Clock).
0	Reserved	0	Reserved. Reads returns 0.

1.16.1.2.17 DDR PLL Divider 1 Register (DDRPLL_DIV1)

The DDRPLL_DIV1 register is used to control the DDR PLL Clock 1 post-divider frequency of the DDR clock. The default DIV1 value is 1.

The DDR PLL Divider 1 Register (DDRPLL_DIV1) is shown in Figure 1-136 and described in Table 1-185.

Figure 1-136. DDR PLL Divider 1 Register (DDRPLL_DIV1)

31	9	8	7	0
Reserved		DDR_LDMDIV1	DDR_MDIV1	
R-0		R/W-1	R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-185. DDR PLL Divider 1 Register (DDRPLL_DIV1) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	DDR_LDMDIV1	1-0	Load Synth1 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into DDR Synthesizer1.
7-0	DDR_MDIV1	0-FFh	Synth1 Frequency M Post Divider

1.16.1.2.18 DDR PLL Frequency 2 Register (DDRPLL_FREQ2)

The DDRPLL_FREQ2 register is used to control the DDR PLL Clock 2 pre-divider frequency of the SYSCLK9 (16 MHz - VTP) and SYSCLK10 (48 MHz - UART, SPI, CEC, etc.) clock. The default FREQ2 value is 8.85.

The DDR PLL Frequency 2 Register (DDRPLL_FREQ2) is shown in Figure 1-137 and described in Table 1-186.

Figure 1-137. DDR PLL Frequency 2 Register (DDRPLL_FREQ2)

31	30	29	28	27	24	23	0
DDR_LDFREQ2	Reserved		DDR_TRUNC2	DDR_INTFREQ2		DDR_FRACFREQ2	
R/W-1	R-0		R/W-0	R/W-8h		R/W-D99999h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-186. DDR PLL Frequency 2 Register (DDRPLL_FREQ2) Field Descriptions

Bit	Field	Value	Description
31	DDR_LDFREQ2	1-0	Load Synth2 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into DDR Synthesizer2.
30-29	Reserved	0	Reserved. Reads returns 0.
28	DDR_TRUNC2	1-0	Synth2 Enable Truncate Correction
27-24	DDR_INTFREQ2	8h-Fh	Synth2 Frequency integer divider.
23-0	DDR_FRACFREQ2	0-FF FFFFh	Synth2 Frequency fractional divider.

1.16.1.2.19 DDR PLL Divider 2 Register (DDRPLL_DIV2)

The DDRPLL_DIV2 register is used to control the DDR PLL Clock 2 post-divider frequency of the SYSCLK9 (VTP), and SYSCLK10 (UART, SPI, CEC, etc.) clocks. The default DIV2 value is 30.

The DDR PLL Divider 2 Register (DDRPLL_DIV2) is shown in Figure 1-138 and described in Table 1-187.

Figure 1-138. DDR PLL Divider 2 Register (DDRPLL_DIV2)

31	9	8	7	0
Reserved		DDR_LDMDIV2	DDR_MDIV2	
R-0		R/W-1	R/W-1Eh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-187. DDR PLL Divider 2 Register (DDRPLL_DIV2) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	DDR_LDMDIV2	1-0	Load Synth2 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into DDR Synthesizer2.
7-0	DDR_MDIV2	0-FFh	Synth2 Frequency M Post Divider.

1.16.1.2.20 DDR PLL Frequency 3 Register (DDRPLL_FREQ3)

The DDRPLL_FREQ3 register is used to control the DDR PLL Clock 3 pre-divider frequency of SYSCLK8 (364 MHz DMM, EMIF clock; 400 MHz DDR2/3 clock).

The DDR PLL Frequency 3 Register (DDRPLL_FREQ3) is shown in Figure 1-139 and described in Table 1-188.

Figure 1-139. DDR PLL Frequency 3 Register (DDRPLL_FREQ3)

31	30	29	28	27	24	23	0
DDR_LDFREQ3	Reserved		DDR_TRUNC3	DDR_INTFREQ3		DDR_FRACFREQ3	
R/W-1	R-0		R/W-0	R/W-8h		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-188. DDR PLL Frequency 3 Register (DDRPLL_FREQ3) Field Descriptions

Bit	Field	Value	Description
31	DDR_LDFREQ3	1-0	Load Synth3 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into DDR Synthesizer3.
30-29	Reserved	0	Reserved. Reads returns 0.
28	DDR_TRUNC3	1-0	Synth3 Enable Truncate Correction
27-24	DDR_INTFREQ3	8h-Fh	Synth3 Frequency integer divider.
23-0	DDR_FRACFREQ3	0-FF FFFFh	Synth3 Frequency fractional divider.

1.16.1.2.21 DDR PLL Divider 3 Register (DDRPLL_DIV3)

The DDRPLL_DIV3 register is used to control the DDR PLL Clock 3 (SYSCLK8) post-divider. The default DIV3 value is 4.

The DDR PLL Divider 3 Register (DDRPLL_DIV3) is shown in Figure 1-140 and described in Table 1-189.

Figure 1-140. DDR PLL Divider 3 Register (DDRPLL_DIV3)

31	9	8	7	0
Reserved		DDR_LDMDIV3	DDR_MDIV3	
R-0		R/W-1	R/W-4h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-189. DDR PLL Divider 3 Register (DDRPLL_DIV3) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	DDR_LDMDIV3	1-0	Load Synth3 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into DDR Synthesizer3. Bit is cleared by hardware after load is complete
7-0	DDR_MDIV3	0-FFh	Synth3 Frequency M Post Divider

1.16.1.2.22 DDR PLL Frequency 4 Register (DDRPLL_FREQ4)

The DDRPLL_FREQ4 register is used to control the expansion DDR PLL Clock 4 pre-divider frequency. The default FREQ4 value is 14.

The DDR PLL Frequency 4 Register (DDRPLL_FREQ4) is shown in Figure 1-141 and described in Table 1-190.

Figure 1-141. DDR PLL Frequency 4 Register (DDRPLL_FREQ4)

31	30	29	28	27	24	23	0
DDR_LDFREQ4	Reserved		DDR_TRUNC4	DDR_INTFREQ4		DDR_FRACFREQ4	
R/W-1	R-0		R/W-0	R/W-Eh		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-190. DDR PLL Frequency 4 Register (DDRPLL_FREQ4) Field Descriptions

Bit	Field	Value	Description
31	DDR_LDFREQ4	1-0	Load Synth4 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into DDR Synthesizer4.
30-29	Reserved	0	Reserved. Reads returns 0.
28	DDR_TRUNC4	1-0	Synth4 Enable Truncate Correction.
27-24	DDR_INTFREQ4	8h-Fh	Synth4 Frequency integer divider.
23-0	DDR_FRACFREQ4	0-FF FFFFh	Synth4 Frequency fractional divider.

1.16.1.2.23 DDR PLL Divider 4 Register (DDRPLL_DIV4)

The DDRPLL_DIV4 register is used to control the expansion DDR PLL Clock 4 post-divider. The default DIV4 value is 4.

The DDR PLL Divider 4 Register (DDRPLL_DIV4) is shown in Figure 1-142 and described in Table 1-191.

Figure 1-142. DDR PLL Divider 4 Register (DDRPLL_DIV4)

31	9	8	7	0
Reserved		DDR_LDMDIV4	DDR_MDIV4	
R-0		R/W-1	R/W-4h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-191. DDR PLL Divider 4 Register (DDRPLL_DIV4) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	DDR_LDMDIV4	1-0	Load Synth4 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into DDR Synthesizer4.
7-0	DDR_MDIV4	0-FFh	Synth4 Frequency M Post Divider.

1.16.1.2.24 DDR PLL Frequency 5 Register (DDRPLL_FREQ5)

The DDRPLL_FREQ5 register is used to control the expansion DDR PLL Clock 5 pre-divider frequency. The default FREQ5 value is 14.

The DDR PLL Frequency 5 Register (DDRPLL_FREQ5) is shown in Figure 1-143 and described in Table 1-192.

Figure 1-143. DDR PLL Frequency 5 Register (DDRPLL_FREQ5)

31	30	29	28	27	24	23	0
DDR_LDFREQ5	Reserved		DDR_TRUNC5	DDR_INTFREQ5		DDR_FRACFREQ5	
R/W-1	R-0		R/W-0	R/W-Eh		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-192. DDR PLL Frequency 5 Register (DDRPLL_FREQ5) Field Descriptions

Bit	Field	Value	Description
31	DDR_LDFREQ5	1-0	Load Synth5 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into DDR Synthesizer5.
30-29	Reserved	0	Reserved. Reads returns 0.
28	DDR_TRUNC5	1-0	Synth5 Enable Truncate Correction.
27-24	DDR_INTFREQ5	8h-Fh	Synth5 Frequency integer divider.
23-0	DDR_FRACFREQ5	0-FF FFFFh	Synth5 Frequency fractional divider.

1.16.1.2.25 DDR PLL Divider 5 Register (DDRPLL_DIV5)

The DDRPLL_DIV5 register is used to control the expansion DDR PLL Clock 5 post-divider. The default DIV5 value is 4.

The DDR PLL Divider 5 Register (DDRPLL_DIV5) is shown in Figure 1-144 and described in Table 1-193.

Figure 1-144. DDR PLL Divider 5 Register (DDRPLL_DIV5)

31	9	8	7	0
Reserved		DDR_LDMDIV5	DDR_MDIV5	
R-0		R/W-1	R/W-4h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-193. DDR PLL Divider 5 Register (DDRPLL_DIV5) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	DDR_LDMDIV5	1-0	Load Synth5 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into DDR Synthesizer5.
7-0	DDR_MDIV5	0-FFh	Synth5 Frequency M Post Divider.

1.16.1.2.26 Video PLL Control Register (VIDEOPLL_CTRL)

The VIDEOPLL_CTRL register is used to control the base frequency of the Video PLL. The default values set the frequency at 1485 MHz.

The Video PLL Control Register (VIDEOPLL_CTRL) is shown in Figure 1-145 and described in Table 1-194.

Figure 1-145. Video PLL Control Register (VIDEOPLL_CTRL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VIDEO_N															
R/W-6Eh															

15	14	13	12	11	10	9	8
VIDEO_P							
R/W-2h							

7	6	5	4	3	2	1	0
VIDEO_LOCK	Reserved			VIDEO_PLLEN	VIDEO_BP	Reserved	VIDEO_LOC_CTL
R-0	R-0			R/W-1	R/W-1	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-194. Video PLL Control Register (VIDEOPLL_CTRL) Field Descriptions

Bit	Field	Value	Description
31-16	VIDEO_N	0-FFFFh	VIDEO PLL N multiplier value.
15-8	VIDEO_P	0-FFh	VIDEO PLL P divider value.
7	VIDEO_LOCK	1-0	VIDEO PLL lock status bit.
6-4	Reserved	0	Reserved. Reads returns 0.
3	VIDEO_PLLEN		VIDEO PLL enable.
		0	PLL is disabled (power down state).
		1	PLL is enabled (normal operation).
2	VIDEO_BP		VIDEO PLL bypass enable.
		0	Normal operation.
		1	PLL in bypass mode, reference clock driven at output.
1	Reserved	0	Reserved. Reads returns 0.
0	VIDEO_LOC_CTL		VIDEO PLL lock output select.
		0	Analog lock
		1	Digital lock

1.16.1.2.27 Video PLL Powerdown Register (VIDEOPLL_PWD)

The VIDEOPLL_PWD register is used to powerdown the individual output clocks of the Video PLL.

The Video PLL Powerdown Register (VIDEOPLL_PWD) is shown in Figure 1-146 and described in Table 1-195.

Figure 1-146. Video PLL Powerdown Register (VIDEOPLL_PWD)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		PWD_CLK3	PWD_CLK2	PWD_CLK1	Reserved
R-0		R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-195. Video PLL Powerdown Register (VIDEOPLL_PWD) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. Reads returns 0.
3	PWD_CLK3	1-0	Video PLL Clock3 Powerdown. Setting this bit powers down clock 3 (Source of SYSCLK15 and STC0 source2).
2	PWD_CLK2	1-0	Video PLL Clock2 Powerdown. Setting this bit powers down clock 2 (Source of SYSCLK13, STC0 source0 and STC1 source1).
1	PWD_CLK1	1-0	Video PLL Clock1 Powerdown. Setting this bit powers down clock 1 (Source of SYSCLK17 and STC1 source0).
0	Reserved	0	Reserved. Reads returns 0.

1.16.1.2.28 Video PLL Frequency 1 Register (VIDEOPLL_FREQ1)

The VIDEOPLL_FREQ1 register is used to control the Video PLL Clock 1 pre-divider frequency of SYSCLK17 (SD_VENC) and STC1 source clocks. The default FREQ1 value is 11.

Video PLL Frequency 1 Register (VIDEOPLL_FREQ1) is shown in Figure 1-147 and described in Table 1-196.

Figure 1-147. Video PLL Frequency 1 Register (VIDEOPLL_FREQ1)

31	30 29	28	27 24	23 0
VID_LDFREQ1	Reserved	VID_TRUNC1	VID_INTFREQ1	VID_FRACFREQ1
R/W-1	R-0	R/W-0	R/W-Bh	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-196. Video PLL Frequency 1 Register (VIDEOPLL_FREQ1) Field Descriptions

Bit	Field	Value	Description
31	VID_LDFREQ1	1-0	Load Synth1 FREQ value Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into VIDEO Synthesizer1.
30-29	Reserved	0	Reserved. Reads returns 0.
28	VID_TRUNC1	1-0	Synth1 Enable Truncate Correction.
27-24	VID_INTFREQ1	8h-Fh	Synth1 Frequency integer divider.
23-0	VID_FRACFREQ1	0-FF FFFFh	Synth1 Frequency fractional divider.

1.16.1.2.29 Video PLL Divider 1 Register (VIDEOPLL_DIV1)

The VIDEOPLL_DIV1 register is used to control the VIDEO PLL Clock 1 post-divider frequency of the SYSCLK17 and STC0 source clocks. The default DIV1 value is 5.

The Video PLL Divider 1 Register (VIDEOPLL_DIV1) is shown in Figure 1-148 and described in Table 1-197.

Figure 1-148. Video PLL Divider 1 Register (VIDEOPLL_DIV1)

31 9	8	7 0
Reserved	VID_LDMDIV1	VID_MDIV1
R-0	R/W-1	R/W-5h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-197. Video PLL Divider 1 Register (VIDEOPLL_DIV1) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	VID_LDMDIV1	1-0	Load Synth1 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into VIDEO Synthesizer1.
7-0	VID_MDIV1	0-FFh	Synth1 Frequency M Post Divider.

1.16.1.2.30 Video PLL Frequency 2 Register (VIDEOPLL_FREQ2)

The VIDEOPLL_FREQ2 register is used to control the Video PLL Clock 2 pre-divider frequency of the SYSCLK13 (HD_VENC_D clock) and STC0/STC1 source clocks. If the Video PLL Clock 2 is locked to a transport stream System Time Clock (STC), the FREQ2 value will be modified by software to push or pull the clock frequency as part of the clock recovery process. The default FREQ2 value is 10.0 to generate a 74.25 MHz SYSCLK13.

The Video PLL Frequency 2 Register (VIDEOPLL_FREQ2) is shown in Figure 1-149 and described in Table 1-198.

Figure 1-149. Video PLL Frequency 2 Register (VIDEOPLL_FREQ2)

31	30 29	28	27 24	23 0
VID_LDFREQ2	Reserved	VID_TRUNC2	VID_INTFREQ2	VID_FRACFREQ2
R/W-1	R-0	R/W-0	R/W-Ah	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-198. Video PLL Frequency 2 Register (VIDEOPLL_FREQ2) Field Descriptions

Bit	Field	Value	Description
31	VID_LDFREQ2	1-0	Load Synth2 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into VIDEO Synthesizer2.
30-29	Reserved	0	Reserved. Reads returns 0.
28	VID_TRUNC2	1-0	Synth2 Enable Truncate Correction.
27-24	VID_INTFREQ2	8h-Fh	Synt25 Frequency integer divider.
23-0	VID_FRACFREQ2	0-FF FFFFh	Synth2 Frequency fractional divider.

1.16.1.2.31 Video PLL Divider 2 Register (VIDEOPLL_DIV2)

The VIDEOPLL_DIV2 register is used to control the VIDEO PLL Clock 2 post-divider frequency of the SYSCLK13 and STC0/STC1 source clocks. The default DIV2 value is 2 to generate a 74.25 MHz.

The Video PLL Divider 2 Register (VIDEOPLL_DIV2) is shown in Figure 1-150 and described in Table 1-199.

Figure 1-150. Video PLL Divider 2 Register (VIDEOPLL_DIV2)

31 9	8	7 0
Reserved	VID_LDMDIV2	VID_MDIV2
R-0	R/W-1	R/W-2h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-199. Video PLL Divider 2 Register (VIDEOPLL_DIV2) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	VID_LDMDIV2	1-0	Load Synth2 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into VIDEO Synthesizer2.
7-0	VID_MDIV2	0-FFh	Synth2 Frequency M Post Divider.

1.16.1.2.32 Video PLL Frequency 3 Register (VIDEOPLL_FREQ3)

The VIDEOPLL_FREQ3 register is used to control the Video PLL Clock 3 pre-divider frequency of the SYSCLK15 (HD_VENC_A clock) and STC0 source clock. If the Video PLL Clock 3 is locked to a transport stream System Time Clock (STC), the FREQ3 value will be modified by software to push or pull the clock frequency as part of the clock recovery process. The default FREQ3 value is 10.0 to generate a 74.25 MHz SYSCLK15.

The Video PLL Frequency 3 Register (VIDEOPLL_FREQ3) is shown in Figure 1-151 and described in Table 1-200.

Figure 1-151. Video PLL Frequency 3 Register (VIDEOPLL_FREQ3)

31	30 29	28	27 24	23 0
VID_LDFREQ3	Reserved	VID_TRUNC3	VID_INTFREQ3	VID_FRACFREQ3
R/W-1	R-0	R/W-0	R/W-Ah	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-200. Video PLL Frequency 3 Register (VIDEOPLL_FREQ3) Field Descriptions

Bit	Field	Value	Description
31	VID_LDFREQ3	1-0	Load Synth3 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into VIDEO Synthesizer3.
30-29	Reserved	0	Reserved. Reads returns 0.
28	VID_TRUNC3	1-0	Synth3 Enable Truncate Correction.
27-24	VID_INTFREQ3	8h-Fh	Synth3 Frequency integer divider.
23-0	VID_FRACFREQ3	0-FF FFFFh	Synth3 Frequency fractional divider.

1.16.1.2.33 Video PLL Divider 3 Register (VIDEOPLL_DIV3)

The VIDEOPLL_DIV3 register is used to control the VIDEO PLL Clock 3 post-divider frequency of the SYSCLK15 and STC0 source clocks. The default DIV3 value is 2.

The Video PLL Divider 3 Register (VIDEOPLL_DIV3) is shown in Figure 1-152 and described in Table 1-201.

Figure 1-152. Video PLL Divider 3 Register (VIDEOPLL_DIV3)

31 9	8	7 0
Reserved	VID_LDMDIV3	VID_MDIV3
R-0	R/W-1	R/W-2h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-201. Video PLL Divider 3 Register (VIDEOPLL_DIV3) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	VID_LDMDIV5	1-0	Load Synth3 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into VIDEO Synthesizer3.
7-0	VID_MDIV5	0-FFh	Synth3 Frequency M Post Divider.

1.16.1.2.34 Audio PLL Control Register (AUDIOPLL_CTRL)

The AUDIOPLL_CTRL register is used to control the base frequency of the Audio PLL. The PLL is sourced by the audio reference clock from the Main PLL. The default frequency of the Audio PLL is 1105.92 MHz.

The Audio PLL Control Register (AUDIOPLL_CTRL) is shown in Figure 1-153 and described in Table 1-202.

Figure 1-153. Audio PLL Control Register (AUDIOPLL_CTRL)

31	16
AUDIO_N	
R/W-40h	

15	8
AUDIO_P	
R/W-19h	

7	6 4	3	2	1	0
AUDIO_LOCK	Reserved	AUDIO_PLLEN	AUDIO_BP	Reserved	AUDIO_LOC_CTL
R-0	R-0	R/W-1	R/W-1	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-202. Audio PLL Control Register (AUDIOPLL_CTRL) Field Descriptions

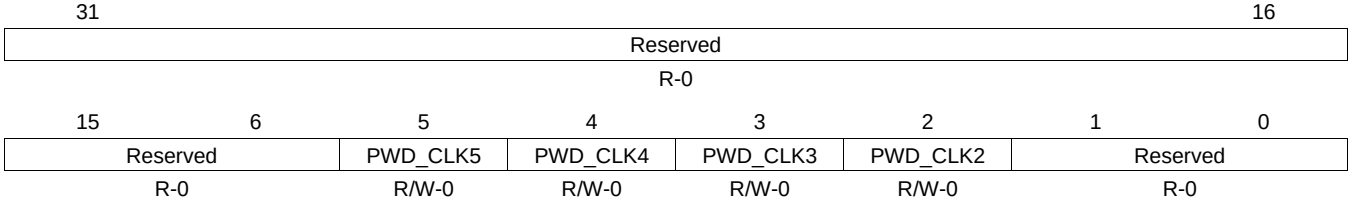
Bit	Field	Value	Description
31-16	AUDIO_N	0-FFFFh	AUDIO PLL N multiplier value.
15-8	AUDIO_P	0-FFh	AUDIO PLL P divider value.
7	AUDIO_LOCK	1-0	AUDIO PLL lock status bit.
6-4	Reserved	0	Reserved. Reads returns 0.
3	AUDIO_PLLEN		AUDIO PLL enable.
		0	PLL is disabled (power down state).
		1	PLL is enabled (normal operation).
2	AUDIO_BP		AUDIO PLL bypass enable.
		0	Normal operation.
		1	PLL in bypass mode, reference clock driven at output.
1	Reserved	0	Reserved. Reads returns 0.
0	AUDIO_LOC_CTL		AUDIO PLL lock output select.
		0	Analog lock
		1	Digital lock

1.16.1.2.35 Audio PLL Powerdown Register (AUDIOPLL_PWD)

The AUDIOPLL_PWD register is used to powerdown the individual output clocks of the Audio PLL.

The Audio PLL Powerdown Register (AUDIOPLL_PWD) is shown in Figure 1-154 and described in Table 1-203.

Figure 1-154. Audio PLL Powerdown Register (AUDIOPLL_PWD)



LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-203. Audio PLL Powerdown Register (AUDIOPLL_PWD) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved. Reads returns 0.
5	PWD_CLK5	1-0	AUDIO PLL Clock5 Powerdown. Setting this bit powers down clock 5 (source clock for SYSCLK22).
4	PWD_CLK4	1-0	AUDIO PLL Clock4 Powerdown. Setting this bit powers down clock 4 (source clock for SYSCLK21).
3	PWD_CLK3	1-0	AUDIO PLL Clock3 Powerdown. Setting this bit powers down clock 3 (source clock for SYSCLK20).
2	PWD_CLK2	1-0	AUDIO PLL Clock2 Powerdown. Setting this bit powers down clock 2 (source clock for SYSCLK19).
1-0	Reserved	0	Reserved. Reads returns 0.

1.16.1.2.36 Audio PLL Frequency 2 Register (AUDIOPLL_FREQ2)

The AUDIOPLL_FREQ2 register is used to control the Audio PLL Clock 2 pre-divider frequency of the SYSCLK19 (Transport Out) clock. The default FREQ2 value is 14.0 to generate a SYSCLK19 of approximately 160 MHz.

The Audio PLL Frequency 2 Register (AUDIOPLL_FREQ2) is shown in Figure 1-155 and described in Table 1-204.

Figure 1-155. Audio PLL Frequency 2 Register (AUDIOPLL_FREQ2)

31	30 29	28	27 24	23 0
AUD_LDFREQ2	Reserved	AUD_TRUNC2	AUD_INTFREQ2	AUD_FRACFREQ2
R/W-1	R-0	R/W-0	R/W-Eh	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-204. Audio PLL Frequency 2 Register (AUDIOPLL_FREQ2) Field Descriptions

Bit	Field	Value	Description
31	AUD_LDFREQ2	1-0	Load Synth2 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into AUDIO Synthesizer2.
30-29	Reserved	0	Reserved. Reads returns 0.
28	AUD_TRUNC2	1-0	Synth2 Enable Truncate Correction.
27-24	AUD_INTFREQ2	8h-Fh	Synth2 Frequency integer divider.
23-0	AUD_FRACFREQ2	0-FF FFFFh	Synth2 Frequency fractional divider.

1.16.1.2.37 Audio PLL Divider 2 Register (AUDIOPLL_DIV2)

The AUDIOPLL_DIV2 register is used to control the AUDIO PLL Clock 2 post-divider frequency of the SYSCLK19 clock. The default DIV2 value is 4 that results in an Audio PLL CLK2 of approximately160 MHz.

The Audio PLL Divider 2 Register (AUDIOPLL_DIV2) is shown in Figure 1-156 and described in Table 1-205.

Figure 1-156. Audio PLL Divider 2 Register (AUDIOPLL_DIV2)

31 9	8	7 0
Reserved	AUD_LDMDIV2	AUD_MDIV2
R-0	R/W-1	R/W-4h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-205. Audio PLL Divider 2 Register (AUDIOPLL_DIV2) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	AUD_LDMDIV2	1-0	Load Synth2 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into AUDIO Synthesizer2.
7-0	AUD_MDIV2	0-FFh	Synth2 Frequency M Post Divider.

1.16.1.2.38 Audio PLL Frequency 3 Register (AUDIOPLL_FREQ3)

The AUDIOPLL_FREQ3 register is used to control the Audio PLL Clock 3 pre-divider frequency of the SYSCLK20 (Audio1) clock. The default FREQ3 value is 9.0 for a SYSCLK20 frequency of 196.608 MHz.

The Audio PLL Frequency 3 Register (AUDIOPLL_FREQ3) is shown in Figure 1-157 and described in Table 1-206.

Figure 1-157. Audio PLL Frequency 3 Register (AUDIOPLL_FREQ3)

31	30 29	28	27 24	23 0
AUD_LDFREQ3	Reserved	AUD_TRUNC3	AUD_INTFREQ3	AUD_FRACFREQ3
R/W-1	R-0	R/W-0	R/W-9h	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-206. Audio PLL Frequency 3 Register (AUDIOPLL_FREQ3) Field Descriptions

Bit	Field	Value	Description
31	AUD_LDFREQ3	1-0	Load Synth3 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into AUDIO Synthesizer3.
30-29	Reserved	0	Reserved. Reads returns 0.
28	AUD_TRUNC3	1-0	Synth3 Enable Truncate Correction.
27-24	AUD_INTFREQ3	8h-Fh	Synth3 Frequency integer divider.
23-0	AUD_FRACFREQ3	0-FF FFFFh	Synth3 Frequency fractional divider.

1.16.1.2.39 Audio PLL Divider 3 Register (AUDIOPLL_DIV3)

The AUDIOPLL_DIV3 register is used to control the AUDIO PLL Clock 3 post-divider frequency of the SYSCLK20 clock. The default DIV3 value is 5 which results in an Audio PLL CLK3 of 196.6080 MHz.

The Audio PLL Divider 3 Register (AUDIOPLL_DIV3) is shown in Figure 1-158 and described in Table 1-207.

Figure 1-158. Audio PLL Divider 3 Register (AUDIOPLL_DIV3)

31 9	8	7 0
Reserved	AUD_LDMDIV3	AUD_MDIV3
R-0	R/W-1	R/W-5h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-207. Audio PLL Divider 3 Register (AUDIOPLL_DIV3) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	AUD_LDMDIV3	1-0	Load Synth3 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into AUDIO Synthesizer3.
7-0	AUD_MDIV3	0-FFh	Synth3 Frequency M Post Divider.

1.16.1.2.40 Audio PLL Frequency 4 Register (AUDIOPLL_FREQ4)

The AUDIOPLL_FREQ4 register is used to control the Audio PLL Clock 4 pre-divider frequency of the SYSCLK21 (Audio2) clock. The default FREQ4 value is 9.795918.

The Audio PLL Frequency 4 Register (AUDIOPLL_FREQ4) is shown in Figure 1-159 and described in Table 1-208.

Figure 1-159. Audio PLL Frequency 4 Register (AUDIOPLL_FREQ4)

31	30 29	28	27 24	23 0
AUD_LDFREQ4	Reserved	AUD_TRUNC4	AUD_INTFREQ4	AUD_FRACFREQ4
R/W-1	R-0	R/W-0	R/W-9h	R/W-CBC148h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-208. Audio PLL Frequency 4 Register (AUDIOPLL_FREQ4) Field Descriptions

Bit	Field	Value	Description
31	AUD_LDFREQ4	1-0	Load Synth4 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into AUDIO Synthesizer4.
30-29	Reserved	0	Reserved. Reads returns 0.
28	AUD_TRUNC4	1-0	Synth4 Enable Truncate Correction.
27-24	AUD_INTFREQ4	8h-Fh	Synth4 Frequency integer divider.
23-0	AUD_FRACFREQ4	0-FF FFFFh	Synth4 Frequency fractional divider.

1.16.1.2.41 Audio PLL Divider 4 Register (AUDIOPLL_DIV4)

The AUDIOPLL_DIV4 register is used to control the AUDIO PLL Clock 4 post-divider frequency of the SYSCLK21 clock. The default DIV4 value is 20. which results in an Audio PLL CLK4 of 45.1584 MHz.

The Audio PLL Divider 4 Register (AUDIOPLL_DIV4) is shown in Figure 1-160 and described in Table 1-209.

Figure 1-160. Audio PLL Divider 4 Register (AUDIOPLL_DIV4)

31 9	8	7 0
Reserved	AUD_LDMDIV4	AUD_MDIV4
R-0	R/W-1	R/W-5h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-209. Audio PLL Divider 4 Register (AUDIOPLL_DIV4) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	AUD_LDMDIV4	1-0	Load Synth4 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into AUDIO Synthesizer4.
7-0	AUD_MDIV4	0-FFh	Synth4 Frequency M Post Divider.

1.16.1.2.42 Audio PLL Frequency 5 Register (AUDIOPLL_FREQ5)

The AUDIOPLL_FREQ5 register is used to control the Audio PLL Clock 5 pre-divider frequency of the SYSCLK22 (Audio3) clock. The default FREQ5 value is 13.5.

The Audio PLL Frequency 5 Register (AUDIOPLL_FREQ5) is shown in Figure 1-161 and described in Table 1-210.

Figure 1-161. Audio PLL Frequency 5 Register (AUDIOPLL_FREQ5)

31	30 29	28	27 24	23 0
AUD_LDFREQ5	Reserved	AUD_TRUNC5	AUD_INTFREQ5	AUD_FRACFREQ5
R/W-1	R-0	R/W-0	R/W-Dh	R/W-800000h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-210. Audio PLL Frequency 5 Register (AUDIOPLL_FREQ5) Field Descriptions

Bit	Field	Value	Description
31	AUD_LDFREQ5	1-0	Load Synth5 FREQ value. Setting this bit to 1 causes the INTFREQ and FRACFREQ values to be loaded into AUDIO Synthesizer5.
30-29	Reserved	0	Reserved. Reads returns 0.
28	AUD_TRUNC5	1-0	Synth5 Enable Truncate Correction.
27-24	AUD_INTFREQ5	8h-Fh	Synth5 Frequency integer divider.
23-0	AUD_FRACFREQ5	0-FF FFFFh	Synth5 Frequency fractional divider.

1.16.1.2.43 Audio PLL Divider 5 Register (AUDIOPLL_DIV5)

The AUDIOPLL_DIV5 register is used to control the AUDIO PLL Clock 5 post-divider frequency of the SYSCLK22 clock. The default DIV5 value is 20 which results in an Audio PLL CLK5 of 32.768 MHz.

The Audio PLL Divider Register (AUDIOPLL_DIV5) is shown in Figure 1-162 and described in Table 1-211.

Figure 1-162. Audio PLL Divider 5 Register (AUDIOPLL_DIV5)

31 9	8	7 0
Reserved	AUD_LDMDIV5	AUD_MDIV5
R-0	R/W-1	R/W-14h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-211. Audio PLL Divider 5 Register (AUDIOPLL_DIV5) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Reads returns 0.
8	AUD_LDMDIV5	1-0	Load Synth5 M Divider value. Setting this bit to 1 causes the M Divider value to be loaded into AUDIO Synthesizer5.
7-0	AUD_MDIV5	0-FFh	Synth5 Frequency M Post Divider.

1.16.1.3 Device Configuration Registers

Table 1-212 lists the registers of the device configuration registers. For the base address of these registers, see Table 1-12.

Table 1-212. Device Configuration Registers

Address Offset	Acronym	Register Description	Section
600h	DEVICE_ID	Device Identification Register	Section 1.16.1.3.1
608h	INIT_PRESSURE_0	Initiator Pressure 0 Register	Section 1.16.1.3.2
60Ch	INIT_PRESSURE_1	Initiator Pressure 1 Register	Section 1.16.1.3.3
610h	MMU_CFG	MMU Configuration Register	Section 1.16.1.3.4
614h	TPTC_CFG	TPTC Configuration Register	Section 1.16.1.3.5
618h	DDR_CTRL	DDR Control Register	Section 1.16.1.3.6
61Ch	DSP_IDLE_CFG	DSP Standby/Idle Management Register	Section 1.16.1.3.7
620h	USB_CTRL	USB Control Register	Section 1.16.1.3.8
624h	USBPHY_CTRL0	USB Phy Control Register 0	Section 1.16.1.3.9
628h	Reserved	Reserved	
62Ch	USBPHY_CTRL1	USB Phy Control Register 1	Section 1.16.1.3.10
630h	MAC_ID0_LO	Ethernet MAC ID0 Low Register	Section 1.16.1.3.11
634h	MAC_ID0_HI	Ethernet MAC ID0 High Register	Section 1.16.1.3.12
638h	MAC_ID1_LO	Ethernet MAC ID1 Low Register	Section 1.16.1.3.13
63Ch	MAC_ID1_HI	Ethernet MAC ID1 High Register	Section 1.16.1.3.14
640h	PCIe_CFG	PCIe Configuration Register	Section 1.16.1.3.15
644h	Reserved	Reserved	
648h	CLK_CTL	Clock Control Register	Section 1.16.1.3.16
64Ch	AUD_CTRL	Audio Interface Control Register	Section 1.16.1.3.17
650h	DSPMEM_SLEEP	DSP L2 Memory Sleep Mode Register	Section 1.16.1.3.18
654h	OCMEM_SLEEP	On-Chip Memory Sleep Mode Register	Section 1.16.1.3.19
660h	HD_DAC_CTRL	HD DAC Control Register	Section 1.16.1.3.20
664h	HD_DACA_CAL	HD DAC A Calibration Register	Section 1.16.1.3.21
668h	HD_DACB_CAL	HD DAC B Calibration Register	Section 1.16.1.3.22
66Ch	HD_DACC_CAL	HD DAC C Calibration Register	Section 1.16.1.3.23
670h	SD_DAC_CTRL	SD DAC Control Register	Section 1.16.1.3.24
674h	SD_DACA_CAL	SD DAC A Calibration Register	Section 1.16.1.3.25
678h	SD_DACB_CAL	SD DAC B Calibration Register	Section 1.16.1.3.26
67Ch	SD_DACC_CAL	SD DAC C Calibration Register	Section 1.16.1.3.27
680h	SD_DACD_CAL	SD DAC D Calibration Register	Section 1.16.1.3.28
684h-68Ch	Reserved	Reserved	
690h	HW_EVT_SEL_GRP1	HW Event Select (Group 1) Register	Section 1.16.1.3.29
694h	HW_EVT_SEL_GRP2	HW Event Select (Group 2) Register	Section 1.16.1.3.30
698h	HW_EVT_SEL_GRP3	HW Event Select (Group 3) Register	Section 1.16.1.3.31
69Ch	HW_EVT_SEL_GRP4	HW Event Select (Group 4) Register	Section 1.16.1.3.32
6F8h	HDMI_OBSCLK_CTRL	HDMI Observe Clock Control	Section 1.16.1.3.33
6FCh	SERDES_CTRL	Serdes Control Register	Section 1.16.1.3.34
700h	USB_CLK_CTL	USB Clock Control Register	Section 1.16.1.3.35
704h	PLL_OBSCLK_CTRL	PLL Observe Clock Control Register	Section 1.16.1.3.36
70Ch	DDR_RCD	DDR RCD Register	Section 1.16.1.3.37

1.16.1.3.1 Device Identification Register (DEVICE_ID)

The DEVICE_ID register contains a software readable version of the device JTAG ID. Software can use this register to determine the version of the device on which it is executing.

The Device Identification Register (DEVICE_ID) is shown in Figure 1-163 and described in Table 1-213.

Figure 1-163. Device Identification Register (DEVICE_ID)

31 28	27 12	11 1	0
DEVREV	PARTNUM	MFGR	Reserved
R-eFuse	R-B81Eh	R-017h	R-1

LEGEND: R = Read only; -*n* = value after reset

Table 1-213. Device Identification Register (DEVICE_ID) Field Descriptions

Bit	Field	Value	Description
31-28	DEVREV	0-Fh	Device revision.
27-12	PARTNUM	0-FFFFh	Device part number (unique JTAG ID).
11-1	MFGR	0-7FFh	Manufacturer's JTAG ID.
0	Reserved	0	Reserved. Always 1.

1.16.1.3.2 Initiator Pressure 0 Register (INIT_PRESSURE_0)

The INIT_PRESSURE_0 register configures the infrastructure “pressure” input for L3 initiators to provide dynamic pressure escalation.

Pressure value of 2h is reserved. Pressure value of 3h wins arbitration over a pressure of 1h which wins over a pressure of 0h.

The Initiator Pressure 0 Register (INIT_PRESSURE_0) is shown in Figure 1-164 and described in Table 1-214.

Figure 1-164. Initiator Pressure 0 Register (INIT_PRESSURE_0)

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
TCWR3	TCRD3	TCWR2	TCRD2	TCWR1	TCRD1	TCWR0	TCRD0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
JTAG	Reserved	DSS1	DSS0	System MMU	GEM_CFG	GEM_MDMA	HOST_ARM
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-214. Initiator Pressure 0 Register (INIT_PRESSURE_0) Field Descriptions

Bit	Field	Value	Description
31-30	TCWR3	0-3h	TPTC 3 Write Port initiator pressure.
29-28	TCRD3	0-3h	TPTC 3 Read Port initiator pressure.
27-26	TCWR2	0-3h	TPTC 2 Write Port initiator pressure.
25-24	TCRD2	0-3h	TPTC 2 Read Port initiator pressure.
23-22	TCWR1	0-3h	TPTC 1 Write Port initiator pressure.
21-20	TCRD1	0-3h	TPTC 1 Read Port initiator pressure.
19-18	TCWR0	0-3h	TPTC 0 Write Port initiator pressure.
17-16	TCRD0	0-3h	TPTC 0 Read Port initiator pressure.
15-14	JTAG	0-3h	JTAG Interface Port Initiator pressure.
13-12	Reserved	0	Reserved. Reads returns 0.
11-10	DSS1	0-3h	Display Subsystem Port 1 initiator pressure.
9-8	DSS0	0-3h	Display Subsystem Port 0 initiator pressure.
7-6	System MMU	0-3h	System MMU initiator pressure.
5-4	GEM_CFG	0-3h	C674x DSP CFG port initiator pressure.
3-2	GEM_MDMA	0-3h	C674x DSP DMA port initiator pressure.
1-0	HOST_ARM	0-3h	Cortex-A8 initiator pressure.

1.16.1.3.3 Initiator Pressure 1 Register (INIT_PRESSURE_1)

The INIT_PRESSURE_1 register configures the infrastructure “pressure” input for L3 initiators to provide dynamic pressure escalation.

The Initiator Pressure 1 Register (INIT_PRESSURE_1) is shown in Figure 1-165 and described in Table 1-215.

Figure 1-165. Initiator Pressure 1 Register (INIT_PRESSURE_1)

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
IVAHD2	IVAHD1	IVAHD0	DEBUG	EXP	GRFX	Reserved	PCIE
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15 14	13 10	9 8	7 6	5 4	3 2	1 0
DSS_CTLR	Reserved	SATA	USB_QMGR	USB_DMA	CPGMAC1	CPGMAC0
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-215. Initiator Pressure 1 Register (INIT_PRESSURE_1) Field Descriptions

Bit	Field	Value	Description
31-30	IVAHD2	0-3h	IVAHD2 (HDVICP2-2) initiator pressure.
29-28	IVAHD1	0-3h	IVAHD1 (HDVICP2-1) initiator pressure.
27-26	IVAHD0	0-3h	IVAHD0 (HDVICP2-0) initiator pressure.
25-24	DEBUG	0-3h	Debug Subsystem initiator pressure.
23-22	EXP	0-3h	Expansion Slot Port initiator pressure.
21-20	GRFX	0-3h	3D Graphics (SGX530) initiator pressure.
19-18	Reserved	0	Reserved.
17-16	PCIE	0-3h	PCIe initiator pressure.
15-14	DSS_CTLR	0-3h	HDVPSS Controller initiator pressure
13-10	Reserved	0	Reserved. Reads returns 0.
9-8	SATA	0-3h	SATA initiator pressure.
7-6	USB_QMGR	0-3h	USB Queue Manager initiator pressure.
5-4	USB_DMA	0-3h	USB DMA port initiator pressure.
3-2	CPGMAC1	0-3h	CPGMAC1 initiator pressure.
1-0	CPGMAC0	0-3h	CPGMAC0 initiator pressure.

1.16.1.3.4 MMU Configuration Register (MMU_CFG)

The MMU_CFG register is used to optionally route certain initiators through the System MMU. The C674x DSP MDMA port accesses will always be routed through the System MMU so no configuration bit is instantiated for C674x DSP. Note that when TPTC0 is routed through the MMU accesses to the C674x DSP SDMA port will still by-pass the MMU for increased performance.

The MMU Configuration Register (MMU_CFG) is shown in Figure 1-166 and described in Table 1-216.

Figure 1-166. MMU Configuration Register (MMU_CFG)

31	16
Reserved	
R-0	

15	14 8	7	6 2	1	0
MMU_ABORT	Reserved	MMU_EN	Reserved	EXPMMU	TC0MMU
R/W-0	R-0	R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-216. MMU Configuration Register (MMU_CFG) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. Reads returns 0.
15	MMU_ABORT	1-0	MMU abort operation. This bit causes the MMU to abort the current operation in case of lockup.
14-8	Reserved	0	Reserved. Reads returns 0.
7	MMU_EN	1-0	MMU Enable. Clearing this bit disables MMU table lookup and causes accesses to use the non-translated address. This bit defaults to enabled but an identical bit within an MMU configuration register defaults to disabled and must be set after the page tables are programmed for MMU operation.
6-2	Reserved	0	Reserved. Reads returns 0.
1	EXPMMU	1-0	Expansion Slot uses MMU.
0	TC0MMU	1-0	TPTC0 Uses MMU.

1.16.1.3.5 TPTC Configuration Register (TPTC_CFG)

The TPTC_CFG register configures the default burst size for TPTC0, 1, 2, and 3.

The TPTC Configuration Register (TPTC_CFG) is shown in Figure 1-167 and described in Table 1-217.

Figure 1-167. TPTC Configuration Register (TPTC_CFG)

31	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
TC3DBS		TC2DBS		TC1DBS		TC0DBS	
R/W-3h		R/W-3h		R/W-3h		R/W-3h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-217. TPTC Configuration Register (TPTC_CFG) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Reads returns 0.
7-6	TC3DBS		TC3 Default Burst Size
		0	16 byte
		1h	32 byte
		2h	64 byte
		3h	128 byte
5-4	TC2DBS		TC2 Default Burst Size.
		0	16 byte
		1h	32 byte
		2h	64 byte
		3h	128 byte
3-2	TC1DBS		TC1 Default Burst Size.
		0	16 byte
		1h	32 byte
		2h	64 byte
		3h	128 byte
1-0	TC0DBS		TC0 Default Burst Size.
		0	16 byte
		1h	32 byte
		2h	64 byte
		3h	128 byte

1.16.1.3.6 DDR Control Register (DDR_CTRL)

The DDR_CTRL register reflects the DDR I/O timing as programmed in the device eFuse and controls DDR self-refresh operation.

The DDR Control Register (DDR_CTRL) is shown in Figure 1-168 and described in Table 1-218.

Figure 1-168. DDR Control Register (DDR_CTRL)

31	30	29	28	27	26	25	24
Reserved		FORCE_PHY_RST	DIS_DEV_RST	Reserved		KEEPSREF1	SREF1
R-0		R/W-0	R/W-0	R-0		R/W-0	R/W-0

23	20	19	18	17	16
Reserved		DDRDATA_SLEW1		DDRCMD_SLEW1	
R-0		R-eFuse		R-eFuse	

15	10	9	8
Reserved		KEEPSREF0	SREF0
R-0		R/W-0	R/W-0

7	4	3	2	1	0
Reserved		DDRDATA_SLEW0		DDRCMD_SLEW0	
R-0		R-eFuse		R-eFuse	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-218. DDR Control Register (DDR_CTRL) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved. Reads returns 0.
29	FORCE_PHY_RST	1-0	Force DDR Phy Reset. When set, this bit forces a reset of the DDR Phys to allow DDR clock frequency to be changed.
28	DIS_DEV_RST	1-0	Disable DDR Device Reset. Setting this bit cause the RESETn outputs for the DDR0 and DDR1 interfaces to be held high. This allows power-down and subsequent power-up of EMIFs without resetting DDR devices.
27-26	Reserved	0	Reserved. Reads returns 0.
25	KEEPSREF1	1-0	Keep DDR1 Self Refreshed. Setting this bit keeps the DDR1 CLKEN driven to maintain self refresh when the Active power domain is disabled.
24	SREF1	1-0	Enable DDR1 Self Refresh.
23-20	Reserved	0	Reserved. Reads returns 0.
19-18	DDRDATA_SLEW1	0-3h	DDR1 Data Slew. Reflects the DDR1 Data Macro slew adjustment programmed in eFuse.
17-16	DDRCMD_SLEW1	0-3h	DDR1 CMD Slew. Reflects the DDR1 CMD Macro slew adjustment programmed in eFuse.
15-10	Reserved	0	Reserved. Reads returns 0.
9	KEEPSREF0	1-0	Keep DDR0 Self Refreshed. Setting this bit keeps the DDR0 CLKEN driven to maintain self refresh when the Active power domain is disabled.
8	SREF0	1-0	Enable DDR0 Self Refresh.
7-4	Reserved	0	Reserved. Reads returns 0.
3-2	DDRDATA_SLEW0	0-3h	DDR0 Data Slew. Reflects the DDR0 Data Macro slew adjustment programmed in eFuse.
1-0	DDRCMD_SLEW0	0-3h	DDR0 CMD Slew. Reflects the DDR0 CMD Macro slew adjustment programmed in eFuse.

1.16.1.3.7 DSP Standby/Idle Management Register (DSP_IDLE_CFG)

The DSP_IDLE_CFG register controls the state management of the C674x DSP initiator and target port interfaces.

The DSP Standby/Idle Management Register (DSP_IDLE_CFG) is shown in Figure 1-169 and described in Table 1-219.

Figure 1-169. DSP Standby/Idle Management Register (DSP_IDLE_CFG)

31	16
Reserved	
R-0	

15	14 6	5 4	3 2	1 0
DSPSTBY	Reserved	STBYMODE	IDLEMODE	Reserved
R/W-0	R-0	R/W-10	R/W-10	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-219. DSP Standby/Idle Management Register (DSP_IDLE_CFG) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. Reads returns 0.
15	DSPSTBY	1-0	Setting this bit will initiate the standby for DSP.
14-6	Reserved	0	Reserved. Reads returns 0.
5-4	STBYMODE		Initiator state management mode
		0	Force-standby mode: Initiator is unconditionally placed in standby state. (for debug only)
		1h	No-standby mode: Initiator is unconditionally removed from standby state. (for debug only)
		2h	Smart-standby mode: Initiator standby status depends on local condition (module functional requirements of initiator). No initiator related wake-up events.
		3h	Smart-standby wakeup-capable mode. : Initiator standby status depends on local condition (module functional requirements of initiator). Initiator related wake-up events may be generated when in standby.
3-2	IDLEMODE		Target state management mode.
		0	Force-idle mode: Target acknowledges idle requests unconditionally (debug only)
		1h	No-idle mode: Target never enters idle state. (for debug only)
		2h	Smart-idle mode: Target acknowledges idle requests after fulfilling IP internal requirements. No IRQ/DMA related wake-up events.
		3h	Smart-idle wakeup-capable mode: Target acknowledges idle requests after fulfilling IP internal requirements. IRQ/DMA related wake-up events may be generated when in standby.
1-0	Reserved	0	Reserved. Reads returns 0.

1.16.1.3.8 USB Control Register (USB_CTRL)

The USB_CTRL register controls various features of the USB subsystem.

The USB Control Register (USB_CTRL) is shown in Figure 1-170 and described in Table 1-220.

Figure 1-170. USB Control Register (USB_CTRL)

31	16
Reserved	
R-0	

15	9	8	7	2	1	0
Reserved		PHYCLKSRC	Reserved		PHYSLEEP1	PHYSLEEP0
R-0		R/W-0	R-0		R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-220. USB Control Register (USB_CTRL) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	PHYCLKSRC		USB PHY reference clock source
		0	Use PLL reference clock
		1	Use USB oscillator
7-2	Reserved	0	Reserved
1	PHYSLEEP1		USB PHY1 sleep mode control. Places Phy1 in Sleep mode per USB 2.0 LPM addendum.
		0	Sleep mode
		1	Normal operating mode
0	PHYSLEEP0		USB PHY0 sleep mode control. Places Phy0 in Sleep mode per USB 2.0 LPM addendum.
		0	Sleep mode
		1	Normal operating mode

1.16.1.3.9 USB Phy Control Register 0 (USBPHY_CTRL0)

The USBPHY_CTRL0 register controls various features of the USB0 Phy.

The USB Phy Control Register 0 (USBPHY_CTRL0) is shown in Figure 1-171 and described in Table 1-221.

Figure 1-171. USB Phy Control Register 0 (USBPHY_CTRL0)

31	30 28	27 23	22 20	19 16
Rsvd	COMPDISTUNE	Reserved	SQRXTUNE	TXFSLSTUNE
R-0	R/W-4h	R-0	R/W-3h	R/W-3h

15	14 13	12	11 8	7 6	5 4	3 0
Rsvd	TXPREEMTUNE	TXRISETUNE	TXVREFTUNE	Reserved	TXHSXVTUNE	Reserved
R-0	R/W-0	R/W-0	R/W-8h	R-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-221. USB Phy Control Register 0 (USBPHY_CTRL0) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	COMPDISTUNE	1-0	Disconnect Threshold Adjust.
27-23	Reserved	0	Reserved
22-20	SQRXTUNE	1-0	Squelch Threshold Adjust.
19-16	TXFSLSTUNE	1-0	FS/LS Source Impedance Adjust.
15	Reserved	0	Reserved
14-13	TXPREEMTUNE	1-0	HS Transmit Pre-Emphasis Enable.
12	TXRISETUNE	1-0	HS Transmit Rise/Fall Time Adjust.
11-8	TXVREFTUNE	1-0	HS DC Voltage Level Adjust.
7-6	Reserved	0	Reserved
5-4	TXHSXVTUNE	1-0	Transmit High-Speed Crossover Adjust.
3-0	Reserved	0	Reserved

1.16.1.3.10 USB Phy Control Register 1 (USBPHY_CTRL1)

The USBPHY_CTRL1 register controls various features of the USB1 Phy.

The USB Phy Control Register 1 (USBPHY_CTRL1) is shown in Figure 1-172 and described in Table 1-222.

Figure 1-172. USB Phy Control Register 1 (USBPHY_CTRL1)

31	30 28	27 23	22 20	19 16
Rsvd	COMPDISTUNE	Reserved	SQRXTUNE	TXFSLSTUNE
R-0	R/W-4h	R-0	R/W-3h	R/W-3h

15	14 13	12	11 8	7 6	5 4	3 0
Rsvd	TXPREEMTUNE	TXRISETUNE	TXVREFTUNE	Reserved	TXHSXVTUNE	Reserved
R-0	R/W-0	R/W-0	R/W-8h	R-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-222. USB Phy Control Register 1 (USBPHY_CTRL1) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	COMPDISTUNE	1-0	Disconnect Threshold Adjust.
27-23	Reserved	0	Reserved
22-20	SQRXTUNE	1-0	Squelch Threshold Adjust.
19-16	TXFSLSTUNE	1-0	FS/LS Source Impedance Adjust.
15	Reserved	0	Reserved
14-13	TXPREEMTUNE	1-0	HS Transmit Pre-Emphasis Enable.
12	TXRISETUNE	1-0	HS Transmit Rise/Fall Time Adjust.
11-8	TXVREFTUNE	1-0	HS DC Voltage Level Adjust.
7-6	Reserved	0	Reserved
5-4	TXHSXVTUNE	1-0	Transmit High-Speed Crossover Adjust.
3-0	Reserved	0	Reserved

1.16.1.3.11 Ethernet MAC ID0 Low Register (MAC_ID0_LO)

The MAC_ID0_LO register contains the lower 2 bytes of the 48-bit ID for MAC0.

The Ethernet MAC ID0 Low Register (MAC_ID0_LO) is shown in Figure 1-173 and described in Table 1-223.

Figure 1-173. Ethernet MAC ID0 Low Register (MAC_ID0_LO)

31 16	15 8	7 0
Reserved	MACADDR[7:0]	MACADDR[15:8]
R-0	R-eFuse	R-eFuse

LEGEND: R = Read only; -*n* = value after reset

Table 1-223. Ethernet MAC ID0 Low Register (MAC_ID0_LO) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. Reads returns 0.
15-8	MACADDR[7:0]	0-FFh	MAC0 Address – Byte 0.
7-0	MACADDR[15:8]	0-FFh	MAC0 Address – Byte 1.

1.16.1.3.12 Ethernet MAC ID0 High Register (MAC_ID0_HI)

The MAC_ID0_HI register contains the upper 4 bytes of the 48-bit ID for MAC0.

The Ethernet MAC ID0 High Register (MAC_ID0_HI) is shown in Figure 1-174 and described in Table 1-224.

Figure 1-174. Ethernet MAC ID0 High Register (MAC_ID0_HI)

31 24	23 16	15 8	7 0
MACADDR[23:16]	MACADDR[31:24]	MACADDR[39:32]	MACADDR[47:40]
R-eFuse	R-eFuse	R-eFuse	R-eFuse

LEGEND: R = Read only; -*n* = value after reset

Table 1-224. Ethernet MAC ID0 High Register (MAC_ID0_HI) Field Descriptions

Bit	Field	Value	Description
31-24	MACADDR[23:16]	0-FFh	MAC0 Address – Byte 2.
23-16	MACADDR[31:24]	0-FFh	MAC0 Address – Byte 3.
15-8	MACADDR[39:32]	0-FFh	MAC0 Address – Byte 4.
7-0	MACADDR[47:40]	0-FFh	MAC0 Address – Byte 5.

1.16.1.3.13 Ethernet MAC ID1 Low Register (MAC_ID1_LO)

The MAC_ID1_LO register contains the lower 2 bytes of the 48-bit ID for MAC1.

The Ethernet MAC ID1 Low Register (MAC_ID1_LO) is shown in Figure 1-175 and described in Table 1-225.

Figure 1-175. Ethernet MAC ID1 Low Register (MAC_ID1_LO)

31 16	15 8	7 0
Reserved	MACADDR[7:0]	MACADDR[15:8]
R-0	R-eFuse	R-eFuse

LEGEND: R = Read only; -*n* = value after reset

Table 1-225. Ethernet MAC ID1 Low Register (MAC_ID1_LO) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. Reads returns 0.
15-8	MACADDR[7:0]	0-FFh	MAC1 Address – Byte 0.
7-0	MACADDR[15:8]	0-FFh	MAC1 Address – Byte 1.

1.16.1.3.14 Ethernet MAC ID1 High Register (MAC_ID1_HI)

The MAC_ID1_HI register contains the upper 4 bytes of the 48-bit ID for MAC1.

The Ethernet MAC ID1 High Register (MAC_ID1_HI) is shown in Figure 1-176 and described in Table 1-226.

Figure 1-176. Ethernet MAC ID1 High Register (MAC_ID1_HI)

31 24	23 16	15 8	7 0
MACADDR[23:16]	MACADDR[31:24]	MACADDR[39:32]	MACADDR[47:40]
R-eFuse	R-eFuse	R-eFuse	R-eFuse

LEGEND: R = Read only; -*n* = value after reset

Table 1-226. Ethernet MAC ID1 High Register (MAC_ID1_HI) Field Descriptions

Bit	Field	Value	Description
31-24	MACADDR[23:16]	0-FFh	MAC1 Address – Byte 2.
23-16	MACADDR[31:24]	0-FFh	MAC1 Address – Byte 3.
15-8	MACADDR[39:32]	0-FFh	MAC1 Address – Byte 4.
7-0	MACADDR[47:40]	0-FFh	MAC1 Address – Byte 5.

1.16.1.3.15 PCIE Configuration Register (PCIE_CFG)

The PCIE_CFG Register controls operation of the PCI Express module.

The PCIE Configuration Register (PCIE_CFG) is shown in Figure 1-177 and described in Table 1-227.

Figure 1-177. PCIE Configuration Register (PCIE_CFG)

31	16
PCIE_CFGPLL	
R/W-01C9h	

15	12	11	8	7	2	1	0
Reserved		PCIE_STSPLL		Reserved		PCIE_DEVTYPE	
R-0		R-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-227. PCIE Configuration Register (PCIE_CFG) Field Descriptions

Bit	Field	Value	Description
31-16	PCIE_CFGPLL	0-FFFFh	PCIe PLL Configuration. Enables 25× multiplier. **Bit [31]** is **Reserved** and must be kept at 0. **Bit [30:29] CLKBYP** Clock bypass. Facilitates bypassing of the PLL with **refclkp / n**. 0: The transmitter operates normally from PLL. 1h-2h: Reserved 3h: The PLL clock is bypassed by **refclkp / n**. **Bit [28:27] LB** Keep these bits at 0 (default). **Bit [26] SLEEP_PLL** Puts the PLL into low power sleep state when written to '1'. In this mode, the core of PLL remains locked to refclkp/n, but all clock distribution and associated circuits are powered down. 0: Normal mode 1: Sleep mode **Bit [25] VRANGE** Write this bit to : 0 : If **PLL Output Clock Frequency >= 2.17 GHz.** 1 : If **PLL Output Clock Frequency < 2.17 GHz** **Bit [24] ENDIVCLK** Enable DIVCLK output : 0: Disables output of a divide-by-5 of PLL clock. 1 : Enables output of a divide-by-5 of PLL clock. **Bit [23:17] MPY** Select PLL Multiply factors between 4 and 25. MPY bits default value **(0b1100100)** can be used to enable a 25× multiplier considering a 100 MHz clock source at PLL input. 0b0010000: multiply by 4 0b0010100: multiply by 5 0b0011000: multiply by 6 0b0100000: multiply by 8 0b0100001: multiply by 8.25 0b0101000: multiply by 10 0b0110000: multiply by 12 0b0110010: multiply by 12.5 0b0111100: multiply by 15 0b1000000: multiply by 16 0b1000010: multiply by 16.5 0b1010000: multiply by 20 0b1011000: multiply by 22 0b1100100: multiply by 25 (default) **Bit [16] ENPLL** Enables the PLL. After setting this bit, it is necessary to allow 350 ns for the regulator to stabilize. Thereafter, the PLL will take no longer than 200 cycles (of **refclkp / n**) to lock to the required frequency, provided refclkp / n are stable. The LOCK bit of PCIE_STSPLL bitfield will be driven high by the digital lock detector. 0: Disable PLL. The PLL is fully powered down. 1: Enable PLL
15-12	Reserved	0	Reserved. Reads returns 0.

Table 1-227. PCIE Configuration Register (PCIE_CFG) Field Descriptions (continued)

Bit	Field	Value	Description
11-8	PCIE_STSPLL	0-Fh	PCIe PLL Status. **Bits [11:10]** are Reserved and always read as 0. **Bit[9] DIVCLK** : This bit indicates if divided-by-5 PLL clock is running. DIVCLK bit is held low (divided clock is NOT running) unless both the ENPLL and ENDIVCLK bits of the bitfield PCIE_CFGPLL are set high. Note that the "divide-by-5" clock is synchronous to the refclkp / n and NOT to the recovered recive clocks. Furthermore it will be subject to frequency overshoot of < 5% until the PLL has indicated lock in LOCK status bit. 0: DIVCLK is NOT running 1: DIVCLK is running **Bit[8] LOCK** : PLL Lock status bit. Driven high asynchronously between 2048 - 3071 **refclkp / n** cycles after the PLL has locked, which will occur within 200 **refclkp / n** cycles. Whilst LOCK is low, the PLL output frequency may overshoot by no more than <5 %, provided **MPY** is NOT changed to select a lower multiplicaiton factor. The same percentage overshoot will be mirrored by the bus clocks originating from the SerDes. 0: PLL has NOT locked 1 : PLL has locked
7-2	Reserved	0	Reserved. Reads returns 0.
1-0	PCIE_DEVTYPE		PCIe Module Device Type.
		0	Endpoint (EP) operation.
		1h	Legacy Endpoint operation.
		2h	Root Complex (RC) operation.
		3h	Reserved.

1.16.1.3.16 Clock Control Register (CLK_CTL)

The CLKIN oscillator and the clock source of the CLKOUT output are controlled by the CLKCTL register. The Clock Control Register (CLK_CTL) is shown in Figure 1-178 and described in Table 1-228.

Figure 1-178. Clock Control Register (CLK_CTL)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		DEVRSELECT	DEVSW2	DEVSW1	Reserved
R-0		R/W-0	R-1	R-1	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-228. Clock Control Register (CLK_CTL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. Reads returns 0.
3	DEVRSELECT		Device (27 MHz) Oscillator Bias Resistor.
			Controls internal bias resistor connection.
		0	Internal bias resistor connected (normal operation).
		1	Internal bias resistor disconnected (external bias resistor required or clock input used).
2	DEVSW2		Device Oscillator Frequency Selection.
		0	5-20 MHz
		1h	20-30 MHz
		2h	30-40 MHz
		3h	40-50 MHz
1	DEVSW1		Device Oscillator Frequency Selection.
		0	5-20 MHz
		1h	20-30 MHz
		2h	30-40 MHz
		3h	40-50 MHz
0	Reserved	0	Reserved. Reads returns 0.

1.16.1.3.17 Audio Interface Control Register (AUD_CTRL)

The AUD_CTRL Register controls features of the McASP and McBSP serial audio interfaces. It determines how the AMUTEIN input of McASP1 and McAPS2 will be driven. This allows all McASPs to share a common hardware mute input (MCA0_AMUTEIN) or have unique mute input controls. McASP0 always uses MCA0_AMUTEIN. The register also controls loopback of McBSP data for testing purposes.

The Audio Interface Control Register (AUD_CTRL) is shown in Figure 1-179 and described in Table 1-229.

Figure 1-179. Audio Interface Control Register (AUD_CTRL)

31	18	17	16
Reserved		MCB_LBFSX	MCB_LBCLKX
R-0		R/W-0	R/W-0

15	3	2	1	0
Reserved		ASP2MUTESRC	ASP1MUTESRC	Reserved
R-0		R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-229. Audio Interface Control Register (AUD_CTRL) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved. Reads returns 0.
17	MCB_LBFSX		McBSP FSX Loopback enable – Drives McBSP PIALBCTRLRX[1].
		0	No loopback.
		1	Enable loopback.
16	MCB_LBCLKX		McBSP CLKX Loopback enable – Drives McBSP PIALBCTRLRX[0].
		0	No loopback.
		1	Enable loopback.
15-3	Reserved	0	Reserved. Reads returns 0.
2	ASP2MUTESRC		McASP2 AMUTEIN source select.
		0	Use MCA2_AMUTEIN
		1	Use MCA0_AMUTEIN
1	ASP1MUTESRC		McASP1 AMUTEIN source select.
		0	Use MCA1_AMUTEIN.
		1	Use MCA0_AMUTEIN.
0	Reserved	0	Reserved. Reads returns 0.

1.16.1.3.18 DSP L2 Memory Sleep Mode Register (DSPMEM_SLEEP)

The DSPMEM_SLEEP Register determines the sleep behavior of the DSP L2 memories. The DSP L2 sleep mode may only be modified by Supervisor access.

The DSP L2 Memory Sleep Mode Register (DSPMEM_SLEEP) is shown in Figure 1-180 and described in Table 1-230.

Figure 1-180. DSP L2 Memory Sleep Mode Register (DSPMEM_SLEEP)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		DSPMEM_SD	DSPMEM_DS	DSPMEM_LS
R-0		R/W-1	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-230. DSP L2 Memory Sleep Mode Register (DSPMEM_SLEEP) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. Reads returns 0.
2	DSPMEM_SD	1-0	DSP L2 Memory in Shutdown Mode. Memory contents are lost.
1	DSPMEM_DS	1-0	DSP L2 Memory in Deep Sleep Mode. Memory contents are preserved.
0	DSPMEM_LS	1-0	DSP L2 Memory in Light Sleep Mode. Memory contents are preserved.

1.16.1.3.19 On-Chip Memory Sleep Mode Register (OCMEM_SLEEP)

The OCMEM_SLEEP Register determines the sleep behavior of the on-chip memories.

The On-Chip Memory Sleep Mode Register (OCMEM_SLEEP) is shown in Figure 1-181 and described in Table 1-231.

Figure 1-181. On-Chip Memory Sleep Mode Register (OCMEM_SLEEP)

31							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
Reserved	OCM1_SD	OCM1_DS	OCM1_LS	Reserved	OCM0_SD	OCM0_DS	OCM0_LS
R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-231. On-Chip Memory Sleep Mode Register (OCMEM_SLEEP) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved. Reads returns 0.
6	OCM1_SD	1-0	OCM1 Memory in Shutdown Mode. Memory contents are lost.
5	OCM1_DS	1-0	OCM1 Memory in Deep Sleep Mode. Memory contents are preserved.
4	OCM1_LS	1-0	OCM1 Memory in Light Sleep Mode. Memory contents are preserved.
3	Reserved	0	Reserved. Reads returns 0.
2	OCM0_SD	1-0	OCM0 Memory in Shutdown Mode. Memory contents are lost.
1	OCM0_DS	1-0	OCM0 Memory in Deep Sleep Mode. Memory contents are preserved.
0	OCM0_LS	1-0	OCM0 Memory in Light Sleep Mode. Memory contents are preserved.

1.16.1.3.20 HD DAC Control Register (HD_DAC_CTRL)

The HD_DAC_CTRL Register controls operation of the High Definition video DACs.

The HD DAC Control Register (HD_DAC_CTRL) is shown in Figure 1-182 and described in Table 1-232.

Figure 1-182. HD DAC Control Register (HD_DAC_CTRL)

31	4	3	2	1	0
Reserved		HD_CALSEL	HD_MIDRND		RESET_HD
R-0		R/W-0	R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-232. HD DAC Control Register (HD_DAC_CTRL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. Reads returns 0.
3	HD_CALSEL		HD DAC Calibration Select. Controls the video path mux between the Display Subsystem HD output and calibration register values.
		0	Normal operation (DSS output).
		1	Output calibration value.
2-1	HD_MIDRND		HD DAC MID randomizer mode.
		0	Bypass mode.
		1h	Swap 1.
		2h	Swap 2.
		3h	Bit shift.
0	RESET_HD	1-0	Reset HD DACs

1.16.1.3.21 HD DAC A Calibration Register (HD_DACA_CAL)

The HD_DACA_CAL Register is used to calibrate the output level of HD DAC A (HD Red/Y). Users can use this register to drive a known value to the DAC input for performing system video level calibration.

The HD DAC A Calibration Register (HD_DACA_CAL) is shown in Figure 1-183 and described in Table 1-233.

Figure 1-183. HD DAC A Calibration Register (HD_DACA_CAL)

31	12	11	0
Reserved		HDDAC_A_CAL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-233. HD DAC A Calibration Register (HD_DACA_CAL) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved. Reads returns 0.
11-0	HDDAC_A_CAL	0-FFFh	HD DAC A calibration value. This is driven as a constant value to the DAC when HD_CALSEL in the HD_DAC_CTRL register is set to 1.

1.16.1.3.22 HD DAC B Calibration Register (HD_DACB_CAL)

The HD_DACB_CAL Register is used to calibrate the output level of HD DAC B (HD Green/Pr). Users can use this register to drive a known value to the DAC input for performing system video level calibration.

The HD DAC B Calibration Register (HD_DACB_CAL) is shown in Figure 1-184 and described in Table 1-234.

Figure 1-184. HD DAC B Calibration Register (HD_DACB_CAL)

31	12	11	0
Reserved		HDDAC_B_CAL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-234. HD DAC B Calibration Register (HD_DACB_CAL) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved. Reads returns 0.
11-0	HDDAC_B_CAL	0-FFFh	HD DAC B calibration value. This is driven as a constant value to the DAC when HD_CALSEL in the HD_DAC_CTRL register is set to 1.

1.16.1.3.23 HD DAC C Calibration Register (HD_DACC_CAL)

The HD_DACC_CAL Register is used to calibrate the output level of HD DAC C (HD Blue/Pb). Users can use this register to drive a known value to the DAC input for performing system video level calibration.

The HD DAC C Calibration Register (HD_DACC_CAL) is shown in Figure 1-185 and described in Table 1-235.

Figure 1-185. HD DAC C Calibration Register (HD_DACC_CAL)

31	12	11	0
Reserved		HDDAC_C_CAL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-235. HD DAC C Calibration Register (HD_DACC_CAL) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved. Reads returns 0.
11-0	HDDAC_C_CAL	0-FFFh	HD DAC C calibration value. This is driven as a constant value to the DAC when HD_CALSEL in the HD_DAC_CTRL register is set to 1.

1.16.1.3.24 SD DAC Control Register (SD_DAC_CTRL)

The SD_DAC_CTRL Register controls operation of the Standard Definition video DACs.

The SD DAC Control Register (SD_DAC_CTRL) is shown in Figure 1-186 and described in Table 1-236.

Figure 1-186. SD DAC Control Register (SD_DAC_CTRL)

31	4	3	2	1	0
Reserved		SD_CALSEL	SD_MIDRND		RESET_SD
R-0		R/W-0	R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-236. SD DAC Control Register (SD_DAC_CTRL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. Reads returns 0.
3	SD_CALSEL		SD DAC Calibration Select. Controls the video path mux between the Display Subsystem SD output and calibration register values.
		0	Normal operation (DSS output).
		1	Output calibration value.
2-1	SD_MIDRND		SD DAC MID randomizer mode.
		0	Bypass mode.
		1h	Swap 1.
		2h	Swap 2.
		3h	Bit shift.
0	RESET_SD	1-0	Reset SD DACs

1.16.1.3.25 SD DAC A Calibration Register (SD_DACA_CAL)

The SD_DACA_CAL Register is used to calibrate the output level of SD DAC A (SD Red/Y). Users can use this register to drive a known value to the DAC input for performing system video level calibration.

The SD DAC A Calibration Register (SD_DACA_CAL) is shown in Figure 1-187 and described in Table 1-237.

Figure 1-187. SD DAC A Calibration Register (SD_DACA_CAL)

31	10	9	0
Reserved		SDDAC_A_CAL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-237. SD DAC A Calibration Register (SD_DACA_CAL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved. Reads returns 0.
9-0	SDDAC_A_CAL	0-3FFh	SD DAC A calibration value. This is driven as a constant value to the DAC when SD_CALSEL in the SD_DAC_CTRL register is set to 1.

1.16.1.3.26 SD DAC B Calibration Register (SD_DACB_CAL)

The SD_DACB_CAL Register is used to calibrate the output level of SD DAC B (SD Green/C). Users can use this register to drive a known value to the DAC input for performing system video level calibration.

The SD DAC B Calibration Register (SD_DACB_CAL) is shown in Figure 1-188 and described in Table 1-238.

Figure 1-188. SD DAC B Calibration Register (SD_DACB_CAL)

31	10	9	0
Reserved		SDDAC_B_CAL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-238. SD DAC B Calibration Register (SD_DACB_CAL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved. Reads returns 0.
9-0	SDDAC_B_CAL	0-3FFh	SD DAC B calibration value. This is driven as a constant value to the DAC when SD_CALSEL in the SD_DAC_CTRL register is set to 1.

1.16.1.3.27 SD DAC C Calibration Register (SD_DACC_CAL)

The SD_DACC_CAL Register is used to calibrate the output level of SD DAC C (SD Blue). Users can use this register to drive a known value to the DAC input for performing system video level calibration.

The SD DAC C Calibration Register (SD_DACC_CAL) is shown in Figure 1-189 and described in Table 1-239.

Figure 1-189. SD DAC C Calibration Register (SD_DACC_CAL)

31	10	9	0
Reserved		SDDAC_C_CAL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-239. SD DAC C Calibration Register (SD_DACC_CAL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved. Reads returns 0.
9-0	SDDAC_C_CAL	0-3FFh	SD DAC C calibration value. This is driven as a constant value to the DAC when SD_CALSEL in the SD_DAC_CTRL register is set to 1.

1.16.1.3.28 SD DAC D Calibration Register (SD_DACD_CAL)

The SD_DACD_CAL Register is used to calibrate the output level of SD DAC D (SD CVBS). Users can use this register to drive a known value to the DAC input for performing system video level calibration.

The SD DAC D Calibration Register (SD_DACD_CAL) is shown in Figure 1-190 and described in Table 1-240.

Figure 1-190. SD DAC D Calibration Register (SD_DACD_CAL)

31	10	9	0
Reserved		SDDAC_D_CAL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-240. SD DAC D Calibration Register (SD_DACD_CAL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved. Reads returns 0.
9-0	SDDAC_D_CAL	0-3FFh	SD DAC D calibration value. This is driven as a constant value to the DAC when SD_CALSEL in the SD_DAC_CTRL register is set to 1.

1.16.1.3.29 HW Event Select (Group 1) Register (HW_EVT_SEL_GRP1)

The HW_EVT_SEL_GRP1 Register selects 4 of 64 events from event group 1 for system trace.

The HW Event Select (Group 1) Register (HW_EVT_SEL_GRP1) is shown in Figure 1-191 and described in Table 1-241.

Figure 1-191. HW Event Select (Group 1) Register (HW_EVT_SEL_GRP1)

31	30	29	24	23	22	21	16
Reserved		EVENT4		Reserved		EVENT3	
R-0		R/W-0		R-0		R/W-0	

15	14	13	8	7	6	5	0
Reserved		EVENT2		Reserved		EVENT1	
R-0		R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-241. HW Event Select (Group 1) Register (HW_EVT_SEL_GRP1) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved. Reads returns 0.
29-24	EVENT4	0-3Fh	Select 4th trace event from group 1.
23-22	Reserved	0	Reserved. Reads returns 0.
21-16	EVENT3	0-3Fh	Select 3rd trace event from group 1.
15-14	Reserved	0	Reserved. Reads returns 0.
13-8	EVENT2	0-3Fh	Select 2nd trace event from group 1.
7-6	Reserved	0	Reserved. Reads returns 0.
5-0	EVENT1	0-3Fh	Select 1st trace event from group 1.

1.16.1.3.30 HW Event Select (Group 2) Register (HW_EVT_SEL_GRP2)

The HW_EVT_SEL_GRP2 Register selects 4 of 64 events from event group 2 for system trace.

The HW Event Select (Group 2) Register (HW_EVT_SEL_GRP2) is shown in Figure 1-192 and described in Table 1-242.

Figure 1-192. HW Event Select (Group 2) Register (HW_EVT_SEL_GRP2)

31	30	29	24	23	22	21	16
Reserved		EVENT4		Reserved		EVENT3	
R-0		R/W-0		R-0		R/W-0	

15	14	13	8	7	6	5	0
Reserved		EVENT2		Reserved		EVENT1	
R-0		R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-242. HW Event Select (Group 2) Register (HW_EVT_SEL_GRP2) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved. Reads returns 0.
29-24	EVENT4	0-3Fh	Select 4th trace event from group 2.
23-22	Reserved	0	Reserved. Reads returns 0.
21-16	EVENT3	0-3Fh	Select 3rd trace event from group 2.
15-14	Reserved	0	Reserved. Reads returns 0.
13-8	EVENT2	0-3Fh	Select 2nd trace event from group 2.
7-6	Reserved	0	Reserved. Reads returns 0.
5-0	EVENT1	0-3Fh	Select 1st trace event from group 2.

1.16.1.3.31 HW Event Select (Group 3) Register (HW_EVT_SEL_GRP3)

The HW_EVT_SEL_GRP3 Register selects 4 of 64 events from event group 3 for system trace.

The HW Event Select (Group 3) Register (HW_EVT_SEL_GRP3) is shown in Figure 1-193 and described in Table 1-243.

Figure 1-193. HW Event Select (Group 3) Register (HW_EVT_SEL_GRP3)

31	30	29	24	23	22	21	16
Reserved		EVENT4		Reserved		EVENT3	
R-0		R/W-0		R-0		R/W-0	

15	14	13	8	7	6	5	0
Reserved		EVENT2		Reserved		EVENT1	
R-0		R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-243. HW Event Select (Group 3) Register (HW_EVT_SEL_GRP3) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved. Reads returns 0.
29-24	EVENT4	0-3Fh	Select 4th trace event from group 3.
23-22	Reserved	0	Reserved. Reads returns 0.
21-16	EVENT3	0-3Fh	Select 3rd trace event from group 3.
15-14	Reserved	0	Reserved. Reads returns 0.
13-8	EVENT2	0-3Fh	Select 2nd trace event from group 3.
7-6	Reserved	0	Reserved. Reads returns 0.
5-0	EVENT1	0-3Fh	Select 1st trace event from group 3.

1.16.1.3.32 HW Event Select (Group 4) Register (HW_EVT_SEL_GRP4)

The HW_EVT_SEL_GRP4 Register selects 4 of 64 events from event group 4 for system trace.

The HW Event Select (Group 4) Register (HW_EVT_SEL_GRP4) is shown in Figure 1-194 and described in Table 1-244.

Figure 1-194. HW Event Select (Group 4) Register (HW_EVT_SEL_GRP4)

31	30	29	24	23	22	21	16
Reserved		EVENT4		Reserved		EVENT3	
R-0		R/W-0		R-0		R/W-0	

15	14	13	8	7	6	5	0
Reserved		EVENT2		Reserved		EVENT1	
R-0		R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-244. HW Event Select (Group 4) Register (HW_EVT_SEL_GRP4) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved. Reads returns 0.
29-24	EVENT4	0-3Fh	Select 4th trace event from group 4.
23-22	Reserved	0	Reserved. Reads returns 0.
21-16	EVENT3	0-3Fh	Select 3rd trace event from group 4.
15-14	Reserved	0	Reserved. Reads returns 0.
13-8	EVENT2	0-3Fh	Select 2nd trace event from group 4.
7-6	Reserved	0	Reserved. Reads returns 0.
5-0	EVENT1	0-3Fh	Select 1st trace event from group 4.

1.16.1.3.33 HDMI Observe Clock Control (HDMI_OBSCLK_CTRL)

The HDMI_OBSCLK_CTRL Register controls operation of the HDMI clock observation LVDS buffer and bandgap reference.

The HDMI Observe Clock Control (HDMI_OBSCLK_CTRL) is shown in Figure 1-195 and described in Table 1-245.

Figure 1-195. HDMI Observe Clock Control (HDMI_OBSCLK_CTRL)

31						10	9	8
Reserved							SUBLVDS_EN	BIAS_TRIM1
R-0							R/W-0	R/W-eFuse
7	6	5	4	3	2	1	0	
BIAS_TRIM0	BIAS_TRIMEN	BIAS_EFUSESET	BIAS_PWRDN	PWRDN	LPSEL	LOPWRB	LOPWRA	
R/W-eFuse	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-245. HDMI Observe Clock Control (HDMI_OBSCLK_CTRL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved. Reads returns 0.
9	SUBLVDS_EN		Sub-LVDS Mode Enable. Allows LVDS transmitter to operate in sub-LVDS mode.
		0	Common mode voltage is 1.2 V
		1	Common mode voltage is 0.9 V
8	BIAS_TRIM1	1-0	Badgap Reference Trimming bits.
7	BIAS_TRIM0		The BIAS_TRIM bits allow the Bandgap voltage to be adjusted when BIAS_TRIMEN is high.
		00	1.2 V – 3%
		01	1.2V
		10	1.2V + 3%
		11	1.2V – 1%
6	BIAS_TRIMEN		Bandgap Bias eFuse Trim Enable.
		0	Trimming is disabled.
		1	Trimming is enabled.
5	BIAS_EFUSESET		Bandgap Reference eFuse Set.
		0	Trimming is enabled by BIAS_TRIMEN.
		1	Trimming is disabled (1.2V NOM output).
4	BIAS_PWRDN		Bandgap Reference Powerdown.
		0	Bias voltage outputs are valid.
		1	Bias voltage outputs are at Vss.
3	PWRDN		Buffer Powerdown
		0	PAD and $\overline{\text{PAD}}$ outputs are driven.
		1	PAD and$\overline{\text{PAD}}$ outputs are low.
2	LPSEL		LPSEL selects internal termination between PAD and $\overline{\text{PAD}}$ of 100 Ω when low and 200 Ω when high to enable low power operation. When used in conjunction with LOPWRA and LOPWRB, the output current is selected as follows:
1	LOPWRB		LPSEL = 0 / LPSEL = 1
			11 - 4.0 mA / 11 - 4.0 mA
0	LOPWRA		10 - 3.5 mA / 10 - 3.33 mA
			01 - 3.0 mA / 01 - 2.66 mA
			00 - 2.5 mA / 00 - 2.00 mA

1.16.1.3.34 Serdes Control Register (SERDES_CTRL)

The SERDES_CTRL Register controls the SERDES reference clock input buffer and distribution.

The Serdes Control Register (SERDES_CTRL) is shown in Figure 1-196 and described in Table 1-246.

Figure 1-196. Serdes Control Register (SERDES_CTRL)

31	2	1	0
Reserved		SERDES_PWRDN	RCD_PWRDN
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-246. Serdes Control Register (SERDES_CTRL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Reads returns 0.
1	SERDES_PWRDN	1-0	Controls Serdes reference clock input buffer powerdown.
0	RCD_PWRDN	1-0	Controls Serdes Reference Clock Distribution powerdown.

1.16.1.3.35 USB Clock Control Register (USB_CLK_CTL)

The USB oscillator is controlled by the USB_CLK_CTL register.

The USB Clock Control Register (USB_CLK_CTL) is shown in Figure 1-197 and described in Table 1-247.

Figure 1-197. USB Clock Control Register (USB_CLK_CTL)

31	2	1	0
Reserved		(USBSW2,USBSW1)	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-247. USB Clock Control Register (USB_CLK_CTL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Reads returns 0.
1-0	(USBSW2,USBSW1)		USB Oscillator Frequency Selection.
		0	5-20 MHz
		1h	20-30 MHz
		2h	30-40 MHz
		3h	40-50 MHz

1.16.1.3.36 PLL Observe Clock Control Register (PLL_OBSCLK_CTRL)

The PLL_OBSCLK_CTRL Register controls the PLL Observe clock output.

The PLL Observe Clock Control Register (PLL_OBSCLK_CTRL) is shown in Figure 1-198 and described in Table 1-248.

Figure 1-198. PLL Observe Clock Control Register (PLL_OBSCLK_CTRL)

31	2	1	0
Reserved		CML_PWRDN	RCD_PWRDN
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-248. PLL Observe Clock Control Register (PLL_OBSCLK_CTRL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Reads returns 0.
1	CML_PWRDN	1-0	PLL Observe clock CML driver powerdown.
0	RCD_PWRDN	1-0	PLL Observe clock Reference Clock Distribution powerdown.

1.16.1.3.37 DDR RCD Register (DDR_RCD)

The DDR_RCD Register controls power to RCD.

The DDR RCD Register (DDR_RCD) is shown in Figure 1-199 and described in Table 1-249.

Figure 1-199. DDR RCD Register (DDR_RCD)

31	1	0
Reserved		PWRDN
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-249. DDR RCD Register (DDR_RCD) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reads returns 0.
0	PWRDN		Power enable/disable for RCD.
		0	Power down.
		1	Normal mode.

1.17 Pin Multiplexing Control

Device-level pin multiplexing is controlled on a pin-by-pin basis by the MUXMODE bits of the PINCTRL1-PINCTRL321 registers in the SYSCFG module. The default state for each multiplexed pin is MUXMODE = 000.

Pin multiplexing selects which of several peripheral pin functions control the pin's IO buffer output data values. The input from each pin is routed to all of the peripherals that share the pin, regardless of the MUXMODE setting.

The PINCTRL*n* register is shown in Figure 1-200 and described in Table 1-250.

Figure 1-200. PINCTRL*n* Register

31	16
Reserved	
R-0	

15	5	4	3	2	0
Reserved		PULLTYPESEL	PULLDIS	MUXMODE	
R-0		R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-250. PINCTRL*n* Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved. Reads returns 0.
4	PULLTYPESEL		Pad Pullup/Pulldown Type Selection
		0	Pulldown is selected
		1	Pullup is selected
3	PULLDIS		Pad Pullup/Pulldown Disable
		0	Pullup/Pulldown is enabled
		1	Pullup/Pulldown is disabled
2-0	MUXMODE	0-3h	Pad Functional Signal Mux Select. See Table 1-251.

Table 1-251. PINCTRL*n* Register Pin Multiplexing

Address	Register	PULLTYPESEL	PULLDIS	MUXMODE[2:0]			
				000	001	010	011
4814 0800h	PINCTRL1	0	0				
4814 0804h	PINCTRL2	0	0				
4814 0808h	PINCTRL3	0	0				
4814 080Ch	PINCTRL4	0	0				
4814 0810h	PINCTRL5	0	0				
4814 0814h	PINCTRL6	0	0		VOUT[1]_C[3]	VIN[1]A_D[9]	
4814 0818h	PINCTRL7	0	0		VOUT[1]_C[4]	VIN[1]A_D[10]	
4814 081Ch	PINCTRL8	0	0		VOUT[1]_C[5]	VIN[1]A_D[11]	
4814 0820h	PINCTRL9	0	0		VOUT[1]_C[6]	VIN[1]A_D[12]	
4814 0824h	PINCTRL10	0	0		VOUT[1]_C[7]	VIN[1]A_D[13]	
4814 0828h	PINCTRL11	0	0		VIN[1]A_D[14]		
4814 082Ch	PINCTRL12	0	0		VIN[0]A_D[20]	VIN[0]B_DE	
4814 0830h	PINCTRL13	0	0		VIN[0]A_D[21]	VIN[0]B_FLD	
4814 0834h	PINCTRL14	0	0		VIN[0]A_D[22]	VIN[0]B_VSYNC	
4814 0838h	PINCTRL15	0	0		VIN[0]A_D[23]	VIN[0]B_HSYNC	

Table 1-251. PINCTRL*n* Register Pin Multiplexing (continued)

Address	Register	PULLTYPESEL	PULLDIS	MUXMODE[2:0]			
				000	001	010	011
4814 083Ch	PINCTRL16	0	0		VOUT[1]_Y_YC[6]	VIN[1]A_D[4]	
4814 0840h	PINCTRL17	0	0		VOUT[1]_Y_YC[7]	VIN[1]A_D[5]	
4814 0844h	PINCTRL18	0	0		VOUT[1]_Y_YC[8]	VIN[1]A_D[6]	
4814 0848h	PINCTRL19	0	0		VOUT[1]_Y_YC[9]	VIN[1]A_D[7]	
4814 084Ch	PINCTRL20	0	0		VOUT[1]_C[2]	VIN[1]A_D[8]	
4814 0850h	PINCTRL21	0	0		VOUT[1]_HSYNC (silicon revision 1.x) DAC_VOUT[1]_HSYNC (silicon revision 2.x)	VIN[1]A_D[15]	
4814 0854h	PINCTRL22	0	0		VIN[0]A_D[16]	VIN[1]A_HSYNC	VOUT[1]_FLD
4814 0858h	PINCTRL23	0	0		VIN[0]A_D[17]	VIN[1]A_VSYNC	VOUT[1]_VSYNC (silicon revision 1.x) DAC_VOUT[1]_VSYNC (silicon revision 2.x)
4814 085Ch	PINCTRL24	0	0		VIN[0]A_D[18]	VIN[1]A_FLD	VOUT[1]_C[8]
4814 0860h	PINCTRL25	0	0		VIN[0]A_D[19]	VIN[1]A_DE	VOUT[1]_C[9]
4814 0864h	PINCTRL26	0	0		VOUT[0]_R_CR[0]	VOUT[1]_C[8]	VOUT[1]_CLK
4814 0868h	PINCTRL27	0	0		VOUT[0]_B_CB_C[0]	VOUT[1]_C[9]	VIN[1]B_HSYNC_DE
4814 086Ch	PINCTRL28	0	0		VOUT[0]_B_CB_C[1]	VOUT[1]_HSYNC (silicon revision 1.x) DAC_VOUT[1]_HSYNC (silicon revision 2.x)	VOUT[1]_AVID
4814 0870h	PINCTRL29	0	0		VOUT[0]_G_Y_YC[0]	VOUT[1]_VSYNC (silicon revision 1.x) DAC_VOUT[1]_VSYNC (silicon revision 2.x)	VIN[1]B_VSYNC
4814 0874h	PINCTRL30	0	0		VOUT[0]_G_Y_YC[1]	VOUT[1]_FLD	VIN[1]B_FLD
4814 0878h	PINCTRL31	0	0		VOUT[1]_AVID	VIN[1]B_CLK	
4814 087Ch	PINCTRL32	0	0		VIN[0]A_HSYNC		
4814 0880h	PINCTRL33	0	0		VIN[0]A_VSYNC		
4814 0884h	PINCTRL34	0	0		VIN[0]A_FLD		
4814 0888h	PINCTRL35	0	0		VIN[0]A_DE		
4814 088Ch	PINCTRL36	0	0		VOUT[0]_HSYNC		
4814 0890h	PINCTRL37	0	0		VOUT[0]_VSYNC		
4814 0894h	PINCTRL38	0	0		VOUT[0]_FLD (silicon revision 1.x) DAC_VSYNC_VOUT[0]_FLD (silicon revision 2.x)		
4814 0898h	PINCTRL39	0	0		VOUT[0]_AVID (silicon revision 1.x) DAC_HSYNC_VOUT[0]_AVID (silicon revision 2.x)		
4814 089Ch	PINCTRL40	0	0		VOUT[0]_R_CR[1]		
4814 08A0h	PINCTRL41	0	1				
4814 08A4h	PINCTRL42	0	1				
4814 08A8h	PINCTRL43	0	1				
4814 08ACh	PINCTRL44	0	1				
4814 08B0h	PINCTRL45	0	1				
4814 08B4h	PINCTRL46	0	1		VOUT[1]_CLK	VIN[1]A_CLK	
4814 08B8h	PINCTRL47	0	1		VOUT[1]_Y_YC[2]	VIN[1]A_D[0]	

Table 1-251. PINCTRL*n* Register Pin Multiplexing (continued)

Address	Register	PULLTYPESEL	PULLDIS	MUXMODE[2:0]			
				000	001	010	011
4814 08BCh	PINCTRL48	0	1		VOUT[1]_Y_YC[3]	VIN[1]A_D[1]	
4814 08C0h	PINCTRL49	0	1		VOUT[1]_Y_YC[4]	VIN[1]A_D[2]	
4814 08C4h	PINCTRL50	0	1		VOUT[1]_Y_YC[5]	VIN[1]A_D[3]	
4814 08C8h	PINCTRL51	0	0		EMAC[1]_RXCLK		
4814 08CCh	PINCTRL52	0	0		EMAC[1]_RXD[0]		
4814 08D0h	PINCTRL53	0	0		EMAC[1]_RXD[1]		
4814 08D4h	PINCTRL54	0	0		EMAC[1]_RXD[2]		
4814 08D8h	PINCTRL55	0	0		EMAC[1]_RXD[3]		
4814 08DCh	PINCTRL56	0	0		EMAC[1]_RXD[4]		
4814 08E0h	PINCTRL57	0	0		EMAC[1]_RXD[5]		
4814 08E4h	PINCTRL58	0	0		EMAC[1]_RXD[6]		
4814 08E8h	PINCTRL59	0	0		EMAC[1]_RXD[7]		
4814 08ECh	PINCTRL60	0	0		EMAC[1]_RXDV		
4814 08F0h	PINCTRL61	0	1		EMAC[1]_GMTCLK		
4814 08F4h	PINCTRL62	0	1		EMAC[1]_TXD[0]		
4814 08F8h	PINCTRL63	0	1		EMAC[1]_TXD[1]		
4814 08FCh	PINCTRL64	0	1		EMAC[1]_TXD[2]		
4814 0900h	PINCTRL65	0	1		EMAC[1]_TXD[3]		
4814 0904h	PINCTRL66	0	1		EMAC[1]_TXD[4]		
4814 0908h	PINCTRL67	0	1		EMAC[1]_TXD[5]		
4814 090Ch	PINCTRL68	0	1		EMAC[1]_TXD[6]		
4814 0910h	PINCTRL69	0	1		EMAC[1]_TXD[7]		
4814 0914h	PINCTRL70	0	1		EMAC[1]_TXEN		
4814 0918h	PINCTRL71	0	1		EMAC[1]_TXCLK		
4814 091Ch	PINCTRL72	0	1		EMAC[1]_COL		
4814 0920h	PINCTRL73	0	0		EMAC[1]_CRS		
4814 0924h	PINCTRL74	0	1		EMAC[1]_RXER		
4814 0928h	PINCTRL75	0	0				
4814 092Ch	PINCTRL76	0	0				
4814 0930h	PINCTRL77	0	0				
4814 0934h	PINCTRL78	0	0				
4814 0938h	PINCTRL79	0	0				
4814 093Ch	PINCTRL80	0	1				
4814 0940h	PINCTRL81	0	1				
4814 0944h	PINCTRL82	0	1				
4814 0948h	PINCTRL83	0	0	VIN[0]A_CLK			
4814 094Ch	PINCTRL84	0	0	VIN[0]B_CLK			
4814 0950h	PINCTRL85	0	0	VIN[0]A_D[0]			
4814 0954h	PINCTRL86	0	0	VIN[0]A_D[1]			
4814 0958h	PINCTRL87	0	0	VIN[0]A_D[2]			
4814 095Ch	PINCTRL88	0	0	VIN[0]A_D[3]			
4814 0960h	PINCTRL89	0	0	VIN[0]A_D[4]			
4814 0964h	PINCTRL90	0	0	VIN[0]A_D[5]			
4814 0968h	PINCTRL91	0	0	VIN[0]A_D[6]			
4814 096Ch	PINCTRL92	0	0	VIN[0]A_D[7]			
4814 0970h	PINCTRL93	0	0	VIN[0]A_D[8]			
4814 0974h	PINCTRL94	0	0	VIN[0]A_D[9]			
4814 0978h	PINCTRL95	0	0	VIN[0]A_D[10]			
4814 097Ch	PINCTRL96	0	0	VIN[0]A_D[11]			
4814 0980h	PINCTRL97	0	0	VIN[0]A_D[12]			

Table 1-251. PINCTRL*n* Register Pin Multiplexing (continued)

Address	Register	PULLTYPESEL	PULLDIS	MUXMODE[2:0]			
				000	001	010	011
4814 0984h	PINCTRL98	0	0	VIN[0]A_D[13]			
4814 0988h	PINCTRL99	0	0	VIN[0]A_D[14]			
4814 098Ch	PINCTRL100	0	0	VIN[0]A_D[15]			
4814 0990h	PINCTRL101	0	1	VOUT[0]_CLK			
4814 0994h	PINCTRL102	0	1	VOUT[0]_G_Y_YC[2]			
4814 0998h	PINCTRL103	0	1	VOUT[0]_G_Y_YC[3]			
4814 099Ch	PINCTRL104	0	1	VOUT[0]_G_Y_YC[4]			
4814 09A0h	PINCTRL105	0	1	VOUT[0]_G_Y_YC[5]			
4814 09A4h	PINCTRL106	0	1	VOUT[0]_G_Y_YC[6]			
4814 09A8h	PINCTRL107	0	1	VOUT[0]_G_Y_YC[7]			
4814 09ACh	PINCTRL108	0	1	VOUT[0]_G_Y_YC[8]			
4814 09B0h	PINCTRL109	0	1	VOUT[0]_G_Y_YC[9]			
4814 09B4h	PINCTRL110	0	1	VOUT[0]_B_CB_C[2]			
4814 09B8h	PINCTRL111	0	1	VOUT[0]_B_CB_C[3]			
4814 09BCh	PINCTRL112	0	1	VOUT[0]_B_CB_C[4]			
4814 09C0h	PINCTRL113	0	1	VOUT[0]_B_CB_C[5]			
4814 09C4h	PINCTRL114	0	1	VOUT[0]_B_CB_C[6]			
4814 09C8h	PINCTRL115	0	1	VOUT[0]_B_CB_C[7]			
4814 09CCh	PINCTRL116	0	1	VOUT[0]_B_CB_C[8]			
4814 09D0h	PINCTRL117	0	1	VOUT[0]_B_CB_C[9]			
4814 09D4h	PINCTRL118	0	1	VOUT[0]_R_CR[2]	VOUT[0]_HSYNC	VOUT[1]_Y_YC[2]	
4814 09D8h	PINCTRL119	0	1	VOUT[0]_R_CR[3]	VOUT[0]_VSYNC	VOUT[1]_Y_YC[3]	
4814 09DCh	PINCTRL120	0	1	VOUT[0]_R_CR[4]	VOUT[0]_FLD	VOUT[1]_Y_YC[4]	
4814 09E0h	PINCTRL121	0	1	VOUT[0]_R_CR[5]	VOUT[0]_AVID	VOUT[1]_Y_YC[5]	
4814 09E4h	PINCTRL122	0	1	VOUT[0]_R_CR[6]	VOUT[0]_G_Y_YC[0]	VOUT[1]_Y_YC[6]	
4814 09E8h	PINCTRL123	0	1	VOUT[0]_R_CR[7]	VOUT[0]_G_Y_YC[1]	VOUT[1]_Y_YC[7]	
4814 09ECh	PINCTRL124	0	1	VOUT[0]_R_CR[8]	VOUT[0]_B_CB_C[0]	VOUT[1]_Y_YC[8]	
4814 09F0h	PINCTRL125	0	1	VOUT[0]_R_CR[9]	VOUT[0]_B_CB_C[1]	VOUT[1]_Y_YC[9]	
4814 09F4h	PINCTRL126	0	0	MCA[0]_ACLKR			
4814 09F8h	PINCTRL127	0	0	MCA[0]_AHCLKR			
4814 09FCh	PINCTRL128	0	0	MCA[0]_AFSR			
4814 0A00h	PINCTRL129	0	0	MCA[0]_ACLKX			
4814 0A04h	PINCTRL130	0	0	MCA[0]_ACLKHX			
4814 0A08h	PINCTRL131	0	0	MCA[0]_AFSX			
4814 0A0Ch	PINCTRL132	0	0	MCA[0]_AMUTE			
4814 0A10h	PINCTRL133	0	0	MCA[0]_AXR[0]			
4814 0A14h	PINCTRL134	0	0	MCA[0]_AXR[1]			
4814 0A18h	PINCTRL135	0	0	MCA[0]_AXR[2]	MCB_FSX		
4814 0A1Ch	PINCTRL136	0	0	MCA[0]_AXR[3]	MCB_FSR		
4814 0A20h	PINCTRL137	0	0	MCA[0]_AXR[4]	MCB_DX		
4814 0A24h	PINCTRL138	0	0	MCA[0]_AXR[5]	MCB_DR		
4814 0A28h	PINCTRL139	0	0	MCA[1]_ACLKR			
4814 0A2Ch	PINCTRL140	0	0	MCA[1]_AHCLKR			
4814 0A30h	PINCTRL141	0	0	MCA[1]_AFSR			
4814 0A34h	PINCTRL142	0	0	MCA[1]_ACLKX			
4814 0A38h	PINCTRL143	0	0	MCA[1]_ACLKHX			
4814 0A3Ch	PINCTRL144	0	0	MCA[1]_AFSX			
4814 0A40h	PINCTRL145	0	0	MCA[1]_AMUTE			
4814 0A44h	PINCTRL146	0	0	MCA[1]_AXR[0]			
4814 0A48h	PINCTRL147	0	0	MCA[1]_AXR[1]			

Table 1-251. PINCTRL*n* Register Pin Multiplexing (continued)

Address	Register	PULLTYPESEL	PULLDIS	MUXMODE[2:0]			
				000	001	010	011
4814 0A4Ch	PINCTRL148	0	0	MCA[2]_ACLKR	MCB_CLKR	MCB_DR	
4814 0A50h	PINCTRL149	0	0	MCA[2]_AHCLKR	MCB_CLKS		
4814 0A54h	PINCTRL150	0	0	MCA[2]_AFSR	MCB_CLKX	MCB_FSR	
4814 0A58h	PINCTRL151	0	0	MCA[2]_ACLKX	MCB_CLKX		
4814 0A5Ch	PINCTRL152	0	0	MCA[2]_ACLKHX	MCB_CLKR		
4814 0A60h	PINCTRL153	0	0	MCA[2]_AFSX	MCB_CLKS	MCB_FSX	
4814 0A64h	PINCTRL154	0	0	MCA[2]_AMUTE			
4814 0A68h	PINCTRL155	0	0	MCA[2]_AXR[0]			
4814 0A6Ch	PINCTRL156	0	0	MCA[2]_AXR[1]	MCB_DX		
4814 0A70h	PINCTRL157	0	1	SD_POW	GPMC_A[14]	GP1[0]	
4814 0A74h	PINCTRL158	0	1	SD_CLK	GPMC_A[13]	GP1[1]	
4814 0A78h	PINCTRL159	0	1	SD_CMD	GPMC_A[21]	GP1[2]	
4814 0A7Ch	PINCTRL160	0	0	SD_DAT[0]	GPMC_A[20]	GP1[3]	
4814 0A80h	PINCTRL161	0	0	SD_DAT[1]_SDIRQ	GPMC_A[19]	GP1[4]	
4814 0A84h	PINCTRL162	0	0	SD_DAT[2]_SDRW	GPMC_A[18]	GP1[5]	
4814 0A88h	PINCTRL163	0	0	SD_DAT[3]	GPMC_A[17]	GP1[6]	
4814 0A8Ch	PINCTRL164	0	0	SD_SDCD	GPMC_A[16]	GP1[7]	
4814 0A90h	PINCTRL165	0	0	SD_SDWP	GPMC_A[15]	GP1[8]	
4814 0A94h	PINCTRL166	0	0	SPI_SCLK			
4814 0A98h	PINCTRL167	1	0	SPI_SCS[0]			
4814 0A9Ch	PINCTRL168	1	0	SPI_SCS[1]	GPMC_A[23]		
4814 0AA0h	PINCTRL169	1	0	SPI_SCS[2]	GPMC_A[22]		
4814 0AA4h	PINCTRL170	1	0	SPI_SCS[3]	GPMC_A[21]	GP1[22]	
4814 0AA8h	PINCTRL171	0	0	SPI_D[0]			
4814 0AACh	PINCTRL172	0	0	SPI_D[1]			
4814 0AB0h	PINCTRL173	0	0	UART0_RXD			
4814 0AB4h	PINCTRL174	0	1	UART0_TXD			
4814 0AB8h	PINCTRL175	1	1	$\overline{\text{UART0\_RTS}}$	GP1[27]		
4814 0ABCh	PINCTRL176	1	0	$\overline{\text{UART0\_CTS}}$	GP1[28]		
4814 0AC0h	PINCTRL177	1	1	$\overline{\text{UART0\_DTR}}$	GPMC_A[20]	GPMC_A[12]	GP1[16]
4814 0AC4h	PINCTRL178	1	0	$\overline{\text{UART0\_DSR}}$	GPMC_A[19]	GPMC_A[24]	GP1[17]
4814 0AC8h	PINCTRL179	1	0	$\overline{\text{UART0\_DCD}}$	GPMC_A[18]	GPMC_A[23]	GP1[18]
4814 0ACCh	PINCTRL180	1	0	UART0_RIN	GPMC_A[17]	GPMC_A[22]	GP1[19]
4814 0AD0h	PINCTRL181	0	0	UART1_RXD	GPMC_A[26]	GPMC_A[20]	
4814 0AD4h	PINCTRL182	0	1	UART1_TXD	GPMC_A[25]	GPMC_A[19]	
4814 0AD8h	PINCTRL183	1	1	$\overline{\text{UART1\_RTS}}$	GPMC_A[14]	GPMC_A[18]	GP1[25]
4814 0ADCh	PINCTRL184	1	0	$\overline{\text{UART1\_CTS}}$	GPMC_A[13]	GPMC_A[17]	GP1[26]
4814 0AE0h	PINCTRL185	0	0	UART2_RXD			
4814 0AE4h	PINCTRL186	0	0	UART2_TXD			
4814 0AE8h	PINCTRL187	1	1	$\overline{\text{UART2\_RTS}}$	GPMC_A[15]	GPMC_A[26]	GP1[23]
4814 0AECh	PINCTRL188	1	0	$\overline{\text{UART2\_CTS}}$	GPMC_A[16]	GPMC_A[25]	GP1[24]
4814 0AF0h	PINCTRL189	0	0		GPMC_A[27]	GP1[9]	
4814 0AF4h	PINCTRL190	0	1		GPMC_A[22]	GP1[10]	
4814 0AF8h	PINCTRL191	0	1		GPMC_A[26]	GP1[11]	
4814 0AFCh	PINCTRL192	0	0		GPMC_A[25]	GP1[12]	
4814 0B00h	PINCTRL193	0	1		GP1[13]		
4814 0B04h	PINCTRL194	0	1		GPMC_A[23]	GP1[14]	
4814 0B08h	PINCTRL195	0	1		GPMC_A[24]	GP1[15]	
4814 0B0Ch	PINCTRL196	0	0		GPMC_A[16]	GP0[21]	
4814 0B10h	PINCTRL197	0	1		GPMC_A[15]	GP0[22]	

Table 1-251. PINCTRL*n* Register Pin Multiplexing (continued)

Address	Register	PULLTYPESEL	PULLDIS	MUXMODE[2:0]			
				000	001	010	011
4814 0B14h	PINCTRL198	0	1		GPMC_A[14]	GP0[23]	
4814 0B18h	PINCTRL199	0	0		GPMC_A[13]	GP0[24]	
4814 0B1Ch	PINCTRL200	0	1		GP0[25]		
4814 0B20h	PINCTRL201	0	1		GPMC_A[21]	GP0[26]	
4814 0B24h	PINCTRL202	0	1		GPMC_A[12]	GP0[27]	
4814 0B28h	PINCTRL203	0	0	TIM4_OUT	GP0[28]		
4814 0B2Ch	PINCTRL204	0	0	TIM5_OUT	GP0[29]		
4814 0B30h	PINCTRL205	0	0	TIM6_OUT	GPMC_A[24]	GP0[30]	
4814 0B34h	PINCTRL206	0	0	TIM7_OUT	GPMC_A[12]	GP0[31]	
4814 0B38h	PINCTRL207	1	0	GPMC_CS[0]			
4814 0B3Ch	PINCTRL208	1	0	GPMC_CS[1]			
4814 0B40h	PINCTRL209	1	0	GPMC_CS[2]			
4814 0B44h	PINCTRL210	1	0	GPMC_CS[3]			
4814 0B48h	PINCTRL211	1	0	GPMC_CS[4]	GP1[21]		
4814 0B4Ch	PINCTRL212	1	0	GPMC_CS[5]	GPMC_A[12]		
4814 0B50h	PINCTRL213	1	0	GPMC_WE			
4814 0B54h	PINCTRL214	1	1	GPMC_OE_RE			
4814 0B58h	PINCTRL215	0	1	GPMC_BE0_CLE			
4814 0B5Ch	PINCTRL216	0	1	GPMC_BE1			
4814 0B60h	PINCTRL217	0	1	GPMC_ADV_ALE			
4814 0B64h	PINCTRL218	0	1	GPMC_DIR	GP1[20]		
4814 0B68h	PINCTRL219	0	0	GPMC_WP			
4814 0B6Ch	PINCTRL220	0	0	GPMC_WAIT			
4814 0B70h	PINCTRL221	0	1	GPMC_A[0]	GP0[8]		
4814 0B74h	PINCTRL222	0	1	GPMC_A[1]	GP0[9]		
4814 0B78h	PINCTRL223	0	1	GPMC_A[2]	GP0[10]		
4814 0B7Ch	PINCTRL224	0	1	GPMC_A[3]	GP0[11]		
4814 0B80h	PINCTRL225	0	1	GPMC_A[4]	GP0[12]		
4814 0B84h	PINCTRL226	0	1	GPMC_A[5]	GP0[13]		
4814 0B88h	PINCTRL227	0	1	GPMC_A[6]	GP0[14]		
4814 0B8Ch	PINCTRL228	0	1	GPMC_A[7]	GP0[15]		
4814 0B90h	PINCTRL229	0	1	GPMC_A[8]	GP0[16]		
4814 0B94h	PINCTRL230	0	1	GPMC_A[9]	GP0[17]		
4814 0B98h	PINCTRL231	0	1	GPMC_A[10]	GP0[18]		
4814 0B9Ch	PINCTRL232	0	1	GPMC_A[11]	GP0[19]		
4814 0BA0h	PINCTRL233	0	1	GPMC_A[27]	GP0[20]		
4814 0BA4h	PINCTRL234	0	0	GPMC_D[0]			
4814 0BA8h	PINCTRL235	0	0	GPMC_D[1]			
4814 0BACh	PINCTRL236	0	0	GPMC_D[2]			
4814 0BB0h	PINCTRL237	0	0	GPMC_D[3]			
4814 0BB4h	PINCTRL238	0	0	GPMC_D[4]			
4814 0BB8h	PINCTRL239	0	0	GPMC_D[5]			
4814 0BBCh	PINCTRL240	0	0	GPMC_D[6]			
4814 0BC0h	PINCTRL241	0	0	GPMC_D[7]			
4814 0BC4h	PINCTRL242	0	0	GPMC_D[8]			
4814 0BC8h	PINCTRL243	0	0	GPMC_D[9]			
4814 0BCCh	PINCTRL244	0	0	GPMC_D[10]			
4814 0BD0h	PINCTRL245	0	0	GPMC_D[11]			
4814 0BD4h	PINCTRL246	0	0	GPMC_D[12]			
4814 0BD8h	PINCTRL247	0	0	GPMC_D[13]			

Table 1-251. PINCTRL*n* Register Pin Multiplexing (continued)

Address	Register	PULLTYPESEL	PULLDIS	MUXMODE[2:0]			
				000	001	010	011
4814 0BDCh	PINCTRL248	0	0	GPMC_D[14]			
4814 0BE0h	PINCTRL249	0	0	GPMC_D[15]			
4814 0BE4h	PINCTRL250	0	1	GPMC_CLK	GP1[29]		
4814 0BE8h	PINCTRL251	0	0	EMAC[0]_COL			
4814 0BECh	PINCTRL252	0	0	EMAC[0]_CRS			
4814 0BF0h	PINCTRL253	0	1	EMAC[0]_GMTCLK			
4814 0BF4h	PINCTRL254	1	0	EMAC[0]_RXCLK			
4814 0BF8h	PINCTRL255	1	0	EMAC[0]_RXD[0]			
4814 0BFCh	PINCTRL256	1	0	EMAC[0]_RXD[1]			
4814 0C00h	PINCTRL257	1	0	EMAC[0]_RXD[2]			
4814 0C04h	PINCTRL258	1	0	EMAC[0]_RXD[3]			
4814 0C08h	PINCTRL259	1	0	EMAC[0]_RXD[4]			
4814 0C0Ch	PINCTRL260	1	0	EMAC[0]_RXD[5]			
4814 0C10h	PINCTRL261	1	0	EMAC[0]_RXD[6]			
4814 0C14h	PINCTRL262	1	0	EMAC[0]_RXD[7]			
4814 0C18h	PINCTRL263	1	0	EMAC[0]_RXDV			
4814 0C1Ch	PINCTRL264	1	0	EMAC[0]_RXER			
4814 0C20h	PINCTRL265	0	1	EMAC[0]_TXCLK			
4814 0C24h	PINCTRL266	0	1	EMAC[0]_TXD[0]			
4814 0C28h	PINCTRL267	0	1	EMAC[0]_TXD[1]			
4814 0C2Ch	PINCTRL268	0	1	EMAC[0]_TXD[2]			
4814 0C30h	PINCTRL269	0	1	EMAC[0]_TXD[3]			
4814 0C34h	PINCTRL270	0	1	EMAC[0]_TXD[4]			
4814 0C38h	PINCTRL271	0	1	EMAC[0]_TXD[5]			
4814 0C3Ch	PINCTRL272	0	1	EMAC[0]_TXD[6]			
4814 0C40h	PINCTRL273	0	1	EMAC[0]_TXD[7]			
4814 0C44h	PINCTRL274	0	1	EMAC[0]_TXEN			
4814 0C48h	PINCTRL275	1	0	MDIO_MCLK			
4814 0C4Ch	PINCTRL276	1	0	MDIO_MDIO			
4814 0C50h	PINCTRL277	1	0				
4814 0C54h	PINCTRL278	1	0				
4814 0C58h	PINCTRL279	0	1				
4814 0C5Ch	PINCTRL280	0	1				
4814 0C60h	PINCTRL281	0	1				
4814 0C64h	PINCTRL282	0	1				
4814 0C68h	PINCTRL283	0	0				
4814 0C6Ch	PINCTRL284	0	0				
4814 0C70h	PINCTRL285	0	0				
4814 0C74h	PINCTRL286	0	0				
4814 0C78h	PINCTRL287	1	1	I2C[0]_SCL			
4814 0C7Ch	PINCTRL288	1	1	I2C[0]_SDA			
4814 0C80h	PINCTRL289	1	1	I2C[1]_SCL			
4814 0C84h	PINCTRL290	1	1	I2C[1]_SDA			
4814 0C88h	PINCTRL291	0	0	GP0[0]			
4814 0C8Ch	PINCTRL292	0	0	GP0[1]			
4814 0C90h	PINCTRL293	0	0	GP0[2]			
4814 0C94h	PINCTRL294	0	0	GP0[3]	TCLKIN		
4814 0C98h	PINCTRL295	0	0	GP0[4]			
4814 0C9Ch	PINCTRL296	0	0	GP0[5]	MCA[2]_AMUTEIN	GPMC_A[24]	
4814 0CA0h	PINCTRL297	0	0	GP0[6]	MCA[1]_AMUTEIN	GPMC_A[23]	

Table 1-251. PINCTRL*n* Register Pin Multiplexing (continued)

Address	Register	PULLTYPESEL	PULLDIS	MUXMODE[2:0]			
				000	001	010	011
4814 0CA4h	PINCTRL298	0	0	GP0[7]	MCA[0]_AMUTEIN		
4814 0CA8h	PINCTRL299	0	0	GP1[30]	SATA_ACT0_LED (silicon revision 1.x) SATA_ACT1_LED (silicon revision 2.x)		
4814 0CACh	PINCTRL300	0	0	GP1[31]	SATA_ACT1_LED (silicon revision 1.x) SATA_ACT0_LED (silicon revision 2.x)		
4814 0CB0h	PINCTRL301	0	1	HDMI_SCL			
4814 0CB4h	PINCTRL302	0	1	HDMI_SDA			
4814 0CB8h	PINCTRL303	1	0	HDMI_CEC			
4814 0CBCh	PINCTRL304	0	0	HDMI_HPDET			
4814 0CC0h	PINCTRL305	1	0	TCLK			
4814 0CC4h	PINCTRL306	0	1	RTCK			
4814 0CC8h	PINCTRL307	1	0	TDI			
4814 0CCCh	PINCTRL308	0	1	TDO			
4814 0CD0h	PINCTRL309	1	0	TMS			
4814 0CD4h	PINCTRL310	0	0	$\overline{\text{TRST}}$			
4814 0CD8h	PINCTRL311	1	0	EMU0			
4814 0CDCh	PINCTRL312	1	0	EMU1			
4814 0CE0h	PINCTRL313	1	0	EMU2			
4814 0CE4h	PINCTRL314	1	0	EMU3			
4814 0CE8h	PINCTRL315	1	0	EMU4			
4814 0CECh	PINCTRL316	1	0	$\overline{\text{RESET}}$			
4814 0CF0h	PINCTRL317	1	0	$\overline{\text{NMI}}$			
4814 0CF4h	PINCTRL318	1	0	$\overline{\text{RSTOUT}}$			
4814 0CF8h	PINCTRL319	1	0	WD_OUT			
4814 0CFCh	PINCTRL320	0	1	CLKOUT			
4814 0D00h	PINCTRL321	0	0	CLKIN32			
4814 0D04h	PINCTRL322	0	0	USB0_DRVVBUS			
4814 0D08h	PINCTRL323	0	0	USB1_DRVVBUS			
4814 0D0Ch - 4814 0FFFh	Reserved						

1.18 Interrupt Controller

The device provides two interrupt controller (INTC) modules that controls interrupts at the device level: the Cortex-A8 MPU subsystem interrupt controller and the DSP subsystem (DSPSS) interrupt controller.

Figure 1-201 shows the top level block diagram of interrupt controller in device.

1.18.1 Cortex-A8 MPU Subsystem Interrupt Controller

This INTC module is a single functional unit that is integrated in the Cortex-A8 multiprocessor core alongside Cortex-A8 processors. It provides:

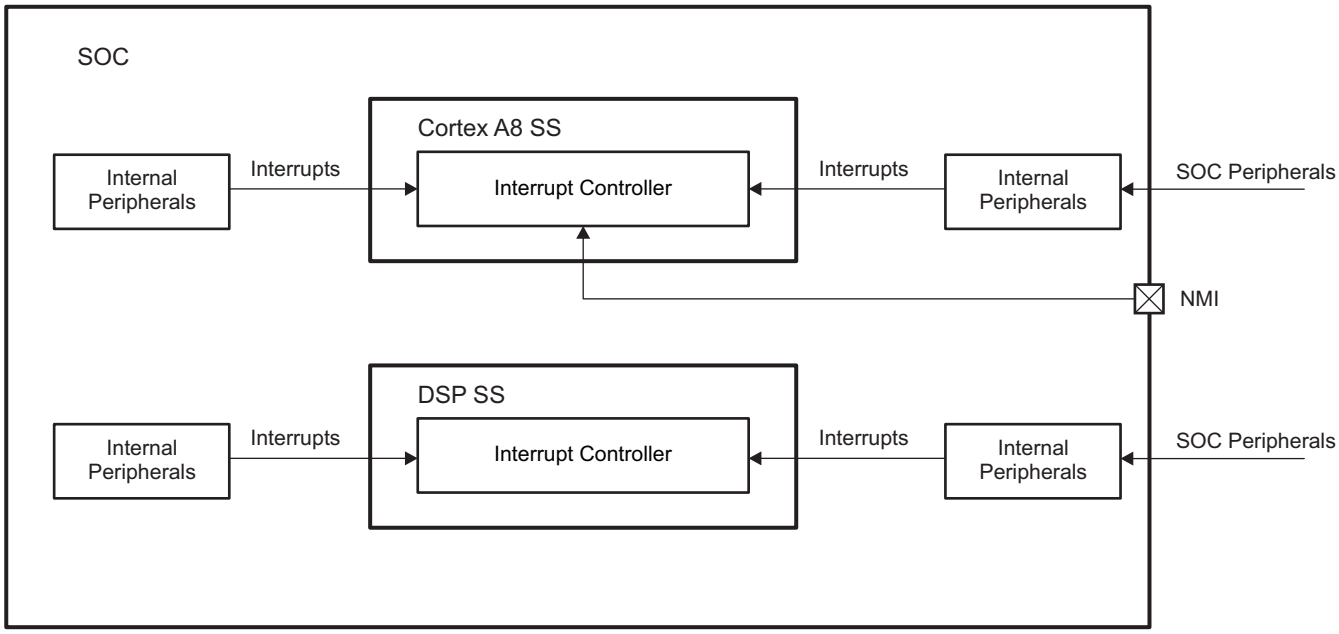
- 128 hardware interrupt inputs.
- Generation of interrupts by software.
- Prioritization of interrupts.
- Masking of any interrupts.
- Distribution of the interrupts to the target Cortex-A8 processor(s).
- Tracking the status of interrupts.

For detailed information about this module, see the *ARM Interrupt Controller (AINTC)* chapter.

1.18.2 DSP Subsystem (DSPSS) Interrupt Controller

This module is used for the DSP, but is not described in this section. For detailed information about this module, see Section 1.3.

Figure 1-201. Interrupt Controller in Device



1.19 Resets

The device has several types of system-level resets. The reset types differ by how they are initiated and by their effect on the device. Table 1-252 lists these reset types, along with the reset initiator and the effects of each reset on the device. Each type is further described in the Reset section of the device-specific data manual and the *Power, Reset, and Clock Management (PRCM) Module* chapter.

Table 1-252. System-Level Reset Sources

Type	Initiator	Resets All Modules,Excluding Emulation	Resets Emulation	Latches Boot Pins	Asserts RSTOUT*n* pin
Power-on Reset (POR)	$\overline{\text{POR}}$ pin	Yes	Yes	Yes	Yes
External Warm Reset	$\overline{\text{RESET}}$ pin	Yes	No	Yes	Yes
Emulation Warm Reset	On-chip Emulation Logic	Yes	No	No	Yes
Watchdog Reset	Watchdog Timer	Yes	No	No	Yes
Software Global Cold Reset	Software	Yes	Yes	No	Yes
Software Global Warm Reset	Software	Yes	No	No	Yes
Test Reset	$\overline{\text{TRST}}$ pin	No	Yes	No	No

The $\overline{\text{RSTOUT}}$ pin on the device reflects the device reset status. The $\overline{\text{RSTOUT}}$ pin remains asserted until the PRCM module releases the host ARM Cortex-A8 processor for reset. This output is always asserted when any of the following resets occur:

- Power-On Reset ($\overline{\text{POR}}$)
- External Warm Reset
- Emulation Warm Reset ($\overline{\text{RESET}}$)
- Software Global Cold/Warm Reset
- Watchdog Timer Reset

If any of the previous reset sources occur simultaneously, the device only processes the highest-priority reset request. The reset request priorities, from high to low, are as follows:

1. Power-On Reset ($\overline{\text{POR}}$)
2. Test Reset ($\overline{\text{TRST}}$)
3. External Warm Reset ($\overline{\text{RESET}}$)
4. Emulation Cold/Warm Resets
5. Watchdog Reset
6. Software Global Cold/Warm Resets

Reset Sources and Reset Sequences are described in detail in the Reset Management section of the *Power, Reset, and Clock Management (PRCM) Module* chapter.

Refer to the device-specific data manual for Pin Behaviors at Reset. Internal pull-up/down resistors are enabled during and immediately after reset as described in the Terminal Functions section of the data manual.

High-Definition Video Processing Subsystem (HDVPSS)

This chapter describes the High-Definition Video Processing Subsystem (HDVPSS).

Topic **Page**

2.1 Introduction

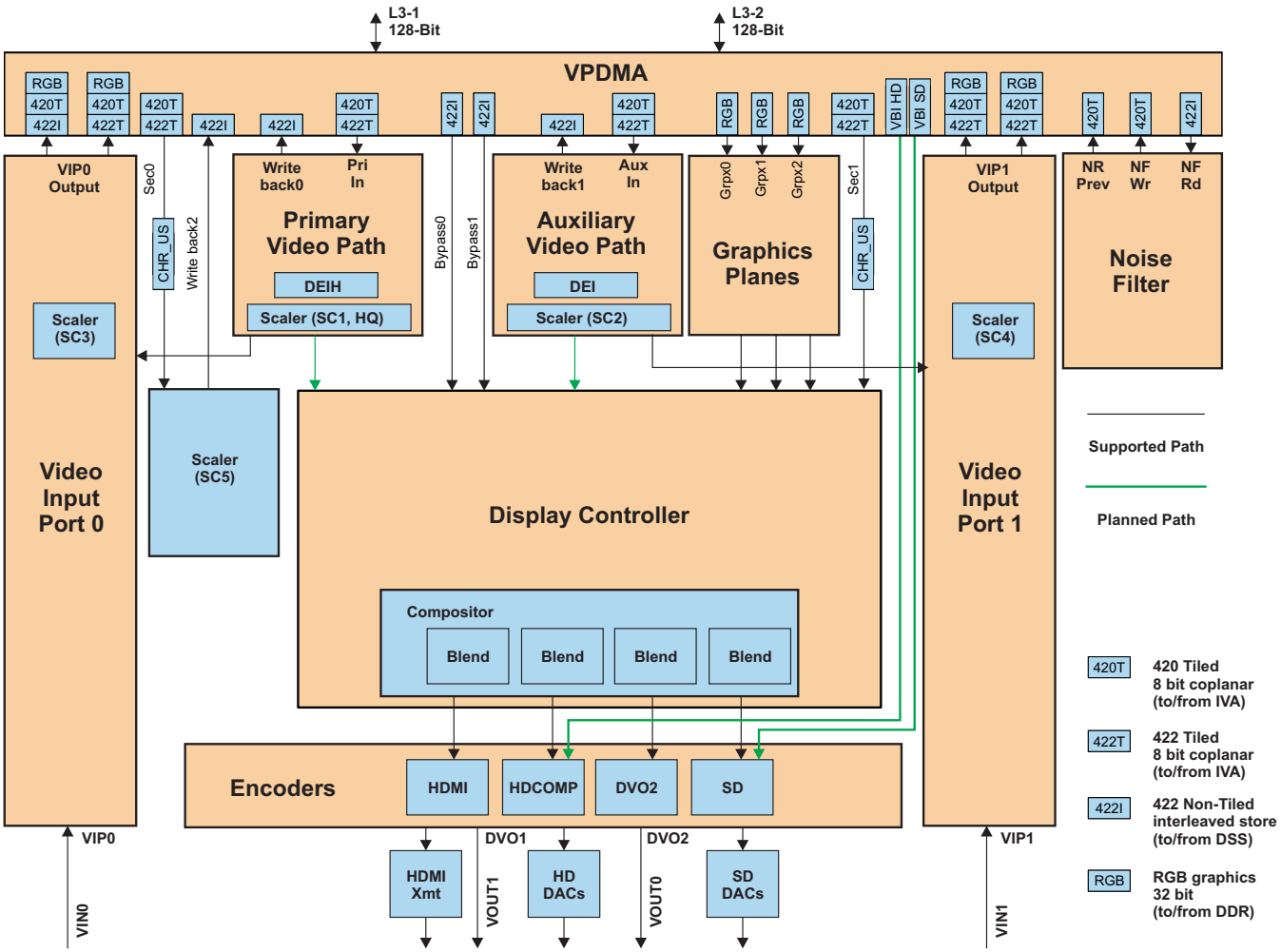
2.1.1 Overview

The high-definition video processing subsystem (HDVPSS) includes video/graphics display and capture processing using the latest TI developed algorithms, flexible compositing and blending engine, full range of external video interfaces in order to deliver a high quality video contents to the end devices.

2.1.2 Functional Block Diagram

Figure 2-1 shows a block diagram of the HDVPSS.

Figure 2-1. HDVPSS Detailed Block Diagram



2.1.3 Terminology Used in this Document

The following is a brief explanation of some terms used in this document:

COMP— Compositor

DEI— De-Interlacer

DEIH— High quality De-Interlacer

DVO— Digital Video Output

GRPX— Graphics Pipeline

HD— High Definition

HDCOMP— High Definition Component

HDMI— High Definition Multimedia Interface

HDVPSS— High Definition Video Processing Subsystem

NF— Noise Filter

NTSC— National Television System Committee

PAL— Phase Alternating Line

SC— Scaler

SD— Standard Definition

SDK— Software Development Kit

TILER— Tiling and Isometric Lightweight Engine for Rotation

VENC— Video Encoder

VIP— Video Input Port

VPDMA— Video Port Direct Memory Access

2.1.4 Data Format

Table 2-1 lists the data formats for the HDVPSS. Note that for the 422T input port, the maximum input width is 960 pixels if the data format is YUV422I_YUYV.

Table 2-1. Data Format

Name	Data Format	Arrangement	TILER Support
422I	YUV422I_YUYV	Single buffer: Y U Y V Y U Y V	No
420T	YUV420SP_UV	Y buffer: Y Y Y Y UV buffer: U V U V	Y: 8-bit container UV: 16-bit container
422T	YUV422SP_UV	Y buffer: Y Y Y Y UV buffer: U V U V	Y: 8-bit container UV: 16-bit container
422T	YUV422I_YUYV	Single buffer: Y U Y V Y U Y V	No

2.1.5 HDVPSS Features

2.1.5.1 Overall Features

- Two independent video capture input ports up to 165 MHz, each VIP instance supports scaling, pixel

format conversion, and up to one 1080P60 or 8-channel multiplexed D1 data.

- Two video processing engines with de-interlacing, scaling, noise reduction, format conversion (aspect-ratio, and pixel format).
- The HDVPSS is able to accept the HDVICP2 video decoder output formats and adjust to several video formats. This includes (but not limited to) tiled and raster data formats, scan format conversion, aspect-ratio conversion, and frame-size conversion.
- Three independent graphics processing engines with scaling capability, alpha-blending, chroma keying.
- Four independent compositors (3 HD + 1 SD) supports composition of video and graphics overlays to provide various display combinations, each compositor support up to 5 display overlays (2 video + 3 graphics), alpha-blending, chroma keying and display order reshuffling.
- Four VENCs (2 HD Digital, 1 HD analog, and 1 SD analog) supports up to 3 HD (up to 1080P60) + 1 SD display simultaneously.
- The HDVPSS handles both video and graphics efficiently to create high-quality user interfaces. This includes (but not limited to) de-interlacing, scaling, noise reduction, alpha blending, chroma keying, flicker filtering, and pixel format conversion.
- HDMI 1.3a compliant transmitter up to 162 MHz.

2.1.5.2 Video Processing Features

- Two parallel video processing pipelines (primary and auxiliary) for concurrent video stream processing are supported.
- The primary video pipeline is used for processing video for the full-size HD display. The primary video pipeline employs highest-quality image processing, per pixel motion adaptive temporal and spatial noise reduction, motion-adaptive de-interlacer, edge-directed scaler, and spatial edge enhancement.
- The auxiliary video pipeline is used for processing the video for the full HD/SD video output. The pipeline employs an area-efficient algorithm that includes a motion-adaptive 3D de-interlacer and a non-edge adaptive scaler.
- The noise filter (NF) performs a memory to memory spatial/temporal noise filter algorithm on a 422 raster input source and produces a 420 tiled output source.
- Supported video input formats are 4:2:0 (aligned-chroma, semi-planar, frame/field), and 4:2:2 (planar, semi-planar, frame/field). YUV420, the output formats of the HDVICP2 and video formats of captured external digital video data, is supported.
- Scan format conversion (interlaced to progressive and conversely) is supported. Especially the interlaced to progressive conversion employs a high quality motion-adaptive 3D de-interlacer to properly render both static and dynamic objections in scenes.
- The output of the video processing is sent to either compositor or external memory. When the output is sent to external memory, context switching between multiple inputs is handled efficiently, that is, multi-CH mode of operation.
- Both the main and auxiliary video pipelines include a write-back path to the external memory to support memory to memory scaling of video frames independently from the display output frame timing.
- Color keying (transparency) is supported.

2.1.5.3 Graphics Features

- Three independently generated graphics layers are supported.
- Each graphic layer supports full-screen resolution graphics.
- Each of the graphics pipelines includes an up/down scaler optimized for graphics application. The scaler supports scaling ratios from 0.25× to 4× with 0.01 scaling step size.
- Supported graphics formats are:
  - 32-bit: ARGB8888, RGBA8888
  - 24-bit: RGB888, ARGB6666,RGBA6666
  - 16-bit: ARGB1555, RGB565, ARGB4444, RGBA5551, RGB4444

  - Bitmap: 1, 2, 4, 8-bit CLUT table
- Global and pixel-level alpha blending value is supported (256 levels). For pixel-level blending, the alpha value can come from either source or CLUT table.
- Color keying (transparency) is supported.
- Stenciling (pixel masking with a separate 1-bit mask plane) is supported independently for each graphics layer.

2.1.5.4 High Definition/Standard Definition (HD/SD) Compositor Features

- Four independently controlled compositors (HDMI/DVO1, HDComp, DVO2, SD) are supported to drive the corresponding display encoder outputs.
- The HD compositor supports composition of video and graphics layers to provide full-size video display, graphics overlay, and video-in-graphics for HD video outputs.
- The SD compositor supports video display, graphics overlay, and video-in-graphics for SD video outputs.
- Each input layer is given a display order priority that determines the display and blending orders.
- Each output supports independent layer visibility control.
- The compositor supports 256-level alpha blending of two overlapping layers.

2.1.5.5 High Definition/Standard Definition (HD/SD) Video Signal Encoding Features (Video Displays)

- A single HDMI 1.3 compliant interface
- Two DVO (VOUT) interfaces that support up to 1080p@60 Hz, 8/16/24 YCbCr/RGB formats are included.
- VOUT0 supports up to 10/20/30-bit and VOUT1 supports only 20-bit.
- Each VOUT port supports both embedded and separated sync outputs.
- HD Component: meets all requirements defined in CEA 770.3-D.
- SD Composite/S-video (NTSC/PAL): meets all requirements defined in ITU-R BT.470-6 (TBD: SECAM support).
- Supported VESA resolutions up to 165 MHz. Note that the maximum supported line width is 1920 pixels. Furthermore, width and height values must always be an even number.
- Support Simultaneous HD and SD output.
- 2 video clocks to support two independent HD displays, the clock of third HD display will be tied to one of the other two.

2.1.5.6 Video Capture Features

- The HDVPSS supports two independently configurable external video input capture ports with up to 165 MHz.
- Each video input capture port can be operated as one 16-bit input channel (with separate Y and Cb/Cr inputs) or two clock independent 8-bit input channels (with interleaved Y/C data input). Further one of the VIP ports can operate in 24-bit mode to support RGB capture.
- Supports both embedded sync and discrete sync.
- The video capture port channel supports de-multiplexing of both pixel-to-pixel and line-to-line multiplexed streams.
- Up to 1920 × 1200@60 Hz (165 MHz) input data rate supports 16-bit mode input port.
- Each video capture port supports one scaler capable of both up and down scaling of one non-multiplexed input stream (one of two 8-bit channel inputs or 16-bit channel input data). Note that if the source is from external video decoder/camera, only down scaling is supported.
- Each video capture port supports one programmable color space conversion to convert between 24-bit RGB data and YCbCr data.
- The VIP supports data storage in RGB, 422, and 420 formats.
- Each video capture port channel supports chroma down-sampling (422 to 420) for any non-multiplexed

input data. The chroma down-sampling for multiplexed streams is done as memory to memory operations outside of the HDVPSS on an individual frame data.

2.1.5.7 Other Features

- The HDVPSS supports 2 secondary video input sources in 420 or 422 tiled format used for memory-memory operations and SD output display.
- The HDVPSS supports 2 bypass video input sources in YUV422 YUYV interleaved non-tiled format for display.
- The HDVPSS supports video data display for the following; there is no processing performed, that is, no scaler, de-interlacing, format conversion.
  1. Secondary 1 video input source in the SD display only.
  2. 2 bypass video input sources in YUV422 YUYV interleaved format for display.
- The HDVPSS supports video data write back paths (memory-to-memory) for the following:
  1. Scaled video from secondary 0 video input sources - saved as YUV422 YUYV interleaved format video.
  2. Scaled video from auxiliary DEI video channel saved as YUV422 YUYV interleaved format video.
  3. Independent scaled video from primary DEIH output - saved as a 420 video using the scaler and chroma_downsampler resources in VIN0 port
  4. Scaled video from primary DEIH video channel saved as YUV422 YUYV interleaved format video.
  5. Independent scaled video from auxiliary DEI output – saved as a 420 video using the scaler and chroma_downsampler resources in the VIN1 port.

2.2 Chroma Up-Sampler (CHR_US) Module

The chroma up-sampler (CHR_US) module is used to convert from YCbCr 4:2:0 data format input to YCbCr 4:2:2 format output.

2.2.1 Features

- Input:
  - YUV420 Semi-planar Tiled Memory
  - YUV422 Semi-planar Tiled Memory
  - YUV420 Semi-planar Non-Tiled Memory
  - YUV422 Semi-planar Non-Tiled Memory
  - YUV422 YUYV Interleaved Non-Tiled Memory
- Output:
  - YUV422 YUYV Interleave non-Tiled Memory
- Supports both interlaced and progressive inputs
- Support bypass mode for 4:2:2 input

2.2.2 Functional Description

The up-sampling is performed by an interpolation filter that uses Catmull-Rom algorithm.

2.3 De-Interlacer (DEI) Module

The de-interlacer (DEI) module is used to generate progressive data output from interlaced input format (also known as, de-interlacing).

2.3.1 Features

- Input:
  - YUV420 Semi-planar Tiled Memory

  - YUV422 Semi-planar Tiled Memory
  - YUV420 Semi-planar Non-Tiled Memory
  - YUV422 Semi-planar Non-Tiled Memory
  - YUV422 YUYV interleaved Non-Tiled Memory
- Output:
  - Output is always tied to the Scalar input
- Motion-adaptive de-interlacing
- Edge-Directed Interpolation (EDI)
- Bad Edit Detection (BED)
- Interlace Bypass:
  - For Interlaced Input, the module can pass the inputs data directly to the outputs in a bypass configuration. No internal processing is performed.

2.3.2 Functional Description

The DEI is primarily used to convert interlaced video source material to progressive form. This particular module is a reduced feature set of the DEIH module, it does not perform Temporal Noise Reduction and is limited to 4 field motion detection. It performs edge directed interpolation, but utilizes a simpler (and smaller) algorithm compared to the DEIH module. It can perform de-interlacing on up to 1080i video input source, producing 1080p video output.

2.3.3 Modes of Operation

The DEI can be operated in three modes:

- Interlace bypass mode
- Progressive bypass mode
- De-interlacer mode

2.3.3.1 Interlace Bypass Mode

In the interlace bypass mode, input luma and chroma are buffered and sent to the stage after DEI without processing.

2.3.3.2 Progressive Bypass Mode

In the progressive bypass mode, the top and bottom YUV streams are combined into a continuous frame and are sent to the following stage as a frame without any processing.

2.3.3.3 Interlace Mode

In the interlace mode, DEI takes in alternative field YUV data and sends out a sequential frame YUV data by interpolation techniques. DEI supports four interpolation modes when converting interlaced pictures to progressive pictures.

2.3.4 Modes of Interpolation

2.3.4.1 Interpolation Mode 0

In mode 0, the interpolated field is created by simple line averaging from the original YUV data. That is, the interpolated line is created by averaging its top and bottom line. Setting of MDT mode has no effect on output pictures.

2.3.4.2 Interpolation Mode 1

In mode 1, the interpolated field is created by averaging pixels from fields before and after the current field. In other words, if the current field is a top field, the interpolated bottom field picture is created by averaging pixels from bottom field pictures before and after the current field. MDT setting has no effect on the result.

2.3.4.3 Interpolation Mode 2 (EDI)

Mode 2 is an edge assisted interlace mode with edges detected from the luma information of a 2 × 7 (H × W) frame window. It detects seven possible edges between 45 and 135 degrees. Edges with slopes greater than 135 degrees are treated as 135 degree lines, and edges with slopes less than 45 degrees are treated as 45 degree lines.

Luma for missing lines are interpolated using original luma along the detected edge.

Motion value (MV) from the MDT module is used to select coefficients from a look up table on how 2D interpolation from the current field and 3D interpolation from two fields adjacent to the current fields are blended. The way to perform blending for chroma is slightly different than for luma, because the MV is based on luma and extra care needs to be taken when blending is applied to chroma.

At this mode, edge-directed interpolation is applied to luma only. Vertical interpolation is performed for chroma data.

2.3.4.4 Interpolation Mode 3 (EDI)

The edge detection method used in this mode is similar to interpolation mode 2. The only difference is that the edge-directed interpolation is performed on both luma and chroma. Chroma is interpolated similarly according to the edge vectors obtained based on luma information. The reason chroma is interpolated slightly different is because of the down-sampled chroma data.

2.3.5 Motion Detection (MDT)

Motion values from the MDT module are calculated only based on luma information. The motion value is used to determine how 2D and 3D interpolated results should be blended. The motion detection module in DEI supports 4-field mode.

2.4 High-Quality De-Interlacer (DEIH) Module

The high quality de-interlacer (DEI_H) module is used to generate progressive data output from interlaced input format (also known as, de-interlacing).

2.4.1 Features

- Input:
  - YUV420 Semi-planar Tiled Memory
  - YUV422 Semi-planar Tiled Memory
  - YUV420 Semi-planar Non-Tiled Memory
  - YUV422 Semi-planar Non-Tiled Memory
  - YUV422 YUY`V interleaved Non-Tiled Memory
- Output:
  - Output is always tied to the Scalar input
- Motion-adaptive de-interlacing:
  - Motion detection (MDT) based on both luma and chroma
  - Two modes of motion detection: 4-field, 5-field
- Temporal Noise Reduction (TNR):
  - Motion directed 3D noise reduction. Performed before EDI.
  - Allows noise reduction for both luma and chroma

- Edge-Directed Interpolation (EDI)
- Spatial Nose Reduction (SNR)
- Bad Edit Detection (BED)
- TNR support on progressive input Interlace Bypass
  - For Interlaced Input, the module can pass the inputs data directly to the outputs in a bypass configuration. No internal processing is performed.

2.4.2 Functional Description

The DEIH is primarily used to convert interlaced video source material to progressive form. This particular module incorporates features such as Temporal Noise Reduction, 4- and 5-field motion detection and very fine edge detection capabilities to produce a very high-quality de-interlaced output. It can perform de-interlacing on up to 1080i video input source, producing 1080p video output.

2.4.3 Modes of Operation

The DEIH can be operated in four modes:

- Interlace bypass mode
- Progressive bypass mode
- Progressive mode
- De-interlace mode

2.4.3.1 Interlace Bypass Mode

In the interlace mode, data are buffered and sent to the stage after DEIH without processing.

2.4.3.2 Progressive Bypass Mode

In the progressive mode, the top and bottom YUV streams are combined into a continuous frame and are sent to the following stage as a frame.

2.4.3.3 Progressive Mode

The only available data processing in the progressive mode are 2D (spatial) and 3D (temporal) noise reduction. Those two noise reduction modes can be independently enabled. If 3D noise reduction is turned on, the top and bottom YTNR and UVTNR (filtered Y and UV) are sent to external memory and should be returned to the DEIH after 1 field delay. The top and bottom YUV streams are treated as two separate streams and are combined into a frame before being sent out.

2.4.3.4 Interlace Mode

In the Interlace mode, DEIH takes in alternative field YUV data and sends out a sequential frame YUV data by interpolation techniques mentioned below. DEIH supports 4 interpolation modes, 3 motion detection (MDT) modes, 3D temporal noise reduction (TNR) and 2D spatial noise reduction (SNR) when converting interlaced pictures to progressive pictures.

2.4.4 Modes of Interpolation

2.4.4.1 Interpolation Mode 0

In mode 0, the interpolated field is created by simple line averaging from the original YUV data. That is, the interpolated line is created by averaging its top and bottom line. MDT setting has no effect on the result. TNR and SNR can be independently applied.

2.4.4.2 Interpolation Mode 1

In mode 1, the interpolated field is created by averaging pixels from fields before and after the current field. In other words, if current field is a top field, the interpolated bottom field picture is created by averaging pixels from bottom field pictures before and after the current field. TNR and SNR can be independently applied.

2.4.4.3 Interpolation Mode 2 (EDI)

Mode 2 is an edge assisted interlace mode with edges detected from the luma information of a 7 × 2 (W × H) frame window. It detects seven possible edges between 45 and 135 degrees. Edges with slopes greater than 135 degrees are treated as 135 degree lines, and edges with slopes less than 45 degrees are treated as 45 degree lines.

Luma for missing lines are interpolated using original luma along the detected edge. Chroma is interpolated similarly according to the edge vectors obtained based on luma information.

2.4.4.4 Interpolation Mode 3 (EDI)

Mode 3 is an edge assisted interlace mode with edges detected from a 17 × 2 (W × H) frame window. It detects edges between 11.3 and 168.7 degrees with 1/8 pixel resolution.

Luma for missing lines are interpolated from 2 methods: 4-tap vertical interpolation and 2-tap interpolation along detected edges.

Chroma is interpolated using the pixel resolution edge vectors obtained based on luma information.

2.4.5 Motion Detection (MDT)

Motion values from the MDT module are calculated only based on luma information. The motion value is used to determine how 2D and 3D interpolated results should be blended.

The TNR module needs a motion value to determine the mixing coefficient of YUV from current field and YTNR/UVTNR from previous frame. MV from both luma and chroma data are calculated within the TNR module and the larger MV from Y or UV is used to determine the blending factor.

The SNR module also needs MV to decide how data are blended in the sigma filter. MV for SNR sigma filters is provided by the TNR. The MDT modes discussed in the following sections apply only to the MDT module and has no effect on how MV is determined in the TNR.

2.4.5.1 MDT Modes

There are 2 modes of motion detections in the DEIH:

- 4-field
- 5-field

2.4.5.1.1 4-Field Mode

Motion value (MV) is first estimated from y_f0 and y_f2 (current and two field delayed Y). The maximum value of MV and one field delayed data of MV, MV_f1, is used to create the overall MV per pixel. In addition to y_f0 and y_f2, y_f1 is used to detect scene change. It is also used to detect slow motion in the advanced mode. Since this motion detection method utilizes three fields of luma data and one field delayed MV, it is called 4-field mode as MVs come from four fields of luma data.

There is one field delay between the YUV inputs and the YUV outputs. The “current_field” that the interpolator works on is the YUV_f1 data.

2.4.5.1.2 5-Field Mode

Motion value (MV) is first calculated from yuv_f0 and yuv_f2, like the 4-field mode, but the overall MV per pixel is calculated using both MV_f1 and MV_f2 as the inputs. Instead of yuv_f1, yuv_f3 is used in the scene change detection. It is called 5-field mode since MVs come from five fields of luma data.

There are two fields delay between the YUV inputs from the VPI and the YUV outputs. The “current_field” interpolator works on is yuv_f2.

2.4.6 Temporal Noise Reduction (TNR)

Temporal nose filter is a 3D noise filter that does IIR filtering temporarily. For an interlaced picture, pixels delayed by two fields, pixels of two neighboring top fields or two neighboring bottom fields are used in the IIR filtering. When DEIH is at the progressive mode, a picture frame is broken into top and bottom fields. TNR is performed on top and bottom fields, simultaneously. TNR data only needs to be delayed by one field instead of two fields at progressive mode.

2.4.7 Spatial Noise Reduction (SNR)

The spatial noise filter does 2D noise filtering on a ‘frame’ picture after the interpolation step. It does not differentiate if DEIH is in the progressive mode or the interlace mode. The SNR contains two filter algorithms: Impulse noise filter and Sigma filter.

The impulse noise filter aims at removing isolated speckles. A set of median filters is adaptively selected to achieve this goal depending on the amount of impulse noise detected. The Gaussian noise filter is targeting removing Gaussian noise.

The sigma filter is an edge-friendly filter by including only neighboring pixels that have a value not deviating from the current pixel by a given range.

2.5 Video Compositor (COMP) Module

The video compositor (COMP) module is used to blend the input video and graphics sources into a single stream to drive video encoders (VENCs).

2.5.1 Features

- Four independently controlled compositors
- Supports four displays: 3 HD and 1 SD
- Supports up to 5 input layers for each display: 2 videos and 3 graphics
- Supports 256-level cascade alpha blending for up to 2 video input layers first and up to 3 graphic input layers later
- Supports programmable display priority for input layers
- Supports programmable background color

2.5.2 Functional Description

The COMP module has four independently controlled blenders (compositors), each of which can take up to five input layers (2 videos and 3 graphics) to generate one composite output layer. Each blender is associated with one of the four video encoders (VENCs) to drive the display data. The accessibility of each blender to the input layer is shown in Table 2-2.

Each of the input layers can be associated to one or more VENCs in the HDVPSS. All the input layers associated with a VENC should have the same size and type (progressive or interlaced) of the display. This implies that if an input layer is shared with multiple VENCs, all those VENCs should have the same size, type and frame rate. For example, if the 1080p HDMI output is desired to have primary video input and auxiliary video input with 1 graphics overlay, then the primary and auxiliary video input must be 1080p, and one of the graphics must be 1080p. If another blender is setup for 480i, it cannot use either the primary or auxiliary video inputs, and must use a different graphics input.

Input layers associated to a VENC are grouped together, reordered according to the user programmable display priority order list, and blended using embedded alpha values between overlapped layers.

Table 2-2. Input Layers and Associated Blenders

Input Layers\Compositor	HDMI	HDCOMP	DVO2	SD
Primary HD video	Yes	No	Yes	No
HDCOMP video	No	Yes	No	No
SD video	No	No	No	Yes
HD auxiliary video	Yes	Yes	Yes	No
GRPX0	Yes	Yes	Yes	Yes
GRPX1	Yes	Yes	Yes	Yes
GRPX2	Yes	Yes	Yes	Yes

2.5.2.1 Input Sources

Each blender in the COMP module receives up to five input sources to blend and sends the composited output to the corresponding VENC.

All the inputs that are connected to a single blender should:

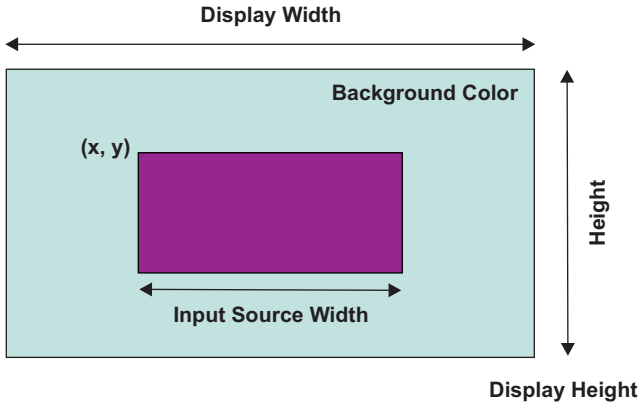
(a) Have the same size of its corresponding VENC display size. This is done by forming a WINDOW (see Figure 2-2) in the upstream modules. Note that it does not imply any restriction on the actual input video sizes in external memory to be the same. For example, Video-0 can be 1920 × 1080 and Video-1 can be 720 × 480 in the memory.

(b) Carry their priority levels to re-order themselves as layers on the final display stream (only for graphic inputs).

(c) Carry alpha values used to blend with each other using the above priority levels.

If the size of any input is not the same as the display size, background color is inserted in the remaining portion as shown in Figure 2-2.

Figure 2-2. Upstream Module Window Coloring



2.5.2.2 Blender

The compositor blends two video input sources together to generate a single video output first, and then reshuffles the video and graphics inputs to allow flexible display order based on programmable display priority, and finally blends all of them together to generate a single display output to the VENC.

When two input sources, input1 and input2, are blended, the resulting source is calculated as follows:

```
Blended Output     = (a)*input1 + (1-a)*input2, if input1 is top layer and input2 is bottom layer
Blended Output     = (1-b)*input1 + (b)*input2, if input2 is top layer and input1 is bottom layer
```

Where a and b are alpha values of input1 and input2 windows, respectively. The layer order (top or bottom) is decided by priority levels of the two video inputs.

2.5.2.3 Background Color

There is a programmable background color in the COMP module. Background color is configured in RGB format with 10-bits in each R, G, and B. Any pixel with an alpha value of 0 is replaced by this color. If the video bottom layer is disabled, the video top layer is blended with background color. If both video layers are disabled, the background color is used as the output of video alpha blending. For SD channel, only one video stream comes in and it is blended with the background color.

2.6 Graphics (GRPX) Module

The graphics module (GRPX) is a graphics processor that composes RGB or Bit Map data to create a display plane input for the video compositor (COMP) module.

2.6.1 Features

- Non-Tiled memory only
- Input:
  - RGB565
  - RGB888
  - ARGB1555
  - ARGB4444
  - ARGB6666
  - ARGB8888
  - RGBA5551
  - RGBA4444
  - RGBA6666
  - RGBA8888
  - Bitmap data (8/4/2/1 bit) with configurable CLUT
  - Non-Tiled memory
  - Supports optional stenciling (1-bit mask) table
- Output:
  - Output data format is ARGB32
- Width, Height, placement (X & Y) parameters
- Display priority
- Scalar enable/disable
- Scaling ratio attributes
- Stenciling enable/disable
- Support for global, color and pixel level Blending
- Configurable Chroma Keying options
- Bounding box enable

2.6.2 Functional Description

The GRPX module takes in RGB or bitmap data, applies attributes, and sends out the data output to the COMP module.

2.6.2.1 Disp_Priority

An inter-graphics-pipeline display priority level may be used to switch the order from two GRPX pipelines. The GRPX module simply passes these values along with each pixel data to the COMP module and the COMP uses it to reorder between two GRPX data. This should only be used when both GRPX outputs are tied to the same display output format and rate.

2.6.2.2 Scaler

The scaler used in the GRPX module consists of a 5-tap/8-phase horizontal and 4-tap/8-phase vertical polyphase filters.

2.6.2.3 Anti-Flicker Filter Implementation

When data is scaled up or down, the poly-phase vertical filter inherently provides anti-flicker filtering. Even when data is not needed to be scaled, the scaler still is enabled to perform low-pass filtering along the vertical direction with scaling ratio = 1.

2.6.2.4 Stenciling

The VPDMA sends 1 bit for each pixel that has the “stenciling” feature enabled. This bit is used to force the alpha value of the pixel to 0, in order to mask-off the pixel (that is, make the pixel transparent). This feature can be used by an application to assign an arbitrary mask shape to a rectangular graphics.

2.6.2.5 Boundary Box Blending

The GRPX module supports overwriting alpha values of pixels that make up a 1-pixel wide boundary box with a semitransparent (alpha) value (default = 80h) so that the flickering around the edges can be minimized. A programmed boundary box alpha value is assigned to each boundary pixel.

2.6.2.6 Blending

The GRPX module supports four blending modes:

- no-blending
- global
- CLUT
- pixel

When no-blending is selected, the GRPX forces each alpha to FFh. In global blending, a programmed alpha value is assigned to each pixel. For color or pixel blending, the alpha value with CLUT mapping or embedded alpha value (argb format source) is used, respectively.

The blending level supported is from 0 (transparent) to 255 (totally opaque).

2.6.2.7 Transparency Handling

The GRPX module supports a transparency mode to determine how an alpha value for each pixel is set. When transparency is enabled, each pixel color is compared against the programmed transparency color (RGB888) to determine whether the pixel is to be transparent or not. If the colors match, the alpha value is forced to 00h. Otherwise, the alpha value remains as programmed. During the comparison, the application can assign LSB bit masking to notify the module which LSBs are masked. The GRPX supports three different LSB bit masks: masking LSB bit0; masking LSB bit[1:0] and masking LSB[2:0]. For example, if masking LSB[2:0] is selected, then the top 5 bits of the R/G/B component data are compared to the corresponding 5 bits of the programmed transparency R/G/B color.

Because the alpha value is also scaled in the same way the color components (RGB) are scaled, all enabled blending/transparency related tasks are performed in the following order:

1. Application of no-blending (force alpha = FFh) or Global blending (force alpha = programmed global blend level)
2. Transparency check to force alpha = 0 for transparent pixels
3. Stenciling application to force masked pixels (stencil data = 1) with alpha = 0
4. Scaling of alpha (if scaling is enabled)
5. Box blending (force alpha = programmed box_blend_level)

2.7 High-Definition Video Encoder (HD_VENC) Module

There are three high-definition video encoders (HD_VENCs) in the HDVPSS. The HD_VENC module encodes the active video data from the COMP module and sends it up to a 30-bit wide DVO (digital video output) port by generating timing signals to control the data flow.

2.7.1 Features

- Two independent digital video output ports (DVO1 and DVO2)
- The DVO ports support output data rates up to 165 MHz
- The DVO1 (VOUT1) port supports 20-bit wide output with embedded sync or discrete sync
- The DVO2 (VOUT0) port supports up to 10/20/30-bit wide output with embedded sync or discrete sync
- Display timing generator: generates the display timing signals VS, HS, FID, HBI, VBI, and ACT_VID for OSD
- Analog output support embedded sync only
- Color Space Conversion

2.7.2 Functional Description

The HD_VENC primarily has two blocks:

- OSD
- encoder

The OSD module is used to get the data from the COMP module using the sync signals generated by the encoder module. In the device, the HD_VENC supports either digital (HDMI/DVO1, DVO2) or analog output (HDCOMP), but not both. Table 2-3 lists the possible display ports and corresponding VENC names.

Table 2-3. Display Ports and VENC Names

Display Port Name	Type	VENC Name	Port Name for PINMUX
HDMI/DVO1	Digital	HDMI	VOUT1
DVO2	Digital	DVO2	VOUT0
HDCOMP	Analog	HDCOMP	Not Applicable
SD	Analog	SD	Not Applicable

2.7.2.1 OSD Interface

The interface between the OSD and the video display encoder consists of a 30-bit data bus, a pixel clock, and five sync signals. By default, 30-bit RGB video data is passed from the OSD to the encoders on every rising edge of the pixel clock during an active video period. Table 2-4 lists all the control signals of the OSD interface.

Table 2-4. OSD Interface Signals

Signal Names	Descriptions
FID	Field ID signal. In interlace mode, this signal toggles between 1 and 0 on every field. In progressive mode, this signal toggles between 1 and 0 on every frame.
VS	Vertical sync signal. This signal is a one-line long pulse. In interlace mode, this pulse indicates the first line of each field. In progressive mode, this pulse indicates the first line of each frame.
VBI	Vertical blank interval signal. This signal goes to 1 during a non-active video period; this signal stays at 0 during an active video period.
HS	Horizontal sync signal. This is the horizontal sync signal.
HBI	This is a 4-pixels wide and active-high signal. It appears once every video line.

Table 2-4. OSD Interface Signals (continued)

Signal Names	Descriptions
ACT_VID	This is an active-video qualification signal. When this signal is high, the encoder is expecting an active video data output from the OSD after 1 clock delay.

2.7.2.2 Digital Video Output (DVO)

Table 2-5 lists all formats that the digital video output supports.

Table 2-5. DVO Formats

Formats	Bits
Single-stream 656	YCbCr-10-bits
Dual-stream 656	Y/CbCr-20-bits
Triple-stream 656	Y/Cb/Cr/R/G/B - 30-bits
YUV422 20-bit output with discrete sync	Y/CbCr - 20-bits
30-bit output with discrete sync	Y/Cb/Cr/R/G/B - 30-bits

The DVO outputs both embedded sync and discrete sync. In embedded sync, the DVO supports single/dual/triple stream. In discrete sync, the following are the sync signals that the DVO outputs:

- HS (horizontal sync signal)
- VS (vertical sync signal)
- ACT_VID (active-video signal)
- FID (field ID signal)

2.7.2.3 Color Space Converter (CSC)

The color space converter (CSC) module is used to convert the input data from one color space to another by 3 × 3 matrix.

2.7.2.4 DAC Interface

There are three 12-bit video DACs inside the device. The active video digital combining with the sync signals is directly synthesized in the digital domain and sent to the DACs.

In component video format, the sync signal is inserted in the Y channel or G (green) channel. In ITU and SMPTE standards, the sync signals are also allowed to be inserted in the R. B, Pb, and Pr channels. The device implements the sync insertion only in Y and G channels.

2.8 Noise Filter (NF) Module

The noise filter (NF) module is used to reduce noise in a video sequence in order to improve video image quality and compression efficiency.

2.8.1 Features

- Input:
  - YUV422 YUYV interleaved Non-Tiled memory
- Output:
  - YUV420 Semi-Planar Tiled Memory YUV420 Semi-Planar Non-Tiled Memory
- Spatial video noise filtering
- Spatial video noise filtering bypass
- Temporal video noise filtering
- Temporal video noise filtering bypass

- Chroma down-sampling

2.8.2 Functional Description

The NF module is a combination of spatial and temporal IIR filters and chroma down-sampling. The spatial filter is performed with neighboring pixels in the current frame; the temporal filter is performed with a past filtered frame to suppress both spatial and temporal noises.

The NF module performs a memory-to-memory spatial/temporal noise filter algorithm on a 422 raster input source and produces a 420 tiled output source. Its primary use mode is part of the video input port processing. In this use mode, the external video source is captured by the video input port and sent to the memory as 422 raster format. It is then brought back into the NF module from memory, filtered, and sent back to the memory as 420 tiled format.

The NF module can be configured to perform spatial-temporal filtering, spatial-only filtering, temporal-only filtering, or at bypass mode. When a filtering function is enabled, the previous NF filtered output is required to use as an additional input to perform the noise reduction. Previous noise filter output is not required if the NF module is in bypass mode, and the NF module performs YUV422 to YUV420 chroma down-sampling only.

2.9 High-Quality Scalar (SC_H) and Scalar (SC)

The SC_H (high-quality scalar) and SC are used for scaling the incoming video stream to other resolutions.

2.9.1 Features

- Input:
  - DEI output, Chroma_up output of VIP output
- Output:
  - YUV422 YUYV Interleaved
- Horizontal/Vertical cropping
- Polyphase filter for horizontal
- Polyphase filter/Running average filter for vertical
- Scaling up to a maximum of 1920 pixels in the horizontal direction and up to 1080 pixels in the vertical direction

2.9.2 Functional Description

The Scaler takes the data from the upstream module. The input image is resized to the desired output size. The module sends the output image to the downstream module. It can scale full HD (1080p) and output full HD (1080p). The SC_H is a high profile version configured for the best quality video scaling and performs edge-directed vertical scaling. It is only used in the primary video data path. The SC is a medium profile version optimally configured for good scaling quality without the edge-directed scaling. This scalar is used in the auxiliary video path and in all other video write-back data paths in the HDVPSS module.

For both of these scalars, the scaling is performed in following three steps:

1. Trimming
2. Vertical Scaling
3. Horizontal scaling

2.9.2.1 Trimmer

The trimmer can be programmed to redefine a new source image within the input video frame sent from an upstream module before it is sent to the scaling submodules. This feature enables small area zoom out, source pan/scan, or removal of unwanted area in the video.

2.9.2.2 Vertical Scaling

The vertical scalar has a polyphase and a running average filter. While the polyphase filter is used for any up-scaling and preferably downscaling to a 3/16 scale factor, the running average filter is used only for downscaling to a ½ or less size. Selection between these two scalars is based on the user setting of "use_rav" parameter, according to the user preference of the tradeoff between sharpness preserving and introduced artifacts.

2.9.2.3 Polyphase Scaler

In both scalars (SC_H and SC), the vertical polyphase scalar is mainly implemented using a 32-phase 5-tap polyphase filter. For SC_H, a 2-tap edge-directed bilinear filter is added to improve quality making use of the edge information in the source image. The bi-linear vertical interpolation is only available in the SC_H, and it uses two source lines to generate missing lines in the output image. Due to the aliasing effects in the downscaling, this scaling is to be used only for up-scaling cases.

2.9.2.4 Running Average Filter

When a polyphase filter is used, usually it has to have many taps in order to achieve acceptable quality for very small downscaling ratio, which requires the use of many line buffers. In the HDVPSS, there is a weighted running average filter for downscaling when the scaling factor is small (for example, when the scaling ratio is less than 0.5). This highly optimized design requires only one line buffer for luma and one for chroma, which still achieves acceptable quality. The output of the running average filter is based on a weighted average of pixels in the current and previous rows in the vertical direction.

2.9.2.5 Horizontal Scaling

The horizontal scalar is implemented using a 32-phase × 7-tap polyphase filter preceded by two sets of 1/2× decimators. The general configuration of the horizontal scalar is performed as follows:

1. For upscaling, the input video is interpolated using the polyphase scalar.
2. For downscaling, input video is decimated by 2 until the modified scale factor falls between ½ and 1. Then, a polyphase filter is configured with coefficients selected based on the scale factor.

2.10 Standard-Definition Video Encoder (SD_VENC) Module

The standard-definition video encoder (SD_VENC) converts digital component YCbCr/RGB video signals to conform to the various TV standard analog video.

2.10.1 Features

- Master Clock Input - 27 MHz
- Support PAL/NTSC Standard
- Composite (CVBS)
- S-Video (Y/C)
- Color Space Conversion

2.10.2 Functional Description

The SD_VENC module operates in synchronization with horizontal/vertical sync signals generated in the built-in sync signal generator. For the analog video output, one of the following TV formats (525i or 625i) needs to be specified. The format setting specifies the format-dependent timing parameters such as sync rise/fall build-up time, active video rise/fall time, vertical sync, and equalizing pulse position.

2.10.2.1 Color Space Converter (CSC)

The 2× interpolated data then comes into a color space converter (CSC) to be converted into the desired color format.

2.10.2.2 Macrovision

The SD_VENC module supports analog copy protection Macrovision Rev7.1 to the SDTV CVBS output. Due to strict security control by Macrovision, any information such as Macrovision register map and usages are not directly distributed to the customers. This information is supplied by Macrovision upon certification.

2.10.2.3 DAC Output

The SD_ VENC module has 4 channel, 12-bit digital outputs for DAC input.

2.11 Video Input Parser (VIP) Module

The video input parser (VIP) module is used to capture the video data into the HDVPSS module.

2.11.1 Features

- Input:
  - YUV422 8-bit embedded sync mode (exclude BT. 1120)
  - YUV422 8-bit discrete sync mode
  - YUV422 16-bit embedded sync mode
  - YUV422 16-bit discrete sync mode
  - YUV422 8-bit 2×/4× pixel multiplexed mode
  - YUV422 8-bit 4× line multiplexed mode
  - RGB 24-bit embedded sync mode
  - RGB 24-bit discrete sync mode
  - YUV444 24-bit embedded sync
  - YUV444 24-bit discrete sync mode
- Output:
  - YUV422 YUYV interleaved format
  - YUV420 Semi-planar format
  - RGB 24-bit interleaved format
  - YUV422 Semi-planer format
- Two 165 Mhz video capture instance of the VIP Subsystem
- VIP instance 0 is 16/24-bit channel, VIP instance 1 is 16 bit channel
- For each VIP instance, two pixel clock input domains are supported (Port A and Port B)
- Pixel clock input domain Port A supports one input data bus up to 24-bit
- Pixel clock input domain Port B supports one 8-bit input data bus
- Each VIP instance can be in dual 8-bit capture channels
- Embedded sync data interface mode supports single or multiplexed sources
- Discrete sync data interface mode supports only single source input
- The two pixel clock input domains can be individually configured in any combination of embedded sync or discrete sync
- Multiplexed data can only appear in embedded sync mode
- Where possible, blanking pixels that may contain embedded vertical ancillary data will be stored in a dedicated buffer per each video source
- Color Space Conversion (CSC)
- Chroma Down-Sampling
- Scaling
- Multiplexed data can not use CSC, Chroma Down-Sampling, and scaling

2.11.2 Functional Description

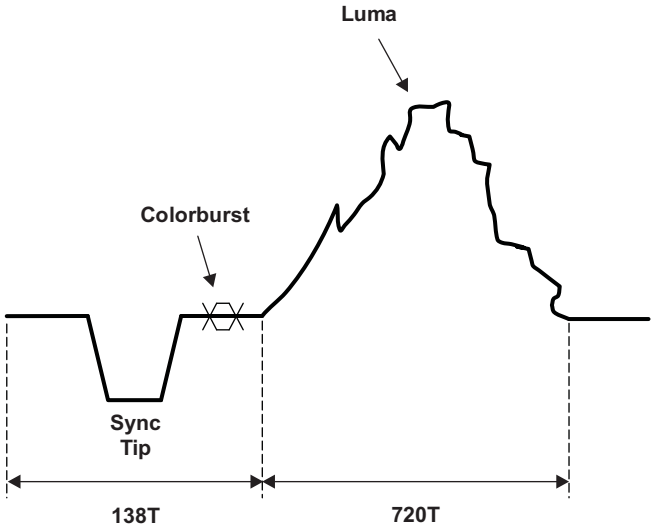
The video data is captured from the external video source by the VIP parser sub-block in the VIP block. The VIP parser then sends the captured data for further processing in the VIP block that includes color space conversion, scaling, chroma down-sampling, and finally writes the video data to external DDR memory. Color space conversion, scaling, and chroma down-sampling are all optional for the incoming stream.

The scaler and chroma down-sample module inside the VIP module is also used for the memory-to-memory operation, if they are not used in the capture mode.

2.11.2.1 Analog Video

A digital interface stream is based on analog video. The waveform for a line of NTSC analog video is shown in Figure 2-3.

Figure 2-3. NTSC Analog Video Waveform for One Horizontal Line



T is a time constant. For NTSC, T = 1/13.5 Mhz = 74 ns

2.11.2.2 Digitized Video

Digitized video is based on scan lines found in analog video. BT.601 uses various sync signals to specify when a new field and a new line starts. BT.656 and BT.1120 use sync words embedded in the data stream to specify the start of a field and the start of a line.

An image can be digitized into regions as shown in Figure 2-4.

With the capability to encode sync words inside the data stream, there is more flexibility for adding non-video related data, called Ancillary Data. Also, code words embedded in the digital stream can be used as a type of identifier for multiplexing several sources of video into one data stream. Such a multi-camera multiplex is useful in the digital security markets.

Figure 2-4. Digitized Video

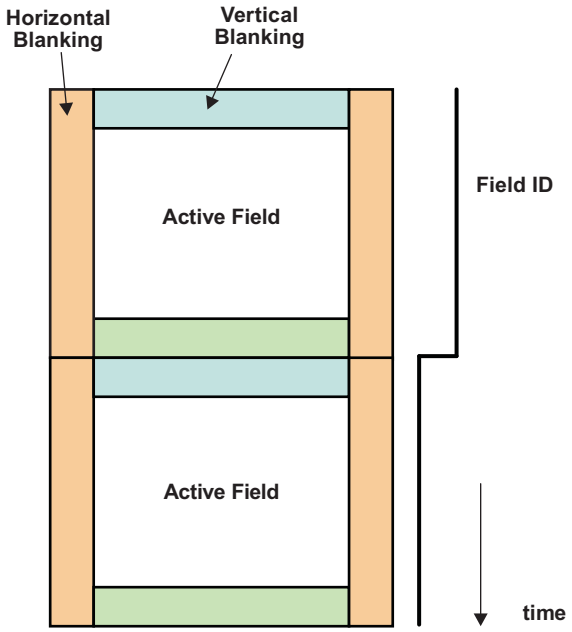


Figure 2-5 shows End-of-Active-Video (EAV) and Start-of-Active-Video (SAV) code words added to a video transmission. The period between the EAV and SAV is equivalent to horizontal blanking. The period between the SAV to the next EAV is active video or vertical blanking.

In the BT.656 or BT.1120 embedded code word scheme, three bits of the EAV/SAV code word are important: F (field), H (horizontal blanking), and V (vertical blanking).

Figure 2-6 shows the values of F, V, and H flags at different locations in the picture. The F flag specifies the state of the field ID for the picture. For progressive frames, F is always 0. The V flag specifies the vertical blanking areas. The H flag specifies the horizontal blanking portions of the picture.

Figure 2-5. Code Word Embedded Video Format

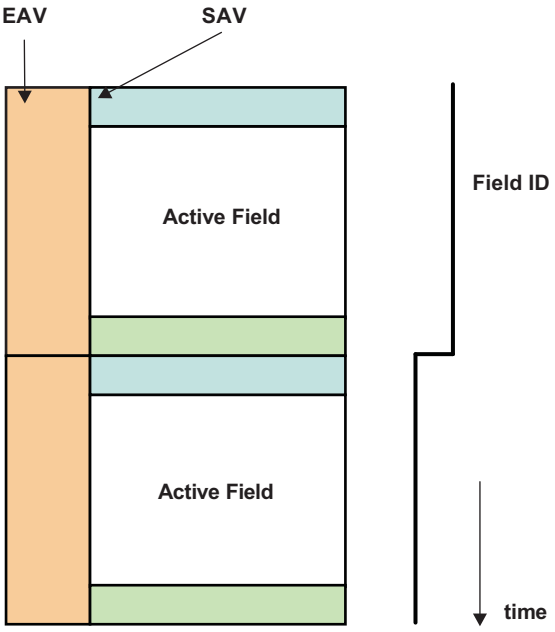
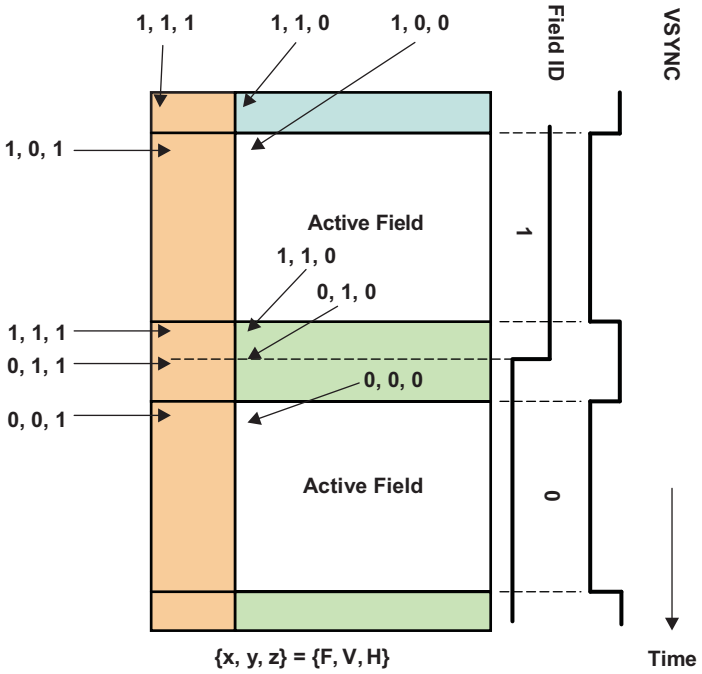


Figure 2-6. Digitized Video with F, V, and H Flags in EAV/SAV



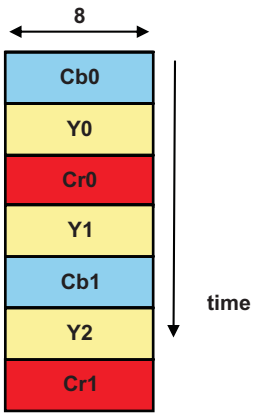
2.11.2.3 Input Data Interface

This section describes how the data (luma and chroma data for YUV422 format capture and R/G/B data for RGB888 format capture) is multiplexed for the interface modes.

2.11.2.3.1 8b Interface Mode

In 8b interface mode, the input pixels are multiplexed according to Figure 2-7. The chroma format is 4:2:2. Sites with Cb/Cr pixels are known as chroma sites. Those sites with Y pixels are known as luma sites.

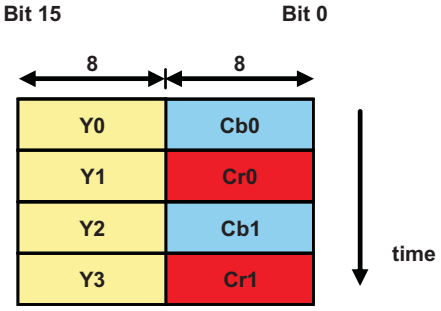
Figure 2-7. 8b Interface Discrete Sync Pixel Multiplexing



2.11.2.3.2 16b Interface Mode

In 16b interface mode, luma is on 8 bits of the data bus and Cb/Cr chroma pixels alternate on the other 8 bits of the data bus, as shown in Figure 2-8.

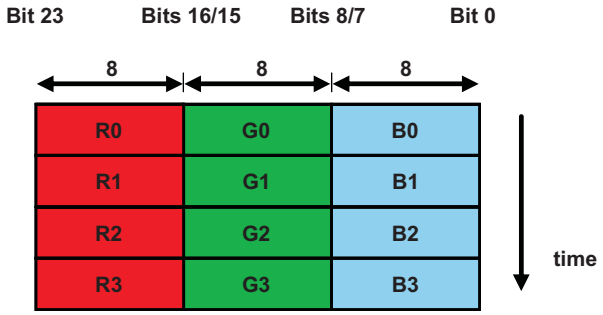
Figure 2-8. 16b Interface Discrete Sync Pixel Multiplexing



2.11.2.3.3 24b Interface Mode

In 24b interface mode, RGB data is sent. The three components are packed into the data bus and sent to the video port direct memory access (VPDMA), as shown in Figure 2-9. The 24-bit luma VPI client to the VPDMA carries all three components. This data is saved to the DDR in packed mode. That is, the three components are not separated by hardware.

Figure 2-9. 24b Interface RGB Discrete Sync



In 24b interface mode YUV4:4:4 data is also supported. The flow for YUV4:4:4 data is the same as RGB data (Figure 2-9). Bits [23:16] of the input data bus are placed on bits [23:16] of the data bus going to VPDMA. Bits [15:8] of the input bus are placed on bits [15:8] of the data bus going to VPDMA. Bits [7:0] of the input data bus, are placed on bits [7:0] on the bus going to VPDMA.

2.11.2.4 Input Ports and Sharing of the 24-Bit Data Bus

The VIP parser supports two independent pixel clock input domains: Port A and Port B. Port A supports a single 24-bit data bus at the instance level. Port B supports a single 8-bit data bus at the instance level. For the device, a single set of 24 device pins are used to drive both Ports A and B.

The two VIP instances are not identical from the chip level. VIP instance 0 is a 24-bit interface and VIP instance 1 is a 16-bit interface. The configuration for each device input port is described in Table 2-6. Each port can individually be configured as Discrete Sync or Embedded Sync.

Table 2-6. Valid Input Port Configurations

Port A	Port B
8 Bit	Off
16 Bit	Off

Table 2-6. Valid Input Port Configurations (continued)

Port A	Port B
24 Bit	Off
8 Bit	8 Bit
Off	8 Bit

2.11.2.5 Frame Buffers

The VIP/VPDMA supports frame buffers in DDR for active video. 4:2:2 data is always saved in packed pixel buffers. 4:2:0 data is saved in planar luma buffers and planar CbCr pair buffers. A luma frame buffer is a planar storage area. Each line is the width in pixels (1Byte/pixel) of the output picture size format. The frame buffer contains the number of active video lines in the output picture size format. A chroma pair frame buffer is planar storage of CbCr pixel pairs, with each pixel being a byte. For 4:2:0 storage, N lines in the output picture active video results in N/2 lines of CbCr pairs being stored.

2.11.2.6 Source Multiplexing

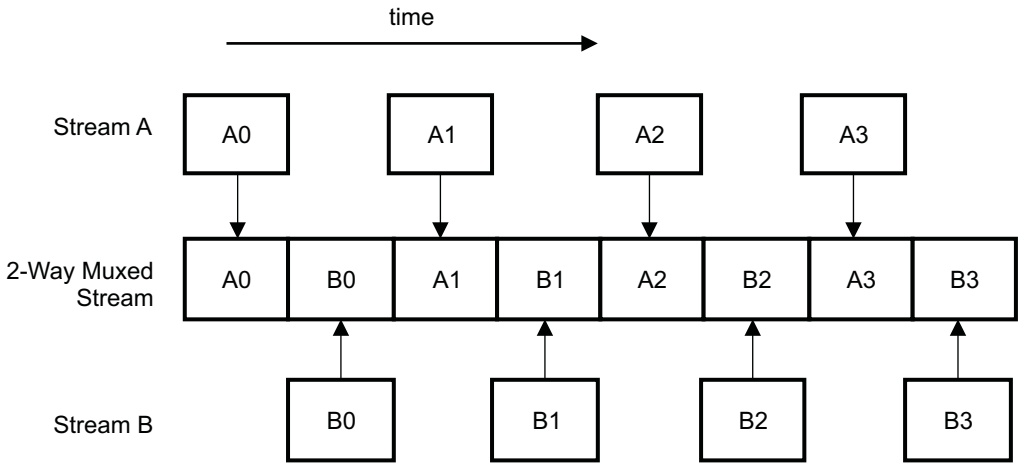
Some applications require multiple camera sources to be used at the same time. For example, video surveillance systems can record and display multiple camera sources. 16 camera sources are limited to be saved to DDR per pixel clock input domain.

2.11.2.6.1 2-Way Multiplexing

For 2-way multiplexing, two embedded sync streams are interleaved a pixel at a time as shown in Figure 2-10.

The sync codeword, FF-00-00-XY, is replicated in both source streams. In 2-way multiplexing, the sizes of both camera sources must be the same. Likewise, the vertical ancillary data size for both sources must be identical. However, the two streams are not necessarily sending the same pixel site in adjacent clock cycles.

Figure 2-10. 2-Way Multiplexing

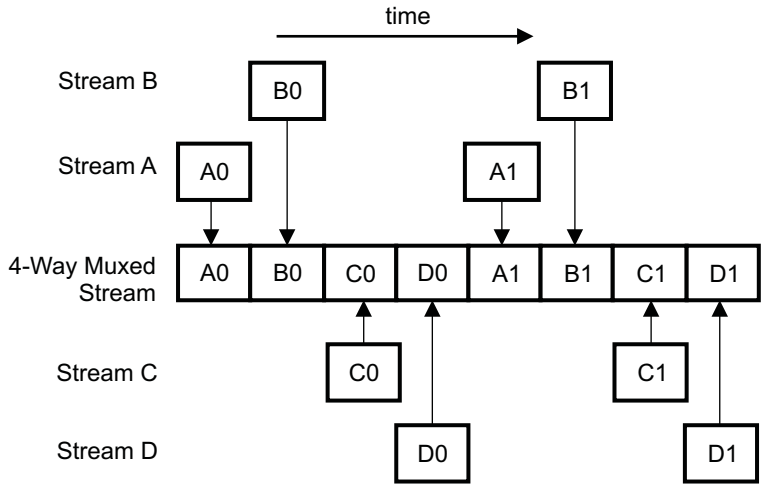


2.11.2.6.2 4-Way Multiplexing

For 4-way multiplexing, four embedded sync streams are multiplexed into one as shown in Figure 2-11.

The sync codeword is replicated in all four sources. Like 2-way multiplexing, the sizes of the four camera sources are the same and the sizes of the vertical ancillary data regions are the same. The four streams are not necessarily sending the same pixel site in adjacent clock cycles.

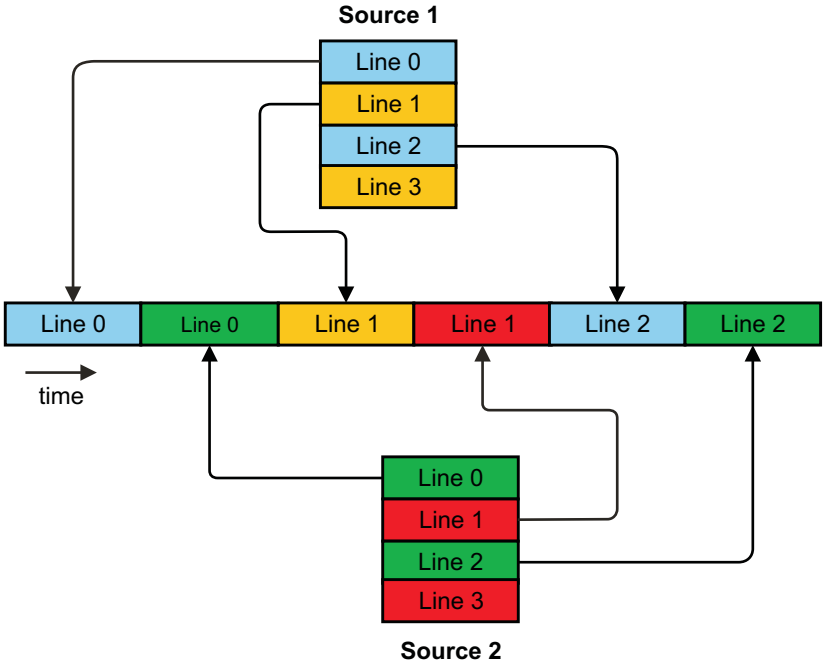
Figure 2-11. Example of 4-Way Multiplexing



2.11.2.6.3 Line Multiplexing

In line multiplexing, *n*-different sources are sent into the VIP a complete line at a time, using a modified version of embedded sync. An example of line multiplexing for two sources is shown in Figure 2-12.

Figure 2-12. Example of Line Multiplexing



2.11.2.7 Embedded Sync Mux Modes and Data Bus Widths

Valid combinations of embedded sync mux modes and data bus widths are listed in Table 2-7.

Table 2-7. Valid Embedded Sync Mux Mode and Data Bus Width Combinations

Data Bus Width	1× Mux	2× Mux	4× Mux	Line Mux
8 bit	Yes	Yes	Yes	Yes
16 bit	Yes	No	No	Yes
24 bit	Yes	No	No	No

2.12 Other Video Input Ports

2.12.1 Bypass Video Input Ports

2.12.1.1 Features

- Input: YUV422 YUYV Interleaved Non-Tiled memory

2.12.1.2 Functional Description

The HDVPSS has two independent bypass video input ports: bypass0 and bypass1. Both ports are used for the display purchase without any processing.

2.12.2 Secondary Video Input Ports

2.12.2.1 Features

- Input:
  - YUV420 Semi-Planar Tiled memory
  - YUV422 Semi-Planar Tiled memory
  - YUV420 Semi-Planar Non-Tiled memory
  - YUV422 Semi-Planar Non-Tiled memory
  - YUV422 YUYV interleaved Non-Tiled memory
- Output: YUV422 YUYV interleaved Non-Tiled memory (memory to memory operations)

2.12.2.2 Functional Description

The HDVPSS has two secondary video input ports: secondary0 and secondary1. Secondary0 is used for the memory-to-memory operations through scaler5. Secondary1 is used for SD display only through the SD_VENC.

Power Management

This chapter describes the power management in the device.

Topic **Page**

3.1 Introduction

Power management is an important aspect for most embedded applications. For several applications and target markets, there may be a specific power budget and requirements to minimize power consumption for both power supply sizing and thermal considerations. Additionally, lower power consumption results in more optimal and efficient designs from cost, design, and energy perspectives. This device has several means of managing the power consumption. This chapter discusses the various power management features.

Power consumed by semiconductor devices has two components: dynamic and static.

- Dynamic power is the power consumed to perform work when the device is in active modes (clocks applied, busses, and I/O switching), that is, analog circuits changing states. There are several features in design as well as user driven software control to reduce dynamic power consumption. The design features (not under user control) include a power optimized clock tree design to reduce overall clock tree power consumption and automatic clock gating in several modules when the logic in the modules is not active.
- Static power is essentially a function of the "leakage", or the power consumed by the logic when it is not switching or is not performing any work. Leakage current is dependent mostly on the manufacturing process used, the size of the die, and so on. Leakage current is unavoidable while power is applied and scales roughly with the operating junction temperatures. Leakage power can only be avoided by removing power completely from a device or power domain.

Refer to the *TMS320DM816x/TMS320C6A816x/AM389x Power Estimation Spreadsheet* (SPRABK3) for details on how to model power consumption for a specific user's application.

3.2 Power Management

The on-chip power, reset, and clock management (PRCM) module manages the switching on and off of the power supply to the device modules. To minimize device power consumption, the power to the modules can be switched off when they are not in use. Independent power control of sections of the device allow the PRCM module to turn on and off specific sections of the device without affecting the others. Software controlled module level clock gating reduces both clock tree and module power by basically disabling the clocks when the modules are not being used. Clock management also allows clocks to be slowed down to reduce the dynamic power.

Table 3-1 describes the static power management features.

3.2.1 Power Domains

A power domain is a section (that is, a group of modules) of the device with an independent and dedicated power manager. A power domain can be turned on and off without affecting the other parts of the device.

The AVS and 1V constant voltage domains have multiple power domains. All other voltage domains have only always-on power domain. Within the 1-V AVS and 1-V constant voltage domains, each power domain, except for the always-on domain, has an internal power switch that can isolate the power supply rail from that domain. At power-up, all domains, except always-on, come-up as power gated. Since there is an always-on power domain in each voltage domain, all power supplies are expected to be ON all the time (as long as the device is in use).

Refer to the Power Management section of the *Power, Reset, and Clock Management (PRCM) Module* chapter for the Power-Down and Power-Up Sequences.

Table 3-1. Static Power Management Features

Power Management	Description	Feature
	Clock Management	
PLL bypass and power-down	PLLs can be powered-down and run in bypass mode when not in use.	Saves power when PLL is in powered down mode.
Module clock ON	Module clocks can be turned on/off without requiring reconfiguring the registers.	When the module clock is turned OFF, this reduces the module's dynamic/switching power consumption down to zero.
	Memory Power Management	
Memory power management modes	The memory in each functional block is assigned to both a voltage domain and a power domain. The modes are ACTIVE and SHUT-DOWN.	In order to reduce SRAM leakage, many SRAM blocks can be switched from ACTIVE mode to SHUT-DOWN mode.
	System/Device Sleep Management	
Standby modes	All switchable power domains are in off state, the Cortex-A8 is in lowest frequency of operation executing in an idle process loop and all functional blocks not needed for a given application are clock gated.	
Deep sleep modes	The mode is enabled to further eliminate active power when memories are not being used.	
	Peripheral I/O Power Management	
Video DACs power down	Video DACs can be powered down.	Video DACs can be powered down; results in power save.
DDR2/DDR3 data and command macros	DDR data and command macros can be powered down to reduce power.	DDR data and command macros can be powered down; results in power save.
USB PHY power down	USB PHY can be powered down.	USB2.0 PHY can be powered down; results in power save.
HDMI PHY power down	HDMI PHY can be powered down.	HDMI PHY can be powered down; results in power save.
PCIe PHY low power	PCIe PHY low-power state	PCIe PHY is in low-power state when PCIe is disabled.
SATA PHY low power	SATA PHY low-power state	SATA PHY is in low-power state when SATA is disabled.

3.2.2 Memory Power Management

The device memories offer three different modes to save power when memories are not being used, see Table 3-2.

Table 3-2. Memory Power Management Modes

Mode	Power Saving	Wakeup Latency	Memory Contents
Light Sleep (LS)	~60%	Low	Preserved
Deep Sleep (DS)	~75%	Medium	Preserved
Shut Down (SD)	~95%	High	Lost

The device provides a feature that allows software to put the chip level memories (C674x L2, OCMC RAMs) in any of the three (LS, DS, or SD) modes. There are control registers in the Control Module to control the power down state of C674x L2, OCMC RAM0, and OCMC RAM1. There are also status registers that are used during power-up to check if memories are powered-up. Status registers are useful when coming out of SD and DS mode as the wakeup latency is significant. Wakeup latency associated with DS and SD mode are longer due to following reasons:

- In SD and DS modes, power to the memory periphery is cut-off and there will be ramp-up time required for the memory supply when memory is coming out of these modes.
- In SD mode, even the array power is cut-off and wakeup latency depends upon the ramp-up time required for the memory supply for array and periphery both.
- To avoid large current sink during memory power-up, hardware sequences the power-up of memory instances in such a way that all instances are not powered-up at the same time. This adds to the wake-up latency.

Memories inside switchable domains go to the SD state whenever the power domain goes to the OFF state. Memories come back to a functional state along with the domain power-up.

3.2.2.1 OCMC RAM Power Down Sequence

1. Software writes to module enable/disable register in PRCM to request disabling of OCMC RAM.
2. PRCM initiates IDLE protocol with OCMC RAM module.
3. OCMC RAM completes pending read/write transactions and enters in IDLE mode and gates-off all internal clocks.
4. OCMC RAM sends acknowledge back to PRCM.
5. Software writes to Control Module registers to put OCMC RAMs in one of the power down modes (LS, DS, or SD).

3.2.2.2 OCMC RAM Power Up Sequence

1. Software writes to Control Module registers to bring out OCMC RAM from the power down state.
2. Software writes to module enable/disable register in PRCM to bring OCMC RAM in functional mode.
3. PRCM de-asserts IDLE request and waits for the ack.
4. OCMC RAM un-gates its internal clocks and sends acknowledge to PRCM.

3.2.2.3 C674x L2 Power Down Sequence

Note that the C674x L2 is powered down only when Active domain is OFF. When Active domain is turned off, the C674x (including L1s) gets power gated automatically, but L2 is still powered. The sequence to power down the C674x L2 is:

1. Software writes to Control Module register to put L2 in one of the power down modes (LS, DS, or SD).

3.2.2.4 C674x L2 Power Up Sequence

The C674x L2 needs to be powered-up before Active domain is powered up. The sequence to power up the C674x L2 is:

1. Software writes to Control Module register to bring the C674x L2 out of power down state.
2. Software can then turn on Active domain.

3.2.3 Voltage Management

The ARM Cortex-A8 in conjunction with the SmartReflex™ sensors control the voltage scaling (that is, switching the voltage in discrete steps or in a continuum within a range of possible values) of the power sources of the device. This allows controlling the device power consumption according to the defined performance criteria.

3.2.3.1 Voltage Domains

A voltage domain is a section of the device supplied by a dedicated voltage source (that is, an internal LDO or external switch mode power supply [SMPS]). When a voltage domain is controlled by the PRCM module, a dedicated voltage manager is associated. The voltage manager allows regulation of the voltage level of the source for the voltage domain, independently of other voltage domains of the device.

3.2.3.2 SmartReflex Adaptive Voltage Scaling

SmartReflex™ power and performance technologies is a power-management technique for control of the operating voltage of a module to reduce both leakage and dynamic power consumption. With SmartReflex, power-supply voltage is dynamically adapted to silicon performance (based on the temperature-induced real-time performance of the device). A comparison of these predefined performance points to the real-time on-chip measured performance determines whether to raise or lower the power-supply voltage. SmartReflex achieves the optimal performance/power trade-off for all devices across the technology process spectrum and across temperature variations.

Adaptive Voltage Scaling (AVS) is a power-management technique based in SmartReflex that is used for automatic control of the operating voltages of the device to reduce active power consumption. Based on the silicon process and temperature, the SmartReflex modules guide software in adjusting the Core Logic Voltage Domain supply voltages within the voltage range. The Vmin to Vmax voltage range is split into equidistant steps (8, 16, or 32 are recommended). The advantage of using more steps is that it enables the system to more closely approach the optimum operating voltage and results in a flatter leakage power curve (see Figure 3-2 and Figure 3-3).

AVS achieves the optimal performance/power trade-off for all devices across the technology process spectrum and across temperature variation. The device voltage is automatically adapted to maintain performance of the device. This ensures optimal power consumption. AVS occurs continuously and in real time, helping to minimize power consumption in response to changing operating conditions. Not using AVS will result in a considerable increase in device power consumption and the voltage applied to the device may be too high for devices from the hot end of the process distribution which can lead to hold timing violations, and hence incorrect operation. Therefore, AVS is required for reliable operation.

TI recommends using a fault-tolerant power supply design to protect against over-current conditions. Refer to the *TMS320DM816x/TMS320C6A816x/AM389x Power Estimation Spreadsheet* (SPRABK3) for AVS Disable data to aid in design of robust power supplies that may withstand momentary AVS control failure.

Figure 3-1, Figure 3-2, and Figure 3-3 show examples of AVS power management as compared to a constant voltage system, and the resulting leakage power curves.

Figure 3-1. Constant Voltage

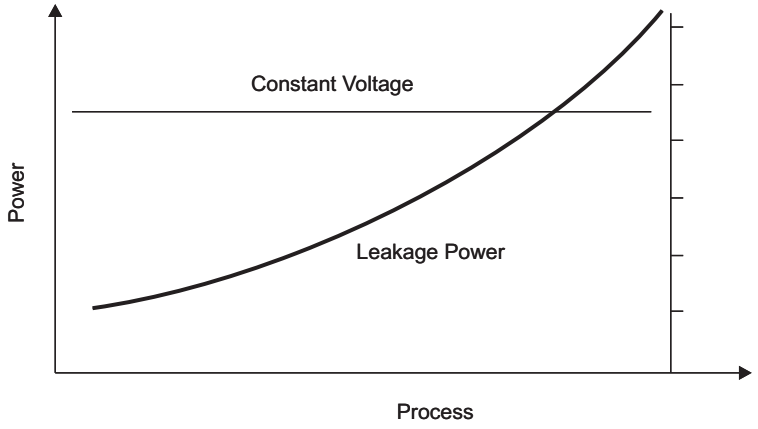


Figure 3-2. AVS Example – 8 Steps

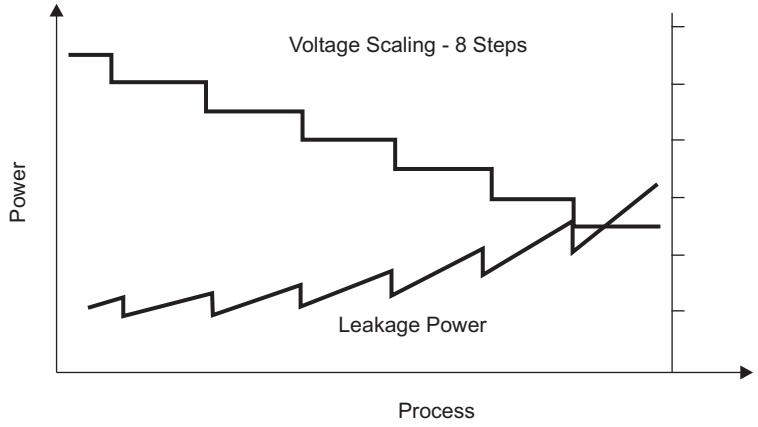
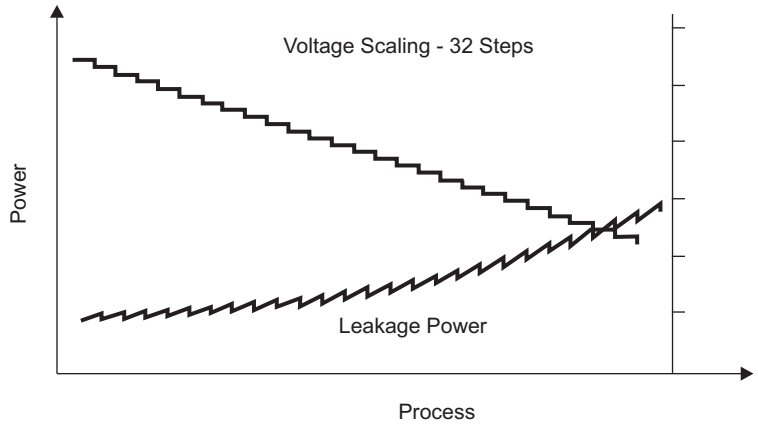


Figure 3-3. AVS Example – 32 Steps



3.2.3.2.1 General SmartReflex Operation

The device supports a class of SmartReflex technologies that enables dynamic voltage adjustments by software. The SmartReflex module provides an interrupt to help the CPU adjust the supply voltage. The SmartReflex module also provides a parameter that can be sampled to know the magnitude and direction of the voltage adjustment required. When an interrupt reaches the CPU, software will service the interrupt and adjust the voltage supply accordingly.

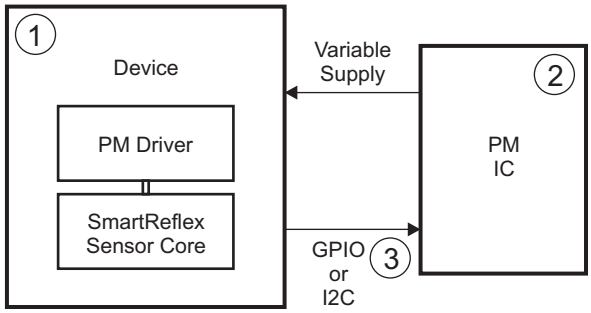
3.2.3.2.2 Device-Specific Software Driver Operation

Texas Instruments provides a SmartReflex driver to insure proper implementation of SmartReflex AVS closed-loop operation. The Power Management (PM) driver basically operates as a performance monitor where environmental changes may require adjustment of the voltage to maintain device performance. The SmartReflex module generates a system interrupt when a measured average performance parameter goes beyond programmed limits. The driver monitors this interrupt and then determines the appropriate corrective action - whether to raise or lower the AVS power supply level. The communication link between the host processor and the external regulators (Power Management IC or PMIC) is a system-level decision and can be accomplished using GPIOs or I2C. The major SmartReflex system components are shown in Figure 3-4 and the basic operation is:

1. The Device PM Driver responds to SmartReflex Sensor interrupts and adjusts power supply up or down via commands to the PMIC.
2. The PMIC is commanded to raise or lower the Device supply voltage. The PMIC driver responds to voltage adjustment commands by setting the PMIC output voltage to the voltage step nearest and above the level requested. The system designer decides how many voltage steps to use (8, 16, or 32 are recommended) and creates a table of voltages in the desired range that includes that number of voltage steps.
3. Communication between the Device and the PMIC is either via GPIO or I2C. The commands sent to the PMIC are continual, not just at startup. This allows compensation for temperature changes over time.

The Power Management driver uses basic API's to exchange information between the device and PMIC for voltage level changes. Refer to the Power Management Software Guide for details on using the Power Management driver.

Figure 3-4. SmartReflex System Block Diagram



3.2.4 Clock Management

3.2.4.1 Module Clock ON/OFF

The module clock on/off feature allows software to disable clocks to module individually, in order to reduce the module's dynamic/switching power consumption down to zero. This device is designed in full static CMOS; thus, when a module clock stops, the module's state is preserved and retained. When the clock is restarted, the module resumes operating from the stopping point.

NOTE: Stopping clocks to a module only affects dynamic power consumption, it does not affect static power consumption of the module or the device.

The power, reset, and clock management (PRCM) module controls module clock gating. If a module's clock(s) is stopped while being accessed, the access may not occur, and it can potentially result in unexpected behavior. The PRCM provides some protection against such erroneous conditions by monitoring the internal bus activity to ensure there are no accesses to the module from the internal bus, before allowing module's internal clock to be gated. However, it is still recommended that software must ensure that all of the transactions to the module are finished prior to disabling the clocks.

NOTE: To preserve the state of the module, the module state in the PRCM must be set to Disable.

In this state, the module reset is not asserted and only the module clock is turned off. Furthermore, special consideration must be given to clock on/off.

Additionally some peripherals implement additional power saving features by automatically shutting off the clock to components within the module, when the logic is not active. This is transparent to the user, but reduces overall dynamic power consumption when modules are not active.

3.2.4.2 Module Clock Frequency Scaling

This device supports static frequency scaling using dividers in the FA-PLL and in the PRCM. Divide ratios supported by these dividers are detailed in the Clocking section. Frequency change should be made when the corresponding clock domain is completely idle. Additionally, some modules might also have internal clock dividers. Reducing the clock frequency reduces the dynamic/switching power consumption, which scales linearly with frequency.

3.2.5 Standby Mode

The lowest power mode of the device is Standby Mode. In Standby Mode, all switchable power domains are in an off state, the Cortex-A8 is in the lowest frequency of operation executing in an idle process loop and all functional blocks not needed for a given application are clock gated.

Exit from standby is performed by Cortex-A8 polling, loop count time out, or responding to an interrupt from a peripheral. The clock must be maintained to those peripherals that generate the interrupt.

3.2.6 Additional Peripheral Power Management Considerations

This section lists additional power management features and considerations that might be part of other chip-level or peripheral logic, apart from the features supported by the core, PLL controller (PLLC), and power, reset, and clock management (PRCM) module.

3.2.6.1 USB PHY Power Down Control

The USB modules can be clock gated using the PRCM; however, this does not power down/clock gate the PHY logic. You can put the USB PHY in the lowest power state, when not in use, by writing to the PWRDN bits in the USB control register (USB_CTRLx) of the Device Configuration Registers.

3.2.6.2 HDMI PHY Power Down Control

The HDMI PHY supports a standby power mode that yields significant power reduction during periods in which the PHY is not used. In applications in which the HDMI is not used at all, the power supply to the HDMI PHY must be connected to 1.0V source.

3.2.6.3 Video DACs Power Down Control

In applications in which the Video DACs are not used at all, the power supply to the Video DACs can be powered down by configuring the HD_DAC_CTRL and SD_DAC_CTRL registers of the Device Configuration Registers.

3.2.6.4 DDR2/DDR3 Memory Controller Clock Gating

The DDR2/DDR3 memory controller supports different methods for reducing its power consumption including clock gating.

Additionally, the DDR2/DDR3 memory controller DLL, PHY, and the receivers at the I/O pins can be disabled.

For details of power management in the DDR2/DDR3 memory controller, see the *DDR2/DDR3 Memory Controller* chapter.

3.2.6.5 Floating Input Pins

In general, you should ensure that all input pins are always pulled to a logic-high or a logic-low voltage level.

A floating input pin can consume a small amount of I/O leakage current. The I/O leakage current can be greatly multiplied in the case of several floating inputs pins. This device includes internal pull-up and pull-down resistors that prevent floating input pins. For unused input pins, the internal pull resistor should be enabled, or an external pull resistor should be used, to prevent floating inputs. For more details, refer to the data manual sections on Pin Multiplexing Control and How to Handle Unused Pins.

3.2.6.6 PCIe PHY Power Down Control

If the PCIe module has not been enabled, the SERDES/PHY will be kept in a low-power state.

3.2.6.7 SATA PHY Power Down Control

If the SATA module has not been enabled, the SERDES/PHY will be kept in a low-power state.

DMM/TILER

This chapter describes the Dynamic Memory Manager (DMM) and its Tiling and Isometric Lightweight Engine for Rotation (TILER) submodule.

Topic **Page**

4.1 Introduction

4.1.1 Overview

This section describes the Dynamic Memory Manager (DMM) and its Tiling and Isometric Lightweight Engine for Rotation (TILER) submodule.

As shown in Figure 4-1, the DMM is located in front to the SDRAM controllers and thus interfaces to the memory accesses generated by all the initiators.

The dynamic memory manager (DMM), is a module aimed at managing – in a broad sense – various aspects of memory accesses such as:

- Initiator-indexed priority generation
- Multizone SDRAM memory interleaving configuration
- Block object transfer optimization – tiling and sub-tiling
- Centralized low-latency page translation – MMU-like feature

The dynamic qualifier for memory management highlights the software configurability – and, hence, the run-time nature – of the four aspects of memory management handled by the DMM.

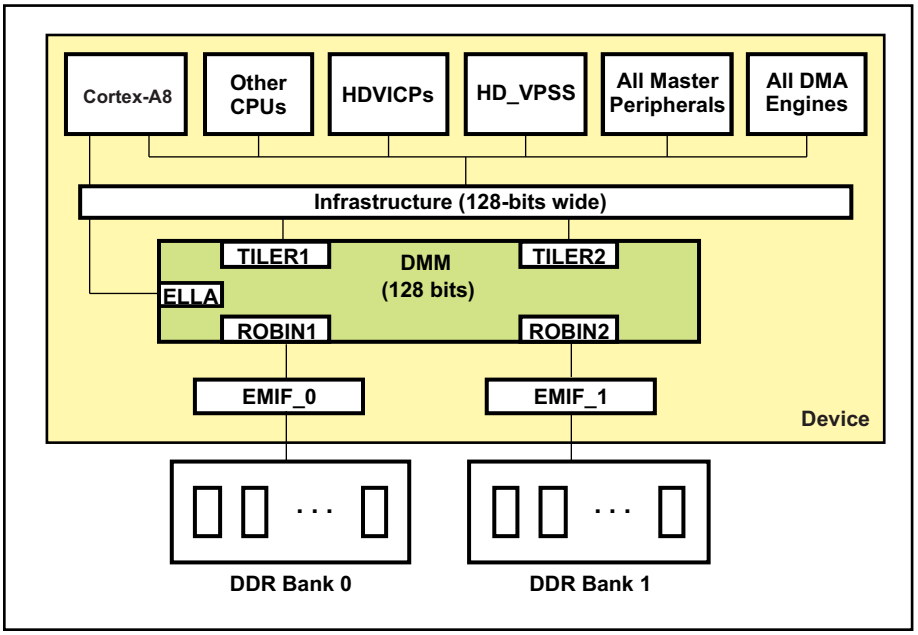
On a functional perspective, the role of the DMM is to:

- add initiator-based priority to any incoming requests
- perform to-and-fro tiling conversions of "tiled" requests
- provide an optional low-latency page-based translation to handle memory fragmentation - MMU
- distribute the traffic on both the attached memory controllers according to the interleaving configuration

The TILER is a submodule within the DMM aimed at efficient handling of two dimensional data, such as video/graphics accesses for HDVICP2, by the use of tiled format.

- optionally managing the memory fragmentation and zero-copy physical frame buffers swapping through a page-grained translation
- making on-the-fly, zero overhead transforms, such as 90°, 180°, or 270° rotations, with either a horizontal or vertical reflection

Figure 4-1. DMM Integration



4.1.2 Features

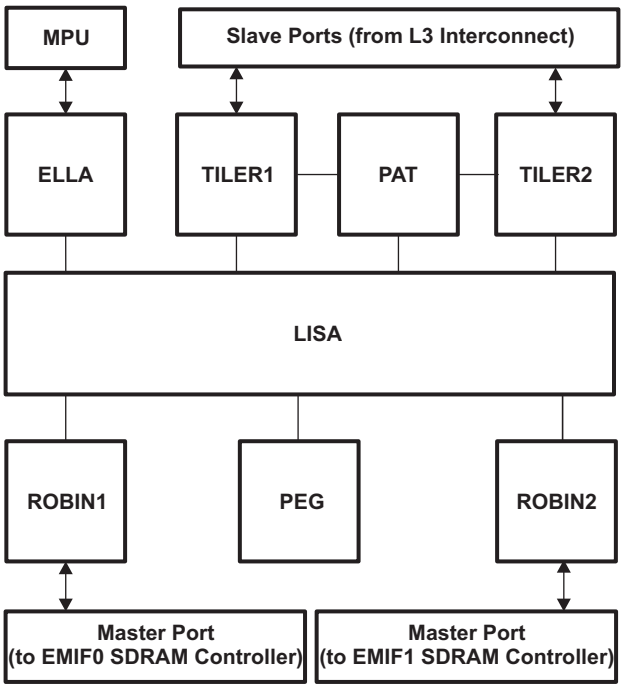
- Special low latency interconnect port. ELLA, only for Cortex-A8 accesses.
- Ability to interleave the DDR data between two EMIF banks, using the programmable multi-zone DRAM memory mapping . This increases the memory throughput by a factor of 2. Up-to four unique memory sections supported.
- Programmable Initiator based request priority extension, for up to 16 Initiator groups.
- Support for address translations of tiled data, on a 4KB page granularity using the PAT. This helps manage memory fragmentation.
- Two internal address lookup tables (LUT), each with 256x128 entries. Four refill engines to program the LUTs, with automatic synchronized reload.
- Supports up to 4 unique PAT views.

4.1.3 Functional Block Diagram

Figure 4-2 shows the DMM macro architecture. The DMM consists of six blocks:

- A Priority Extension Generator (PEG) to generate priorities required by the SDRAM controller, note that these priorities are not used in the DMM
- One Extra Low Latency Access (ELLA), having its own interconnect slave port, for providing lower latency accesses to the memory
- A Local Interconnect and Synchronization Agent (LISA) to synchronize all DMM subsystems and provide access to their configuration registers
- A Physical Address Translator (PAT) for managing the memory fragmentation
- Two Re-Ordering Buffer and Initiator Nodes (ROBIN), having each their own interconnect master port, to initiate requests to the SDRAM controller and allow tiled data, tiled response and split response reconstruction. The ROBIN block is only managing the re-ordering buffer and performing the data re-ordering due to the orientation.
- Two Tiling and Isometric Lightweight Engine for Rotation (TILER), having each their own interconnect slave port, for converting requests to-and-fro between the input virtual addressing mode and the output physical tiled addressing mode. Note that, the tiling conversions of requests, write data and responses is fully performed in the TILER blocks.

Figure 4-2. DMM Block Diagram



4.1.4 Terminologies and Acronyms Used in this Document

The following is a brief explanation of some terms used in this document:

bpp— Bits per pixel

DMM— Dynamic Memory Manager

ELLA— Extra Low Latency Access

GB, GiB— Both imply Giga Byte

Initiator— A node inside the Device and is either a CPU, peripheral or DMA engine, which can be internal bus Master. Each initiator is identified by a ConnID (Connection ID). With a limit of only 16 ConnIDs, some of the Initiators are grouped together with same ConnID.

Interlaced— Qualifier for accesses skipping one line every line

IVA— Image Video Accelerator. Also called HDVICP2, IVA_HD.

LISA— Local Interconnect and Synchronisation Agent

KB, KiB— Both imply Kilo Byte

LUT— Look Up Table

MMU— Memory Management Unit

MPU— Main Processing Unit. For this device, it is Cortex-A8

PAT— Physical Address Translator

PEG— Priority Extension Generator

Progressive— Qualifier for line-by-line accesses

ROBIN— Re-Ordering Buffer and Initiator Node

Tiled access— 1D or 2D Access to the tiled area, where the image data is read and written in two dimensional format, to improve the efficiency of accesses of 2D accesses, example image macro block. TILER Tiling and Isometric Lightweight Engine for Rotation 1D access A simple linear read or write access request. The DMM responds with read/write from/to the contiguous memory starting with address specified in the request.

2D access— HDVICP2 and HD_VPSS, can generate a special access to 2D image buffers, with read/write request, with Height and Width information. The DMM-TILER decodes the access type based on the Height, width and address, and responds with read/write from/to data in the physical memory, which has been co-located, with sub-tile granularity.

4.2 Architecture

4.2.1 DMM Functional Description

4.2.1.1 Priority Extension Generator (PEG)

The priority extension generator (PEG) is a dynamic software-programmable initiator-indexed table of priorities. Its unique role is to generates priorities for each access forwarded by the DMM to the SDRAM controller. For initiators that do not generate their own per-transaction priority, the priority value that is programmed in the table, will be assigned to all SDRAM accesses generated by that initiator. These priorities are not used by the DMM, for internal arbitrations.

The 16 priority entries are software-programmable with DMM_PEG_PRIO0 (for the first eight entries of the ConnID table) and DMM_PEG_PRIO1 (for the last eight entries) registers and set on a 3-bit scale, ranging from values 0-7, with Priority=0, being the highest priority.

When a request is passed to the SDRAM controller, the priority from the corresponding DMM_PEG_PRIOx field for an initiator is passed to SDRAM. For example, an access from the C674x DSP is sent to SDRAM with the priority captured in DMM_PEG_PRIOn. Note that the priority does not impact the behavior in DMM itself. It is passed to the SDRAM controller and the controller may use it in the prioritization of requests. However, for HD_VPSS, the peripheral itself generates a priority. The DMM_PEG_PRIOx field for HD_VPSS is bypassed and the priority indicated by HD_VPSS access is sent to SDRAM controller.

4.2.1.2 ELLA

The Extra Low Latency Access (ELLA) of the DMM is a simple interface port for all accesses made by Cortex-A8. As the name suggests, this interface is simplified to reduce latency to the request from Cortex-A8. In this respect, the ELLA block:

- Is limited to 1D bursts only
- Is not capable of performing tiling conversions
- Does not interact with the PAT block

The ELLA block main role is to split incoming requests at DMM atomic unit boundaries to ensure that requests sent to the SDRAM controller fit in a single SDRAM page. More precisely, the ELLA block is responsible for:

- Allocating an internal response context to timely generate the appropriate responses
- Splitting the incoming requests at DMM atomic section boundaries
- Requesting internal buffer allocation in the appropriate ROBIN
- In case of a write request, allocating and updating an internal write context to subsequently direct the incoming write data into the relevant re-ordering buffer

NOTE: In the device, Cortex-A8 accesses in the system address space of 8000 0000h-FFFF FFFFh, are only routed through ELLA port. Accesses made by Cortex-A8, in the all four tiled modes, including Paged mode, will be routed through the TILER ports, thus not getting benefit of lower latency of the ELLA port.

ELLA sub-module is not software configurable.

4.2.1.3 LISA

The Local Interconnect and Synchronisation Agent (LISA) is a hardware interconnect module, aimed at setting priorities, managing tags and memory mapping. LISA maps the system addresses on the incoming DMM requests to the SDRAM addresses, as per the LISA sections programming.

LISA interconnect routes:

1. ELLA and TILER requests on the ROBIN initiator nodes.
2. ELLA and TILER write data on the ROBIN write buffers.

3. ROBIN read data to the relevant TILER initiators or to ELLA initiator.

The LISA block registers are DMM_LISA_MAP_i (i = 0 to 3) and DMM_LISA_LOCK.

LISA block manages all the incoming requests to ELLA, TILER and PAT and translates then to requests to Registers and ROBIN ports. Highest priority is given to accesses coming to the low-latency port (ELLA). When both DMM-ROBIN ports are in use, LISA module interleaves the two ports at a programmable boundary from 128 Bytes or more, as programmed in the DMM_LISA_MAPn registers. See sub-section on Section Mapping for details about LISA sections programming.

4.2.1.3.1 LISA LOCK

The DMM_LISA_LOCK register is used to lock the configuration once set. If written to one, the LOCK bit prevents further writes to all DMM_LISA_MAP_i registers. The LOCK itself cannot be written back to 0. A reset is required to re-program the sections.

It is expected that LISA MAP is set up once during system configuration and not changed subsequently. The LOCK feature helps accomplish this and is expected to be set once the configuration is done.

4.2.1.4 PAT, Physical Address Translator Engine

The physical address translation engine (PAT) of the DMM is composed of two 32-k entry physical address translation vector table and a set of four refill engines. The refill engine is a specialized DMA aimed at refilling the content of the physical address translation table.

The address translation mechanism is only available when the incoming request is hitting a page mode or tiled mode container, that is, when the incoming address is targeting the TILER or its aliased view in the system addressing space. Otherwise the physical address translation logic is bypassed so that the resulting physical address corresponds to the input address.

The PAT is supporting multiple address translation schemes, called views, which can be bound to one or more initiator through a view mapping mechanism.

The usage of the Refill Engines is described in detail in the later section.

4.2.1.4.1 PAT Views

A PAT view defines the kind of physical address translation to perform for each tiled mode accesses (page, 8-bit, 16-bit and 32-bit). Each mode of each PAT view can be configured to either use PAT Direct Access Translation or PAT In-Direct Access Translation.

The PAT supports up to four unique Views, using the DMM_PAT_VIEW_MAP[0..3] Registers.

4.2.1.4.2 PAT View Mappings

The 16 groups of initiators of the Device, each identified by their ConnID, share a set of 4 PAT views. The connection from an initiator to a PAT view is made through the DMM_PAT_VIEW register. The register DMM_PAT_VIEW0 is used for the first eight initiator groups and DMM_PAT_VIEW1 for the second eight. Thus each initiator can choose one of the four configurable PAT views in the system.

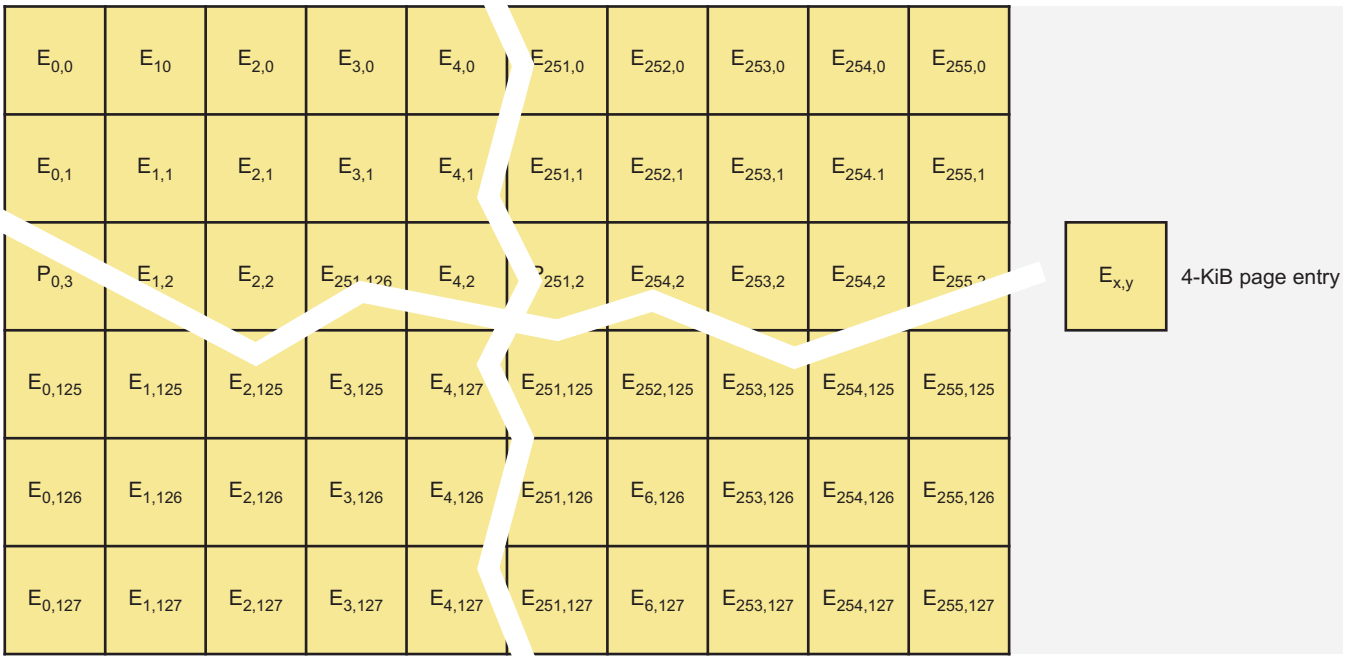
4.2.1.4.3 PAT View Map Base Address

The PAT view map base address is defining the base address of all PAT translated addresses. The bit 31 of all PAT translated address is set to BASE_ADDR. In the Device, the bit 31 should be set to 1, which corresponds top 2GB assigned for external SDRAM in the system memory map. Hence the addresses translated by PAT range in addresses 8000 0000h-FFFF FFFFh.

4.2.1.4.4 PAT - Look Up Tables (LUTs)

The PAT contains two LUTs, each of has 32K (256 × 128) entries. This geometry corresponds to that of the Virtual TILER container. So incoming address maps the actual entry in the LUT. The PAT shall then translate to a physical memory mapped to any 4K page the DDR. Each entry of the PAT address corresponds to the page in the DMM container that has the same location. For example - The entry (74, 42) in the table corresponds to the page (74, 42) in any DMM container.

Figure 4-3. DMM Look-Up Table



Each of this entry points to a 4K page in the memory. Hence a total of 128 MB of tiled memory can be mapped using one LUT. With upto 2G of DDR space, supported in the Device, each LUT entry is 19 bits wide.

The PAT uses the LUTs only in case of PAT In-Direct Access Translation.

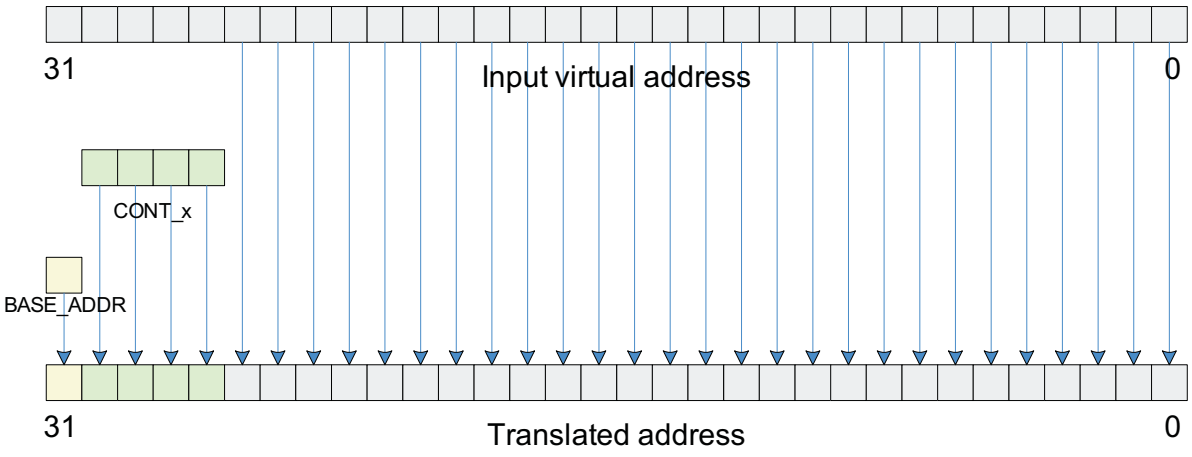
4.2.1.4.5 PAT Direct Access Translation

The container-grained translation is named the direct access. In this mode, the translation vector is directly given by the CONT_x field, in DMM_PAT_VIEW_MAP register, corresponding to the accessed mode(8-bit/16-bit/32-bit/paged mode).

With PAT Direct Access translation, the 128 MB virtual containers for all the four modes, 8-bit, 16-bit, 32-bit, and paged modes can be mapped to their unique System Address, by defining their base addresses.

Figure 4-4 describes the actual translation that happens.

Figure 4-4. DMM PAT Direct Access Translation

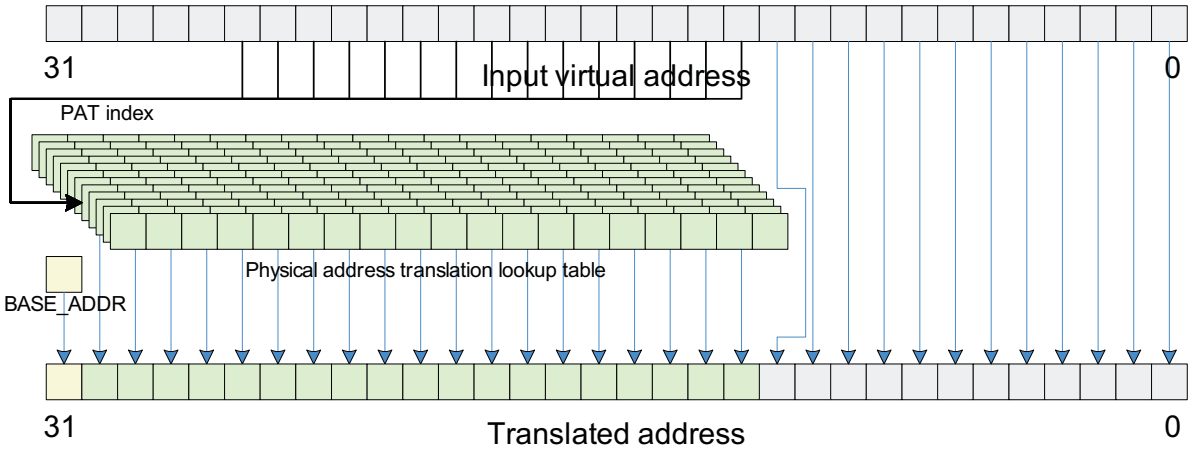


4.2.1.4.6 PAT In-Direct Access Translation - Paged Mode Access

DMM_PAT_VIEW[0..3] specify, whether to do Direct or In-direct access translation. Per each of the 8-bit,16-bit ,32-bit and paged container mode, user specifies which LUT to use for address lookup, by using CONT_x field, in DMM_PAT_VIEW_MAP registers. Then bits [19:12] of input address specify the X co-ordinate and bits [26:20] of input address specify the Y co-ordinate LUT entry. The PAT replaces bits[30:12] of input address, with 19-bit value specified in LUT entry.

This translation happens on the 4K page boundary and hence the lower 12 bits are not translated. Figure 4-5 describes the actual translation that happens.

Figure 4-5. DMM PAT In-Direct Access Translation



4.2.1.4.7 PAT In-Direct Access Translation - 8-bit, 16-bit, 32-bit Mode Access

DMM_PAT_VIEW[0..3] specify, whether to do Direct or In-direct access translation. Per each of the 8-bit,16-bit ,32-bit and paged container mode, user specifies which LUT to use for address lookup, by using CONT_x field, in DMM_PAT_VIEW_MAP registers.

How the 26 LSBs of this incoming virtual address is interpreted by the PAT is different for 8-bit,16-bit ,32-bit mode accesses, as compared to the paged mode access described in the preceding section. Please refer the TILER section describes the rationale behind decoding the address this way.

Virtual address decoding in 8-bit mode access mode is:

- Bits 0:5 - 6 bits offset into the horizontal line of the page
- Bits 6:13 - 8 bits, that select horizontal page in the tiler container as well as the X co-ordinate of the LUT table
- Bits 14:19 - 6 bits offset to select the line inside the page
- Bits 20:26 - 7 bits, that select vertical page in the tiler container and also Y co-ordinate of the LUT table
- Bits 27:31 - In 8 bit mode, their value is binary (01100), that is address in range 6000 0000h-67FF FFFFh

Virtual address decoding in 16-bit mode access mode is:

- Bit 0 - is always 0, as accesses are at-least 16-bit aligned
- Bits 1:6 - 6 bits offset into the horizontal line of the page
- Bits 7:14 - 8 bits, that select horizontal page in the tiler container as well as the X co-ordinate of the LUT table
- Bits 15:19 - 5 bits offset to select the line inside the page
- Bits 20:26 - 7 bits, that select vertical page in the tiler container and also Y co-ordinate of the LUT table

- Bits 27:31 - In 16 bit mode, their value is binary (01101), that is address in range 6800 0000h-6FFF FFFFh

Virtual address decoding in 32-bit mode access mode is:

- Bit 0,1 - are always 0, as accesses are at-least 32-bit aligned
- Bits 2:6 - 5 bits offset into the horizontal line of the page
- Bits 7:14 - 8 bits, that select horizontal page in the tiler container as well as the X co-ordinate of the LUT table
- Bits 15:19 - 5 bits offset to select the line inside the page
- Bits 20:26 - 7 bits, that select vertical page in the tiler container and also Y co-ordinate of the LUT table
- Bits 27:31 - In 32 bit mode, their value is binary (01110), that is address in range 7000 0000h-77FF FFFFh

4.2.1.5 ROBIN

The re-ordering buffer and initiator node (ROBIN) is a block aimed at providing some working buffering for converting data and responses to-and-fro between raster and tiled organizations and a master port to connect to the SDRAM controller.

The ROBIN block does:

- Request Forwarding
- Buffering of Write access data and buffering of Read access Response data
- Keeps write data ordering
- Intra-word tiling and orientation transforms
- Tag handling

NOTE: Both ROBIN sub-modules are not software configurable.

4.2.1.6 Section Mapping

In the device, DMM supports two unique SDRAM controllers, with a software configurable option to interleave data between both banks, at granularity of 128 Bytes, 256 Bytes and 512 Bytes. When accessing tiled data in 8-bit, 16-bit and 32-bit modes, which in the interleaved section of the memory, the interleaving will happen at the tile boundary of 1KB, over-riding the interleaving definition of the section.

For optimal system performance, it is recommended to enable interleaving between the 2 EMIF banks and thus have same sized memory on both the EMIF banks. Example : 512MB of DDR3 on EMIF bank 0 and 512MB of DDR3 on EMIF bank 1, for a total system DDR3 memory of 1GB.

NOTE: This interleaving is supported only if DDRs system connected to both the SDRAM controllers have same electrical characteristics.

If for cost reduction, one chooses to build a system with asymmetrical memory on both EMIF banks, then a few limited configurations can be supported by using the LISA section programming feature of DMM.

The address mapping inside the DMM is configurable through up to four sections. Each section is defined based on:

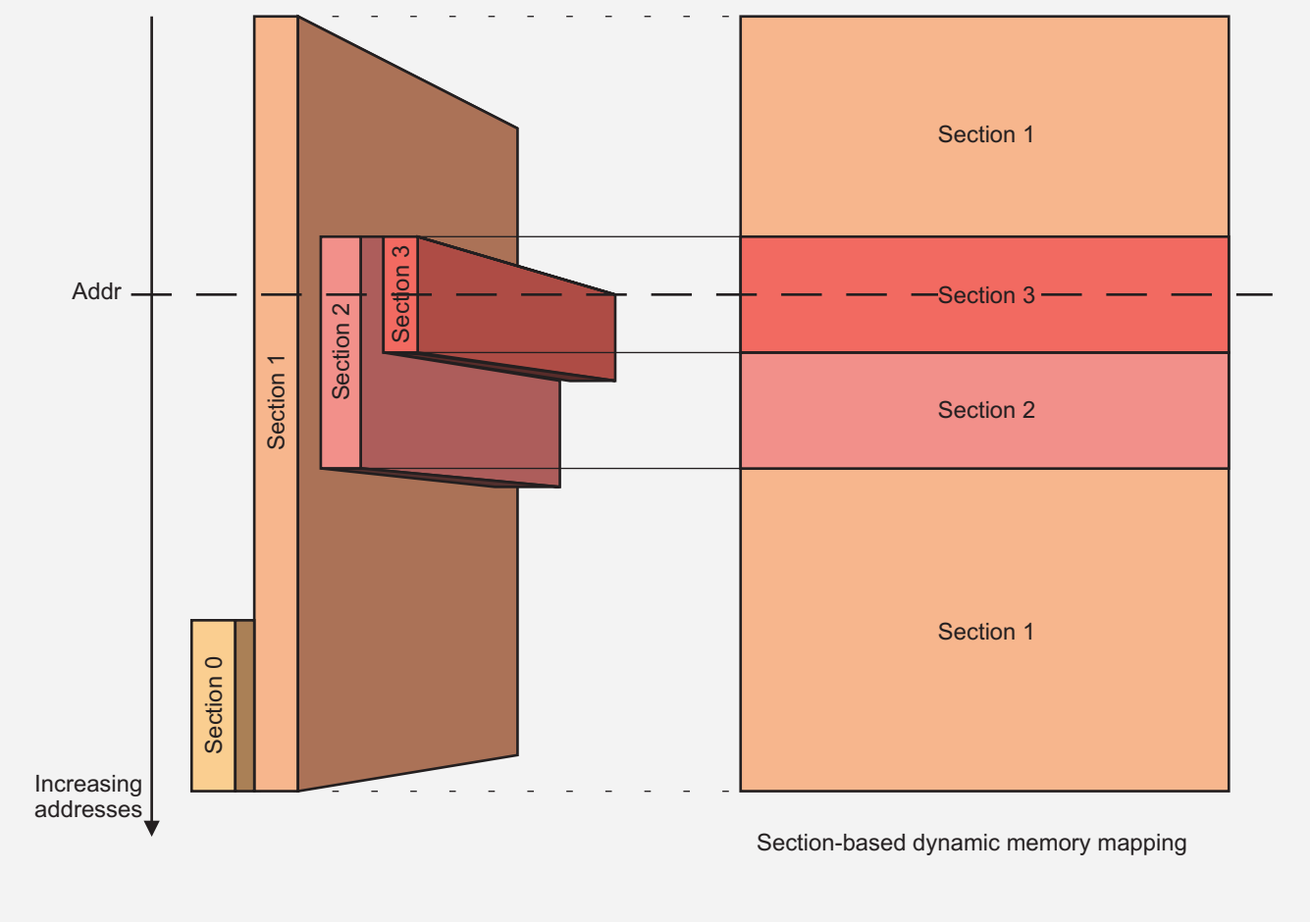
- Its system address: the base address of the decoding range for the section. This is the address of the incoming access to the DMM, also referred to as System Address.
- Its size: the encoding is the number of bits actually used in the upper 8-bits of the incoming system address. Hence the size of the section can be a value 16MB-2GB and power of 2. With interleaving enabled, the minimum size of the section is 32 MB.
- Its physical address: the base address of the memory range access in the external memory controller. It is also referred to as SDRC address.

- The target memory controller: A section may hit either of EMIF banks or both.
- Its interleaving definition : Interleave at 128Bytes, 256 Bytes, 512 Bytes or no interleaving.

A LISA section is:

- a memory segment of size 16MB – 2GB, which is a power of two and aligned to that size in the system map. If the section is configured to interleave data between 2 EMIF, then the minimum section size is 32 MB.
- An area with constant interleaving scheme, with fixed interleaving granularity. Interleaving between 2 EMIF banks can be also disabled, for this section.
- If two sections overlap, then the property of higher section number will be applied, as shown in Figure 4-6. There can be a total of 4 unique section definition in DMM.

Figure 4-6. DMM Section and Memory Mapping



In Figure 4-6, a request at the system address Addr will follow the interleaving scheme of section 3, although hitting sections 1, 2 and 3. Similarly, this DMM configuration also prevents any request to use the interleaving scheme of section 0, since section 0 is fully masked by section 1 – which has a higher priority.

The DMM section specifies the range of incoming (to DMM) System address (granularity of 16 MB) and how it maps to the physical SDRC address of EMIF0 and/or EMIF 1, with similarly aligned granularity.

The interleaving of data between two EMIF banks is transparent to all the initiators of the system and will not require special address conversions. Figure 4-7 and Figure 4-8 give examples of all the interleaving schemes.

Figure 4-7. 128B and 256B Interleaving

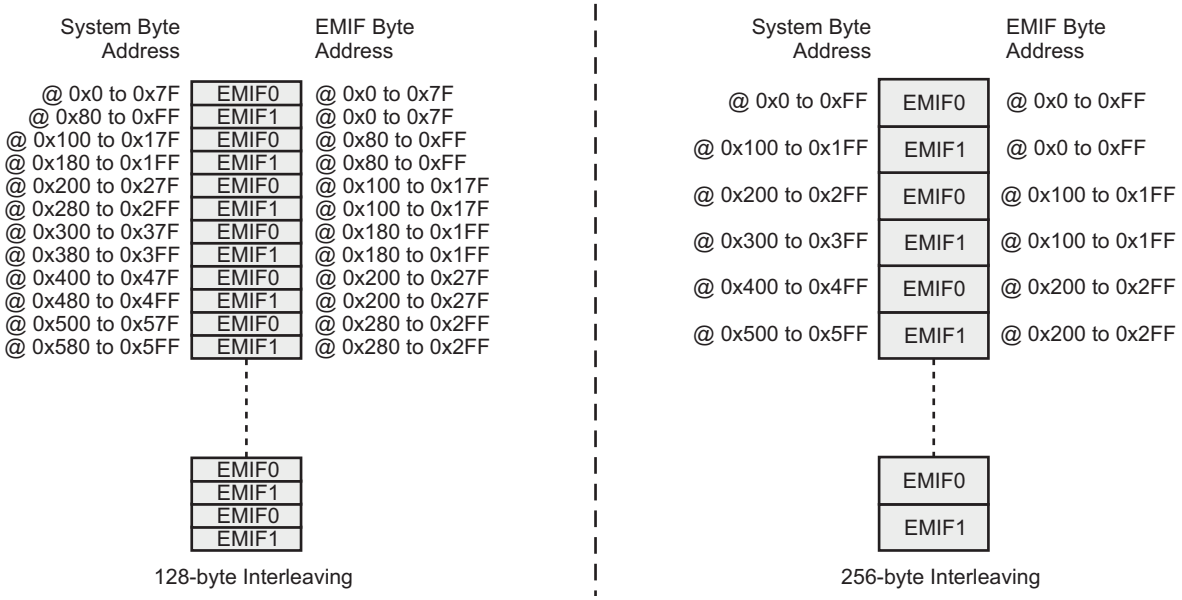
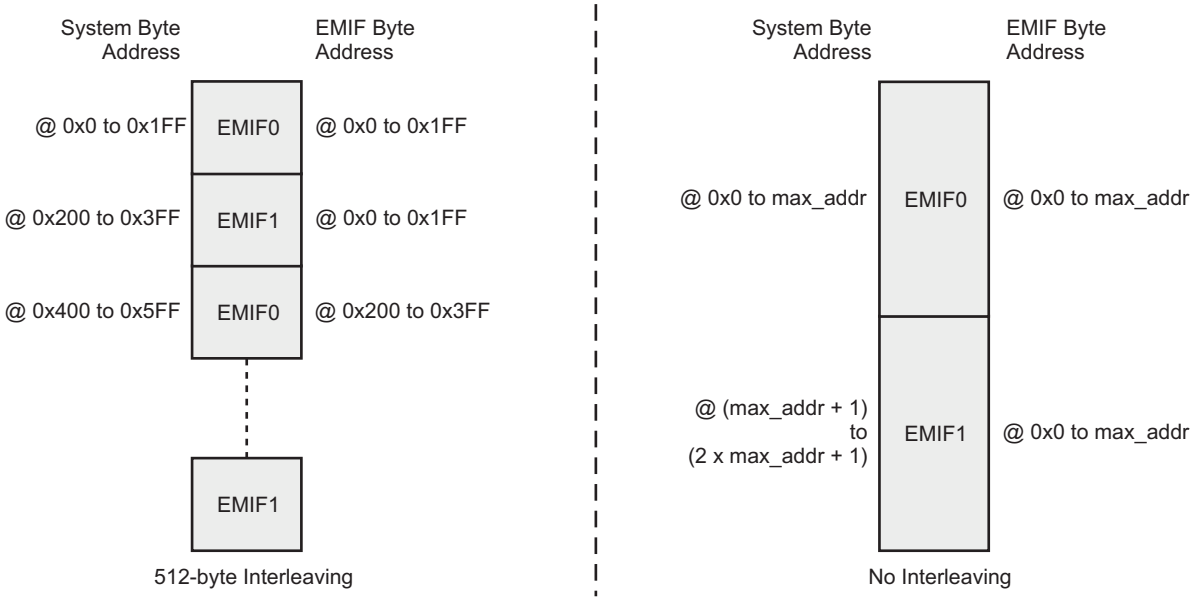


Figure 4-8. 512KB and 1KB Interleaving



4.2.1.7 TILER

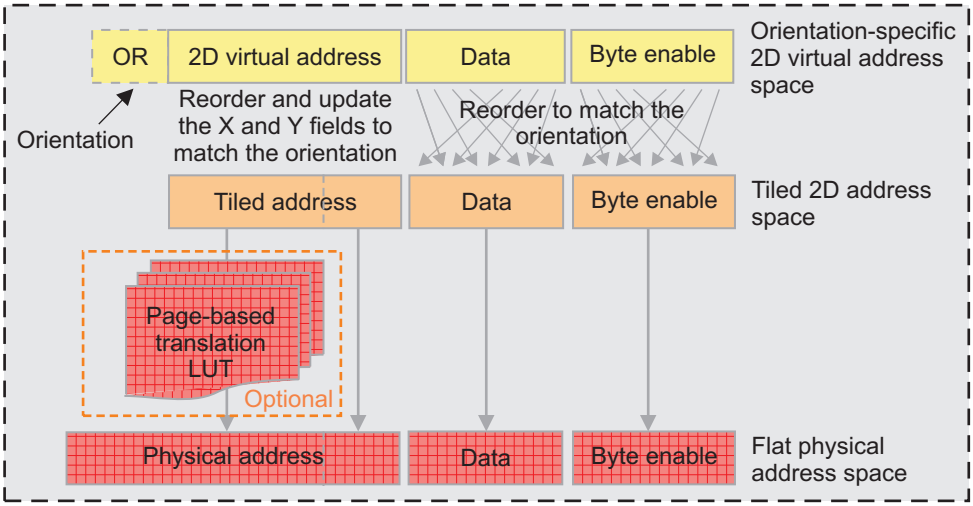
As can be seen in Figure 4-2, the two TILER ports interface to the incoming accesses to DMM. Its primary function is increase efficiency of a 2D Tiled Block access. Mentioned below are the main functions of the TILER module:

1. primary handling efficiently 2 dimensional data mapped in tiles, such as video or graphics macro blocks.
2. optionally managing the memory fragmentation and zero-copy physical frame buffers swapping through a page-grained translation, on 4KB page granularity.
3. making isometric – distance preserving – transforms, such as 90°, 180°, or 270° rotations, with either a horizontal or vertical reflection, with no overhead.

Written differently, the functionality of this TILER module is to map a 2D virtually-addressed incoming request into one or more physically-addressed SDRAM requests by:

1. Decoding the incoming access address to qualify whether the request targets the TILER or the memory directly.
2. If the incoming access is a TILER-specific requests, then transforming to the virtual address, data and byte enable to match the requested 0°, 90°, 180°, or 270° orientation in a tiled 2D addressing space.
3. optionally translating the oriented tiled address by a page-specific vector to manage memory fragmentation and physical object aliasing, as shown in Figure 4-9.
4. Splitting tiled requests at tile boundaries and non-tiled requests at DMM atomic section boundaries
5. Requesting the page-based address translation
6. Requesting buffer allocation in the appropriate ROBIN.
7. In case of a write request, allocating and updating an internal write context to subsequently direct the incoming write data into the relevant re-ordering buffer.

Figure 4-9. Overview of Request Conversion

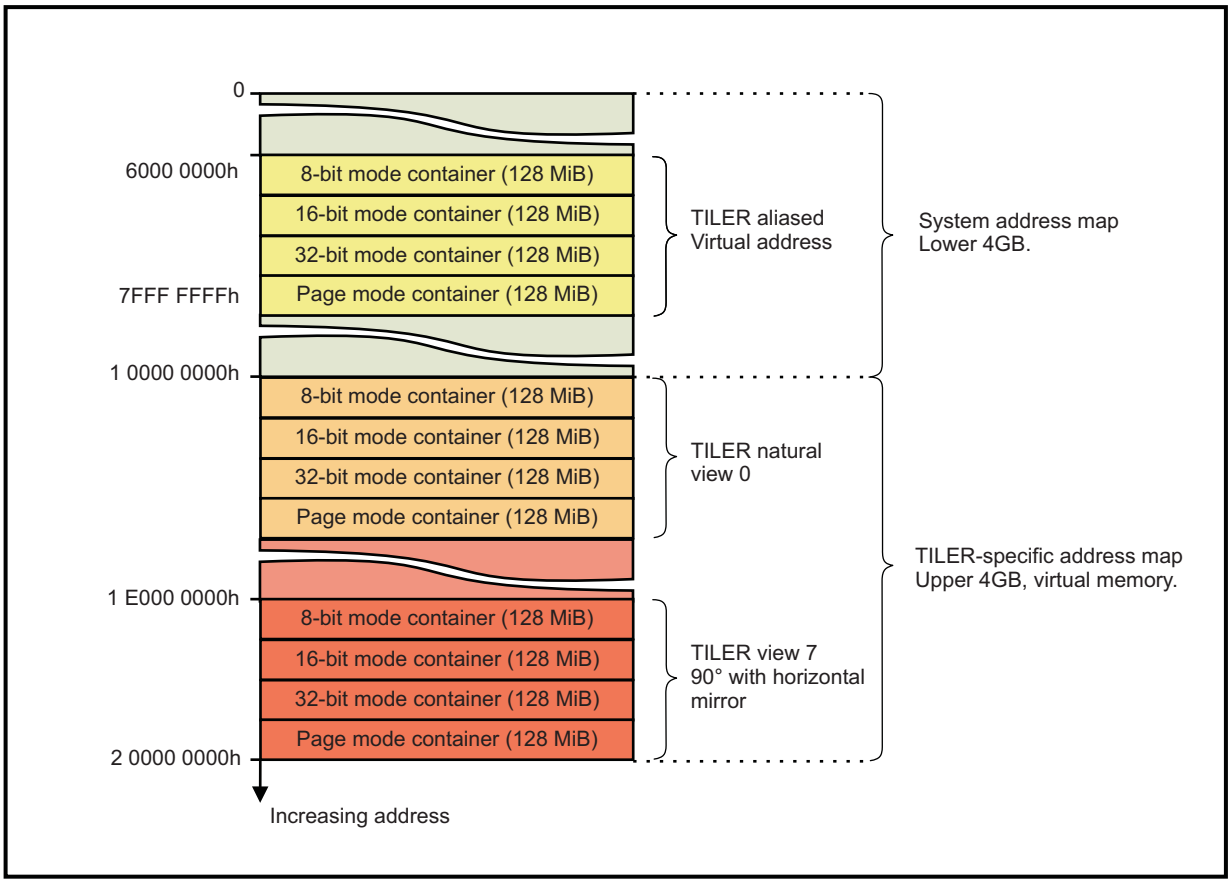


4.2.1.7.1 Device Memory Map - For Tiled Data

In the Device, the initiators read and write tiled data by generating accesses to the tiled space by the one of the two methods:

1. For all initiators except, HD_VPSS, the tiled space is accessed in the 512 MB address range of (6000 0000h to 7FFF FFFFh). These initiators use the access the any of 8 views by using the respective DMM_TILER_OR0 or DMM_TILER_OR1 registers.
2. For HD_VPSS, the TILER module supports its own 4-GB virtual addressing space, from address (1 0000 0000h to 1 FFFF FFFFh), and allows any of the eight 512-MB oriented sub-spaces – or views – to be remapped in the system virtual addressing space.

Figure 4-10. Memory Map



NOTE: Despite the larger virtual address windows, 512 MB in the system memory map and 4GB in the TILER memory map, the maximum size of physical tiled data is only 128MB. This 128MB is co-located in a 128MB aligned space, if using PAT direct access translation. In case of using the optional PAT Indirect access translation, then this 128MB is mapped to any location in the SDRAM, using the LUT tables, at the granularity of 4KB page.

4.2.1.8 Clock

The DMM is a synchronous design and operates from the same clock as the internal L3 interconnect. All timings use this clock as a reference.

4.2.1.9 Interrupts

DMM module generates interrupt to Cortex-A8, for status and error conditions related to PAT refills engines. There 8 different conditions which can be individually enabled or disabled. With 4 Refill engines, there are total 32 unique sources of interrupts.

Refer to Section 4.3.3 for detailed description on using the PAT refill engines.

Per Refill engine, status and error conditions are mentioned below:

1. FILL_DSCn: Refill of any descriptor, for engine n, complete.
2. FILL_LSTn: Refill of the last descriptor, for engine n, complete.
3. ERR_INV_DSCn: Invalid descriptor pointer, for engine n.
4. ERR_INV_DATAn: Invalid data table pointer, for engine n.
5. ERR_UPD_AREAn: Error caused by area register update while refilling, for engine n.
6. ERR_UPD_CTRLn: Error caused by control register update while refilling, for engine n.
7. ERR_UPD_DATAn: Error caused by data register update while refilling, for engine n.
8. ERR_LUT_MISSn: Access to a yet-to-be-filled entry, that is part of the area being refilled, for engine n.

The six error conditions mentioned above are also reported in the respective DMM_PAT_STATUSn register.

4.2.1.10 Address Translations Within DMM, For Tiled Accesses

This section describes address translations that happen at various interfaces within DMM, for tiled accesses.

From a user perspective, the PAT can be seen as an additional step in the process applied to the tiled addresses by the TILER:

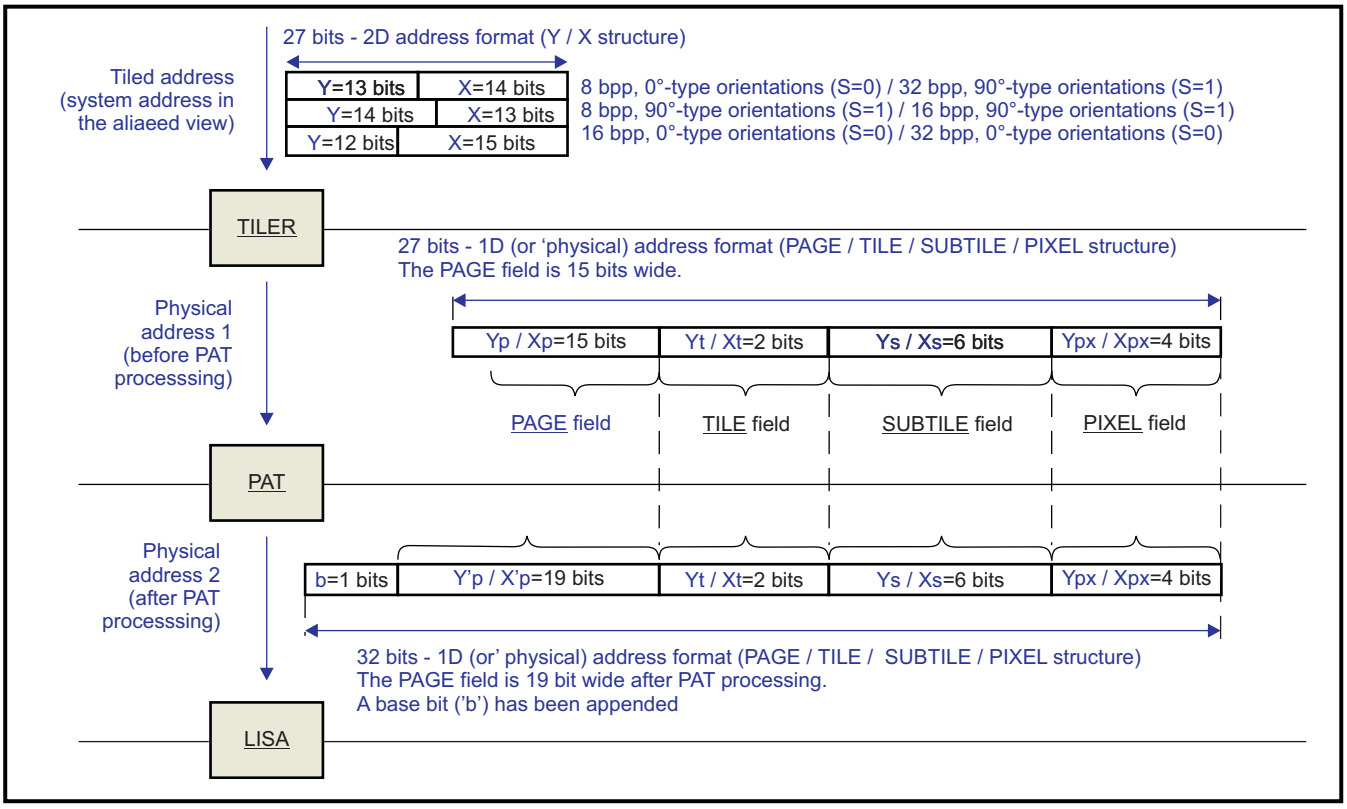
- the TILER first transforms a tiled address into a 27 bit physical address which fits within a 128MB address range.
- the PAT then processes the high order bits of the physical address output by the TILER, and outputs a 32-bit physical address which covers a 4GB address range. Section 4.2.1.4 describes the exact details for this address translation.

The purpose of the PAT is to map the tiled data anywhere in the 4GB physical address range, with a PAGE granularity (The TILER page is the granularity of physical memory allocation in TILER container. Each page is 4KB). This can be summed-up in Figure 4-11 where the PAT process appears in red.

Only the high order bits of the physical address are modified by the PAT: the 12 low-order bits remain unchanged. This means that the data ordering within each 4kB PAGE remains as calculated by the TILER.

A PAT view is defining the kind of physical address translation to perform for each of the page, 8-bit, 16-bit and 32-bit mode accesses. Each mode in each PAT view can be programmed in 2 different modes (direct translation and indirect translation) which will be described in PAT section. Note that indirect mode is the most commonly used. Direct mode is used only for debug or in case of a DMM without PAT module.

Figure 4-11. DMM Address Translations



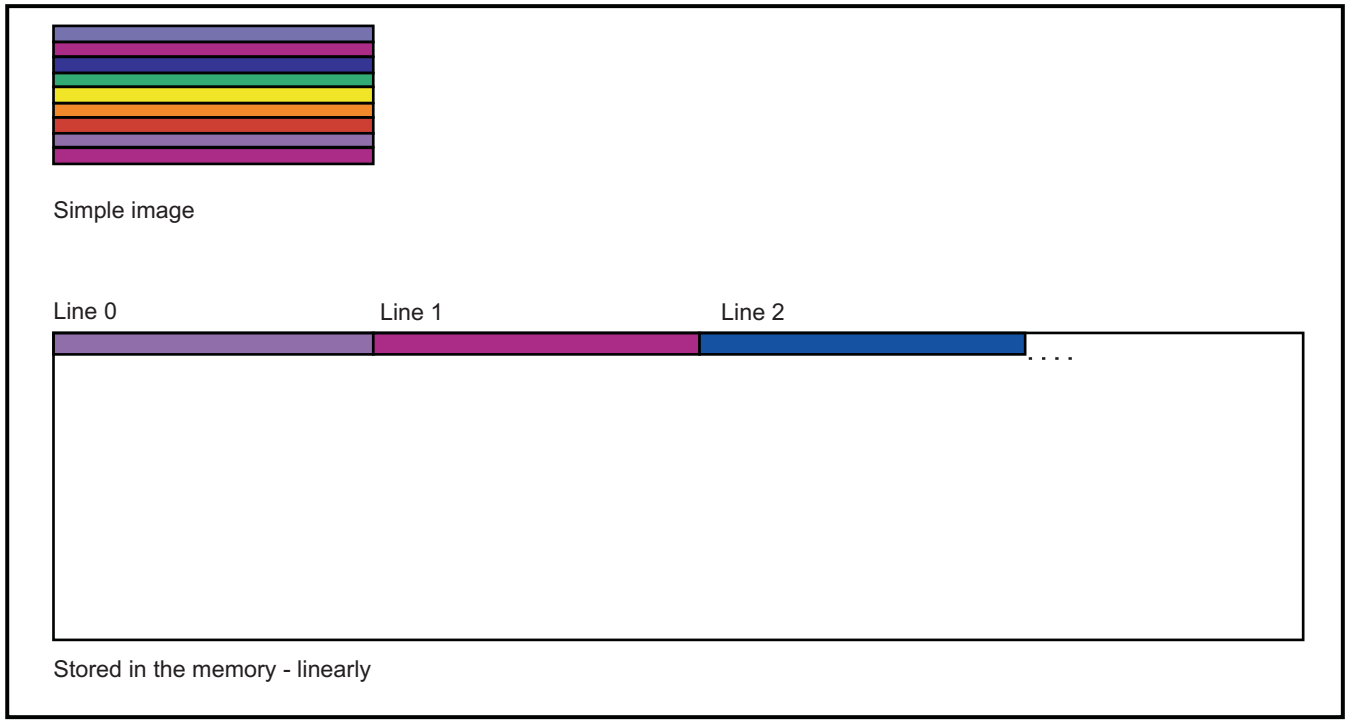
4.2.2 TILER Functional Description

4.2.2.1 Introduction to TILER

4.2.2.1.1 Need for a TILER

Image processing algorithms like H.264 encode and decode, which works on the principle of spatial locality, access the image data in two-dimensional format, for example, 16 pixels wide by 16 pixels high. If the image is stored in the memory in linear format then it would appear as in Figure 4-12. Thus, to access the macroblock, multiple memory accesses would need to be made. In the device, each memory access can be up to 128 bytes large. However, because macroblock is not stored contiguously in the memory, only partial pixels is useful in each memory access and the rest is discarded and a new access is generated to fetch rest of the data.

Figure 4-12. Image Stored in the Memory Linearly

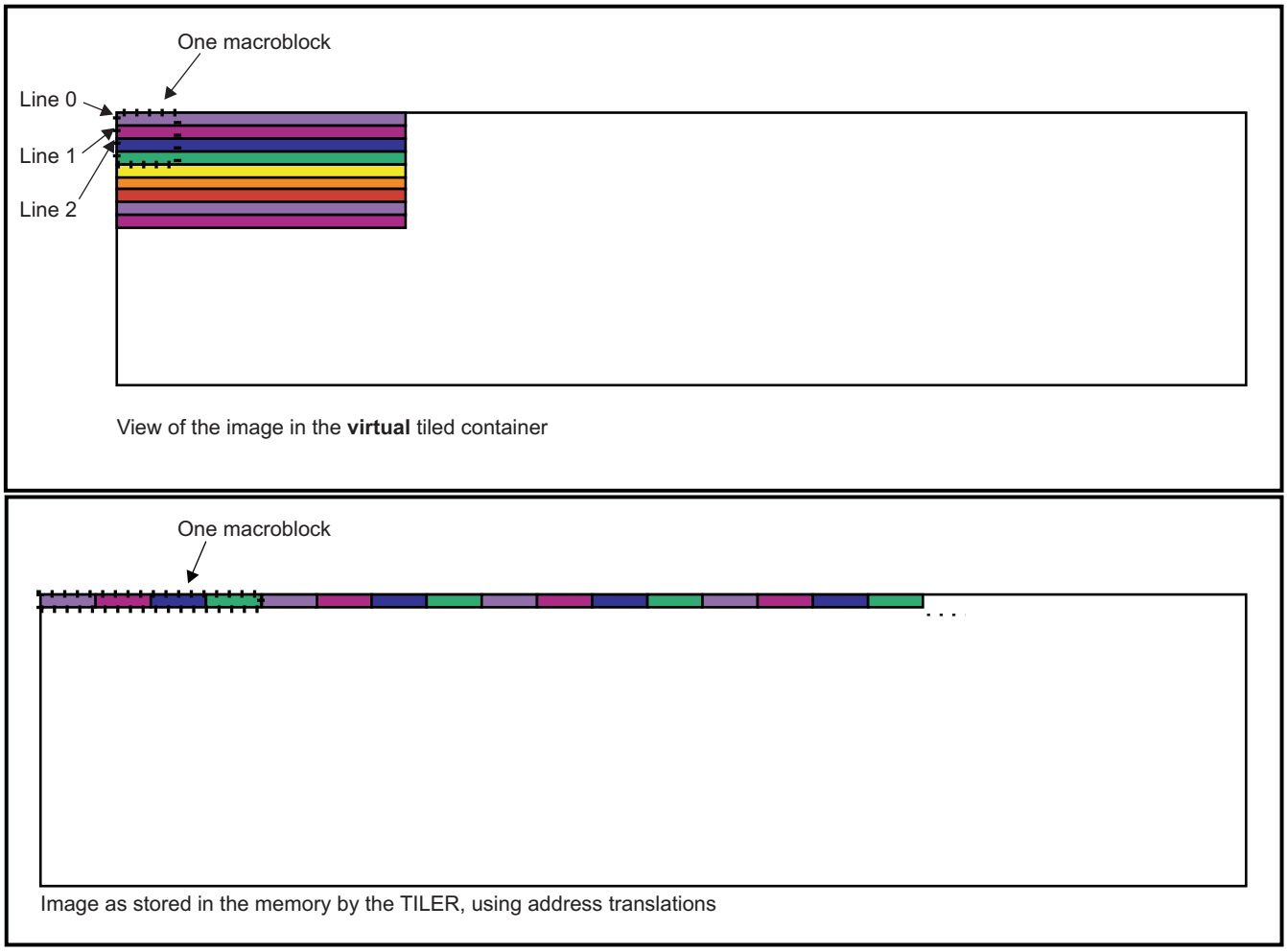


Instead, if the image is stored in bi-dimensional format, much fewer accesses are needed to read the entire macroblock of data, which directly transforms into several advantages:

1. Faster processing of algorithms.
2. Fewer SDRAM page opens.
3. Lesser wastage of memory bandwidth in the system.

This concept is the primary motive for storing the image data in the tiled format.

Figure 4-13. Image Stored in the Memory by TILER



NOTE : This is only a pictorial representation and actual arrangement of the data in the 128-bit subtile may be different.

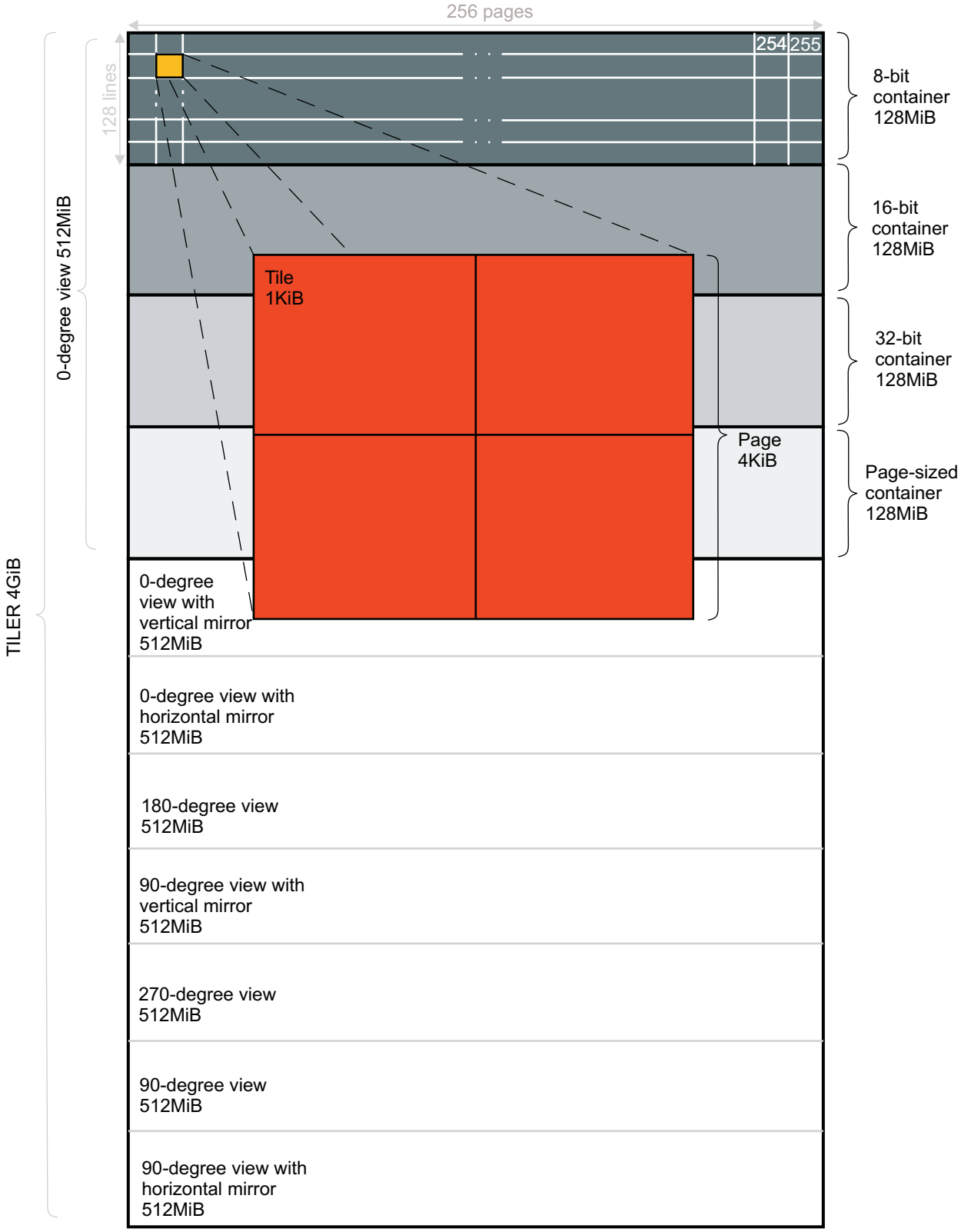
4.2.2.1.2 Consuming Tiled Data

In the device, the DMA engine of HD_VPSS, also known as VPDMA, has internal line buffers of its capture and display paths (called clients). For every tiled access, it is ensured that rastering tiled data has zero overhead, when processing an entire frame, or 2^{2n} number of lines.

4.2.2.2 TILER Rationale - A Top-down Approach

This section is a synthesis of all TILER concepts, through a top-down approach starting from the main object container, giving one rule per TILER structure level. Figure 4-14 shows the TILER address space structure for tiled modes.

Figure 4-14. Address Space Structure for Tiled Modes



4.2.2.2.1 The TILER is a 4-GB Virtual Address Space Composed of Eight Views

A TILER is addressed using a virtual 4 GB address space, containing 8 unique orientation views of 512 MB each. There is one view for each of the eight possible way of scanning a frame-buffer.

1. From left to right then from top to bottom (0 natural orientation)
2. From right to left then from top to bottom
3. From left to right then from bottom to top
4. From right to left then from bottom to top
5. From top to bottom then from left to right
6. From top to bottom then from right to left
7. From bottom to top then from left to right
8. From bottom to top then from right to left

Section 4.2.2.7 summarizes how the view can be visualized on a 128MB TILER container.

4.2.2.2.2 A View is a 512-MB Virtual Address Space Composed of Four Containers

There is one container per element size to allow correct access patterns in any of the eight possible orientations. The container is the entity where all objects of a given element type are allocated. The element is the entity of maximum size - 8 bits, 16 bits, 32 bits or page-sized - which is invariant in any orientation.

4.2.2.2.3 A Container is a 128-MB Virtual Address Space

A 128 MB container can be visualized as an array of 128 x 256 pages of 4 KB each. The page defines the granularity of physical memory allocation through a physical address translation unit - MMU.

4.2.2.2.4 A Page is a 4-KB Virtual Address Space

A page is composed of two lines. Each line consists of two tiles. Figure 4-15, Figure 4-16, and Figure 4-17 illustrate the Page geometry for 8-,16-, and 32-bit views, respectively.

Figure 4-15. 4KB Page, 8-bit Mode

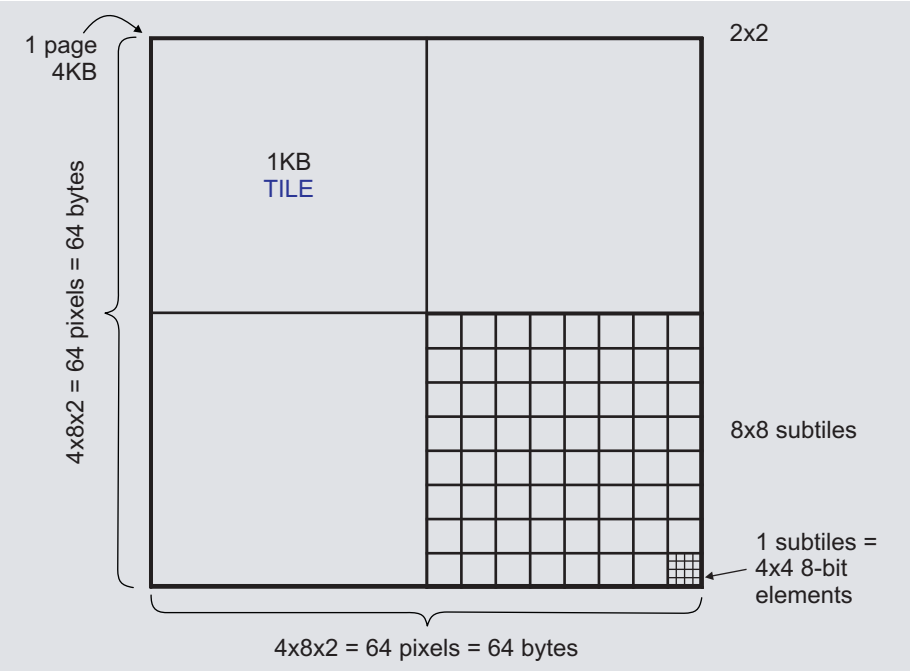


Figure 4-16. 4KB Page, 16-bit Mode

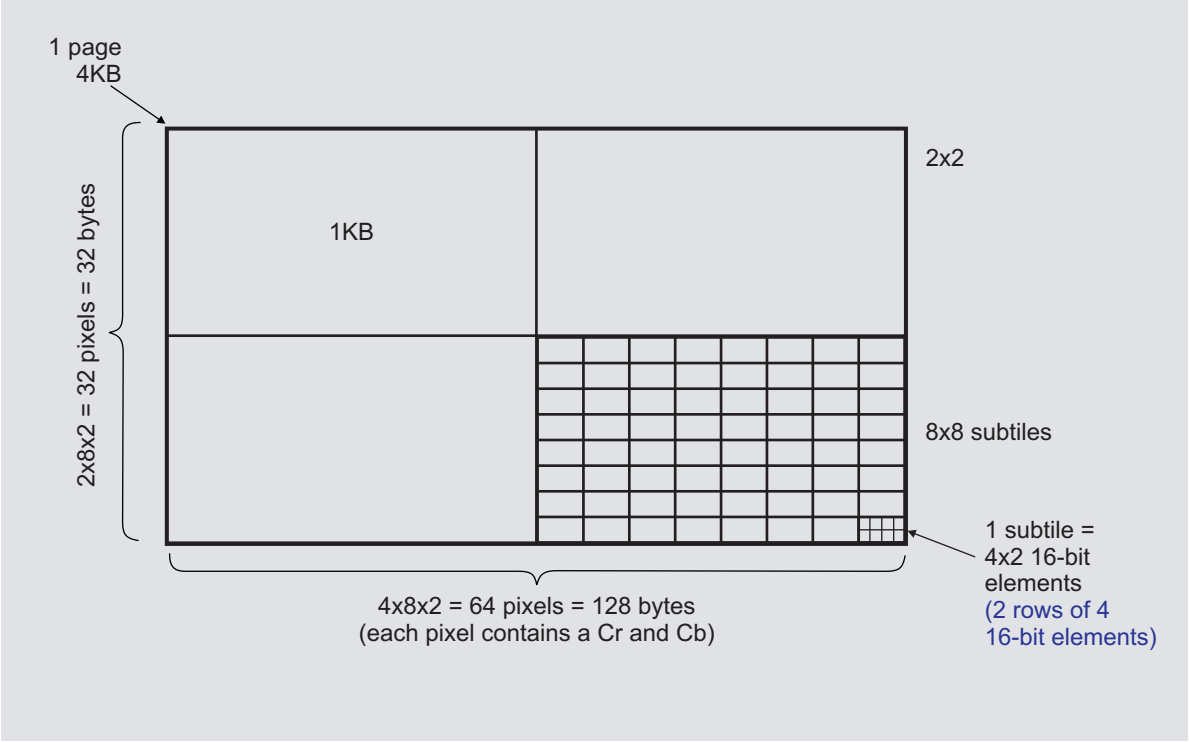
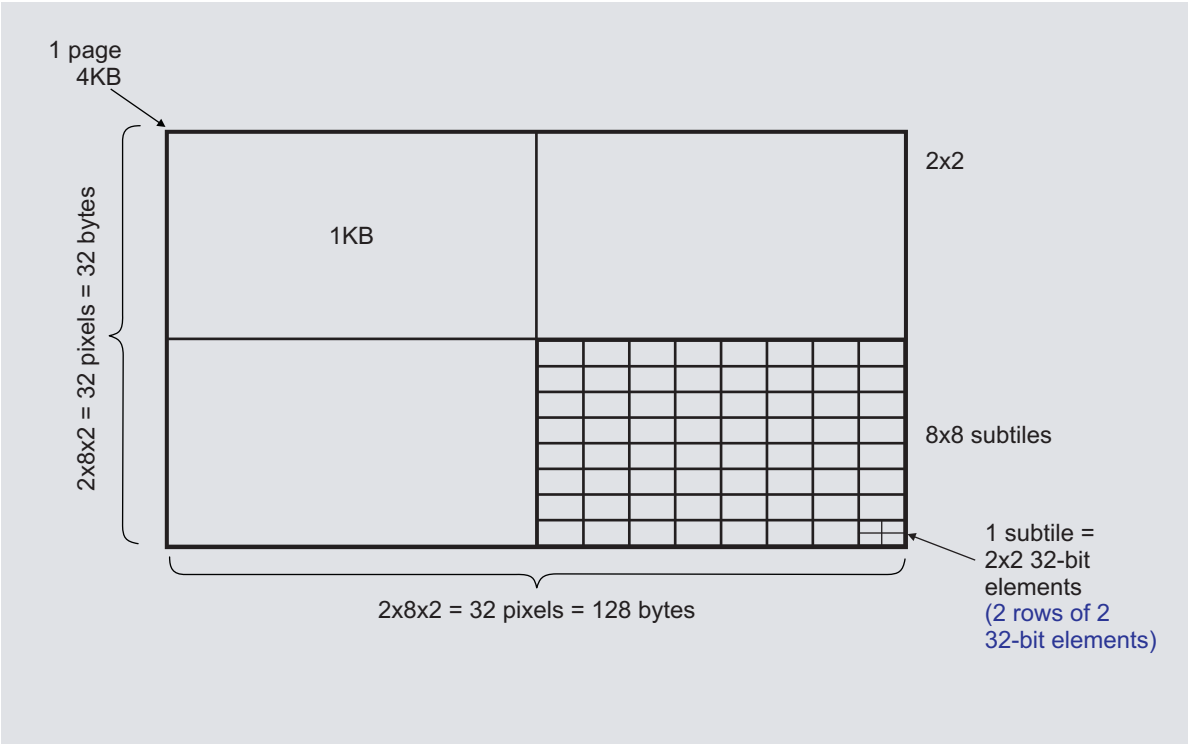


Figure 4-17. 4KB Page, 32-bit Mode

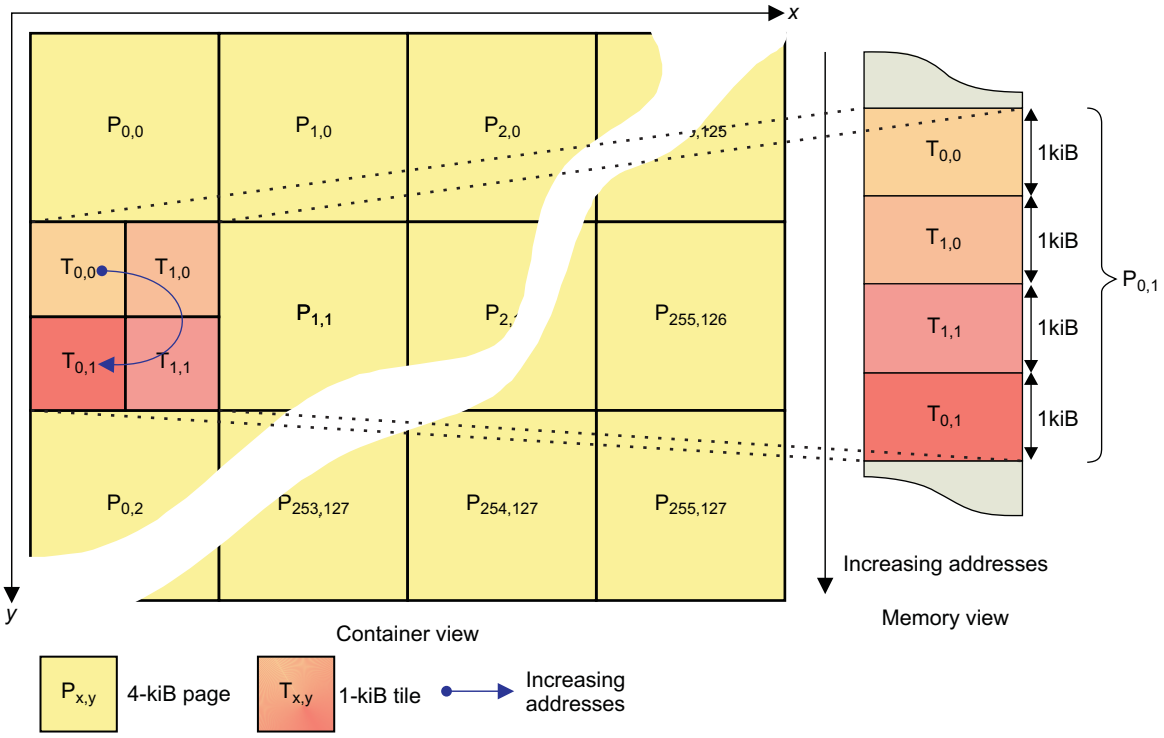


4.2.2.2.5 A Tile is a 1-KB Address Space

The tile is designed to offer bi-dimensional data locality in a single SDRAM page. In this respect, it has been sized to 1 KB (the size of the smallest SDRAM page).

Figure 4-18 shows how 4 tiles are arranged in a page. This arrangement ensures that any two adjacent tiles in a page are not stored on the same EMIF(when interleaving is enabled). Thus a large tiled access will be efficient as it will be evenly spread across both SDRAM controllers.

Figure 4-18. Four 1KB Tiles in One 4KB Page



4.2.2.2.6 A Sub-tile is a 128-bit Address Space

A 1KB tile is further decomposed into 64 sub-tiles, each sized 128 bits. The sub-tile structure is such that it balances accesses in two-dimensional modes, thus improving SDRAM access efficiency.

Furthermore, using Sub-tile pairing, the interlaced accesses to the tiled data can be improved. A Sub-tile is defined as follows:

1. An 8-bit sub-tile (Figure 4-19) is defined as an array of four horizontal lines of four 8-bit data. Thus, in 8-bit tiled mode, the TILER container is a 16384 × 8192 2D array of 8-bit elements.
2. A 16-bit sub-tile (Figure 4-20) is defined as an array of two horizontal lines of four 16-bit data. Thus, in 16-bit tiled mode, the TILER container is a 16384 × 4096 2D array of 16-bit elements.
3. A 32-bit tiled (Figure 4-21) mode as an array of two horizontal lines of two 32-bit data. Thus, in 32-bit tiled mode, the TILER container is an 8192 × 4096 2D array of 32-bit elements.

Figure 4-19. 8-bit Sub-tile

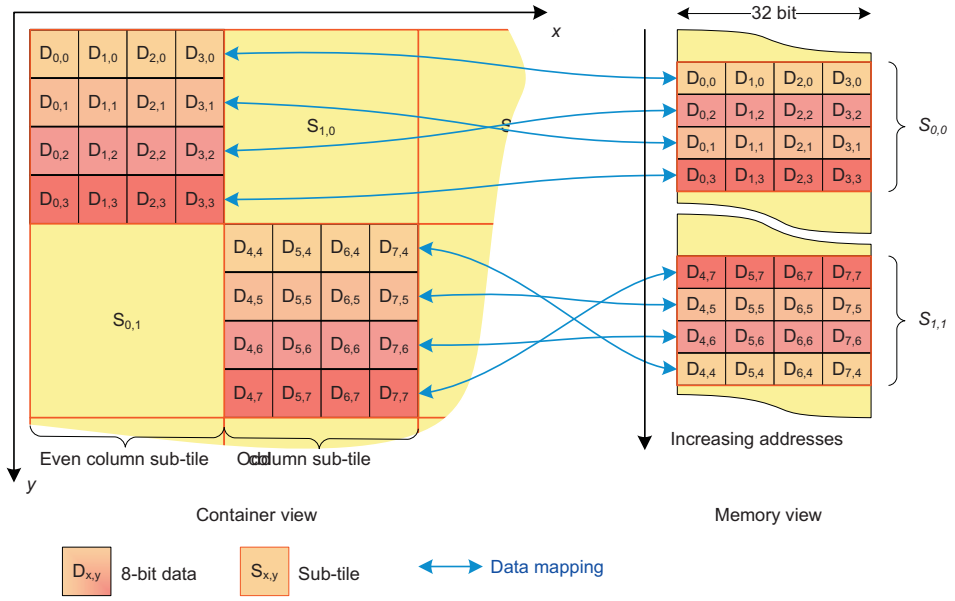


Figure 4-20. 16-bit Sub-tile

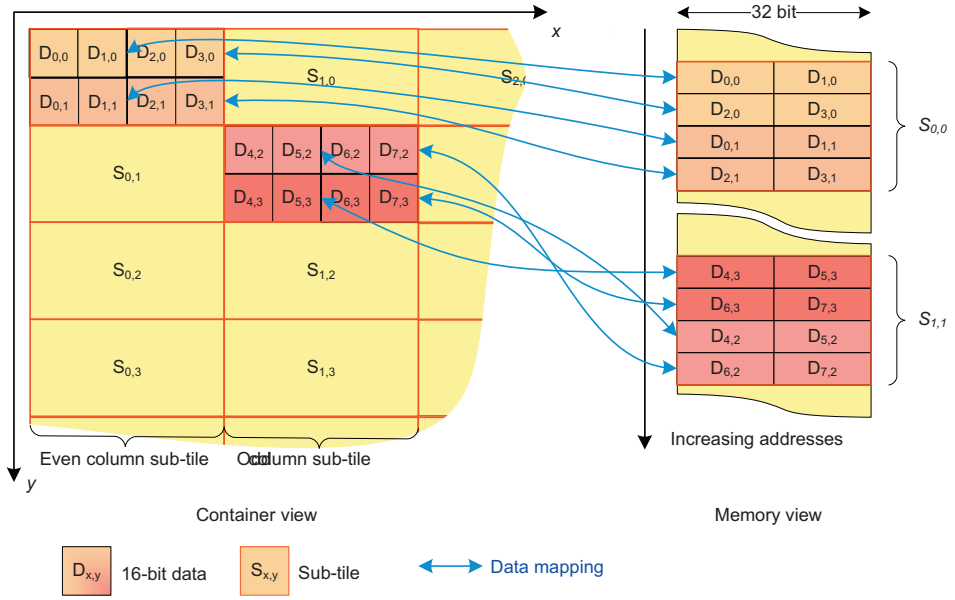
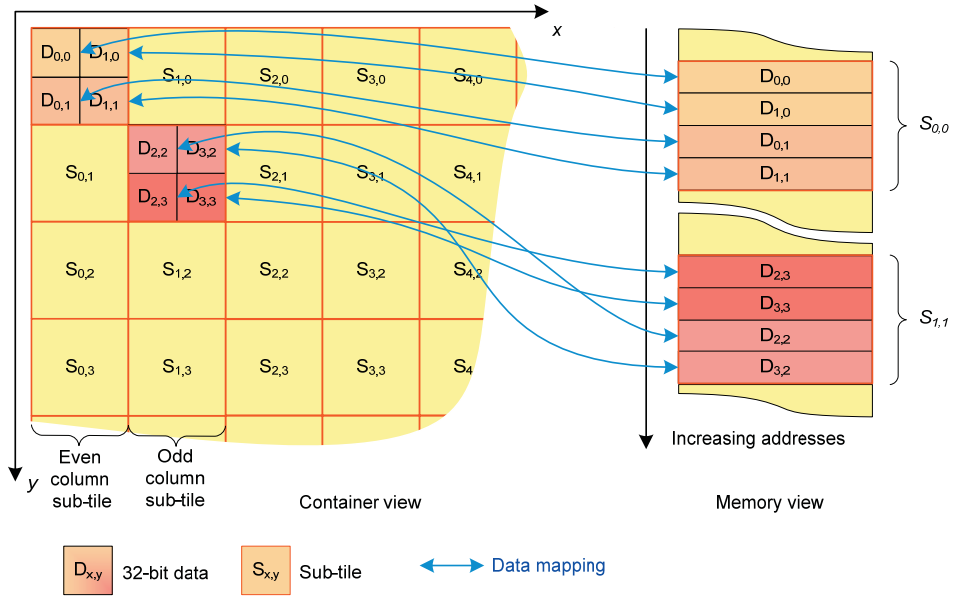


Figure 4-21. 32-bit Sub-tile



4.2.2.3 TILER Container - Summary

TILER is a 128-MB virtual container arranged in 2 dimensions as 256 × 128 pages, each of 4KB.

Each 4-KB page is has 4 tiles.

Each 1-KB tile has 64 subtiles, each of size 128 bits.

Based on the mode of access, 8/16/32 bit, the data stored can be aligned differently, as explained in the previous section.

4.2.2.3.1 Where Will the Tiled Data Be Stored in Physical Memory?

Using PAT Direct Access Translation:

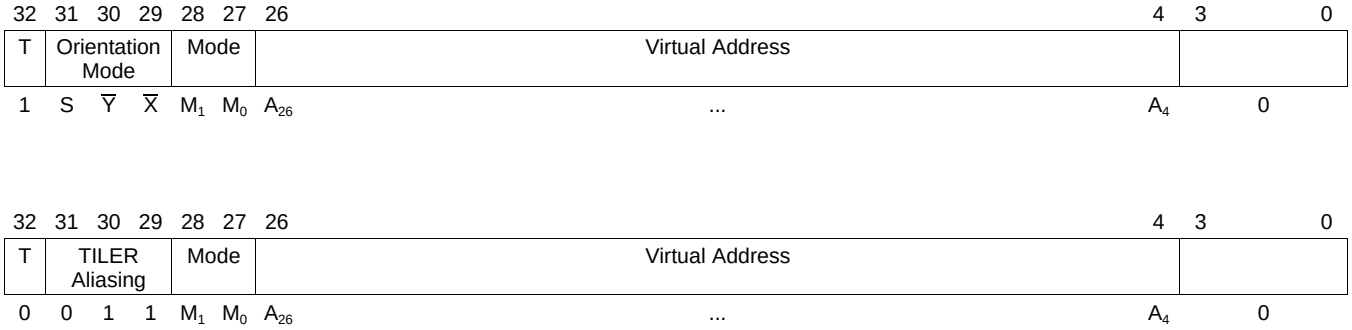
By bypassing the LUT, the entire 128MB space has to be contiguous in the physical memory. Then each of the 8-bit 128 MB virtual container, 16-bit 128 MB virtual container, 32-bit 128 MB virtual container and paged container, can be mapped to either the same physical 128 MB space or at different locations. Refer to the PAT_VIEW_MAP register, which defines the base address of this 128 MB space. As can be seen, this base address is 256 MB aligned.

Using PAT In-Direct Access Translation:

A more common use case would be to use the two LUTs in the DMM to map the virtual data to physical address, at 4KB granularity. Then the physical 128MB need not be contiguous. With two LUTs in the system, only 256 MB of data is possible. Hence, the 8, 16, 32 bit and paged containers may need to be aliased to same pages. For example - use one LUT to map 8-bit, 16-bit, and 32-bit containers; hence, they would all correspond to the same physical memory pages. The other LUT would be used to map the 128 MB contained for paged accesses, as shown in Figure 4-24.

4.2.2.3.2 Addressing Formats

Figure 4-22. Address Format



1. The 33rd bit, noted T, is aimed at distinguishing the standard 4-GB system address map from the 4-GB TILER-specific address map In the Device, only HD_VPSS can generate accesses in the latter space.
2. The orientation bits, noted S, Y, and X, define the request orientation as specified in Section 4.2.2.7.
3. The mode bits, noted M1 and M0, encoded as in Section 4.2.2.8.1.
4. The remaining 27 bits, noted A0 to A26, define the mode and orientation specific virtual address.

4.2.2.4 TILER Modes

The TILER is supporting three major access modes - bypass, page and tiled - each having a specific output request generation.

4.2.2.4.1 Bypass Mode

This mode is transparent at the TILER perspective. Bypass mode is for accesses generated by initiators with System addresses outside the virtual tiled address range, since TILER will bypass PAT. Still, at the DMM perspective:

- 2D block bursts are broken down on a line-basis in a set of incremental bursts
- Incremental bursts - including those issued by a 2D block burst breakdown - are split at the DMM atomic unit of the section hit by the burst at:
  - at the interleaving granularity of the section - 128 byte, 256 byte or 512 byte - in interleaved sections
  - at 1-KB boundary in non-interleaved sections

4.2.2.4.2 Paged Mode

The purpose of this mode is to use the DMM PAT address translation mechanism for non-tiled accesses. In this respect it is similar to the bypass mode:

- 2D block bursts are broken down on a line-basis in a set of incremental bursts
- Incremental bursts - including those issued by a 2D block burst breakdown - are split at:
  - the interleaving granularity of the section - 128 byte, 256 byte or 512 byte - in interleaved sections
  - 1-KB boundary in non-interleaved sections

4.2.2.4.3 Tiled Mode

When the being accessed in tiled mode, it can be either of the following:

- A well formed 2D block requests - conforming to the orientation, mode and stride
- A 1D incremental requests and ill-formed 2D block requests

Table 4-1. Stride for Well-formed Tiled Mode 2D Block Requests

Orientation			Mode		Stride (bytes)	Description
S	Y	X	M1	M0		
0	x	x	0	0	16384	Plain access to an 8-bit progressive frame in 0° or 180°
					32768	Field access to an 8-bit interlaced frame in 0° or 180°
			0	1	32768	Plain access to a 16-bit progressive frame in 0° or 180°
					65536	Field access to a 16-bit interlaced frame in 0° or 180°
			1	0	32768	Plain access to a 32-bit progressive frame in 0° or 180°
					65536	Field access to a 32-bit interlaced frame in 0° or 180°
1	x	x	0	0	8192	Plain access to an 8-bit progressive frame in 90° or 270°
					16384	Field access to an 8-bit interlaced frame in 90° or 270°
			0	1	8192	Plain access to a 16-bit progressive frame in 90° or 270°
					16384	Field access to a 16-bit interlaced frame in 90° or 270°
			1	0	16384	Plain access to a 32-bit progressive frame in 90° or 270°
					32768	Field access to a 32-bit interlaced frame in 90° or 270°

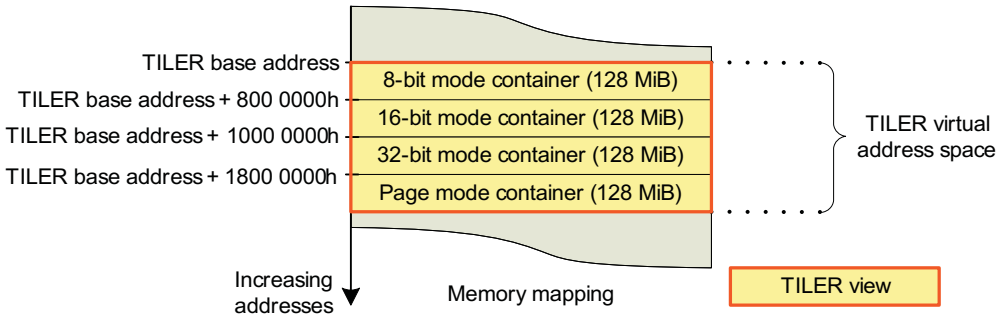
4.2.2.5 Object Container Definition

The object container is the unique addressable entry point of the TILER. It is a 128-MB virtual address space, where all objects of a same kind - and orientation - are allocated. Four main types of containers are present in the TILER, each one being referred by a mode:

- A 8-bit element mode, for efficiently accessing bi-dimensional arrays of 8-bit data, example 8-bits per pixel Luma buffers of image.
- A 16-bit element mode, for efficiently accessing bi-dimensional arrays of 16-bit data, example 16-bits per pixel interleaved-Chroma(CbCr) buffers of image.
- A 32-bit element mode, for efficiently accessing bi-dimensional arrays of 32-bit data, example 32-bits per pixel, ARGB graphics buffers.
- A page mode, for efficient 1D accesses

Figure 4-23 shows the TILER object containers and views.

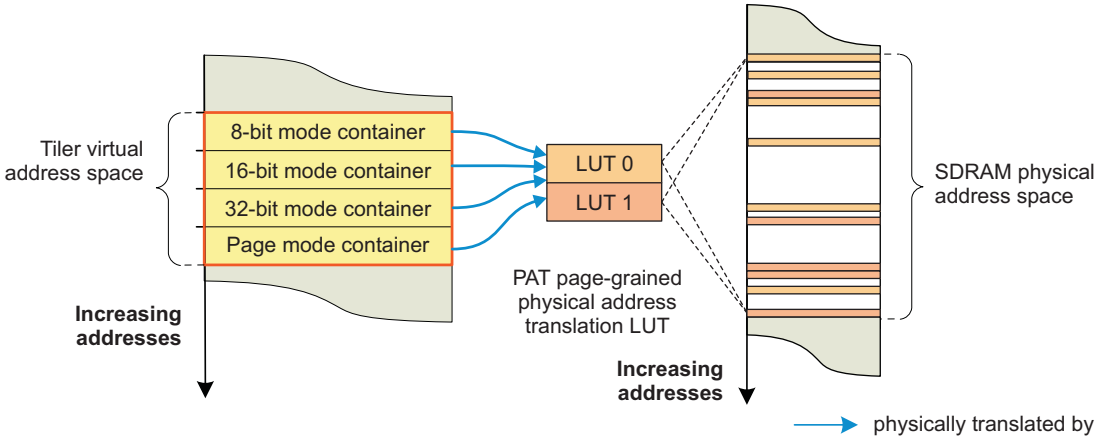
Figure 4-23. TILER Object Containers and Views



In the device, there are two LUTs in the PAT. Each LUT can map up to 128MB of objects at a 4KB page granularity. The four modes share the two LUTs .

One of the recommended usages, as shown in Figure 4-24, is to use one LUT for 8,16, and 32-bit modes and the other LUT for paged mode accesses. With this scheme, up to 128MB of objects can be available simultaneously in 8,16 and 32-bit modes and another 128MB of objects for paged mode accesses.

Figure 4-24. Using LUT to Translate Tiled Virtual Address to Physical SDRC Address



NOTE: You need to ensure that an object allocated in a particular container mode does not physically overlap with any other object either in the same mode or other container modes.

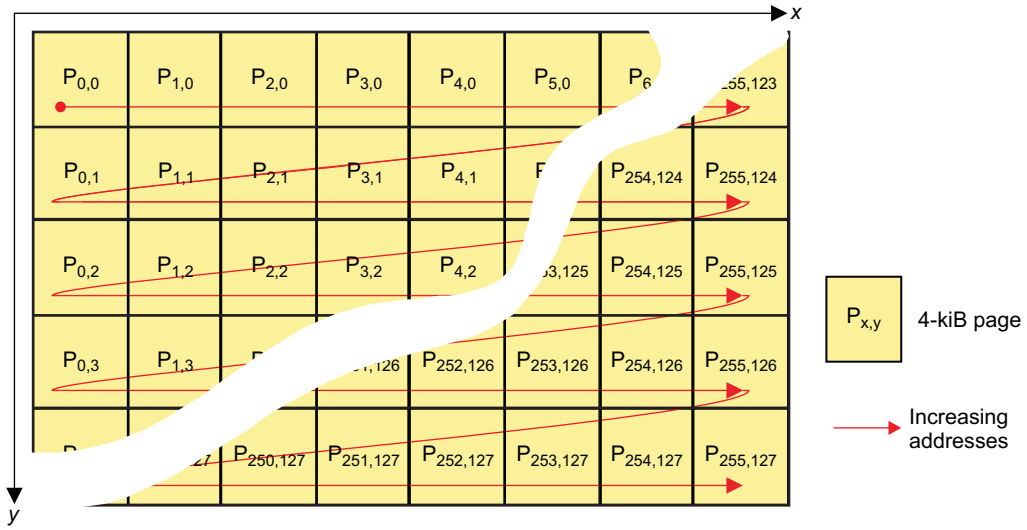
4.2.2.6 Page Definition

A TILER page defines the granularity of object allocation in virtual TILER containers. Given that the sub-page structure is mode specific, the 4KB page is the smallest granularity common to all modes, making it the granularity to consider in the TILER resource manager for object allocation.

4.2.2.6.1 Container Geometry with 4-kiB Pages

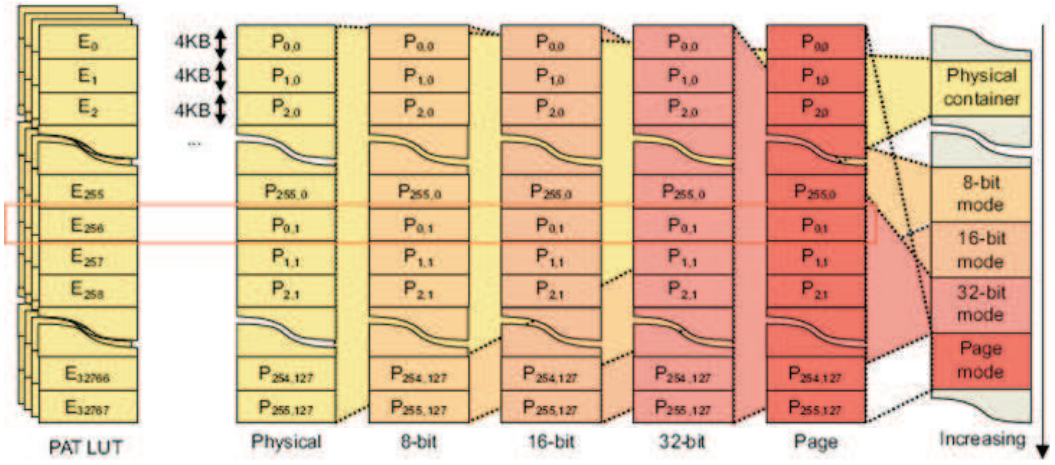
As pages size is 4KB, any 128MB object container is a set of 256 × 128 = 32768 pages, organized in an array of 256 columns and 128 rows, as shown in Figure 4-25.

Figure 4-25. Object Container Geometry with 4KB Pages



This array of pages is mapped to the system address space as shown in Figure 4-26.

Figure 4-26. TILER Page Mapping When Using 4KB Pages



In any 128-MB object container, the 4KB page Px,y at column x (0 ≤ x < 256) and row y (0 ≤ y < 128), is found at an offset of 4096. (x + 256.y) bytes from the base address of the related object container.

Similarly, the page Px,y at column x (0 ≤ x < 256) and row y (0 ≤ y < 128), is translated by the LUT entry Ex + 256.y found at the index x + 256.y.

4.2.2.6.2 Container Geometry and Page Mapping Summary

The TILER has a page size of 4096 bytes. The page Px,y:

- has maxx = 256 and maxy = 128
- is found at an offset of: 4096.(x + maxx.y) bytes from the base address of the related object container
- is translated by the entry found at the index (x + maxx.y) of the LUT table

4.2.2.7 Orientation

This section describes the eight on-the-fly orientation-related isometric transforms, corresponding to all available changes of orthonormal basis in the TILER container bi-dimensional space.

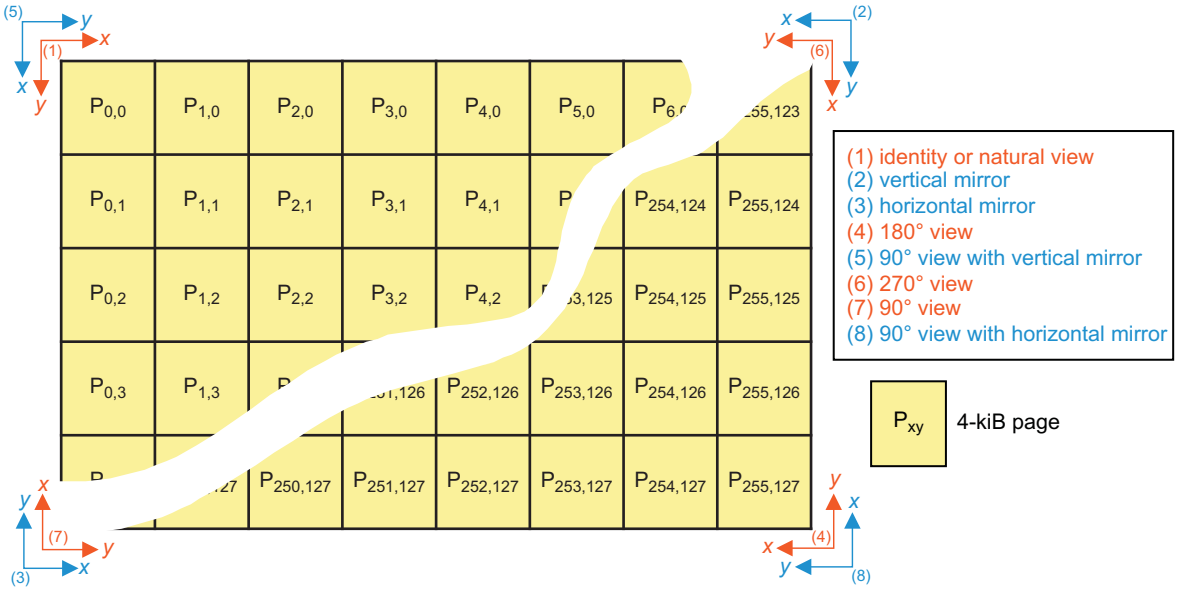
Figure 4-27 shows isometric transforms in the TILER container.

Mathematically speaking, all these transforms correspond to the composition of a 0º, 90º, 180º, or 270º rotation with an optional reflection. The nature of this orientation is based on the three following binary parameters:

- X to change the direction of the x axis of the TILER container
- Y to change the direction of the y axis of the TILER container
- S to swap the modified x and y axis

From now on, and in the rest of this document, the term orientation refers to any composition of a "quadrant" rotation with an optional horizontal - flip-flop - or vertical mirroring.

Figure 4-27. Isometric Transforms in the TILER Container



Selection of orientation:

1. For all initiators except, HD_VPSS, the tiled space is accessed in the 512 MB address range of (6000 0000h to 7FFF FFFFh). These initiators use the access the any of the 8 views by using the respective DMM_TILER_OR0 or DMM_TILER_OR1 registers.
2. For HD_VPSS, the TILER module supports its own 4-GB virtual addressing space, from address (1 0000 0000h to 1 FFFF FFFFh). These eight 512MB spaces correspond to the 8 availables views. With multiple display, capture and transcode path, using the unique 512 MB address space, these ports of HD_VPSS can simultaneously generate access in any of the 8 views.

4.2.2.8 Tile Definition

A tile is a subdivision of a page, aimed at:

- Representing a 2D block to better balance the accesses in both directions
- Ensuring that any "tiled" access that fits within a tile is made atomic in the SDRAM controller, and fits in a single SDRAM memory page
- Minimising the number of SDRAM page openings per 2D block transfer

The tile is defined as a 1KB 2D block, and a 4KB page as an array of 2 lines of 2 tiles each, as described in section detailing a 4KB page.

When the considered page is in an interleaved DMM section, it is necessary that:

- the DMM memory interleaving size of tiled accesses is set to 1 KB - a tile size - so that any tiled request that fits within a tile, fits in a single SDRAM memory page
- any request that spans over 2 or 4 tiles, is distributed on a maximum number of SDRAM memory controllers

4.2.2.8.1 TILER Virtual Addressing

The TILER can be virtually accessed in four different modes, namely 8-bit, 16-bit, 32-bit and page modes. Each mode defines the element granularity to apply isometric transforms, as summarized in Table 4-2.

Table 4-2. TILER Modes

Mode	Name	Granularity (element size)
0	8-bit tiled mode	8 bits
1	16-bit tiled mode	16 bits
2	32-bit tiled mode	32 bits
3	Page mode	4096 bytes

For instance, making a vertical mirroring of a 16-byte horizontal line containing the word 0001 0203 0405 0607 0809 0A0B 0C0D 0E0Fh, leads to:

- 0F0E 0D0C 0B0A 0908 0706 0504 0302 0100h in 8-bit tiled mode
- 0E0F 0C0D 0A0B 0809 0607 0405 0203 0001h in 16-bit tiled mode
- 0C0D 0E0F 0809 0A0B 0405 0607 0001 0203h in 32-bit tiled mode
- 0001 0203 0405 0607 0809 0A0B 0C0D 0E0Fh in page mode - unchanged as the element granularity is 4KB

Besides, as each of the eight orientations is available for any of the four modes, this gives 32 TILER addressing possibilities.

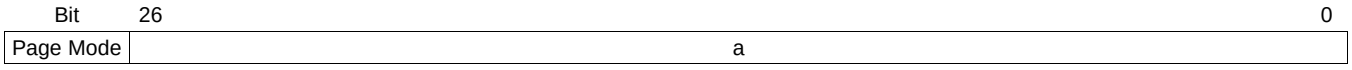
NOTE: The format that the TILER manages the tiled data is dictated by which 4 modes (8-,16-, 32-bit or paged) and which orientation (8 views) is used while accessing (read and write) the data.

As long as the data accessed is written and read in the same mode, it is assured that the response to the access will be correct. While accessing in the same mode, if written in one view and read back in the other view, then the read back data will be rotated and/or mirrored as per the view used.

4.2.2.8.2 Page Mode Virtual Addressing and Characteristics

When used in page mode, the 128-MiB TILER space is seen as an orientation-specific sequence of 32768 pages of 4 kiB each. The access sequence inside a page is left unchanged. Therefore, in page mode, the TILER is accessed similarly to any 128-MiB memory, with a 27-bit byte-based address.

Figure 4-28. Paged Mode Addressing



4.2.2.8.3 Tiled Mode Virtual Addressing and Characteristics

When used in tiled mode, the 128-MiB TILER space is seen as a giant frame-buffer - the container. The addressing and characteristics of this giant frame-buffer depends on:

- The tiled mode, which defines the considered atomic element size
- The orientation, which potentially swaps its x and y axis - and hence the container geometry

Table 4-3 summarizes tiled mode container characteristics.

Table 4-3. Tiled Mode Container Characteristics

Orientation			Element Size (bits)	Width (elements)	Height (elements)	Stride (bytes)	
S	Y	X				Progressive	Interlaced
0	x	x	8	16384	8192	16384	32768
			16	16384	4096	32768	65536
			32	8192	4096	32768	65536
1	x	x	8	8192	16384	8192	16384
			16	4096	16384	8192	16384
			32	4096	8192	16384	32768

As a result, the coordinate (x0, y0) of a pixel in an oriented view is translated in a virtual address as shown in Figure 4-29 and Figure 4-30, for all the four orientations.

Figure 4-29. Tiled Mode Addressing in 0° or 180° Orientations, (S = 0)

Bit	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8-bit tiled mode	y_0													x_0													
16-bit tiled mode	y_0													x_0													
32-bit tiled mode	y_0													x_0													

Figure 4-30. Tiled Mode Addressing in 90° or 270° Orientations, (S = 1)

Bit	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8-bit tiled mode	y_0														x_0												
16-bit tiled mode	y_0 x_0														x_0												
32-bit tiled mode	y_0														x_0												

Having understood the characteristic of the tiled container and the how the data is stored in bi-dimensional manner, it is easy to understand how stride is calculated. Stride is calculated as the number bytes between pixel (m,n) and pixel (m,(n+1)), where m = x co-ordinate of the image buffer, n = y co-ordinate of the image buffer.

For S = 0, that is, all views with 0º and 180º:

- 8-bit mode: stride_8bitmode = 4 bytes per sub-tile × 1 byte height of a sub-tile line × 8 sub-tiles per tile × 2 tiles per page × 256 pages = 4 × 1 × 8 × 2 × 256 = 16KB
- 16-bit mode: stride_16bitmode = 4 bytes per sub-tile × 2 bytes height of a sub-tile line × 8 sub-tiles per tile × 2 tiles per page × 256 pages = 4 × 2 × 8 × 2 × 256 = 32KB
- 32-bit mode: stride_32bitmode = 4 bytes per sub-tile × 2 bytes height of a sub-tile line × 8 sub-tiles per tile × 2 tiles per page × 256 pages = 4 × 2 × 8 × 2 × 256 = 32KB

For S = 1, that is, all views with 90º and 270º:

- 8-bit mode: stride_8bitmode = 4 bytes per sub-tile × 1 byte height of a sub-tile line × 8 sub-tiles per tile × 2 tiles per page × 128 pages = 4 × 1 × 8 × 2 × 128 = 8KB
- 16-bit mode: stride_16bitmode = 4 bytes per sub-tile × 1 byte height of a sub-tile line × 8 sub-tiles per tile × 2 tiles per page × 128 pages = 4 × 1 × 8 × 2 × 128 = 16KB
- 32-bit mode: stride_32bitmode = 4 bytes per sub-tile × 2 bytes height of a sub-tile line × 8 sub-tiles per tile × 2 tiles per page × 128 pages = 4 × 2 × 8 × 2 × 128 = 16KB

4.2.2.8.4 Element Ordering in the TILER Container

This section highlights how elements - 8-bit data, 16-bit data, 32-bit data or 4-kiB page - are ordered in the container. In other words, this section highlights how the path of incrementing virtual addresses is mapped in the container.

It is interesting to note that whatever the mode, and hence the element size, the sequence for ordering the elements in their related container is strictly similar and only depends on the related orientation. In other words:

- Mode is about element granularity
- Orientation is about change of orthornormal basis for ordering the elements in the mode-specific container

A corollary to the previous statements is that in a given mode the internal structure of an element is unchanged whatever the orientation. In page mode for instance, the offset of a word inside a page is invariant by orientation; the content of a page is always accessed in the same manner.

In the next sections, the natural container orthonormal basis is referenced as: (Xn , Yn) and the oriented orthonormal basis as: (Xo , Yo).

4.2.2.8.4.1 Natural View or 0º View (Orientation 0)

This orientation defined by S = 0, Y = 0, and X = 0 and means that the operated change of basis is:

X0 = Xn

Y0 = Yn

In any TILER mode, the elements are then ordered from left-to-right and then from top-to-bottom in their container, as shown in Figure 4-31 and Figure 4-32.

Figure 4-31. Tiled Mode Ordering of Elements in Natural View

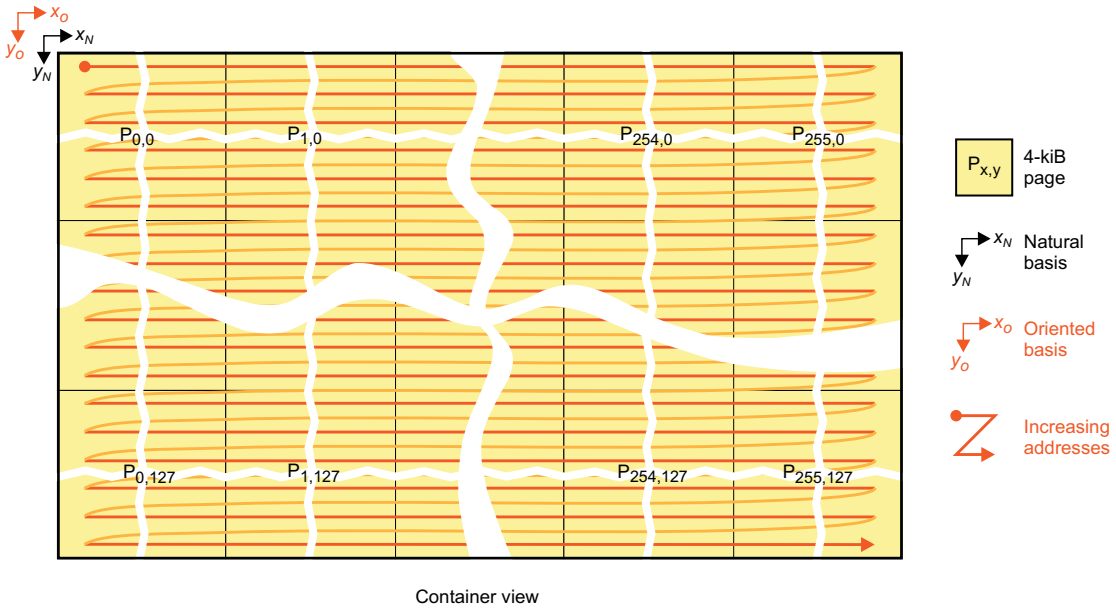
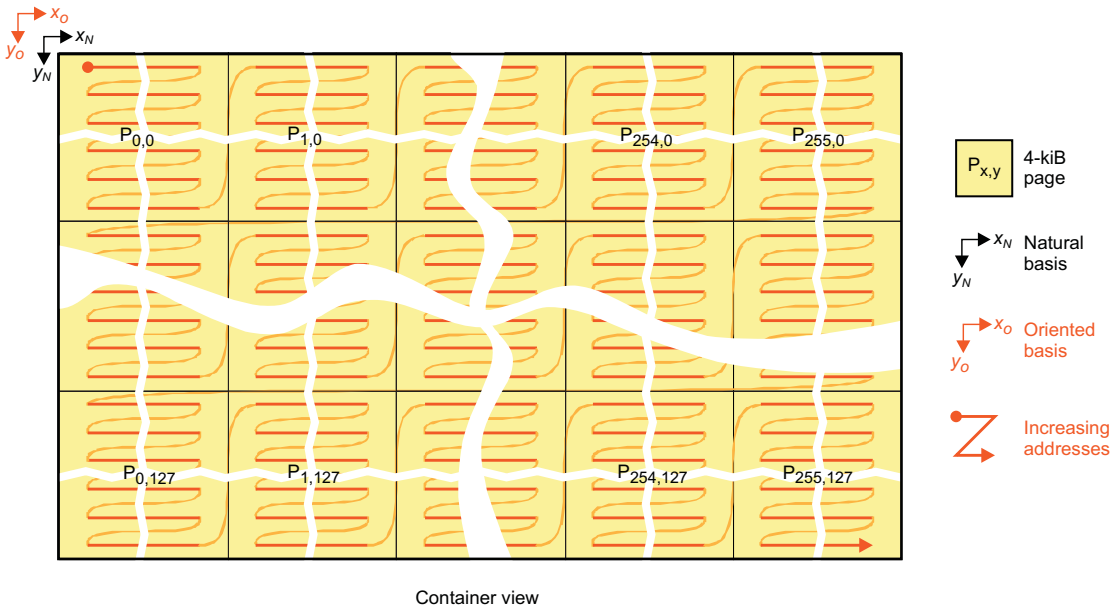


Figure 4-32. Page Mode Ordering of Elements in Natural View



4.2.2.8.4.2 0º View with Vertical Mirror or 180º View with Horizontal Mirror (Orientation 1)

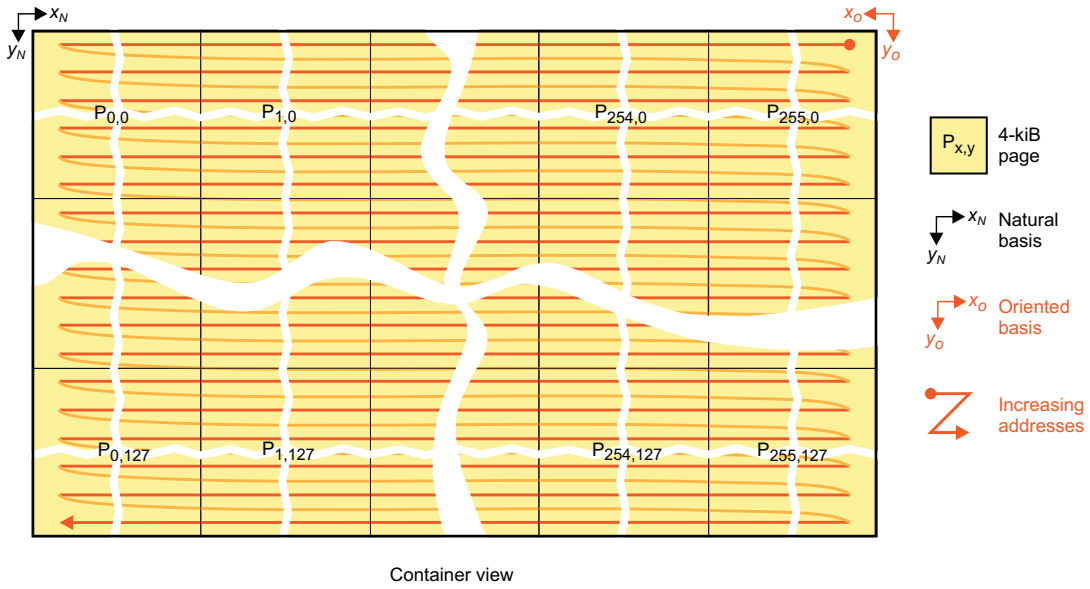
This orientation defined by S = 0, Y = 0, and X = 1 and means that the operated change of basis is:

X0 = -Xn

Y0 = Yn

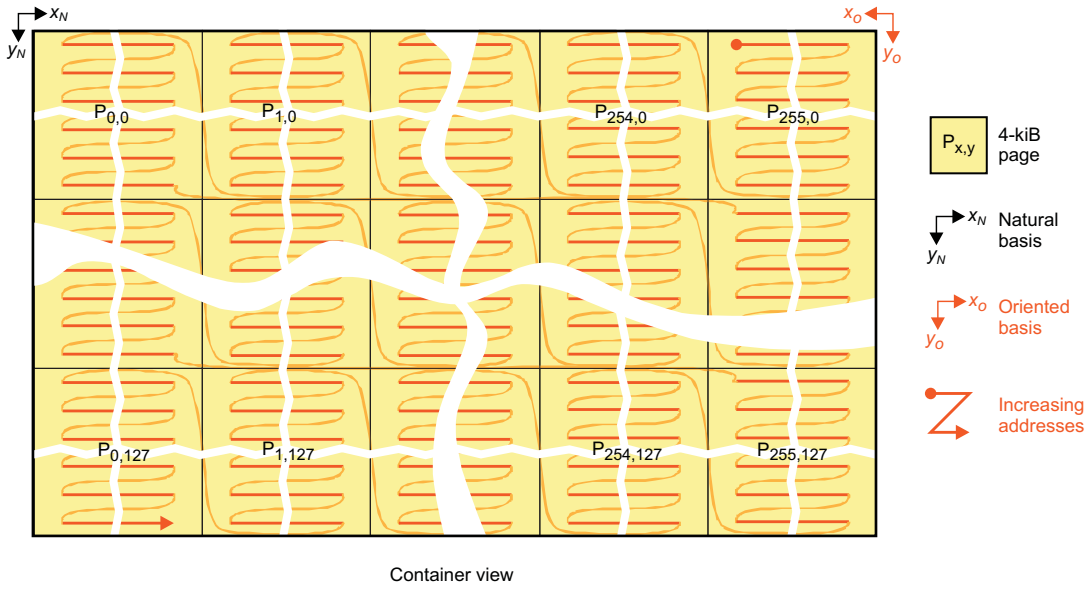
In any TILER mode, the elements are then ordered from right-to-left and then from top-to-bottom in their container, as shown in Figure 4-33 and Figure 4-34.

Figure 4-33. Tiled Mode Ordering of Elements in 0° View with Vertical Mirror



Container view

Figure 4-34. Page Mode Ordering of Elements in 0° View with Vertical Mirror



Container view

4.2.2.8.4.3 0º View with Horizontal Mirror or 180º View with Vertical Mirror (Orientation 2)

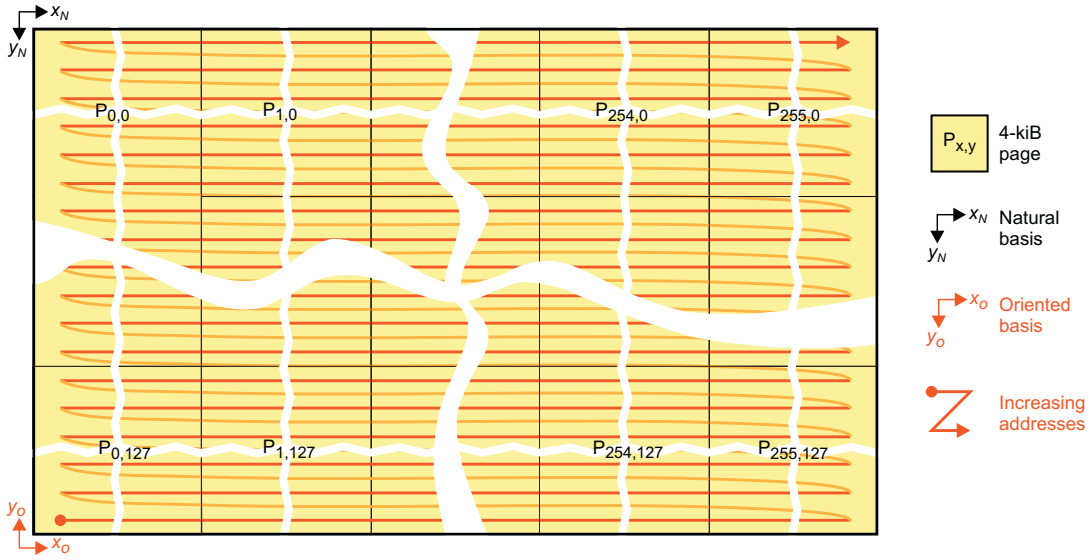
This orientation defined by S = 0, Y = 1, and X = 0 and means that the operated change of basis is:

X0 = Xn

Y0 = -Yn

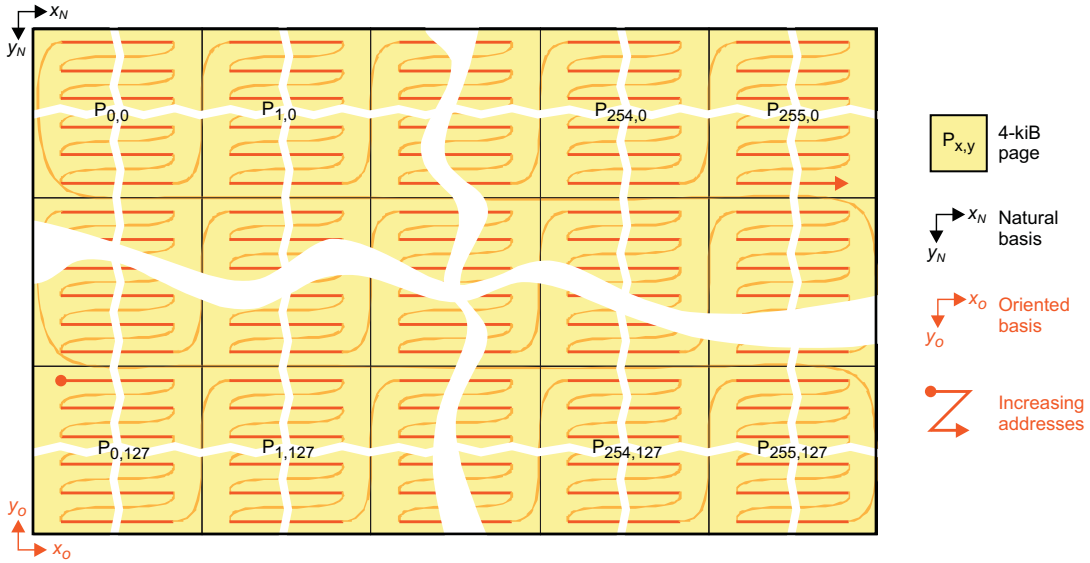
In any TILER mode, the elements are then ordered from left-to-right and then from bottom-to-top in their container, as shown in Figure 4-35 and Figure 4-36.

Figure 4-35. Tiled Mode Ordering of Elements in 0° View with Horizontal Mirror



Container view

Figure 4-36. Page Mode Ordering of Elements in 0° View with Horizontal Mirror



Container view

4.2.2.8.4.4 180º View (Orientation 3)

This orientation defined by S = 0, Y = 1, and X = 1 and means that the operated change of basis is:

X0 = -Xn

Y0 = -Yn

In any TILER mode, the elements are then ordered from right-to-left and then from bottom-to-top in their container, as shown in Figure 4-37 and Figure 4-38.

Figure 4-37. Tiled Mode Ordering of Elements in 180° View

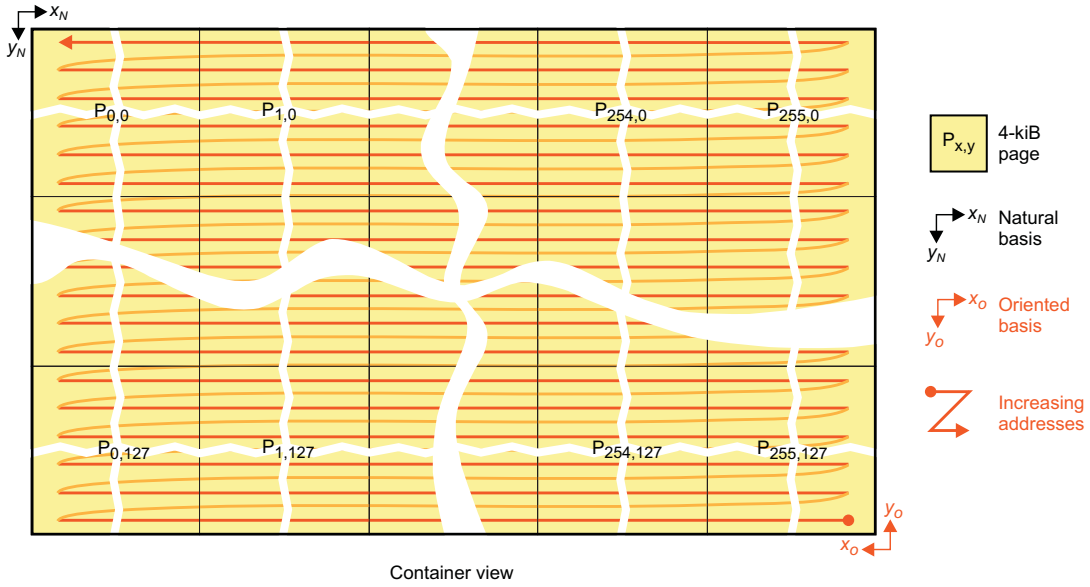
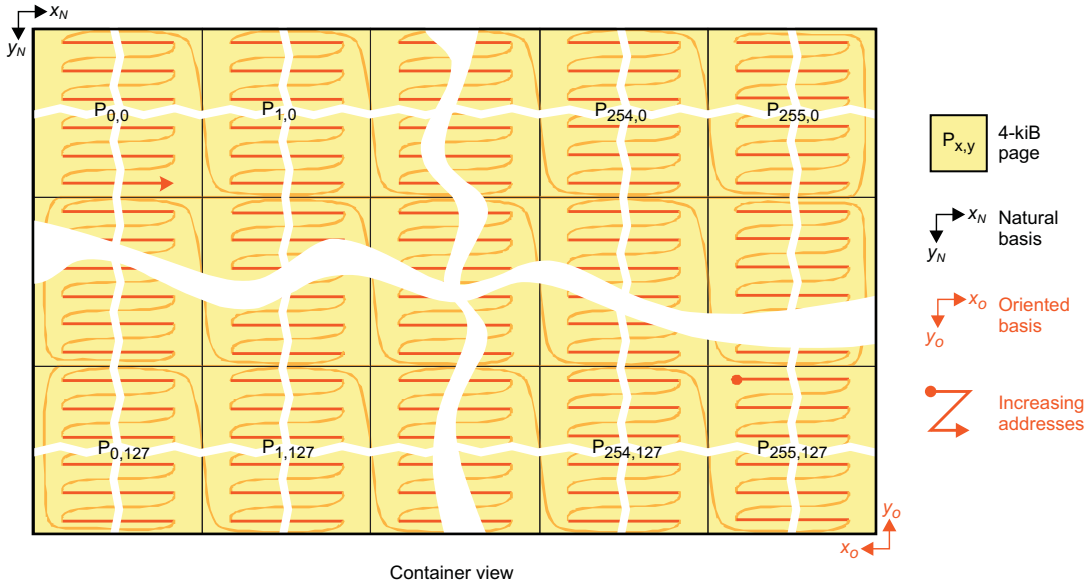


Figure 4-38. Page Mode Ordering of Elements in 180° View



4.2.2.8.4.5 90º View with Vertical Mirror or 270º View with Horizontal Mirror (Orientation 4)

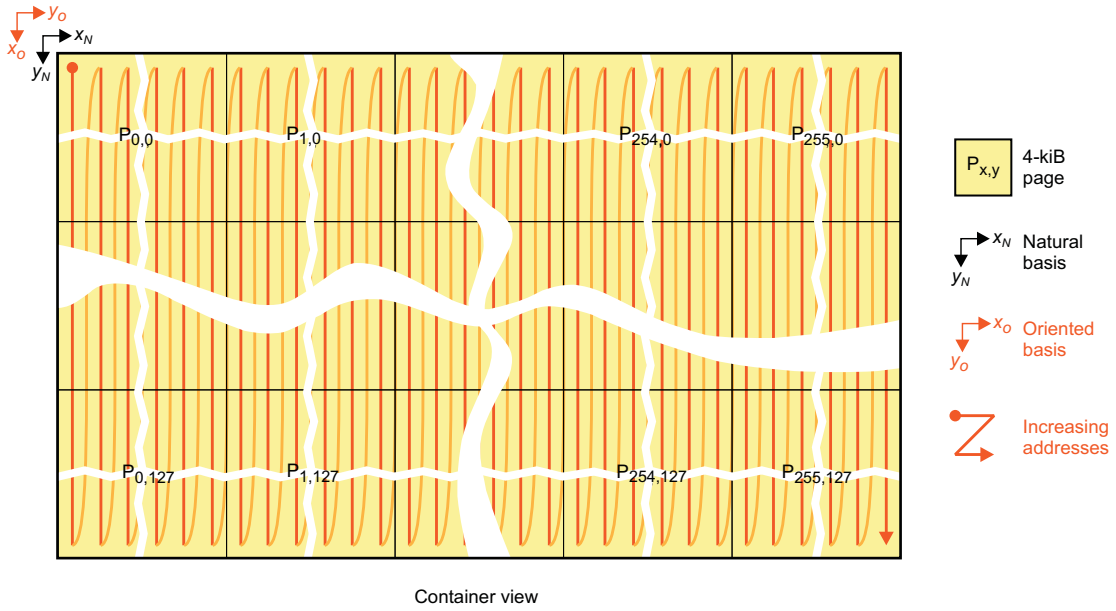
This orientation defined by S = 1, Y = 0, and X = 0 and means that the operated change of basis is:

X0 = Yn

Y0 = Xn

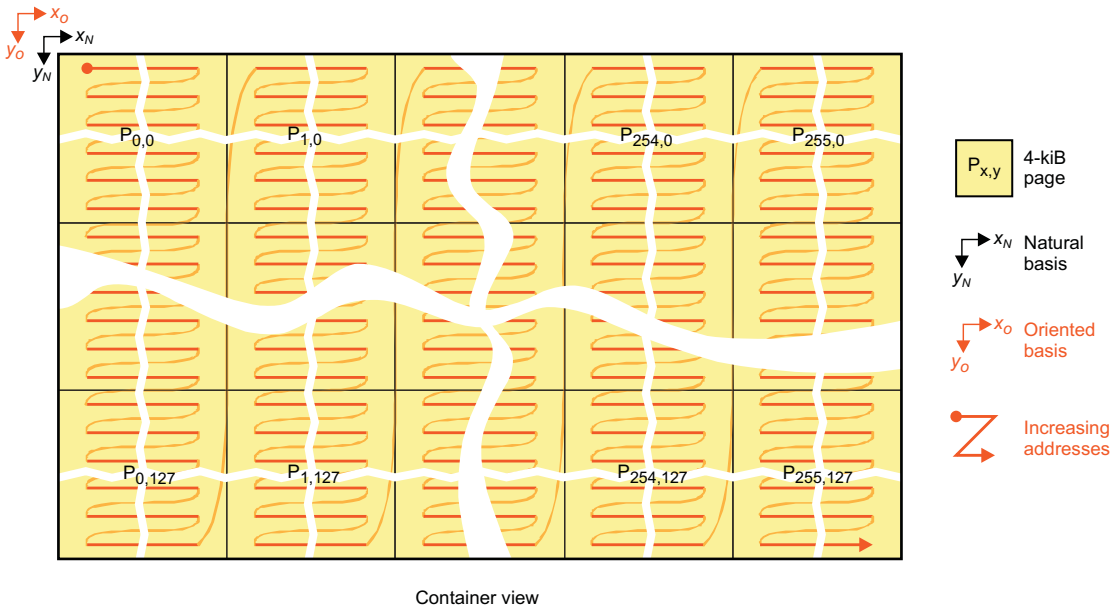
In any TILER mode, the elements are then ordered from top-to-bottom and then from left-to-right in their container, as shown in Figure 4-39 and Figure 4-40.

Figure 4-39. Tiled Mode Ordering of Elements in 90º View with Vertical Mirror



Container view

Figure 4-40. Page Mode Ordering of Elements in 90º View with Vertical Mirror



Container view

4.2.2.8.4.6 270º View (Orientation 5)

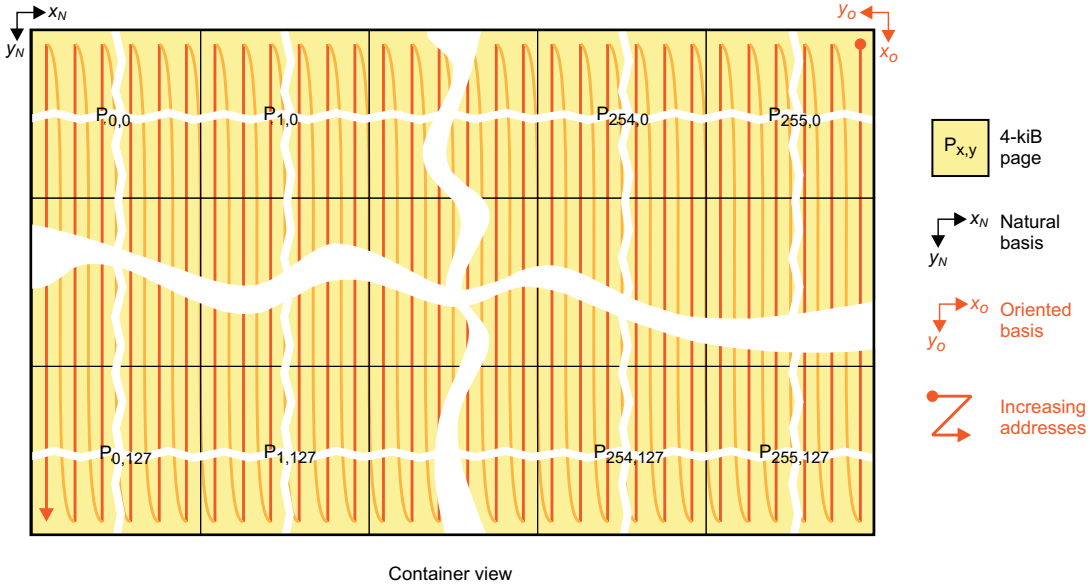
This orientation defined by S = 1, Y = 0, and X = 1 and means that the operated change of basis is:

X0 = -Yn

Y0 = -Xn

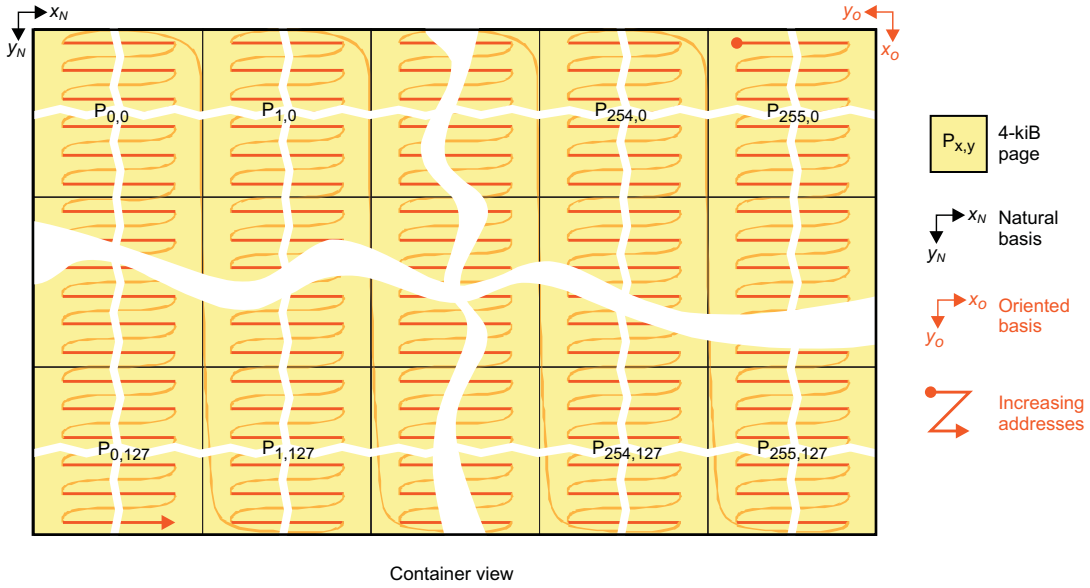
In any TILER mode, the elements are then ordered from top-to-bottom and then from right-to-left in their container, as shown in Figure 4-41 and Figure 4-42.

Figure 4-41. Tiled Mode Ordering of Elements in 270° View



Container view

Figure 4-42. Page Mode Ordering of Elements in 270° View



Container view

4.2.2.8.4.7 90º View (Orientation 6)

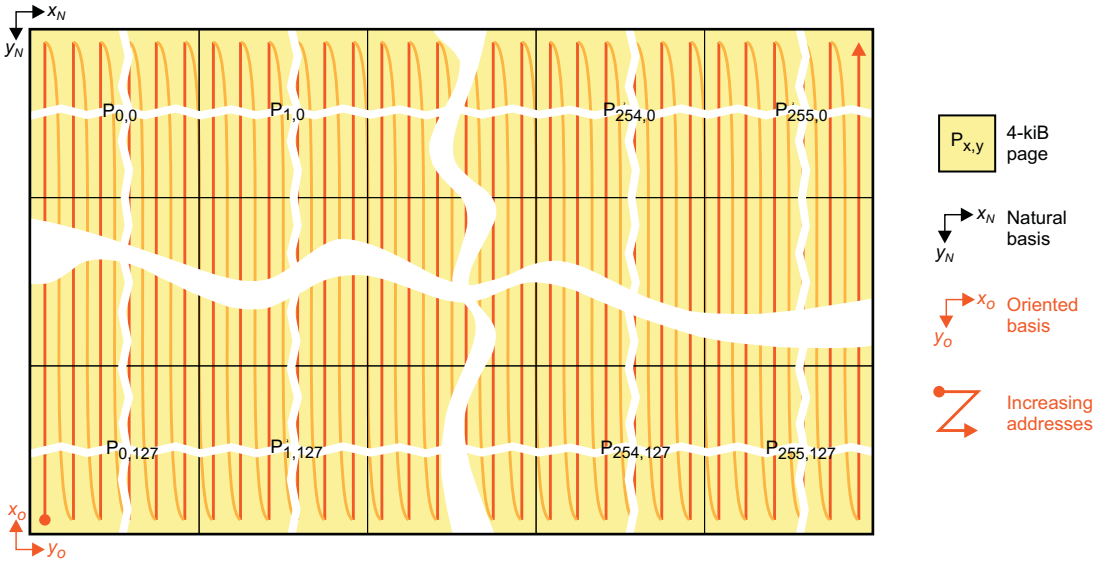
This orientation defined by S = 1, Y = 1, and X = 0 and means that the operated change of basis is:

X0 = -Yn

Y0 = Xn

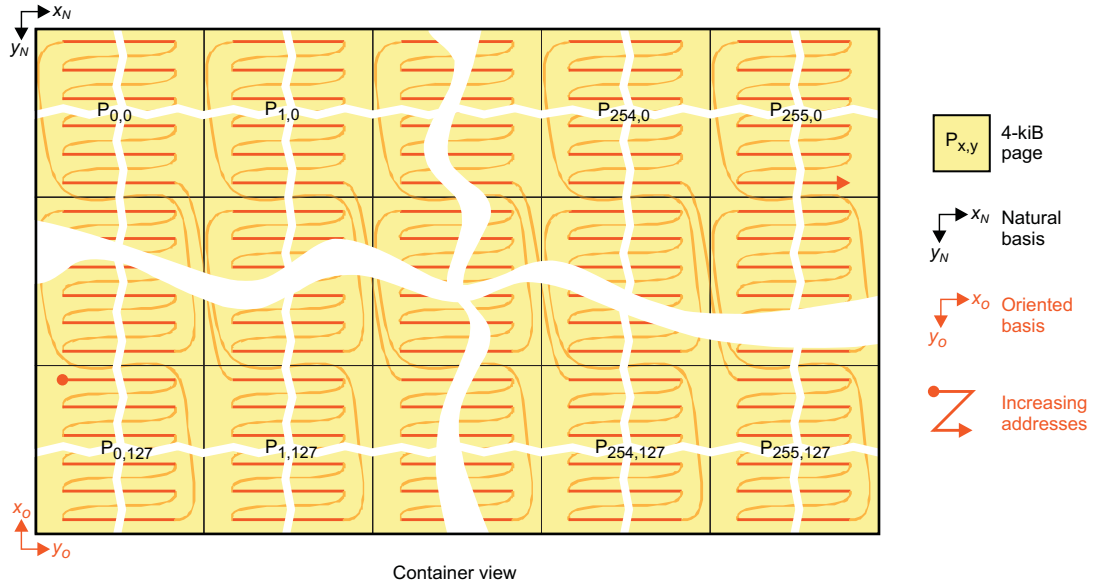
In any TILER mode, the elements are then ordered from bottom-to-top and then from left-to-right in their container, as shown in Figure 4-43 and Figure 4-44.

Figure 4-43. Tiled Mode Ordering of Elements in 90° View



Container view

Figure 4-44. Page Mode Ordering of Elements in 90° View



Container view

4.2.2.8.4.8 90º View with Horizontal Mirror or 270º View with Vertical Mirror (Orientation 7)

This orientation defined by S = 1, Y = 1, and X = 1 and means that the operated change of basis is:

X0 = -Yn

Y0 = -Xn

In any TILER mode, the elements are then ordered from bottom-to-top and then from right-to-left in their container, as shown in Figure 4-45 and Figure 4-46.

Figure 4-45. Tiled Mode Ordering of Elements in 90º View with Horizontal Mirror

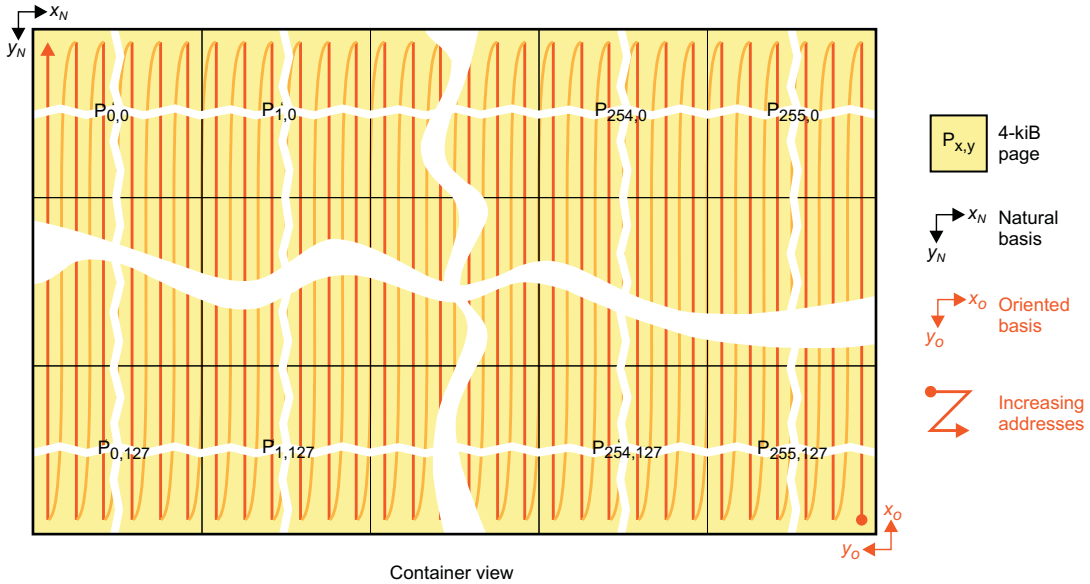
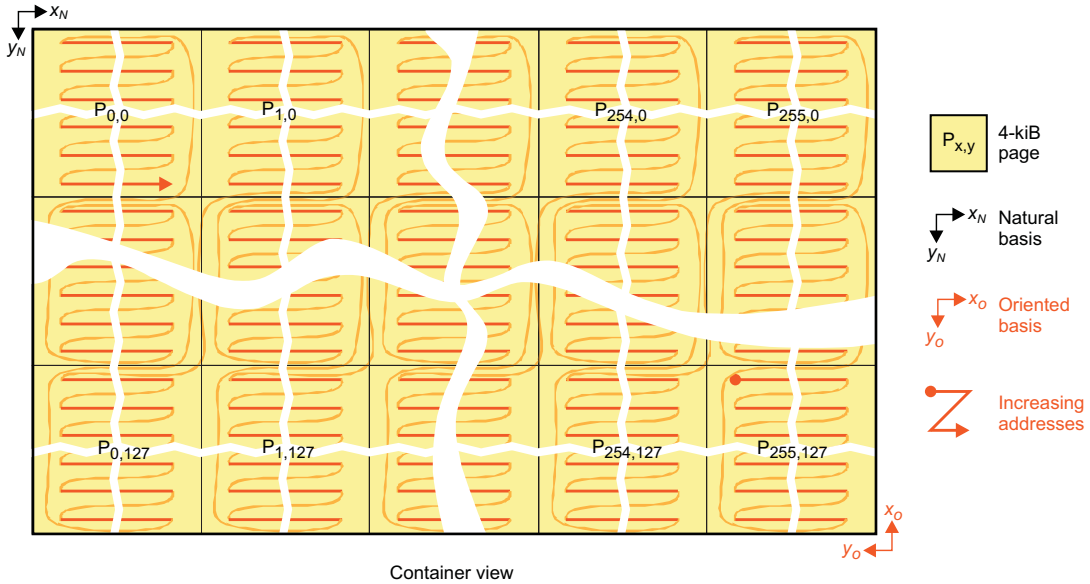


Figure 4-46. Page Mode Ordering of Elements in 90º View with Horizontal Mirror



4.2.2.8.5 Tiled and Rotational Efficiency

Because of the distribution of the image data in both direction and co-locating them, using sub-tiling, the Image processing algorithms running on the HDVICPs are more efficient by making fewer accesses to the SDRAM, to fetch Macro-blocks of data.

In the Device, the DMA engine of HD_VPSS, also known as VPDMA, has internal line buffers of its clients. For every tiled access, it is ensured that rastering tiled data has zero overhead, when processing an entire frame, or 2^{2n} number of lines.

4.3 Use Case

4.3.1 DMM Basic Register Setup

This section describes basic steps to initialize the DMM, in the device.

1. Set base address, DMM_PAT_VIEW_MAP_BASE to value 8000 0000h, which corresponds to base SDRAM address in the device memory map.
2. Program the four PAT views, by writing DMM_PAT_VIEW_MAP__0..3 registers. By default, all the four PAT views are having value of 0, which implies Direct access translation for all the four tiled modes of (8,16,32-bit and paged mode) and all the four containers will overlap with their base address mapped to 8000 0000h.
3. Program PAT views for all initiators, indexed by ConnID, by programming the DMM_PAT_VIEW__0..1 registers. There are 4 possible views available in device. Default is for all initiators to use PAT view 0.
4. Program PEG priority for all the initiators. By default - all are having priority = 0.
5. Program the TILER orientation for all the initiators, by writing to DMM_TILER_OR__0..1 registers. By default - all will access tiled data in Orientation = 0, which is the normal view.
6. Program Section mapping, based on the SDRAM configuration of the system, which depends on the size of the SDRAM on one or both the banks.
   - Example 1: Symmetrical DDRs on both EMIF banks. Each size = 512MB to add up to 1GB of total SDRAM memory. In all DMM_LISA_MAP__0..3 registers, program value 8064 0300h, which implies 1GB section starting from System address : 8000 0000h, mapped to 0000 0000h of both EMIF0 and EMIF1, with interleaving of 128 bytes enabled. Note that the above example uses only one DMM section and the three other are identical. There are other ways to define up-to 4 sections and configure each differently

4.3.1.1 PAT Direct Access Translation with Distinct 128 MB Containers for Each of the 4 Modes

Following is a simplistic use case, bypassing the LUTs for address translation and setting aside a chunk of 128MB contiguous memory, with base address 128MB aligned, for each of the 4 modes (8, 16,32-bit and paged mode). The same addresses will be used by all initiators in the system.

Configure PAT view 0:

Program DMM_PAT_VIEW_MAP__0 to a value of 0302 0100h. Here : 8-bit mode container will be mapped to 8000 0000h. 16-bit mode container will be mapped to 8800 0000h. 32-bit mode container will be mapped to 9000 0000h. Paged mode container will be mapped to 9800 0000h.

Configure all initiators to use PAT view 0:

Program DMM_PAT_VIEW__0..1 to a value of 0.

4.3.2 Simple LUT Bypass Use Case: Arrangement of Video Buffers

This section describes one possible arrangement of H.264 video buffers which are in YUV 420 format. Setup is as follows:

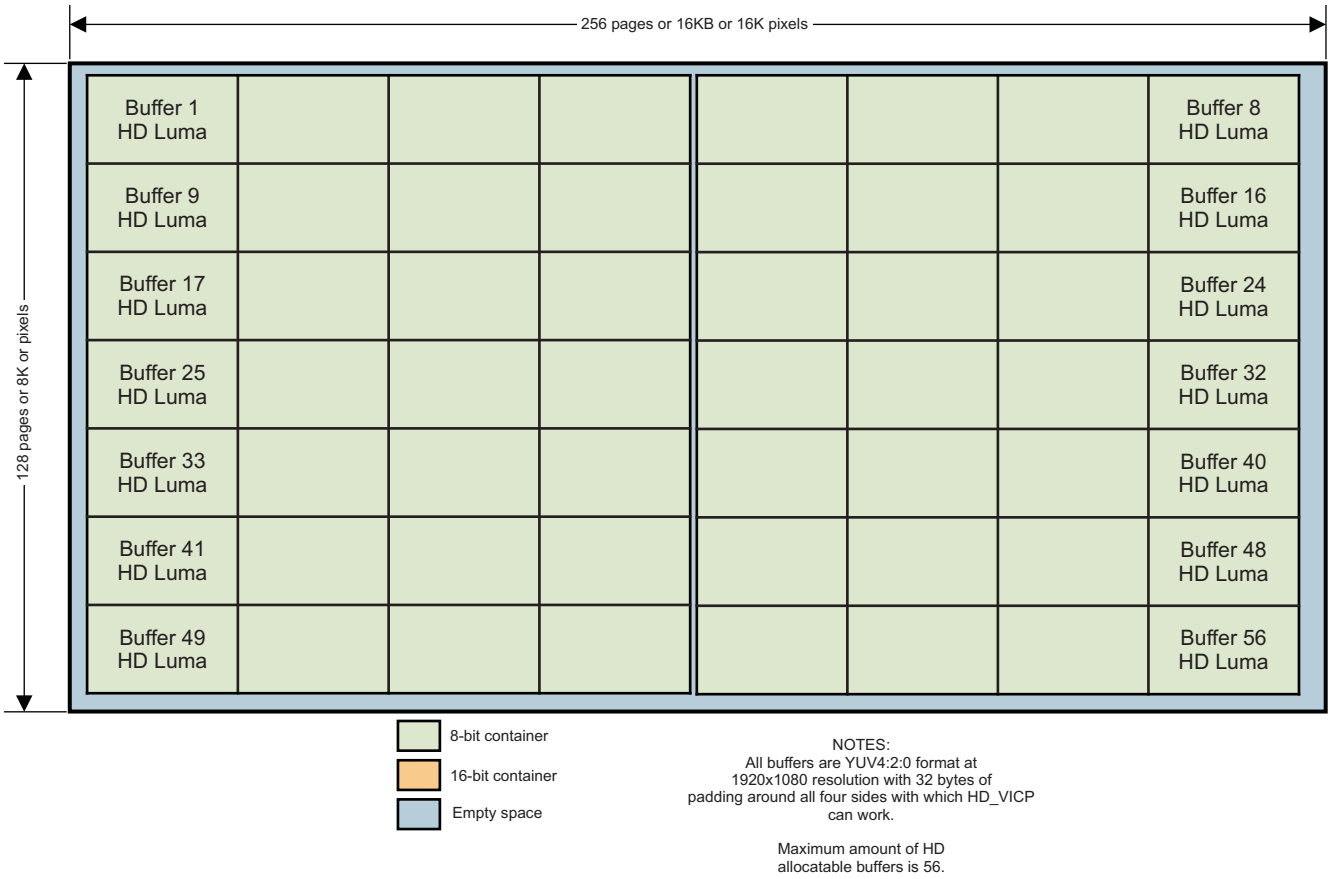
1. The buffers size if 1920 × 1080
2. The Luma buffers are allocated in the 8-bit mode container
3. The Chroma buffers are allocated in the 16-bit mode container
4. The H.264 algorithm needs 32 bytes of padding on each side for Luma buffer and 16 bytes padding in each side for a chroma buffer
5. All buffers are allocated on a 4K aligned address

4.3.2.1 Luma Buffers

In 8-bit tiled mode (Figure 4-47), each 4K page is arranged as 64-pixels wide and 64-pixels high, where each pixel is of size 8-bits. In YUV 420 format, the Luma buffer of the image is 1920 × 1080 and each pixel is 8-bits. Thus, each buffer is 1920 bytes wide. With a pad of 32 bytes on each side, the buffer is (1920 + 64 =) 1984-bytes wide. So one video buffer needs (1984/64 =) 31 pages, along the width of the tiler container. So the tiler container can fit (256/31 ~) 8 buffers, along its width.

Similarly, with a pad of 32 bytes on top and bottom, the buffer is (1080 + 64 =) 1144. Rounding it to (1144 + 8 =) 1152, so that buffer is allocated on page boundary. So one video buffer needs (1152/64 =) 18 pages, along the height of the tiler container. So the tiler container can fit (128/18 ~) 7 buffers, along its height.

Figure 4-47. Buffer Arrangement for HD Luma Buffers in 128MB 8-bit Mode Container

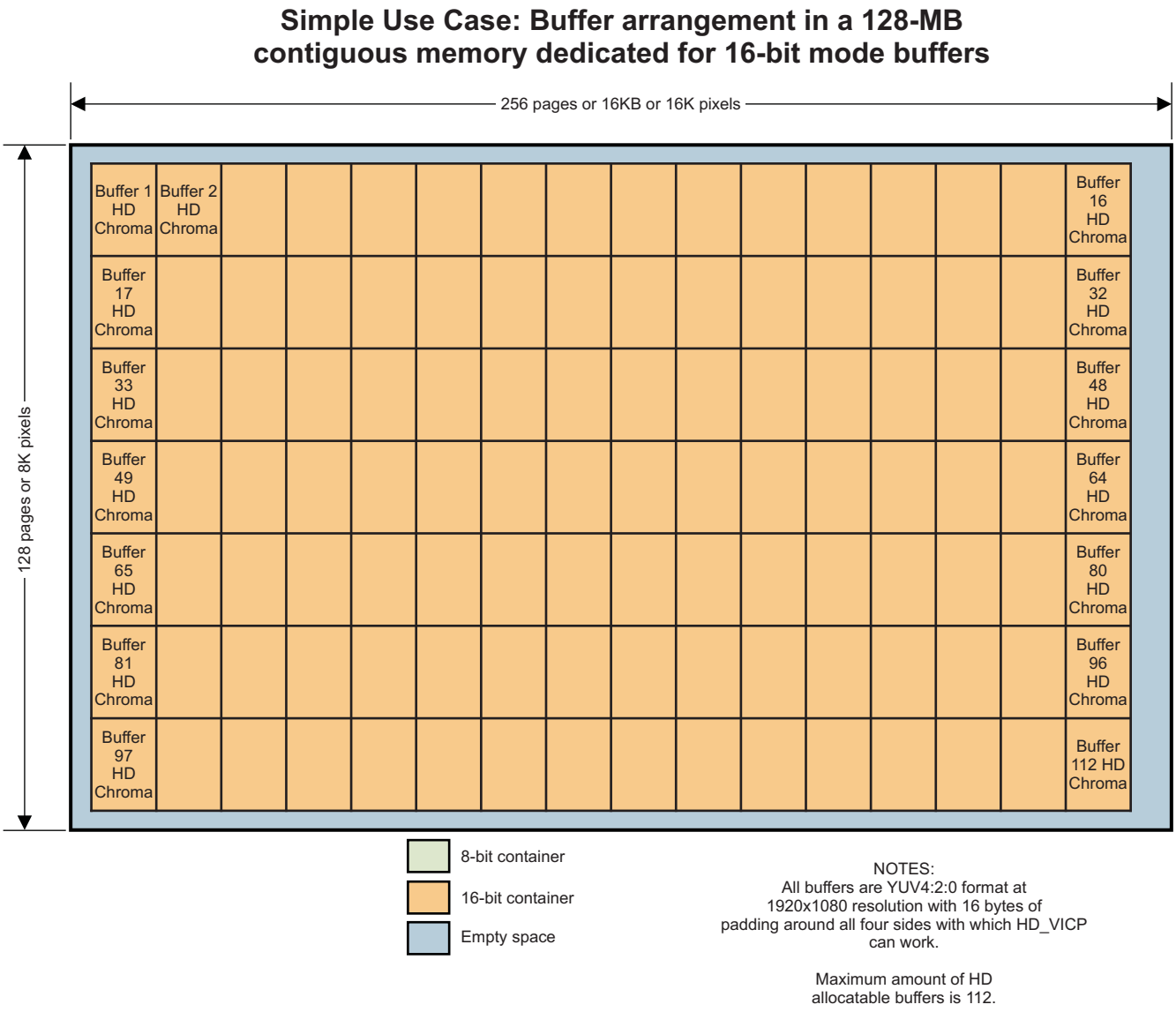


4.3.2.2 Chroma Buffers

In 16-bit tiled mode (Figure 4-48), each 4K page is arranged as 64-pixels wide and only 32-pixels high, where each pixel is of size 16-bits. In YUV 420 format, the Chroma buffer of the image is 960 × 540 and each pixel is 16-bits. Thus, each buffer is 960-bytes wide. With a pad of 16 bytes on each side, the buffer is (960 + 32 =) 992-bytes wide. Rounding it to (992 + 32 =) 1024, so that buffer is allocated on page boundary. So one video buffer needs (1024/64 =) 16 pages, along the width of the tiler container. So the tiler container can fit (256/16 =) 16 buffers, along its width.

Similarly, with a pad of 16 bytes on top and bottom, the buffer is (540 + 32 =) 572. Rounding it to (572 + 4 =) 576, so that buffer is allocated on page boundary. So one video buffer needs (576/32 =) 18 pages, along the height of the tiler container. So the tiler container can fit (128/18 ~) 7 buffers, along its height.

Figure 4-48. Buffer Arrangement for HD Chroma Buffers in 128MB 16-bit Mode Container



NOTE: The above scheme of Luma and Chroma buffer arrangement can be optimized further, by allocating buffers on sub-tile boundary, instead of page boundary (which is required when using LUT for address translation).

4.3.3 LUT Refill Using the PAT Refill Engines

There are two LUTs, each of size 256 × 128 elements, in the device.

There four refill engines, along with their own set of registers, to program the LUTs.

Also in this section are described in five different ways to program the LUT.

1. Simple manual area refill
2. Single auto-configured area refill
3. Chained auto-configured area refill
4. Synchronised auto-configured area refill
5. Cyclic synchronised auto-configured area refill

Some of the guidelines, in programming the LUTs are mentioned below:

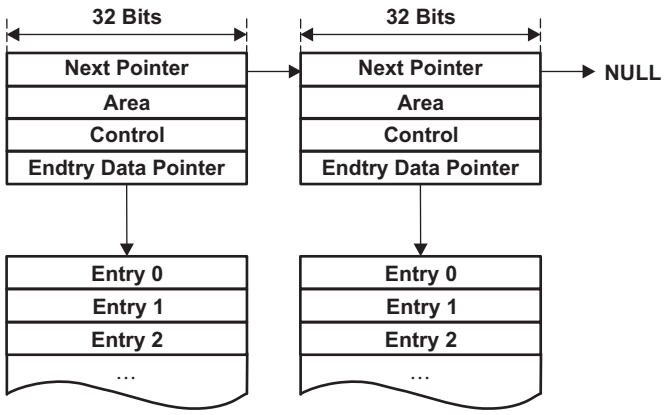
1. Entry table must be in DDR and cannot be split between 2 EMIF banks, when data interleaving enabled.
2. The data table should start from address 16-byte aligned.
3. With one set of descriptor, up to 64 LUT entries can be programmed, in any 2D area dimension.

4.3.3.1 PAT Memory Mapped Refill Descriptors

A PAT descriptor is a singly-linked list node as shown in Figure 4-49, containing:

- a description of the LUT area to reload
- a description of how to reload the defined LUT area
- a pointer to the location where the corresponding LUT entries are stored

Figure 4-49. PAT Descriptor Node



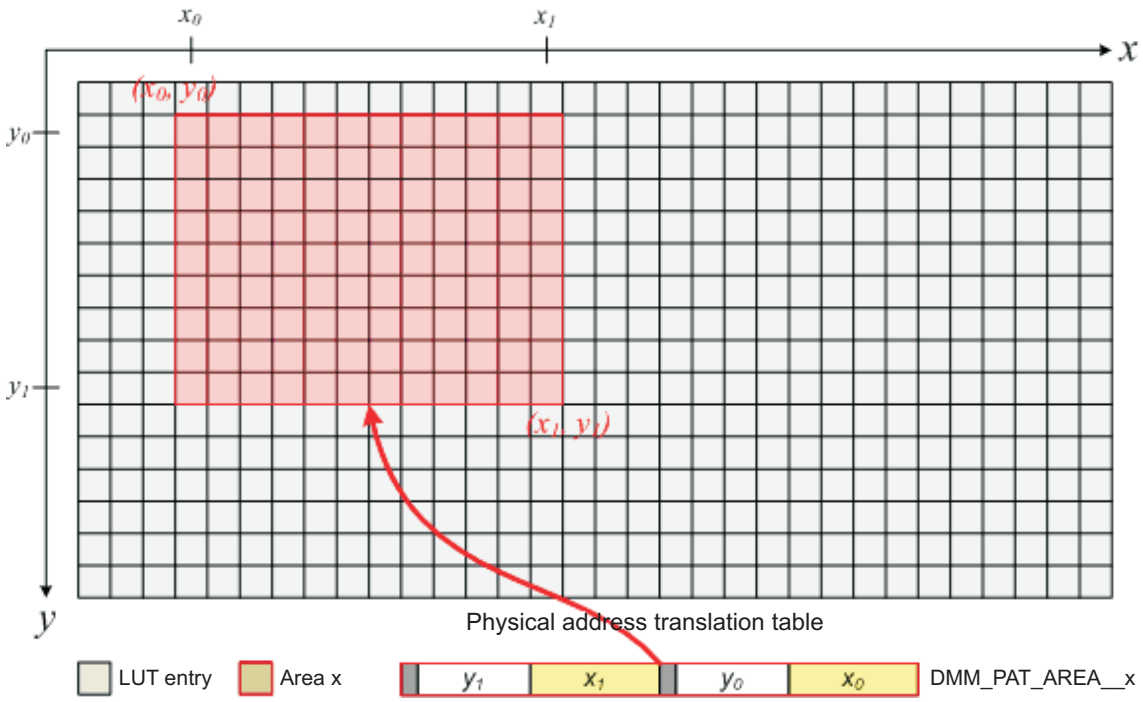
The PAT descriptor node and the entry data pointer must both be 16-byte aligned. A typical 'C' language description of this node is as follows:

```
struct PAT_desc {
         struct PAT_desc *next;
   u32_t            area;
   u32_t            ctrl;
   PAT_data        *data;
} __attribute__ ((aligned(16)));
```

4.3.3.1.1 PAT Area Refill Description

The area refill description is actually describing the top-left and bottom right corner of the PAT vector table as shown in Figure 4-50.

Figure 4-50. PAT Area Description



4.3.3.1.2 PAT Data Description

The pointer to the address in SDRAM, containing the memory based Data table, should be 16 byte aligned. Each field in this table is a 32 bit value, corresponding to the value to be programmed in one LUT entry. Though it is a 32 bit value, only 19 bits [30:12] are relevant. Bit 31 and bits [0:11] are ignored by the PAT refill engine.

4.3.3.1.3 PAT Memory Dump

This section presents how to dump the PAT LUT. This has to be done using the direct access mode of PAT, following the below steps:

1. Set at least one of the refill engine to direct LUT access mode. This has to be done using DMM_PAT_CONFIG register. Only bit 0 to 3 are used for mode of refill engine 0 to 3. (Value 0 is normal mode, value 1 is direct LUT access mode).
2. Set system coordinates we would like to read/write in the LUT. This has to be done using DMM_PAT_AREA__x. x is the refill engine index that has to be coherent with the refill engine used in direct mode (DMM_PAT_CONFIG). Only (x0,y0) is used in direct mode.
3. Select the LUT_ID using DMM_PAT_CTRL__x register.
4. Read (or write) the data in DMM_PAT_DATA__x register. x is the refill engine index.

4.3.3.1.4 PAT Refill Direction

The DMM_PAT_CTRL__x.DIRECTION field gives the “order” of the area refill with the same “SYX” encoding as in the TILER orientations:

- 0: from left to right then from top to bottom
- 1: from right to left then from top to bottom
- 2: from left to right then from bottom to top
- 3: from right to left then from bottom to top
- 4: from top to bottom then from left to right
- 5: from top to bottom then from right to left
- 6: from bottom to top then from left to right
- 7: from bottom to top then from right to left

This feature allows the initiators HD_VPSS and HDVICPs to start an access to an on going refill area. In this case, the direction field is set in accordance with the orientation accesses made by the initiators. The DMM_PAT_STATUS__x.ERROR is asserted to 1 if an access is done to a not yet refilled page.

4.3.3.2 Simple Manual Area Refill

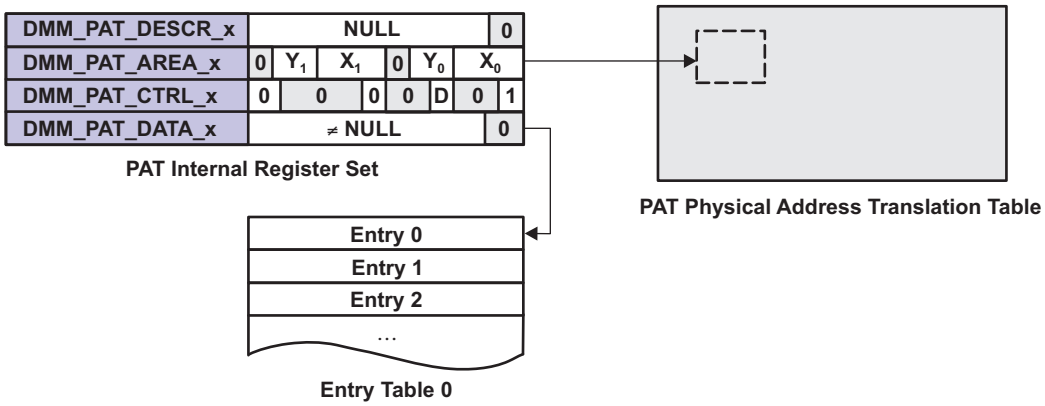
These steps have to be performed to create the 16-byte aligned memory-mapped entry table containing all entries of the defined area, as seen in Figure 4-51.

- Write the DMM_PAT_AREA_i register with the relevant (x0, y0) (x1, y1) area definition
- Write the DMM_PAT_DATA_i register with the physical address of the created entry table
- Write the DMM_PAT_CTRL_i register with the requested refill direction and assert the

DMM_PAT_CTRL_i[0] START bit:

- The refill is done when the DMM_PAT_STATUS_i[3] DONE bit is set.
- A new refill can be initiated when the DMM_PAT_STATUS_i[0] READY bit is set.

Figure 4-51. DMM Simple Manual Area Refill

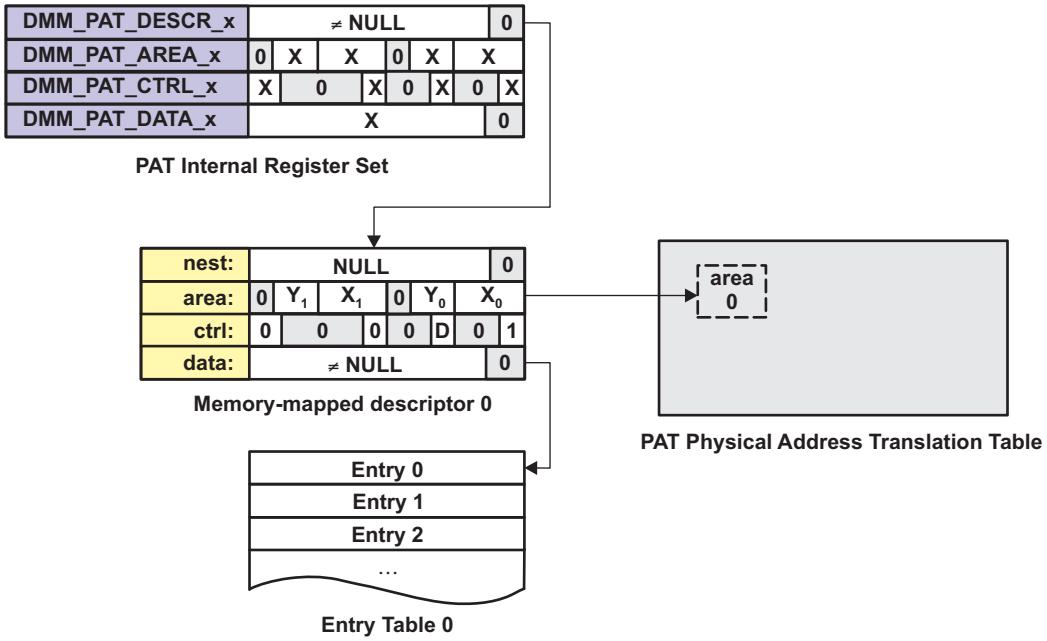


4.3.3.3 Single Auto-Configured Area Refill

These steps have to be performed to create the 16-byte aligned memory-mapped entry table containing all entries of the defined area, as seen in Figure 4-52.

1. Create a 16-byte aligned memory-mapped descriptor structure where:
   (a) the next field is set to NULL
   (b) the area field is set with the relevant (x0, y0) (x1, y1) area definition
   (c) the ctrl field is set with the requested direction D and the start bit asserted to start refilling as soon as this descriptor enters the PAT refill engine
   (d) the data field is set to the physical address of the created entry table
2. The DMM_PAT_DESCR_i register with the physical address of the created descriptor.
3. The refill is done when the DMM_PAT_STATUS_i[3] DONE bit is set.
4. A new refill can be initiated when the DMM_PAT_STATUS_i[0] READY bit is set.

Figure 4-52. DMM Single Auto-configured Area Refill

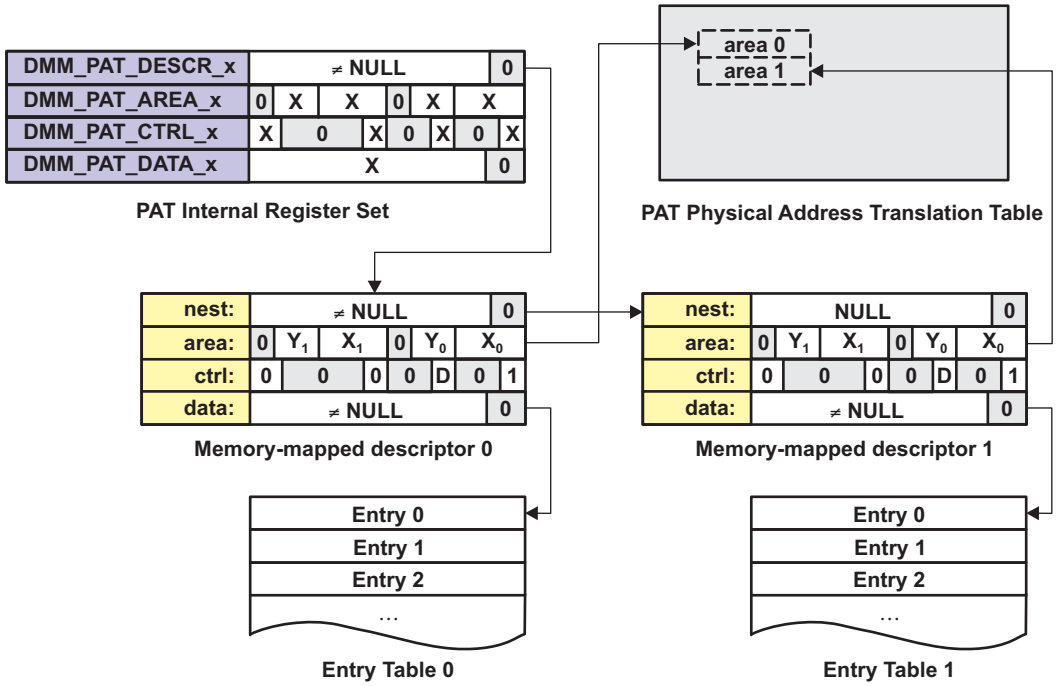


4.3.3.4 Chained Auto-Configured Area Refill

These steps have to be performed to create one 16-byte aligned memory-mapped entry table per area containing the entries for the corresponding area, as seen in Figure 4-53.

1. Create one 16-byte aligned memory-mapped descriptor structures per area where:
   (a) the next field is set to the physical address of the next descriptor or NULL for the last one
   (b) the area field is set with the relevant (x0, y0) (x1, y1) area definition
   (c) the ctrl field is set with the requested direction D and the START bit asserted to start refilling as soon as the previous area refill is done
   (d) the data field is set to the physical address of the corresponding entry table
2. Write the DMM_PAT_DESCR_i register with the physical address of the first created descriptor.
3. Each area refill is done when the DMM_PAT_STATUS_i[3] DONE bit is set.
4. All area refills are done when the DMM_PAT_STATUS_i[0] READY bit is set.
5. A new refill can be initiated when the DMM_PAT_STATUS_i[0] READY bit is set.

Figure 4-53. DMM Chained Auto-configured Area Refill

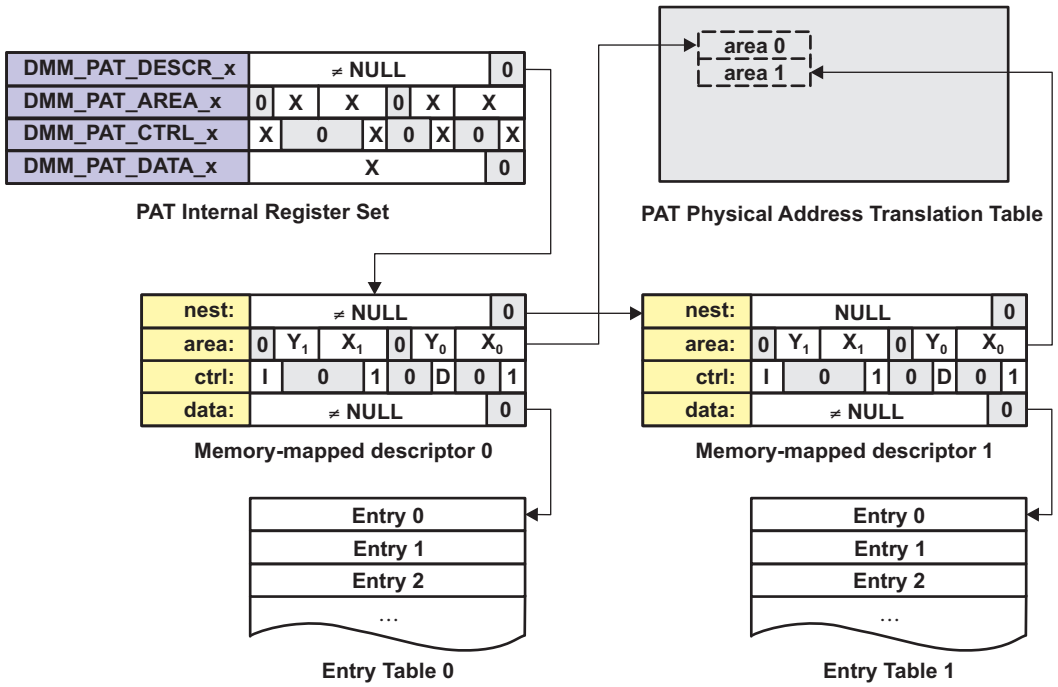


4.3.3.5 Synchronised Auto-configured Area Refill

These steps have to be performed to create one 16-byte aligned memory-mapped entry table per area containing the entries for the corresponding area, as seen in Figure 4-54.

1. Create one 16-byte aligned memory-mapped descriptor structures per area where:
   (a) the next field is set to the physical address of the next descriptor or NULL for the last one
   (b) the area field is set with the relevant (x0, y0) (x1, y1) area definition
   (c) the ctrl field is set with the synchronising initiator identifier I, the SYNC bit asserted, the requested direction D, and the START bit asserted to start refilling as soon as the previous area refill is done and initiator I has made one access in the previous area
   (d) the data field is set to the physical address of the corresponding entry table
2. Write the DMM_PAT_DESCR_i register with the physical address of the first created descriptor.
3. Each area refill is done when the DMM_PAT_STATUS_i[3] DONE bit is set.
4. All area refills are done when the DMM_PAT_STATUS_i[0] READY bit is set.
5. A new refill can be initiated when the DMM_PAT_STATUS_i[0] READY bit is set.

Figure 4-54. DMM Synchronised Auto-configured Area Refill

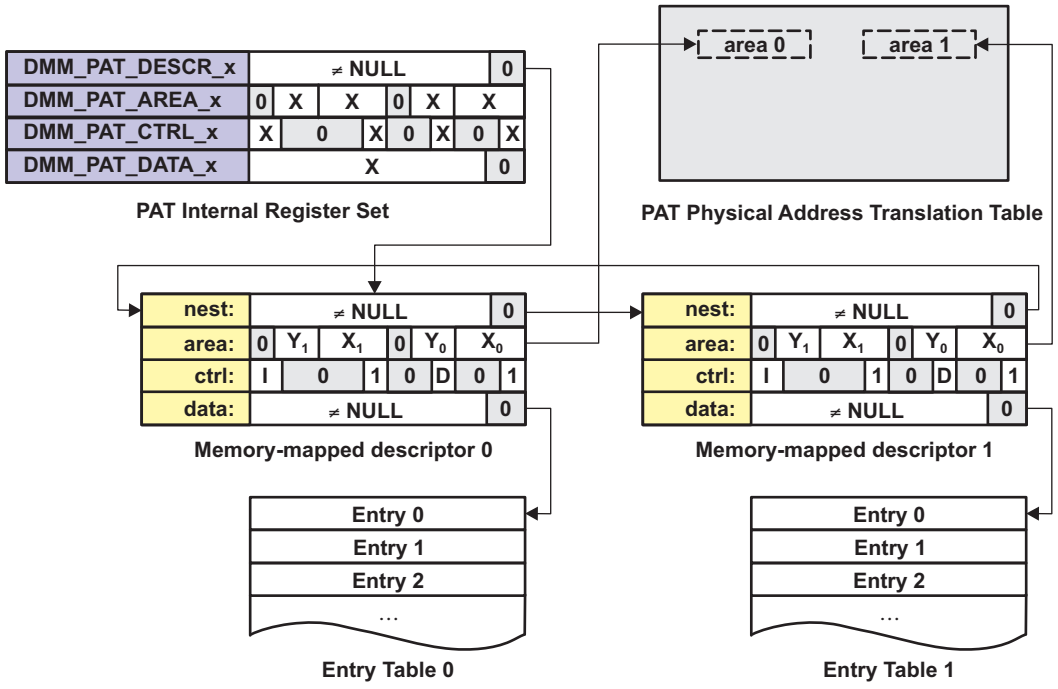


4.3.3.6 Cyclic Snchronised Auto-Configured Area Refill

These steps have to be performed to create one 16-byte aligned memory-mapped entry table per area containing the entries for the corresponding area, as seen in Figure 4-55.

1. Create one 16-byte aligned memory-mapped descriptor structures per area where:
   (a) the next field is set to the physical address of the next descriptor in the circular list
   (b) the area field is set with the relevant (x0, y0) (x1, y1) area definition
   (c) the ctrl field is set with the synchronising initiator identifier I, the SYNC bit asserted, the requested direction D, and the START bit asserted to start refilling as soon as the previous area refill is done and initiator I has made one access in the previous area
   (d) the data field is set to the physical address of the corresponding entry table
2. Write the DMM_PAT_DESCR_i register with the physical address of the initial descriptor.
3. Each area refill is done when the DMM_PAT_STATUS_i[3] DONE bit is set.
4. A new refill can be initiated by writing any value to the DMM_PAT_DESCR_i to abort the current one.

Figure 4-55. DMM Cyclic Synchronised Auto-configured Area Refill



4.3.4 Address Management Using LISA Sections

In the device, there are either one ot two memory controllers (EMIFs).

4.3.4.1 DMM Sections

The DMM section definition is certainly the very first step in the DMM software configuration, as it is about defining the system address map to be used when accessing the SDRAM controllers.

A DMM section description fits in a single 32-bit register (DMM_LISA_MAP_x), where:

- SYS_ADDR defines the base address of the decoding range for the section
- SYS_SIZE defines the size of the section. Note that the encoding of this parameter is the number of bits actually used in the upper 8-bits of the incoming system address.
- SDRC_ADDR defines the physical base address of the section in the external memory controller
- SDRC_ADDRSPC defines the address space used on the SDRAM controller when hitting this section. The address space feature is not supported.
- SDRC_MAP defines the target memory controllers for this section. A section may hit a one or two controllers. The section is not used if this parameter is set to zero.
- SDRC_INTL defines the granularity of the interleaving if the section is mapped on more than one memory controllers

A couple of examples are given in the next subsections.

4.3.4.1.1 Case 1: Two Memory Controllers, 2GB DDR Symmetrical Distribution

Use Case: In this example we assume 2GB of external memory, spread evenly between 2 SDRAM controllers.

Table 4-4. Case 1 Memory Controllers

System Address Range	Memory Controller	EMIF Address Range
8000 0000-FFFF FFFFh	1 and 2, interleaved at 256 Bytes	0000 0000h-7FFF FFFFh

Configuration:

Table 4-5. Controller Configuration

Bit	Field	Section 0
31-24	SYS_ADDR	80h (incoming System Address MSB)
22-20	SYS_SIZE	7h (2 GB)
19-18	SDRC_INTL	2h (256 bytes interleaving)
17-16	SDRC_ADDRSPC	0 (Unused Reserved field)
8	SDRC_MAP	3h (Map to both EMIF0 and EMIF1
7-0	SDRC_ADDR	0 (SDRC address MSB)

NOTE: Section 1, Section 2 and Section 3 - identical to Section 0.

How DMM decodes and translates:

1. To check if an address hits a section, use the eight upper address bits of the address and mask them with the hit mask (2 8 - 2SYS_SIZE). If the result is equal to SYS_ADDR, the section is hit.
2. To define the physical address to be issued to the memory controller, use the eight upper address of the system address, mask them with the address mask ($2^{SYS\_SIZE} – 1$), and then OR them with SDRC_ADDR. This will give the resulting eight upper physical address bits. All lower address bits are forwarded unchanged.
3. If interleaving enabled, the physical address generation is modified. In case of a 256 bytes interleaving, the first chunk of 256 bytes is mapped to the first controller, the second chunk to the second controller, and so on. This results in system address bit 8 to be decoded as the SDRAM controller ID, 0 for the first controller and 1 for the second one (system address bit 7 would be used for interleaving at 128 bytes boundary, and bit 9 for 512 bytes). This bit is not included in the computed physical address, meaning the upper system address bits 9 to 31 are shifted to bits 8 to 30 when generating the physical address.

Example with above configuration:

1. Incoming access's System address 99AE 37F0h:
2. Hit mask: $2^8 - 2^7 = 80h$
3. Masked upper address bits: 80h AND 80h = 80h (the SYS_ADDR) , thus hits section 0
4. Address mask: $2^7 - 1 = 7Fh$
5. Masked upper address bits: 99h AND 7Fh = 19h
6. Full masked address: 19AE 37F0h
7. Bit 8 is 1 -> targets the second memory controller
8. Full masked shifted address (suppressing bit 8): 0CD7 1BF0h
9. OR upper physical address bits with SDRC_ADDR: 0CD7 1BF0h
10. Physical address: 0CD7 1BF0h

This request will be forwarded to address 0CD7 1BF0h, of the second memory controller, that is, EMIF1.

4.3.4.1.2 Case 2: Two Memory Controllers, 1152 MB (1 GB + 128 MB) DDR Symmetrical Distribution

In this example we assume 2GB of external memory, spread evenly between two EMIF controllers.

Table 4-6. Case 2 Memory Controllers

System Address Range	SDRAM Controller	EMIF Address Range
8000 0000h-C7FF FFFFh	1 and 2, interleaved at 256 Bytes	0000 0000h-47FF FFFFh

Configuration:

Table 4-7. Controller Configuration

Bit	Field	Section 0	Section 1
31-24	SYS_ADDR	80h (incoming System Address MSB)	C0h (incoming System Address MSB)
22-20	SYS_SIZE	6h (1 GB)	3h (128 MB)
19-18	SDRC_INTL	2h (256 bytes interleaving)	1 (128 bytes interleaving)
17-16	SDRC_ADDRSPC	0 (Unused Reserved field)	0 (Unused Reserved field)
8	SDRC_MAP	3h (Map to both EMIF0 and EMIF1)	3h (Map to both EMIF0 and EMIF1)
7-0	SDRC_ADDR	0 (SDRC address MSB)	20h (SDRC address MSB)

NOTE: Section 2 and Section 3 are identical to Section 1.

How DMM decodes and translates:

Identical to Section 4.3.4.1.1.

See Figure 4-56 for System Address Memory Range and its mapping to EMIFs.

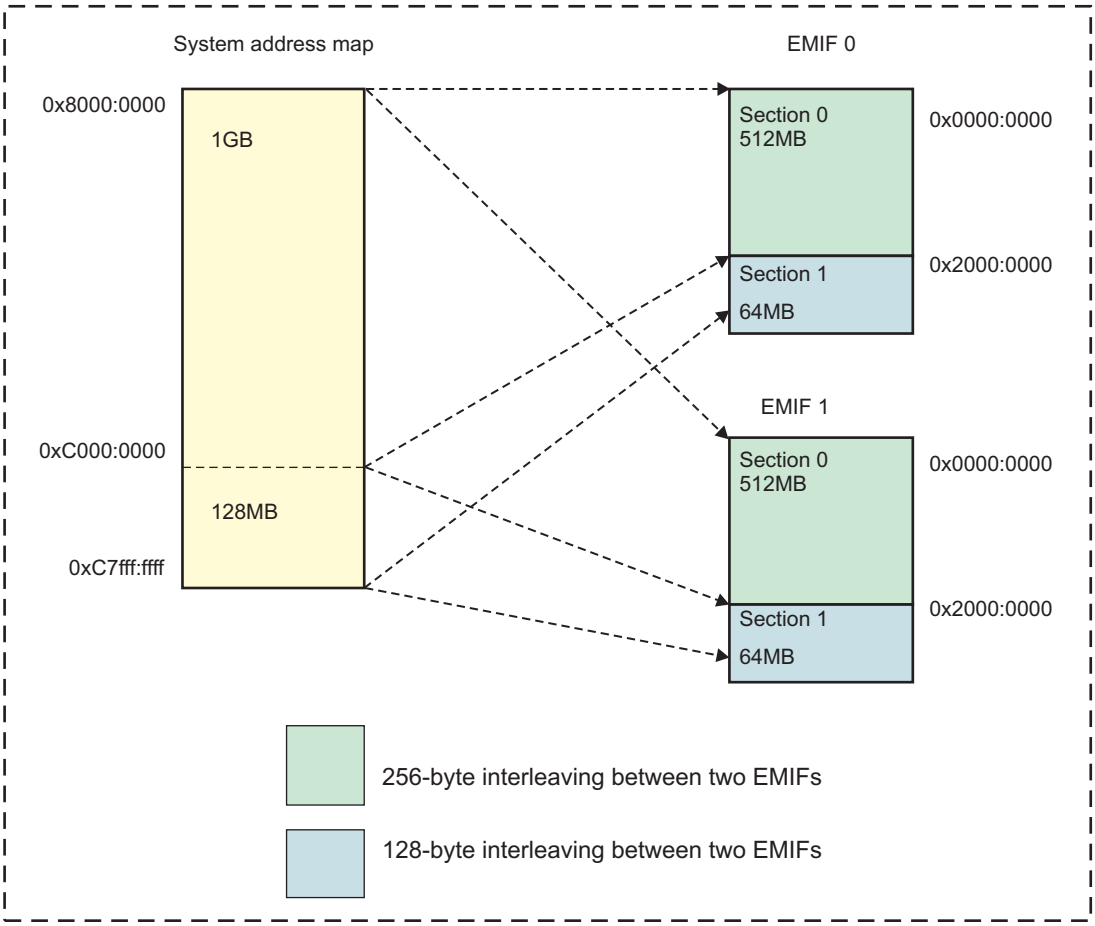
Example with above configuration:

Incoming accesses System address C022 34C0h:

1. Upper address bits: C0h
2. Hit mask: $2^8 - 2^3$ = F8h
3. Masked upper address bits: C0h AND F8h = C0h (the SYS_ADDR) , thus hits section 1
4. Address mask: $2^3 - 1$ = 7h
5. Masked upper address bits: C0h AND 7h = 0
6. Full masked address: 0022 34C0h
7. Bit 7 is 1 -> targets the second memory controller
8. Full masked shifted address (suppressing bit 7): 0011 1A60h
9. OR upper physical address bits with SDRC_ADDR: 0011 1A60h
10. Physical address: 0011 1A60h

This request will be forwarded to address 0011 1A60h, of the second memory controller; that is, EMIF1.

Figure 4-56. DMM Section Use-Case 2



4.3.4.1.3 Case 3: Two SDRAM Controllers, 1152 MB (1 GB + 128 MB) DDR Asymmetrical Distribution

NOTE: For optimal system performance, symmetric configuration is HIGHLY recommended. Unless dictated by system cost and other constraints, Asymmetrical distribution of memory should not be considered.

There are many ways to configure the section mapping in this use case.

Option 1: Asymmetrical section of EMIF0 is mapped to system address C000 0000h-C7FF FFFFh, higher portion of the SDRAM in system memory map.

Table 4-8. Section Mapping Option 1

Bit	Field	Section 0	Section 1
31-24	SYS_ADDR	80h (incoming System Address MSB)	88h (incoming System Address MSB)
22-20	SYS_SIZE	3h (128 MB)	6h (1 GB)
19-18	SDRC_INTL	0 (No interleaving)	2h (256 bytes interleaving)
17-16	SDRC_ADDRSPC	0 (Unused Reserved field)	0 (Unused Reserved field)
8	SDRC_MAP	1 (Map to EMIF0)	3h (Map to both EMIF0 and EMIF1)
7-0	SDRC_ADDR	20h (SDRC address MSB)	0 (SDRC address MSB)

Option 2: Asymmetrical section of EMIF0 is mapped to system address 8000 0000h-87FF FFFFh, Lower portion of the SDRAM in system memory map.

Table 4-9. Section Mapping Option 2

Bit	Field	Section 0	Section 1
31-24	SYS_ADDR	80h (incoming System Address MSB)	C0h (incoming System Address MSB)
22-20	SYS_SIZE	6h (1 GB)	3h (128 MB)
19-18	SDRC_INTL	2h (256 bytes interleaving)	0 (No interleaving)
17-16	SDRC_ADDRSPC	0 (Unused Reserved field)	0 (Unused Reserved field)
8	SDRC_MAP	3h (Map to both EMIF0 and EMIF1)	1 (Map to EMIF0)
7-0	SDRC_ADDR	0 (SDRC address MSB)	20h (SDRC address MSB)

4.4 DMM/TILER Registers

Table 4-10 lists the DMM/TILER registers. For the base address of these registers, see Table 1-11.

Table 4-10. DMM/TILER Registers

Address Offset	Register Name	Section
0	DMM_REVISION	Section 4.4.1
10h	DMM_SYSCONFIG	Section 4.4.2
1Ch	DMM_LISA_LOCK	Section 4.4.3
40h-4Ch	DMM_LISA_MAP_0-3	Section 4.4.4
220h-224h	DMM_TILER_OR_0-1	Section 4.4.5
410h	DMM_PAT_CONFIG	Section 4.4.6
420h-424h	DMM_PAT_VIEW_0-3	Section 4.4.7
440h-44Ch	DMM_PAT_VIEW_MAP_0-3	Section 4.4.8
460h	DMM_PAT_VIEW_MAP_BASE	Section 4.4.9
478h	DMM_PAT_IRQ_EOI	Section 4.4.10
480h	DMM_PAT_IRQSTATUS_RAW	Section 4.4.11
490h	DMM_PAT_IRQSTATUS	Section 4.4.12
4A0h	DMM_PAT_IRQENABLE_SET	Section 4.4.13
4B0h	DMM_PAT_IRQENABLE_CLR	Section 4.4.14
4C0h-4CCh	DMM_PAT_STATUS_0-3	Section 4.4.15
500h-530h	DMM_PAT_DESCR_0-3	Section 4.4.16
504h-534h	DMM_PAT_AREA_0-3	Section 4.4.17
508h-538h	DMM_PAT_CTRL_0-3	Section 4.4.18
50Ch-53Ch	DMM_PAT_DATA_0-3	Section 4.4.19
620h-624h	DMM_PEG_PRIO_0-1	Section 4.4.20
640h	DMM_PEG_PRIO_PAT	Section 4.4.21

4.4.1 DMM Revision Register: DMM_REVISION

The DMM Revision register is shown in Figure 4-57 and described in Table 4-11.

Figure 4-57. DMM_REVISION Register

31	30	29	28	27	16
SCHEME		Reserved		FUNC	
R-1		R-0		R-0	

15	11	10	8	7	6	5	0
RRTL		XMAJOR		CUSTOM		YMINOR	
R-0		R-0		R-0		R-8h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-11. DMM_REVISION Register Field Descriptions

Bit	Field	Value	Description	Type
31-30	SCHEME	1	Used to distinguish between old scheme and current.	R
29-28	Reserved	0	Reserved	R
27-16	FUNC	0	Software compatability level	R
15-11	RRTL	0	RTL Version (R)	R
10-8	XMAJOR	0	Major Revision (X)	R
7-6	CUSTOM	0	Special DMM version	R
5-0	YMINOR	8h	Minor Revision (Y)	R

4.4.2 DMM Clock Management Configuration: DMM_SYSCONFIG

The DMM Clock Management Configuration register is shown in Figure 4-58 and described in Table 4-12.

Figure 4-58. DMM_SYSCONFIG Register

31	4	3	2	1	0
Reserved		IDLEMODE		Rsvd	
R-0		R/W-2h		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-12. DMM_SYSCONFIG Register Field Descriptions

Bit	Field	Value	Description	Type
31-4	Reserved	0	Reserved	R
3-2	IDLEMODE		Configuration of the local target state management mode.	R/W
		0	Force-idle mode. Backup mode, for debug only.	
		1h	No idle mode. Backup mode, for debug only.	
		2h	Smart-idle mode: local target's idle state eventually follows systems IDLE request.	
		3h	Reserved	
1-0	Reserved	0	Reserved	R

4.4.3 LISA Configuration Locking Register: DMM_LISA_LOCK

The LISA Configuration Locking register is shown in Figure 4-59 and described in Table 4-13.

Figure 4-59. DMM_LISA_LOCK Register

31	1	0
Reserved		LOCK
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-13. DMM_LISA_LOCK Register Field Descriptions

Bit	Field	Value	Description	Type
31-1	Reserved	0	Reserved	R
0	LOCK		DMM lock map	R/W
		R0	DMM_LISA_MAP__x un-locked	
		W0	No effect (clear on reset only)	
		R1	DMM_LISA_MAP__xlocked	
		W1	Locking DMM_LISA_MAP__x registers	

4.4.4 DMM LISA MAP Registers: DMM_LISA_MAP_0-DMM_LISA_MAP_3

The DMM LISA MAP register is shown in Figure 4-60 and described in Table 4-14.

Figure 4-60. DMM_LISA_MAP Registers

31	24	23	22	20	19	18	17	16
SYS_ADDR		Rsvd	SYS_SIZE		SDRC_INTL		SDRC_ADDRSPC	
R/W-0		R-0	R/W-0		R/W-0		R/W-0	

15	10	9	8	7	0
Reserved		SDRC_MAP		SDRC_ADDR	
R-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-14. DMM_LISA_MAP Registers Field Descriptions

Bit	Field	Value	Description	Type
31-24	SYS_ADDR	0	DMM system section address MSB	R/W
23	Reserved	0	Reserved	R
22-20	SYS_SIZE		DMM system section size	R/W
		0h	16-MB section	
		1h	32-MB section	
		2h	64-MB section	
		3h	128-MB section	
		4h	256-MB section	
		5h	512-MB section	
		6h	1-GB section	
		7h	2-GB section	
19-18	SDRC_INTL		SDRAM controller interleaving mode	R/W
		0h	No interleaving	
		1h	128-byte interleaving	
		2h	256-byte interleaving	
		3h	512-byte interleaving	
17-16	SDRC_ADDRSPC	0h	Reserved. Should be 0.	R/W
15-10	Reserved	0	Reserved	R
9-8	SDRC_MAP		SDRAM controller mapping	R/W
		0h	Un-mapped	
		1h	Mapped on SDRC 0 only (not interleaved)	
		2h	Mapped on SDRC 1 only (not interleaved)	
		3h	Mapped on SDRC 0 and SDRC 1 (interleaved)	
7-0	SDRC_ADDR	0h	SDRAM controller address MSB	R/W

4.4.5 DMM TILER Orientation Registers: DMM_TILER_OR0-DMM_TILER_OR1

The DMM TILER Orientation register is shown in Figure 4-61 and described in Table 4-15.

Figure 4-61. DMM_TILER_OR Registers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W7	OR7			W6	OR6			W5	OR5			W4	OR4		
R/W-0	R/W-0			R/W-0	R/W-0			R/W-0	R/W-0			R/W-0	R/W-0		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W3	OR3			W2	OR2			W1	OR1			W0	OR0		
R/W-0	R/W-0			R/W-0	R/W-0			R/W-0	R/W-0			R/W-0	R/W-0		

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-15. DMM_TILER_OR Registers Field Descriptions

Bit	Field	Value	Description	Type
31	W7	0	Write-enable for OR7 bit-field; OR7 field is unchanged	R/W
30-28	OR7	0	Orientation for initiator 8.n+7	R/W
27	W6	0	Write-enable for OR6 bit-field; OR6 field is unchanged	R/W
26-24	OR6	0	Orientation for initiator 8.n+6	R/W
23	W5	0	Write-enable for OR5 bit-field; OR5 field is unchanged	R/W
22-20	OR5	0	Orientation for initiator 8.n+5	R/W
19	W4	0	Write-enable for OR4 bit-field; OR4 field is unchanged	R/W
18-16	OR4	0	Orientation for initiator 8.n+4	R/W
15	W3	0	Write-enable for OR3 bit-field; OR3 field is unchanged	R/W
14-12	OR3	0	Orientation for initiator 8.n+3	R/W
11	W2	0	Write-enable for OR2 bit-field; OR2 field is unchanged	R/W
10-8	OR2	0	Orientation for initiator 8.n+2	R/W
7	W1	0	Write-enable for OR1 bit-field; OR1 field is unchanged	R/W
6-4	OR1	0	Orientation for initiator 8.n+1	R/W
3	W0	0	Write-enable for OR0 bit-field; OR0 field is unchanged	R/W
2-0	OR0	0	Orientation for initiator 8.n+0	R/W

4.4.6 DMM PAT Configuration Register: DMM_PAT_CONFIG

The DMM PAT Configuration register is shown in Figure 4-62 and described in Table 4-16.

Figure 4-62. DMM_PAT_CONFIG Register

31	4	3	2	1	0
Reserved		MODE3	MODE2	MODE1	MODE0
R-0		R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-16. DMM_PAT_CONFIG Register Field Descriptions

Bit	Field	Value	Description	Type
31-4	Reserved	0	Reserved	R
3	MODE3	0	Mode of Refill Engine 3 0 : Normal Mode 1h : Direct LUT access	R/W
2	MODE2	0	Mode of Refill Engine 1 0 : Normal Mode 1h : Direct LUT access	R/W
1	MODE1	0	Mode of Refill Engine 2 0 : Normal Mode 1h : Direct LUT access	R/W
0	MODE0	0	Mode of Refill Engine 0 0 : Normal Mode 1h : Direct LUT access	R/W

4.4.7 DMM PAT View Registers: DMM_PAT_VIEW_0-DMM_PAT_VIEW_2

The DMM PAT View register is shown in Figure 4-63 and described in Table 4-17.

Figure 4-63. DMM_PAT_VIEW Registers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W7	Rsvd	V7		W6	Rsvd	V6		W5	Rsvd	V5		W4	Rsvd	V4	
R/W-0	R-0	R/W-0		R/W-0	R-0	R/W-0		R/W-0	R-0	R/W-0		R/W-0	R-0	R/W-0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W3	Rsvd	V3		W2	Rsvd	V2		W1	Rsvd	V1		W0	Rsvd	V0	
R/W-0	R-0	R/W-0		R/W-0	R-0	R/W-0		R/W-0	R-0	R/W-0		R/W-0	R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-17. DMM_PAT_VIEW Registers Field Descriptions

Bit	Field	Value	Description	Type
31	W7	0	Write-enable for V7 bit-field; V7 field is unchanged	R/W
30	Reserved	0	Reserved	R
29-28	V7	0	PAT view for initiator 8.n+7	R/W
27	W6	0	Write-enable for V6 bit-field; V6 field is unchanged	R/W
26	Reserved	0	Reserved	R
25-24	V6	0	PAT view for initiator 8.n+6	R/W
23	W5	0	Write-enable for V5 bit-field; V5 field is unchanged	R/W
22	Reserved	0	Reserved	R
21-20	V5	0	PAT view for initiator 8.n+5	R/W
19	W4	0	Write-enable for V4 bit-field; V4 field is unchanged	R/W
18	Reserved	0	Reserved	R
17-16	V4	0	PAT view for initiator 8.n+4	R/W
15	W3	0	Write-enable for V3 bit-field; V3 field is unchanged	R/W
14	Reserved	0	Reserved	R
13-12	V3	0	PAT view for initiator 8.n+3	R/W
11	W2	0	Write-enable for V2 bit-field; V2 field is unchanged	R/W
10	Reserved	0	Reserved	R
9-8	V2	0	PAT view for initiator 8.n+2	R/W
7	W1	0	Write-enable for V1 bit-field; V1 field is unchanged	R/W
6	Reserved	0	Reserved	R
5-4	V1	0	PAT view for initiator 8.n+1	R/W
3	W0	0	Write-enable for V0 bit-field; V0 field is unchanged	R/W
2	Reserved	0	Reserved	R
1-0	V0	0	PAT view for initiator 8.n+0	R/W

4.4.8 DMM View Map Registers: DMM_PAT_VIEW_MAP_0-DMM_PAT_VIEW_MAP_4

The DMM View Map register is shown in Figure 4-64 and described in Table 4-18.

Figure 4-64. DMM_PAT_VIEW_MAP Registers

31	30 28	27 24	23	22 20	19 16
ACCESS_PAGE	Reserved	CONT_PAGE	ACCESS_32	Reserved	CONT_32
R/W-0	R-0	R/W-0	R/W-0	R-0	R/W-0

15	14 12	11 8	7	6 4	3 0
ACCESS_16	Reserved	CONT_16	ACCESS_8	Reserved	CONT_8
R/W-0	R-0	R/W-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-18. DMM_PAT_VIEW_MAP Registers Field Descriptions

Bit	Field	Value	Description	Type
31	ACCESS_PAGE		Kind of access for this page mode container	R/W
		0	Direct access, container base address given in CONT_PAGE	
		1	indirect access through the LUT indexed by CONT_PAGE	
30-28	Reserved	0	Reserved	R
27-24	CONT_PAGE	0	Base address of Container for page mode -or - LUT index	R/W
23	ACCESS_32		Kind of access for this 32-bit mode container	R/W
		0	Direct access, container base address given in CONT_32	
		1	indirect access through the LUT indexed by CONT_32	
22-20	Reserved	0	Reserved	R
19-16	CONT_32	0	Base address of Container for 32-bit mode -or - LUT index	R/W
15	ACCESS_16	0	Kind of access for this 16-bit mode container	R/W
		0	Direct access, container base address given in CONT_16	
		1	indirect access through the LUT indexed by CONT_16	
14-12	Reserved	0	Reserved	R
11-8	CONT_16	0	Base address of Container for 16-bit mode -or - LUT index	R/W
7	ACCESS_8		Kind of access for this 8-bit mode container	R/W
		0	Direct access, container base address given in CONT_8	
		1h	indirect access through the LUT indexed by CONT_8	
6-4	Reserved	0	Reserved	R
3-0	CONT_8	0	Base address of Container for 8-bit mode -or - LUT index	R/W

4.4.9 DMM PAT View Mapping Base Address Register: DMM_PAT_VIEW_MAP_BASE

The DMM PAT View Mapping Base Address register is shown in Figure 4-65 and described in Table 4-19.

Figure 4-65. DMM_PAT_VIEW_MAP_BASE Register

31	30	0
BASEADDR	Reserved	
R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-19. DMM_PAT_VIEW_MAP_BASE Register Field Descriptions

Bit	Field	Value	Description	Type
31	BASEADDR	0	MSB of the PAT view mapping base address	R/W
30-0	Reserved	0	Reserved	R

4.4.10 DMM PAT End of Interrupt Register: DMM_PAT_IRQ_EOI

The DMM PAT End of Interrupt register is shown in Figure 4-66 and described in Table 4-20. This register is per-event raw interrupt status vector. The purpose is mostly for debug. Raw status is set even if the related event is not enabled. Write 1 to set the (raw) status.

Figure 4-66. DMM_PAT_IRQ_EOI Register

31	30	29	28	27	26	25	24
ERR_LUT_MISS3	ERR_UPD_DATA3	ERR_UPD_CTRL3	ERR_UPD_AREA3	ERR_INV_DATA3	ERR_INV_DSC3	FILL_LST3	FILL_DSC3
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0
23	22	21	20	19	18	17	16
ERR_LUT_MISS2	ERR_UPD_DATA2	ERR_UPD_CTRL2	ERR_UPD_AREA2	ERR_INV_DATA2	ERR_INV_DSC2	FILL_LST2	FILL_DSC2
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0
15	14	13	12	11	10	9	8
ERR_LUT_MISS1	ERR_UPD_DATA1	ERR_UPD_CTRL1	ERR_UPD_AREA1	ERR_INV_DATA1	ERR_INV_DSC1	FILL_LST1	FILL_DSC1
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0
7	6	5	4	3	2	1	0
ERR_LUT_MISS0	ERR_UPD_DATA0	ERR_UPD_CTRL0	ERR_UPD_AREA0	ERR_INV_DATA0	ERR_INV_DSC0	FILL_LST0	FILL_DSC0
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-20. DMM_PAT_IRQ_EOI Register Field Descriptions

Bit	Field	Value	Description	Type
31	ERR_LUT_MISS3	0	Unexpected Access to a yet-to-be-refilled area event in area 3	R/W1S
30	ERR_UPD_DATA3	0	Data register update whilst refilling error event in area 3	R/W1S
29	ERR_UPD_CTRL3	0	Control register update whilst refilling error event in area 3	R/W1S
28	ERR_UPD_AREA3	0	Area register update whilst refilling error event in area 3	R/W1S
27	ERR_INV_DATA3	0	Invalid entry-table pointer error event in area 3	R/W1S
26	ERR_INV_DSC3	0	Invalid descriptor pointer error event in area 3	R/W1S
25	FILL_LST3	0	End of refill event for the last descriptor in area 3	R/W1S
24	FILL_DSC3	0	End of refill event for any descriptor in area 3	R/W1S
23	ERR_LUT_MISS2	0	Unexpected Access to a yet-to-be-refilled area event in area 2	R/W1S

Table 4-20. DMM_PAT_IRQ_EOI Register Field Descriptions (continued)

Bit	Field	Value	Description	Type
22	ERR_UPD_DATA2	0	Data register update whilst refilling error event in area 2	R/W1S
21	ERR_UPD_CTRL2	0	Control register update whilst refilling error event in area 2	R/W1S
20	ERR_UPD_AREA2	0	Area register update whilst refilling error event in area 2	R/W1S
19	ERR_INV_DATA2	0	Invalid entry-table pointer error event in area 2	R/W1S
18	ERR_INV_DSC2	0	Invalid descriptor pointer error event in area 2	R/W1S
17	FILL_LST2	0	End of refill event for the last descriptor in area 2	R/W1S
16	FILL_DSC2	0	End of refill event for any descriptor in area 2	R/W1S
15	ERR_LUT_MISS1	0	Unexpected Access to a yet-to-be-refilled area event in area 1	R/W1S
14	ERR_UPD_DATA1	0	Data register update whilst refilling error event in area 1	R/W1S
13	ERR_UPD_CTRL1	0	Control register update whilst refilling error event in area 1	R/W1S
12	ERR_UPD_AREA1	0	Area register update whilst refilling error event in area 1	R/W1S
11	ERR_INV_DATA1	0	Invalid entry-table pointer error event in area 1	R/W1S
10	ERR_INV_DSC1	0	Invalid descriptor pointer error event in area 1	R/W1S
9	FILL_LST1	0	End of refill event for the last descriptor in area 1	R/W1S
8	FILL_DSC1	0	End of refill event for any descriptor in area 1	R/W1S
7	ERR_LUT_MISS0	0	Unexpected Access to a yet-to-be-refilled area event in area 0	R/W1S
6	ERR_UPD_DATA0	0	Data register update whilst refilling error event in area 0	R/W1S
5	ERR_UPD_CTRL0	0	Control register update whilst refilling error event in area 0	R/W1S
4	ERR_UPD_AREA0	0	Area register update whilst refilling error event in area 0	R/W1S
3	ERR_INV_DATA0	0	Invalid entry-table pointer error event in area 0	R/W1S
2	ERR_INV_DSC0	0	Invalid descriptor pointer error event in area 0	R/W1S
1	FILL_LST0	0	End of refill event for the last descriptor in area 0	R/W1S
0	FILL_DSC0	0	End of refill event for any descriptor in area 0	R/W1S

4.4.11 DMM PAT Raw Interrupt Status Register: DMM_PAT_IRQSTATUS_RAW

The DMM PAT Raw Interrupt Status Register is shown in Figure 4-67 and described in Table 4-21. Per-event raw interrupt status vector. Raw status is set even if the related event is not enabled. Write 1 to set the (raw) status, mostly for debug.

Figure 4-67. DMM_PAT_IRQSTATUS_RAW Register

31	30	29	28	27	26	25	24
ERR_LUT_MISS3	ERR_UPD_DATA3	ERR_UPD_CTRL3	ERR_UPD_AREA3	ERR_INV_DATA3	ERR_INV_DSC3	FILL_LST3	FILL_DSC3
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0
23	22	21	20	19	18	17	16
ERR_LUT_MISS2	ERR_UPD_DATA2	ERR_UPD_CTRL2	ERR_UPD_AREA2	ERR_INV_DATA2	ERR_INV_DSC2	FILL_LST2	FILL_DSC2
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0
15	14	13	12	11	10	9	8
ERR_LUT_MISS1	ERR_UPD_DATA1	ERR_UPD_CTRL1	ERR_UPD_AREA1	ERR_INV_DATA1	ERR_INV_DSC1	FILL_LST1	FILL_DSC1
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0
7	6	5	4	3	2	1	0
ERR_LUT_MISS0	ERR_UPD_DATA0	ERR_UPD_CTRL0	ERR_UPD_AREA0	ERR_INV_DATA0	ERR_INV_DSC0	FILL_LST0	FILL_DSC0
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-21. DMM_PAT_IRQSTATUS_RAW Register Field Descriptions

Bit	Field	Value	Description	Type
31	ERR_LUT_MISS3	0	Unexpected Access to a yet-to-be-refilled area event in area 3	R/W1S
30	ERR_UPD_DATA3	0	Data register update whilst refilling error event in area 3	R/W1S
29	ERR_UPD_CTRL3	0	Control register update whilst refilling error event in area 3	R/W1S
28	ERR_UPD_AREA3	0	Area register update whilst refilling error event in area 3	R/W1S
27	ERR_INV_DATA3	0	Invalid entry-table pointer error event in area 3	R/W1S
26	ERR_INV_DSC3	0	Invalid descriptor pointer error event in area 3	R/W1S
25	FILL_LST3	0	End of refill event for the last descriptor in area 3	R/W1S
24	FILL_DSC3	0	End of refill event for any descriptor in area 3	R/W1S
23	ERR_LUT_MISS2	0	Unexpected Access to a yet-to-be-refilled area event in area 2	R/W1S
22	ERR_UPD_DATA2	0	Data register update whilst refilling error event in area 2	R/W1S
21	ERR_UPD_CTRL2	0	Control register update whilst refilling error event in area 2	R/W1S
20	ERR_UPD_AREA2	0	Area register update whilst refilling error event in area 2	R/W1S
19	ERR_INV_DATA2	0	Invalid entry-table pointer error event in area 2	R/W1S
18	ERR_INV_DSC2	0	Invalid descriptor pointer error event in area 2	R/W1S
17	FILL_LST2	0	End of refill event for the last descriptor in area 2	R/W1S
16	FILL_DSC2	0	End of refill event for any descriptor in area 2	R/W1S
15	ERR_LUT_MISS1	0	Unexpected Access to a yet-to-be-refilled area event in area 1	R/W1S
14	ERR_UPD_DATA1	0	Data register update whilst refilling error event in area 1	R/W1S
13	ERR_UPD_CTRL1	0	Control register update whilst refilling error event in area 1	R/W1S
12	ERR_UPD_AREA1	0	Area register update whilst refilling error event in area 1	R/W1S
11	ERR_INV_DATA1	0	Invalid entry-table pointer error event in area 1	R/W1S
10	ERR_INV_DSC1	0	Invalid descriptor pointer error event in area 1	R/W1S
9	FILL_LST1	0	End of refill event for the last descriptor in area 1	R/W1S
8	FILL_DSC1	0	End of refill event for any descriptor in area 1	R/W1S
7	ERR_LUT_MISS0	0	Unexpected Access to a yet-to-be-refilled area event in area 0	R/W1S

Table 4-21. DMM_PAT_IRQSTATUS_RAW Register Field Descriptions (continued)

Bit	Field	Value	Description	Type
6	ERR_UPD_DATA0	0	Data register update whilst refilling error event in area 0	R/W1S
5	ERR_UPD_CTRL0	0	Control register update whilst refilling error event in area 0	R/W1S
4	ERR_UPD_AREA0	0	Area register update whilst refilling error event in area 0	R/W1S
3	ERR_INV_DATA0	0	Invalid entry-table pointer error event in area 0	R/W1S
2	ERR_INV_DSC0	0	Invalid descriptor pointer error event in area 0	R/W1S
1	FILL_LST0	0	End of refill event for the last descriptor in area 0	R/W1S
0	FILL_DSC0	0	End of refill event for any descriptor in area 0	R/W1S

4.4.12 DMM PAT Interrupt Status Register: DMM_PAT_IRQSTATUS

The DMM PAT Interrupt Status register is shown in Figure 4-68 and described in Table 4-22. Per-event "enabled" interrupt status vector. Error status is not set unless the event is enabled.

Figure 4-68. DMM_PAT_IRQSTATUS Register

31	30	29	28	27	26	25	24
ERR_LUT_MISS3	ERR_UPD_DATA3	ERR_UPD_CTRL3	ERR_UPD_AREA3	ERR_INV_DATA3	ERR_INV_DSC3	FILL_LST3	FILL_DSC3
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0
23	22	21	20	19	18	17	16
ERR_LUT_MISS2	ERR_UPD_DATA2	ERR_UPD_CTRL2	ERR_UPD_AREA2	ERR_INV_DATA2	ERR_INV_DSC2	FILL_LST2	FILL_DSC2
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0
15	14	13	12	11	10	9	8
ERR_LUT_MISS1	ERR_UPD_DATA1	ERR_UPD_CTRL1	ERR_UPD_AREA1	ERR_INV_DATA1	ERR_INV_DSC1	FILL_LST1	FILL_DSC1
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0
7	6	5	4	3	2	1	0
ERR_LUT_MISS0	ERR_UPD_DATA0	ERR_UPD_CTRL0	ERR_UPD_AREA0	ERR_INV_DATA0	ERR_INV_DSC0	FILL_LST0	FILL_DSC0
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-22. DMM_PAT_IRQSTATUS Register Field Descriptions

Bit	Field	Value	Description	Type
31	ERR_LUT_MISS3	0	Unexpected Access to a yet-to-be-refilled area event in area 3	R/W1C
30	ERR_UPD_DATA3	0	Data register update whilst refilling error event in area 3	R/W1C
29	ERR_UPD_CTRL3	0	Control register update whilst refilling error event in area 3	R/W1C
28	ERR_UPD_AREA3	0	Area register update whilst refilling error event in area 3	R/W1C
27	ERR_INV_DATA3	0	Invalid entry-table pointer error event in area 3	R/W1C
26	ERR_INV_DSC3	0	Invalid descriptor pointer error event in area 3	R/W1C
25	FILL_LST3	0	End of refill event for the last descriptor in area 3	R/W1C
24	FILL_DSC3	0	End of refill event for any descriptor in area 3	R/W1C
23	ERR_LUT_MISS2	0	Unexpected Access to a yet-to-be-refilled area event in area 2	R/W1C
22	ERR_UPD_DATA2	0	Data register update whilst refilling error event in area 2	R/W1C
21	ERR_UPD_CTRL2	0	Control register update whilst refilling error event in area 2	R/W1C
20	ERR_UPD_AREA2	0	Area register update whilst refilling error event in area 2	R/W1C
19	ERR_INV_DATA2	0	Invalid entry-table pointer error event in area 2	R/W1C
18	ERR_INV_DSC2	0	Invalid descriptor pointer error event in area 2	R/W1C
17	FILL_LST2	0	End of refill event for the last descriptor in area 2	R/W1C
16	FILL_DSC2	0	End of refill event for any descriptor in area 2	R/W1C
15	ERR_LUT_MISS1	0	Unexpected Access to a yet-to-be-refilled area event in area 1	R/W1C
14	ERR_UPD_DATA1	0	Data register update whilst refilling error event in area 1	R/W1C
13	ERR_UPD_CTRL1	0	Control register update whilst refilling error event in area 1	R/W1C
12	ERR_UPD_AREA1	0	Area register update whilst refilling error event in area 1	R/W1C
11	ERR_INV_DATA1	0	Invalid entry-table pointer error event in area 1	R/W1C
10	ERR_INV_DSC1	0	Invalid descriptor pointer error event in area 1	R/W1C
9	FILL_LST1	0	End of refill event for the last descriptor in area 1	R/W1C
8	FILL_DSC1	0	End of refill event for any descriptor in area 1	R/W1C
7	ERR_LUT_MISS0	0	Unexpected Access to a yet-to-be-refilled area event in area 0	R/W1C
6	ERR_UPD_DATA0	0	Data register update whilst refilling error event in area 0	R/W1C

Table 4-22. DMM_PAT_IRQSTATUS Register Field Descriptions (continued)

Bit	Field	Value	Description	Type
5	ERR_UPD_CTRL0	0	Control register update whilst refilling error event in area 0	R/W1C
4	ERR_UPD_AREA0	0	Area register update whilst refilling error event in area 0	R/W1C
3	ERR_INV_DATA0	0	Invalid entry-table pointer error event in area 0	R/W1C
2	ERR_INV_DSC0	0	Invalid descriptor pointer error event in area 0	R/W1C
1	FILL_LST0	0	End of refill event for the last descriptor in area 0	R/W1C
0	FILL_DSC0	0	End of refill event for any descriptor in area 0	R/W1C

4.4.13 DMM PAT Interrupt Enable Register: DMM_PAT_IRQENABLE_SET

The DMM PAT Interrupt Enable register is shown in Figure 4-69 and described in Table 4-23. Per-event interrupt enable bit vector. Write 1 to set (enable interrupt).

Figure 4-69. DMM_PAT_IRQENABLE_SET Register

31	30	29	28	27	26	25	24
ERR_LUT_MISS3	ERR_UPD_DATA3	ERR_UPD_CTRL3	ERR_UPD_AREA3	ERR_INV_DATA3	ERR_INV_DSC3	FILL_LST3	FILL_DSC3
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0
23	22	21	20	19	18	17	16
ERR_LUT_MISS2	ERR_UPD_DATA2	ERR_UPD_CTRL2	ERR_UPD_AREA2	ERR_INV_DATA2	ERR_INV_DSC2	FILL_LST2	FILL_DSC2
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0
15	14	13	12	11	10	9	8
ERR_LUT_MISS1	ERR_UPD_DATA1	ERR_UPD_CTRL1	ERR_UPD_AREA1	ERR_INV_DATA1	ERR_INV DSC1	FILL_LST1	FILL_DSC1
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0
7	6	5	4	3	2	1	0
ERR_LUT_MISS0	ERR_UPD_DATA0	ERR_UPD_CTRL0	ERR_UPD_AREA0	ERR_INV_DATA0	ERR_INV_DSC0	FILL_LST0	FILL_DSC0
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-23. DMM_PAT_IRQENABLE_SET Register Field Descriptions

Bit	Field	Value	Description	Type
31	ERR_LUT_MISS3	0	Unexpected Access to a yet-to-be-refilled area event in area 3	R/W1S
30	ERR_UPD_DATA3	0	Data register update whilst refilling error event in area 3	R/W1S
29	ERR_UPD_CTRL3	0	Control register update whilst refilling error event in area 3	R/W1S
28	ERR_UPD_AREA3	0	Area register update whilst refilling error event in area 3	R/W1S
27	ERR_INV_DATA3	0	Invalid entry-table pointer error event in area 3	R/W1S
26	ERR_INV_DSC3	0	Invalid descriptor pointer error event in area 3	R/W1S
25	FILL_LST3	0	End of refill event for the last descriptor in area 3	R/W1S
24	FILL_DSC3	0	End of refill event for any descriptor in area 3	R/W1S
23	ERR_LUT_MISS2	0	Unexpected Access to a yet-to-be-refilled area event in area 2	R/W1S
22	ERR_UPD_DATA2	0	Data register update whilst refilling error event in area 2	R/W1S
21	ERR_UPD_CTRL2	0	Control register update whilst refilling error event in area 2	R/W1S
20	ERR_UPD_AREA2	0	Area register update whilst refilling error event in area 2	R/W1S
19	ERR_INV_DATA2	0	Invalid entry-table pointer error event in area 2	R/W1S
18	ERR_INV_DSC2	0	Invalid descriptor pointer error event in area 2	R/W1S
17	FILL_LST2	0	End of refill event for the last descriptor in area 2	R/W1S
16	FILL_DSC2	0	End of refill event for any descriptor in area 2	R/W1S
15	ERR_LUT_MISS1	0	Unexpected Access to a yet-to-be-refilled area event in area 1	R/W1S
14	ERR_UPD_DATA1	0	Data register update whilst refilling error event in area 1	R/W1S
13	ERR_UPD_CTRL1	0	Control register update whilst refilling error event in area 1	R/W1S
12	ERR_UPD_AREA1	0	Area register update whilst refilling error event in area 1	R/W1S
11	ERR_INV_DATA1	0	Invalid entry-table pointer error event in area 1	R/W1S
10	ERR_INV_DSC1	0	Invalid descriptor pointer error event in area 1	R/W1S
9	FILL_LST1	0	End of refill event for the last descriptor in area 1	R/W1S
8	FILL_DSC1	0	End of refill event for any descriptor in area 1	R/W1S
7	ERR_LUT_MISS0	0	Unexpected Access to a yet-to-be-refilled area event in area 0	R/W1S
6	ERR_UPD_DATA0	0	Data register update whilst refilling error event in area 0	R/W1S

Table 4-23. DMM_PAT_IRQENABLE_SET Register Field Descriptions (continued)

Bit	Field	Value	Description	Type
5	ERR_UPD_CTRL0	0	Control register update whilst refilling error event in area 0	R/W1S
4	ERR_UPD_AREA0	0	Area register update whilst refilling error event in area 0	R/W1S
3	ERR_INV_DATA0	0	Invalid entry-table pointer error event in area 0	R/W1S
2	ERR_INV_DSC0	0	Invalid descriptor pointer error event in area 0	R/W1S
1	FILL_LST0	0	End of refill event for the last descriptor in area 0	R/W1S
0	FILL_DSC0	0	End of refill event for any descriptor in area 0	R/W1S

4.4.14 DMM PAT Interrupt Disable Register: DMM_PAT_IRQENABLE_CLR

The DMM PAT Interrupt Disable register is shown in Figure 4-70 and described in Table 4-24. Per-event interrupt enable bit vector. Write 1 to clear (disable interrupt).

Figure 4-70. DMM_PAT_IRQENABLE_CLR Register

31	30	29	28	27	26	25	24
ERR_LUT_MISS3	ERR_UPD_DATA3	ERR_UPD_CTRL3	ERR_UPD_AREA3	ERR_INV_DATA3	ERR_INV_DSC3	FILL_LST3	FILL_DSC3
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0
23	22	21	20	19	18	17	16
ERR_LUT_MISS2	ERR_UPD_DATA2	ERR_UPD_CTRL2	ERR_UPD_AREA2	ERR_INV_DATA2	ERR_INV_DSC2	FILL_LST2	FILL_DSC2
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0
15	14	13	12	11	10	9	8
ERR_LUT_MISS1	ERR_UPD_DATA1	ERR_UPD_CTRL1	ERR_UPD_AREA1	ERR_INV_DATA1	ERR_INV_DSC1	FILL_LST1	FILL_DSC1
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0
7	6	5	4	3	2	1	0
ERR_LUT_MISS0	ERR_UPD_DATA0	ERR_UPD_CTRL0	ERR_UPD_AREA0	ERR_INV_DATA0	ERR_INV_DSC0	FILL_LST0	FILL_DSC0
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-24. DMM_PAT_IRQENABLE_CLR Register Field Descriptions

Bit	Field	Value	Description	Type
31	ERR_LUT_MISS3	0	Unexpected Access to a yet-to-be-refilled area event in area 3	R/W1C
30	ERR_UPD_DATA3	0	Data register update whilst refilling error event in area 3	R/W1C
29	ERR_UPD_CTRL3	0	Control register update whilst refilling error event in area 3	R/W1C
28	ERR_UPD_AREA3	0	Area register update whilst refilling error event in area 3	R/W1C
27	ERR_INV_DATA3	0	Invalid entry-table pointer error event in area 3	R/W1C
26	ERR_INV_DSC3	0	Invalid descriptor pointer error event in area 3	R/W1C
25	FILL_LST3	0	End of refill event for the last descriptor in area 3	R/W1C
24	FILL_DSC3	0	End of refill event for any descriptor in area 3	R/W1C
23	ERR_LUT_MISS2	0	Unexpected Access to a yet-to-be-refilled area event in area 2	R/W1C
22	ERR_UPD_DATA2	0	Data register update whilst refilling error event in area 2	R/W1C
21	ERR_UPD_CTRL2	0	Control register update whilst refilling error event in area 2	R/W1C
20	ERR_UPD_AREA2	0	Area register update whilst refilling error event in area 2	R/W1C
19	ERR_INV_DATA2	0	Invalid entry-table pointer error event in area 2	R/W1C
18	ERR_INV_DSC2	0	Invalid descriptor pointer error event in area 2	R/W1C
17	FILL_LST2	0	End of refill event for the last descriptor in area 2	R/W1C
16	FILL_DSC2	0	End of refill event for any descriptor in area 2	R/W1C
15	ERR_LUT_MISS1	0	Unexpected Access to a yet-to-be-refilled area event in area 1	R/W1C
14	ERR_UPD_DATA1	0	Data register update whilst refilling error event in area 1	R/W1C
13	ERR_UPD_CTRL1	0	Control register update whilst refilling error event in area 1	R/W1C
12	ERR_UPD_AREA1	0	Area register update whilst refilling error event in area 1	R/W1C
11	ERR_INV_DATA1	0	Invalid entry-table pointer error event in area 1	R/W1C
10	ERR_INV_DSC1	0	Invalid descriptor pointer error event in area 1	R/W1C
9	FILL_LST1	0	End of refill event for the last descriptor in area 1	R/W1C
8	FILL_DSC1	0	End of refill event for any descriptor in area 1	R/W1C
7	ERR_LUT_MISS0	0	Unexpected Access to a yet-to-be-refilled area event in area 0	R/W1C
6	ERR_UPD_DATA0	0	Data register update whilst refilling error event in area 0	R/W1C

Table 4-24. DMM_PAT_IRQENABLE_CLR Register Field Descriptions (continued)

Bit	Field	Value	Description	Type
5	ERR_UPD_CTRL0	0	Control register update whilst refilling error event in area 0	R/W1C
4	ERR_UPD_AREA0	0	Area register update whilst refilling error event in area 0	R/W1C
3	ERR_INV_DATA0	0	Invalid entry-table pointer error event in area 0	R/W1C
2	ERR_INV_DSC0	0	Invalid descriptor pointer error event in area 0	R/W1C
1	FILL_LST0	0	End of refill event for the last descriptor in area 0	R/W1C
0	FILL_DSC0	0	End of refill event for any descriptor in area 0	R/W1C

4.4.15 DMM PAT Status Registers: DMM_PAT_STATUS_0-DMM_PAT_STATUS_3

The DMM PAT Status register is shown in Figure 4-71 and described in Table 4-25.

Figure 4-71. DMM_PAT_STATUS Registers

31	25	24	16
Reserved		CNT	
R-0		R-0	

15	10	9	8
ERROR		Reserved	
R-0		R-0	

7	6	4	3	2	1	0
BYPASSED	Reserved		DONE	RUN	VALID	READY
R-0	R-0		R-0	R-0	R-0	R-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-25. DMM_PAT_STATUS Registers Field Descriptions

Bit	Field	Value	Description	Type
31-25	Reserved	0	Reserved	R
24-16	CNT	0	Counter of remaining lines to re-load for engine n	R
15-10	ERROR		Error occurred in engine n	R
		0	No error	
		1h	Invalid descriptor provided	
		2h	Invalid data pointer provided	
		4h	Unexpected area register update whilst refilling	
		8h	Unexpected control register update whilst refilling	
		10h	Unexpected data register update whilst refilling	
		20h	Unexpected access to a yet-to-be-refilled location	
9-8	Reserved	0	Reserved	R
7	BYPASSED	0	Engine n is bypassed. Direct access to the LUT is provided.	R
6-4	Reserved	0	Reserved	R
3	DONE	0	Area reloading finished for engine n	R
2	RUN	0	Area currently reloading for engine n	R
1	VALID	0	Valid area description for engine n	R
0	READY	1	Area registers ready for engine n	R

4.4.16 DMM PAT Descriptor Registers: DMM_PAT_DESCR_0-DMM_PAT_DESCR_3

The DMM PAT Descriptor register is shown in Figure 4-72 and described in Table 4-26.

- Physical address of the next table refill descriptor
- Writing to this register aborts the current ongoing area reload
- Write also resets the corresponding DMM_PAT_AREA_x, DMM_PAT_CTRL_x and DMM_PAT_DATA_x registers
- The descriptor address must be in the DDR and not any other internal memory

Figure 4-72. DMM_PAT_DESCR Registers

31	4	3	0
ADDR		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-26. DMM_PAT_DESCR Registers Field Descriptions

Bit	Field	Value	Description	Type
31-4	ADDR	0	Physical address of the next table refill descriptor.	R/W
3-0	Reserved	0	Reserved	R

4.4.17 DMM PAT Area Geometry Registers: DMM_PAT_AREA_0-DMM_PAT_AREA_3

The DMM PAT Area Geometry register is shown in Figure 4-73 and described in Table 4-27.

Figure 4-73. DMM_PAT_AREA Registers

31	30 24	23 16	15	14 8	7 0
Rsvd	Y1	X1	Rsvd	Y0	X0
R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-27. DMM_PAT_AREA Registers Field Descriptions

Bit	Field	Value	Description	Type
31	Reserved	0	Reserved	R
30-24	Y1	0	Y-coordinate of the bottom right corner of the PAT area	R/W
23-16	X1	0	X-coordinate of the bottom right corner of the area	R/W
15	Reserved	0	Reserved	R
14-8	Y0	0	Y-coordinate of the top left corner of the PAT area	R/W
7-0	X0	0	X-coordinate of the top left corner of the PAT area	R/W

4.4.18 DMM PAT Control Registers: DMM_PAT_CTRL_0-DMM_PAT_CTRL_3

The DMM PAT Control register is shown in Figure 4-74 and described in Table 4-28.

Figure 4-74. DMM_PAT_CTRL Registers

31 28	27 17	16
INITIATOR	Reserved	SYNC
R/W-0	R-0	R/W-0

15 10	9 8	7	6 4	3 1	0
Reserved	LUT_ID	Rsvd	DIRECTION	Reserved	START
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-28. DMM_PAT_CTRL Registers Field Descriptions

Bit	Field	Value	Description	Type
31-28	INITIATOR	0	CONT_ID of the initiator for synchronisation	R/W
27-17	Reserved	0	Reserved	R
16	SYNC		DMM PAT table reload synchronisation	R/W
		0	Not synchronised	
		1	Synchronised	
15-10	Reserved	0	Reserved	R
9-8	LUT_ID	0	PAT LUT index	R/W
7	Reserved	0	Reserved	R
6-4	DIRECTION	0	Direction of this PAT table refill	R/W
3-1	Reserved	0	Reserved	R
0	START	0	Starting a PAT table refill	R/W

4.4.19 DMM PAT Area Entry Data Registers: DMM_PAT_DATA_0-DMM_PAT_DATA_3

The DMM PAT Area Entry Data register is shown in Figure 4-75 and described in Table 4-29.

- Physical address of the current table refill entry data
- The entry data table must in the DDR and the address should be 32 bytes aligned

Figure 4-75. DMM_PAT_DATA Registers

31 4	3 0
ADDR	Reserved
R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-29. DMM_PAT_DATA Registers Field Descriptions

Bit	Field	Value	Description	Type
31-4	ADDR	0	Physical address of the current table refill entry data or single actual entry data when in manual mode	R/W
3-0	Reserved	0	Reserved	R

4.4.20 DMM PEG Priority Registers: DMM_PEG_PRIO_0-DMM_PEG_PRIO_1

The DMM PEG Priority register is shown in Figure 4-76 and described in Table 4-30.

Figure 4-76. DMM_PEG_PRIO Registers

31	30 28	27	26 24	23	22 20	19	18 16
W7	P7	W6	P6	W5	P5	W4	P4
R/W-0	R/W-4h	R/W-0	R/W-4h	R/W-0	R/W-4h	R/W-0	R/W-4h

15	14 13 12	11	10 9 8	7	6 4	3	2 0
W3	P3	W2	P2	W1	P1	W0	P0
R/W-0	R/W-4h	R/W-0	R/W-4h	R/W-0	R/W-4h	R/W-0	R/W-4h

LEGEND: R/W = Read/Write; R = Read only; n = 0 for the first priority register; n = 1 for the second priority register

Table 4-30. DMM_PEG_PRIO Registers Field Descriptions

Bit	Field	Value	Description	Type
31	W7	0	Write-enable for P7 bit-field; P7 field is updated	R/W
30-28	P7	4h	Priority for initiator 8.n+7	R/W
27	W6	0	Write-enable for P6 bit-field; P6 field is updated	R/W
26-24	P6	4h	Priority for initiator 8.n+6	R/W
23	W5	0	Write-enable for P5 bit-field; P5 field is updated	R/W
22-20	P5	4h	Priority for initiator 8.n+5	R/W
19	W4	0	Write-enable for P4 bit-field; P4 field is updated	R/W
18-16	P4	4h	Priority for initiator 8.n+4	R/W
15	W3	0	Write-enable for P3 bit-field; P3 field is updated	R/W
14-12	P3	4h	Priority for initiator 8.n+3	R/W
11	W2	0	Write-enable for P2 bit-field; P2 field is updated	R/W
10-8	P2	4h	Priority for initiator 8.n+2	R/W
7	W1	0	Write-enable for P1 bit-field; P1 field is updated	R/W
6-4	P1	4h	Priority for initiator 8.n+1	R/W
3	W0	0	Write-enable for P0 bit-field; P0 field is updated	R/W
2-0	P0	4h	Priority for initiator 8.n	R/W

4.4.21 DMM PEG Priority Registers for PAT: DMM_PEG_PRIO_PAT

The DMM PEG Priority register for PAT is shown in Figure 4-77 and described in Table 4-31. The DMM PEG Priority register is for the internal PAT engine.

Figure 4-77. DMM_PEG_PRIO_PAT Register

31 4	3	2 0
Reserved	W_PAT	P_PAT
R-0	R/W-0	R/W-4h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-31. DMM_PEG_PRIO_PAT Register Field Descriptions

Bit	Field	Value	Description	Type
31-4	Reserved	0	Reserved	R
3	W_PAT	0	Write-enable for P_PAT bit-field; P_PAT field is updated	R/W
2-0	P_PAT	4h	Priority for PAT engine.	R/W

Enhanced Direct Memory Access (EDMA3) Controller

This chapter describes the features and operations of the enhanced direct memory access (EDMA3) controller.

Topic **Page**

5.1 Introduction

5.1.1 Overview

The enhanced direct memory access (EDMA3) controller's primary purpose is to service user-programmed data transfers between two memory-mapped slave endpoints on the device.

Typical usage includes, but is not limited to the following:

- Servicing software-driven paging transfers (for example, transfers from external memory, such as DDR2 to internal device memory, such as DSP L2 SRAM).
- Servicing event-driven peripherals, such as a serial port.
- Performing sorting or sub-frame extraction of various data structures.
- Offloading data transfers from the main device CPU(s) or DSP(s) (see the device-specific data manual for specific peripherals that are accessible via the EDMA3 controller.).

The EDMA3 controller has a different architecture from the previous EDMA2 controller on the TMS320C621x/C671x DSPs and TMS320C64x DSPs. (See the *EDMA v3.0 (EDMA3) Migration Guide for TMS320C645x DSP* (SPRAAB9) for more information on new/advanced features.)

The EDMA3 controller consists of two principal blocks:

- EDMA3 channel controller (EDMA3CC).
- EDMA3 transfer controller(s) (EDMA3TC).

The EDMA3 channel controller serves as the user interface for the EDMA3 controller. The EDMA3CC includes parameter RAM (PaRAM), channel control registers, and interrupt control registers. The EDMA3CC serves to prioritize incoming software requests or events from peripherals and submits transfer requests (TRs) to the transfer controller.

The EDMA3 transfer controllers are slaves to the EDMA3 channel controller that is responsible for data movement. The transfer controller issues read/write commands to the source and destination addresses that are programmed for a given transfer. The operation is transparent to user.

5.1.2 Features

The EDMA3 channel controller has following features:

- Fully orthogonal transfer description:
  - Three transfer dimensions.
  - A-synchronized transfers: one-dimension serviced per event.
  - AB-synchronized transfers: two-dimensions serviced per event.
  - Independent indexes on source and destination.
  - Chaining feature allows a 3-D transfer based on a single event.
- Flexible transfer definition:
  - Increment or FIFO transfer addressing modes.
  - Linking mechanism allows automatic PaRAM set update.
  - Chaining allows multiple transfers to execute with one event.
- Interrupt generation for the following:
  - Transfer completion.
  - Error conditions.
  - Error conditions routed to Cortex-A8 only.
- Up to 8 interrupt outputs for multi-core support.
- Debug visibility:
  - Queue water marking/threshold.
  - Error and status recording to facilitate debug.

- 64 DMA channels:
  - Event synchronization.
  - Manual synchronization (CPU(s) write to event set register).
  - Chain synchronization (completion of one transfer triggers another transfer).
  - Support for programmable DMA channel to PaRAM mapping.
- Eight QDMA channels:
  - QDMA channels trigger automatically upon writing to a parameter RAM (PaRAM) set entry.
  - Support for programmable QDMA channel to PaRAM mapping.
- 512 PaRAM sets:
  - Each PaRAM set can be used for a DMA channel, QDMA channel, or link set.
- Four transfer controllers/event queues. You program the system-level priority of these queues. (See the device data manual for the possible system priorities.)
- 16 event entries per event queue.
- Memory protection support:
  - Proxy memory protection for TR submission.
  - Active memory protection for accesses to PaRAM and registers.

The EDMA3 transfer controller has the following features:

- Four transfer controllers (TC0-TC3).
- 128-bit wide read and write ports per TC.
- Up to four in-flight transfer requests (TRs).
- Programmable priority level.
- Supports two-dimensional transfers with independent indexes on source and destination (EDMA3CC manages the 3rd dimension).
- Support for increment or constant addressing mode transfers.
- Interrupt and error support.
- Memory-Mapped Register (MMR) bit fields are fixed position in 32-bit MMR regardless of endianness.

5.1.3 Terminology Used in This Document

The following is a brief explanation of some terms that are used in this document:

A-synchronized transfer— A transfer type where one dimension is serviced per synchronization event.

AB-synchronized transfer— A transfer type where two dimensions are serviced per synchronization event.

Chaining— A trigger mechanism in which a transfer can be initiated at the completion of another transfer or sub-transfer.

CPU(s)— The main processing engine or engines on a device. The CPU is typically a DSP or general-purpose processor (see the device-specific data manual to learn more about the CPU on your system.)

DMA channel— One of the 64 channels that external, manual, or chained events can trigger. All direct memory access (DMA) channels exist in the EDMA3CC.

Dummy set or dummy PaRAM set— A PaRAM set for which at least one of the count fields is equal to 0 and at least one of the count fields is nonzero. All of the count fields are cleared in a null PaRAM set.

Dummy transfer— A dummy set results in the EDMA3CC performing a dummy transfer. This is not an error condition. A null set results in an error condition.

EDMA3 channel controller (EDMA3CC)— The EDMA3CC is the portion of the EDMA3 that you program. The EDMA3CC contains the parameter RAM (PaRAM), event processing logic, DMA/QDMA channels, and event queues. The EDMA3CC service events (external, manual, chained, and QDMA) and is responsible for submitting transfer requests to the transfer controllers (EDMA3TC) that perform the actual transfer.

EDMA3 programmer— Any entity on the chip that has read/write access to the EDMA3 registers and can program an EDMA3 transfer.

EDMA3 transfer controller(s) (EDMA3TC)— Transfer controllers are the transfer engines for the EDMA3 controller. They perform the read/writes, as dictated by the EDMA3CC's transfer requests.

Enhanced direct memory access (EDMA3) controller— EDMA3 consists of the EDMA3 channel controller (EDMA3CC) and the EDMA3 transfer controller(s) (EDMA3TC), referred to as EDMA3 in this document.

L3— System bus infrastructure that arbitrates and decodes bus transactions from multiple masters to multiple slaves.

Link parameter set — A PaRAM set that is used for linking.

Linking— The mechanism of reloading a PaRAM set with new transfer characteristics on completion of the current transfer.

Memory-mapped slave— All on-chip memories, off-chip memories, and slave peripherals. These typically rely on the EDMA3 (or other master peripheral) to perform transfers to and from them.

Master peripherals— All peripherals that are capable of initiating read and write transfers to the system that may not solely rely on the EDMA3 for their data transfers.

Null set or null PaRAM set— A PaRAM set that has all count fields cleared (except for the link field). A dummy PaRAM set has at least one of the count fields nonzero.

Null transfer— A trigger event for a null PaRAM set results in the EDMA3CC performing a null transfer. This is an error condition. A dummy transfer is not an error condition.

Parameter RAM (PaRAM)— Programmable RAM that stores PaRAM sets that DMA channels, QDMA channels, and linking uses.

Parameter RAM (PaRAM) set— The PaRAM set is a 32-byte EDMA3 channel transfer definition. Each parameter set consists of eight words (that are four bytes each) that store the context for a DMA/QDMA/link transfer. A PaRAM set includes source address, destination address, counts, indexes, and options.

Parameter RAM (PaRAM) set entry— A PaRAM set entry occurs when one of the eight four-byte components of the parameter set.

QDMA channel— A QDMA channel is one of the four channels that you can trigger when writing to the trigger word (TRWORD) of a PaRAM set. All QDMA channels exist in the EDMA3CC.

Slave end points— Slave end points are all on-chip memories, off-chip memories, and slave peripherals. Slave end points may rely on the EDMA3 to perform transfers to and from them.

Transfer request (TR)— A command for data movement that is issued from the EDMA3CC to the EDMA3TC. A TR includes source and destination addresses, counts, indexes, and options.

Trigger event— A trigger event is an action that causes the EDMA3CC to service the channel and to submit a transfer request to the EDMA3TC. Trigger events for the DMA channels include events that are triggered manually, externally, and by chain. Trigger events for QDMA channels include events that are triggered automatically and by link.

Trigger word— For QDMA channels, the trigger word specifies the PaRAM set entry that results in a QDMA trigger event when it is written. The trigger word is programmed via the QDMA channel map register (QCHMAP) and can point to any of the PaRAM set entries.

TR synchronization (sync) event— See Trigger event.

5.2 Architecture

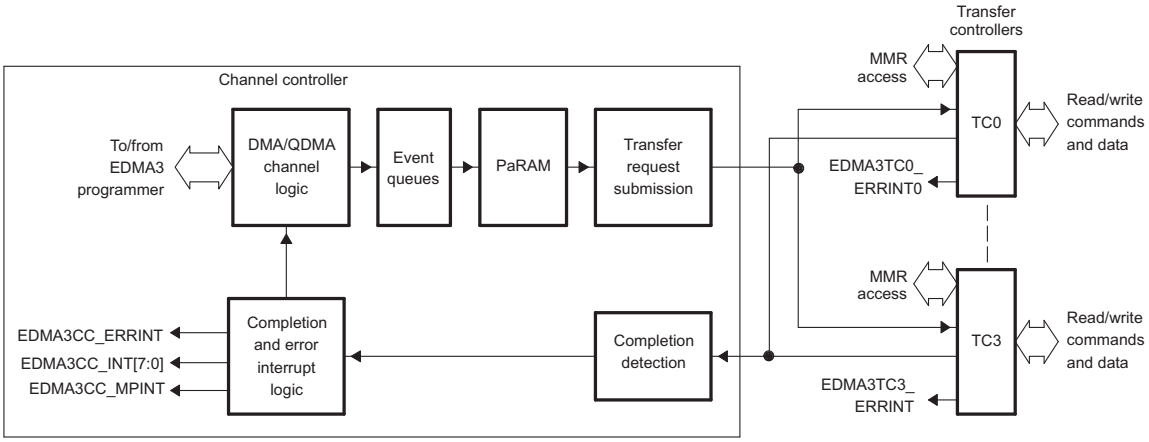
This chapter discusses the architecture of the EDMA3 controller.

5.2.1 Functional Overview

5.2.1.1 EDMA3 Controller Block Diagram

Figure 5-1 shows a block diagram for the EDMA3 controller.

Figure 5-1. EDMA3 Controller Block Diagram



5.2.1.2 EDMA3 Channel Controller (EDMA3CC)

Figure 5-2 shows a functional block diagram of the EDMA3 channel controller (EDMA3CC).

The main blocks of the EDMA3CC are as follows:

- Parameter RAM (PaRAM): The PaRAM maintains parameter sets for channel and reload parameter sets. You must write the PaRAM with the transfer context for the desired channels and link parameter sets. EDMA3CC processes sets based on a trigger event and submits a transfer request (TR) to the transfer controller.
- EDMA3 event and interrupt processing registers: Allows mapping of events to parameter sets, enable/disable events, enable/disable interrupt conditions, and clearing interrupts.
- Completion detection: The completion detect block detects completion of transfers by the EDMA3TC and/or slave peripherals. You can optionally use completion of transfers to chain trigger new transfers or to assert interrupts.
- Event queues: Event queues form the interface between the event detection logic and the transfer request submission logic.
- Memory protection registers: Memory protection registers define the accesses (privilege level and requestor(s)) that are allowed to access the DMA channel shadow region view(s) and regions of PaRAM.

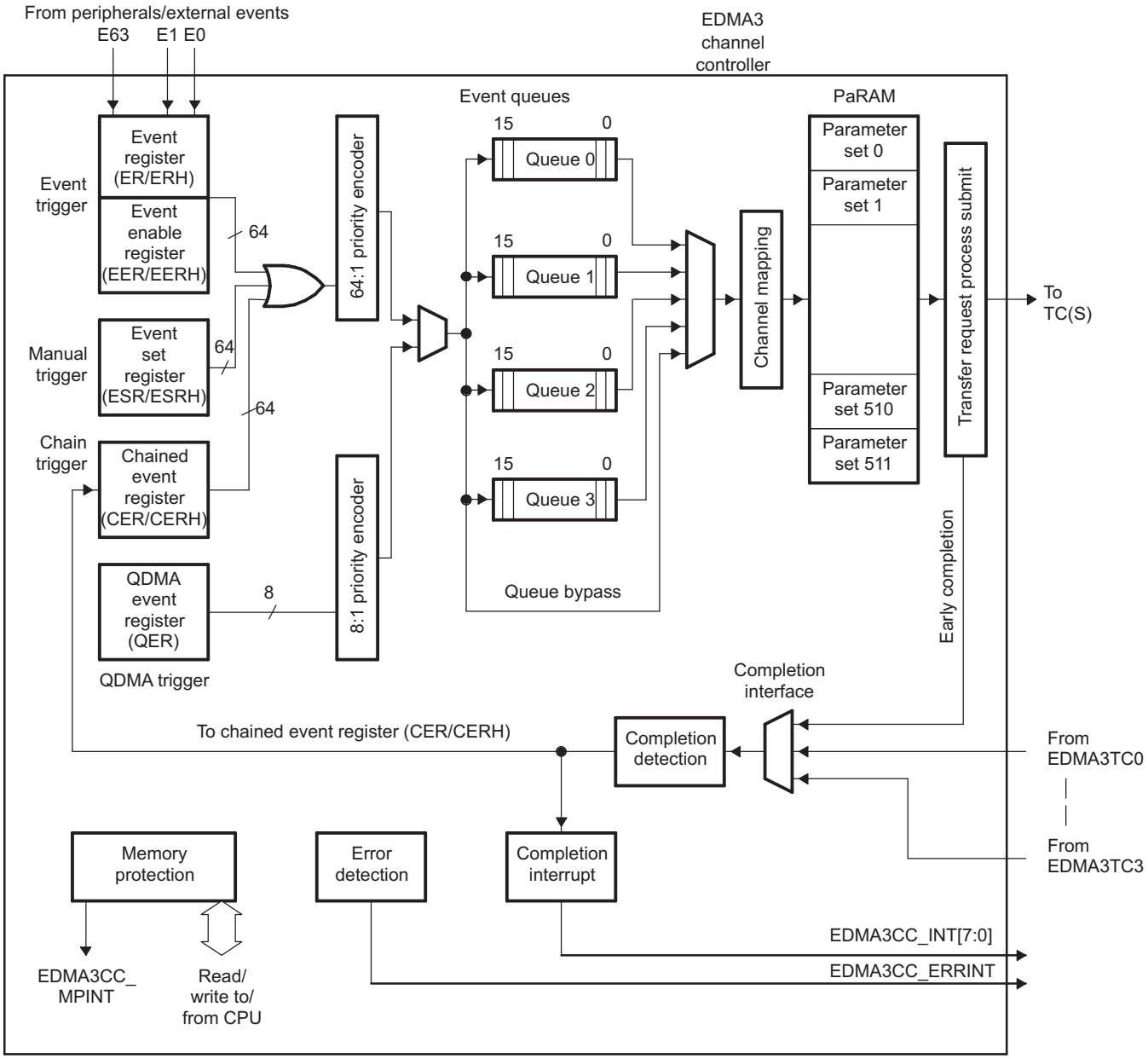
Other functions include the following:

- Region registers: Region registers allow DMA resources (DMA channels and interrupts) to be assigned to unique regions that different EDMA3 programmers own (for example, ARM or DSP).
- Debug registers: Debug registers allow debug visibility by providing registers to read the queue status, controller status, and missed event status.

The EDMA3CC includes two channel types: DMA channels (64 channels) and QDMA channels (8 channels).

Each channel is associated with a given event queue/transfer controller and with a given PaRAM set. The main thing that differentiates a DMA channel from a QDMA channel is the method that the system uses to trigger transfers. See Section 5.2.4.

Figure 5-2. EDMA3 Channel Controller (EDMA3CC) Block Diagram



A trigger event is needed to initiate a transfer. A trigger event may be due to an external event, manual write to the event set register, or chained event for DMA channels. QDMA channels auto-trigger when a write to the trigger word that you program occurs on the associated PaRAM set. All such trigger events are logged into appropriate registers upon recognition.

Once a trigger event is recognized, the appropriate event gets queued in the EDMA3CC event queue. The assignment of each DMA/QDMA channel to an event queue is programmable. Each queue is 16 events deep; therefore, you can queue up to 16 events (on a single queue) in the EDMA3CC at a time. Additional pending events that are mapped to a full queue are queued when the event queue space becomes available. See Section 5.2.11.

If events on different channels are detected simultaneously, the events are queued based on a fixed priority arbitration scheme with the DMA channels being higher priority events than the QDMA channels. Among the two groups of channels, the lowest-numbered channel is the highest priority.

Each event in the event queue is processed in FIFO order. When the head of the queue is reached, the PaRAM associated with that channel is read to determine the transfer details. The TR submission logic evaluates the validity of the TR and is responsible for submitting a valid transfer request (TR) to the appropriate EDMA3TC (based on the event queue to the EDMA3TC association, Q0 goes to TC0 , Q1 goes to TC1, Q2 goes to TC2, and Q3 goes to TC3). For more information, refer to Section 5.2.3.

The EDMA3TC receives the request and is responsible for data movement, as specified in the transfer request packet (TRP), other necessary tasks like buffering, and ensuring transfers are carried out in an optimal fashion wherever possible. For more information on EDMA3TC, refer to Section 5.2.1.3.

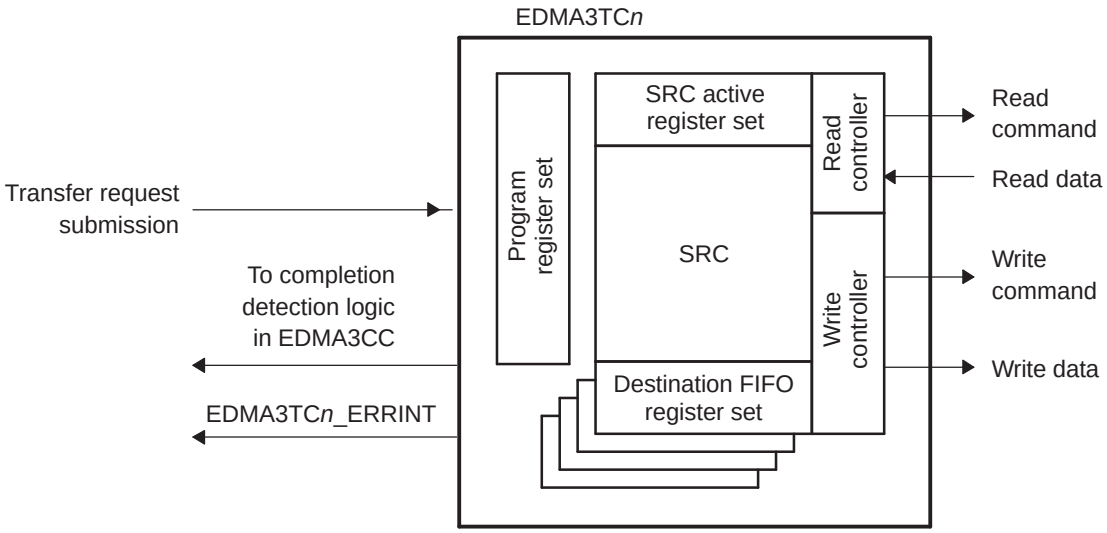
If you have decided to receive an interrupt or to chain to another channel on completion of the current transfer, the EDMA3TC signals completion to the EDMA3CC completion detection logic when the transfer is complete. You can alternately choose to trigger completion when a TR leaves the EDMA3CC boundary, rather than wait for all of the data transfers to complete. Based on the setting of the EDMA3CC interrupt registers, the completion interrupt generation logic is responsible for generating EDMA3CC completion interrupts to the CPU. For more information, refer to Section 5.2.5.

Additionally, the EDMA3CC also has an error detection logic that causes an error interrupt generation on various error conditions (like missed events, exceeding event queue thresholds, etc.). For more information on error interrupts, refer to Section 5.2.9.4.

5.2.1.3 EDMA3 Transfer Controller (EDMA3TC)

Section 5.2.9.4 shows a functional block diagram of the EDMA3 transfer controller (EDMA3TC).

Figure 5-3. EDMA3 Transfer Controller (EDMA3TC) Block Diagram



The main blocks of the EDMA3TC are:

- DMA program register set: The DMA program register set stores the transfer requests received from the EDMA3 channel controller (EDMA3CC).
- DMA source active register set: The DMA source active register set stores the context for the DMA transfer request currently in progress in the read controller.
- Read controller: The read controller issues read commands to the source address.
- Destination FIFO register set: The destination (DST) FIFO register set stores the context for the DMA transfer request(s) currently in progress in the write controller.
- Write controller: The write controller issues write commands/write data to the destination slave.

- Data FIFO: The data FIFO exists for holding temporary in-flight data.
- Completion interface: The completion interface sends completion codes to the EDMA3CC when a transfer completes, and generates interrupts and chained events (also, see Section 5.2.1.2 for more information on transfer completion reporting).

When the EDMA3TC is idle and receives its first TR, DMA program register set receives the TR, where it transitions to the DMA source active set and the destination FIFO register set immediately. The second TR (if pending from EDMA3CC) is loaded into the DMA program set, ensuring it can start as soon as possible when the active transfer completes. As soon as the current active set is exhausted, the TR is loaded from the DMA program register set into the DMA source active register set as well as to the appropriate entry in the destination FIFO register set.

The read controller issues read commands governed by the rules of command fragmentation and optimization. These are issued only when the data FIFO has space available for the data read. When sufficient data is in the data FIFO, the write controller starts issuing a write command again following the rules for command fragmentation and optimization. For more information on command fragmentation and optimization, refer to Section 5.2.12.1.1.

Depending on the number of entries, the read controller can process up to two or four transfer requests ahead of the destination subject to the amount of free data FIFO.

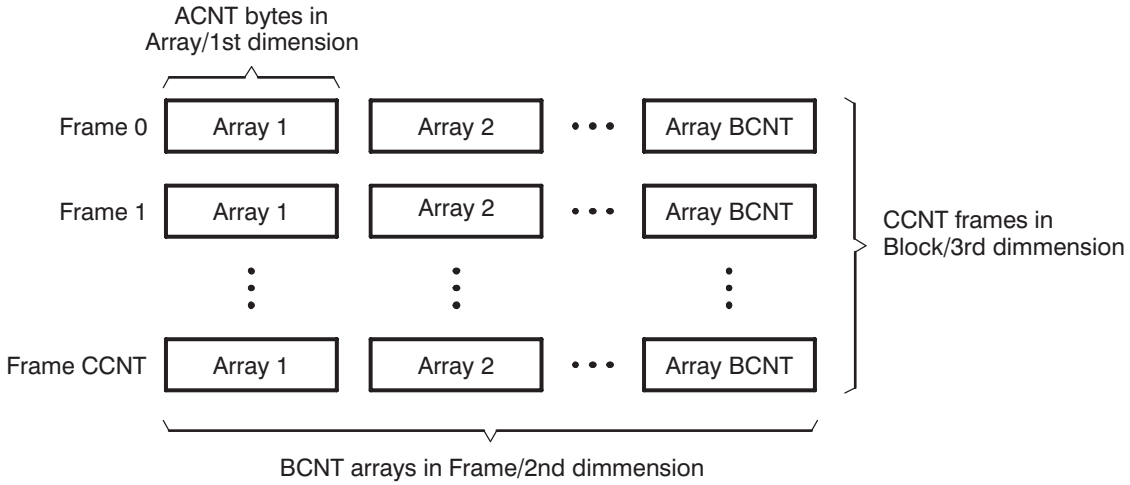
5.2.2 Types of EDMA3 Transfers

An EDMA3 transfer is always defined in terms of three dimensions. Figure 5-4 shows the three dimensions used by EDMA3 transfers. These three dimensions are defined as:

- 1st Dimension or Array (A): The 1st dimension in a transfer consists of ACNT contiguous bytes.
- 2nd Dimension or Frame (B): The 2nd dimension in a transfer consists of BCNT arrays of ACNT bytes. Each array transfer in the 2nd dimension is separated from each other by an index programmed using SRCBIDX or DSTBIDX.
- 3rd Dimension or Block (C): The 3rd dimension in a transfer consists of CCNT frames of BCNT arrays of ACNT bytes. Each transfer in the 3rd dimension is separated from the previous by an index programmed using SRCCIDX or DSTCIDX.

Note that the reference point for the index depends on the synchronization type. The amount of data transferred upon receipt of a trigger/synchronization event is controlled by the synchronization types (SYNCDIM bit in OPT). Of the three dimensions, only two synchronization types are supported: A-synchronized transfers and AB-synchronized transfers.

Figure 5-4. Definition of ACNT, BCNT, and CCNT



5.2.2.1 A-Synchronized Transfers

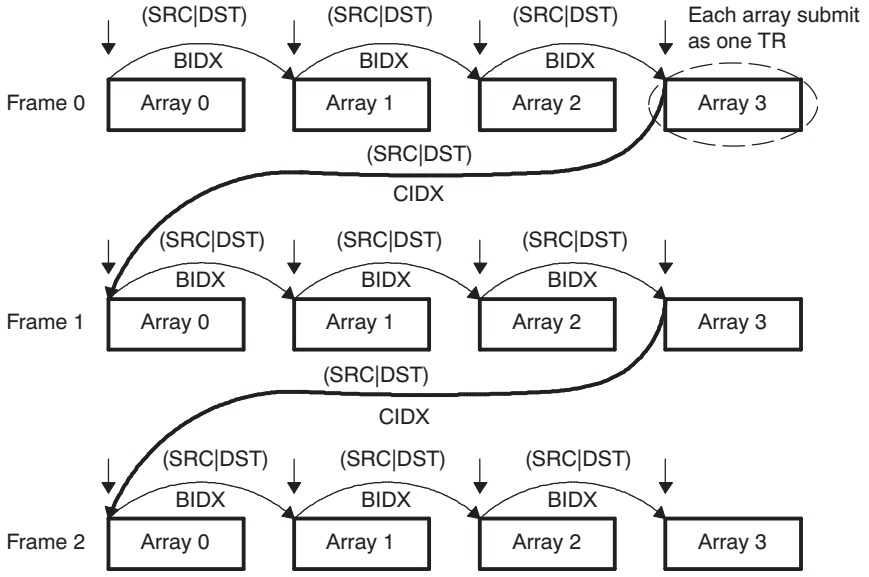
In an A-synchronized transfer, each EDMA3 sync event initiates the transfer of the 1st dimension of ACNT bytes, or one array of ACNT bytes. In other words, each event/TR packet conveys the transfer information for one array only. Thus, BCNT × CCNT events are needed to completely service a PaRAM set.

Arrays are always separated by SRCBIDX and DSTBIDX, as shown in Figure 5-5, where the start address of Array N is equal to the start address of Array N – 1 plus source (SRC) or destination (DST) BIDX.

Frames are always separated by SRCCIDX and DSTCIDX. For A-synchronized transfers, after the frame is exhausted, the address is updated by adding SRCCIDX/DSTCIDX to the beginning address of the last array in the frame. As in Figure 5-5, SRCCIDX/DSTCIDX is the difference between the start of Frame 0 Array 3 to the start of Frame 1 Array 0.

Figure 5-5 shows an A-synchronized transfer of 3 (CCNT) frames of 4 (BCNT) arrays of n (ACNT) bytes. In this example, a total of 12 sync events (BCNT × CCNT) exhaust a PaRAM set. See Section 5.2.3.6 for details on parameter set updates.

Figure 5-5. A-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3)



5.2.2.2 AB-Synchronized Transfers

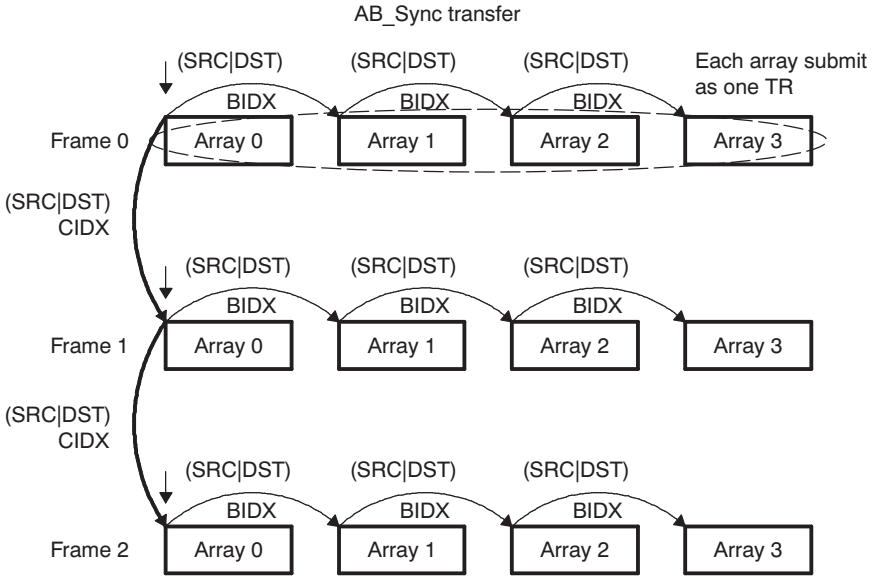
In a AB-synchronized transfer, each EDMA3 sync event initiates the transfer of 2 dimensions or one frame. In other words, each event/TR packet conveys information for one entire frame of BCNT arrays of ACNT bytes. Thus, CCNT events are needed to completely service a PaRAM set.

Arrays are always separated by SRCBIDX and DSTBIDX as shown in Figure 5-6. Frames are always separated by SRCCIDX and DSTCIDX.

Note that for AB-synchronized transfers, after a TR for the frame is submitted, the address update is to add SRCCIDX/DSTCIDX to the beginning address of the beginning array in the frame. This is different from A-synchronized transfers where the address is updated by adding SRCCIDX/DSTCIDX to the start address of the last array in the frame. See Section 5.2.3.6 for details on parameter set updates.

Figure 5-6 shows an AB-synchronized transfer of 3 (CCNT) frames of 4 (BCNT) arrays of *n* (ACNT) bytes. In this example, a total of 3 sync events (CCNT) exhaust a PaRAM set; that is, a total of 3 transfers of 4 arrays each completes the transfer.

Figure 5-6. AB-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3)



NOTE: ABC-synchronized transfers are not directly supported. But can be logically achieved by chaining between multiple AB-synchronized transfers.

5.2.3 Parameter RAM (PaRAM)

The EDMA3 controller is a RAM-based architecture. The transfer context (source/destination addresses, count, indexes, etc.) for DMA or QDMA channels is programmed in a parameter RAM table within EDMA3CC, referred to as PaRAM. The PaRAM table is segmented into multiple PaRAM sets. Each PaRAM set includes eight four-byte PaRAM set entries (32-bytes total per PaRAM set), which includes typical DMA transfer parameters such as source address, destination address, transfer counts, indexes, options, etc.

The PaRAM structure supports flexible ping-pong, circular buffering, channel chaining, and auto-reloading (linking).

The contents of the PaRAM include the following:

- 512 PaRAM sets
- 64 channels that are direct mapped and can be used as link or QDMA sets if not used for DMA channels
- 64 channels remain for link or QDMA sets

By default, all channels map to PaRAM set to 0. These should be remapped before use. For more information, see Section 5.4.1.1.3 (DCHMAP registers) and Section 5.4.1.1.4 (QCHMAP registers).

Table 5-1. EDMA3 Parameter RAM Contents

PaRAM Set Number	Address[1]	Parameters[2]
0	EDMA Base Address + 4000h to EDMA Base Address + 401Fh	PaRAM set 0
1	EDMA Base Address + 4020h to EDMA Base Address + 403Fh	PaRAM set 1
2	EDMA Base Address + 4040h to EDMA Base Address + 405Fh	PaRAM set 2
3	EDMA Base Address + 4060h to EDMA Base Address + 407Fh	PaRAM set 3
4	EDMA Base Address + 4080h to EDMA Base Address + 409Fh	PaRAM set 4
5	EDMA Base Address + 40A0h to EDMA Base Address + 40BFh	PaRAM set 5
6	EDMA Base Address + 40C0h to EDMA Base Address + 40DFh	PaRAM set 6
7	EDMA Base Address + 40E0h to EDMA Base Address + 40FFh	PaRAM set 7
8	EDMA Base Address + 4100h to EDMA Base Address + 411Fh	PaRAM set 8
9	EDMA Base Address + 4120h to EDMA Base Address + 413Fh	PaRAM set 9
...	...	...
63	EDMA Base Address + 47E0h to EDMA Base Address + 47FFh	PaRAM set 63
64	EDMA Base Address + 4800h to EDMA Base Address + 481Fh	PaRAM set 64
65	EDMA Base Address + 4820h to EDMA Base Address + 483Fh	PaRAM set 65
...	...	...
510	EDMA Base Address + 7FC0h to EDMA Base Address + 7FDFh	PaRAM set 510
511	EDMA Base Address + 7FE0h to EDMA Base Address + 7FFFh	PaRAM set 511

(1) The EDMA base addess can be obtained from the device-specific data manual.
(2) The device has 8 QDMA channels that can be mapped to any parameter set number from 0 to 511.

5.2.3.1 PaRAM

Each parameter set of PaRAM is organized into eight 32-bit words or 32 bytes, as shown in Figure 5-7 and described in Table 5-2. Each PaRAM set consists of 16-bit and 32-bit parameters.

Figure 5-7. PaRAM Set

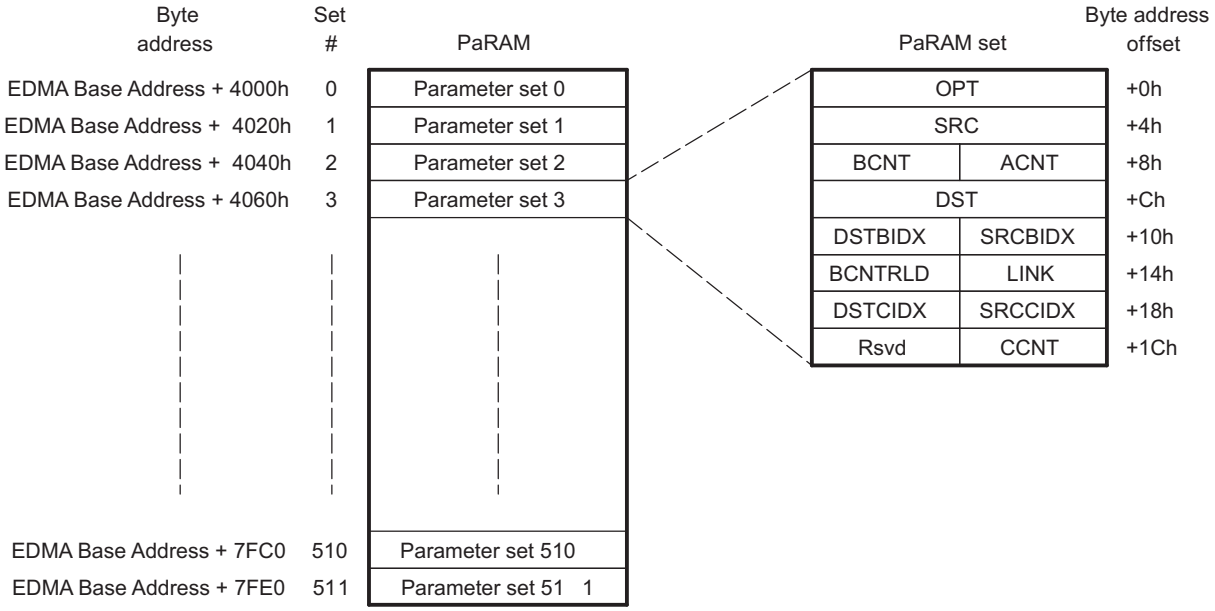


Table 5-2. EDMA3 Channel Parameter Description

Offset Address (bytes)	Acronym	Parameter	Description
0h	OPT	Channel Options	Transfer configuration options
4h	SRC	Channel Source Address	The byte address from which data is transferred
8h[(1)]	ACNT	Count for 1st Dimension	Unsigned value specifying the number of contiguous bytes within an array (first dimension of the transfer). Valid values range from 1 to 65 535.
	BCNT	Count for 2nd Dimension	Unsigned value specifying the number of arrays in a frame, where an array is ACNT bytes. Valid values range from 1 to 65 535.
Ch	DST	Channel Destination Address	The byte address to which data is transferred
10h[(1)]	SRCBIDX	Source BCNT Index	Signed value specifying the byte address offset between source arrays within a frame (2nd dimension). Valid values range from –32 768 and 32 767.
	DSTBIDX	Destination BCNT Index	Signed value specifying the byte address offset between destination arrays within a frame (2nd dimension). Valid values range from –32 768 and 32 767.
14h[(1)]	LINK	Link Address	The PaRAM address containing the PaRAM set to be linked (copied from) when the current PaRAM set is exhausted. A value of FFFFh specifies a null link.
	BCNTRLD	BCNT Reload	The count value used to reload BCNT when BCNT decrements to 0 (TR is submitted for the last array in 2nd dimension). Only relevant in A-synchronized transfers.
18h[(1)]	SRCCIDX	Source CCNT Index	Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from –32 768 and 32 767. A-synchronized transfers: The byte address offset from the beginning of the last source array in a frame to the beginning of the first source array in the next frame. AB-synchronized transfers: The byte address offset from the beginning of the first source array in a frame to the beginning of the first source array in the next frame.
	DSTCIDX	Destination CCNT index	Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from –32 768 and 32 767. A-synchronized transfers: The byte address offset from the beginning of the last destination array in a frame to the beginning of the first destination array in the next frame. AB-synchronized transfers: The byte address offset from the beginning of the first destination array in a frame to the beginning of the first destination array in the next frame.
1Ch	CCNT	Count for 3rd Dimension	Unsigned value specifying the number of frames in a block, where a frame is BCNT arrays of ACNT bytes. Valid values range from 1 to 65 535.
	RSVD	Reserved	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.

(1) If OPT, SRC, or DST is the trigger word for a QDMA transfer then it is required to do a 32-bit access to that field. Furthermore, it is recommended to perform only 32-bit accesses on the parameter RAM for best code compatibility. For example, switching the endianness of the processor swaps addresses of the 16-bit fields, but 32-bit accesses avoid the issue entirely.

5.2.3.2 EDMA3 Channel PaRAM Set Entry Fields

5.2.3.2.1 Channel Options Parameter (OPT)

The channel options parameter (OPT) is shown in Figure 5-8 and described in Table 5-3.

Figure 5-8. Channel Options Parameter (OPT)

31	30 28	27 24	23	22	21	20	19 18	17 16
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC
R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15 12	11	10 8	7 4	3	2	1	0
TCC	TCCMODE	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Table 5-3. Channel Options Parameters (OPT) Field Descriptions

Bit	Field	Value	Description
31	PRIV		Privilege level (supervisor versus user) for the host/CPU/DMA that programmed this PaRAM set. This value is set with the EDMA3 master's privilege value when any part of the PaRAM set is written.
		0	User level privilege.
		1	Supervisor level privilege.
30-28	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
27-24	PRIVID	0-Fh	Privilege identification for the external host/CPU/DMA that programmed this PaRAM set. This value is set with the EDMA3 master's privilege identification value when any part of the PaRAM set is written.
23	ITCCHEN		Intermediate transfer completion chaining enable.
		0	Intermediate transfer complete chaining is disabled.
		1	Intermediate transfer complete chaining is enabled.
			When enabled, the chained event register (CER/CERH) bit is set on every intermediate chained transfer completion (upon completion of every intermediate TR in the PaRAM set, except the final TR in the PaRAM set). The bit (position) set in CER or CERH is the TCC value specified.
22	TCCHEN		Transfer complete chaining enable.
		0	Transfer complete chaining is disabled.
		1	Transfer complete chaining is enabled.
			When enabled, the chained event register (CER/CERH) bit is set on final chained transfer completion (upon completion of the final TR in the PaRAM set). The bit (position) set in CER or CERH is the TCC value specified.
21	ITCINTEN		Intermediate transfer completion interrupt enable.
		0	Intermediate transfer complete interrupt is disabled.
		1	Intermediate transfer complete interrupt is enabled.
			When enabled, the interrupt pending register (IPR / IPRH) bit is set on every intermediate transfer completion (upon completion of every intermediate TR in the PaRAM set, except the final TR in the PaRAM set). The bit (position) set in IPR or IPRH is the TCC value specified. To generate a completion interrupt to the CPU, the corresponding IER [TCC] / IERH [TCC] bit must be set.
20	TCINTEN		Transfer complete interrupt enable.
		0	Transfer complete interrupt is disabled.
		1	Transfer complete interrupt is enabled.
			When enabled, the interrupt pending register (IPR / IPRH) bit is set on transfer completion (upon completion of the final TR in the PaRAM set). The bit (position) set in IPR or IPRH is the TCC value specified. To generate a completion interrupt to the CPU, the corresponding IER[TCC] / IERH [TCC] bit must be set.
19-18	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
17-12	TCC	0-3Fh	Transfer complete code. This 6-bit code sets the relevant bit in the chaining enable register (CER [TCC] /CERH [TCC]) for chaining or in the interrupt pending register (IPR [TCC] / IPRH [TCC]) for interrupts.

Table 5-3. Channel Options Parameters (OPT) Field Descriptions (continued)

Bit	Field	Value	Description
11	TCCMODE		Transfer complete code mode. Indicates the point at which a transfer is considered completed for chaining and interrupt generation.
		0	Normal completion: A transfer is considered completed after the data has been transferred.
		1	Early completion: A transfer is considered completed after the EDMA3CC submits a TR to the EDMA3TC. TC may still be transferring data when the interrupt/chain is triggered.
10-8	FWID	0-7h	FIFO Width. Applies if either SAM or DAM is set to constant addressing mode.
		0	FIFO width is 8-bit.
		1h	FIFO width is 16-bit.
		2h	FIFO width is 32-bit.
		3h	FIFO width is 64-bit.
		4h	FIFO width is 128-bit.
		5h	FIFO width is 256-bit.
		6h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-4	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3	STATIC		Static set.
		0	Set is not static. The PaRAM set is updated or linked after a TR is submitted. A value of 0 should be used for DMA channels and for non-final transfers in a linked list of QDMA transfers.
		1	Set is static. The PaRAM set is not updated or linked after a TR is submitted. A value of 1 should be used for isolated QDMA transfers or for the final transfer in a linked list of QDMA transfers.
2	SYNCDIM		Transfer synchronization dimension.
		0	A-synchronized. Each event triggers the transfer of a single array of ACNT bytes.
		1	AB-synchronized. Each event triggers the transfer of BCNT arrays of ACNT bytes.
1	DAM		Destination address mode.
		0	Increment (INCR) mode. Destination addressing within an array increments. Destination is not a FIFO.
		1	Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width.
0	SAM		Source address mode.
		0	Increment (INCR) mode. Source addressing within an array increments. Source is not a FIFO.
		1	Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width.

5.2.3.2.2 Channel Source Address (SRC)

The 32-bit source address parameter specifies the starting byte address of the source. For SAM in increment mode, there are no alignment restrictions imposed by EDMA3. For SAM in constant addressing mode, you must program the source address to be aligned to a 256-bit aligned address (5 LSBs of address must be 0). The EDMA3TC will signal an error, if this rule is violated. See Section 5.2.12.3 for additional details.

5.2.3.2.3 Channel Destination Address (DST)

The 32-bit destination address parameter specifies the starting byte address of the destination. For DAM in increment mode, there are no alignment restrictions imposed by EDMA3. For DAM in constant addressing mode, you must program the destination address to be aligned to a 256-bit aligned address (5 LSBs of address must be 0). The EDMA3TC will signal an error, if this rule is violated. See Section 5.2.12.3 for additional details.

5.2.3.2.4 Count for 1st Dimension (ACNT)

ACNT represents the number of bytes within the 1st dimension of a transfer. ACNT is a 16-bit unsigned value with valid values between 0 and 65 535. Therefore, the maximum number of bytes in an array is 65 535 bytes (64K – 1 bytes). ACNT must be greater than or equal to 1 for a TR to be submitted to EDMA3TC. A transfer with ACNT equal to 0 is considered either a null or dummy transfer. A dummy or null transfer generates a completion code depending on the settings of the completion bit fields in OPT.

See Section 5.2.3.5 and Section 5.2.5.3 for details on dummy/null completion conditions.

5.2.3.2.5 Count for 2nd Dimension (BCNT)

BCNT is a 16-bit unsigned value that specifies the number of arrays of length ACNT. For normal operation, valid values for BCNT are between 1 and 65 535. Therefore, the maximum number of arrays in a frame is 65 535 (64K – 1 arrays). A transfer with BCNT equal to 0 is considered either a null or dummy transfer. A dummy or null transfer generates a completion code depending on the settings of the completion bit fields in OPT.

See Section 5.2.3.5 and Section 5.2.5.3 for details on dummy/null completion conditions.

5.2.3.2.6 Count for 3rd Dimension (CCNT)

CCNT is a 16-bit unsigned value that specifies the number of frames in a block. Valid values for CCNT are between 1 and 65 535. Therefore, the maximum number of frames in a block is 65 535 (64K – 1 frames). A transfer with CCNT equal to 0 is considered either a null or dummy transfer. A dummy or null transfer generates a completion code depending on the settings of the completion bit fields in OPT.

A CCNT value of 0 is considered either a null or dummy transfer. See Section 5.2.3.5 and Section 5.2.5.3 for details on dummy/null completion conditions.

5.2.3.2.7 BCNT Reload (BCNTRLD)

BCNTRLD is a 16-bit unsigned value used to reload the BCNT field once the last array in the 2nd dimension is transferred. This field is only used for A-synchronized transfers. In this case, the EDMA3CC decrements the BCNT value by 1 on each TR submission. When BCNT reaches 0, the EDMA3CC decrements CCNT and uses the BCNTRLD value to reinitialize the BCNT value.

For AB-synchronized transfers, the EDMA3CC submits the BCNT in the TR and the EDMA3TC decrements BCNT appropriately. For AB-synchronized transfers, BCNTRLD is not used.

5.2.3.2.8 Source B Index (SRCBIDX)

SRCBIDX is a 16-bit signed value (2s complement) used for source address modification between each array in the 2nd dimension. Valid values for SRCBIDX are between –32 768 and 32 767. It provides a byte address offset from the beginning of the source array to the beginning of the next source array. It applies to both A-synchronized and AB-synchronized transfers. Some examples:

- SRCBIDX = 0000h (0): no address offset from the beginning of an array to the beginning of the next array. All arrays are fixed to the same beginning address.
- SRCBIDX = 0003h (+3): the address offset from the beginning of an array to the beginning of the next array in a frame is 3 bytes. For example, if the current array begins at address 1000h, the next array begins at 1003h.
- SRCBIDX = FFFFh (–1): the address offset from the beginning of an array to the beginning of the next array in a frame is –1 byte. For example, if the current array begins at address 5054h, the next array begins at 5053h.

5.2.3.2.9 Destination B Index (DSTBIDX)

DSTBIDX is a 16-bit signed value (2s complement) used for destination address modification between each array in the 2nd dimension. Valid values for DSTBIDX are between –32 768 and 32 767. It provides a byte address offset from the beginning of the destination array to the beginning of the next destination array within the current frame. It applies to both A-synchronized and AB-synchronized transfers. See SRCBIDX for examples.

5.2.3.2.10 Source C Index (SRCCIDX)

SRCCIDX is a 16-bit signed value (2s complement) used for source address modification in the 3rd dimension. Valid values for SRCCIDX are between –32 768 and 32 767. It provides a byte address offset from the beginning of the current array (pointed to by SRC address) to the beginning of the first source array in the next frame. It applies to both A-synchronized and AB-synchronized transfers. Note that when SRCCIDX is applied, the current array in an A-synchronized transfer is the last array in the frame (Figure 5-5), while the current array in an AB-synchronized transfer is the first array in the frame (Figure 5-6).

5.2.3.2.11 Destination C Index (DSTCIDX)

DSTCIDX is a 16-bit signed value (2s complement) used for destination address modification in the 3rd dimension. Valid values are between –32 768 and 32 767. It provides a byte address offset from the beginning of the current array (pointed to by DST address) to the beginning of the first destination array TR in the next frame. It applies to both A-synchronized and AB-synchronized transfers. Note that when DSTCIDX is applied, the current array in an A-synchronized transfer is the last array in the frame (Figure 5-5), while the current array in a AB-synchronized transfer is the first array in the frame (Figure 5-6).

5.2.3.2.12 Link Address (LINK)

The EDMA3CC provides a mechanism, called linking, to reload the current PaRAM set upon its natural termination (that is, after the count fields are decremented to 0) with a new PaRAM set. The 16-bit parameter LINK specifies the byte address offset in the PaRAM from which the EDMA3CC loads/reloads the next PaRAM set during linking.

You must program the link address to point to a valid aligned 32-byte PaRAM set. The 5 LSBs of the LINK field should be cleared to 0.

The EDMA3CC ignores the upper 2 bits of the LINK entry, allowing the programmer the flexibility of programming the link address as either an absolute/literal byte address or use the PaRAM-base-relative offset address. Therefore, if you make use of the literal address with a range from 4000h to 7FFFh, it will be treated as a PaRAM-base-relative value of 0000h to 3FFFh.

You should make sure to program the LINK field correctly, so that link update is requested from a PaRAM address that falls in the range of the available PaRAM addresses on the device.

A LINK value of FFFFh is referred to as a NULL link that should cause the EDMA3CC to perform an internal write of 0 to all entries of the current PaRAM set, except for the LINK field that is set to FFFFh. Also, see Section 5.2.5 for details on terminating a transfer.

5.2.3.3 Null PaRAM Set

A null PaRAM set is defined as a PaRAM set where all count fields (ACNT, BCNT, and CCNT) are cleared to 0. If a PaRAM set associated with a channel is a NULL set, then when serviced by the EDMA3CC, the bit corresponding to the channel is set in the associated event missed register (EMR, EMRH, or QEMR). This bit remains set in the associated secondary event register (SER, SERH, or QSER). *This implies that any future events on the same channel are ignored by the EDMA3CC and you are required to clear the bit in SER, SERH, or QSER for the channel.* This is considered an error condition, since events are not expected on a channel that is configured as a null transfer. See Section 5.4.1.6.8 and Section 5.4.1.2.1 for more information on the SER and EMR registers, respectively.

5.2.3.4 Dummy PaRAM Set

A dummy PaRAM set is defined as a PaRAM set where at least one of the count fields (ACNT, BCNT, or CCNT) is cleared to 0 and at least one of the count fields is nonzero.

If a PaRAM set associated with a channel is a dummy set, then when serviced by the EDMA3CC, it will not set the bit corresponding to the channel (DMA/QDMA) in the event missed register (EMR, EMRH, or QEMR) and the secondary event register (SER, SERH, or QSER) bit gets cleared similar to a normal transfer. Future events on that channel are serviced. A dummy transfer is a legal transfer of 0 bytes. See Section 5.4.1.6.8 and Section 5.4.1.2.1 for more information on the SER and EMR registers, respectively.

5.2.3.5 Dummy Versus Null Transfer Comparison

There are some differences in the way the EDMA3CC logic treats a dummy versus a null transfer request. A null transfer request is an error condition, but a dummy transfer is a legal transfer of 0 bytes. A null transfer causes an error bit (E*n*) in EMR to get set and the E*n* bit in SER remains set, essentially preventing any further transfers on that channel without clearing the associated error registers.

Table 5-4 summarizes the conditions and effects of null and dummy transfer requests.

Table 5-4. Dummy and Null Transfer Request

Feature	Null TR	Dummy TR
EMR/EMRH/QEMR is set	Yes	No
SER/SERH/QSER remains set	Yes	No
Link update (STATIC = 0 in OPT)	Yes	Yes
QER is set	Yes	Yes
IPR/IPRH CER/CERH is set using early completion	Yes	Yes

5.2.3.6 Parameter Set Updates

When a TR is submitted for a given DMA/QDMA channel and its corresponding PaRAM set, the EDMA3CC is responsible for updating the PaRAM set in anticipation of the next trigger event. For events that are not final, this includes address and count updates; for final events, this includes the link update.

The specific PaRAM set entries that are updated depend on the channel's synchronization type (A-synchronized or B-synchronized) and the current state of the PaRAM set. A B-update refers to the decrementing of BCNT in the case of A-synchronized transfers after the submission of successive TRs. A C-update refers to the decrementing of CCNT in the case of A-synchronized transfers after BCNT TRs for ACNT byte transfers have submitted. For AB-synchronized transfers, a C-update refers to the decrementing of CCNT after submission of every transfer request.

See Table 5-5 for details and conditions on the parameter updates. A link update occurs when the PaRAM set is exhausted, as described in Section 5.2.3.7.

After the TR is read from the PaRAM (and is in process of being submitted to EDMA3TC), the following fields are updated if needed:

- A-synchronized: BCNT, CCNT, SRC, DST.
- AB-synchronized: CCNT, SRC, DST.

The following fields are not updated (except for during linking, where all fields are overwritten by the link PaRAM set):

- A-synchronized: ACNT, BCNTRLD, SRCBIDX, DSTBIDX, SRCCIDX, DSTCIDX, OPT, LINK.
- AB-synchronized: ACNT, BCNT, BCNTRLD, SRCBIDX, DSTBIDX, SRCCIDX, DSTCIDX, OPT, LINK.

Note that PaRAM updates only pertain to the information that is needed to properly submit the next transfer request to the EDMA3TC. Updates that occur while data is moved within a transfer request are tracked within the transfer controller, and is detailed in Section 5.2.12. For A-synchronized transfers, the EDMA3CC always submits a TRP for ACNT bytes (BCNT = 1 and CCNT = 1). For AB-synchronized transfers, the EDMA3CC always submits a TRP for ACNT bytes of BCNT arrays (CCNT = 1). The EDMA3TC is responsible for updating source and destination addresses within the array based on ACNT and FWID (in OPT). For AB-synchronized transfers, the EDMA3TC is also responsible to update source and destination addresses between arrays based on SRCBIDX and DSTBIDX.

Table 5-5 shows the details of parameter updates that occur within EDMA3CC for A-synchronized and AB-synchronized transfers.

Table 5-5. Parameter Updates in EDMA3CC (for Non-Null, Non-Dummy PaRAM Set)

	A-Synchronized Transfer			AB-Synchronized Transfer		
	B-Update	C-Update	Link Update	B-Update	C-Update	Link Update
Condition:	BCNT > 1	BCNT == 1 && CCNT > 1	BCNT == 1 && CCNT == 1	N/A	CCNT > 1	CCNT == 1
SRC	+= SRCBIDX	+= SRCCIDX	= Link.SRC	in EDMA3TC	+= SRCCIDX	= Link.SRC
DST	+= DSTBIDX	+= DSTCIDX	= Link.DST	in EDMA3TC	+= DSTCIDX	= Link.DST
ACNT	None	None	= Link.ACNT	None	None	= Link.ACNT
BCNT	–= 1	= BCNTRLD	= Link.BCNT	in EDMA3TC	N/A	= Link.BCNT
CCNT	None	–= 1	= Link.CCNT	in EDMA3TC	–=1	= Link.CCNT
SRCBIDX	None	None	= Link.SRCBIDX	in EDMA3TC	None	= Link.SRCBIDX
DSTBIDX	None	None	= Link.DSTBIDX	None	None	= Link.DSTBIDX
SRCCIDX	None	None	= Link.SRCBIDX	in EDMA3TC	None	= Link.SRCBIDX
DSTCIDX	None	None	= Link.DSTBIDX	None	None	= Link.DSTBIDX
LINK	None	None	= Link.LINK	None	None	= Link.LINK
BCNTRLD	None	None	= Link.BCNTRLD	None	None	= Link.BCNTRLD
OPT[1]	None	None	= LINK.OPT	None	None	= LINK.OPT

(1) In all cases, no updates occur if OPT.STATIC == 1 for the current PaRAM set.

NOTE: The EDMA3CC includes no special hardware to detect when an indexed address update calculation overflows/underflows. The address update will wrap across boundaries as programmed by the user. You should ensure that no transfer is allowed to cross internal port boundaries between peripherals. A single TR must target a single source/destination slave endpoint.

5.2.3.7 Linking Transfers

The EDMA3CC provides a mechanism known as linking, which allows the entire PaRAM set to be reloaded from a location within the PaRAM memory map (for both DMA and QDMA channels). Linking is especially useful for maintaining ping-pong buffers, circular buffering, and repetitive/continuous transfers with no CPU intervention. Upon completion of a transfer, the current transfer parameters are reloaded with the parameter set pointed that the 16-bit link address field of the current parameter set points to. Linking only occurs when the STATIC bit in OPT is cleared.

NOTE: You should always link a transfer (EDMA3 or QDMA) to another useful transfer. If you must terminate a transfer, then you should link the transfer to a NULL parameter set. See Section 5.2.3.3.

The link update occurs after the current PaRAM set event parameters have been exhausted. An event's parameters are exhausted when the EDMA3 channel controller has submitted all of the transfers that are associated with the PaRAM set.

A link update occurs for null and dummy transfers depending on the state of the STATIC bit in OPT and the LINK field. In both cases (null or dummy), if the value of LINK is FFFFh, then a null PaRAM set (with all 0s and LINK set to FFFFh) is written to the current PaRAM set. Similarly, if LINK is set to a value other than FFFFh, then the appropriate PaRAM location that LINK points to is copied to the current PaRAM set.

Once the channel completion conditions are met for an event, the transfer parameters that are located at the link address are loaded into the current DMA or QDMA channel's associated parameter set. This indicates that the EDMA3CC reads the entire set (eight words) from the PaRAM set specified by LINK and writes all eight words to the PaRAM set that is associated with the current channel. Figure 5-9 shows an example of a linked transfer.

Any PaRAM set in the PaRAM can be used as a link/reload parameter set. The PaRAM sets associated with peripheral synchronization events (see Section 5.2.6) should only be used for linking if the corresponding events are disabled.

If a PaRAM set location is defined as a QDMA channel PaRAM set (by QCHMAP*n*), then copying the link PaRAM set into the current QDMA channel PaRAM set is recognized as a trigger event. It is latched in QER because a write to the trigger word was performed. You can use this feature to create a linked list of transfers using a single QDMA channel and multiple PaRAM sets. See Section 5.2.4.2.

Linking to itself replicates the behavior of auto-initialization, thus facilitating the use of circular buffering and repetitive transfers. After an EDMA3 channel exhausts its current PaRAM set, it reloads all of the parameter set entries from another PaRAM set, which is initialized with values that are identical to the original PaRAM set. Figure 5-9 shows an example of a linked to self transfer. Here, the PaRAM set 511 has the link field pointing to the address of parameter set 511 (linked to self).

NOTE: If the STATIC bit in OPT is set for a PaRAM set, then link updates are not performed.

5.2.3.8 Constant Addressing Mode Transfers/Alignment Issues

If either SAM or DAM is set (constant addressing mode), then the source or destination address must be aligned to a 256-bit aligned address, respectively, and the corresponding BIDX should be an even multiple of 32 bytes (256 bits). The EDMA3CC does not recognize errors here, but the EDMA3TC asserts an error if this is not true. See Section 5.2.12.3.

NOTE: The constant addressing (CONST) mode has limited applicability. The EDMA3 should be configured for the constant addressing mode (SAM/DAM = 1) only if the transfer source or destination (on-chip memory, off-chip memory controllers, slave peripherals) support the constant addressing mode. See the device-specific data manual and/or peripheral user`s guide to verify if the constant addressing mode is supported. If the constant addressing mode is not supported, the similar logical transfer can be achieved using the increment (INCR) mode (SAM/DAM =0) by appropriately programming the count and indices values.

5.2.3.9 Element Size

The EDMA3 controller does not use element-size and element-indexing. Instead, all transfers are defined in terms of all three dimensions: ACNT, BCNT, and CCNT. An element-indexed transfer is logically achieved by programming ACNT to the size of the element and BCNT to the number of elements that need to be transferred. For example, if you have 16-bit audio data and 256 audio samples that must be transferred to a serial port, you can only do this by programming the ACNT = 2 (2 bytes) and BCNT = 256.

Figure 5-9. Linked Transfer

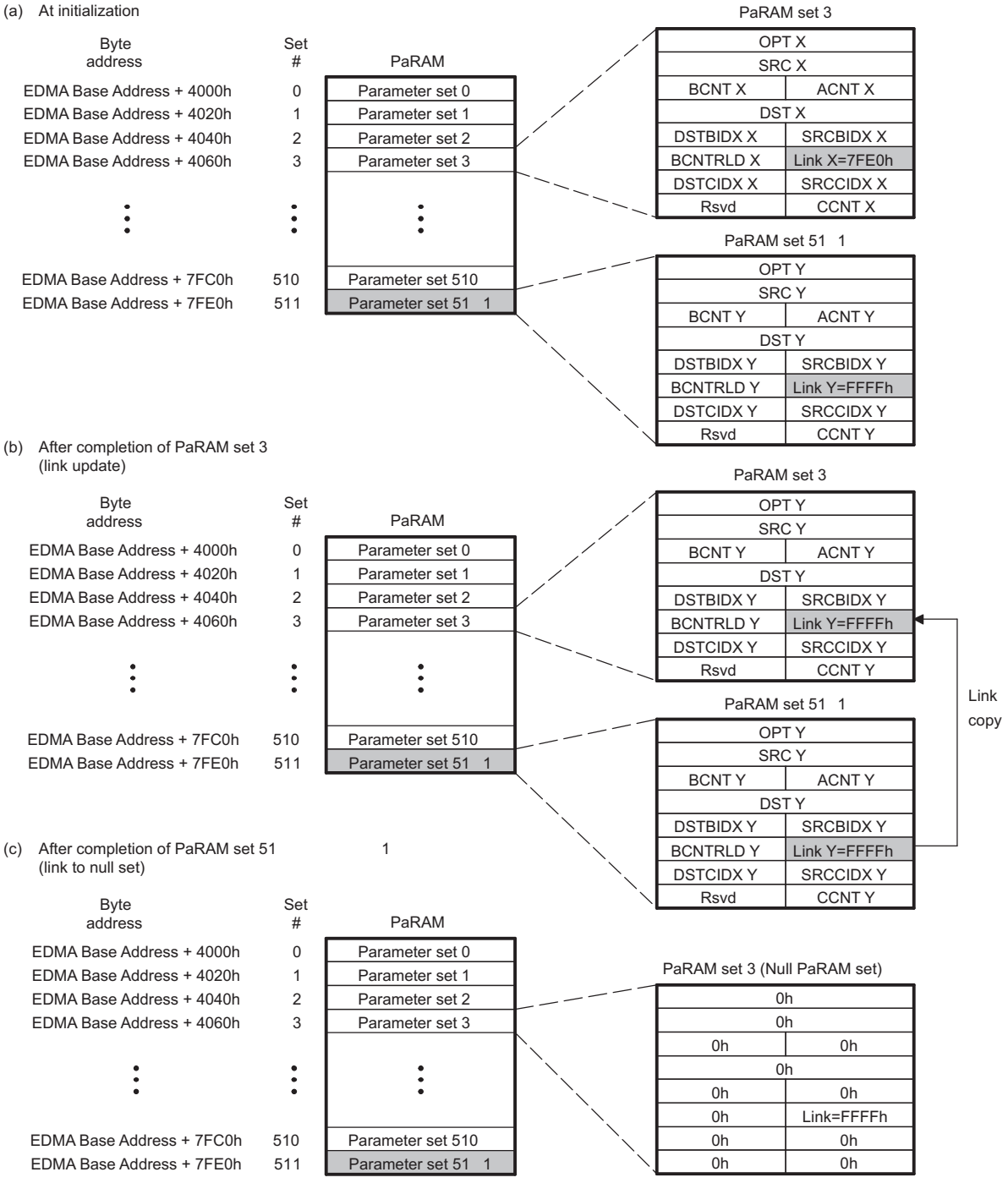
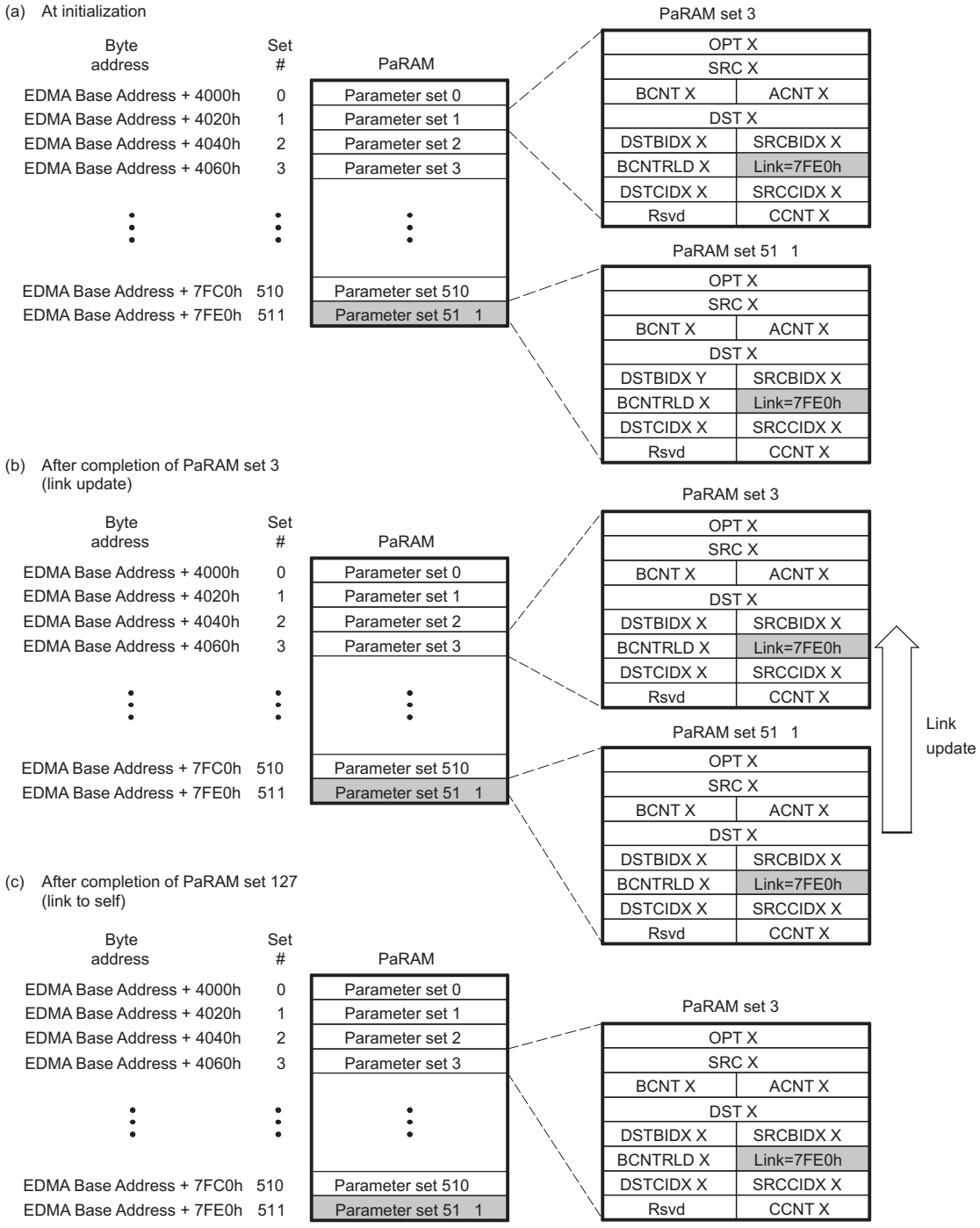


Figure 5-10. Link-to-Self Transfer



5.2.4 Initiating a DMA Transfer

There are multiple ways to initiate a programmed data transfer using the EDMA3 channel controller. Transfers on DMA channels are initiated by three sources.

They are listed as follows:

- **Event-triggered transfer request** (this is the more typical usage of EDMA3): A peripheral, system, or externally-generated event triggers a transfer request.
- **Manually-triggered transfer request:**The CPU to manually triggers a transfer by writing a 1 to the corresponding bit in the event set register (ESR/ESRH).
- **Chain-triggered transfer request:** A transfer is triggered on the completion of another transfer or sub-transfer.

Transfers on QDMA channels are initiated by two sources. They are as follows:

- **Auto-triggered transfer request:** Writing to the programmed trigger word triggers a transfer.
- **Link-triggered transfer requests:** Writing to the trigger word triggers the transfer when linking occurs.

5.2.4.1 DMA Channel

5.2.4.1.1 Event-Triggered Transfer Request

When an event is asserted from a peripheral or device pins, it gets latched in the corresponding bit of the event register (ER.E*n* = 1). For peripheral event to DMA event mapping, see the device-specific data manual. If the corresponding event in the event enable register (EER) is enabled (EER.E*n* = 1), then the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

If the PaRAM set is valid (not a NULL set), then a transfer request packet (TRP) is submitted to the EDMA3TC and the E*n* bit in ER is cleared. At this point, a new event can be safely received by the EDMA3CC.

If the PaRAM set associated with the channel is a NULL set (see Section 5.2.3.3), then no transfer request (TR) is submitted and the corresponding E*n* bit in ER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.E*n* = 1) to indicate that the event was discarded due to a null TR being serviced. Good programming practices should include cleaning the event missed error before re-triggering the DMA channel.

When an event is received, the corresponding event bit in the event register is set (ER.E*n* = 1), regardless of the state of EER.E*n*. If the event is disabled when an external event is received (ER.E*n* = 1 and EER.E*n* = 0), the ER.E*n* bit remains set. If the event is subsequently enabled (EER.E*n* = 1), then the pending event is processed by the EDMA3CC and the TR is processed/submitted, after which the ER.E*n* bit is cleared.

If an event is being processed (prioritized or is in the event queue) and another sync event is received for the same channel prior to the original being cleared (ER.E*n* != 0), then the second event is registered as a missed event in the corresponding bit of the event missed register (EMR.E*n* = 1).

The EDMA section in the device-specific data manual gives the table of the synchronization events associated with each of the programmable DMA channels in the device. It also provides the mapping of events to the EDMA event crossbar. See the device-specific data manual to determine the event to channel mapping.

5.2.4.1.2 Manually-Triggered Transfer Request

The CPU or any EDMA programmer initiates a DMA transfer by writing to the event set register (ESR). Writing a 1 to an event bit in the ESR results in the event being prioritized/queued in the appropriate event queue, regardless of the state of the EER.E*n* bit. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If the PaRAM set associated with the channel is a NULL set (see Section 5.2.3.3), then no transfer request (TR) is submitted and the corresponding E*n* bit in ER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.E*n* = 1) to indicate that the event was discarded due to a null TR being serviced. Good programming practices should include clearing the event missed error before re-triggering the DMA channel.

If an event is being processed (prioritized or is in the event queue) and the same channel is manually set by a write to the corresponding channel bit of the event set register (ESR.E*n* = 1) prior to the original being cleared (ESR.E*n* = 0), then the second event is registered as a missed event in the corresponding bit of the event missed register (EMR.E*n* = 1).

5.2.4.1.3 Chain-Triggered Transfer Request

Chaining is a mechanism by which the completion of one transfer automatically sets the event for another channel. When a chained completion code is detected, the value of which is dictated by the transfer completion code (TCC[5:0] in OPT of the PaRAM set associated with the channel), it results in the corresponding bit in the chained event register (CER) to be set (CER.E[TCC] = 1).

Once a bit is set in CER, the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If the PaRAM set associated with the channel is a NULL set (see Section 5.2.3.3), then no transfer request (TR) is submitted and the corresponding E*n* bit in CER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.E*n* = 1) to indicate that the event was discarded due to a null TR being serviced. In this case, the error condition must be cleared by you before the DMA channel can be re-triggered. Good programming practices might include clearing the event missed error before re-triggering the DMA channel.

If a chaining event is being processed (prioritized or queued) and another chained event is received for the same channel prior to the original being cleared (CER.E*n* != 0), then the second chained event is registered as a missed event in the corresponding channel bit of the event missed register (EMR.E*n* = 1).

NOTE: Chained event registers, event registers, and event set registers operate independently. An event (E*n*) can be triggered by any of the trigger sources (event-triggered, manually-triggered, or chain-triggered).

5.2.4.2 QDMA Channels

5.2.4.2.1 Auto-triggered and Link-Triggered Transfer Request

NOTE: If OPT, SRC, or DST is the trigger word for a QDMA transfer then it is required to do a 32-bit access to that field.

QDMA-based transfer requests are issued when a QDMA event gets latched in the QDMA event register (QER.E*n* = 1). A bit corresponding to a QDMA channel is set in the QDMA event register (QER) when the following occurs:

- A CPU (or any EDMA3 programmer) write occurs to a PaRAM address that is defined as a QDMA channel trigger word (programmed in the QDMA channel mapping register (QCHMAP*n*)) for the particular QDMA channel and the QDMA channel is enabled via the QDMA event enable register (QEER.E*n* = 1).
- EDMA3CC performs a link update on a PaRAM set address that is configured as a QDMA channel (matches QCHMAP*n* settings) and the corresponding channel is enabled via the QDMA event enable register (QEER.E*n* = 1).

Once a bit is set in QER, the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If a bit is already set in QER (QER.E*n* = 1) and a second QDMA event for the same QDMA channel occurs prior to the original being cleared, the second QDMA event gets captured in the QDMA event miss register (QEMR.E*n* = 1).

5.2.4.3 Comparison Between DMA and QDMA Channels

The primary difference between DMA and QDMA channels is the event/channel synchronization. QDMA events are either auto-triggered or link triggered. auto-triggering allows QDMA channels to be triggered by CPU(s) with a minimum number of linear writes to PaRAM. Link triggering allows a linked list of transfers to be executed, using a single QDMA PaRAM set and multiple link PaRAM sets.

A QDMA transfer is triggered when a CPU (or other EDMA3 programmer) writes to the trigger word of the QDMA channel parameter set (auto-triggered) or when the EDMA3CC performs a link update on a PaRAM set that has been mapped to a QDMA channel (link triggered). Note that for CPU triggered (manually triggered) DMA channels, in addition to writing to the PaRAM set, it is required to write to the event set register (ESR) to kick-off the transfer.

QDMA channels are typically for cases where a single event will accomplish a complete transfer since the CPU (or EDMA3 programmer) must reprogram some portion of the QDMA PaRAM set in order to re-trigger the channel. In other words, QDMA transfers are programmed with BCNT = CCNT = 1 for A-synchronized transfers, and CCNT = 1 for AB-synchronized transfers.

Additionally, since linking is also supported (if STATIC = 0 in OPT) for QDMA transfers, it allows you to initiate a linked list of QDMAs, so when EDMA3CC copies over a link PaRAM set (including the write to the trigger word), the current PaRAM set mapped to the QDMA channel will automatically be recognized as a valid QDMA event and initiate another set of transfers as specified by the linked set.

5.2.5 Completion of a DMA Transfer

A parameter set for a given channel is complete when the required number of transfer requests is submitted (based on receiving the number of synchronization events). The expected number of TRs for a non-null/non-dummy transfer is shown in Table 5-6 for both synchronization types along with state of the PaRAM set prior to the final TR being submitted. When the counts (BCNT and/or CCNT) are this value, the next TR results in a:

- Final chaining or interrupt codes to be sent by the transfer controllers (instead of intermediate).
- Link updates (linking to either null or another valid link set).

Table 5-6. Expected Number of Transfers for Non-Null Transfer

Sync Mode	Counts at time 0	Total # Transfers	Counts prior to final TR
A-synchronized	ACNT BCNT CCNT	(BCNT × CCNT) TRs of ACNT bytes each	BCNT == 1 && CCNT == 1
AB-synchronized	ACNT BCNT CCNT	CCNT TRs for ACNT × BCNT bytes each	CCNT == 1

You must program the PaRAM OPT field with a specific transfer completion code (TCC) along with the other OPT fields (TCCHEN, TCINTEN, ITCCHEN, and ITCINTEN bits) to indicate whether the completion code is to be used for generating a chained event or/and for generating an interrupt upon completion of a transfer.

The specific TCC value (6-bit binary value) programmed dictates which of the 64-bits in the chain event register (CER[TCC]) and/or interrupt pending register (IPR[TCC]) is set.

You can also selectively program whether the transfer controller sends back completion codes on completion of the final transfer request (TR) of a parameter set (TCCHEN or TCINTEN), for all but the final transfer request (TR) of a parameter set (ITCCHEN or ITCINTEN), or for all TRs of a parameter set (both). See Section 5.2.8 for details on chaining (intermediate/final chaining) and Section 5.2.9 for details on intermediate/final interrupt completion.

A completion detection interface exists between the EDMA3 channel controller and transfer controller(s). This interface sends back information from the transfer controller to the channel controller to indicate that a specific transfer is completed. Completion of a transfer is used for generating chained events and/or generating interrupts to the CPU(s).

All DMA/QDMA PaRAM sets must also specify a link address value. For repetitive transfers such as ping-pong buffers, the link address value should point to another predefined PaRAM set. Alternatively, a non-repetitive transfer should set the link address value to the null link value. The null link value is defined as FFFFh. See Section 5.2.3.7 for more details.

NOTE: Any incoming events that are mapped to a null PaRAM set results in an error condition. The error condition should be cleared before the corresponding channel is used again. See Section 5.2.3.5.

There are three ways the EDMA3CC gets updated/informed about a transfer completion: normal completion, early completion, and dummy/null completion. This applies to both chained events and completion interrupt generation.

5.2.5.1 Normal Completion

In normal completion mode (TCCMODE = 0 in OPT), the transfer or sub-transfer is considered to be complete when the EDMA3 channel controller receives the completion codes from the EDMA3 transfer controller. In this mode, the completion code to the channel controller is posted by the transfer controller after it receives a signal from the destination peripheral. Normal completion is typically used to generate an interrupt to inform the CPU that a set of data is ready for processing.

5.2.5.2 Early Completion

In early completion mode (TCCMODE = 1 in OPT), the transfer is considered to be complete when the EDMA3 channel controller submits the transfer request (TR) to the EDMA3 transfer controller. In this mode, the channel controller generates the completion code internally. Early completion is typically useful for chaining, as it allows subsequent transfers to be chained-triggered while the previous transfer is still in progress within the transfer controller, maximizing the overall throughput of the set of the transfers.

5.2.5.3 Dummy or Null Completion

This is a variation of early completion. Dummy or null completion is associated with a dummy set (Section 5.2.3.4) or null set (Section 5.2.3.3). In both cases, the EDMA3 channel controller does not submit the associated transfer request to the EDMA3 transfer controller(s). However, if the set (dummy/null) has the OPT field programmed to return completion code (intermediate/final interrupt/chaining completion), then it will set the appropriate bits in the interrupt pending registers (IPR/IPRH) or chained event register (CER/CERH). The internal early completion path is used by the channel controller to return the completion codes internally (that is, EDMA3CC generates the completion code).

5.2.6 Event, Channel, and PaRAM Mapping

Several of the 64 DMA channels are tied to a specific hardware event, thus allowing events from device peripherals or external hardware to trigger transfers. A DMA channel typically requests a data transfer when it receives its event (apart from manually-triggered, chain-triggered, and other transfers). The amount of data transferred per synchronization event depends on the channel's configuration (ACNT, BCNT, CCNT, etc.) and the synchronization type (A-synchronized or AB-synchronized).

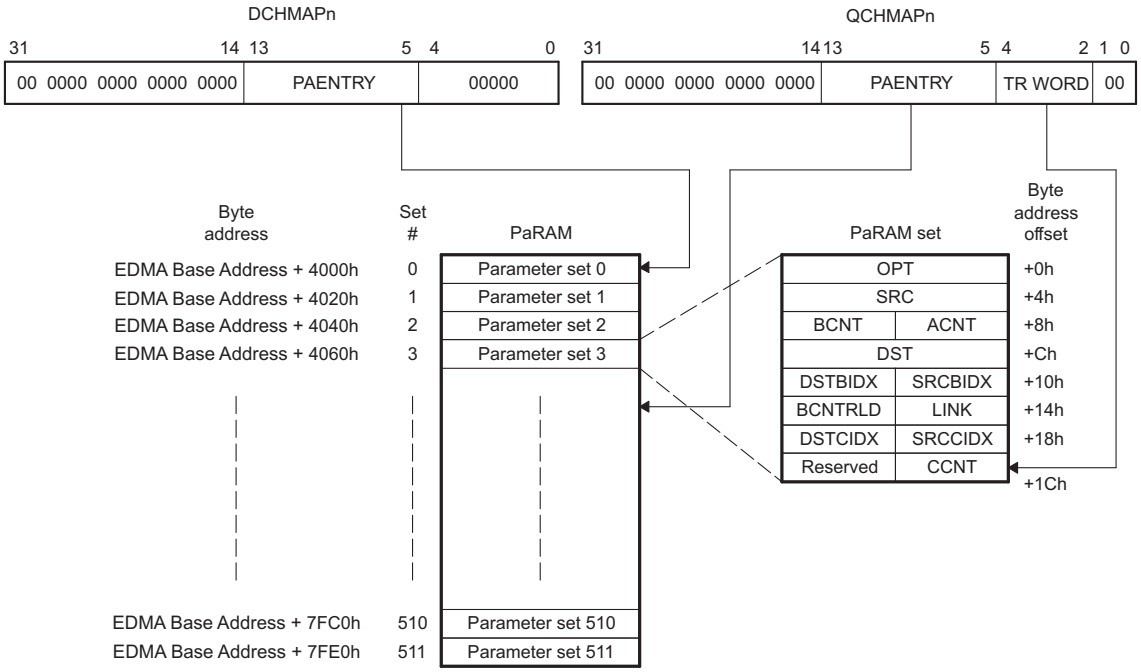
The association of an event to a channel is fixed, each DMA channel has one specific event associated with it. The device-specific data manual provides the synchronization events that are associated with each of the programmable DMA channels.

In an application, if a channel does not use the associated synchronization event or if it does not have an associated synchronization event (unused), that channel can be used for manually-triggered or chained-triggered transfers, for linking/reloading, or as a QDMA channel.

5.2.6.1 DMA Channel to PaRAM Mapping

The mapping between the DMA channel numbers and the PaRAM sets is programmable (see Table 5-1). The DMA channel mapping registers (DCHMAPn) in the EDMA3CC provide programmability that allows the DMA channels to be mapped to any of the PaRAM sets in the PaRAM memory map. Figure 5-11 illustrates the use of DCHMAP. There is one DCHMAP register per channel.

Figure 5-11. DMA Channel and QDMA Channel to PaRAM Mapping

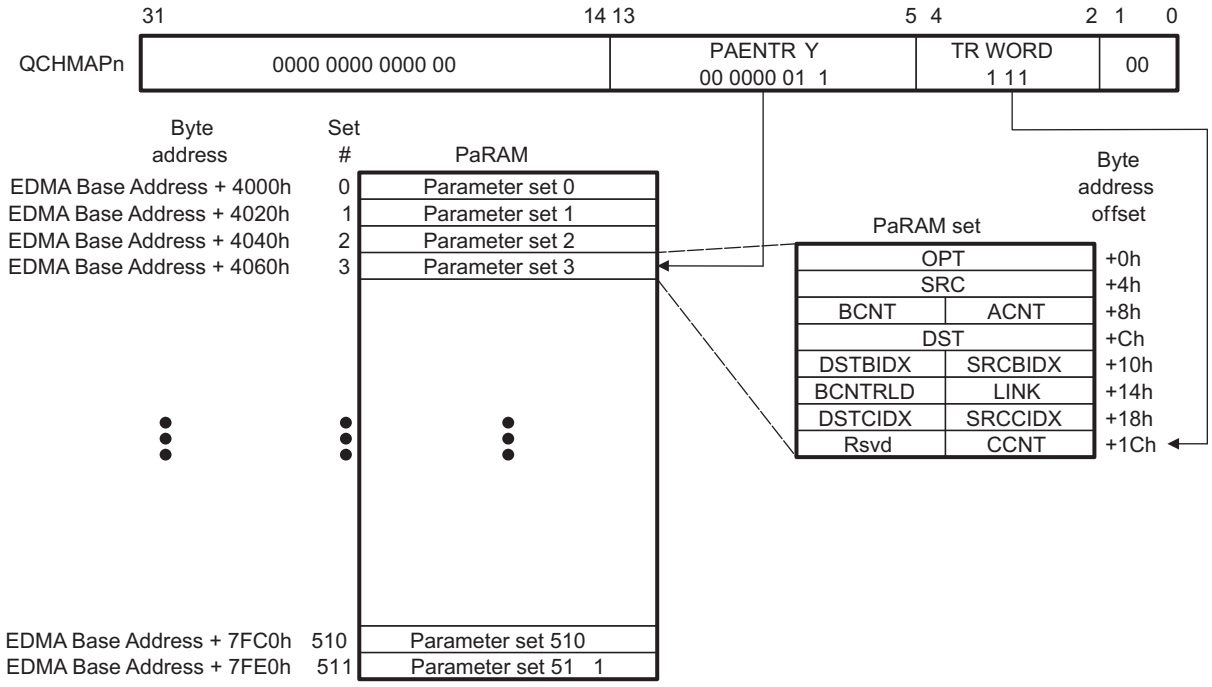


5.2.6.2 QDMA Channel to PaRAM Mapping

The mapping between the QDMA channels and the PaRAM sets is programmable. The QDMA channel mapping register (QCHMAP) in the EDMA3CC allows you to map the QDMA channels to any of the PaRAM sets in the PaRAM memory map. Figure 5-12 illustrates the use of QCHMAP.

Additionally, QCHMAP allows you to program the trigger word in the PaRAM set for the QDMA channel. A trigger word is one of the eight words in the PaRAM set. For a QDMA transfer to occur, a valid TR synchronization event for EDMA3CC is a write to the trigger word in the PaRAM set pointed to by QCHMAP for a particular QDMA channel. By default, QDMA channels are mapped to PaRAM set 0. You must appropriately re-map PaRAM set 0 before you use it.

Figure 5-12. QDMA Channel to PaRAM Mapping



5.2.7 EDMA3 Channel Controller Regions

The EDMA3 channel controller divides its address space into eight regions. Individual channel resources are assigned to a specific region, where each region is typically assigned to a specific EDMA programmer.

You can design the application software to use regions or to ignore them altogether. You can use active memory protection in conjunction with regions so that only a specific EDMA programmer (for example, privilege identification) or privilege level (for example, user vs. supervisor) is allowed access to a given region, and thus to a given DMA or QDMA channel. This allows robust system-level DMA code where each EDMA programmer only modifies the state of the assigned resources. Memory protection is described in Section 5.2.10.

5.2.7.1 Region Overview

The EDMA3 channel controller memory-mapped registers are divided in three main categories:

1. Global registers
2. Global region channel registers
3. Shadow region channel registers

The global registers are located at a single/fixed location in the EDMA3CC memory map. These registers control EDMA3 resource mapping and provide debug visibility and error tracking information. See the device-specific data manual for the EDMA3CC memory map.

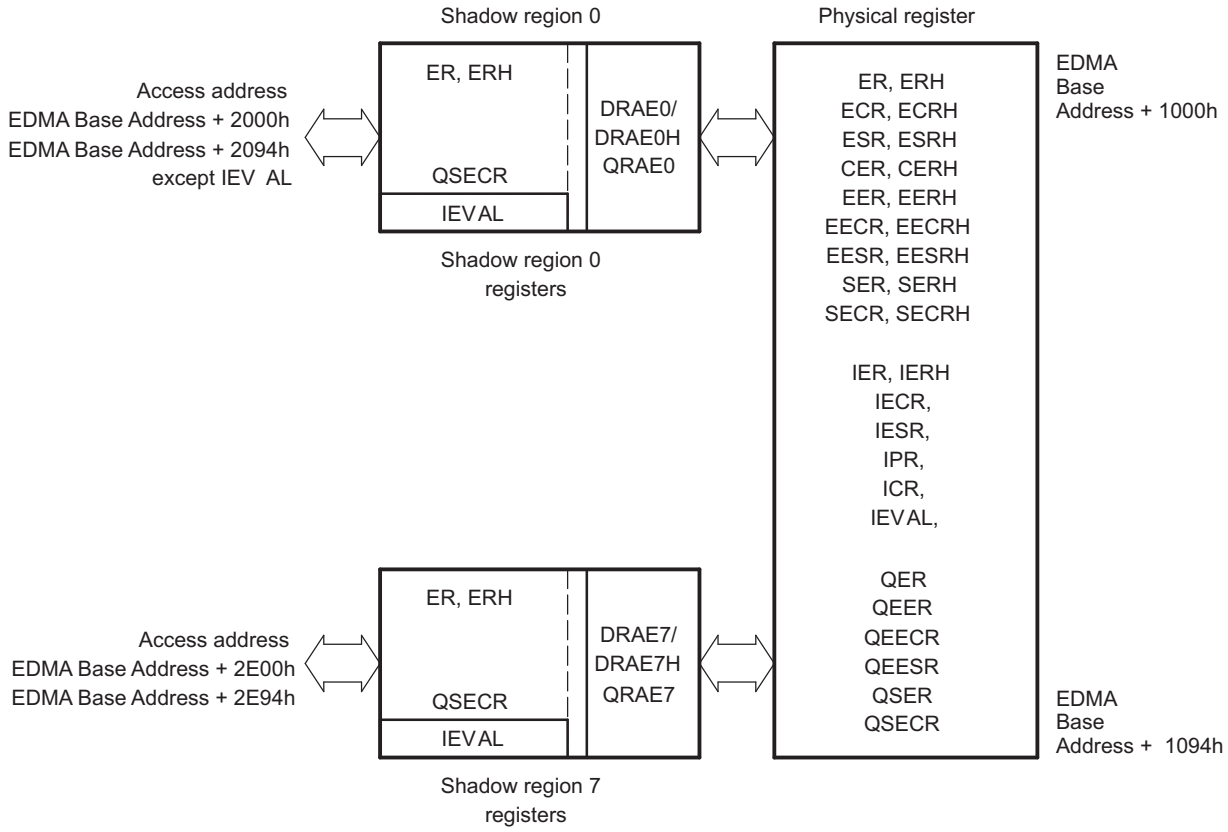
The channel registers (including DMA, QDMA, and interrupt registers) are accessible via the global channel region address range, or in the shadow *n* channel region address range(s). For example, the event enable register (EER) is visible at the global address of EDMA Base Address + 1020h or region addresses of EDMA Base Address + 2020h for region 0, EDMA Base Address + 2220h for region 1, … EDMA Base Address + 2E20h for region 7.

The DMA region access enable registers (DRAE*m*) and the QDMA region access enable registers (QRAE*n*) control the underlying control register bits that are accessible via the shadow region address space (except for IEVAL*n*). Table 5-7 lists the registers in the shadow region memory map. See the EDMA3CC memory map (Table 5-19) for the complete global and shadow region memory maps. Figure 5-13 illustrates the conceptual view of the regions.

Table 5-7. Shadow Region Registers

DRAE*m*	DRAEH*m*	QRAE*n*
ER	ERH	QER
ECR	ECRH	QEER
ESR	ESRH	QEECR
CER	CERH	QEESR
EER	EERH	
EECR	EECRH	
EESR	EESRH	
SER	SERH	
SECR	SECRH	
IER	IERH	
IECR	IECRH	
IESR	IESRH	
IPR	IPRH	
ICR	ICRH	
Register not affected by DRAE\DRAEH		
IEVAL		

Figure 5-13. Shadow Region Registers



5.2.7.2 Channel Controller Regions

There are eight EDMA3 shadow regions (and associated memory maps). Associated with each shadow region are a set of registers defining which channels and interrupt completion codes belong to that region. These registers are user-programmed per region to assign ownership of the DMA/QDMA channels to a region.

- DRAE*m* and DRAEH*m*: One register pair exists for each of the shadow regions. The number of bits in each register pair matches the number of DMA channels (64 DMA channels). These registers need to be programmed to assign ownership of DMA channels and interrupt (or TCC codes) to the respective region. Accesses to DMA and interrupt registers via the shadow region address view are filtered through the DRAE/DRAEH pair. A value of 1 in the corresponding DRAE(H) bit implies that the corresponding DMA/interrupt channel is accessible; a value of 0 in the corresponding DRAE(H) bit forces writes to be discarded and returns a value of 0 for reads.
- QRAE*n*: One register exists for every region. The number of bits in each register matches the number of QDMA channels (4 QDMA channels). These registers must be programmed to assign ownership of QDMA channels to the respective region. To enable a channel in a shadow region using shadow region 0 QEER, the respective bit in QRAE must be set or writing into QEESR will not have the desired effect.
- MPPA*n* and MPPAG: One register exists for every region. This register defines the privilege level, requestor, and types of accesses allowed to a region's memory-mapped registers.

It is typical for an application to have a unique assignment of QDMA/DMA channels (and, therefore, a given bit position) to a given region.

The use of shadow regions allows for restricted access to EDMA3 resources (DMA channels, QDMA channels, TCC, interrupts) by tasks in a system by setting or clearing bits in the DRAE/ORAE registers. If exclusive access to any given channel / TCC code is required for a region, then only that region's DRAE/ORAE should have the associated bit set.

Example 5-1. Resource Pool Division Across Two Regions

This example illustrates a judicious resource pool division across two regions, assuming region 0 must be allocated 16 DMA channels (0-15) and 1 QDMA channel (0) and 32 TCC codes (0-15 and 48-63). Region 1 needs to be allocated 16 DMA channels (16-32) and the remaining 7 QDMA channels (1-7) and TCC codes (16-47). DRAE should be equal to the OR of the bits that are required for the DMA channels and the TCC codes:

```
Region 0: DRAEH, DRAE = 0xFFFF0000, 0x0000FFFF QRAE = 0x0000001
Region 1: DRAEH, DRAE = 0x0000FFFF, 0xFFFF0000 QRAE = 0x00000FE
```

5.2.7.3 Region Interrupts

In addition to the EDMA3CC global completion interrupt, there is an additional completion interrupt line that is associated with every shadow region. Along with the interrupt enable register (IER), DRAE acts as a secondary interrupt enable for the respective shadow region interrupts. See Section 5.2.9 for more information.

5.2.8 Chaining EDMA3 Channels

The channel chaining capability for the EDMA3 allows the completion of an EDMA3 channel transfer to trigger another EDMA3 channel transfer. The purpose is to allow you the ability to chain several events through one event occurrence.

Chaining is different from linking (Section 5.2.3.7). The EDMA3 link feature reloads the current channel parameter set with the linked parameter set. The EDMA3 chaining feature does not modify or update any channel parameter set; it provides a synchronization event to the chained channel (see Section 5.2.4.1.3 for chain-triggered transfer requests).

Chaining is achieved at either final transfer completion or intermediate transfer completion, or both, of the current channel. Consider a channel *m* (DMA/QDMA) required to chain to channel *n*. Channel number *n* (0-63) needs to be programmed into the TCC bit of channel *m* channel options parameter (OPT) set.

- If final transfer completion chaining (TCCHEN = 1 in OPT) is enabled, the chain-triggered event occurs after the submission of the last transfer request of channel *m* is either submitted or completed (depending on early or normal completion).
- If intermediate transfer completion chaining (ITCCHEN = 1 in OPT) is enabled, the chain-triggered event occurs after every transfer request, except the last of channel *m* is either submitted or completed (depending on early or normal completion).
- If both final and intermediate transfer completion chaining (TCCHEN = 1 and ITCCHEN = 1 in OPT) are enabled, then the chain-trigger event occurs after every transfer request is submitted or completed (depending on early or normal completion).

Table 5-8 illustrates the number of chain event triggers occurring in different synchronized scenarios. Consider channel 31 programmed with ACNT = 3, BCNT = 4, CCNT = 5, and TCC = 30.

Table 5-8. Chain Event Triggers

	(Number of chained event triggers on channel 30)	
Options	**A-Synchronized**	**AB-Synchronized**
TCCHEN = 1, ITCCHEN = 0	1 (Owing to the last TR)	1 (Owing to the last TR)
TCCHEN = 0, ITCCHEN = 1	19 (Owing to all but the last TR)	4 (Owing to all but the last TR)
TCCHEN = 1, ITCCHEN = 1	20 (Owing to a total of 20 TRs)	5 (Owing to a total of 5 TRs)

5.2.9 EDMA3 Interrupts

The EDMA3 interrupts are divided into 2 categories: transfer completion interrupts and error interrupts.

There are nine region interrupts, eight shadow regions and one global region. The transfer completion interrupts are listed in Table 5-9. The transfer completion interrupts and the error interrupts from the transfer controllers are all routed to the DSP and ARM interrupt controllers.

Table 5-9. EDMA3 Transfer Completion Interrupts

Name	Description
EDMA3CC_INT0	EDMA3CC Transfer Completion Interrupt Shadow Region 0
EDMA3CC_INT1	EDMA3CC Transfer Completion Interrupt Shadow Region 1
EDMA3CC_INT2	EDMA3CC Transfer Completion Interrupt Shadow Region 2
EDMA3CC_INT3	EDMA3CC Transfer Completion Interrupt Shadow Region 3
EDMA3CC_INT4	EDMA3CC Transfer Completion Interrupt Shadow Region 4
EDMA3CC_INT5	EDMA3CC Transfer Completion Interrupt Shadow Region 5
EDMA3CC_INT6	EDMA3CC Transfer Completion Interrupt Shadow Region 6
EDMA3CC_INT7	EDMA3CC Transfer Completion Interrupt Shadow Region 7

Table 5-10. EDMA3 Error Interrupts

Name	Description
EDMA3CC_ERRINT	EDMA3CC Error Interrupt
EDMA3CC_MPINT	EDMA3CC Memory Protection Interrupt
EDMA3TC0_ERRINT	TC0 Error Interrupt
EDMA3TC1_ERRINT	TC1 Error Interrupt
EDMA3TC2_ERRINT	TC2 Error Interrupt
EDMA3TC3_ERRINT	TC3 Error Interrupt

5.2.9.1 Transfer Completion Interrupts

The EDMA3CC is responsible for generating transfer completion interrupts to the CPU(s) (and other EDMA3 masters). The EDMA3 generates a single completion interrupt per shadow region, as well as one for the global region on behalf of all 64 channels. The various control registers and bit fields facilitate EDMA3 interrupt generation.

The software architecture should either use the global interrupt or the shadow interrupts, but not both.

The transfer completion code (TCC) value is directly mapped to the bits of the interrupt pending register (IPR/IPRH), as shown in Table 5-11. For example, if TCC = 10 0001b, IPRH[1] is set after transfer completion, and results in interrupt generation to the CPU(s) if the completion interrupt is enabled for the CPU. See Section 5.2.9.1.1 for details on enabling EDMA3 transfer completion interrupts.

When a completion code is returned (as a result of early or normal completions), the corresponding bit in IPR/IPRH is set if transfer completion interrupt (final/intermediate) is enabled in the channel options parameter (OPT) for a PaRAM set associated with the transfer.

You can program the transfer completion code (TCC) to any value for a DMA/QDMA channel. A direct relation between the channel number and the transfer completion code value does not need to exist. This allows multiple channels having the same transfer completion code value to cause a CPU to execute the same interrupt service routine (ISR) for different channels.

If the channel is used in the context of a shadow region and you intend for the shadow region interrupt to be asserted, then ensure that the bit corresponding to the TCC code is enabled in IER/IERH and in the corresponding shadow region's DMA region access registers (DRAE/DRAEH).

Table 5-11. Transfer Complete Code (TCC) to EDMA3CC Interrupt Mapping

TCC Bits in OPT (TCINTEN/ITCINTEN = 1)	IPR Bit Set	TCC Bits in OPT (TCINTEN/ITCINTEN = 1)	IPRH Bit Set[1]
0	IPR0	20h	IPR32/IPRH0
1	IPR1	21h	IPR33/IPRH1
2h	IPR2	22h	IPR34/IPRH2
3h	IPR3	23h	IPR35/IPRH3
4h	IPR4	24h	IPR36/IPRH4
...	...	...	...
1Eh	IPR30	3Eh	IPR62/IPRH30
1Fh	IPR31	3Fh	IPR63/IPRH31

(1) Bit fields IPR[32-63] correspond to bits 0 to 31 in IPRH, respectively.

You can enable Interrupt generation at either final transfer completion or intermediate transfer completion, or both. Consider channel *m* as an example.

- If the final transfer interrupt (TCCINT = 1 in OPT) is enabled, the interrupt occurs after the last transfer request of channel *m* is either submitted or completed (depending on early or normal completion).
- If the intermediate transfer interrupt (ITCCINT = 1 in OPT) is enabled, the interrupt occurs after every transfer request, except the last TR of channel *m* is either submitted or completed (depending on early or normal completion).
- If both final and intermediate transfer completion interrupts (TCCINT = 1, and ITCCINT = 1 in OPT) are enabled, then the interrupt occurs after every transfer request is submitted or completed (depending on early or normal completion).

Table 5-12 shows the number of interrupts that occur in different synchronized scenarios. Consider channel 31, programmed with ACNT = 3, BCNT = 4, CCNT = 5, and TCC = 30.

Table 5-12. Number of Interrupts

Options	A-Synchronized	AB-Synchronized
TCINTEN = 1, ITCINTEN = 0	1 (Last TR)	1 (Last TR)
TCINTEN = 0, ITCINTEN = 1	19 (All but the last TR)	4 (All but the last TR)
TCINTEN = 1, ITCINTEN = 1	20 (All TRs)	5 (All TRs)

5.2.9.1.1 Enabling Transfer Completion Interrupts

For the EDMA3 channel controller to assert a transfer completion to the external environment, the interrupts must be enabled in the EDMA3CC. This is in addition to setting up the TCINTEN and ITCINTEN bits in OPT of the associated PaRAM set.

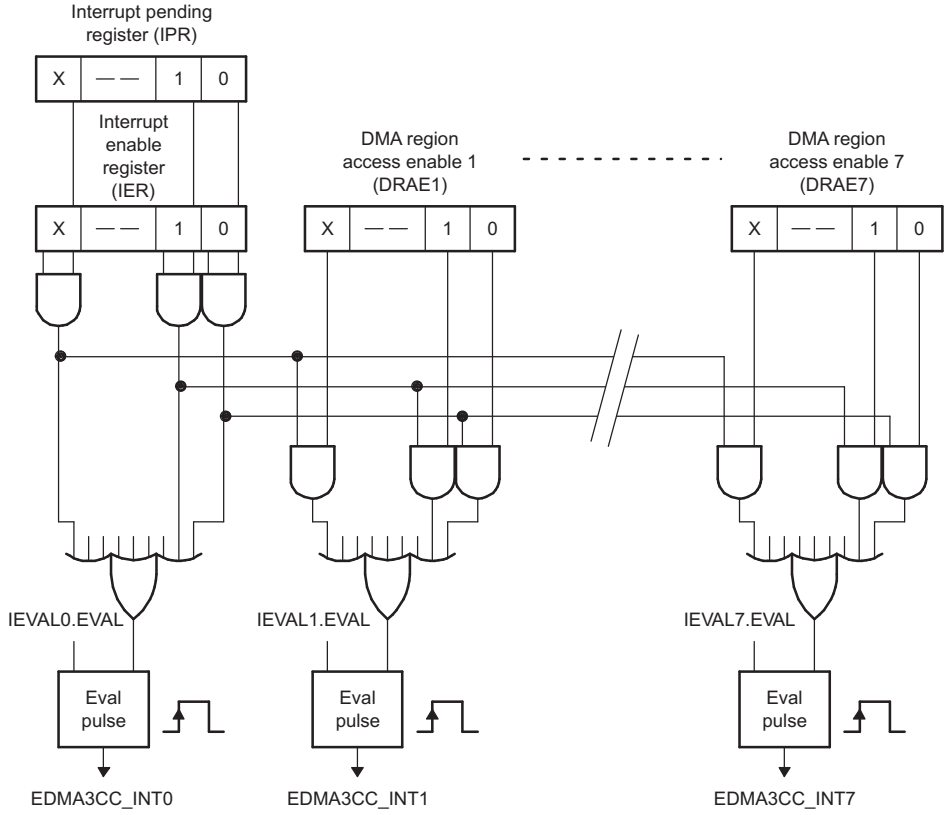
The EDMA3 channel controller has interrupt enable registers (IER/IERH) and each bit location in IER/IERH serves as a primary enable for the corresponding interrupt pending registers (IPR/IPRH).

All of the interrupt registers (IER, IESR, IECR, and IPR) are either manipulated from the global DMA channel region, or by the DMA channel shadow regions. The shadow regions provide a view to the same set of physical registers that are in the global region.

The EDMA3 channel controller has a hierarchical completion interrupt scheme that uses a single set of interrupt pending registers (IPR/IPRH) and single set of interrupt enable registers (IER/IERH). The programmable DMA region access enable registers (DRAE/DRAEH) provides a second level of interrupt masking. The global region interrupt output is gated based on the enable mask that is provided by IER/IERH. see Figure 5-14

The region interrupt outputs are gated by IER and the specific DRAE/DRAEH associated with the region. See Figure 5-14.

Figure 5-14. Interrupt Diagram



For the EDMA3CC to generate the transfer completion interrupts that are associated with each shadow region, the following conditions must be true:

- EDMA3CC_INT0: (IPR.E0 & IER.E0 & DRAE0.E0) | (IPR.E1 & IER.E1 & DRAE0.E1) | …|(IPRH.E63 & IERH.E63 & DRAHE0.E63)
- EDMA3CC_INT1: (IPR.E0 & IER.E0 & DRAE1.E0) | (IPR.E1 & IER.E1 & DRAE1.E1) | …| (IPRH.E63 & IERH.E63 & DRAHE1.E63)
- EDMA3CC_INT2 : (IPR.E0 & IER.E0 & DRAE2.E0) | (IPR.E1 & IER.E1 & DRAE2.E1) | …|(IPRH.E63 & IERH.E63 & DRAHE2.E63)....
- Up to EDMA3CC_INT7 : (IPR.E0 & IER.E0 & DRAE7.E0) | (IPR.E1 & IER.E1 & DRAE7.E1) | …|(IPRH.E63 & IERH.E63 & DRAEH7.E63)

NOTE: The DRAE/DRAEH for all regions are expected to be set up at system initialization and to remain static for an extended period of time. The interrupt enable registers should be used for dynamic enable/disable of individual interrupts.

Because there is no relation between the TCC value and the DMA/QDMA channel, it is possible, for example, for DMA channel 0 to have the OPT.TCC = 63 in its associated PaRAM set. This would mean that if a transfer completion interrupt is enabled (OPT.TCINTEN or OPT.ITCINTEN is set), then based on the TCC value, IPRH.E63 is set up on completion. For proper channel operations and interrupt generation using the shadow region map, you must program the DRAE/DRAEH that is associated with the shadow region to have read/write access to both bit 0 (corresponding to channel 0) and bit 63 (corresponding to IPRH bit that is set upon completion).

5.2.9.1.2 Clearing Transfer Completion Interrupts

Transfer completion interrupts that are latched to the interrupt pending registers (IPR/IPRH) are cleared by writing a 1 to the corresponding bit in the interrupt pending clear register (ICR/ICRH). For example, a write of 1 to ICR.E0 clears a pending interrupt in IPR.E0.

If an incoming transfer completion code (TCC) gets latched to a bit in IPR/IPRH, then additional bits that get set due to a subsequent transfer completion will not result in asserting the EDMA3CC completion interrupt. In order for the completion interrupt to be pulsed, the required transition is from a state where no enabled interrupts are set to a state where at least one enabled interrupt is set.

5.2.9.2 EDMA3 Interrupt Servicing

Upon completion of a transfer (early or normal completion), the EDMA3 channel controller sets the appropriate bit in the interrupt pending registers (IPR/IPRH), as the transfer completion codes specify. If the completion interrupts are appropriately enabled, then the CPU enters the interrupt service routine (ISR) when the completion interrupt is asserted.

After servicing the interrupt, the ISR should clear the corresponding bit in IPR/IPRH, thereby enabling recognition of future interrupts. The EDMA3CC will only assert additional completion interrupts when all IPR/IPRH bits clear.

When one interrupt is serviced many other transfer completions may result in additional bits being set in IPR/IPRH, thereby resulting in additional interrupts. Each of the bits in IPR/IPRH may need different types of service; therefore, the ISR may check all pending interrupts and continue until all of the posted interrupts are serviced appropriately.

Examples of pseudo code for a CPU interrupt service routine for an EDMA3CC completion interrupt are shown in Example 5-2 and Example 5-3.

The ISR routine in Example 5-2 is more exhaustive and incurs a higher latency.

Example 5-2. Interrupt Servicing

The pseudo code:

1. Reads the interrupt pending register (IPR/IPRH).
2. Performs the operations needed.
3. Writes to the interrupt pending clear register (ICR/ICRH) to clear the corresponding IPR/IPRH bit(s).
4. Reads IPR/IPRH again:
   (a) If IPR/IPRH is not equal to 0, repeat from step 2 (implies occurrence of new event between step 2 to step 4).
   (b) If IPR/IPRH is equal to 0, this should assure you that all of the enabled interrupts are inactive.

NOTE: An event may occur during step 4 while the IPR/IPRH bits are read as 0 and the application is still in the interrupt service routine. If this happens, a new interrupt is recorded in the device interrupt controller and a new interrupt generates as soon as the application exits in the interrupt service routine.

Example 5-3 is less rigorous, with less burden on the software in polling for set interrupt bits, but can occasionally cause a race condition as mentioned above.

Example 5-3. Interrupt Servicing

If you want to leave any enabled and pending (possibly lower priority) interrupts; you must force the interrupt logic to reassert the interrupt pulse by setting the EVAL bit in the interrupt evaluation register (IEVAL).

The pseudo code is as follows:

1. Enters ISR.
2. Reads IPR/IPRH.
3. For the condition that is set in IPR/IPRH that you want to service, do the following:
   (a) Service interrupt as the application requires.
   (b) Clear the bit for serviced conditions (others may still be set, and other transfers may have resulted in returning the TCC to EDMA3CC after step 2).
4. Reads IPR/IPRH prior to exiting the ISR:
   (a) If IPR/IPRH is equal to 0, then exit the ISR.
   (b) If IPR/IPRH is not equal to 0, then set IEVAL so that upon exit of ISR, a new interrupt triggers if any enabled interrupts are still pending.

5.2.9.3 Interrupt Evaluation Operations

The EDMA3CC has interrupt evaluate registers (IEVAL) that exist in the global region and in each shadow region. The registers in the shadow region are the only registers in the DMA channel shadow region memory map that are not affected by the settings for the DMA region access enable registers (DRAE/DRAEH). Writing a 1 to the EVAL bit in the registers that are associated with a particular shadow region results in pulsing the associated region interrupt (global or shadow), if any enabled interrupt (via IER/IERH) is still pending (IPR/IPRH). This register assures that the CPU does not miss the interrupts (or the EDMA3 master associated with the shadow region) if the software architecture chooses not to use all interrupts. See Example 5-3 for the use of IEVAL in the EDMA3 interrupt service routine (ISR).

Similarly an error evaluation register (EEVAL) exists in the global region. Writing a 1 to the EVAL bit in EEVAL causes the pulsing of the error interrupt if any pending errors are in EMR/EMRH, QEMR, or CCERR. See Section 5.2.9.4 for additional information regarding error interrupts.

NOTE: While using IEVAL for shadow region completion interrupts, you should make sure that the IEVAL operated upon is from that particular shadow region memory map.

5.2.9.4 Error Interrupts

The EDMA3CC error registers provide the capability to differentiate error conditions (event missed, threshold exceed, etc.). Additionally, setting the error bits in these registers results in asserting the EDMA3CC error interrupt. If the EDMA3CC error interrupt is enabled in the device interrupt controller(s), then it allows the CPU(s) to handle the error conditions.

The EDMA3CC has a single error interrupt (EDMA3CC_ERRINT) that is asserted for all EDMA3CC error conditions. There are four conditions that cause the error interrupt to pulse:

- DMA missed events: for all 64 DMA channels. DMA missed events are latched in the event missed registers (EMR/EMRH).
- QDMA missed events: for all 8 QDMA channels. QDMA missed events are latched in the QDMA event missed register (QEMR).
- Threshold exceed: for all event queues. These are latched in EDMA3CC error register (CCERR).
- TCC error: for outstanding transfer requests that are expected to return completion code (TCCHEN or TCINTEN bit in OPT is set to 1) exceeding the maximum limit of 63. This is also latched in the EDMA3CC error register (CCERR).

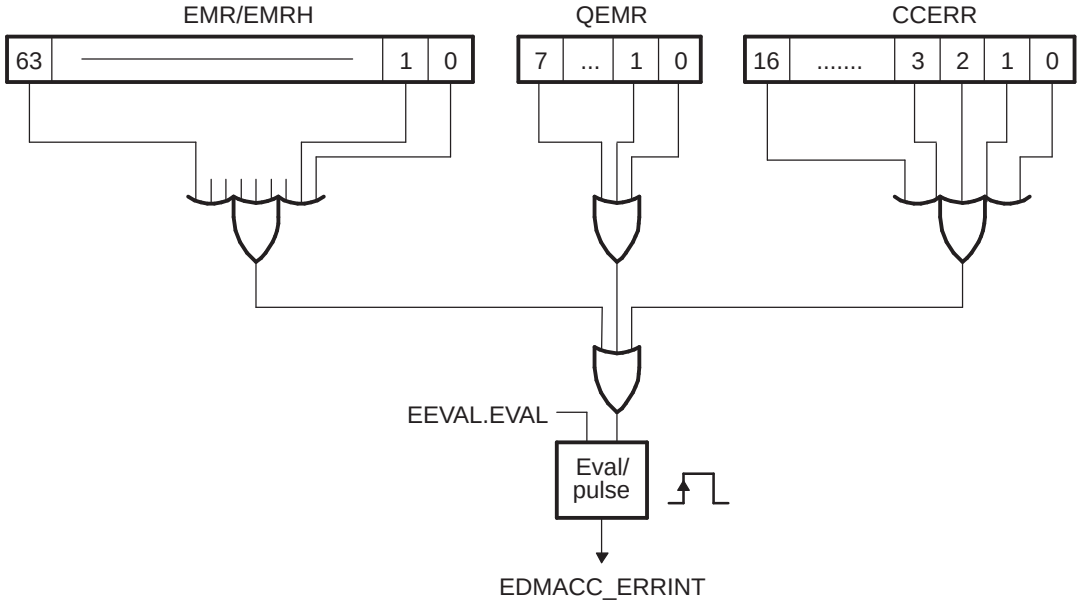
Figure 5-15 illustrates the EDMA3CC error interrupt generation operation.

If any of the bits are set in the error registers due to any error condition, the EDMA3CC_ERRINT is always asserted, as there are no enables for masking these error events. Similar to transfer completion interrupts (EDMA3CC_INT), the error interrupt also only pulses when the error interrupt condition transitions from no errors being set to at least one error being set. If additional error events are latched prior to the original error bits clearing, the EDMA3CC does not generate additional interrupt pulses.

To reduce the burden on the software, there is an error evaluate register (EEVAL) that allows re-evaluation of pending set error events/bits, similar to the interrupt evaluate register (IEVAL). You can use this so that the CPU(s) does not miss any error events.

NOTE: It is good practice to enable the error interrupt in the device interrupt controller and to associate an interrupt service routine with it to address the various error conditions appropriately. Doing so puts less burden on the software (polling for error status); additionally, it provides a good debug mechanism for unexpected error conditions.

Figure 5-15. Error Interrupt Operation



5.2.10 Memory Protection

The EDMA3 channel controller supports two kinds of memory protection: active and proxy.

5.2.10.1 Active Memory Protection

Active memory protection is a feature that allows or prevents read and write accesses (by any EDMA3 programmer) to the EDMA3CC registers (based on permission characteristics that you program). Active memory protection is achieved by a set of memory protection permissions attribute (MPPA) registers.

The EDMA3CC register map is divided into three categories:

- a global region.
- a global channel region.
- eight shadow regions.

Each shadow region consists of the respective shadow region registers and the associated PaRAM. For more detailed information regarding the contents of a shadow region, refer to section Table 5-7.

Each of the eight shadow regions has an associated MPPA register (MPPA*n*) that defines the specific requestor(s) and types of requests that are allowed to the regions resources.

The global channel region is also protected with a memory-mapped register (MPPAG). The MPPAG applies to the global region and to the global channel region, except the other MPPA registers themselves. For more detailed information on the list of the registers in each region, refer to the register memory-map in section Table 5-14.

See Section 5.4.1.5.4 for the bit field descriptions of MPPA*n*. The MPPA*n* have a certain set of access rules.

Table 5-13 shows the accesses that are allowed or not allowed to the MPPAG and MPPA*n*. The active memory protection uses the PRIV and PRIVID attributes of the EDMA programmer. The PRIV is the privilege level (user versus supervisor). The PRIVID refers to a privilege ID with a number that is associated with an EDMA3 programmer. See the device-specific data manual for the PRIVIDs that are associated with potential EDMA3 programmers.

Table 5-13. Allowed Accesses

Access	Supervisor	User
Read	Yes	Yes
Write	Yes	No

Table 5-14 describes the MPPA register mapping for the shadow regions (which includes shadow region registers and PaRAM addresses).

The region-based MPPA registers are used to protect accesses to the DMA shadow regions and the associated region PaRAM. Because there are eight regions, there are eight MPPA region registers (MPPA[0-7]).

Table 5-14. MPPA Registers to Region Assignment

Register	Registers Protect	Address Range	PaRAM Protect[1]	Address Range
MPPAG	Global Range	0000h-1FFCh	N/A	N/A
MPPA0	DMA Shadow 0	2000h-21FCh	1st octant	4000h-47FCh
MPPA1	DMA Shadow 1	2200h-23FCh	2nd octant	4800h-4FFCh
MPPA2	DMA Shadow 2	2400h-25FCh	3rd octant	5000h-57FCh
MPPA3	DMA Shadow 3	2600h-27FCh	4th octant	5800h-5FFCh
MPPA4	DMA Shadow 4	2800h-29FCh	5th octant	6000h-67FCh
MPPA5	DMA Shadow 5	2A00h-2BFCh	6th octant	6800h-6FFCh
MPPA6	DMA Shadow 6	2C00h-2DFCh	7th octant	7000h-77FCh
MPPA7	DMA Shadow 7	2E00h-2FFCh	8th octant	7800h-7FFCh

(1) The PARAM region is divided into 8 regions referred to as an octant.

Example Access denied.

Write access to shadow region 7's event enable set register (EESR):

1. The original value of the event enable register (EER) at address offset 1020h is 0.
2. The MPPA[7] is set to prevent user level accesses (UW = 0, UR = 0), but it allows supervisor level accesses (SW = 1, SR = 1) with a privilege ID of 0. (AID0 = 1).
3. An EDMA3 programmer with a privilege ID of 0 attempts to perform a user-level write of a value of FF00 FF00h to shadow region 7's event enable set register (EESR) at address offset 2E30h. Note that the EER is a read-only register and the only way that you can write to it is by writing to the EESR. Also remember that there is only one physical register for EER, EESR, etc. and that the shadow regions only provide to the same physical set.
4. Since the MPPA[7] has UW = 0, though the privilege ID of the write access is set to 0, the access is not allowed and the EER is not written to.

Table 5-15. Example Access Denied

Register	Value	Description
EER (offset 1020h)	0000 0000h	Value in EER to begin with.
EESR (offset 2E30h)	FF00 FF00h ↓	Value attempted to be written to shadow region 7's EESR. This is done by an EDMA3 programmer with a privilege level of User and Privilege ID of 0.
MPPA[7] (offset 082Ch)	0000 04B0h	Memory Protection Filter AID0 = 1, UW = 0, UR = 0, SW = 1, SR = 1.
	X	Access Denied
EER (offset 1020h)	0000 0000h	Final value of EER

Example Access Allowed

Write access to shadow region 7's event enable set register (EESR):

1. The original value of the event enable register (EER) at address offset 1020h is 0.
2. The MPPA[7] is set to allow user-level accesses (UW = 1, UR = 1) and supervisor-level accesses (SW = 1, SR = 1) with a privilege ID of 0. (AID0 = 1).
3. An EDMA3 programmer with a privilege ID of 0, attempts to perform a user-level write of a value of ABCD 0123h to shadow region 7's event enable set register (EESR) at address offset 2E30h. Note that the EER is a read-only register and the only way that you can write to it is by writing to the EESR. Also remember that there is only one physical register for EER, EESR, etc. and that the shadow regions only provide to the same physical set.
4. Since the MPPA[7] has UW = 1 and AID0 = 1, the user-level write access is allowed.
5. Remember that accesses to shadow region registers are masked by their respective DRAE register. In this example, the DRAE[7] is set of 9FF0 0FC2h.
6. The value finally written to EER is 8BC0 0102h.

Table 5-16. Example Access Allowed

Register	Value	Description
EER (offset 1020h)	0000 0000h	Value in EER to begin with.
EESR (offset 2E30h)	FF00 FF00h	Value attempted to be written to shadow region 7's EESR. This is done by an EDMA3 programmer with a privilege level of User and Privilege ID of 0.
MPPA[7] (offset 082Ch)	0000 04B3h	Memory Protection Filter AID = 1, UW = 1, UR = 1, SW = 1, SR = 1.
	√ ↓	Access allowed.
DRAE[7] (offset 0378h)	9FF0 0FC2h ↓	DMA Region Access Enable Filter
EESR (offset 2E30h)	8BC0 0102h ↓	Value written to shadow region 7's EESR. This is done by an EDMA3 programmer with a privilege level of User and a Privilege ID of 0.
EER (offset 1020h)	↓ BC0 0102h	Final value of EER.

5.2.10.2 Proxy Memory Protection

Proxy memory protection allows an EDMA3 transfer programmed by a given EDMA3 programmer to have its permissions travel with the transfer through the EDMA3TC. The permissions travel along with the read transactions to the source and the write transactions to the destination endpoints. The PRIV bit and PRIVID bit in the channel options parameter (OPT) is set with the EDMA3 programmer's PRIV value and PRIVID values, respectively, when any part of the PaRAM set is written.

The PRIV is the privilege level (user versus supervisor). The PRIVID refers to a privilege ID with a number that is associated with an EDMA3 programmer.

See the data manual for the PRIVIDs that are associated with potential EDMA3 programmers.

These options are part of the TR that are submitted to the transfer controller. The transfer controller uses the above values on their respective read and write command bus so that the target endpoints can perform memory protection checks based on these values.

Consider a parameter set that is programmed by a CPU in user privilege level for a simple transfer with the source buffer on an L2 page and the destination buffer on an L1D page. The PRIV is 0 for user-level and the CPU has a PRIVID of 0.

The PaRAM set is shown in Figure 5-16.

The PRIV and PRIVID information travels along with the read and write requests that are issued to the source and destination memories.

For example, if the access attributes that are associated with the L2 page with the source buffer only allow supervisor read, write accesses (SR,SW), the user-level read request above is refused. Similarly, if the access attributes that are associated with the L1D page with the destination buffer only allow supervisor read and write accesses (SR, SW), the user-level write request above is refused. For the transfer to succeed, the source and destination pages should have user-read and user-write permissions, respectively, along with allowing accesses from a PRIVID 0. For more information regarding how to set memory protection attributes for pages of memory in L2/L1D, see the *TMS320C674x DSP CPU and Instruction Set Reference Guide* (SPRUFE8).

Because the programmers privilege level and privilege identification travel with the read and write requests, EDMA3 acts as a proxy.

Figure 5-17 illustrates the propagation of PRIV and PRIVID at the boundaries of all the interacting entities (CPU, EDMA3CC, EDMA3TC, and slave memories).

Figure 5-16. PaRAM Set Content for Proxy Memory Protection Example

(a) EDMA3 Parameters

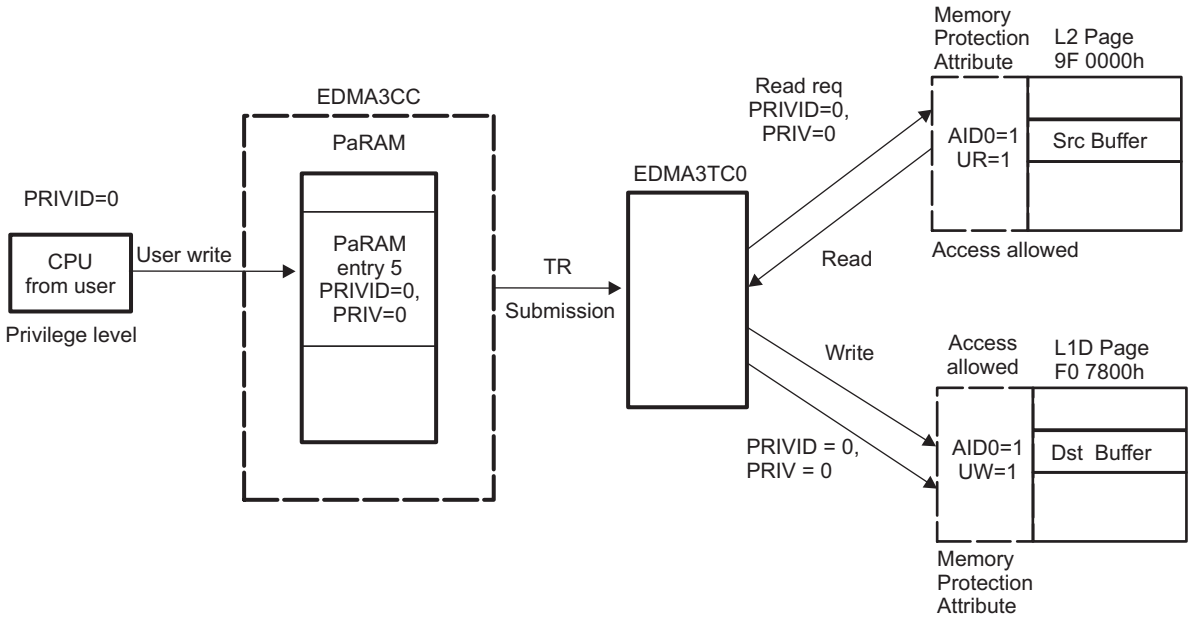
Parameter Contents	Parameter
0010 0007h	Channel Options Parameter (OPT)
009F 0000h	Channel Source Address (SRC)
0001h | 0004h	Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT)
00F0 7800h	Channel Destination Address (DST)
0001h | 0001h	Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX)
0000h | FFFFh	BCNT Reload (BCNTRLD) | Link Address (LINK)
0001h | 1000h	Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX)
0000h | 0001h	Reserved | Count for 3rd Dimension (CCNT)

(d) Channel Options Parameter (OPT) Content

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	0	1	1	1
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

Figure 5-17. Proxy Memory Protection Example



5.2.11 Event Queue(s)

Event queues are a part of the EDMA3 channel controller. Event queues form the interface between the event detection logic in the EDMA3CC and the transfer request (TR) submission logic of the EDMA3CC. Each queue is 16 entries deep; thus, each event queue can queue a maximum of 16 events. If there are more than 16 events, then the events that cannot find a place in the event queue remain set in the associated event register and the CPU does not stall.

There are four event queues for the device: Queue0, Queue1, Queue2, and Queue3. Events in Queue0 result in submission of its associated transfer requests (TRs) to TC0. Similarly, transfer requests that are associated with events in Queue3 are submitted to TC3.

An event that wins prioritization against other DMA and/or QDMA pending events is placed at the tail of the appropriate event queue. Each event queue is serviced in FIFO order. Once the event reaches the head of its queue and the corresponding transfer controller is ready to receive another TR, the event is de-queued and the PaRAM set corresponding to the de-queued event is processed and submitted as a transfer request packet (TRP) to the associated EDMA3 transfer controller.

Queue0 has highest priority and Queue3 has the lowest priority, if Queue0 and Queue1 both have at least one event entry and if both TC0 and TC1 can accept transfer requests, then the event in Queue0 is de-queued first and its associated PaRAM set is processed and submitted as a transfer request (TR) to TC0.

See Section 5.2.11.4 for system-level performance considerations. All of the event entries in all of the event queues are software readable (not writeable) by accessing the event entry registers (Q0E0, Q0E1,…Q1E15, etc.). Each event entry register characterizes the queued event in terms of the type of event (manual, event, chained or auto-triggered) and the event number. See Section 5.4.1.4.1 for a description of the bit fields in the queue event entry registers.

5.2.11.1 DMA/QDMA Channel to Event Queue Mapping

Each of the 64 DMA channels and eight QDMA channels are programmed independently to map to a specific queue, using the DMA queue number register (DMAQNUM) and the QDMA queue number register (QDMANUM). The mapping of DMA/QDMA channels is critical to achieving the desired performance level for the EDMA and most importantly, in meeting real-time deadlines. See Section 5.2.11.4.

NOTE: If an event is ready to be queued and both the event queue and the EDMA3 transfer controller that is associated to the event queue are empty, then the event bypasses the event queue, and moves the PaRAM processing logic, and eventually to the transfer request submission logic for submission to the EDMA3TC. In this case, the event is not logged in the event queue status registers.

5.2.11.2 Queue RAM Debug Visibility

There are four event queues and each queue has 16 entries. These 16 entries are managed in a circular FIFO manner. There is a queue status register (QSTAT) associated with each queue. These along with all of the 16 entries per queue can be read via registers QSTAT*n* and QxEy, respectively.

These registers provide user visibility and may be helpful while debugging real-time issues (typically post-mortem), involving multiple events and event sources. The event queue entry register (Q*x*E*y*) uniquely identifies the specific event type (event-triggered, manually-triggered, chain-triggered, and QDMA events) along with the event number (for all DMA/QDMA event channels) that are in the queue or have been de-queued (passed through the queue).

Each of the 16 entries in the event queue are read using the EDMA3CC memory-mapped register. By reading the event queue, you see the history of the last 16 TRs that have been processed by the EDMA3 on a given queue. This provides user/software visibility and is helpful for debugging real-time issues (typically post-mortem), involving multiple events and event sources.

The queue status register (QSTAT*n*) includes fields for the start pointer (STRTPTR) which provides the offset to the head entry of an event. It also includes a field called NUMVAL that provides the total number of valid entries residing in the event queue at a given instance of time. The STRTPTR may be used to index appropriately into the 16 event entries. NUMVAL number of entries starting from STRTPTR are indicative of events still queued in the respective queue. The remaining entry may be read to determine what's already de-queued and submitted to the associated transfer controller.

5.2.11.3 Queue Resource Tracking

The EDMA3CC event queue includes watermarking/threshold logic that allows you to keep track of maximum usage of all event queues. This is useful for debugging real-time deadline violations that may result from head-of-line blocking on a given EDMA3 event queue.

You can program the maximum number of events that can queue up in an event queue by programming the threshold value (between 0 to 15) in the queue watermark threshold A register (QWMTHRA). The maximum queue usage is recorded actively in the watermark (WM) field of the queue status register (QSTAT*n*) that keeps getting updated based on a comparison of number of valid entries, which is also visible in the NUMVAL bit in QSTAT*n* and the maximum number of entries (WM bit in QSTAT*n*).

If the queue usage is exceeded, this status is visible in the EDMA3CC registers: the QTHRXCD*n* bit in the channel controller error register (CCERR) and the THRXCD bit in QSTAT*n*, where *n* stands for the event queue number. Any bits that are set in CCERR also generate an EDMA3CC error interrupt.

5.2.11.4 Performance Considerations

The main system bus infrastructure (L3) (see the device-specific data manual) arbitrates bus requests from all of the masters (TCs, CPU(S), PCIe and other bus masters) to the shared slave resources (peripherals and memories).

The priorities of transfer requests (read and write commands) from the EDMA3 transfer controllers with respect to other masters within the system crossbar are programmed using the queue priority register (QUEPRI). QUEPRI programs the priority of the event queues (or indirectly, TC0-TC3, because Queue*N* transfer requests are submitted to TC*n*).

Therefore, the priority of unloading queues has a secondary affect compared to the priority of the transfers as they are executed by the EDMA3TC (dictated by the priority set using QUEPRI).

5.2.12 EDMA3 Transfer Controller (EDMA3TC)

The EDMA3 channel controller is the user-interface of the EDMA3 and the EDMA3 transfer controller (EDMA3TC) is the data movement engine of the EDMA3. The EDMA3CC submits transfer requests (TR) to the EDMA3TC and the EDMA3TC performs the data transfers dictated by the TR; thus, the EDMA3TC is a slave to the EDMA3CC.

5.2.12.1 Architecture Details

5.2.12.1.1 Command Fragmentation

The TC read and write controllers in conjunction with the source and destination register sets are responsible for issuing optimally-sized reads and writes to the slave endpoints. An optimally-sized command is defined by the transfer controller default burst size (DBS), which is defined in Section 5.2.12.5.

The EDMA3TC attempts to issue the largest possible command size as limited by the DBS value or the ACNT/BCNT value of the TR. EDMA3TC obeys the following rules:

- The read/write controllers always issue commands less than or equal to the DBS value.
- The first command of a 1D transfer command always aligns the address of subsequent commands to the DBS value.

Table 5-17 lists the TR segmentation rules that are followed by the EDMA3TC. In summary, if the ACNT value is larger than the DBS value, then the EDMA3TC breaks the ACNT array into DBS-sized commands to the source/destination addresses. Each BCNT number of arrays are then serviced in succession.

For BCNT arrays of ACNT bytes (that is, a 2D transfer), if the ACNT value is less than or equal to the DBS value, then the TR may be optimized into a 1D-transfer in order to maximize efficiency. The optimization takes place if the EDMA3TC recognizes that the 2D-transfer is organized as a single dimension (ACNT == BIDX) and the ACNT value is a power of 2.

Table 5-17 lists conditions in which the optimizations are performed.

Table 5-17. Read/Write Command Optimization Rules

ACNT ≤ DBS	ACNT is power of 2	BIDX = ACNT	BCNT ≤ 1023	SAM/DAM = Increment	Description
Yes	Yes	Yes	Yes	Yes	Optimized
No	x	x	x	x	Not Optimized
x	No	x	x	x	Not Optimized
x	x	No	x	x	Not Optimized
x	x	x	No	x	Not Optimized
x	x	x	x	No	Not Optimized

5.2.12.1.2 TR Pipelining

TR pipelining refers to the ability of the source active set to proceed ahead of the destination active set. Essentially, the reads for a given TR may already be in progress while the writes of a previous TR may not have completed.

The number of outstanding TRs is limited by the number of destination FIFO register entries.

TR pipelining is useful for maintaining throughput on back-to-back small TRs. It minimizes the startup overhead because reads start in the background of a previous TR writes.

Example 5-4. Command Fragmentation (DBS = 64)

The pseudo code:

1. ACNT = 8, BCNT = 8, SRCBIDX = 8, DSTBIDX = 10, SRCADDR = 64, DSTADDR = 191

   Read Controller: This is optimized from a 2D-transfer to a 1D-transfer such that the read side is equivalent to ACNT = 64, BCNT = 1.

   Cmd0 = 64 byte

   Write Controller: Because DSTBIDX != ACNT, it is not optimized.

   Cmd0 = 8 byte, Cmd1 = 8 byte, Cmd2 = 8 byte, Cmd3 = 8 byte, Cmd4 = 8 byte, Cmd5 = 8 byte, Cmd6 = 8 byte, Cmd7 = 8 byte.

2. ACNT=128, BCNT = 1,SRCADDR = 63, DSTADDR = 513

   Read Controller: Read address is not aligned.

   Cmd0 = 1 byte, (now the SRCADDR is aligned to 64 for the next command)

   Cmd1 = 64 bytes

   Cmd2 = 63 bytes

   Write Controller: The write address is also not aligned.

   Cmd0 = 63 bytes, (now the DSTADDR is aligned to 64 for the next command)

   Cmd1 = 64 bytes

   Cmd2 = 1 byte

5.2.12.1.3 Performance Tuning

By default, reads are as issued as fast as possible. In some cases, the reads issued by the EDMA3TC could fill the available command buffering for a slave, delaying other (potentially higher priority) masters from successfully submitting commands to that slave. The rate at which read commands are issued by the EDMA3TC is controlled by the RDRATE register. The RDRATE register defines the number of cycles that the EDMA3TC read controller waits before issuing subsequent commands for a given TR, thus minimizing the chance of the EDMA3TC consuming all available slave resources. The RDRATE value should be set to a relatively small value if the transfer controller is targeted for high priority transfers and to a higher value if the transfer controller is targeted for low priority transfers.

In contrast, the Write Interface does not have any performance turning knobs because writes always have an interval between commands as write commands are submitted along with the associated write data.

5.2.12.2 Memory Protection

The transfer controller plays an important role in handling proxy memory protection. There are two access properties associated with a transfer: for instance, the privilege id (system-wide identification assigned to a master) of the master initiating the transfer, and the privilege level (user versus supervisor) used to program the transfer. This information is maintained in the PaRAM set when it is programmed in the channel controller. When a TR is submitted to the transfer controller, this information is made available to the EDMA3TC and used by the EDMA3TC while issuing read and write commands. The read or write commands have the same privilege identification, and privilege level as that programmed in the EDMA3 transfer in the channel controller.

5.2.12.3 Error Generation

Errors are generated if enabled under three conditions:

- EDMA3TC detection of an error signaled by the source or destination address.
- Attempt to read or write to an invalid address in the configuration memory map.
- Detection of a constant addressing mode TR violating the constant addressing mode transfer rules (the source/destination addresses and source/destination indexes must be aligned to 32 bytes).

Either or all error types may be disabled. If an error bit is set and enabled, the error interrupt for the concerned transfer controller is pulsed.

5.2.12.4 Debug Features

The DMA program register set, DMA source active register set, and the destination FIFO register set are used to derive a brief history of TRs serviced through the transfer controller.

Additionally, the EDMA3TC status register (TCSTAT) has dedicated bit fields to indicate the ongoing activity within different parts of the transfer controller:

- The SRCACTV bit indicates whether the source active set is active.
- The DSTACTV bit indicates the number of TRs resident in the destination register active set at a given instance.
- The PROGBUSY bit indicates whether a valid TR is present in the DMA program set.

If the TRs are in progression, caution must be used and you must realize that there is a chance that the values read from the EDMA3TC status registers will be inconsistent since the EDMA3TC may change the values of these registers due to ongoing activities.

It is recommended that you ensure no additional submission of TRs to the EDMA3TC in order to facilitate ease of debug.

5.2.12.4.1 Destination FIFO Register Pointer

The destination FIFO register pointer is implemented as a circular buffer with the start pointer being DFSTRTPTR and a buffer depth of usually 2 or 4. The EDMA3TC maintains two important status details in TCSTAT that may be used during advanced debugging, if necessary. The DFSTRTPTR is a start pointer, that is, the index to the head of the destination FIFO register. The DSTACTV is a counter for the number of valid (occupied) entries. These registers may be used to get a brief history of transfers.

Examples of some register field values and their interpretation:

- DFSTRTPTR = 0 and DSTACTV = 0 implies that no TRs are stored in the destination FIFO register.
- DFSTRTPTR = 1 and DSTACTV = 2h implies that two TRs are present. The first pending TR is read from the destination FIFO register entry 1 and the second pending TR is read from the destination FIFO register entry 2.
- DFSTRTPTR = 3h and DSTACTV = 2h implies that two TRs are present. The first pending TR is read from the destination FIFO register entry 3 and the second pending TR is read from the destination FIFO register entry 0.

5.2.12.5 EDMA3TC Configuration

Table 5-18 provides the configuration of the individual EDMA3 transfer controllers present on the device. The DBS for each transfer controller is configurable using the TPTC_CFG register in the chip configuration module.

Table 5-18. EDMA3 Transfer Controller Configurations

Name	TC0	TC1	TC2	TC3
FIFOSIZE	1024 bytes	1024 bytes	1024 bytes	1024 bytes
BUSWIDTH	16 bytes	16 bytes	16 bytes	16 bytes
DSTREGDEPTH	4 entries	4 entries	4 entries	4 entries
DBS	Configurable	Configurable	Configurable	Configurable

5.2.13 Event Dataflow

This section summarizes the data flow of a single event, from the time the event is latched to the channel controller to the time the transfer completion code is returned. The following steps list the sequence of EDMA3CC activity:

1. Event is asserted from an external source (peripheral or external interrupt). This also is similar for a manually-triggered, chained-triggered, or QDMA-triggered event. The event is latched into the ER.E*n*/ERH.E*n* (or CER.E*n*/CERH.E*n*, ESR.E*n* /ESRH.E*n*, QER.E*n*) bit.
2. Once an event is prioritized and queued into the appropriate event queue, the SER.E*n*\SERH.E*n* (or QSER.E*n*) bit is set to inform the event prioritization/processing logic to disregard this event since it is already in the queue. Alternatively, if the transfer controller and the event queue are empty, then the event bypasses the queue.
3. The EDMA3CC processing and the submission logic evaluates the appropriate PaRAM set and determines whether it is a non-null and non-dummy transfer request (TR).
4. The EDMA3CC clears the ER.E*n*/ERH.E*n* (or CER.E*n*/CERH.E*n*, ESR.E*n*/ESRH.E*n*, QER.E*n*) bit and the SER.E*n*/SERH.E*n* bit as soon as it determines the TR is non-null. In the case of a null set, the SER.E*n*/SERH.E*n* bit remains set. It submits the non-null/non-dummy TR to the associated transfer controller. If the TR was programmed for early completion, the EDMA3CC immediately sets the interrupt pending register (IPR.I[TCC]/IPRH.I[TCC]-32).
5. If the TR was programmed for normal completion, the EDMA3CC sets the interrupt pending register (IPR.I[TCC]/IPRH.I[TCC]) when the EDMA3TC informs the EDMA3CC about completion of the transfer (returns transfer completion codes).
6. The EDMA3CC programs the associated EDMA3TC*n*'s Program Register Set with the TR.
7. The TR is then passed to the Source Active set and the DST FIFO Register Set, if both the register sets are available.
8. The Read Controller processes the TR by issuing read commands to the source slave endpoint. The Read Data lands in the Data FIFO of the EDMA3TC*n*.
9. As soon as sufficient data is available, the Write Controller begins processing the TR by issuing write commands to the destination slave endpoint.
10. This continues until the TR completes and on receiving the acknowledgement signal from the destination slave end point, the EDMA3TC*n* then signals completion status to the EDMA3CC.

5.2.14 EDMA3 Prioritization

The EDMA3 controller has many implementation rules to deal with concurrent events/channels, transfers, etc. The following subsections detail various arbitration details whenever there might be occurrence of concurrent activity. Figure 5-18 shows the different places EDMA3 priorities come into play.

5.2.14.1 Channel Priority

The DMA event registers (ER and ERH) capture up to 64 events; likewise, the QDMA event register (QER) captures QDMA events for all QDMA channels; therefore, it is possible for events to occur simultaneously on the DMA/QDMA event inputs. For events arriving simultaneously, the event associated with the lowest channel number is prioritized for submission to the event queues (for DMA events, channel 0 has the highest priority and channel 63 has the lowest priority; similarly, for QDMA events, channel 0 has the highest priority and channel 7 has the lowest priority). This mechanism only sorts simultaneous events for submission to the event queues.

If a DMA and QDMA event occurs simultaneously, the DMA event always has prioritization against the QDMA event for submission to the event queues.

5.2.14.2 Trigger Source Priority

If a DMA channel is associated with more than one trigger source (event trigger, manual trigger, and chain trigger), and if multiple events are set simultaneously for the same channel (ER.E*n* = 1, ESR.E*n* = 1 , CER.E*n* = 1) , then the EDMA3CC always services these events in the following priority order: event trigger (via ER) is higher priority than chain trigger (via CER) and chain trigger is higher priority than manual trigger (via ESR).

This implies that if for channel 0, both ER.E0 = 1 and CER.E0 = 1 at the same time, then the ER.E0 event is always queued before the CER.E0 event.

5.2.14.3 Dequeue Priority

The priority of the associated transfer request (TR) is further mitigated by which event queue is being used for event submission (dictated by DMAQNUM and QDMAQNUM). For submission of a TR to the transfer request, events need to be de-queued from the event queues. Queue 0 has the highest dequeue priority and queue 3 the lowest.

5.2.14.4 System (Transfer Controller) Priority

INIT_PRIORITY_0 and INIT_PRIORITY_1 registers in the chip configuration module are used to configure the EDMA TC's priority through the system bus infrastructure. Additionally, the priority settings for DDR memory accesses are defined in the dynamic memory manager (DMM).

NOTE: The default priority for all TCs is the same, 0 or highest priority relative to other masters. It is recommended that this priority be changed based on system level considerations, such as real-time deadlines for all masters including the priority of the transfer controllers with respect to each other.

5.2.15 EDMA3 Operating Frequency (Clock Control)

The EDMA3 channel controller and transfer controller are clocked from PLL_L3 SYSCLK4. The EDMA3 system runs at the L3 clock frequency.

5.2.16 Reset Considerations

A hardware reset resets the EDMA3 (EDMA3CC and EDMA3TC) and the EDMA3 configuration registers. The PaRAM memory contents are undefined after device reset and you should not rely on parameters to be reset to a known state. The PaRAM entry must be initialized to a desired value before it is used.

5.2.17 Power Management

The EDMA3 (EDMA3CC and EDMA3TC) can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the power reset clock management (PRCM). The PRCM acts as a master controller for power management for all peripherals on the device.

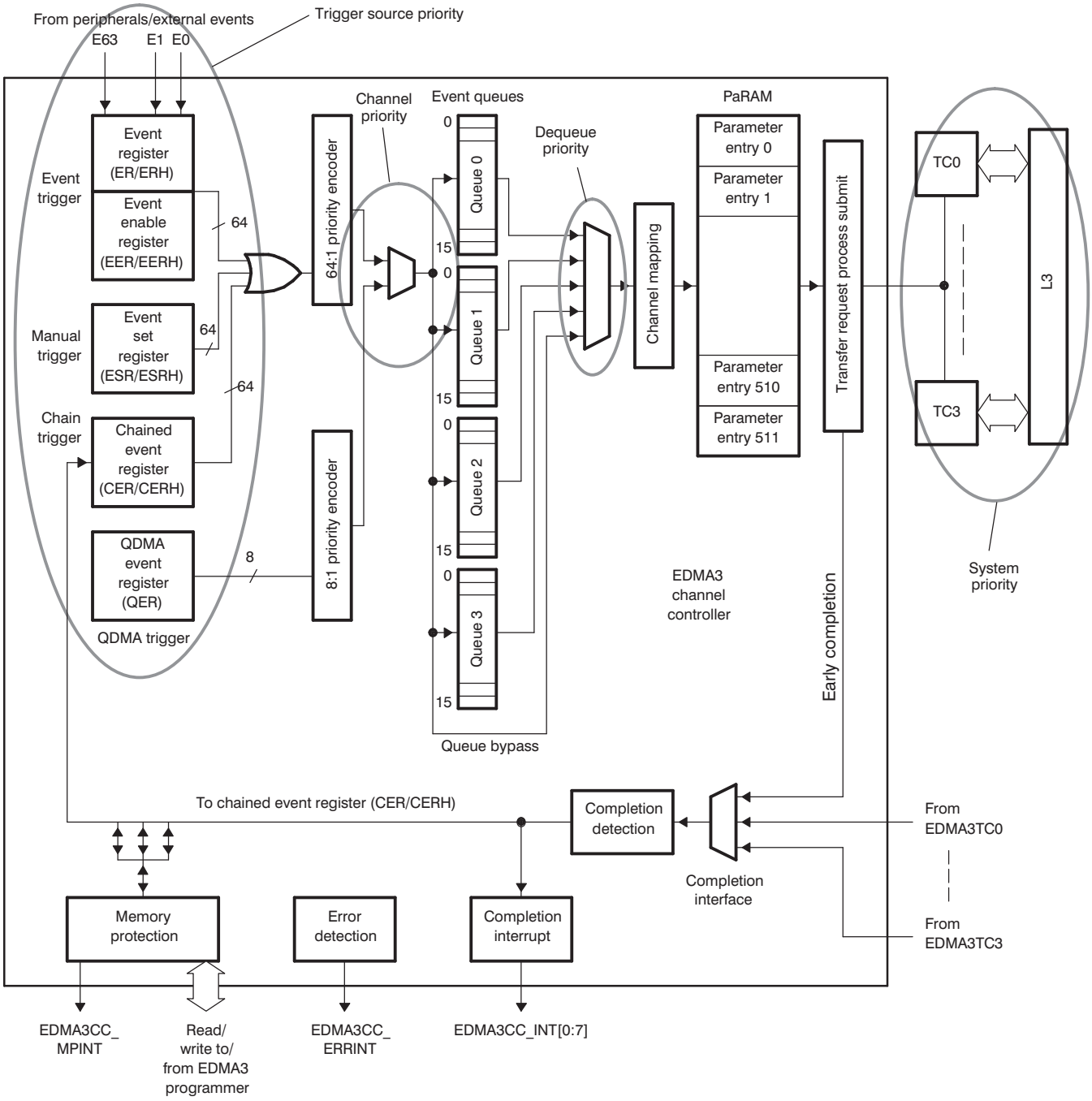
The EDMA3 controller can be idled on receiving a clock stop request from the PRCM. The requests to EDMA3CC and EDMA3TC are separate. In general, it should be verified that there are no pending activities in the EDMA3 controller

5.2.18 Emulation Considerations

During debug when using the emulator, the CPU(s) may be halted on an execute packet boundary for single-stepping, benchmarking, profiling, or other debug purposes. During an emulation halt, the EDMA3 channel controller and transfer controller operations continue. Events continue to be latched and processed and transfer requests continue to be submitted and serviced.

Since EDMA3 is involved in servicing multiple master and slave peripherals, it is not feasible to have an independent behavior of the EDMA3 for emulation halts. EDMA3 functionality would be coupled with the peripherals it is servicing, which might have different behavior during emulation halts. For example, if a McASP is halted during an emulation access (FREE = 0 and SOFT = 0 or 1 in McASP registers), the McASP stops generating the McASP receive or transmit events (REVT or XEVT) to the EDMA. From the point of view of the McASP, the EDMA3 is suspended, but other peripherals (for example, a timer) still assert events and will be serviced by the EDMA.

Figure 5-18. EDMA3 Prioritization



5.3 EDMA Transfer Examples

The EDMA3 channel controller performs a variety of transfers depending on the parameter configuration. The following sections provide a description and PaRAM configuration for some typical use case scenarios.

5.3.1 Block Move Example

The most basic transfer performed by the EDMA3 is a block move. During device operation it is often necessary to transfer a block of data from one location to another, usually between on-chip and off-chip memory.

In this example, a section of data is to be copied from external memory to internal L2 SRAM as shown in Figure 5-19. Figure 5-20 shows the parameters for this transfer.

The source address for the transfer is set to the start of the data block in external memory, and the destination address is set to the start of the data block in L2. If the data block is less than 64K bytes, the PaRAM configuration shown in Figure 5-20 holds true with the synchronization type set to A-synchronized and indexes cleared to 0. If the amount of data is greater than 64K bytes, BCNT and the B-indexes need to be set appropriately with the synchronization type set to AB-synchronized. The STATIC bit in OPT is set to prevent linking.

This transfer example may also be set up using QDMA. For successive transfer submissions, of a similar nature, the number of cycles used to submit the transfer are fewer depending on the number of changing transfer parameters. You may program the QDMA trigger word to be the highest numbered offset in the PaRAM set that undergoes change.

Figure 5-19. Block Move Example

Figure 5-20. Block Move Example PaRAM Configuration

(a) EDMA Parameters

Parameter Contents	
0010 0008h	
Channel Source Address (SRC)	
0001h	0100h
Channel Destination Address (DST)	
0000h	0000h
0000h	FFFFh
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	1	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

5.3.2 Subframe Extraction Example

The EDMA3 can efficiently extract a small frame of data from a larger frame of data. By performing a 2D-to-1D transfer, the EDMA3 retrieves a portion of data for the CPU to process. In this example, a 640 × 480-pixel frame of video data is stored in external memory. Each pixel is represented by a 16-bit halfword. The CPU extracts a 16 × 12-pixel subframe of the image for processing. To facilitate more efficient processing time by the CPU, the EDMA3 places the subframe in internal L2 SRAM. Figure 5-21 shows the transfer of a subframe from external memory to L2. Figure 5-22 shows the parameters for this transfer.

The same PaRAM entry options are used for QDMA channels, as well as DMA channels. The STATIC bit in OPT is set to prevent linking. For successive transfers, only changed parameters need to be programmed before triggering the channel.

Figure 5-21. Subframe Extraction Example

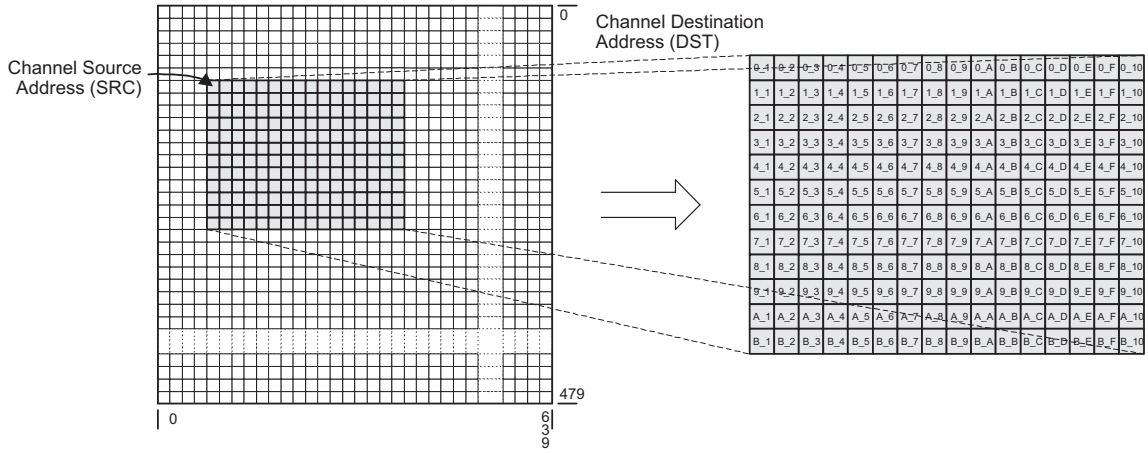


Figure 5-22. Subframe Extraction Example PaRAM Configuration

(a) EDMA Parameters

Parameter Contents	
0010 000Ch	
Channel Source Address (SRC)	
000Ch	0020h
Channel Destination Address (DST)	
0020h	0500h
0000h	FFFFh
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	1	1	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

5.3.3 Data Sorting Example

Many applications require the use of multiple data arrays; it is often desirable to have the arrays arranged such that the first elements of each array are adjacent, the second elements are adjacent, and so on. Often this is not how the data is presented to the device. Either data is transferred via a peripheral with the data arrays arriving one after the other or the arrays are located in memory with each array occupying a portion of contiguous memory spaces. For these instances, the EDMA3 can reorganize the data into the desired format. Figure 5-23 shows the data sorting.

To determine the parameter set values, the following need to be considered:

- ACNT - Program this to be the size in bytes of an element.
- BCNT - Program this to be the number of elements in a frame.
- CCNT - Program this to be the number of frames.
- SRCBIDX - Program this to be the size of the element or ACNT.
- DSTBIDX - CCNT × ACNT
- SRCCDX - ACNT × BCNT
- DSTCIDX - ACNT

The synchronization type needs to be AB-synchronized and the STATIC bit is 0 to allow updates to the parameter set. It is advised to use normal EDMA3 channels for sorting.

It is not possible to sort this with a single trigger event. Instead, the channel can be programmed to be chained to itself. After BCNT elements get sorted, intermediate chaining could be used to trigger the channel again causing the transfer of the next BCNT elements and so on. Figure 5-24 shows the parameter set programming for this transfer, assuming channel 0 and an element size of 4 bytes.

Figure 5-23. Data Sorting Example

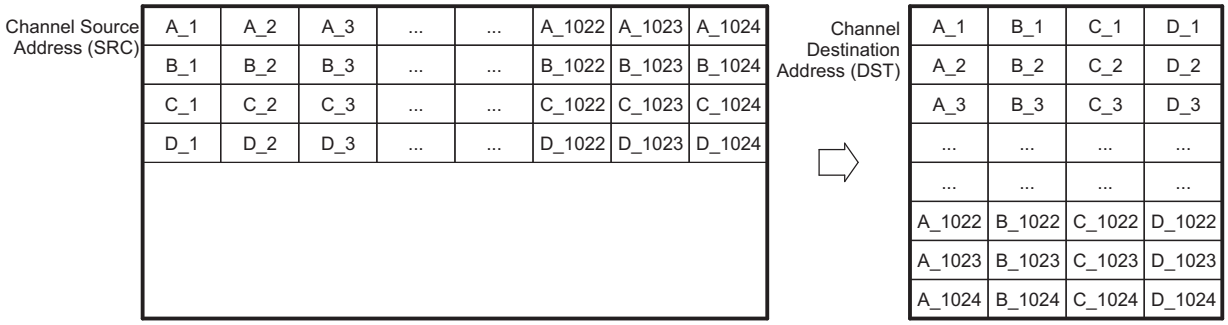


Figure 5-24. Data Sorting Example PaRAM Configuration

(a) EDMA Parameters

Parameter Contents	Parameter
0090 0004h	Channel Options Parameter (OPT)
Channel Source Address (SRC)	Channel Source Address (SRC)
0400h | 0004h	Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	Channel Destination Address (DST)
0010h | 0001h	Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX)
0000h | FFFFh	BCNT Reload (BCNTRLD) | Link Address (LINK)
0001h | 1000h	Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX)
0000h | 0004h	Reserved | Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	1	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	0	1	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

5.3.4 Peripheral Servicing Example

The EDMA3 channel controller also services peripherals in the background of CPU operation, without requiring any CPU intervention. Through proper initialization of the EDMA3 channels, they can be configured to continuously service on-chip and off-chip peripherals throughout the device operation. Each event available to the EDMA3 has its own dedicated channel, and all channels operate simultaneously. The only requirements are to use the proper channel for a particular transfer and to enable the channel event in the event enable register (EER). When programming an EDMA3 channel to service a peripheral, it is necessary to know how data is to be presented to the processor. Data is always provided with some kind of synchronization event as either one element per event (non-bursting) or multiple elements per event (bursting).

5.3.4.1 Non-bursting Peripherals

Non-bursting peripherals include the on-chip multichannel audio serial port (McASP) and many external devices, such as codecs. Regardless of the peripheral, the EDMA3 channel configuration is the same.

The McASP transmit and receive data streams are treated independently by the EDMA3. The transmit and receive data streams can have completely different counts, data sizes, and formats. Figure 5-25 shows servicing incoming McASP data.

To transfer the incoming data stream to its proper location in DDR memory, the EDMA3 channel must be set up for a 1D-to-1D transfer with A-synchronization. Because an event (AREVT) is generated for every word as it arrives, it is necessary to have the EDMA3 issue the transfer request for each element individually. Figure 5-26 shows the parameters for this transfer. The source address of the EDMA3 channel is set to the data port address(DAT) for McASP, and the destination address is set to the start of the data block in DDR. Because the address of serializer buffer is fixed, the source B index is cleared to 0 (no modification) and the destination B index is set to 01b (increment).

Based on the premise that serial data is typically a high priority, the EDMA3 channel should be programmed to be on queue 0.

Figure 5-25. Servicing Incoming McASP Data Example

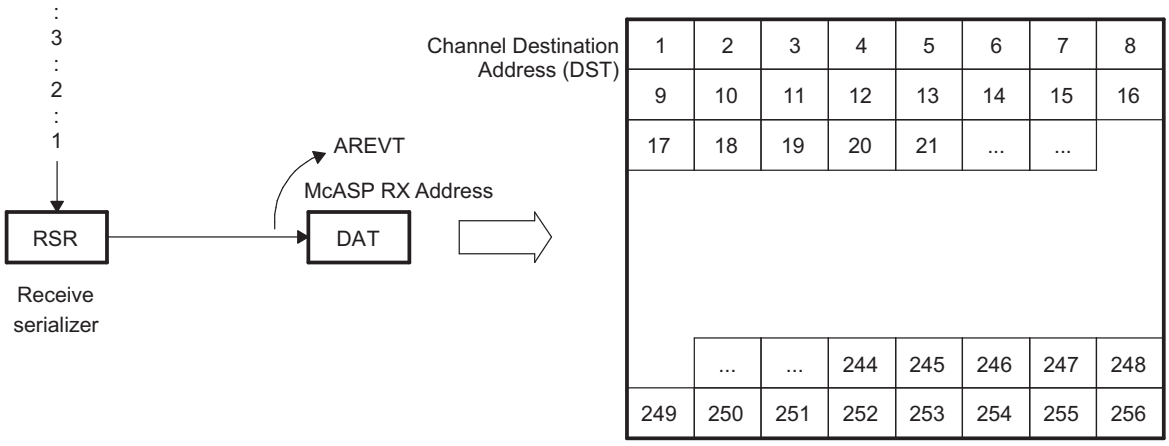


Figure 5-26. Servicing Incoming McASP Data Example PaRAM Configuration

(a) EDMA Parameters

Parameter Contents	Parameter
0010 0000h	Channel Options Parameter (OPT)
McASP RX Address	Channel Source Address (SRC)
0100h / 0001h	Count for 2nd Dimension (BCNT) / Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	Channel Destination Address (DST)
0001h / 0000h	Destination BCNT Index (DSTBIDX) / Source BCNT Index (SRCBIDX)
0000h / FFFFh	BCNT Reload (BCNTRLD) / Link Address (LINK)
0000h / 0000h	Destination CCNT Index (DSTCIDX) / Source CCNT Index (SRCCIDX)
0000h / 0004h	Reserved / Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content

31	30–28	27–24	23	22	21	20	19–18	17–16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15–12	11	10–8	7–4	3	2	1	0
0000	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

5.3.4.2 Bursting Peripherals

Higher bandwidth applications require that multiple data elements be presented to the processor core for every synchronization event. This frame of data can either be from multiple sources that are working simultaneously or from a single high-throughput peripheral that streams data to/from the processor.

In this example, a port is receiving a video frame from a camera and presenting it to the processor one array at a time. The video image is 640 × 480 pixels, with each pixel represented by a 16-bit element. The image is to be stored in external memory. Figure 5-27 shows this example.

To transfer data from an external peripheral to an external buffer one array at a time based on EVT*n* , channel *n* must be configured. Due to the nature of the data (a video frame made up of arrays of pixels) the destination is essentially a 2D entity. Figure 5-28 shows the parameters to service the incoming data with a 1D-to-2D transfer using AB-synchronization. The source address is set to the location of the video framer peripheral, and the destination address is set to the start of the data buffer. Because the input address is static, the SRCBIDX is 0 (no modification to the source address). The destination is made up of arrays of contiguous, linear elements; therefore, the DSTBIDX is set to pixel size, 2 bytes. ANCT is equal to the pixel size, 2 bytes. BCNT is set to the number of pixels in an array, 640. CCNT is equal to the total number of arrays in the block, 480. SRCCIDX is 0 because the source address undergoes no increment. The DSTCIDX is equal to the difference between the starting addresses of each array. Because a pixel is 16 bits (2 bytes), DSTCIDX is equal to 640 × 2.

Figure 5-27. Servicing Peripheral Burst Example

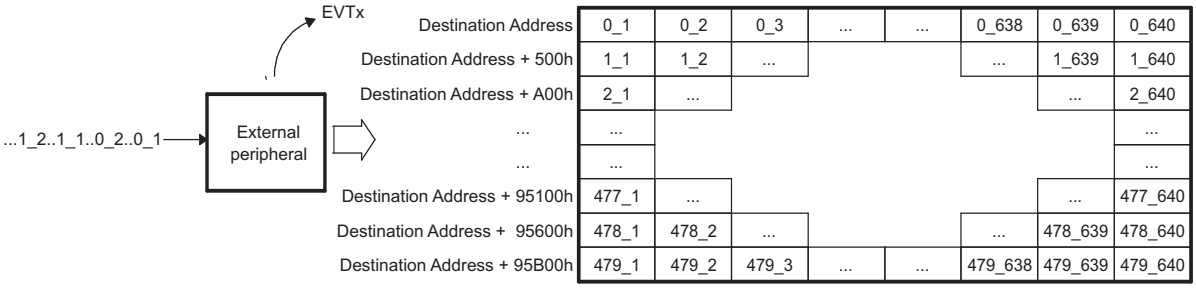


Figure 5-28. Servicing Peripheral Burst Example PaRAM Configuration

(a) EDMA Parameters

Parameter Contents	Parameter
0010 0004h	Channel Options Parameter (OPT)
Channel Source Address	Channel Source Address (SRC)
0280h | 0002h	Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT)
Channel Destination Address	Channel Destination Address (DST)
0002h | 0000h	Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX)
0000h | FFFFh	BCNT Reload (BCNTRLD) | Link Address (LINK)
0500h | 0000h	Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX)
0000h | 01E0h	Reserved | Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	0	1	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

5.3.4.3 Continuous Operation

Configuring an EDMA3 channel to receive a single frame of data is useful, and is applicable to some systems. A majority of the time, however, data is going to be continuously transmitted and received throughout the entire operation of the processor. In this case, it is necessary to implement some form of linking such that the EDMA3 channels continuously reload the necessary parameter sets. In this example, McASP is configured to transmit and receive data on a T1 array. To simplify the example, only two channels are active for both transmit and receive data streams. Each channel receives packets of 128 elements. The packets are transferred from the serial port to internal memory and from internal memory to the serial port, as shown Figure 5-29.

The McASP generates AREVT for every element received and generates AXEVT for every element transmitted. To service the data streams, them DMA channels associated with the McASP (channels 12 and 15) must be setup for 1D-to-1D transfers with A-synchronization.

Figure 5-30 shows the parameter entries for the channel for these transfers. To service the McASP continuously throughout, the channels must be linked to a duplicate PaRAM set in the PaRAM. After all frames have been transferred, the EDMA3 channels reload and continue. Figure 5-31 shows the reload parameters for the channel.

5.3.4.3.1 Receive Channel

EDMA3 channel 15 services the incoming data stream of McASP. The source address is set to that of the receive serializer buffer, and the destination address is set to the first element of the data block. Because there are two data channels being serviced, A and B, they are to be located separately within the L2 SRAM.

To facilitate continuous operation, a copy of the PaRAM set for the channel is placed in PaRAM set 64. The LINK option is set and the link address is provided in the PaRAM set. Upon exhausting the channel 15 parameter set, the parameters located at the link address are loaded into the channel 15 parameter set and operation continues. This function continues throughout device operation until halted by the CPU.

5.3.4.3.2 Transmit Channel

EDMA3 channel 12 services the outgoing data stream of McASP. In this case the destination address needs no update, hence, the parameter set changes accordingly. Linking is also used to allow continuous operation by the EDMA3 channel, with duplicate PaRAM set entries at PaRAM set 65.

Figure 5-29. Servicing Continuous McASP Data Example

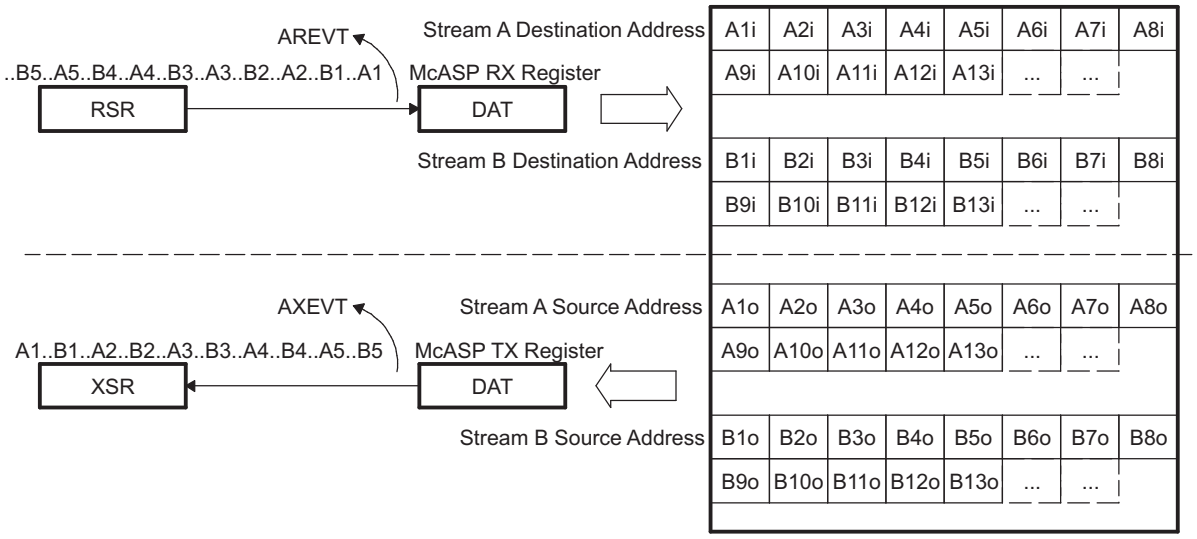


Figure 5-30. Servicing Continuous McASP Data Example PaRAM Configuration

(a) EDMA Parameters for Receive Channel (PaRAM Set 15) being Linked to PaRAM Set 64

Parameter Contents	
0010 0000h	
McASP RX Register	
0080h	0001h
Channel Destination Address (DST)	
0001h	0000h
0080h	4800h
0000h	0000h
0000h	FFFFh

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content for Receive Channel (PaRAM Set 15)

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

(c) EDMA Parameters for Transmit Channel (PaRAM Set 12) being Linked to PaRAM Set 65

Parameter Contents	
0010 1000h	
Channel Source Address (SRC)	
0080h	0001h
McASP TX Register	
0000h	0001h
0080h	4860h
0000h	0000h
0000h	FFFFh

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(d) Channel Options Parameter (OPT) Content for Transmit Channel (PaRAM Set 12)

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0001	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

Figure 5-31. Servicing Continuous McASP Data Example Reload PaRAM Configuration

(a) EDMA Reload Parameters (PaRAM Set 64) for Receive Channel

Parameter Contents	
0010 0000h	
McASP RX Register	
0080h	0001h
Channel Destination Address (DST)	
0001h	0000h
0080h	4800h
0000h	0000h
0000h	FFFFh

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content for Receive Channel (PaRAM Set 64)

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

(c) EDMA Reload Parameters (PaRAM Set 65) for Transmit Channel

Parameter Contents	
0010 1000h	
Channel Source Address (SRC)	
0080h	0001h
McASP TX Register	
0000h	0001h
0080h	4860h
0000h	0000h
0000h	FFFFh

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(d) Channel Options Parameter (OPT) Content for Transmit Channel (PaRAM Set 65)

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0001	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

5.3.4.4 Ping-Pong Buffering

Although the previous configuration allows the EDMA3 to service a peripheral continuously, it presents a number of restrictions to the CPU. Because the input and output buffers are continuously being filled/emptied, the CPU must match the pace of the EDMA3 very closely to process the data. The EDMA3 receive data must always be placed in memory before the CPU accesses it, and the CPU must provide the output data before the EDMA3 transfers it. Though not impossible, this is an unnecessary challenge. It is particularly difficult in a 2-level cache scheme.

Ping-pong buffering is a simple technique that allows the CPU activity to be distanced from the EDMA3 activity. This means that there are multiple (usually two) sets of data buffers for all incoming and outgoing data streams. While the EDMA3 transfers the data into and out of the ping buffers, the CPU manipulates the data in the pong buffers. When both CPU and EDMA3 activity completes, they switch. The EDMA3 then writes over the old input data and transfers the new output data. Figure 5-32 shows the ping-pong scheme for this example.

To change the continuous operation example, such that a ping-pong buffering scheme is used, the EDMA3 channels need only a moderate change. Instead of one parameter set, there are two; one for transferring data to/from the ping buffers and one for transferring data to/from the pong buffers. As soon as one transfer completes, the channel loads the PaRAM set for the other and the data transfers continue. Figure 5-33 shows the EDMA3 channel configuration required.

Each channel has two parameter sets, ping and pong. The EDMA3 channel is initially loaded with the ping parameters (Figure 5-33). The link address for the ping set is set to the PaRAM offset of the pong parameter set (Figure 5-34). The link address for the pong set is set to the PaRAM offset of the ping parameter set (Figure 5-35). The channel options, count values, and index values are all identical between the ping and pong parameters for each channel. The only differences are the link address provided and the address of the data buffer.

5.3.4.4.1 Synchronization with the CPU

To utilize the ping-pong buffering technique, the system must signal the CPU when to begin to access the new data set. After the CPU finishes processing an input buffer (ping), it waits for the EDMA3 to complete before switching to the alternate (pong) buffer. In this example, both channels provide their channel numbers as their report word and set the TCINTEN bit to generate an interrupt after completion. When channel 15 fills an input buffer, the E15 bit in the interrupt pending register (IPR) is set; when channel 12 empties an output buffer, the E12 bit in IPR is set. The CPU must manually clear these bits. With the channel parameters set, the CPU polls IPR to determine when to switch. The EDMA3 and CPU could alternatively be configured such that the channel completion interrupts the CPU. By doing this, the CPU could service a background task while waiting for the EDMA3 to complete.

Figure 5-32. Ping-Pong Buffering for McASP Data Example

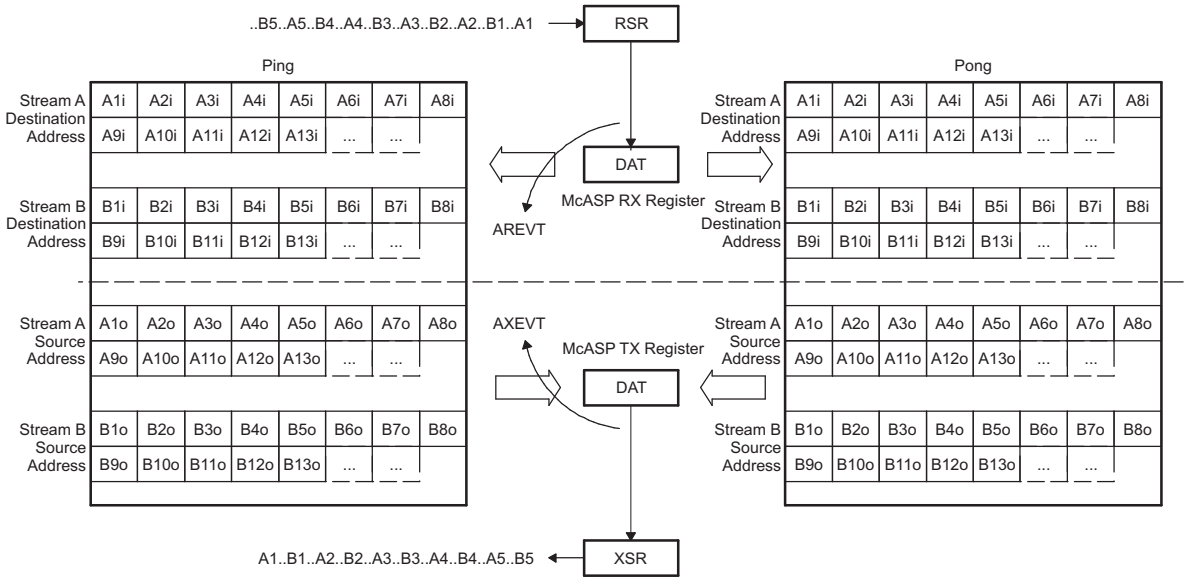


Figure 5-33. Ping-Pong Buffering for McASP Example PaRAM Configuration

(a) EDMA Parameters for Channel 15 (Using PaRAM Set 15 Linked to Pong Set 64)

Parameter Contents	
0010 D000h	
McASP RX Register	
0080h	0001h
Channel Destination Address (DST)	
0001h	0000h
0080h	4800h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content for Channel 15

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
1101	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

(c) EDMA Parameters for Channel 12 (Using PaRAM Set 12 Linked to Pong Set 65)

Parameter Contents	
0010 C000h	
Channel Source Address (SRC)	
0080h	0001h
McASP TX Register	
0000h	0001h
0080h	4840h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(d) Channel Options Parameter (OPT) Content for Channel 12

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
1100	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

Figure 5-34. Ping-Pong Buffering for McASP Example Pong PaRAM Configuration

(a) EDMA Pong Parameters for Channel 15 at Set 64 Linked to Set 65

Parameter Contents		Parameter	
0010 D000h		Channel Options Parameter (OPT)	
McASP RX Register		Channel Source Address (SRC)	
0080h	0001h	Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)		Channel Destination Address (DST)	
0001h	0000h	Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
0080h	4820h	BCNT Reload (BCNTRLD)	Link Address (LINK)
0000h	0000h	Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
0000h	0001h	Reserved	Count for 3rd Dimension (CCNT)

(b) EDMA Pong Parameters for Channel 12 at Set 66 Linked to Set 67

Parameter Contents		Parameter	
0010 C000h		Channel Options Parameter (OPT)	
Channel Source Address (SRC)		Channel Source Address (SRC)	
0080h	0001h	Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
McASP TX Register		Channel Destination Address (DST)	
0000h	0001h	Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
0080h	4860h	BCNT Reload (BCNTRLD)	Link Address (LINK)
0000h	0000h	Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
0000h	0001h	Reserved	Count for 3rd Dimension (CCNT)

Figure 5-35. Ping-Pong Buffering for McASP Example Ping PaRAM Configuration

(a) EDMA Ping Parameters for Channel 15 at Set 65 Linked to Set 64

Parameter Contents		Parameter	
0010 D000h		Channel Options Parameter (OPT)	
McASP RX Register		Channel Source Address (SRC)	
0080h	0001h	Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)		Channel Destination Address (DST)	
0001h	0000h	Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
0080h	4800h	BCNT Reload (BCNTRLD)	Link Address (LINK)
0000h	0000h	Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
0000h	0001h	Reserved	Count for 3rd Dimension (CCNT)

(b) EDMA Ping Parameters for Channel 12 at Set 67 Linked to Set 66

Parameter Contents		Parameter	
0010 C000h		Channel Options Parameter (OPT)	
Channel Source Address (SRC)		Channel Source Address (SRC)	
0080h	0001h	Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
McASP TX Register		Channel Destination Address (DST)	
0000h	0001h	Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
0080h	4840h	BCNT Reload (BCNTRLD)	Link Address (LINK)
0000h	0000h	Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
0000h	0001h	Reserved	Count for 3rd Dimension (CCNT)

5.3.4.5 Transfer Chaining Examples

The following examples explain the intermediate transfer complete chaining function.

5.3.4.5.1 Servicing Input/Output FIFOs with a Single Event

Many systems require the use of a pair of external FIFOs that must be serviced at the same rate. One FIFO buffers data input, and the other buffers data output. The EDMA3 channels that service these FIFOs can be set up for AB-synchronized transfers. While each FIFO is serviced with a different set of parameters, both can be signaled from a single event. For example, an external interrupt pin can be tied to the status flags of one of the FIFOs. When this event arrives, the EDMA3 needs to perform servicing for both the input and output streams. Without the intermediate transfer complete chaining feature this would require two events, and thus two external interrupt pins. The intermediate transfer complete chaining feature allows the use of a single external event (for example, a GPIO event). Figure 5-36 shows the EDMA3 setup and illustration for this example.

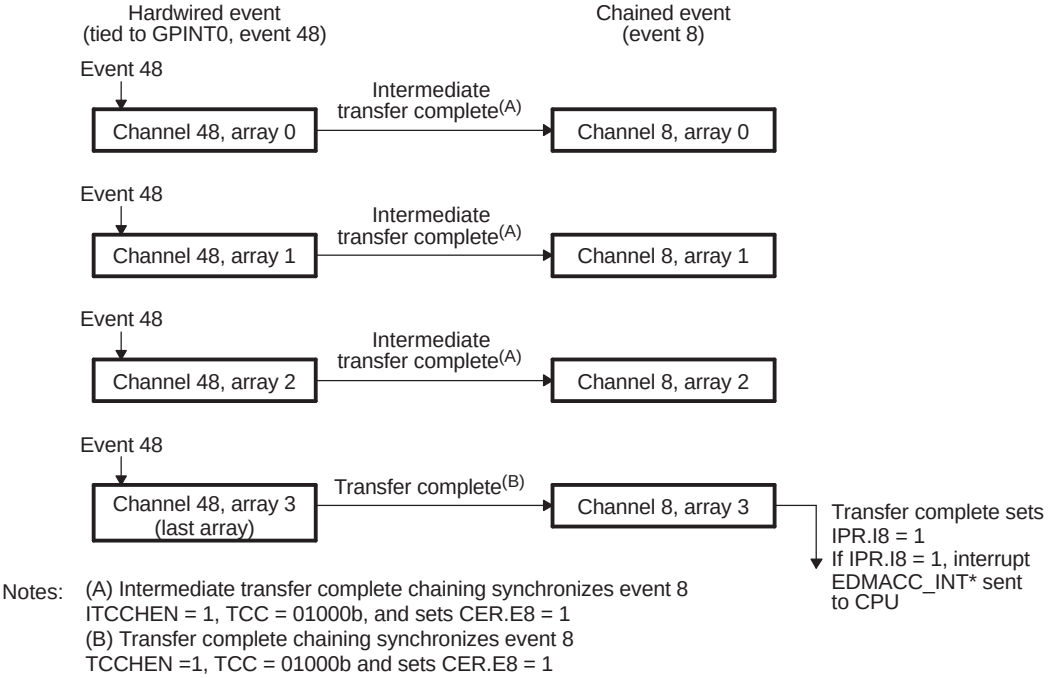
A GPIO event (in this case, GPINT0) triggers an array transfer. Upon completion of each intermediate array transfer of channel 48, intermediate transfer complete chaining sets the E8 bit (specified by TCC of 8) in the chained event register (CER) and provides a synchronization event to channel 8. Upon completion of the last array transfer of channel 48, transfer complete chaining—not intermediate transfer complete chaining—sets the E8 bit in CER (specified by TCCMODE:TCC) and provides a synchronization event to channel 8. The completion of channel 8 sets the I8 bit (specified by TCCMODE:TCC) in the interrupt pending register (IPR), which can generate an interrupt to the CPU, if the I8 bit in the interrupt enable register (IER) is set.

5.3.4.5.2 Breaking Up Large Transfers with Intermediate Chaining

Another feature of intermediate transfer chaining (ITCCHEN) is for breaking up large transfers. A large transfer may lock out other transfers of the same priority level for the duration of the transfer. For example, a large transfer on queue 0 from the internal memory to the external memory using the EMIF may starve other EDMA3 transfers on the same queue. In addition, this large high-priority transfer may prevent the EMIF for a long duration to service other lower priority transfers. When a large transfer is considered to be high priority, it should be split into multiple smaller transfers. Figure 5-37 shows the EDMA3 setup and illustration of an example single large block transfer.

The intermediate transfer chaining enable (ITCCHEN) provides a method to break up a large transfer into smaller transfers. For example, to move a single large block of memory (16K bytes), the EDMA3 performs an A-synchronized transfer. The element count is set to a reasonable value, where reasonable derives from the amount of time it would take to move this smaller amount of data. Assume 1 Kbyte is a reasonable small transfer in this example. The EDMA3 is set up to transfer 16 arrays of 1 Kbyte elements, for a total of 16K byte elements. The TCC field in the channel options parameter (OPT) is set to the same value as the channel number and ITCCHEN are set. In this example, EDMA3 channel 25 is used and TCC is also set to 25. The TCINTEN may also be set to trigger interrupt 25 when the last 1 Kbyte array is transferred. The CPU starts the EDMA3 transfer by writing to the appropriate bit of the event set register (ESR.E25). The EDMA3 transfers the first 1 Kbyte array. Upon completion of the first array, intermediate transfer complete code chaining generates a synchronization event to channel 25, a value specified by the TCC field. This intermediate transfer completion chaining event causes EDMA3 channel 25 to transfer the next 1 Kbyte array. This process continues until the transfer parameters are exhausted, at which point the EDMA3 has completed the 16K byte transfer. This method breaks up a large transfer into smaller packets, thus providing natural time slices in the transfer such that other events may be processed. Figure 5-38 shows the EDMA3 setup and illustration of the broken up smaller packet transfers.

Figure 5-36. Intermediate Transfer Completion Chaining Example



Notes: (A) Intermediate transfer complete chaining synchronizes event 8 ITCCHEN = 1, TCC = 01000b, and sets CER.E8 = 1
(B) Transfer complete chaining synchronizes event 8 TCCHEN =1, TCC = 01000b and sets CER.E8 = 1

Setup

Channel 48 parameters for chaining

- ☐ Enable transfer complete chaining:
  OPT.TCCHEN = 1
  OPT.TCC = 01000b
- ☐ Enable intermediate transfer complete chaining:
  OPT.ITCCHEN = 1
  OPT.TCC = 01000b

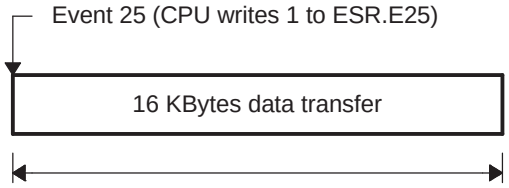
Channel 8 parameters for chaining

- ☐ Enable transfer complete chaining:
  OPT.TCINTEN = 1
  OPT.TCC = 01000b
- ☐ Disable intermediate transfer complete chaining:
  OPT.ITCCHEN = 0

Event enable register (EER)

- ☐ Enable channel 48
  EER.E48 = 1

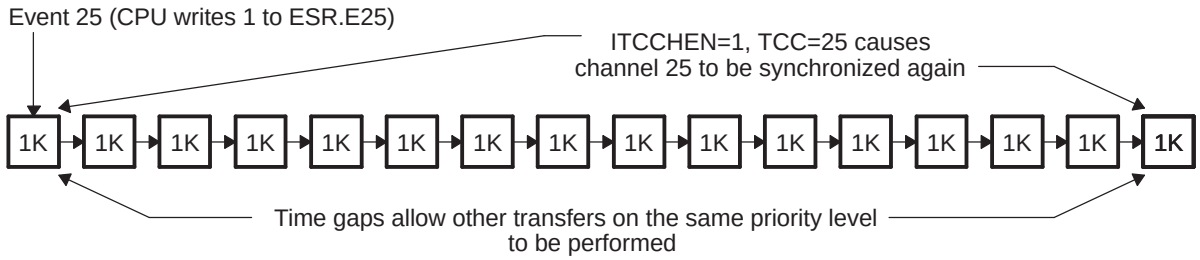
Figure 5-37. Single Large Block Transfer Example



EDMA3 channel 25 setup

ACNT = 16384
BCNT = 1
CCNT = 1
OPT.ITCINTEN = 0
OPT.TCC = Don't care
1D transfer of 16 KByte elements

Figure 5-38. Smaller Packet Data Transfers Example



EDMA3 channel 25 setup

ACNT = 1024
BCNT = 16
CCNT = 1
OPT.SYNCDIM = A SYNC
OPT.ITCCHEN = 1
OPT.TCINTEN = 1
OPT.TCC = 25

5.4 EDMA3 Registers

This section discusses the registers of the EDMA3 controller.

5.4.1 EDMA3 Channel Controller Control (EDMA3CC) Registers

Table 5-19 lists the memory-mapped registers for the EDMA3 channel controller (EDMACC). For the base address of these registers, see Table 1-11. All other register offset addresses not listed in Table 5-19 should be considered as reserved locations and the register contents should not be modified.

Table 5-19. EDMA3 Channel Controller Control (EDMA3CC) Registers

Offset	Acronym	Register Description	Section
00h	PID	Peripheral Identification Register	Section 5.4.1.1.1
04h	CCCFG	EDMA3CC Configuration Register	Section 5.4.1.1.2
0100h-01FCh	DCHMAP0-63	DMA Channel 0-63 Mapping Registers	Section 5.4.1.1.3
0200h	QCHMAP0	QDMA Channel 0 Mapping Register	Section 5.4.1.1.4
0204h	QCHMAP1	QDMA Channel 1 Mapping Register	Section 5.4.1.1.4
0208h	QCHMAP2	QDMA Channel 2 Mapping Register	Section 5.4.1.1.4
020Ch	QCHMAP3	QDMA Channel 3 Mapping Register	Section 5.4.1.1.4
0210h	QCHMAP4	QDMA Channel 4 Mapping Register	Section 5.4.1.1.4
0214h	QCHMAP5	QDMA Channel 5 Mapping Register	Section 5.4.1.1.4
0218h	QCHMAP6	QDMA Channel 6 Mapping Register	Section 5.4.1.1.4
021Ch	QCHMAP7	QDMA Channel 7 Mapping Register	Section 5.4.1.1.4
0240h	DMAQNUM0	DMA Queue Number Register 0	Section 5.4.1.1.5
0244h	DMAQNUM1	DMA Queue Number Register 1	Section 5.4.1.1.5
0248h	DMAQNUM2	DMA Queue Number Register 2	Section 5.4.1.1.5
024Ch	DMAQNUM3	DMA Queue Number Register 3	Section 5.4.1.1.5
0250h	DMAQNUM4	DMA Queue Number Register 4	Section 5.4.1.1.5
0254h	DMAQNUM5	DMA Queue Number Register 5	Section 5.4.1.1.5
0258h	DMAQNUM6	DMA Queue Number Register 6	Section 5.4.1.1.5
025Ch	DMAQNUM7	DMA Queue Number Register 7	Section 5.4.1.1.5
0260h	QDMAQNUM	QDMA Queue Number Register	Section 5.4.1.1.6
0284h	QUEPRI	Queue Priority Register	Section 5.4.1.1.7
0300h	EMR	Event Missed Register	Section 5.4.1.2.1
0304h	EMRH	Event Missed Register High	Section 5.4.1.2.1
0308h	EMCR	Event Missed Clear Register	Section 5.4.1.2.2
030Ch	EMCRH	Event Missed Clear Register High	Section 5.4.1.2.2
0310h	QEMR	QDMA Event Missed Register	Section 5.4.1.2.3
0314h	QEMCR	QDMA Event Missed Clear Register	Section 5.4.1.2.4
0318h	CCERR	EDMA3CC Error Register	Section 5.4.1.2.5
031Ch	CCERRCLR	EDMA3CC Error Clear Register	Section 5.4.1.2.6
0320h	EEVAL	Error Evaluate Register	Section 5.4.1.2.7
0340h	DRAE0	DMA Region Access Enable Register for Region 0	Section 5.4.1.3.1
0344h	DRAEH0	DMA Region Access Enable Register High for Region 0	Section 5.4.1.3.1
0348h	DRAE1	DMA Region Access Enable Register for Region 1	Section 5.4.1.3.1
034Ch	DRAEH1	DMA Region Access Enable Register High for Region 1	Section 5.4.1.3.1
0350h	DRAE2	DMA Region Access Enable Register for Region 2	Section 5.4.1.3.1
0354h	DRAEH2	DMA Region Access Enable Register High for Region 2	Section 5.4.1.3.1
0358h	DRAE3	DMA Region Access Enable Register for Region 3	Section 5.4.1.3.1
035Ch	DRAEH3	DMA Region Access Enable Register High for Region 3	Section 5.4.1.3.1
0360h	DRAE4	DMA Region Access Enable Register for Region 4	Section 5.4.1.3.1
0364h	DRAEH4	DMA Region Access Enable Register High for Region 4	Section 5.4.1.3.1

Table 5-19. EDMA3 Channel Controller Control (EDMA3CC) Registers (continued)

Offset	Acronym	Register Description	Section
0368h	DRAE5	DMA Region Access Enable Register for Region 5	Section 5.4.1.3.1
036Ch	DRAEH5	DMA Region Access Enable Register High for Region 5	Section 5.4.1.3.1
0370h	DRAE6	DMA Region Access Enable Register for Region 6	Section 5.4.1.3.1
0374h	DRAEH6	DMA Region Access Enable Register High for Region 6	Section 5.4.1.3.1
0378h	DRAE7	DMA Region Access Enable Register for Region 7	Section 5.4.1.3.1
037Ch	DRAEH7	DMA Region Access Enable Register High for Region 7	Section 5.4.1.3.1
0380h-039Ch	QRAE0-7	QDMA Region Access Enable Registers for Region 0-7	Section 5.4.1.3.2
0400h-04FCh	Q0E0-Q3E15	Event Queue Entry Registers Q0E0-Q3E15	Section 5.4.1.4.1
0600h-060Ch	QSTAT0-3	Queue Status Registers 0-3	Section 5.4.1.4.2
0620h	QWMTHRA	Queue Watermark Threshold A Register	Section 5.4.1.4.3
0640h	CCSTAT	EDMA3CC Status Register	Section 5.4.1.4.4
0800h	MPFAR	Memory Protection Fault Address Register	Section 5.4.1.5.1
0804h	MPFSR	Memory Protection Fault Status Register	Section 5.4.1.5.2
0808h	MPFCR	Memory Protection Fault Command Register	Section 5.4.1.5.3
080Ch	MPPAG	Memory Protection Page Attribut Register Global	Section 5.4.1.5.4
0810h-082Ch	MPPA0-7	Memory Protection Page Attribute Registers 0-7	Section 5.4.1.5.4
1000h	ER	Event Register	Section 5.4.1.6.1
1004h	ERH	Event Register High	Section 5.4.1.6.1
1008h	ECR	Event Clear Register	Section 5.4.1.6.2
100Ch	ECRH	Event Clear Register High	Section 5.4.1.6.2
1010h	ESR	Event Set Register	Section 5.4.1.6.3
1014h	ESRH	Event Set Register High	Section 5.4.1.6.3
1018h	CER	Chained Event Register	Section 5.4.1.6.4
101Ch	CERH	Chained Event Register High	Section 5.4.1.6.4
1020h	EER	Event Enable Register	Section 5.4.1.6.5
1024h	EERH	Event Enable Register High	Section 5.4.1.6.5
1028h	EECR	Event Enable Clear Register	Section 5.4.1.6.6
102Ch	EECRH	Event Enable Clear Register High	Section 5.4.1.6.6
1030h	EESR	Event Enable Set Register	Section 5.4.1.6.7
1034h	EESRH	Event Enable Set Register High	Section 5.4.1.6.7
1038h	SER	Secondary Event Register	Section 5.4.1.6.8
103Ch	SERH	Secondary Event Register High	Section 5.4.1.6.8
1040h	SECR	Secondary Event Clear Register	Section 5.4.1.6.9
1044h	SECRH	Secondary Event Clear Register High	Section 5.4.1.6.9
1050h	IER	Interrupt Enable Register	Section 5.4.1.7.1
1054h	IERH	Interrupt Enable Register High	Section 5.4.1.7.1
1058h	IECR	Interrupt Enable Clear Register	Section 5.4.1.7.2
105Ch	IECRH	Interrupt Enable Clear Register High	Section 5.4.1.7.2
1060h	IESR	Interrupt Enable Set Register	Section 5.4.1.7.3
1064h	IESRH	Interrupt Enable Set Register High	Section 5.4.1.7.3
1068h	IPR	Interrupt Pending Register	Section 5.4.1.7.4
106Ch	IPRH	Interrupt Pending Register High	Section 5.4.1.7.4
1070h	ICR	Interrupt Clear Register	Section 5.4.1.7.5
1074h	ICRH	Interrupt Clear Register High	Section 5.4.1.7.5
1078h	IEVAL	Interrupt Evaluate Register	Section 5.4.1.7.6
1080h	QER	QDMA Event Register	Section 5.4.1.8.1
1084h	QEER	QDMA Event Enable Register	Section 5.4.1.8.2

Table 5-19. EDMA3 Channel Controller Control (EDMA3CC) Registers (continued)

Offset	Acronym	Register Description	Section
1088h	QEECR	QDMA Event Enable Clear Register	Section 5.4.1.8.3
108Ch	QEESR	QDMA Event Enable Set Register	Section 5.4.1.8.4
1090h	QSER	QDMA Secondary Event Register	Section 5.4.1.8.5
1094h	QSECR	QDMA Secondary Event Clear Register	Section 5.4.1.8.6
		Shadow Region 0 Channel Registers	
2000h	ER	Event Register	
2004h	ERH	Event Register High	
2008h	ECR	Event Clear Register	
200Ch	ECRH	Event Clear Register High	
2010h	ESR	Event Set Register	
2014h	ESRH	Event Set Register High	
2018h	CER	Chained Event Register	
201Ch	CERH	Chained Event Register High	
2020h	EER	Event Enable Register	
2024h	EERH	Event Enable Register High	
2028h	EECR	Event Enable Clear Register	
202Ch	EECRH	Event Enable Clear Register High	
2030h	EESR	Event Enable Set Register	
2034h	EESRH	Event Enable Set Register High	
2038h	SER	Secondary Event Register	
203Ch	SERH	Secondary Event Register High	
2040h	SECR	Secondary Event Clear Register	
2044h	SECRH	Secondary Event Clear Register High	
2050h	IER	Interrupt Enable Register	
2054h	IERH	Interrupt Enable Register High	
2058h	IECR	Interrupt Enable Clear Register	
205Ch	IECRH	Interrupt Enable Clear Register High	
2060h	IESR	Interrupt Enable Set Register	
2064h	IESRH	Interrupt Enable Set Register High	
2068h	IPR	Interrupt Pending Register	
206Ch	IPRH	Interrupt Pending Register High	
2070h	ICR	Interrupt Clear Register	
2074h	ICRH	Interrupt Clear Register High	
2078h	IEVAL	Interrupt Evaluate Register	
2080h	QER	QDMA Event Register	
2084h	QEER	QDMA Event Enable Register	
2088h	QEECR	QDMA Event Enable Clear Register	
208Ch	QEESR	QDMA Event Enable Set Register	
2090h	QSER	QDMA Secondary Event Register	
2094h	QSECR	QDMA Secondary Event Clear Register	
2200h-2294h	-	Shadow Region 1 Channel Registers	
2400h-2494h	-	Shadow Region 2 Channel Registers	
...		...	
2E00h-2E94h	-	Shadow Channel Registers for MP Space 7	

5.4.1.1 Global Registers

5.4.1.1.1 Peripheral Identification Register (PID)

The peripheral identification register (PID) uniquely identifies the EDMA3CC and the specific revision of the EDMA3CC.

The PID is shown in Figure 5-39 and described in Table 5-20.

Figure 5-39. Peripheral ID Register (PID)

31	16	15	0
PID		PID	
R-4001h		R-1B00h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-20. Peripheral ID Register (PID) Field Descriptions

Bit	Field	Value	Description
31-0	PID	0-FFFF FFFFh	Peripheral identifier uniquely identifies the EDMA3CC and the specific revision of the EDMA3CC.

5.4.1.1.2 EDMA3CC Configuration Register (CCCFG)

The EDMA3CC configuration register (CCCFG) provides the features/resources for the EDMA3CC in a particular device.

The CCCFG is shown in Figure 5-40 and described in Table 5-21.

Figure 5-40. EDMA3CC Configuration Register (CCCFG)

31	26	25	24
Reserved		MP_EXIST	CHMAP_EXIST
R-x		R-1	R-1

23	22	21	20	19	18	16
Reserved		NUM_REGN		Reserved	NUM_EVQUE	
R-0		R-3h		R-x	R-3h	

15	14	12	11	10	8
Reserved	NUM_PAENTRY		Reserved	NUM_INTCH	
R-x	R-5h		R-x	R-4h	

7	6	4	3	2	0
Reserved	NUM_QDMACH		Reserved	NUM_DMACH	
R-x	R-4h		R-x	R-5h	

LEGEND: R = Read only; -*n* = value after reset; -x = value is indeterminate after reset

Table 5-21. EDMA3CC Configuration Register (CCCFG) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
25	MP_EXIST		Memory protection existence.
		0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		1	Memory protection logic included.
24	CHMAP_EXIST		Channel mapping existence.
		0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		1	Channel mapping logic included.
23-22	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
21-20	NUM_REGN	0-3h	Number of MP and shadow regions.
		0-2h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		3h	8 regions.
19	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
18-16	NUM_EVQUE	0-7h	Number of queues/number of TCs.
		0-2h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		3h	4 EDMA3TCs/Event Queues.
		4h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
15	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.

Table 5-21. EDMA3CC Configuration Register (CCCFG) Field Descriptions (continued)

Bit	Field	Value	Description
14-12	NUM_PAENTRY	0-7h	Number of PaRAM sets.
		0-3h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		5h	512 PaRAM sets.
		5h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
11	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
10-8	NUM_INTCH	0-7h	Number of interrupt channels.
		0-3h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		4h	64 interrupt channels.
		5h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
6-4	NUM_QDMACH	0-7h	Number of QDMA channels.
		0-1h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		4h	8 QDMA channels.
		3h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
2-0	NUM_DMACH	0-7h	Number of DMA channels.
		0-4h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		5h	64 DMA channels.
		6h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.

5.4.1.1.3 DMA Channel Map *n* Registers (DCHMAP*n*)

The DMA channel map *n* register (DCHMAP*n*) is shown in Figure 5-41 and described in Table 5-22.

Figure 5-41. DMA Channel Map *n* Registers (DCHMAP*n*)

31	16
Reserved	
R-0	

15	14	13	5	4	0
Reserved		PAENTRY		Reserved	
R-0		R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-22. DMA Channel Map *n* Registers (DCHMAP*n*) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13-5	PAENTRY	0-1FFh	Points to the PaRAM set number for DMA channel *n*.
4-0	Reserved	0	Reserved

5.4.1.1.4 QDMA Channel Map *n* Registers (QCHMAP*n*)

Each QDMA channel in EDMA3CC can be associated with any PaRAM set available on the device. Furthermore, the specific trigger word (0-7) of the PaRAM set can be programmed. The PaRAM set association and trigger word for every QDMA channel register is configurable using the QDMA channel map *n* register (QCHMAP*n*).

The QCHMAP*n* is shown in Figure 5-42 and described in Table 5-23.

NOTE: At reset the QDMA channel map registers for all QDMA channels point to PaRAM set 0. If an application makes use of both a DMA channel that points to PaRAM set 0 and any QDMA channels, ensure that QCHMAP*n* is programmed appropriately to point to a different PaRAM entry.

Figure 5-42. QDMA Channel Map *n* Registers (QCHMAP*n*)

31	16
Reserved	
R-0	

15	14	13	5	4	2	1	0
Reserved		PAENTRY		TRWORD		Reserved	
R-0		R/W-0		R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-23. QDMA Channel Map *n* Registers (QCHMAP*n*) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
13-5	PAENTRY	0-1FFh	PAENTRY points to the PaRAM set number for QDMA channel *n*.
		0-FFh	Parameter entry 0 through 511.
		100h-1FFh	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
4-2	TRWORD	0-7h	Points to the specific trigger word of the PaRAM set defined by PAENTRY. A write to the trigger word results in a QDMA event being recognized.
1-0	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.

5.4.1.1.5 DMA Channel Queue *n* Number Registers (DMAQNUM*n*)

The DMA channel queue number register (DMAQNUM*n*) allows programmability of each of the 64 DMA channels in the EDMA3CC to submit its associated synchronization event to any event queue in the EDMA3CC. At reset, all channels point to event queue 0.

The DMAQNUM*n* is shown in Figure 5-43 and described in Table 5-24. Table 5-25 shows the channels and their corresponding bits in DMAQNUM*n*.

NOTE: Because the event queues in EDMA3CC have a fixed association to the transfer controllers, that is, Q0 TRs are submitted to TC0, Q1 TRs are submitted to TC1, etc., by programming DMAQNUM*n* for a particular DMA channel *n* also dictates which transfer controller is utilized for the data movement (or which EDMA3TC receives the TR request).

Figure 5-43. DMA Channel Queue *n* Number Registers (DMAQNUM*n*)

31	30 28	27	26 24	23	22 20	19	18 16
Rsvd	E*n*	Rsvd	E*n*	Rsvd	E*n*	Rsvd	E*n*
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

15	14 12	11	10 8	7	6 4	3	2 0
Rsvd	E*n*	Rsvd	E*n*	Rsvd	E*n*	Rsvd	E*n*
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-24. DMA Channel Queue *n* Number Registers (DMAQNUM*n*) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*	0-7h	DMA queue number. Contains the event queue number to be used for the corresponding DMA channel. Programming DMAQNUM*n* for an event queue number to a value more then the number of queues available in the EDMA3CC results in undefined behavior.
		0	Event *n* is queued on Q0.
		1h	Event *n* is queued on Q1.
		2h	Event *n* is queued on Q2.
		3h	Event *n* is queued on Q3.
		4h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.

Table 5-25. Bits in DMAQNUM*n*

	Channel Number (DMAQNUM*n*)							
E*n* bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0-2	E0	E8	E16	E24	E32	E40	E48	E56
4-6	E1	E9	E17	E25	E33	E41	E49	E57
8-10	E2	E10	E18	E26	E34	E42	E50	E58
12-14	E3	E11	E19	E27	E35	E43	E51	E59
16-18	E4	E12	E20	E28	E36	E44	E52	E60
20-22	E5	E13	E21	E29	E37	E45	E53	E61
24-26	E6	E14	E22	E30	E38	E46	E54	E62
28-30	E7	E15	E23	E31	E39	E47	E55	E63

5.4.1.1.6 QDMA Channel Queue Number Register (QDMAQNUM)

The QDMA channel queue number register (QDMAQNUM) is used to program all the QDMA channels in the EDMA3CC to submit the associated QDMA event to any of the event queues in the EDMA3CC.

The QDMAQNUM is shown in Figure 5-44 and described in Table 5-26.

Figure 5-44. QDMA Channel Queue Number Register (QDMAQNUM)

31	30 28	27	26 24	23	22 20	19	18 16
Rsvd	E7	Rsvd	E6	Rsvd	E5	Rsvd	E4
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

15	14 12	11	10 8	7	6 4	3	2 0
Rsvd	E3	Rsvd	E2	Rsvd	E1	Rsvd	E0
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-26. QDMA Channel Queue Number Register (QDMAQNUM) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*	0-7h	QDMA queue number. Contains the event queue number to be used for the corresponding QDMA channel.
		0	Event *n* is queued on Q0.
		1h	Event *n* is queued on Q1.
		2h	Event *n* is queued on Q2.
		3h	Event *n* is queued on Q3.
		4h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.

5.4.1.1.7 Queue Priority Register (QUEPRI)

The queue priority register (QUEPRI) allows you to change the priority of the individual queues and the priority of the transfer request (TR) associated with the events queued in the queue. Because the queue to EDMA3TC mapping is fixed, programming QUEPRI essentially governs the priority of the associated transfer controller(s) read/write commands with respect to the other bus masters in the device. You can modify the EDMA3TC priority to obtain the desired system performance.

The QUEPRI is shown in Figure 5-45 and described in Table 5-27.

Figure 5-45. Queue Priority Register (QUEPRI)

31															16
Reserved															
R-0															

15	14	12	11	10	8	7	6	4	3	2	0
Rsvd	PRIQ3		Rsvd	PRIQ2		Rsvd	PRIQ1		Rsvd	PRIQ0	
R-0	R/W-0		R-0	R/W-0			R/W-0		R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-27. Queue Priority Register (QUEPRI) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
14-12	PRIQ3	0-7h	Priority level for queue 3. Dictates the priority level used by TC3 relative to other masters in the device. A value of 0 means highest priority and a value of 7 means lowest priority.
11	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
10-8	PRIQ2	0-7h	Priority level for queue 2. Dictates the priority level used by TC2 relative to other masters in the device. A value of 0 means highest priority and a value of 7 means lowest priority.
7	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
6-4	PRIQ1	0-7h	Priority level for queue 1. Dictates the priority level used by TC1 relative to other masters in the device. A value of 0 means highest priority and a value of 7 means lowest priority.
3	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
2-0	PRIQ0	0-7h	Priority level for queue 0. Dictates the priority level used by TC0 relative to other masters in the device. A value of 0 means highest priority and a value of 7 means lowest priority.

5.4.1.2 Error Registers

The EDMA3CC contains a set of registers that provide information on missed DMA and/or QDMA events, and instances when event queue thresholds are exceeded. If any of the bits in these registers is set, it results in the EDMA3CC generating an error interrupt.

5.4.1.2.1 Event Missed Registers (EMR/EMRH)

For a particular DMA channel, if a second event is received prior to the first event getting cleared/serviced, the bit corresponding to that channel is set/asserted in the event missed registers (EMR/EMRH). All trigger types are treated individually, that is, manual triggered (ESR/ESRH), chain triggered (CER/CERH), and event triggered (ER/ERH) are all treated separately. The EMR/EMRH bits for a channel are also set if an event on that channel encounters a NULL entry (or a NULL TR is serviced). If any EMR/EMRH bit is set (and all errors, including bits in other error registers (QEMR, CCERR) were previously cleared), the EDMA3CC generates an error interrupt. See Section 5.2.9.4 for details on EDMA3CC error interrupt generation.

The EMR is shown in Figure 5-46 and described in Table 5-28. The EMRH is shown in Figure 5-47 and described in Table 5-29.

Figure 5-46. Event Missed Register (EMR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-28. Event Missed Register (EMR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Channel 0-31 event missed. E*n* is cleared by writing a 1 to the corresponding bit in the event missed clear register (EMCR).
		0	No missed event.
		1	Missed event occurred.

Figure 5-47. Event Missed Register High (EMRH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E63	E62	E61	E60	E59	E58	E57	E56	E55	E54	E53	E52	E51	E50	E49	E48
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E47	E46	E45	E44	E43	E42	E41	E40	E39	E38	E37	E36	E35	E34	E33	E32
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-29. Event Missed Register High (EMRH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Channel 32–63 event missed. E*n* is cleared by writing a 1 to the corresponding bit in the event missed clear register high (EMCRH).
		0	No missed event.
		1	Missed event occurred.

5.4.1.2.2 Event Missed Clear Registers (EMCR/EMCRH)

Once a missed event is posted in the event missed registers (EMR/EMRH), the bit remains set and you need to clear the set bit(s). This is done by way of CPU writes to the event missed clear registers (EMCR/EMCRH). Writing a 1 to any of the bits clears the corresponding missed event (bit) in EMR/EMRH; writing a 0 has no effect.

The EMCR is shown in Figure 5-48 and described in Table 5-30. The EMCRH is shown in Figure 5-49 and described in Table 5-31.

Figure 5-48. Event Missed Clear Register (EMCR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-30. Event Missed Clear Register (EMCR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event missed 0-31 clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC.
		0	No effect.
		1	Corresponding missed event bit in the event missed register (EMR) is cleared (E*n* = 0).

Figure 5-49. Event Missed Clear Register High (EMCRH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E63	E62	E61	E60	E59	E58	E57	E56	E55	E54	E53	E52	E51	E50	E49	E48
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E47	E46	E45	E44	E43	E42	E41	E40	E39	E38	E37	E36	E35	E34	E33	E32
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-31. Event Missed Clear Register High (EMCRH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event missed 32–63 clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC.
		0	No effect.
		1	Corresponding missed event bit in the event missed register high (EMRH) is cleared (E*n* = 0).

5.4.1.2.3 QDMA Event Missed Register (QEMR)

For a particular QDMA channel, if two QDMA events are detected without the first event getting cleared/serviced, the bit corresponding to that channel is set/asserted in the QDMA event missed register (QEMR). The QEMR bits for a channel are also set if a QDMA event on the channel encounters a NULL entry (or a NULL TR is serviced). If any QEMR bit is set (and all errors, including bits in other error registers (EMR/EMRH, CCERR) were previously cleared), the EDMA3CC generates an error interrupt. See Section 5.2.9.4 for details on EDMA3CC error interrupt generation.

The QEMR is shown in Figure 5-50 and described in Table 5-32.

Figure 5-50. QDMA Event Missed Register (QEMR)

31	16
Reserved	
R-0	

15	7	6	5	4	3	2	1	0
Reserved	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-32. QDMA Event Missed Register (QEMR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-0	E*n*		Channel 0-7 QDMA event missed. E*n* is cleared by writing a 1 to the corresponding bit in the QDMA event missed clear register (QEMCR).
		0	No missed event.
		1	Missed event occurred.

5.4.1.2.4 QDMA Event Missed Clear Register (QEMCR)

Once a missed event is posted in the QDMA event missed registers (QEMR), the bit remains set and you need to clear the set bit(s). This is done by way of CPU writes to the QDMA event missed clear registers (QEMCR). Writing a 1 to any of the bits clears the corresponding missed event (bit) in QEMR; writing a 0 has no effect.

The QEMCR is shown in Figure 5-51 and described in Table 5-33.

Figure 5-51. QDMA Event Missed Clear Register (QEMCR)

31	16
Reserved	
R-0	

15	7	6	5	4	3	2	1	0
Reserved	E7	E6	E5	E4	E3	E2	E1	E0
R-0	w-0	w-0	w-0	w-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-33. QDMA Event Missed Clear Register (QEMCR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-0	E*n*		QDMA event missed clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC.
		0	No effect.
		1	Corresponding missed event bit in the QDMA event missed register (QEMR) is cleared (E*n* = 0).

5.4.1.2.5 EDMA3CC Error Register (CCERR)

The EDMA3CC error register (CCERR) indicates whether or not at any instant of time the number of events queued up in any of the event queues exceeds or equals the threshold/watermark value that is set in the queue watermark threshold register (QWMTHRA). Additionally, CCERR also indicates if when the number of outstanding TRs that have been programmed to return transfer completion code (TRs which have the TCINTEN or TCCHEN bit in OPT set) to the EDMA3CC has exceeded the maximum allowed value of 63. If any bit in CCERR is set (and all errors, including bits in other error registers (EMR/EMRH, QEMR) were previously cleared), the EDMA3CC generates an error interrupt. See Section 5.2.9.4 for details on EDMA3CC error interrupt generation. Once the error bits are set in CCERR, they can only be cleared by writing to the corresponding bits in the EDMA3CC error clear register (CCERRCLR).

The CCERR is shown in Figure 5-52 and described in Table 5-34.

Figure 5-52. EDMA3CC Error Register (CCERR)

31	17	16
Reserved		TCCERR
R-0		R-0

15	4	3	2	1	0
Reserved		QTHRXCD3	QTHRXCD2	QTHRXCD1	QTHRXCD0
R-0		R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-34. EDMA3CC Error Register (CCERR) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
16	TCCERR		Transfer completion code error. TCCERR is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).
		0	Total number of allowed TCCs outstanding has not been reached.
		1	Total number of allowed TCCs has been reached.
15-4	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3	QTHRXCD3		Queue threshold error for queue 3. QTHRXCD3 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).
		0	Watermark/threshold has not been exceeded.
		1	Watermark/threshold has been exceeded.
2	QTHRXCD2		Queue threshold error for queue 2. QTHRXCD2 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).
		0	Watermark/threshold has not been exceeded.
		1	Watermark/threshold has been exceeded.
1	QTHRXCD1		Queue threshold error for queue 1 . QTHRXCD1 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).
		0	Watermark/threshold has not been exceeded.
		1	Watermark/threshold has been exceeded.
0	QTHRXCD0		Queue threshold error for queue 0. QTHRXCD0 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).
		0	Watermark/threshold has not been exceeded.
		1	Watermark/threshold has been exceeded.

5.4.1.2.6 EDMA3CC Error Clear Register (CCERRCLR)

The EDMA3CC error clear register (CCERRCLR) is used to clear any error bits that are set in the EDMA3CC error register (CCERR). In addition, CCERRCLR also clears the values of some bit fields in the queue status registers (QSTAT*n*) associated with a particular event queue. Writing a 1 to any of the bits clears the corresponding bit in CCERR; writing a 0 has no effect.

The CCERRCLR is shown in Figure 5-53 and described in Table 5-35.

Figure 5-53. EDMA3CC Error Clear Register (CCERRCLR)

31	17	16
Reserved		TCCERR
W-0		W-0

15	4	3	2	1	0
Reserved		QTHRXCD3	QTHRXCD2	QTHRXCD1	QTHRXCD0
W-0		W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-35. EDMA3CC Error Clear Register (CCERRCLR) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
16	TCCERR		Transfer completion code error clear.
		0	No effect.
		1	Clears the TCCERR bit in the EDMA3CC error register (CCERR).
15-4	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3	QTHRXCD3		Queue threshold error clear for queue 3.
		0	No effect.
		1	Clears the QTHRXCD3 bit in the EDMA3CC error register (CCERR) and the WM and THRXCD bits in the queue status register 3 (QSTAT3).
2	QTHRXCD2		Queue threshold error clear for queue 2.
		0	No effect.
		1	Clears the QTHRXCD2 bit in the EDMA3CC error register (CCERR) and the WM and THRXCD bits in the queue status register 2 (QSTAT2).
1	QTHRXCD1		Queue threshold error clear for queue 1.
		0	No effect.
		1	Clears the QTHRXCD1 bit in the EDMA3CC error register (CCERR) and the WM and THRXCD bits in the queue status register 1 (QSTAT1).
0	QTHRXCD0		Queue threshold error clear for queue 0.
		0	No effect.
		1	Clears the QTHRXCD0 bit in the EDMA3CC error register (CCERR) and the WM and THRXCD bits in the queue status register 0 (QSTAT0).

5.4.1.2.7 Error Evaluation Register (EEVAL)

The EDMA3CC error interrupt is asserted whenever an error bit is set in any of the error registers (EMR/EMRH, QEMR, and CCERR). For subsequent error bits that get set, the EDMA3CC error interrupt is reasserted only when transitioning from an “all the error bits cleared” to “at least one error bit is set”. Alternatively, a CPU write of 1 to the EVAL bit in the error evaluation register (EEVAL) results in reasserting the EDMA3CC error interrupt, if there are any outstanding error bits set due to subsequent error conditions. Writes of 0 have no effect.

The EEVAL is shown in Figure 5-54 and described in Table 5-36.

Figure 5-54. Error Evaluation Register (EEVAL)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		Rsvd	EVAL
R-0		R/W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 5-36. Error Evaluation Register (EEVAL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	EVAL		Error interrupt evaluate.
		0	No effect.
		1	EDMA3CC error interrupt will be pulsed if any errors have not been cleared in any of the error registers (EMR/EMRH, QEMR, or CCERR).

5.4.1.3 Region Access Enable Registers

The region access enable register group consists of the DMA access enable registers (DRAE*m* and DRAEH*m*) and the QDMA access enable registers (QRAE*m*). Where *m* is the number of shadow regions in the EDMA3CC memory map for a device. You can configure these registers to assign ownership of DMA/QDMA channels to a particular shadow region.

5.4.1.3.1 DMA Region Access Enable for Region m (DRAEm)

The DMA region access enable register for shadow region *m* (DRAE*m*/DRAEH*m*) is programmed to allow or disallow read/write accesses on a bit-by-bit bases for all DMA registers in the shadow region *m* view of the DMA channel registers. See the EDMA3CC register memory map (Table 5-19) for a list of all the DMA channel and interrupt registers mapped in the shadow region view. Additionally, the DRAE*m*/DRAEH*m* configuration determines completion of which DMA channels will result in assertion of the shadow region *m* DMA completion interrupt (see Section 5.2.9).

The DRAE*m* is shown in Figure 5-55 and described in Table 5-37. The DRAEH*m* is shown in Figure 5-56 and described in Table 5-37.

Figure 5-55. DMA Region Access Enable Register for Region *m* (DRAE*m*)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 5-56. DMA Region Access Enable High Register for Region *m* (DRAEH*m*)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E63	E62	E61	E60	E59	E58	E57	E56	E55	E54	E53	E52	E51	E50	E49	E48
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E47	E46	E45	E44	E43	E42	E41	E40	E39	E38	E37	E36	E35	E34	E33	E32
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 5-37. DMA Region Access Enable Registers for Region M (DRAE*m*/DRAEH*m*) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		DMA region access enable for bit *n*/channel *n* in region *m*.
		0	Accesses via region *m* address space to bit *n* in any DMA channel register are not allowed. Reads return 0 on bit *n* and writes do not modify the state of bit *n*. Enabled interrupt bits for bit *n* do not contribute to the generation of a transfer completion interrupt for shadow region *m*.
		1	Accesses via region *m* address space to bit *n* in any DMA channel register are allowed. Reads return the value from bit *n* and writes modify the state of bit *n*. Enabled interrupt bits for bit *n* contribute to the generation of a transfer completion interrupt for shadow region *m*.

5.4.1.3.2 QDMA Region Access Enable Registers (QRAE*m*)

The QDMA region access enable register for shadow region *m* (QRAE*m*) is programmed to allow or disallow read/write accesses on a bit-by-bit bases for all QDMA registers in the shadow region *m* view of the QDMA registers. This includes all 4-bit QDMA registers.

The QRAE*m* is shown in Figure 5-57 and described in Table 5-38.

Figure 5-57. QDMA Region Access Enable for Region *m* (QRAE*m*)32-bit, 2 Rows

31								16
Reserved								
R-0								
15	8	7	6	5	4	3	2	1	0
Reserved		E7	E6	E5	E4	E3	E2	E1	E0
R-0		RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-38. QDMA Region Access Enable for Region M (QRAE*m*) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-0	E*n*		QDMA region access enable for bit *n*/QDMA channel *n* in region *m*.
		0	Accesses via region *m* address space to bit *n* in any QDMA channel register are not allowed. Reads return 0 on bit *n* and writes do not modify the state of bit *n*.
		1	Accesses via region *m* address space to bit *n* in any QDMA channel register are allowed. Reads return the value from bit *n* and writes modify the state of bit *n*.

5.4.1.4 Status/Debug Visibility Registers

The following set of registers provide visibility into the event queues and a TR life cycle. These are useful for system debug as they provide in-depth visibility for the events queued up in the event queue and also provide information on what parts of the EDMA3CC logic are active once the event has been received by the EDMA3CC.

5.4.1.4.1 Event Queue Entry Registers (QxEy)

The event queue entry registers (Q*x*E*y*) exist for all 16 queue entries (the maximum allowed queue entries) for all event queues in the EDMA3CC.

There are Q0E0 to Q0E15, Q1E0 to Q1E15, Q2E0 to Q2E15, and Q3E0 to Q3E15. Each register details the event number (ENUM) and the event type (ETYPE). For example, if the value in Q1E4 is read as 000 004Fh, this means the 4th entry in queue 1 is a manually-triggered event on DMA channel 15.

The Q*x*E*y* is shown in Figure 5-58 and described in Table 5-39.

Figure 5-58. Event Queue Entry Registers (Q*x*E*y*)

31	16
Reserved	
R-0	

15	8	7	6	5	0
Reserved		ETYPE		ENUM	
R-0		R-x		R-x	

LEGEND: R = Read only; -*n* = value after reset; -x = value is indeterminate after reset

Table 5-39. Event Queue Entry Registers (Q*x*E*y*) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-6	ETYPE	0-3h	Event entry *y* in queue *x*. Specifies the specific event type for the given entry in the event queue.
		0	Event triggered via ER.
		1h	Manual triggered via ESR.
		2h	Chain triggered via CER.
		3h	Auto-triggered via QER.
5-0	ENUM	0-3Fh	Event entry *y* in queue *x*. Event number:
		0-3h	QDMA channel number (0 to 3).
		0-3Fh	DMA channel/event number (0 to 63).

5.4.1.4.2 Queue Status Registers (QSTAT*n*)

The queue status registers (QSTAT*n*) is shown in Figure 5-59 and described in Table 5-40.

Figure 5-59. Queue Status Register *n* (QSTAT*n*)

31	25	24	23	21	20	16
Reserved		THRXCD	Reserved		WM	
R-0		R-0	R-0		R-0	

15	13	12	8	7	4	3	0
Reserved		NUMVAL		Reserved		STRTPTR	
R-0		R-0		R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-40. Queue Status Register *n* (QSTAT*n*) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
24	THRXCD		Threshold exceeded. THRXCD is cleared by writing a 1 to the corresponding QTHRXCD*n* bit in the EDMA3CC error clear register (CCERRCLR).
		0	Threshold specified by the Q*n* bit in the queue watermark threshold A register (QWMTHRA) has not been exceeded.
		1	Threshold specified by the Q*n* bit in the queue watermark threshold A register (QWMTHRA) has been exceeded.
23-21	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
20-16	WM	0-10h	Watermark for maximum queue usage. Watermark tracks the most entries that have been in queue *n* since reset or since the last time that the watermark (WM) bit was cleared. WM is cleared by writing a 1 to the corresponding QTHRXCD*n* bit in the EDMA3CC error clear register (CCERRCLR).
		0-10h	Legal values are 0 (empty) to 10h (full).
		11h-1Fh	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
15-13	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
12-8	NUMVAL	0-10h	Number of valid entries in queue *n*. The total number of entries residing in the queue manager FIFO at a given instant. Always enabled.
		0-10h	Legal values are 0 (empty) to 10h (full).
		11h-1Fh	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-4	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3-0	STRTPTR	0-Fh	Start pointer. The offset to the head entry of queue *n*, in units of entries. Always enabled. Legal values are 0 (0th entry) to Fh (15th entry).

5.4.1.4.3 Queue Watermark Threshold A Register (QWMTHRA)

The queue watermark threshold A register (QWMTHRA) is shown in Figure 5-60 and described in Table 5-41.

Figure 5-60. Queue Watermark Threshold A Register (QWMTHRA)

31 29	28 24	23 21	20 16
Reserved	Q3	Reserved	Q2
R-0	R/W-10	R-0	R/W-10

15 13	12 8	7 5	4 0
Reserved	Q1	Reserved	Q0
R-0	R/W-10h	R-0	R/W-10h

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 5-41. Queue Watermark Threshold A Register (QWMTHRA) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
28-24	Q3	0-1Fh	Queue threshold for queue 3 value. The QTHRXCD3 bit in the EDMA3CC error register (CCERR) and the THRXCD bit in the queue status register 3 (QSTAT3) are set when the number of events in queue 3 at an instant in time (visible via the NUMVAL bit in QSTAT3) equals or exceeds the value specified by Q3.
		0-10h	The default is 16 (maximum allowed).
		11h	Disables the threshold errors.
		12h-1Fh	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
23-21	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
20-16	Q2	0-1Fh	Queue threshold for queue 2 value. The QTHRXCD2 bit in the EDMA3CC error register (CCERR) and the THRXCD bit in the queue status register 2 (QSTAT2) are set when the number of events in queue 2 at an instant in time (visible via the NUMVAL bit in QSTAT2) equals or exceeds the value specified by Q2.
		0-10h	The default is 16 (maximum allowed).
		11h	Disables the threshold errors.
		12h-1Fh	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
15-13	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
12-8	Q1		Queue threshold for queue 1 value. The QTHRXCD1 bit in the EDMA3CC error register (CCERR) and the THRXCD bit in the queue status register 1 (QSTAT1) are set when the number of events in queue 1 at an instant in time (visible via the NUMVAL bit in QSTAT1) equals or exceeds the value specified by Q1.
		0-10h	The default is 16 (maximum allowed).
		11h	Disables the threshold errors.
		12h-1Fh	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-5	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
4-0	Q0		Queue threshold for queue 0 value. The QTHRXCD0 bit in the EDMA3CC error register (CCERR) and the THRXCD bit in the queue status register 0 (QSTAT0) are set when the number of events in queue 0 at an instant in time (visible via the NUMVAL bit in QSTAT0) equals or exceeds the value specified by Q0.
		0-10h	The default is 16 (maximum allowed).
		11h	Disables the threshold errors.
		12h-1Fh	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.

5.4.1.4.4 EDMA3CC Status Register (CCSTAT)

The EDMA3CC status register (CCSTAT) has a number of status bits that reflect which parts of the EDMA3CC logic is active at any given instant of time. The CCSTAT is shown in Figure 5-61 and described in Table 5-42.

Figure 5-61. EDMA3CC Status Register (CCSTAT)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved	Reserved			QUEACTV3	QUEACTV2	QUEACTV1	QUEACTV0
R-0	R-0			R-0	R-0	R-0	R-0

15	14	13	12	11	10	9	8
Reserved				COMPACTV			
R-0				R-0			

7	6	5	4	3	2	1	0
Reserved			ACTV	Reserved	TRACTV	QEVTACTV	EVTACTV
R-0			R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-42. EDMA3CC Status Register (CCSTAT) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
19	QUEACTV3		Queue 3 active.
		0	No events are queued in queue 3.
		1	At least one TR is queued in queue 3.
18	QUEACTV2		Queue 2 active.
		0	No events are queued in queue 2.
		1	At least one TR is queued in queue 2.
17	QUEACTV1		Queue 1 active.
		0	No events are queued in queue 1.
		1	At least one TR is queued in queue 1.
16	QUEACTV0		Queue 0 active.
		0	No events are queued in queue 0.
		1	At least one TR is queued in queue 0.
15-14	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
13-8	COMPACTV	0-3Fh	Completion request active. The COMPACTV field reflects the count for the number of completion requests submitted to the transfer controllers. This count increments every time a TR is submitted and is programmed to report completion (the TCINTEN or TCCCHEN bits in OPT in the parameter entry associated with the TR are set). The counter decrements for every valid TCC received back from the transfer controllers. If at any time the count reaches a value of 63, the EDMA3CC will not service any new TRs until the count is less then 63 (or return a transfer completion code from a transfer controller, which would decrement the count).
		0	No completion requests outstanding.
		1h-3Fh	Total of 1 completion request to 63 completion requests are outstanding.
7-5	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.

Table 5-42. EDMA3CC Status Register (CCSTAT) Field Descriptions (continued)

Bit	Field	Value	Description
4	ACTV		Channel controller active. Channel controller active is a logical-OR of each of the *ACTV bits. The ACTV bit remains high through the life of a TR.
		0	Channel is idle..
		1	Channel is busy.
3	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
2	TRACTV		Transfer request active.
		0	Transfer request processing/submission logic is inactive.
		1	Transfer request processing/submission logic is active.
1	QEVTACTV		QDMA event active.
		0	No enabled QDMA events are active within the EDMA3CC.
		1	At least one enabled QDMA event (QER) is active within the EDMA3CC.
0	EVTACTV		DMA event active.
		0	No enabled DMA events are active within the EDMA3CC.
		1	At least one enabled DMA event (ER and EER, ESR, CER) is active within the EDMA3CC.

5.4.1.5 Memory Protection Address Space

5.4.1.5.1 Memory Protection Fault Address Register (MPFAR)

The memory protection fault address register (MPFAR) is shown in Figure 5-62 and described in Table 5-43. A CPU write of 1 to the MPFCLR bit in the memory protection fault command register (MPFCR) causes any error conditions stored in MPFAR to be cleared.

Figure 5-62. Memory Protection Fault Address Register (MPFAR)

31	16
FADDR	
R-0	

15	0
FADDR	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-43. Memory Protection Fault Address Register (MPFAR) Field Descriptions

Bit	Field	Value	Description
31-0	FADDR	0-FFFF FFFFh	Fault address. This 32-bit read-only status register contains the fault address when a memory protection violation is detected. This register can only be cleared via the memory protection fault command register (MPFCR).

5.4.1.5.2 Memory Protection Fault Status Register (MPFSR)

The memory protection fault status register (MPFSR) is shown in Figure 5-63 and described in Table 5-44. A CPU write of 1 to the MPFCLR bit in the memory protection fault command register (MPFCR) causes any error conditions stored in MPFSR to be cleared.

Figure 5-63. Memory Protection Fault Status Register (MPFSR)

31	16
Reserved	
R-0	

15 13	12 9	8 6	5	4	3	2	1	0
Reserved	FID	Reserved	SRE	SWE	SXE	URE	UWE	UXE
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

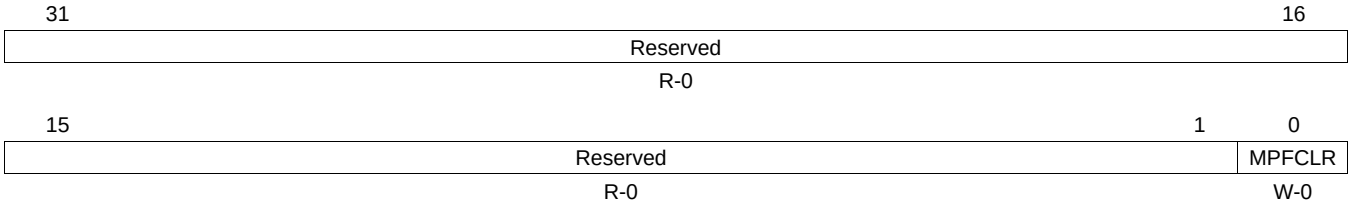
Table 5-44. Memory Protection Fault Status Register (MPFSR) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
12-9	FID	0-Fh	Faulted identification. FID contains valid information if any of the MP error bits (UXE, UWE, URE, SXE, SWE, SRE) are nonzero (that is, if an error has been detected.) The FID field contains the privilege ID for the specific request/requestor that resulted in an MP error.
8-6	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
5	SRE		Supervisor read error.
		0	No error detected.
		1	Supervisor level task attempted to read from a MP page without SR permissions.
4	SWE		Supervisor write error.
		0	No error detected.
		1	Supervisor level task attempted to write to a MP page without SW permissions.
3	SXE		Supervisor execute error.
		0	No error detected.
		1	Supervisor level task attempted to execute from a MP page without SX permissions.
2	URE		User read error.
		0	No error detected.
		1	User level task attempted to read from a MP page without UR permissions.
1	UWE		User write error.
		0	No error detected.
		1	User level task attempted to write to a MP page without UW permissions.
0	UXE		User execute error.
		0	No error detected.
		1	User level task attempted to execute from a MP page without UX permissions.

5.4.1.5.3 Memory Protection Fault Command Register (MPFCR)

The memory protection fault command register (MPFCR) is shown in Figure 5-64 and described in Table 5-45.

Figure 5-64. Memory Protection Fault Command Register (MPFCR)



LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 5-45. Memory Protection Fault Command Register (MPFCR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	MPFCLR		Fault clear register.
		0	CPU write of 0 has no effect.
		1	CPU write of 1 to the MPFCLR bit causes any error conditions stored in the memory protection fault address register (MPFAR) and the memory protection fault status register (MPFSR) to be cleared.

5.4.1.5.4 Memory Protection Page Attribute Register (MPPA*n*)

The memory protection page attribute register (MPPA*n*) is shown in Figure 5-65 and described in Table 5-46.

Figure 5-65. Memory Protection Page Attribute Register (MPPA*n*)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AID5	AID4	AID3	AID2	AID1	AID0	EXT	Rsvd	Reserved		SR	SW	SX	UR	UW	UX
RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	R-0	RW-1		RW-1	RW-1	RW-0	RW-1	RW-1	RW-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-46. Memory Protection Page Attribute Register (MPPA*n*) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
15-10	AID*m*		Allowed ID 'N'
		0	Requests with Privilege ID == N are not allowed to region M, regardless of permission settings (UW, UR, SW, SR).
		1	Requests with Privilege ID == N are permitted, if access type is allowed as defined by permission settings (UW, UR, SW, SR).
9	EXT		External Allowed ID.
		0	Requests with Privilege ID >= 6 are not allowed to region M, regardless of permission settings (UW, UR, SW, SR).
		1	Requests with Privilege ID >= 6 are permitted, if access type is allowed as defined by permission settings (UW, UR, SW, SR).
8	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-6	Reserved	1	Reserved. Always write 1 to this bit.
5	SR		Supervisor read permission.
		0	Supervisor read accesses are not allowed from region M.
		1	Supervisor write accesses are allowed from region M addresses.
4	SW		Supervisor write permission.
		0	Supervisor write accesses are not allowed to region M.
		1	Supervisor write accesses are allowed to region N addresses.
3	SX		Supervisor execute permission.
		0	Supervisor execute accesses are not allowed from region M.
		1	Supervisor execute accesses are allowed from region M addresses.
2	UR		User read permission.
		0	User read accesses are not allowed from region M.
		1	User read accesses are allowed from region N addresses.
1	UW		User write permission.
		0	User write accesses are not allowed to region M.
		1	User write accesses are allowed to region M addresses.
0	UX		User execute permission.
		0	User execute accesses are not allowed from region M.
		1	User execute accesses are allowed from region M addresses.

5.4.1.6 DMA Channel Registers

The following sets of registers pertain to the 64 DMA channels. The 64 DMA channels consist of a set of registers (with exception of DMAQNUM*n*) that each have 64 bits and the bit position of each register matches the DMA channel number. Each register is named with the format *reg_name* that corresponds to DMA channels 0 through 31 and *reg_name_High* that corresponds to DMA channels 32 through 64.

For example, the event register (ER) corresponds to DMA channel 0 through 31 and the event register high register (ERH) corresponds to DMA channel 32 through 63. The register is typically called the event register.

The DMA channel registers are accessible via read/writes to the global address range. They are also accessible via read/writes to the shadow address range. The read/write ability to the registers in the shadow region are controlled by the DMA region access registers (DRAE*m*/DRAEH*m*). The registers are described in Section 5.4.1.3.1 and the details for shadow region/global region usage is explained in Section 5.2.7.

5.4.1.6.1 Event Registers (ER, ERH)

All external events are captured in the event register (ER/ERH). The events are latched even when the events are not enabled. If the event bit corresponding to the latched event is enabled (EER.E*n*/EERH.E*n* = 1), then the event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. The event register bits are automatically cleared (ER.E*n*/ERH.E*n* = 0) once the corresponding events are prioritized and serviced. If ER.E*n*/ERH.E*n* are already set and another event is received on the same channel/event, then the corresponding event is latched in the event miss register (EMR.E*n*/EMRH.E*n*), provided that the event was enabled (EER.E*n*/EERH.E*n* = 1).

Event *n* can be cleared by the CPU writing a 1 to corresponding event bit in the event clear register (ECR/ECRH). The setting of an event is a higher priority relative to clear operations (via hardware or software). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR/EMRH would be set because an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

Table 5-110 provides the type of synchronization events and the EDMA3CC channels associated to each of these external events.

The ER is shown in Figure 5-66 and described in Table 5-47. The ERH is shown in Figure 5-67 and described in Table 5-48.

Figure 5-66. Event Register (ER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	19	E18	E17	E16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-47. Event Register (ER) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event 0-31. Events 0-31 are captured by the EDMA3CC and are latched into ER. The events are set (E*n* = 1) even when events are disabled (E*n* = 0 in the event enable register, EER).
		0	EDMA3CC event is not asserted.
		1	EDMA3CC event is asserted. Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

Figure 5-67. Event Register High (ERH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E63	E62	E61	E60	E59	E58	E57	E56	E55	E54	E53	E52	E51	E50	E49	E48
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E47	E46	E45	E44	E43	E42	E41	E40	E39	E38	E37	E36	E35	E34	E33	E32
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-48. Event Register High (ERH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event 32-63. Events 32-63 are captured by the EDMA3CC and are latched into ERH. The events are set (E*n* = 1) even when events are disabled (E*n* = 0 in the event enable register high, EERH).
		0	EDMA3CC event is not asserted.
		1	EDMA3CC event is asserted. Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

5.4.1.6.2 Event Clear Registers (ECR, ECRH)

Once an event has been posted in the event registers (ER/ERH), the event is cleared in two ways. If the event is enabled in the event enable register (EER/EERH) and the EDMA3CC submits a transfer request for the event to the EDMA3TC, it clears the corresponding event bit in the event register. If the event is disabled in the event enable register (EER/EERH), the CPU can clear the event by way of the event clear registers (ECR/ECRH).

Writing a 1 to any of the bits clears the corresponding event; writing a 0 has no effect. Once an event bit is set in the event register, it remains set until EDMA3CC submits a transfer request for that event or the CPU clears the event by setting the corresponding bit in ECR/ECRH.

The ECR is shown in Figure 5-68 and described in Table 5-49. The ECRH is shown in Figure 5-69 and described in Table 5-50.

Figure 5-68. Event Clear Register (ECR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-49. Event Clear Register (ECR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event clear for event 0-31. Any of the event bits in ECR is set to clear the event (E*n*) in the event register (ER). A write of 0 has no effect.
		0	No effect.
		1	EDMA3CC event is cleared in the event register (ER).

Figure 5-69. Event Clear Register High (ECRH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E63	E62	E61	E60	E59	E58	E57	E56	E55	E54	E53	E52	E51	E50	E49	E48
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E47	E46	E45	E44	E43	E42	E41	E40	E39	E38	E37	E36	E35	E34	E33	E32
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-50. Event Clear Register High (ECRH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event clear for event 32-63. Any of the event bits in ECRH are set to clear the event (E*n*) in the event register high (ERH). A write of 0 has no effect.
		0	No effect.
		1	EDMA3CC event is cleared in the event register high (ERH).

5.4.1.6.3 Event Set Registers (ESR, ESRH)

The event set registers (ESR/ESRH) allow the CPU (EDMA3 programmers) to manually set events to initiate DMA transfer requests. CPU writes of 1 to any event set register (E*n*) bits set the corresponding bits in the registers. The set event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. Writing a 0 has no effect.

The event set registers operate independent of the event registers (ER/ERH), and a write of 1 is always considered a valid event regardless of whether the event is enabled (the corresponding event bits are set or cleared in EER.E*n*/EERH.E*n*).

Once the event is set in the event set registers, it cannot be cleared by CPU writes, in other words, the event clear registers (ECR/ECRH) have no effect on the state of ESR/ESRH. The bits will only be cleared once the transfer request corresponding to the event has been submitted to the transfer controller. The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR/EMRH would be set because an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

Manually-triggered transfers via writes to ESR/ESRH allow the CPU to submit DMA requests in the system, these are relevant for memory-to-memory transfer scenarios. If the ESR.E*n*/ESRH.E*n* bit is already set and another CPU write of 1 is attempted to the same bit, then the corresponding event is latched in the event missed registers (EMR.E*n*/EMRH.E*n* = 1).

The ESR is shown in Figure 5-70 and described in Table 5-51. The ESRH is shown in Figure 5-71 and described in Table 5-52.

Figure 5-70. Event Set Register (ESR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 5-51. Event Set Register (ESR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event set for event 0-31.
		0	No effect.
		1	Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

Figure 5-71. Event Set Register High (ESRH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E63	E62	E61	E60	E59	E58	E57	E56	E55	E54	E53	E52	E51	E50	E49	E48
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E47	E46	E45	E44	E43	E42	E41	E40	E39	E38	E37	E36	E35	E34	E33	E32
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 5-52. Event Set Register High (ESRH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event set for event 32-63.
		0	No effect.
		1	Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

5.4.1.6.4 Chained Event Registers (CER, CERH)

When the OPTIONS parameter for a PaRAM entry is programmed to returned a chained completion code (ITCCHEN = 1 and/or TCCHEN = 1), then the value dictated by the TCC[5:0] (also programmed in OPT) forces the corresponding event bit to be set in the chained event registers (CER/CERH). The set chained event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. This results in a chained-triggered transfer.

The chained event registers do not have any enables. The generation of a chained event is essentially enabled by the PaRAM entry that has been configured for intermediate and/or final chaining on transfer completion. The E*n* bit is set (regardless of the state of EER.E*n*/EERH.E*n*) when a chained completion code is returned from one of the transfer controllers or is generated by the EDMA3CC via the early completion path. The bits in the chained event register are cleared when the corresponding events are prioritized and serviced.

If the E*n* bit is already set and another chaining completion code is return for the same event, then the corresponding event is latched in the event missed registers (EMR.E*n*/EMRH.E*n* = 1). The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR/EMRH would be set because an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The CER is shown in Figure 5-72 and described in Table 5-53. The CERH is shown in Figure 5-73 and described in Table 5-54.

Figure 5-72. Chained Event Register (CER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-53. Chained Event Register (CER) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Chained event for event 0-31.
		0	No effect.
		1	Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

Figure 5-73. Chained Event Register High (CERH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E63	E62	E61	E60	E59	E58	E57	E56	E55	E54	E53	E52	E51	E50	E49	E48
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E47	E46	E45	E44	E43	E42	E41	E40	E39	E38	E37	E36	E35	E34	E33	E32
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-54. Chained Event Register High (CERH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Chained event set for event 32-63.
		0	No effect.
		1	Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

5.4.1.6.5 Event Enable Registers (EER, EERH)

The EDMA3CC provides the option of selectively enabling/disabling each event in the event registers (ER/ERH) by using the event enable registers (EER/EERH). If an event bit in EER/EERH is set (using the event enable set registers, EESR/EESRH), it will enable that corresponding event. Alternatively, if an event bit in EER/EERH is cleared (using the event enable clear registers, EECR/EECRH), it will disable the corresponding event.

The event registers latch all events that are captured by EDMA3CC, even if the events are disabled (although EDMA3CC does not process it). Enabling an event with a pending event already set in the event registers enables the EDMA3CC to process the already set event like any other new event. The EER/EERH settings do not have any effect on chained events (CER.E*n*/CERH.E*n* = 1) and manually set events (ESR.E*n*/ESRH.E*n* = 1).

The EER is shown in Figure 5-74 and described in Table 5-55. Th EERH is shown in Figure 5-75 and described in Table 5-56.

Figure 5-74. Event Enable Register (EER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-55. Event Enable Register (EER) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event enable for events 0-31.
		0	Event is not enabled. An external event latched in the event register (ER) is not evaluated by the EDMA3CC.
		1	Event is enabled. An external event latched in the event register (ER) is evaluated by the EDMA3CC.

Figure 5-75. Event Enable Register High (EERH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E63	E62	E61	E60	E59	E58	E57	E56	E55	E54	E53	E52	E51	E50	E49	E48
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E47	E46	E45	E44	E43	E42	E41	E40	E39	E38	E37	E36	E35	E34	E33	E32
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-56. Event Enable Register High (EERH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event enable for events 32-63.
		0	Event is not enabled. An external event latched in the event register high (ERH) is not evaluated by the EDMA3CC.
		1	Event is enabled. An external event latched in the event register high (ERH) is evaluated by the EDMA3CC.

5.4.1.6.6 Event Enable Clear Register (EECR, EECRH)

The event enable registers (EER/EERH) cannot be modified by directly writing to them. The intent is to ease the software burden for the case where multiple tasks are attempting to simultaneously modify these registers. The event enable clear registers (EECR/EECRH) are used to disable events. Writes of 1 to the bits in EECR/EECRH clear the corresponding event bits in EER/EERH; writes of 0 have no effect.

The EECR is shown in Figure 5-76 and described in Table 5-57. The EECRH is shown in Figure 5-77 and described in Table 5-58.

Figure 5-76. Event Enable Clear Register (EECR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-57. Event Enable Clear Register (EECR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event enable clear for events 0-31.
		0	No effect.
		1	Event is disabled. Corresponding bit in the event enable register (EER) is cleared (E*n* = 0).

Figure 5-77. Event Enable Clear Register High (EECRH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E63	E62	E61	E60	E59	E58	E57	E56	E55	E54	E53	E52	E51	E50	E49	E48
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E47	E46	E45	E44	E43	E42	E41	E40	E39	E38	E37	E36	E35	E34	E33	E32
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-58. Event Enable Clear Register High (EECRH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event enable clear for events 32-63.
		0	No effect.
		1	Event is disabled. Corresponding bit in the event enable register high (EERH) is cleared (E*n* = 0).

5.4.1.6.7 Event Enable Set Registers (EESR, EESRH)

The event enable registers (EER/EERH) cannot be modified by directly writing to them. The intent is to ease the software burden for the case where multiple tasks are attempting to simultaneously modify these registers. The event enable set registers (EESR/EESRH) are used to enable events. Writes of 1 to the bits in EESR/EESRH set the corresponding event bits in EER/EERH; writes of 0 have no effect.

The EESR is shown in Figure 5-78 and described in Table 5-59. The EESRH is shown in Figure 5-79 and described in Table 5-60.

Figure 5-78. Event Enable Set Register (EESR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-59. Event Enable Set Register (EESR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event enable set for events 0-31.
		0	No effect.
		1	Event is enabled. Corresponding bit in the event enable register (EER) is set (E*n* = 1).

Figure 5-79. Event Enable Set Register High (EESRH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E63	E62	E61	E60	E59	E58	E57	E56	E55	E54	E53	E52	E51	E50	E49	E48
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E47	E46	E45	E44	E43	E42	E41	E40	E39	E38	E37	E36	E35	E34	E33	E32
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-60. Event Enable Set Register High (EESRH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event enable set for events 32-63.
		0	No effect.
		1	Event is enabled. Corresponding bit in the event enable register high (EERH) is set (E*n* = 1).

5.4.1.6.8 Secondary Event Registers (SER, SERH)

The secondary event registers (SER/SERH) provide information on the state of a DMA channel or event (0 through 63). If the EDMA3CC receives a TR synchronization due to a manual-trigger, event-trigger, or chained-trigger source (ESR.E*n*/ESRH.E*n* = 1, ER.E*n*/ERH.E*n* = 1, or CER.E*n*/CERH.E*n* = 1), which results in the setting of a corresponding event bit in SER/SERH (SER.E*n*/SERH.E*n* = 1), it implies that the corresponding DMA event is in the queue.

Once a bit corresponding to an event is set in SER/SERH, the EDMA3CC does not prioritize additional events on the same DMA channel. Depending on the condition that lead to the setting of the SER bits, either the EDMA3CC hardware or the software (using SECR/SECRH) needs to clear the SER/SERH bits for the EDMA3CC to evaluate subsequent events (subsequent transfers) on the same channel. See Section 5.1.1 for additional conditions that can cause the secondary event registers to be set.

The SER is shown in Figure 5-80 and described in Table 5-61. The SERH is shown in Figure 5-81 and described in Table 5-62.

Figure 5-80. Secondary Event Register (SER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-61. Secondary Event Register (SER) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Secondary event register. The secondary event register is used along with the event register (ER) to provide information on the state of an event.
		0	Event is not currently stored in the event queue.
		1	Event is currently stored in the event queue. Event arbiter will not prioritize additional events.

Figure 5-81. Secondary Event Register High (SERH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E63	E62	E61	E60	E59	E58	E57	E56	E55	E54	E53	E52	E51	E50	E49	E48
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E47	E46	E45	E44	E43	E42	E41	E40	E39	E38	E37	E36	E35	E34	E33	E32
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-62. Secondary Event Register High (SERH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Secondary event register. The secondary event register is used along with the event register high (ERH) to provide information on the state of an event.
		0	Event is not currently stored in the event queue.
		1	Event is currently stored in the event queue. Event submission/prioritization logic will not prioritize additional events.

5.4.1.6.9 Secondary Event Clear Registers (SECR, SECRH)

The secondary event clear registers (SECR/SECRH) clear the status of the secondary event registers (SER/SERH). CPU writes of 1 clear the corresponding set bits in SER/SERH. Writes of 0 have no effect.

The SECR is shown in Figure 5-82 and described in Table 5-63. The SECRH is shown in Figure 5-83 and described in Table 5-64.

Figure 5-82. Secondary Event Clear Register (SECR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-63. Secondary Event Clear Register (SECR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Secondary event clear register.
		0	No effect.
		1	Corresponding bit in the secondary event register (SER) is cleared (E*n* = 0).

Figure 5-83. Secondary Event Clear Register High (SECRH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E63	E62	E61	E60	E59	E58	E57	E56	E55	E54	E53	E52	E51	E50	E49	E48
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E47	E46	E45	E44	E43	E42	E41	E40	E39	E38	E37	E36	E35	E34	E33	E32
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-64. Secondary Event Clear Register High (SECRH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Secondary event clear register.
		0	No effect.
		1	Corresponding bit in the secondary event registers high (SERH) is cleared (E*n* = 0).

5.4.1.7 Interrupt Registers

All DMA/QDMA channels can be set to assert an EDMA3CC completion interrupt to the CPU on transfer completion, by appropriately configuring the PaRAM entry associated with the channels. The following set of registers is used for the transfer completion interrupt reporting/generating by the EDMA3CC. See Section 5.2.9 for more details on EDMA3CC completion interrupt generation.

5.4.1.7.1 Interrupt Enable Registers (IER, IERH)

Interrupt enable registers (IER/IERH) are used to enable/disable the transfer completion interrupt generation by the EDMA3CC for all DMA/QDMA channels. The IER/IERH cannot be written to directly. To set any interrupt bit in IER/IERH, a 1 must be written to the corresponding interrupt bit in the interrupt enable set registers (IESR/IESRH). Similarly, to clear any interrupt bit in IER/IERH, a 1 must be written to the corresponding interrupt bit in the interrupt enable clear registers (IECR/IECRH).

The IER is shown in Figure 5-84 and described in Table 5-65. The IERH is shown in Figure 5-85 and described in Table 5-66.

Figure 5-84. Interrupt Enable Register (IER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27I	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-65. Interrupt Enable Register (IER) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Interrupt enable for channels 0-31.
		0	Interrupt is not enabled.
		1	Interrupt is enabled.

Figure 5-85. Interrupt Enable Register High (IERH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-66. Interrupt Enable Register High (IERH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Interrupt enable for channels 32-63.
		0	Interrupt is not enabled.
		1	Interrupt is enabled.

5.4.1.7.2 Interrupt Enable Clear Register (IECR, IECRH)

The interrupt enable clear registers (IECR/IECRH) are used to clear interrupts. Writes of 1 to the bits in IECR/IECRH clear the corresponding interrupt bits in the interrupt enable registers (IER/IERH); writes of 0 have no effect.

The IECR is shown in Figure 5-86 and described in Table 5-67. The IECRH is shown in Figure 5-87 and described in Table 5-68.

Figure 5-86. Interrupt Enable Clear Register (IECR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-67. Interrupt Enable Clear Register (IECR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Interrupt enable clear for channels 0-31.
		0	No effect.
		1	Corresponding bit in the interrupt enable register (IER) is cleared (I*n* = 0).

Figure 5-87. Interrupt Enable Clear Register High (IECRH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-68. Interrupt Enable Clear Register High (IECRH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Interrupt enable clear for channels 32-63.
		0	No effect.
		1	Corresponding bit in the interrupt enable register high (IERH) is cleared (I*n* = 0).

5.4.1.7.3 Interrupt Enable Set Registers (IESR, IESRH)

The interrupt enable set registers (IESR/IESRH) are used to enable interrupts. Writes of 1 to the bits in IESR/IESRH set the corresponding interrupt bits in the interrupt enable registers (IER/IERH); writes of 0 have no effect.

The IESR is shown in Figure 5-88 and described in Table 5-69. The IESRH is shown in Figure 5-89 and described in Table 5-70.

Figure 5-88. Interrupt Enable Set Register (IESR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	II8	I17	I16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-69. Interrupt Enable Set Register (IESR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Interrupt enable set for channels 0-31.
		0	No effect.
		1	Corresponding bit in the interrupt enable register (IER) is set (I*n* = 1).

Figure 5-89. Interrupt Enable Set Register High (IESRH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-70. Interrupt Enable Set Register High (IESRH) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Interrupt enable clear for channels 32-63.
		0	No effect.
		1	Corresponding bit in the interrupt enable register high (IERH) is set (I*n* = 1).

5.4.1.7.4 Interrupt Pending Register (IPR, IPRH)

If the TCINTEN and/or ITCINTEN bit in the channel option parameter (OPT) is set in the PaRAM entry associated with the channel (DMA or QDMA), then the EDMA3TC (for normal completion) or the EDMA3CC (for early completion) returns a completion code on transfer or intermediate transfer completion. The value of the returned completion code is equal to the TCC bit in OPT for the PaRAM entry associated with the channel.

When an interrupt transfer completion code with TCC = *n* is detected by the EDMA3CC, then the corresponding bit is set in the interrupt pending register (IPR.I*n*, if *n* = 0 to 31; IPRH.I*n,* if *n* = 32 to 63). Note that once a bit is set in the interrupt pending registers, it remains set; it is your responsibility to clear these bits. The bits set in IPR/IPRH are cleared by writing a 1 to the corresponding bits in the interrupt clear registers (ICR/ICRH).

The IPR is shown in Figure 5-90 and described in Table 5-71. The IPRH is shown in Figure 5-91 and described in Table 5-72.

Figure 5-90. Interrupt Pending Register (IPR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-71. Interrupt Pending Register (IPR) Field Descriptions

Bit	Field	Value	Description
31-0	I*n*		Interrupt pending for TCC = 0-31.
		0	Interrupt transfer completion code is not detected or was cleared.
		1	Interrupt transfer completion code is detected (I*n* = 1, *n* = EDMA3TC[5:0]).

Figure 5-91. Interrupt Pending Register High (IPRH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-72. Interrupt Pending Register High (IPRH) Field Descriptions

Bit	Field	Value	Description
31-0	I*n*		Interrupt pending for TCC = 32-63.
		0	Interrupt transfer completion code is not detected or was cleared.
		1	Interrupt transfer completion code is detected (I*n* = 1, *n* = EDMA3TC[5:0]).

5.4.1.7.5 Interrupt Clear Registers (ICR, ICRH)

The bits in the interrupt pending registers (IPR/IPRH) are cleared by writing a 1 to the corresponding bits in the interrupt clear registers(ICR/ICRH). Writes of 0 have no effect. All set bits in IPR/IPRH must be cleared to allow EDMA3CC to assert additional transfer completion interrupts.

The ICR is shown in Figure 5-92 and described in Table 5-73. The ICRH is shown in Figure 5-93 and described in Table 5-74.

Figure 5-92. Interrupt Clear Register (ICR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-73. Interrupt Clear Register (ICR) Field Descriptions

Bit	Field	Value	Description
31-0	I*n*		Interrupt clear register for TCC = 0-31.
		0	No effect.
		1	Corresponding bit in the interrupt pending register (IPR) is cleared (I*n* = 0).

Figure 5-93. Interrupt Clear Register High (ICRH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -*n* = value after reset

Table 5-74. Interrupt Clear Register High (ICRH) Field Descriptions

Bit	Field	Value	Description
31-0	I*n*		Interrupt clear register for TCC = 32-63.
		0	No effect.
		1	Corresponding bit in the interrupt pending register high (IPRH) is cleared (I*n* = 0).

5.4.1.7.6 Interrupt Evaluate Register (IEVAL)

The interrupt evaluate register (IEVAL) is the only register that physically exists in both the global region and the shadow regions. In other words, the read/write accessibility for the shadow region IEVAL is not affected by the DMA/QDMA region access registers (DRAE*m*/DRAEH*m*, QRAE*n*/QRAEH*n*). IEVAL is needed for robust ISR operations to ensure that interrupts are not missed by the CPU.

The IEVAL is shown in Figure 5-94 and described in Table 5-75.

Figure 5-94. Interrupt Evaluate Register (IEVAL)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		Rsvd	EVAL
R-0		W-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 5-75. Interrupt Evaluate Register (IEVAL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	EVAL		Interrupt evaluate.
		0	No effect.
		1	Causes EDMA3CC completion interrupt to be pulsed, if any enabled (IER*n*/IERH*n* = 1) interrupts are still pending (IPR*n*/IPRH*n* = 1). The EDMA3CC completion interrupt that is pulsed depends on which IEVAL is being exercised. For example, writing to the EVAL bit in IEVAL pulses the global completion interrupt, but writing to the EVAL bit in IEVAL0 pulses the region 0 completion interrupt.

5.4.1.8 QDMA Registers

The following sets of registers control the QDMA channels in the EDMA3CC. The QDMA channels (with the exception of the QDMA queue number register) consist of a set of registers, each of which have a bit location. Each bit position corresponds to a QDMA channel number. The QDMA channel registers are accessible via read/writes to the global address range. They are also accessible via read/writes to the shadow address range. The read/write accessibility in the shadow region address region is controlled by the QDMA region access registers (QRAE*n*/QRAEH*n*). Section 5.2.7 details shadow region/global region usage.

5.4.1.8.1 QDMA Event Register (QER)

The QDMA event register (QER) channel *n* bit is set (E*n* = 1) when the CPU or any EDMA3 programmer (including EDMA3) performs a write to the trigger word (using the QDMA channel mapping register (QCHMAP*n*)) in the PaRAM entry associated with QDMA channel *n* (which is also programmed using QCHMAP*n*). The E*n* bit is also set when the EDMA3CC performs a link update on a PaRAM address that matches the QCHMAP*n* settings. The QDMA event is latched only if the QDMA event enable register (QEER) channel *n* bit is also enabled (QEER.E*n* = 1). Once a bit is set in QER, then the corresponding QDMA event (auto-trigger) is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. See Section 5.1.1 for additional conditions that can lead to the setting of QER bits.

The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then the QDMA event missed register (QEMR) would be set because an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The set bits in QER are only cleared when the transfer request associated with the corresponding channels has been processed by the EDMA3CC and submitted to the transfer controller. If the E*n* bit is already set and a QDMA event for the same QDMA channel occurs prior to the original being cleared, then the second missed event is latched in QEMR (E*n* = 1).

The QER is shown in Figure 5-95 and described in Table 5-76.

Figure 5-95. QDMA Event Register (QER)

31	16
Reserved	
R-0	

15	7	6	5	4	3	2	1	0
Reserved	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-76. QDMA Event Register (QER) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-0	E*n*		QDMA event for channels 0-7.
		0	No effect.
		1	Corresponding QDMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

5.4.1.8.2 QDMA Event Enable Register (QEER)

The EDMA3CC provides the option of selectively enabling/disabling each channel in the QDMA event register (QER) by using the QDMA event enable register (QEER). If any of the event bits in QEER is set (using the QDMA event enable set register, QEESR), it will enable that corresponding event. Alternatively, if any event bit in QEER is cleared (using the QDMA event enable clear register, QEECR), it will disable the corresponding QDMA channel. The QDMA event register will not latch any event for a QDMA channel, if it is not enabled via QEER.

The QEER is shown in Figure 5-96 and described in Table 5-77.

Figure 5-96. QDMA Event Enable Register (QEER)

31	16
Reserved	
R-0	

15	7	6	5	4	3	2	1	0
Reserved	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-77. QDMA Event Enable Register (QEER) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-0	E*n*		QDMA event enable for channels 0-7.
		0	QDMA channel *n* is not enabled. QDMA event will not be recognized and will not latch in the QDMA event register (QER).
		1	QDMA channel *n* is enabled. QDMA events will be recognized and will get latched in the QDMA event register (QER).

5.4.1.8.3 QDMA Event Enable Clear Register (QEECR)

The QDMA event enable register (QEER) cannot be modified by directly writing to the register, to ease the software burden when multiple tasks are attempting to simultaneously modify these registers. The QDMA event enable clear register (QEECR) is used to disable events. Writes of 1 to the bits in QEECR clear the corresponding QDMA channel bits in QEER; writes of 0 have no effect.

The QEECR is shown in Figure 5-97 and described in Table 5-78.

Figure 5-97. QDMA Event Enable Clear Register (QEECR)

31	16
Reserved	
R-0	

15	7	6	5	4	3	2	1	0
Reserved	E7	E6	E5	E4	E3	E2	E1	E0
R-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 5-78. QDMA Event Enable Clear Register (QEECR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-0	E*n*		QDMA event enable clear for channels 0-7.
		0	No effect.
		1	QDMA event is disabled. Corresponding bit in the QDMA event enable register (QEER) is cleared (E*n* = 0).

5.4.1.8.4 QDMA Event Enable Set Register (QEESR)

The QDMA event enable register (QEER) cannot be modified by directly writing to the register, to ease the software burden when multiple tasks are attempting to simultaneously modify these registers. The QDMA event enable set register (QEESR) is used to enable events. Writes of 1 to the bits in QEESR set the corresponding QDMA channel bits in QEER; writes of 0 have no effect.

The QEESR is shown in Figure 5-98 and described in Table 5-79.

Figure 5-98. QDMA Event Enable Set Register (QEESR)

31	16
Reserved	
R-0	

15	7	6	5	4	3	2	1	0
Reserved	E7	E6	E5	E4	E3	E2	E1	E0
R-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 5-79. QDMA Event Enable Set Register (QEESR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-0	E*n*		QDMA event enable set for channels 0-7.
		0	No effect.
		1	QDMA event is enabled. Corresponding bit in the QDMA event enable register (QEER) is set (E*n* = 1).

5.4.1.8.5 QDMA Secondary Event Register (QSER)

The QDMA secondary event register (QSER) provides information on the state of a QDMA event. If at any time a bit corresponding to a QDMA channel is set in QSER, that implies that the corresponding QDMA event is in the queue. Once a bit corresponding to a QDMA channel is set in QSER, the EDMA3CC does not prioritize additional events on the same QDMA channel. Depending on the condition that lead to the setting of the QSER bits, either the EDMA3CC hardware or the software (using QSECR) needs to clear the QSER bits for the EDMA3CC to evaluate subsequent QDMA events on the channel. Based on whether the associated TR request is valid, or it is a null or dummy TR, the implications on the state of QSER and the required user actions to submit another QDMA transfer might be different. See Section 5.1.1 for additional conditions that can cause the secondary event registers (QSER\SER) to be set.

The QSER is shown in Figure 5-99 and described in Table 5-80.

Figure 5-99. QDMA Secondary Event Register (QSER)

31	16
Reserved	
R-0	

15	7	6	5	4	3	2	1	0
Reserved	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-80. QDMA Secondary Event Register (QSER) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-0	E*n*		QDMA secondary event register for channels 0-7.
		0	QDMA event is not currently stored in the event queue.
		1	QDMA event is currently stored in the event queue. EDMA3CC will not prioritize additional events.

5.4.1.8.6 QDMA Secondary Event Clear Register (QSECR)

The QDMA secondary event clear register (QSECR) clears the status of the QDMA secondary event register (QSER) and the QDMA event register (QER). CPU writes of 1 clear the corresponding set bits in QSER and QER. Writes of 0 have no effect. Note that this differs from the secondary event clear register (SECR) operation, which only clears the secondary event register (SER) bits and does not affect the event registers.

The QSECR is shown in Figure 5-100 and described in Table 5-81.

Figure 5-100. QDMA Secondary Event Clear Register (QSECR)

31	16
Reserved	
R-0	

15	7	6	5	4	3	2	1	0
Reserved	E7	E6	E5	E4	E3	E2	E1	E0
R-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 5-81. QDMA Secondary Event Clear Register (QSECR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	E*n*		QDMA secondary event clear register for channels 0-7.
		0	No effect.
		1	Corresponding bit in the QDMA secondary event register (QSER) and the QDMA event register (QER) is cleared (E*n* = 0).

5.4.2 EDMA3 Transfer Controller (EDMA3TC) Control Registers

Table 5-82 lists the memory-mapped registers for the EDMA3 transfer controller (EDMA3TC). For the base address of these registers, see Table 1-11.

Table 5-82. EDMA3 Transfer Controller (EDMA3TC) Control Registers

Offset	Acronym	Register Description	Section
00h	PID	Peripheral Identification Register	Section 5.4.2.1
04h	TCCFG	EDMA3TC Configuration Register	Section 5.4.2.2
0100h	TCSTAT	EDMA3TC Channel Status Register	Section 5.4.2.3
0120h	ERRSTAT	Error Register	Section 5.4.2.4.1
0124h	ERREN	Error Enable Register	Section 5.4.2.4.2
0128h	ERRCLR	Error Clear Register	Section 5.4.2.4.3
012Ch	ERRDET	Error Details Register	Section 5.4.2.4.4
0130h	ERRCMD	Error Interrupt Command Register	Section 5.4.2.4.5
0140h	RDRATE	Read Rate Register	Section 5.4.2.5
0240h	SAOPT	Source Active Options Register	Section 5.4.2.6.1
0244h	SASRC	Source Active Source Address Register	Section 5.4.2.6.2
0248h	SACNT	Source Active Count Register	Section 5.4.2.6.3
024Ch	SADST	Source Active Destination Address Register	Section 5.4.2.6.4
0250h	SABIDX	Source Active Source B-Index Register	Section 5.4.2.6.5
0254h	SAMPPRXY	Source Active Memory Protection Proxy Register	Section 5.4.2.6.6
0258h	SACNTRLD	Source Active Count Reload Register	Section 5.4.2.6.7
025Ch	SASRCBREF	Source Active Source Address B-Reference Register	Section 5.4.2.6.8
0260h	SADSTBREF	Source Active Destination Address B-Reference Register	Section 5.4.2.6.9
0280h	DFCNTRLD	Destination FIFO Set Count Reload	Section 5.4.2.6.16
0284h	DFSRCBREF	Destination FIFO Set Destination Address B Reference Register	Section 5.4.2.6.17
0288h	DFDSTBREF	Destination FIFO Set Destination Address B Reference Register	Section 5.4.2.6.18
0300h	DFOPT0	Destination FIFO Options Register 0	Section 5.4.2.6.10
0304h	DFSRC0	Destination FIFO Source Address Register 0	Section 5.4.2.6.11
0308h	DFCNT0	Destination FIFO Count Register 0	Section 5.4.2.6.12
030Ch	DFDST0	Destination FIFO Destination Address Register 0	Section 5.4.2.6.13
0310h	DFBIDX0	Destination FIFO BIDX Register 0	Section 5.4.2.6.14
0314h	DFMPPRXY0	Destination FIFO Memory Protection Proxy Register 0	Section 5.4.2.6.15
0340h	DFOPT1	Destination FIFO Options Register 1	Section 5.4.2.6.10
0344h	DFSRC1	Destination FIFO Source Address Register 1	Section 5.4.2.6.11
0348h	DFCNT1	Destination FIFO Count Register 1	Section 5.4.2.6.12
034Ch	DFDST1	Destination FIFO Destination Address Register 1	Section 5.4.2.6.13
0350h	DFBIDX1	Destination FIFO BIDX Register 1	Section 5.4.2.6.14
0354h	DFMPPRXY1	Destination FIFO Memory Protection Proxy Register 1	Section 5.4.2.6.15
0380h	DFOPT2	Destination FIFO Options Register 2	Section 5.4.2.6.10
0384h	DFSRC2	Destination FIFO Source Address Register 2	Section 5.4.2.6.11
0388h	DFCNT2	Destination FIFO Count Register 2	Section 5.4.2.6.12
038Ch	DFDST2	Destination FIFO Destination Address Register 2	Section 5.4.2.6.13
0390h	DFBIDX2	Destination FIFO BIDX Register 2	Section 5.4.2.6.14
0394h	DFMPPRXY2	Destination FIFO Memory Protection Proxy Register 2	Section 5.4.2.6.15
03C0h	DFOPT3	Destination FIFO Options Register 3	Section 5.4.2.6.10
03C4h	DFSRC3	Destination FIFO Source Address Register 3	Section 5.4.2.6.11
03C8h	DFCNT3	Destination FIFO Count Register 3	Section 5.4.2.6.12
03CCh	DFDST3	Destination FIFO Destination Address Register 3	Section 5.4.2.6.13

Table 5-82. EDMA3 Transfer Controller (EDMA3TC) Control Registers (continued)

Offset	Acronym	Register Description	Section
03D0h	DFBIDX3	Destination FIFO BIDX Register 3	Section 5.4.2.6.14
03D4h	DFMPPRXY3	Destination FIFO Memory Protection Proxy Register 3	Section 5.4.2.6.15

5.4.2.1 Peripheral Identification Register (PID)

The peripheral identification register (PID) is a constant register that uniquely identifies the EDMA3TC and specific revision of the EDMA3TC. The PID is shown in Figure 5-101 and described in Table 5-83.

Figure 5-101. Peripheral ID Register (PID)

31	16	15	0
PID		PID	
R-0333h		R-4425h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-83. Peripheral ID Register (PID) Field Descriptions

Bit	Field	Value	Description
31-0	PID	0-FFFF FFFFh	Peripheral identifier.

5.4.2.2 EDMA3TC Configuration Register (TCCFG)

The EDMA3TC configuration register (TCCFG) is shown in Figure 5-102 and described in Table 5-84.

Figure 5-102. EDMA3TC Configuration Register (TCCFG)

31	16
Reserved	
R-0	

15	10	9	8	7	6	5	4	3	2	0
Reserved		DREGDEPTH		Reserved		BUSWIDTH		Rsvd	FIFOSIZE	
R-0		R-x		R-0		R-n		R-0	R-x	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset; -x = value is indeterminate after reset

Table 5-84. EDMA3TC Configuration Register (TCCFG) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
9-8	DREGDEPTH	0-3h	Destination register FIFO depth parameterization.
		0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		1h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		2h	4 entry (for TC0, TC1, TC2, and TC3)
		3h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7-6	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
5-4	BUSWIDTH	0-3h	Bus width parameterization.
		0-1h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		2h	128-bit
		3h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
2-0	FIFOSIZE	0-7h	FIFO size
		0-1h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		2h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		3h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		4h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
		5h	1024 byte FIFO (for TC0, TC1, TC2 and TC3)
		6h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.

5.4.2.3 EDMA3TC Channel Status Register (TCSTAT)

The EDMA3TC channel status register (TCSTAT) is shown in Figure 5-103 and described in Table 5-85.

Figure 5-103. EDMA3TC Channel Status Register (TCSTAT)

31	16
Reserved	
R-0	

15	14	13	12	11	9	8
Reserved		DFSTRTPTR		Reserved		Reserved
R-0		R-0		R-0		R-1

7	6	4	3	2	1	0
Reserved	DSTACTV		Reserved	WSACTV	SRCACTV	PROGBUSY
R-0	R-0		R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-85. EDMA3TC Channel Status Register (TCSTAT) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
13-12	DFSTRTPTR	0-3h	Destination FIFO start pointer. Represents the offset to the head entry of the destination register FIFO, in units of entries.
11-9	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
8	Reserved	1	Reserved. Always read as 1.
7	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
6-4	DSTACTV	0-7h	Destination active state. Specifies the number of transfer requests (TRs) that are resident in the destination register FIFO at a given instant. This bit field can be primarily used for advanced debugging. Legal values are constrained by the destination register FIFO depth parameterization (DSTREGDEPTH) parameter.
		0	Destination FIFO is empty.
		1h	Destination FIFO contains 1 TR.
		2h	Destination FIFO contains 2 TRs.
		3h	Destination FIFO contains 3 TRs.
		4h	Destination FIFO contains 4 TRs. (Full if DSTREGDEPTH==4).
			If the destination register FIFO is empty, then any TR written to Prog Set immediately transitions to the destination register FIFO. If the destination register FIFO is not empty and not full, then any TR written to Prog Set immediately transitions to the destination register FIFO set if the source active state (SRCACTV) bit is set to idle.
			If the destination register FIFO is full, then TRs cannot transition to the destination register FIFO. The destination register FIFO becomes not full when the TR at the head of the destination register FIFO is completed.
		5h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
2	WSACTV		Write status active
		0	Write status is not pending. Write status has been received for all previously issued write commands.
		1	Write status is pending. Write status has not been received for all previously issued write commands.
1	SRCACTV		Source active state
		0	Source controller is idle. Source active register set contains a previously processed transfer request.
		1	Source controller is busy servicing a transfer request.

Table 5-85. EDMA3TC Channel Status Register (TCSTAT) Field Descriptions (continued)

Bit	Field	Value	Description
0	PROGBUSY		Program register set busy
		0	Program set idle and is available for programming by the EDMA3CC.
		1	Program set busy

5.4.2.4 Error Registers

5.4.2.4.1 Error Register (ERRSTAT)

The error status register (ERRSTAT) is shown in Figure 5-104 and described in Table 5-86.

Figure 5-104. Error Register (ERRSTAT)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		MMRAERR	TRERR	Reserved	BUSERR
R-0		R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 5-86. Error Register (ERRSTAT) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3	MMRAERR		MMR address error.
		0	Condition is not detected.
		1	User attempted to read or write to an invalid address in configuration memory map.
2	TRERR		Transfer request (TR) error event.
		0	Condition is not detected.
		1	TR detected that violates constant addressing mode transfer (SAM or DAM is set) alignment rules or has ACNT or BCNT == 0.
1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	BUSERR		Bus error event.
		0	Condition is not detected.
		1	EDMA3TC has detected an error at source or destination address. Error information can be read from the error details register (ERRDET).

5.4.2.4.2 Error Enable Register (ERREN)

The error enable register (ERREN) is shown in Figure 5-105 and described in Table 5-87.

When any of the enable bits are set, a bit set in the corresponding ERRSTAT causes an assertion of the EDMA3TC interrupt.

Figure 5-105. Error Enable Register (ERREN)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		MMRAERR	TRERR	Reserved	BUSERR
R-0		R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-87. Error Enable Register (ERREN) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3	MMRAERR		Interrupt enable for MMR address error (MMRAERR).
		0	MMRAERR is disabled.
		1	MMRAERR is enabled and contributes to the state of EDMA3TC error interrupt generation
2	TRERR		Interrupt enable for transfer request error (TRERR).
		0	TRERR is disabled.
		1	TRERR is enabled and contributes to the state of EDMA3TC error interrupt generation.
1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	BUSERR		Interrupt enable for bus error (BUSERR).
		0	BUSERR is disabled.
		1	BUSERR is enabled and contributes to the state of EDMA3TC error interrupt generation.

5.4.2.4.3 Error Clear Register (ERRCLR)

The error clear register (ERRCLR) is shown in Figure 5-106 and described in Table 5-88.

Figure 5-106. Error Clear Register (ERRCLR)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		MMRAERR	TRERR	Reserved	BUSERR
R-0		W-0	W-0	R-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 5-88. Error Clear Register (ERRCLR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3	MMRAERR		Interrupt enable clear for the MMRAERR bit in the error status register (ERRSTAT).
		0	No effect.
		1	Clears the MMRAERR bit in ERRSTAT but does not clear the error details register (ERRDET).
2	TRERR		Interrupt enable clear for the TRERR bit in the error status register (ERRSTAT).
		0	No effect.
		1	Clears the TRERR bit in ERRSTAT but does not clear the error details register (ERRDET).
1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	BUSERR		Interrupt clear for the BUSERR bit in the error status register (ERRSTAT).
		0	No effect.
		1	Clears the BUSERR bit in ERRSTAT and clears the error details register (ERRDET).

5.4.2.4.4 Error Details Register (ERRDET)

The error details register (ERRDET) is shown in Figure 5-107 and described in Table 5-89.

Figure 5-107. Error Details Register (ERRDET)

31	18	17	16
Reserved		TCCHEN	TCINTEN
R-0		R-0	R-0

15	14	13	8	7	4	3	0
Reserved		TCC		Reserved		STAT	
R-0		R-0		R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 5-89. Error Details Register (ERRDET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
17	TCCHEN	0-1	Transfer completion chaining enable. Contains the TCCHEN value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error.
16	TCINTEN	0-1	Transfer completion interrupt enable. Contains the TCINTEN value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error.
15-14	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
13 - 8	TCC	0-3Fh	Transfer complete code. Contains the TCC value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error.
7-4	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3-0	STAT	0-Fh	Transaction status. Stores the nonzero status/error code that was detected on the read status or write status bus. If read status and write status are returned on the same cycle, then the EDMA3TC chooses nonzero version. If both are nonzero, then the write status is treated as higher priority.
		0	No error.
		1h-7h	Read error.
		8h-Fh	Write error.

5.4.2.4.5 Error Interrupt Command Register (ERRCMD)

The error command register (ERRCMD) is shown in Figure 5-108 and described in Table 5-90.

Figure 5-108. Error Interrupt Command Register (ERRCMD)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		Rsvd	EVAL
R-0		W-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 5-90. Error Interrupt Command Register (ERRCMD) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	EVAL		Error evaluate
		0	No effect
		1	EDMA3TC error line is pulsed if any of the error status register (ERRSTAT) bits are set.

5.4.2.5 Read Rate Register (RDRATE)

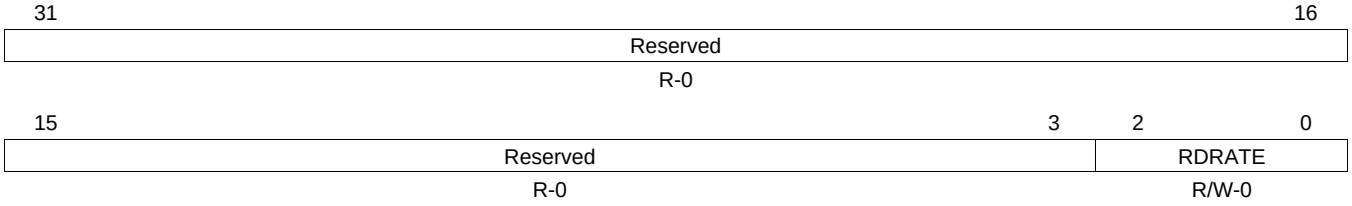
The EDMA3 transfer controller issues read commands at a rate controlled by the read rate register (RDRATE). The RDRATE defines the number of idle cycles that the read controller must wait before issuing subsequent commands. This applies both to commands within a transfer request packet (TRP) and for commands that are issued for different transfer requests (TRs). For instance, if RDRATE is set to 4 cycles between reads, there are 3 inactive cycles between reads.

RDRATE allows flexibility in transfer controller access requests to an endpoint. For an application, RDRATE can be manipulated to slow down the access rate, so that the endpoint may service requests from other masters during the inactive EDMA3TC cycles.

The RDRATE is shown in Figure 5-109 and described in Table 5-91.

NOTE: It is expected that the RDRATE value for a transfer controller is static, as it is decided based on the application requirement. It is not recommended to change this setting on the fly.

Figure 5-109. Read Rate Register (RDRATE)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-91. Read Rate Register (RDRATE) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
2-0	RDRATE	0-7h	Read rate. Controls the number of cycles between read commands. This is a global setting that applies to all TRs for this EDMA3TC.
		0	Reads issued as fast as possible.
		1h	4 cycles between reads.
		2h	8 cycles between reads.
		3h	16 cycles between reads.
		4h	32 cycles between reads.
		5h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.

5.4.2.6 EDMA3TC Channel Registers

The EDMA3TC channel registers are split into three parts: the programming registers, the source active registers, and the destination FIFO register. This section describes the registers and their functions. The program register set is programmed by the channel controller, and is for internal use. The other two sets are read-only and provided to facilitate advanced debug capabilities. The number of destination FIFO register sets depends on the destination FIFO depth.

TC0 has a FIFO depth of 2, so there are two sets of destination FIFO registers associated with TC0. TC1, TC2, and TC3 have a destination FIFO depth of 4, so there are four sets of destination FIFO registers for each of these transfer controllers.

5.4.2.6.1 Source Active Options Register (SAOPT)

The source active options register (SAOPT) is shown in Figure 5-110 and described in Table 5-92.

Figure 5-110. Source Active Options Register (SAOPT)

31	23	22	21	20	19	18	17	16
Reserved		TCCHEN	Rsvd	TCINTEN	Reserved		TCC	
R-0		R/W-0	R-0	R/W-0	R-0		R/W-0	

15	12	11	10	8	7	6	4	3	2	1	0
TCC		Rsvd	FWID		Rsvd	PRI		Reserved		DAM	SAM
R/W-0		R-0	R/W-0		R-0	R/W-0		R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 5-92. Source Active Options Register (SAOPT) Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
22	TCCHEN		Transfer complete chaining enable
		0	Transfer complete chaining is disabled.
		1	Transfer complete chaining is enabled.
21	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
20	TCINTEN		Transfer complete interrupt enable.
		0	Transfer complete interrupt is disabled.
		1	Transfer complete interrupt is enabled.
19-18	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
17-12	TCC	0-3Fh	Transfer complete code. This 6-bit code is used to set the relevant bit in CER or IPR of the EDMA3PCC module.
11	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
10-8	FWID	0-7h	FIFO width. Applies if either SAM or DAM is set to constant addressing mode.
		0	FIFO width is 8-bit.
		1h	FIFO width is 16-bit.
		2h	FIFO width is 32-bit.
		3h	FIFO width is 64-bit.
		4h	FIFO width is 128-bit.
		5h	FIFO width is 256-bit.
		6h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.

Table 5-92. Source Active Options Register (SAOPT) Field Descriptions (continued)

Bit	Field	Value	Description
6-4	PRI	0-7h	Transfer priority. Reflects the values programmed in the QUEPRI register in the EDMACC.
		0	Priority 0 - Highest priority
		1h-6h	Priority 1 to priority 6
		7h	Priority 7 - Lowest priority
3-2	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
1	DAM		Destination address mode within an array
		0	Increment (INCR) mode. Destination addressing within an array increments.
		1	Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width.
0	SAM		Source address mode within an array
		0	Increment (INCR) mode. Source addressing within an array increments.
		1	Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width.

5.4.2.6.2 Source Active Source Address Register (SASRC)

The source active source address register (SASRC) is shown in Figure 5-111 and described in Table 5-93.

Figure 5-111. Source Active Source Address Register (SASRC)

31	0
SADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-93. Source Active Source Address Register (SASRC) Field Descriptions

Bit	Field	Value	Description
31-0	SADDR	0-FFFF FFFFh	Source address for program register set. EDMA3TC updates value according to source addressing mode (SAM bit in the source active options register, SAOPT) .

5.4.2.6.3 Source Active Count Register (SACNT)

The source active count register (SACNT) is shown in Figure 5-112 and described in Table 5-94.

Figure 5-112. Source Active Count Register (SACNT)

31	16
BCNT	
R-0	
15	0
ACNT	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-94. Source Active Count Register (SACNT) Field Descriptions

Bit	Field	Value	Description
31-16	BCNT	0-FFFFh	B dimension count. Number of arrays to be transferred, where each array is ACNT in length. It is decremented after each read command appropriately. Represents the amount of data remaining to be read. It should be 0 when transfer request (TR) is complete.
15-0	ACNT	0-FFFFh	A dimension count. Number of bytes to be transferred in first dimension. It is decremented after each read command appropriately. Represents the amount of data remaining to be read. It should be 0 when transfer request (TR) is complete.

5.4.2.6.4 Source Active Destination Address Register (SADST)

The source active destination address register (SADST) is shown in Figure 5-113 and described in Table 5-95.

Figure 5-113. Source Active Destination Address Register (SADST)

31	16
Reserved	
R-0	

15	0
Reserved	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-95. Source Active Destination Address Register (SADST) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reserved. Always reads as 0.

5.4.2.6.5 Source Active Source B-Dimension Index Register (SABIDX)

The source active set B-dimension index register (SABIDX) is shown in Figure 5-114 and described in Table 5-96.

Figure 5-114. Source Active Source B-Dimension Index Register (SABIDX)

31	16
DBIDX	
R-0	

15	0
SBIDX	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-96. Source Active Source B-Dimension Index Register (SABIDX) Field Descriptions

Bit	Field	Value	Description
31-16	DBIDX	0	B-Index offset between destination arrays. Represents the offset in bytes between the starting address of each destination. Always reads as 0.
15-0	SBIDX	0-FFFFh	B-Index offset between source arrays. Represents the offset in bytes between the starting address of each source array.

5.4.2.6.6 Source Active Memory Protection Proxy Register (SAMPPRXY)

The source active memory protection proxy register (SAMPPRXY) is shown in Figure 5-115 and described in Table 5-97.

Figure 5-115. Source Active Memory Protection Proxy Register (SAMPPRXY)

31	16
Reserved	
R-0	

15	9	8	7	4	3	0
Reserved		PRIV	Reserved		PRIVID	
R-0		R-0	R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-97. Source Active Memory Protection Proxy Register (SAMPPRXY) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
8	PRIV		Privilege level. The privilege level used by the host to set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC. The privilege ID is used while issuing read and write command to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIV of the host that set up the DMA transaction.
		0	User-level privilege.
		1	Supervisor-level privilege.
7-4	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3-0	PRIVID	0-Fh	Privilege ID. This contains the privilege ID of the host that set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC. This PRIVID value is used while issuing read and write commands to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIVID of the host that set up the DMA transaction.

5.4.2.6.7 Source Active Count Reload Register (SACNTRLD)

The source active count reload register (SACNTRLD) is shown in Figure 5-116 and described in Table 5-98.

Figure 5-116. Source Active Count Reload Register (SACNTRLD)

31	16
Reserved	
R-0	

15	0
ACNTRLD	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-98. Source Active Count Reload Register (SACNTRLD) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
15-0	ACNTRLD	0-FFFFh	A-count reload value. Represents the originally programmed value of ACNT. The reload value is used to reinitialize ACNT after each array is serviced.

5.4.2.6.8 Source Active Source Address B-Reference Register (SASRCBREF)

The source active source address B-reference register (SASRCBREF) is shown in Figure 5-117 and described in Table 5-99.

Figure 5-117. Source Active Source Address B-Reference Register (SASRCBREF)

31	16
SADDRBREF	
R-0	

15	0
SADDRBREF	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-99. Source Active Source Address B-Reference Register (SASRCBREF) Field Descriptions

Bit	Field	Value	Description
31-0	SADDRBREF	0-FFFF FFFFh	Source address B-reference. Represents the starting address for the array currently being read.

5.4.2.6.9 Source Active Destination Address B-Reference Register (SADSTBREF)

The source active destination address B-reference register (SADSTBREF) is shown in Figure 5-118 and described in Table 5-100.

Figure 5-118. Source Active Destination Address B-Reference Register (SADSTBREF)

31	16
Reserved	
R-0	

15	0
Reserved	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-100. Source Active Destination Address B-Reference Register (SADSTBREF) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reserved. Always reads as 0.

5.4.2.6.10 Destination FIFO Options Register (DFOPT*n*)

The destination FIFO options register (DFOPT*n*) is shown in Figure 5-119 and described in Table 5-101.

NOTE: The value for *n* varies from 0 to DSTREGDEPTH for the given EDMA3TC.

Figure 5-119. Destination FIFO Options Register (DFOPT*n*)

31	23	22	21	20	19	18	17	16
Reserved		TCCHEN	Rsvd	TCINTEN	Reserved		TCC	
R-0		R/W-0	R-0	R/W-0	R-0		R/W-0	

15	12	11	10	8	7	6	4	3	2	1	0
TCC		Rsvd	FWID		Rsvd	PRI		Reserved		DAM	SAM
R/W-0		R-0	R/W-0		R-0	R/W-0		R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-101. Destination FIFO Options Register (DFOPT*n*) Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
22	TCCHEN		Transfer complete chaining enable
		0	Transfer complete chaining is disabled
		1	Transfer complete chaining is enabled
21	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
20	TCINTEN		Transfer complete interrupt enable.
		0	Transfer complete interrupt is disabled.
		1	Transfer complete interrupt is enabled.
19-18	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
17-12	TCC	0-3Fh	Transfer complete code. This 6-bit code is used to set the relevant bit in CER or IPR of the EDMA3PCC module.
11	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
10-8	FWID	0-7h	FIFO width. Applies if either SAM or DAM is set to constant addressing mode.
		0	FIFO width is 8-bit.
		1h	FIFO width is 16-bit.
		2h	FIFO width is 32-bit.
		3h	FIFO width is 64-bit.
		4h	FIFO width is 128-bit.
		5h	FIFO width is 256-bit.
		6h-7h	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
7	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
6-4	PRI	0-7h	Transfer priority
		0	Priority 0 - Highest priority
		1h-6h	Priority 1 to priority 6
		7h	Priority 7 - Lowest priority
3-2	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.

Table 5-101. Destination FIFO Options Register (DFOPT*n*) Field Descriptions (continued)

Bit	Field	Value	Description
1	DAM		Destination address mode within an array
		0	Increment (INCR) mode. Destination addressing within an array increments.
		1	Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width.
0	SAM		Source address mode within an array
		0	Increment (INCR) mode. Source addressing within an array increments.
		1	Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width.

5.4.2.6.11 Destination FIFO Source Address Register (DFSRC*n*)

The destination FIFO source address register (DFSRC*n*) is shown in Figure 5-120 and described in Table 5-102.

NOTE: The value for *n* varies from 0 to DSTREGDEPTH for the given EDMA3TC.

Figure 5-120. Destination FIFO Source Address Register (DFSRC*n*)

31	16
Reserved	
R-0	

15	0
Reserved	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-102. Destination FIFO Source Address Register (DFSRC*n*) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reserved. Always read as 0.

5.4.2.6.12 Destination FIFO Count Register (DFCNT*n*)

The destination FIFO count register (DFCNT*n*) is shown in Figure 5-121 and described in Table 5-103.

NOTE: The value for *n* varies from 0 to DSTREGDEPTH for the given EDMA3TC.

Figure 5-121. Destination FIFO Count Register (DFCNT*n*)

31	16
BCNT	
R-0	

15	0
ACNT	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-103. Destination FIFO Count Register (DFCNT*n*) Field Descriptions

Bit	Field	Value	Description
31-16	BCNT	0-FFFFh	B-dimension count. Number of arrays to be transferred, where each array is ACNT in length. Count/count remaining for destination register set. Represents the amount of data remaining to be written.
15-0	ACNT	0-FFFFh	A-dimension count. Number of bytes to be transferred in first dimension count/count remaining for destination register set. Represents the amount of data remaining to be written.

5.4.2.6.13 Destination FIFO Destination Address Register (DFDSTn)

The destination FIFO destination address register (DFDSTn) is shown in Figure 5-122 and described in Table 5-104.

NOTE: The value for *n* varies from 0 to DSTREGDEPTH for the given EDMA3TC.

Figure 5-122. Destination FIFO Destination Address Register (DFDSTn)

31	16
DADDR	
R-0	

15	0
DADDR	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-104. Destination FIFO Destination Address Register (DFDSTn) Field Descriptions

Bit	Field	Value	Description
31-0	DADDR	0	Destination address for the destination FIFO register set. When a transfer request (TR) is complete, the final value should be the address of the last write command issued.

5.4.2.6.14 Destination FIFO B-Index Register (DFBIDXn)

The destination FIFO B-index register (DFBIDXn) is shown in Figure 5-123 and described in Table 5-105.

NOTE: The value for *n* varies from 0 to DSTREGDEPTH for the given EDMA3TC.

Figure 5-123. Destination FIFO B-Index Register (DFBIDXn)

31	16
DBIDX	
R-0	

15	0
SBIDX	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-105. Destination FIFO B-Index Register (DFBIDXn) Field Descriptions

Bit	Field	Value	Description
31-16	DBIDX	0-FFFFh	B-Index offset between destination arrays. Represents the offset in bytes between the starting address of each destination.
15-0	SBIDX	0	Always read as 0.

5.4.2.6.15 Destination FIFO Memory Protection Proxy Register (DFMPPRXY*n*)

The destination FIFO memory protection proxy register (DFMPPRXY*n*) is shown in Figure 5-124 and described in Table 5-97.

Figure 5-124. Destination FIFO Memory Protection Proxy Register (DFMPPRXY*n*)

31	16
Reserved	
R-0	

15	9	8	7	4	3	0
Reserved		PRIV	Reserved		PRIVID	
R-0		R-0	R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-106. Destination FIFO Memory Protection Proxy Register (DFMPPRXY*n*) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
8	PRIV		Privilege level. This contains the Privilege level used by the EDMA3 programmer to set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC. The privilege ID is used while issuing read and write command to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIV of the host that set up the DMA transaction.
		0	User-level privilege
		1	Supervisor-level privilege
7-4	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
3-0	PRIVID	0-Fh	Privilege ID. This contains the Privilege ID of the EDMA3 programmer that set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC. This PRIVID value is used while issuing read and write commands to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIVID of the host that set up the DMA transaction.

5.4.2.6.16 Destination FIFO Count Reload Register (DFCNTRLD*n*)

The destination FIFO count reload register (DFCNTRLD*n*) is shown in Figure 5-125 and described in Table 5-107.

Figure 5-125. Destination FIFO Count Reload Register (DFCNTRLD*n*)

31	16
Reserved	
R-0	

15	0
ACNTRLD	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-107. Destination FIFO Count Reload Register (DFCNTRLD*n*) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
15-0	ACNTRLD	0-FFFFh	A-count reload value. Represents the originally programmed value of ACNT. The reload value is used to reinitialize ACNT after each array is serviced.

5.4.2.6.17 Destination FIFO Source Address B-Reference Register (DFSRCBREF*n*)

The destination FIFO source address B-reference register (DFSRCBREF*n*) is shown in Figure 5-126 and described in Table 5-108.

Figure 5-126. Destination FIFO Source Address B-Reference Register (DFSRCBREF*n*)

31	16
Reserved	
R-0	

15	0
Reserved	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-108. Destination FIFO Source Address B-Reference Register (DFSRCBREF*n*) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reserved. Always read as 0.

5.4.2.6.18 Destination FIFO Destination Address B-Reference (DFDSTBREF*n*)

The destination FIFO destination address B-reference register (DFDSTBREF*n*) is shown in Figure 5-127 and described in Table 5-109.

Figure 5-127. Destination FIFO Destination Address B-Reference Register (DFDSTBREF*n*)

31	16
DADDRBREF	
R-0	

15	0
DADDRBREF	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-109. Destination FIFO Destination Address B-Reference Register (DFDSTBREF*n*) Field Descriptions

Bit	Field	Value	Description
31-0	DADDRBREF	0-FFFF FFFFh	Destination address reference for the destination FIFO register set. Represents the starting address for the array currently being written.

5.5 Debug Checklist

This section lists some tips to keep in mind while debugging applications using the EDMA3.

The following table provides some common issues and their probable causes and resolutions.

Table 5-110. Debug List

Issue	Description/Solution
The transfer associated with the channel does not happen. The channel does not get serviced.	The EDMA3CC may not service a transfer request, even though the associated PaRAM set is programmed appropriately. Check for the following: 1) Verify that events are enabled, if an external/peripheral event is latched in Event Registers (ER/ERH), make sure that the event is enabled in the Event Enable Registers (EER/EERH). Similarly, for QDMA channels, make sure that QDMA events are appropriately enabled in the QDMA Event Enable Register (QEER). 2) Verify that the DMA or QDMA Secondary Event Register (SER/SERH/QSERH) bits corresponding to the particular event or channel are not set.
The Secondary Event Registers bits are set, not allowing additional transfers to occur on a channel.	It is possible that a trigger event was received when the parameter set associated with the channel/event was a NULL set for a previous transfer on the channel. This is typical in two cases: 1) QDMA channels: Typically if the parameter set is non-static and expected to be terminated by a NULL set (OPT.STATIC = 0, LINK = FFFFh), the parameter set is updated with a NULL set after submission of the last TR. Because QDMA channels are auto-triggered, this update caused the generation of an event. An event generated for a NULL set causes an error condition and results in setting the bits corresponding to the QDMA channel in the QEMR and QSER. This will disable further prioritization of the channel. 2) DMA channels used in a continuous mode: The peripheral may be set up to continuously generate infinite events (for instance, in case of McASP, every time the data shifts out from the DXR register, it generates an XEVT). The parameter set may be programmed to expect only a finite number of events and to be terminated by a NULL link. After the expected number of events, the parameter set is reloaded with a NULL parameter set. Because the peripheral will generate additional events, an error condition is set in the SER.Ex and EMR.Ex set, preventing further event prioritization. You must ensure that the number of events received is limited to the expected number of events for which the parameter set is programmed, or you must ensure that bits corresponding to particular channel or event are not set in the Secondary event registers (SER/SERH/QSER) and Event Missed Registers (EMR/EMRH/QEMR) before trying to perform subsequent transfers for the event/channel.
Completion interrupts are not asserted, or no further interrupts are received after the first completion interrupt.	You must ensure the following: 1) The interrupt generation is enabled in the OPT of the associated PaRAM set (TCINTEN = 1 and/or ITCINTEN = 1). 2) The interrupts are enabled in the EDMA3 Channel Controller, via the Interrupt Enable Registers (IER/IERH). 3) The corresponding interrupts are enabled in the device interrupt controller. 4) The set interrupts are cleared in the interrupt pending registers (IPR/IPRH) before exiting the transfer completion interrupt service routine (ISR). See Section 5.2.9.1.2 for details on writing EDMA3 ISRs. 5) If working with shadow region interrupts, make sure that the DMA Region Access registers (DRAE/DRAEH) are set up properly, because the DRAE/DRAEH registers act as secondary enables for shadow region completion interrupts, along with the IER/IERH registers. If working with shadow region interrupts, make sure that the bits corresponding to the transfer completion code (TCC) value are also enabled in the DRAE/DRAEH registers. For instance, if the PaRAM set associated with Channel 0 returns a completion code of 63 (OPT.TCC=63), ensure that DRAEH.E63 is also set for a shadow region completion interrupt because the interrupt pending register bit set will be IPRH.I63 (not IPR.I0).

5.6 Miscellaneous Programming/Debug Tips

1. For several registers, the setting and clearing of bits needs to be done via separate dedicated registers. For example, the Event Register (ER/ERH) can only be cleared by writing a 1 to the corresponding bits in the Event Clear Registers (ECR/ECRH). Similarly, the Event Enable Register (EER/EERH) bits can only be set with writes of 1 to the Event Enable Set Registers (EESR/EESRH) and cleared with writes of 1 to the corresponding bits in the Event Enable Clear Register (EECR/EECRH).
2. Writes to the shadow region memory maps are governed by region access registers (DRAE/DRAEH/QRAE). If the appropriate channels are not enabled in these registers, read/write access to the shadow region memory map is not enabled.
3. When working with shadow region completion interrupts, ensure that the DMA Region Access Registers (DRAE/DRAEH) for every region are set in a mutually exclusive way (unless it is a requirement for an application). If there is an overlap in the allocated channels and transfer completion codes (setting of Interrupt Pending Register bits) in the region resource allocation, it results in multiple shadow region completion interrupts. For example, if DRAE0.E0 and DRAE1.E0 are both set, then on completion of a transfer that returns a TCC=0, they will generate both shadow region 0 and 1 completion interrupts.
4. While programming a non-dummy parameter set, ensure the CCNT is not left to zero.
5. Enable the EDMA3CC error interrupt in the device controller and attach an interrupt service routine (ISR) to ensure that error conditions are not missed in an application and are appropriately addressed with the ISR.
6. Depending on the application, you may want to break large transfers into smaller transfers and use self-chaining to prevent starvation of other events in an event queue.
7. In applications where a large transfer is broken into sets of small transfers using chaining or other methods, you might choose to use the early chaining option to reduce the time between the sets of transfers and increase the throughput. However, keep in mind that with early completion, all data might have not been received at the end point when completion is reported because the EDMA3CC internally signals completion when the TR is submitted to the EDMA3TC, potentially before any data has been transferred.
8. The event queue entries can be observed to determine the last few events if there is a system failure (provided the entries were not bypassed).

5.7 Setting Up a Transfer

The following list provides a quick guide for the typical steps involved in setting up a transfer.

1. Initiating a DMA/QDMA channel
   (a) Determine the type of channel (QDMA or DMA) to be used.
   (b) Channel mapping
      (i) If using a QDMA channel, program the QCHMAP with the parameter set number to which the channel maps and the trigger word.
      (ii) If using a DMA channel, program the DCHMAP with the parameter set number to which the channel maps.
   (c) If the channel is being used in the context of a shadow region, ensure the DRAE/DRAEH for the region is properly set up to allow read write accesses to bits in the event registers and interrupt registers in the Shadow region memory map. The subsequent steps in this process should be done using the respective shadow region registers. (Shadow region descriptions and usage are provided in Section 5.2.7.1.)
   (d) Determine the type of triggering used.
      (i) If external events are used for triggering (DMA channels), enable the respective event in EER/EERH by writing into EESR/EESRH.
      (ii) If QDMA Channel is used, enable the channel in QEER by writing into QEESR.
   (e) Queue setup
      (i) If a QDMA channel is used, set up the QDMAQNUM to map the channel to the respective event queue.
      (ii) If a DMA channel is used, set up the DMAQNUM to map the event to the respective event queue.
2. Parameter set setup
   (a) Program the PaRAM set number associated with the channel. Note that if it is a QDMA channel, the PaRAM entry that is configured as trigger word is written to last. Alternatively, enable the QDMA channel (step 1-b-ii above) just before the write to the trigger word.
   See Section 5.3 for parameter set field setups for different types of transfers. See the sections on chaining (Section 5.2.8) and interrupt completion (Section 5.2.9) on how to set up final/intermediate completion chaining and/or interrupts.
3. Interrupt setup
   (a) Enable the interrupt in the IER/IERH by writing into IESR/IESRH.
   (b) Ensure that the EDMA3CC completion interrupt (either the global or the shadow region interrupt) is enabled properly in the device interrupt controller.
   (c) Ensure the EDMA3CC completion interrupt (this refers to either the Global interrupt or the shadow region interrupt) is enabled properly in the Device Interrupt controller.
   (d) Set up the interrupt controller properly to receive the expected EDMA3 interrupt.
4. Initiate transfer
   (a) This step is highly dependent on the event trigger source:
      (i) If the source is an external event coming from a peripheral, the peripheral will be enabled to start generating relevant EDMA3 events that can be latched to the ER transfer.
      (ii) For QDMA events, writes to the trigger word (step 2-a above) will initiate the transfer.
      (iii) Manually triggered transfers will be initiated by writes to the Event Set Registers (ESR/ESRH).
      (iv) Chained-trigger events initiate when a previous transfer returns a transfer completion code equal to the chained channel number.

5. Wait for completion
   (a) If the interrupts are enabled as mentioned in step 3 above, then the EDMA3CC will generate a completion interrupt to the CPU whenever transfer completion results in setting the corresponding bits in the interrupt pending register (IPR/IPRH). The set bits must be cleared in the IPR\IPRH by writing to corresponding bit in ICR\ICRH.
   (b) If polling for completion (interrupts not enabled in the device controller), then the application code can wait on the expected bits to be set in the IPR\IPRH. Again, the set bits in the IPR\IPRH must be manually cleared via ICR\ICRH before the next set of transfers is performed for the same transfer completion code values.

EMAC/MDIO Module

This chapter provides a functional description of the Ethernet Media Access Controller (EMAC) and physical layer (PHY) device Management Data Input/Output (MDIO) module integrated in the device. Included are the features of the EMAC and MDIO modules, a discussion of their architecture and operation, how these modules connect to the outside world, and a description of the registers for each module.

Topic **Page**

6.1 Introduction

6.1.1 Overview

The EMAC controls the flow of packet data from the system to the PHY. The MDIO module controls PHY configuration and status monitoring.

Both the EMAC and the MDIO modules interface to the system core through a custom interface that allows efficient data transmission and reception. This custom interface is referred to as the EMAC control module and is considered integral to the EMAC/MDIO peripheral.

The EMAC module is used to move data between this device and another host connected to the same network, in compliance with the Ethernet protocol.

6.1.2 Features

The EMAC/MDIO has the following features:

- Synchronous 10/100/1000 Mbps operation
- G/MII interface to the physical layer device (PHY)
- Full-duplex gigabit operation (half-duplex not supported)
- EMAC acts as DMA master to either internal or external device memory space
- Hardware error handling including CRC
- Eight receive channels with VLAN tag discrimination for receive quality-of-service (QOS) support
- Eight transmit channels with round-robin or fixed priority for transmit quality-of-service (QOS) support
- Ether-Stats and 802.3-Stats RMON statistics gathering
- Transmit CRC generation selectable on a per channel basis
- Broadcast frames selection for reception on a single channel
- Multicast frames selection for reception on a single channel
- Promiscuous receive mode frames selection for reception on a single channel (all frames, all good frames, short frames, error frames)
- Hardware flow control
- 8K-byte local EMAC descriptor memory that allows the peripheral to operate on descriptors without affecting the CPU. The descriptor memory holds enough information to transfer up to 512 Ethernet packets without CPU intervention.
- Programmable interrupt logic permits the software driver to restrict the generation of back-to-back interrupts, which allows more work to be performed in a single call to the interrupt service routine.
- TI Adaptive Performance Optimization for improved half duplex performance
- Configurable receive address matching/filtering, receive FIFO depth, and transmit FIFO depth
- No-chain mode truncates frame to first buffer for network analysis applications
- Emulation support
- Loopback mode

6.1.3 Functional Block Diagram

Figure 6-1 shows the three main functional modules of the EMAC/MDIO peripheral:

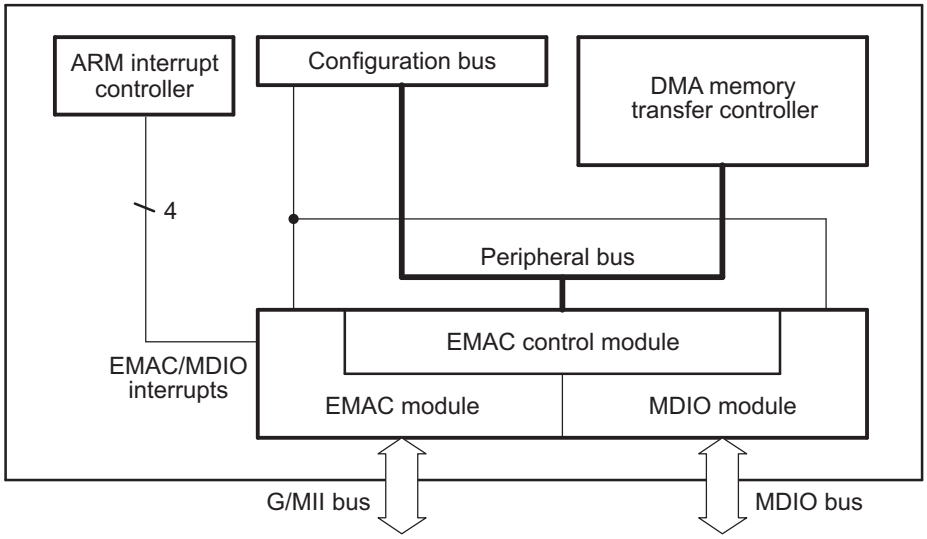
- EMAC control module
- EMAC module
- MDIO module

The EMAC control module is the main interface between the device core processor and the EMAC module and MDIO module. The EMAC control module contains the necessary components to allow the EMAC to make efficient use of device memory, plus it controls device interrupts. The EMAC control module incorporates 8K-byte internal RAM to hold EMAC buffer descriptors.

The MDIO module implements the 802.3 serial management interface to interrogate and control up to 32 Ethernet PHYs connected to the device, using a shared two-wire bus. Host software uses the MDIO module to configure the autonegotiation parameters of each PHY attached to the EMAC, retrieve the negotiation results, and configure required parameters in the EMAC module for correct operation. The module is designed to allow almost transparent operation of the MDIO interface, with very little maintenance from the core processor.

The EMAC module provides an efficient interface between the processor and the networked community. The EMAC on this device supports 10Base-T (10 Mbits/second) and 100BaseTX (100 Mbits/second) in either half-duplex or full-duplex mode and 1000BaseT (1000 Mbits/second) in full-duplex mode, with hardware flow control and quality-of-service (QOS) support.

Figure 6-1. EMAC and MDIO Block Diagram



6.1.4 EMAC and MDIO Block Diagram

Figure 6-1 also shows the main interface between the EMAC control module and the CPU. The following connections are made to the device core:

- The peripheral bus connection from the EMAC control module allows the EMAC module to read and write both internal and external memory through the DMA memory transfer controller.
- The EMAC control module, EMAC, and MDIO all have control registers. These registers are memory-mapped into device memory space via the device configuration bus. Along with these registers, the control module's internal RAM is mapped into this same range.
- The EMAC and MDIO interrupts are combined into a single interrupt within the control module. The interrupt from the control module then goes to the ARM interrupt controller.

The EMAC and MDIO interrupts are combined within the control module, so only the control module interrupt needs to be monitored by the application software or device driver. The EMAC control module combines the EMAC and MDIO interrupts and generates 4 separate interrupts to the ARM through the ARM interrupt controller. See Section 6.2.16.4 for details of interrupt multiplex logic of the EMAC control module.

6.1.5 Industry Standard(s) Compliance Statement

The EMAC peripheral conforms to the IEEE 802.3 standard, describing the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer specifications. The IEEE 802.3 standard has also been adopted by ISO/IEC and re-designated as ISO/IEC 8802-3:2000(E).

In difference from this standard, the EMAC peripheral does not use the Transmit Coding Error signal MTXER. Instead of driving the error pin when an underflow condition occurs on a transmitted frame, the EMAC intentionally generates an incorrect checksum by inverting the frame CRC, so that the transmitted frame is detected as an error by the network.

6.1.6 List of Terms

Broadcast MAC Address— A special Ethernet MAC address used to send data to all Ethernet devices on the local network. The broadcast address is FFh-FFh-FFh-FFh-FFh-FFh. The LSB of the first byte is odd, qualifying it as a group address; however, its value is reserved for broadcast. It is classified separately by the EMAC.

Descriptor (Packet Buffer Descriptor)— A small memory structure that describes a larger block of memory in terms of size, location, and state. Descriptors are used by the EMAC and application to describe the memory buffers that hold Ethernet data.

Device — In this document, device refers to the processor.

Ethernet MAC Address (MAC Address)— A unique 6-byte address that identifies an Ethernet device on the network. In an Ethernet packet, a MAC address is used twice, first to identify the packet's destination, and second to identify the packet's sender or source. An Ethernet MAC address is normally specified in hexadecimal, using dashes to separate bytes. For example, 08h-00h-28h-32h-17h-42h.

The first three bytes normally designate the manufacturer of the device. However, when the first byte of the address is odd (LSB is 1), the address is a group address (broadcast or multicast). The second bit specifies whether the address is globally or locally administrated (not considered in this document).

Ethernet Packet (Packet)— An Ethernet packet is the collection of bytes that represents the data portion of a single Ethernet frame on the wire.

Full Duplex— Full-duplex operation allows simultaneous communication between a pair of stations using point-to-point media (dedicated channel). Full-duplex operation does not require that transmitters defer, nor do they monitor or react to receive activity, as there is no contention for a shared medium in this mode. Full-duplex mode can only be used when all of the following are true:

- The physical medium is capable of supporting simultaneous transmission and reception without interference.
- There are exactly two stations connected with a full duplex point-to-point link. As there is no contention for use of a shared medium, the multiple access (that is, CSMA/CD) algorithms are unnecessary.
- Both stations on the LAN are capable of, and have been configured to use, full-duplex operation.

The most common configuration envisioned for full-duplex operation consists of a central bridge (also known as a switch) with a dedicated LAN connecting each bridge port to a single device.

Full-duplex operation constitutes a proper subset of the MAC functionality required for half-duplex operation.

Half Duplex— In half-duplex mode, the CSMA/CD media access method is the means by which two or more stations share a common transmission medium. To transmit, a station waits (defers) for a quiet period on the medium, that is, no other station is transmitting. It then sends the intended message in bit-serial form. If, after initiating a transmission, the message collides with that of another station, then each transmitting station intentionally transmits for an additional predefined period to ensure propagation of the collision throughout the system. The station remains silent for a random amount of time (backoff) before attempting to transmit again.

Host— The host is an intelligent system resource that configures and manages each communications control module. The host is responsible for allocating memory, initializing all data structures, and responding to port (EMAC) interrupts. In this document, host refers to the device.

Jabber— A condition wherein a station transmits for a period of time longer than the maximum permissible packet length, usually due to a fault condition.

Link— The transmission path between any two instances of generic cabling.

Multicast MAC Address— A class of MAC address that sends a packet to potentially more than one recipient. A group address is specified by setting the LSB of the first MAC address byte to 1. Thus, 01h-02h-03h-04h-05h-06h is a valid multicast address. Typically, an Ethernet MAC looks for only certain multicast addresses on a network to reduce traffic load. The multicast address list of acceptable packets is specified by the application.

Physical Layer and Media Notation— To identify different Ethernet technologies, a simple, three-field, type notation is used. The Physical Layer type used by the Ethernet is specified by these fields:

<data rate in Mb/s><medium type><maximum segment length (×100m)>

The definitions for the technologies mentioned in this document are in Table 6-1.

Table 6-1. Physical Layer Definitions

Term	Definition
10Base-T	IEEE 802.3 Physical Layer specification for a 10 Mb/s CSMA/CD local area network over two pairs of twisted-pair telephone wire.
100Base-T	IEEE 802.3 Physical Layer specification for a 100 Mb/s CSMA/CD local area network over two pairs of Category 5 unshielded twisted-pair (UTP) or shielded twisted-pair (STP) wire.
Twisted pair	A cable element that consists of two insulated conductors twisted together in a regular fashion to form a balanced transmission line.

Port— Ethernet device.

Promiscuous Mode— EMAC receives frames that do not match its address.

6.2 Architecture

This section discusses the architecture and basic function of the EMAC/MDIO module.

6.2.1 Clock Control

The frequencies for the transmit and receive clocks are fixed by the IEEE 802.3 specification as:

- 2.5 MHz at 10 Mbps
- 25 MHz at 100 Mbps
- 125 MHz at 1000 Mbps

All EMAC logic is clocked synchronously with the peripheral clock (SYSCLK5). The MDIO clock can be controlled through the application software, by programming the divide-down factor in the MDIO control register (CONTROL).

6.2.1.1 MII Clocking

In the 10/100 Mbps mode, the transmit and receive clock sources are provided from an external PHY via the MTCLK and MRCLK pins. These clocks are inputs to the EMAC module and operate at 2.5 MHz in 10 Mbps mode and at 25 MHz in 100 Mbps mode. The MII clocking interface is not used in 1000 Mbps mode. For timing purposes, data is transmitted and received with reference to MTCLK and MRCLK, respectively.

6.2.1.2 GMII Clocking

In the 1000 Mbps mode, the transmit and receive clock sources for 10/100 Mbps operation are provided from an external PHY via the MTCLK and MRCLK pins, as in the MII clocking. For 1000 Mbps operation, the receive clock is provided by an external PHY via the MRCLK pin. For transmit in 1000 Mbps mode, the clock is sourced synchronous with the data and is provided by the EMAC to be output on the GMTCLK pin.

The EMAC module is internally clocked at 148.5 MHz. For timing purposes, data in 10/100 Mbps mode is transmitted and received with reference to MTCLK and MRCLK, respectively. For 1000 Mbps mode, receive timing is the same, but transmit is relative to GMTCLK.

6.2.2 Memory Map

The EMAC peripheral includes internal memory that is used to hold information about the Ethernet packets received and transmitted. This internal RAM is 2K × 32 bits in size. Data can be written to and read from the EMAC internal memory by either the EMAC or the CPU. It is used to store buffer descriptors that are 4-words (16-bytes) deep. This 8K local memory holds enough information to transfer up to 512 Ethernet packets without CPU intervention.

The packet buffer descriptors can also be placed in the internal processor memory, or in EMIF memory (DDR). There are some tradeoffs in terms of cache performance and throughput when descriptors are placed in the system memory, versus when they are placed in the EMAC's internal memory. Cache performance is improved when the buffer descriptors are placed in internal memory. However, the EMAC throughput is better when the descriptors are placed in the local EMAC RAM.

6.2.3 Signal Descriptions

The device supports both MII interface (for 10/100 Mbps operation) and GMII interface (for 10/100/1000 Mbps) operation.

Figure 6-2 shows a device with integrated EMAC and MDIO interfaced via a GMII connection. This interface is available in 10 Mbps, 100 Mbps, and 1000 Mbps modes.

The GMII interface supports 10/100/1000 Mbps modes. Only full-duplex mode is available in 1000 Mbps mode. In 10/100 Mbps modes, the GMII interface acts like an MII interface and only the lower 4 bits of data are transferred for each of the data buses. The individual EMAC and MDIO signals for the GMII interface are summarized in Figure 6-2.

Figure 6-2. Ethernet Configuration—GMII Connections

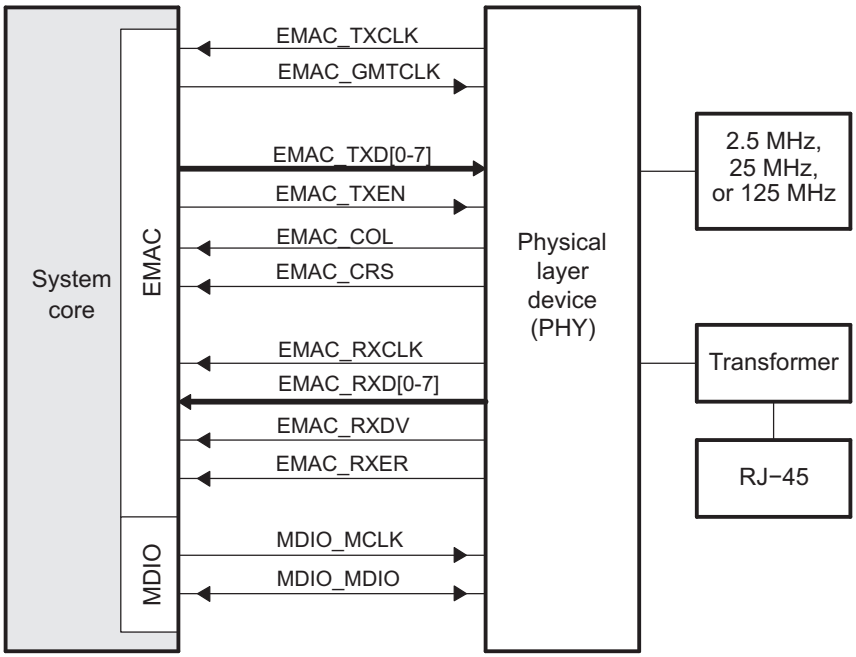


Table 6-2. EMAC and MDIO Signals for GMII Interface

Signal	Type	Description
EMAC_TXCLK	I	Transmit clock (EMAC_TXCLK). The transmit clock is a continuous clock that provides the timing reference for transmit operations in 10/100 Mbps mode. The EMAC_TXD and EMAC_TXEN signals are tied to this clock when in 10/100 Mbps mode. The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation, and 25 MHz at 100 Mbps operation.
EMAC_GMTCLK	O	GMII source asynchronous transmit clock (EMAC_GMTCLK). This clock is used in 1000 Mbps mode only, providing a continuous 125 MHz frequency for transmit operations. The EMAC_TXD and EMAC_TXEN signals are tied to this clock when in Gigabit mode. The clock is generated by the EMAC and is 125 MHz.
EMAC_TXD[0-7]	O	Transmit data (EMAC_TXD). The transmit data pins are a collection of 8 data signals comprising 8 bits of data. EMAC_TXD[0] is the least-significant bit (LSB). The signals are synchronized by EMAC_TXCLK in 10/100 Mbps mode, and by EMAC_GMTCLK in Gigabit mode, and valid only when EMAC_TXEN is asserted.
EMAC_TXEN	O	Transmit data enable (EMAC_TXEN). The transmit data enable signal indicates that the EMAC_TXD pins are generating nibble data for use by the PHY. It is driven synchronously to EMAC_TXCLK in 10/100 Mbps mode, and to EMAC_GMTCLK in Gigabit mode.
EMAC_COL	I	Collision detected (EMAC_COL). The EMAC_COL pin is asserted by the PHY when it detects a collision on the network. It remains asserted while the collision condition persists. This signal is not necessarily synchronous to EMAC_TXCLK nor EMAC_RXCLK. This pin is used in half-duplex operation only.
EMAC_CRS	I	Carrier sense (EMAC_CRS). The EMAC_CRS pin is asserted by the PHY when the network is not idle in either transmit or receive. The pin is deasserted when both transmit and receive are idle. This signal is not necessarily synchronous to EMAC_TXCLK nor EMAC_RXCLK. This pin is used in half-duplex operation only.
EMAC_RXCLK	I	Receive clock (EMAC_RXCLK). The receive clock is a continuous clock that provides the timing reference for receive operations. The EMAC_RXD, EMAC_RXDV, and EMAC_RXER signals are tied to this clock. The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation, 25 MHz at 100 Mbps operation and 125 MHz at 1000 Mbps operation.
EMAC_RXD[0-7]	I	Receive data (EMAC_RXD). The receive data pins are a collection of 8 data signals comprising 8 bits of data. EMAC_RXD[0] is the least-significant bit (LSB). The signals are synchronized by EMAC_RXCLK and valid only when EMAC_RXDV is asserted.
EMAC_RXDV	I	Receive data valid (EMAC_RXDV). The receive data valid signal indicates that the EMAC_RXD pins are generating nibble data for use by the EMAC. It is driven synchronously to EMAC_RXCLK.
EMAC_RXER	I	Receive data error (EMAC_RXER). The receive data error signal is asserted for one or more EMAC_RXCLK periods to indicate that an error was detected in the received frame. This is meaningful only during data reception when EMAC_RXDV is active.
MDIO_MCLK	O	Management data serial clock (MDIO_MCLK). The MDIO data serial clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO_MDIO pin. The frequency of this clock is controlled by the CLKDIV bits in the MDIO control register (CONTROL).
MDIO_MDIO	I/O	Management data input/output (MDIO_MDIO). The MDIO_MDIO pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MDIO_MDIO pin acts as an output for everything except the data bit cycles, when the pin acts as an input for read operations.

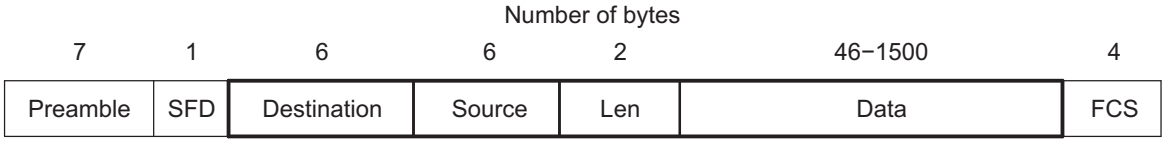
6.2.4 Ethernet Protocol Overview

Ethernet provides an unreliable, connection-less service to a networking application. A brief overview of the Ethernet protocol is given in the following subsections. For in-depth information on the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method, which is the Ethernet's multiple access protocol, see the IEEE 802.3 standard document.

6.2.4.1 Ethernet Frame Format

The format of an Ethernet frame is shown in Figure 6-3 and described in Table 6-3.

Figure 6-3. Ethernet Frame Format



Legend: SFD = Start Frame Delimeter; FCS = Frame Check Sequence (CRC)

Table 6-3. Ethernet Frame Description

Field	Bytes	Description
Preamble	7	Preamble. These 7 bytes have a fixed value of 55h and serve to wake up the receiving EMAC ports and to synchronize their clocks to that of the sender's clock.
SFD	1	Start of Frame Delimiter. This field with a value of 5Dh immediately follows the preamble pattern and indicates the start of important data.
Destination	6	Destination address. This field contains the Ethernet MAC address of the EMAC port for which the frame is intended. It may be an individual or multicast (including broadcast) address. When the destination EMAC port receives an Ethernet frame with a destination address that does not match any of its MAC physical addresses, and no promiscuous, multicast or broadcast channel is enabled, it discards the frame.
Source	6	Source address. This field contains the MAC address of the Ethernet port that transmits the frame to the Local Area Network.
Len	2	Length/Type field. The length field indicates the number of EMAC client data bytes contained in the subsequent data field of the frame. This field can also be used to identify the type of data the frame is carrying.
Data	46 to (RXMAXLEN - 18)	Data field. This field carries the datagram containing the upper layer protocol frame, that is, IP layer datagram. The maximum transfer unit (MTU) of Ethernet is (RXMAXLEN - 18) bytes. This means that if the upper layer protocol datagram exceeds (RXMAXLEN - 18) bytes, then the host has to fragment the datagram and send it in multiple Ethernet packets. The minimum size of the data field is 46 bytes. This means that if the upper layer datagram is less then 46 bytes, the data field has to be extended to 46 bytes by appending extra bits after the data field, but prior to calculating and appending the FCS.
FCS	4	Frame Check Sequence. A cyclic redundancy check (CRC) is used by the transmit and receive algorithms to generate a CRC value for the FCS field. The frame check sequence covers the 60 to (RXMAXLEN - 4) bytes of the packet data. Note that this 4-byte field may or may not be included as part of the packet data, depending on how the EMAC is configured.

6.2.4.2 Ethernet's Multiple Access Protocol

Nodes in an Ethernet Local Area Network are interconnected by a broadcast channel, as a result, when an EMAC port transmits a frame, all the adapters on the local network receive the frame. Carrier Sense Multiple Access with Collision Detection (CSMA/CD) algorithms are used when the EMAC operates in half-duplex mode. When operating in full-duplex mode, there is no contention for use of a shared medium, since there are exactly two ports on the local network.

Each port runs the CSMA/CD protocol without explicit coordination with the other ports on the Ethernet network. Within a specific port, the CSMA/CD protocol works as follows:

1. The port obtains data from upper layers protocols at its node, prepares an Ethernet frame, and puts the frame in a buffer.
2. If the port senses that the medium is idle it starts to transmit the frame. If the port senses that the transmission medium is busy, it waits until it senses no signal energy (plus an Inter-Packet Gap time) and then starts to transmit the frame.
3. While transmitting, the port monitors for the presence of signal energy coming from other ports. If the port transmits the entire frame without detecting signal energy from other Ethernet devices, the port is done with the frame.
4. If the port detects signal energy from other ports while transmitting, it stops transmitting its frame and instead transmits a 48-bit jam signal.
5. After transmitting the jam signal the port enters an exponential backoff phase. Specifically, when transmitting a given frame, after experiencing a number of collisions in a row for the frame, the port chooses a random value that is dependent on the number of collisions. The port then waits an amount of time that is multiple of this random value, and returns to step 2.

6.2.5 Programming Interface

6.2.5.1 Packet Buffer Descriptors

The buffer descriptor is a central part of the EMAC module and is how the application software describes Ethernet packets to be sent and empty buffers to be filled with incoming packet data. The basic descriptor format is shown in Figure 6-4 and described in Table 6-4.

For example, consider three packets to be transmitted, Packet A is a single fragment (60 bytes), Packet B is fragmented over three buffers (1514 bytes total), and Packet C is a single fragment (1514 bytes). The linked list of descriptors to describe these three packets is shown in Figure 6-5.

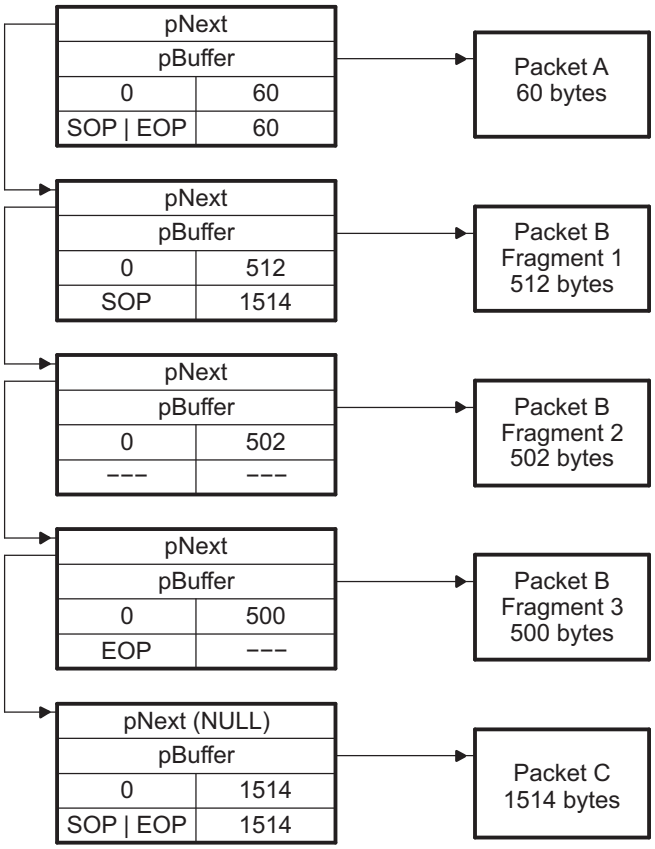
Figure 6-4. Basic Descriptor Format

	Bit Fields	
Word Offset	31 16	15 0
0	Next Descriptor Pointer	
1	Buffer Pointer	
2	Buffer Offset	Buffer Length
3	Flags	Packet Length

Table 6-4. Basic Descriptor Description

Word Offset	Field	Field Description
0	Next Descriptor Pointer	The next descriptor pointer is used to create a single linked list of descriptors. Each descriptor describes a packet or a packet fragment. When a descriptor points to a single buffer packet or the first fragment of a packet, the start of packet (SOP) flag is set in the flags field. When a descriptor points to a single buffer packet or the last fragment of a packet, the end of packet (EOP) flag is set. When a packet is fragmented, each fragment must have its own descriptor and appear sequentially in the descriptor linked list.
1	Buffer Pointer	The buffer pointer refers to the actual memory buffer that contains packet data during transmit operations, or is an empty buffer ready to receive packet data during receive operations.
2	Buffer Offset	The buffer offset is the offset from the start of the packet buffer to the first byte of valid data. This field only has meaning when the buffer descriptor points to a buffer that actually contains data.
	Buffer Length	The buffer length is the actual number of valid packet data bytes stored in the buffer. If the buffer is empty and waiting to receive data, this field represents the size of the empty buffer.
3	Flags	The flags field contains more information about the buffer, such as, is it the first fragment in a packet (SOP), the last fragment in a packet (EOP), or contains an entire contiguous Ethernet packet (both SOP and EOP). The flags are described in Section 6.2.5.4 and Section 6.2.5.5.
	Packet Length	The packet length only has meaning for buffers that both contain data and are the start of a new packet (SOP). In the case of SOP descriptors, the packet length field contains the length of the entire Ethernet packet, regardless if it is contained in a single buffer or fragmented over several buffers.

Figure 6-5. Typical Descriptor Linked List



6.2.5.2 Transmit and Receive Descriptor Queues

The EMAC module processes descriptors in linked list chains as discussed in Section 6.2.5.1. The lists controlled by the EMAC are maintained by the application software through the use of the head descriptor pointer registers (HDP). Since the EMAC supports eight channels for both transmit and receive, there are eight head descriptor pointer registers for both. They are:

- TX*n*HDP - Transmit Channel *n* DMA Head Descriptor Pointer Register
- RX*n*HDP - Receive Channel *n* DMA Head Descriptor Pointer Register

After an EMAC reset and before enabling the EMAC for send or receive, all 16 head descriptor pointer registers must be initialized to 0.

The EMAC uses a simple system to determine if a descriptor is currently owned by the EMAC or by the application software. There is a flag in the buffer descriptor flags called OWNER. When this flag is set, the packet that is referenced by the descriptor is considered to be owned by the EMAC. Note that ownership is done on a packet based granularity, not on descriptor granularity, so only SOP descriptors make use of the OWNER flag. As packets are processed, the EMAC patches the SOP descriptor of the corresponding packet and clears the OWNER flag. This is an indication that the EMAC has finished processing all descriptors up to and including the first with the EOP flag set, indicating the end of the packet (note this may only be one descriptor with both the SOP and EOP flags set).

To add a descriptor or a linked list of descriptors to an EMAC descriptor queue for the first time, the software application simply writes the pointer to the descriptor or first descriptor of a list to the corresponding HDP register. Note that the last descriptor in the list must have its “next” pointer cleared to 0. This is the only way the EMAC has of detecting the end of the list. So in the case where only a single descriptor is added, its “next descriptor” pointer must be initialized to 0.

The HDP must never be written to a second time while a previous list is active. To add additional descriptors to a descriptor list already owned by the EMAC, the NULL “next” pointer of the last descriptor of the previous list is patched with a pointer to the first descriptor in the new list. The list of new descriptors to be appended to the existing list must itself be NULL terminated before the pointer patch is performed.

There is a potential race condition where the EMAC may read the “next” pointer of a descriptor as NULL in the instant before an application appends additional descriptors to the list by patching the pointer. This case is handled by the software application always examining the buffer descriptor flags of all EOP packets, looking for a special flag called end of queue (EOQ). The EOQ flag is set by the EMAC on the last descriptor of a packet when the descriptor’s “next” pointer is NULL. This is the way the EMAC indicates to the software application that it believes it has reached the end of the list. When the software application sees the EOQ flag set, and there are more descriptors to process, the application may at that time submit the new list, or the portion of the appended list that was missed, by writing the new list pointer to the same HDP that started the process.

This process applies when adding packets to a transmit list, and empty buffers to a receive list.

6.2.5.3 Transmit and Receive EMAC Interrupts

The EMAC processes descriptors in linked list chains as discussed in Section 6.2.5.1, using the linked list queue mechanism discussed in Section 6.2.5.2.

The EMAC synchronizes descriptor list processing through the use of interrupts to the software application. The interrupts are controlled by the application using the interrupt masks, global interrupt enable, and the completion pointer register (CP). The CP is also called the interrupt acknowledge register.

As the EMAC supports eight channels for both transmit and receive, there are eight completion pointer registers for both. They are:

- TX*n*CP - Transmit Channel *n* Completion Pointer (Interrupt Acknowledge) Register
- RX*n*CP - Receive Channel *n* Completion Pointer (Interrupt Acknowledge) Register

These registers serve two purposes. When read, they return the pointer to the last descriptor that the EMAC has processed. When written by the software application, the value represents the last descriptor processed by the software application. When these two values do not match, the interrupt remains asserted, after the respective End of interrupt bit is signaled in the EMAC control module.

The system configuration determines whether or not an active interrupt actually interrupts the CPU. In general, the individual interrupts for different events from the EMAC and MDIO must be enabled in the EMAC control module, and it also must be mapped in the ARM interrupt controller and enabled as a CPU interrupt. If the system is configured properly, the interrupt for a specific receive or transmit channel executes under the previously described conditions when the corresponding interrupt is enabled in the EMAC using the receive interrupt mask set register (RXINTMASKSET) or the transmit interrupt mask set register (TXINTMASKSET).

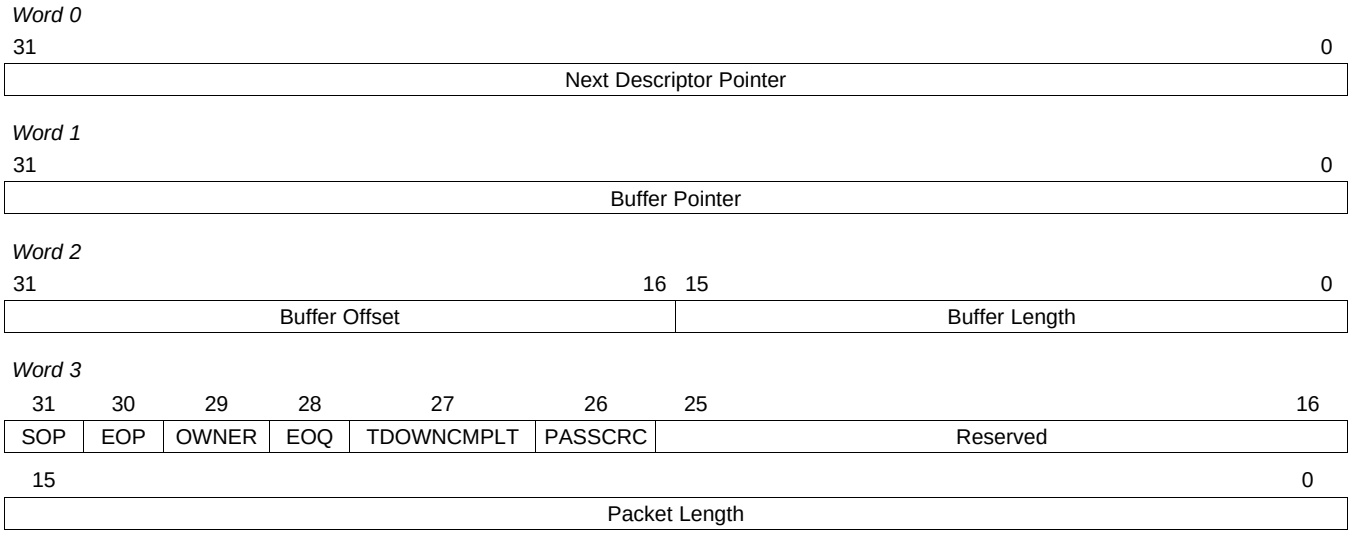
Whether or not the interrupt is enabled, the current state of the receive or transmit channel interrupt can be examined directly by the software application reading the receive interrupt status (unmasked) register (RXINTSTATRAW) and transmit interrupt status (unmasked) register (TXINTSTATRAW).

Interrupts are acknowledged when the application software updates the value of TX*n*CP or RX*n*CP with a value that matches the internal value kept by the EMAC. This mechanism ensures that the application software never misses an EMAC interrupt, since the interrupt and its acknowledgment are tied directly to the actual buffer descriptors processing.

6.2.5.4 Transmit Buffer Descriptor Format

A transmit (TX) buffer descriptor (Figure 6-6) is a contiguous block of four 32-bit data words aligned on a 32-bit boundary that describes a packet or a packet fragment. Example 6-1 shows the transmit buffer descriptor described by a C structure.

Figure 6-6. Transmit Buffer Descriptor Format



Example 6-1. Transmit Buffer Descriptor in C Structure Format

```
/*
// EMAC Descriptor
//
// The following is the format of a single buffer descriptor
// on the EMAC.
*/
typedef struct _EMAC_Desc {
 struct _EMAC_Desc *pNext; /* Pointer to next descriptor in chain */
 Uint8 *pBuffer; /* Pointer to data buffer */
 Uint32 BufOffLen; /* Buffer Offset(MSW) and Length(LSW) */
 Uint32 PktFlgLen; /* Packet Flags(MSW) and Length(LSW) */
} EMAC_Desc;

/* Packet Flags */
#define EMAC_DSC_FLAG_SOP 0x80000000u
#define EMAC_DSC_FLAG_EOP 0x40000000u
#define EMAC_DSC_FLAG_OWNER 0x20000000u
#define EMAC_DSC_FLAG_EOQ 0x10000000u
#define EMAC_DSC_FLAG_TDOWNCMPLT 0x08000000u
#define EMAC_DSC_FLAG_PASSCRC 0x04000000u
```

6.2.5.4.1 Next Descriptor Pointer

The next descriptor pointer points to the 32-bit word aligned memory address of the next buffer descriptor in the transmit queue. This pointer is used to create a linked list of buffer descriptors. If the value of this pointer is zero, then the current buffer is the last buffer in the queue. The software application must set this value prior to adding the descriptor to the active transmit list. This pointer is not altered by the EMAC.

The value of pNext should never be altered once the descriptor is in an active transmit queue, unless its current value is NULL. If the pNext pointer is initially NULL, and more packets need to be queued for transmit, the software application may alter this pointer to point to a newly appended descriptor. The EMAC will use the new pointer value and proceed to the next descriptor unless the pNext value has already been read. In this latter case, the transmitter will halt on the transmit channel in question, and the software application may restart it at that time. The software can detect this case by checking for an end of queue (EOQ) condition flag on the updated packet descriptor when it is returned by the EMAC.

6.2.5.4.2 Buffer Pointer

The buffer pointer is the byte-aligned memory address of the memory buffer associated with the buffer descriptor. The software application must set this value prior to adding the descriptor to the active transmit list. This pointer is not altered by the EMAC.

6.2.5.4.3 Buffer Offset

This 16-bit field indicates how many unused bytes are at the start of the buffer. For example, a value of 0000h indicates that no unused bytes are at the start of the buffer and that valid data begins on the first byte of the buffer, while a value of 000Fh indicates that the first 15 bytes of the buffer are to be ignored by the EMAC and that valid buffer data starts on byte 16 of the buffer. The software application must set this value prior to adding the descriptor to the active transmit list. This field is not altered by the EMAC.

Note that this value is only checked on the first descriptor of a given packet (where the start of packet (SOP) flag is set). It can not be used to specify the offset of subsequent packet fragments. Also, since the buffer pointer may point to any byte–aligned address, this field may be entirely superfluous, depending on the device driver architecture.

The range of legal values for this field is 0 to (Buffer Length – 1).

6.2.5.4.4 Buffer Length

This 16-bit field indicates how many valid data bytes are in the buffer. On single fragment packets, this value is also the total length of the packet data to be transmitted. If the buffer offset field is used, the offset bytes are not counted as part of this length. This length counts only valid data bytes. The software application must set this value prior to adding the descriptor to the active transmit list. This field is not altered by the EMAC.

6.2.5.4.5 Packet Length

This 16-bit field specifies the number of data bytes in the entire packet. Any leading buffer offset bytes are not included. The sum of the buffer length fields of each of the packet's fragments (if more than one) must be equal to the packet length. The software application must set this value prior to adding the descriptor to the active transmit list. This field is not altered by the EMAC. This value is only checked on the first descriptor of a given packet (where the start of packet (SOP) flag is set).

6.2.5.4.6 Start of Packet (SOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is the start of a new packet. In the case of a single fragment packet, both the SOP and end of packet (EOP) flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet sets the EOP flag. This bit is set by the software application and is not altered by the EMAC.

6.2.5.4.7 End of Packet (EOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is last for a given packet. In the case of a single fragment packet, both the start of packet (SOP) and EOP flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet sets the EOP flag. This bit is set by the software application and is not altered by the EMAC.

6.2.5.4.8 Ownership (OWNER) Flag

When set, this flag indicates that all the descriptors for the given packet (from SOP to EOP) are currently owned by the EMAC. This flag is set by the software application on the SOP packet descriptor before adding the descriptor to the transmit descriptor queue. For a single fragment packet, the SOP, EOP, and OWNER flags are all set. The OWNER flag is cleared by the EMAC once it is finished with all the descriptors for the given packet. Note that this flag is valid on SOP descriptors only.

6.2.5.4.9 End of Queue (EOQ) Flag

When set, this flag indicates that the descriptor in question was the last descriptor in the transmit queue for a given transmit channel, and that the transmitter has halted. This flag is initially cleared by the software application prior to adding the descriptor to the transmit queue. This bit is set by the EMAC when the EMAC identifies that a descriptor is the last for a given packet (the EOP flag is set), and there are no more descriptors in the transmit list (next descriptor pointer is NULL).

The software application can use this bit to detect when the EMAC transmitter for the corresponding channel has halted. This is useful when the application appends additional packet descriptors to a transmit queue list that is already owned by the EMAC. Note that this flag is valid on EOP descriptors only.

6.2.5.4.10 Teardown Complete (TDOWNCMPLT) Flag

This flag is used when a transmit queue is being torn down, or aborted, instead of allowing it to be transmitted. This would happen under device driver reset or shutdown conditions. The EMAC sets this bit in the SOP descriptor of each packet as it is aborted from transmission.

Note that this flag is valid on SOP descriptors only. Also note that only the first packet in an unsent list has the TDOWNCMPLT flag set. Subsequent descriptors are not even processed by the EMAC.

6.2.5.4.11 Pass CRC (PASSCRC) Flag

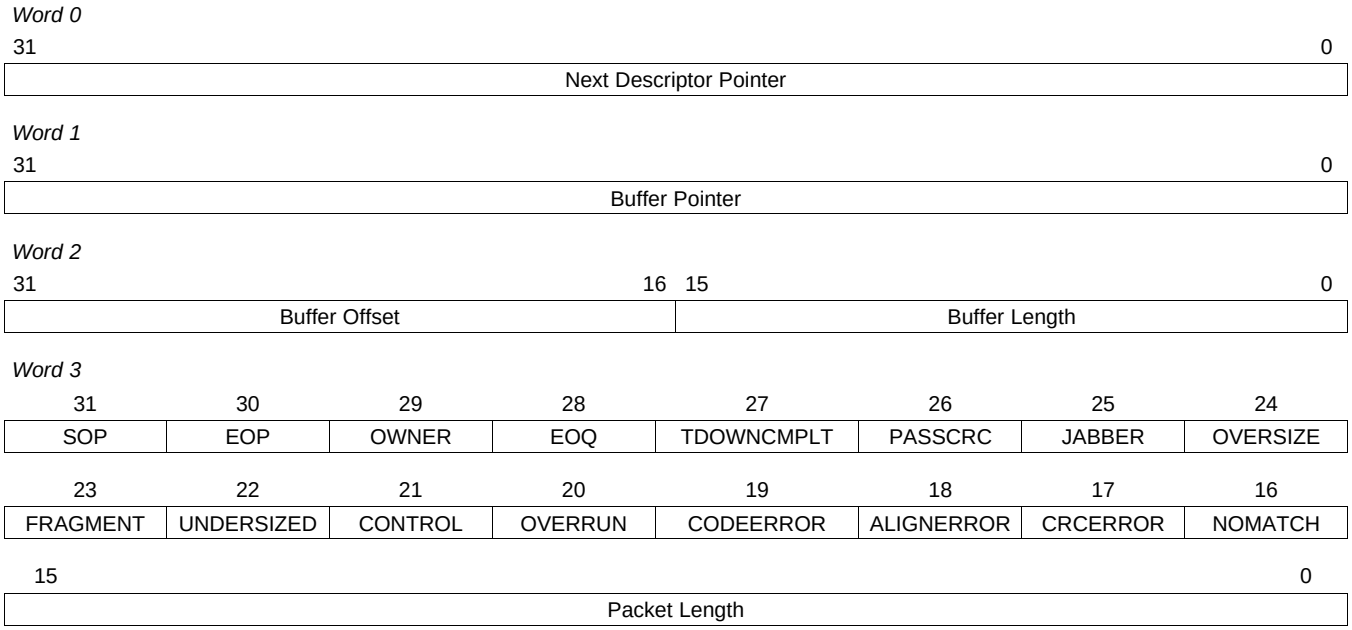
This flag is set by the software application in the SOP packet descriptor before it adds the descriptor to the transmit queue. Setting this bit indicates to the EMAC that the 4 byte Ethernet CRC is already present in the packet data, and that the EMAC should not generate its own version of the CRC.

When the CRC flag is cleared, the EMAC generates and appends the 4-byte CRC. The buffer length and packet length fields do not include the CRC bytes. When the CRC flag is set, the 4-byte CRC is supplied by the software application and is already appended to the end of the packet data. The buffer length and packet length fields include the CRC bytes, as they are part of the valid packet data. Note that this flag is valid on SOP descriptors only.

6.2.5.5 Receive Buffer Descriptor Format

A receive (RX) buffer descriptor (Figure 6-7) is a contiguous block of four 32-bit data words aligned on a 32-bit boundary that describes a packet or a packet fragment. Example 6-2 shows the receive buffer descriptor described by a C structure.

Figure 6-7. Receive Buffer Descriptor Format



6.2.5.5.1 Next Descriptor Pointer

This pointer points to the 32–bit word aligned memory address of the next buffer descriptor in the receive queue. This pointer is used to create a linked list of buffer descriptors. If the value of this pointer is zero, then the current buffer is the last buffer in the queue. The software application must set this value prior to adding the descriptor to the active receive list. This pointer is not altered by the EMAC.

The value of pNext should never be altered once the descriptor is in an active receive queue, unless its current value is NULL. If the pNext pointer is initially NULL, and more empty buffers can be added to the pool, the software application may alter this pointer to point to a newly appended descriptor. The EMAC will use the new pointer value and proceed to the next descriptor unless the pNext value has already been read. In this latter case, the receiver will halt the receive channel in question, and the software application may restart it at that time. The software can detect this case by checking for an end of queue (EOQ) condition flag on the updated packet descriptor when it is returned by the EMAC.

6.2.5.5.2 Buffer Pointer

The buffer pointer is the byte-aligned memory address of the memory buffer associated with the buffer descriptor. The software application must set this value prior to adding the descriptor to the active receive list. This pointer is not altered by the EMAC.

Example 6-2. Receive Buffer Descriptor in C Structure Format

```
/*
// EMAC Descriptor
//
// The following is the format of a single buffer descriptor
// on the EMAC.
*/
typedef struct _EMAC_Desc {
 struct _EMAC_Desc *pNext; /* Pointer to next descriptor in chain */
 Uint8 *pBuffer; /* Pointer to data buffer */
 Uint32 BufOffLen; /* Buffer Offset(MSW) and Length(LSW) */
 Uint32 PktFlgLen; /* Packet Flags(MSW) and Length(LSW) */
} EMAC_Desc;

/* Packet Flags */
#define EMAC_DSC_FLAG_SOP 0x80000000u
#define EMAC_DSC_FLAG_EOP 0x40000000u
#define EMAC_DSC_FLAG_OWNER 0x20000000u
#define EMAC_DSC_FLAG_EOQ 0x10000000u
#define EMAC_DSC_FLAG_TDOWNCMPLT 0x08000000u
#define EMAC_DSC_FLAG_PASSCRC 0x04000000u
#define EMAC_DSC_FLAG_JABBER 0x02000000u
#define EMAC_DSC_FLAG_OVERSIZE 0x01000000u
#define EMAC_DSC_FLAG_FRAGMENT 0x00800000u
#define EMAC_DSC_FLAG_UNDERSIZED 0x00400000u
#define EMAC_DSC_FLAG_CONTROL 0x00200000u
#define EMAC_DSC_FLAG_OVERRUN 0x00100000u
#define EMAC_DSC_FLAG_CODEERROR 0x00080000u
#define EMAC_DSC_FLAG_ALIGNERROR 0x00040000u
#define EMAC_DSC_FLAG_CRCERROR 0x00020000u
#define EMAC_DSC_FLAG_NOMATCH 0x00010000u
```

6.2.5.5.3 Buffer Offset

This 16-bit field must be initialized to zero by the software application before adding the descriptor to a receive queue.

Whether or not this field is updated depends on the setting of the RXBUFFEROFFSET register. When the offset register is set to a non-zero value, the received packet is written to the packet buffer at an offset given by the value of the register, and this value is also written to the buffer offset field of the descriptor.

When a packet is fragmented over multiple buffers because it does not fit in the first buffer supplied, the buffer offset only applies to the first buffer in the list, which is where the start of packet (SOP) flag is set in the corresponding buffer descriptor. In other words, the buffer offset field is only updated by the EMAC on SOP descriptors.

The range of legal values for the BUFFEROFFSET register is 0 to (Buffer Length – 1) for the smallest value of buffer length for all descriptors in the list.

6.2.5.5.4 Buffer Length

This 16-bit field is used for two purposes:

- Before the descriptor is first placed on the receive queue by the application software, the buffer length field is first initialized by the software to have the physical size of the empty data buffer pointed to by the buffer pointer field.
- After the empty buffer has been processed by the EMAC and filled with received data bytes, the buffer length field is updated by the EMAC to reflect the actual number of valid data bytes written to the buffer.

6.2.5.5.5 Packet Length

This 16-bit field specifies the number of data bytes in the entire packet. This value is initialized to zero by the software application for empty packet buffers. The value is filled in by the EMAC on the first buffer used for a given packet. This is signified by the EMAC setting a start of packet (SOP) flag. The packet length is set by the EMAC on all SOP buffer descriptors.

6.2.5.5.6 Start of Packet (SOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is the start of a new packet. In the case of a single fragment packet, both the SOP and end of packet (EOP) flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet has the EOP flag set. This flag is initially cleared by the software application before adding the descriptor to the receive queue. This bit is set by the EMAC on SOP descriptors.

6.2.5.5.7 End of Packet (EOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is last for a given packet. In the case of a single fragment packet, both the start of packet (SOP) and EOP flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet has the EOP flag set. This flag is initially cleared by the software application before adding the descriptor to the receive queue. This bit is set by the EMAC on EOP descriptors.

6.2.5.5.8 Ownership (OWNER) Flag

When set, this flag indicates that the descriptor is currently owned by the EMAC. This flag is set by the software application before adding the descriptor to the receive descriptor queue. This flag is cleared by the EMAC once it is finished with a given set of descriptors, associated with a received packet. The flag is updated by the EMAC on SOP descriptor only. So when the application identifies that the OWNER flag is cleared on an SOP descriptor, it may assume that all descriptors up to and including the first with the EOP flag set have been released by the EMAC. (Note that in the case of single buffer packets, the same descriptor will have both the SOP and EOP flags set.)

6.2.5.5.9 End of Queue (EOQ) Flag

When set, this flag indicates that the descriptor in question was the last descriptor in the receive queue for a given receive channel, and that the corresponding receiver channel has halted. This flag is initially cleared by the software application prior to adding the descriptor to the receive queue. This bit is set by the EMAC when the EMAC identifies that a descriptor is the last for a given packet received (also sets the EOP flag), and there are no more descriptors in the receive list (next descriptor pointer is NULL).

The software application can use this bit to detect when the EMAC receiver for the corresponding channel has halted. This is useful when the application appends additional free buffer descriptors to an active receive queue. Note that this flag is valid on EOP descriptors only.

6.2.5.5.10 Teardown Complete (TDOWNCMPLT) Flag

This flag is used when a receive queue is being torn down, or aborted, instead of being filled with received data. This would happen under device driver reset or shutdown conditions. The EMAC sets this bit in the descriptor of the first free buffer when the tear down occurs. No additional queue processing is performed.

6.2.5.5.11 Pass CRC (PASSCRC) Flag

This flag is set by the EMAC in the SOP buffer descriptor if the received packet includes the 4-byte CRC. This flag should be cleared by the software application before submitting the descriptor to the receive queue.

6.2.5.5.12 Jabber Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is a jabber frame and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE. Jabber frames are frames that exceed the RXMAXLEN in length, and have CRC, code, or alignment errors.

6.2.5.5.13 Oversize Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is an oversized frame and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

6.2.5.5.14 Fragment Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is only a packet fragment and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

6.2.5.5.15 Undersized Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is undersized and was not discarded because the RXCSFEN bit was set in the RXMBPENABLE.

6.2.5.5.16 Control Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is an EMAC control frame and was not discarded because the RXCMFEN bit was set in the RXMBPENABLE.

6.2.5.5.17 Overrun Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet was aborted due to a receive overrun.

6.2.5.5.18 Code Error (CODEERROR) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet contained a code error and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

6.2.5.5.19 Alignment Error (ALIGNERROR) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet contained an alignment error and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

6.2.5.5.20 CRC Error (CRCERROR) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet contained a CRC error and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

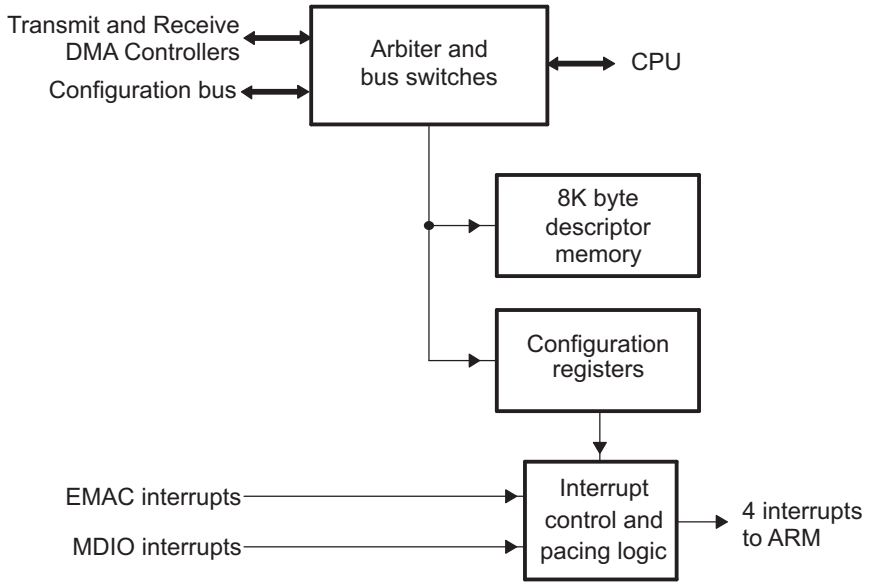
6.2.5.5.21 No Match (NOMATCH) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet did not pass any of the EMAC's address match criteria and was not discarded because the RXCAFEN bit was set in the RXMBPENABLE. Although the packet is a valid Ethernet data packet, it was only received because the EMAC is in promiscuous mode.

6.2.6 EMAC Control Module

The basic functions of the EMAC control module (Figure 6-8) are to interface the EMAC and MDIO modules to the rest of the system, and to provide for a local memory space to hold EMAC packet buffer descriptors. Local memory is used to help avoid contention to device memory spaces. Other functions include the bus arbiter, and interrupt control and pacing logic control.

Figure 6-8. EMAC Control Module Block Diagram



6.2.6.1 Internal Memory

The EMAC control module includes 8K bytes of internal memory. The internal memory block is essential for allowing the EMAC to operate more independently of the CPU. It also prevents memory underflow conditions when the EMAC issues read or write requests to descriptor memory. (Memory accesses to read or write the actual Ethernet packet data are protected by the EMAC's internal FIFOs).

A descriptor is a 16-byte memory structure that holds information about a single Ethernet packet buffer, which may contain a full or partial Ethernet packet. Thus with the 8K memory block provided for descriptor storage, the EMAC module can send and received up to a combined 512 packets before it needs to be serviced by application or driver software.

6.2.6.2 Bus Arbiter

The EMAC control module bus arbiter operates transparently to the rest of the system. It is used:

- To arbitrate between the CPU and EMAC buses for access to internal descriptor memory.
- To arbitrate between internal EMAC buses for access to system memory.

6.2.6.3 Interrupt Control

The EMAC control module combines multiple interrupt conditions generated by the EMAC and MDIO modules into four separate interrupt signals (Table 6-5) that are mapped to a CPU interrupt via the CPU interrupt controller. The four separate sources of interrupt can be individually enabled for each channel by the CMRXTHRESHINTEN, CMRXINTEN, CMTXINTEN, and CMMISCINTEN registers.

Table 6-5. EMAC Control Module Interrupts

ARM Event	Acronym	Source
24	MAC_RXTH	EMAC Receive Threshold
25	MAC_RX	EMAC Receive
26	MAC_TX	EMAC Transmit
27	MAC_MISC	EMAC Miscellaneous

6.2.6.3.1 Transmit Pulse Interrupt

The EMAC control module receives the eight individual transmit interrupts originating from the EMAC module, one for each of the eight channels, and combines them into a single transmit pulse interrupt to the CPU. This transmit pulse interrupt is paced, as described in Section 6.2.6.4. The eight individual transmit pending interrupt(s) are selected at the EMAC control module level, by setting one or more bits in the EMAC control module transmit interrupt enable register (CMTXINTEN). The masked interrupt status can be read in the EMAC control module transmit interrupt status register (CMTXINTSTAT).

Upon reception of a transmit pulse interrupt, the ISR performs the following:

1. Read CMTXINTSTAT to determine which channel(s) caused the interrupt.
2. Process received packets for the interrupting channel(s).
3. Write the appropriate CPGMAC transmit channel *n* completion pointer register(s) (TX*n*CP) with the address of the last buffer descriptor of the last packet processed by the application software.
4. Write the MAC end of interrupt vector register (MACEOIVECTOR) in the EMAC module with a value of 2h to signal the end of the transmit interrupt processing.

6.2.6.3.2 Receive Pulse Interrupt

The EMAC control module receives the eight individual receive interrupts originating from the EMAC module, one for each of the eight channels, and combines them into a single receive pulse interrupt to the CPU. This receive pulse interrupt is paced, as described in Section 6.2.6.4. The eight individual receive pending interrupt(s) are selected at the EMAC control module level, by setting one or more bits in the EMAC control module receive interrupt enable register (CMRXINTEN). The masked interrupt status can be read in the EMAC control module receive interrupt status register (CMRXINTSTAT).

Upon reception of a receive pulse interrupt, the ISR performs the following:

1. Read CMRXINTSTAT to determine which channel(s) caused the interrupt.
2. Process received packets for the interrupting channel(s).
3. Write the appropriate CPGMAC receive channel *n* completion pointer register(s) (RX*n*CP) with the address of the last buffer descriptor of the last packet processed by the application software.
4. Write the MAC end of interrupt vector register (MACEOIVECTOR) in the EMAC module with a value of 1h to signal the end of the receive interrupt processing.

6.2.6.3.3 Receive Threshold Pulse Interrupt

The EMAC control module receives the eight individual receive threshold interrupts originating from the EMAC module, one for each of the eight channels, and combines them into a single receive threshold pulse interrupt to the CPU. This receive threshold pulse interrupt is not paced. The eight individual receive threshold pending interrupt(s) are selected at the EMAC control module level, by setting one or more bits in the EMAC control module receive threshold interrupt enable register (CMRXTHRESHINTEN). The masked interrupt status can be read in the EMAC control module receive threshold interrupt status register (CMRXTHRESHINTSTAT).

Upon reception of a receive threshold pulse interrupt, the ISR performs the following:

1. Read CMRXTHRESHINTSTAT to determine which channel(s) caused the interrupt.
2. Process received packets in order to add more buffers to any channel that is below the threshold value.
3. Write the appropriate CPGMAC receive channel *n* completion pointer register(s) (RX*n*CP) with the address of the last buffer descriptor of the last packet processed by the application software.
4. Write the MAC end of interrupt vector register (MACEOIVECTOR) in the EMAC module with a value of 0 to signal the end of the receive threshold interrupt processing.

6.2.6.3.4 Miscellaneous Pulse Interrupt

The EMAC control module receives the STATPEND and HOSTPEND interrupts from the EMAC module and the MDIO_LINKINT and MDIO_USERINT interrupts from the MDIO module. The EMAC control module combines these four interrupts into a single miscellaneous pulse interrupt to the CPU. This miscellaneous interrupt is not paced. The four individual interrupts are selected at the EMAC control module level, by setting one or more bits in the EMAC control module miscellaneous interrupt enable register (CMMISCINTEN). The masked interrupt status can be read in the EMAC control module miscellaneous interrupt status register (CMMISCINTSTAT).

Upon reception of a miscellaneous pulse interrupt, the ISR performs the following:

1. Read CMMISCINTSTAT to determine which of the four condition(s) caused the interrupt.
2. Process those interrupts accordingly.
3. Write the MAC end of interrupt vector register (MACEOIVECTOR) in the EMAC module with a value of 3h to signal the end of the miscellaneous interrupt processing.

6.2.6.4 Interrupt Pacing

The receive and transmit pulse interrupts can be paced. The receive threshold and miscellaneous interrupts can not be paced. The interrupt pacing feature limits the number of interrupts to the CPU during a given period of time. For heavily loaded systems in which interrupts can occur at a very high rate, the performance benefit is significant due to minimizing the overhead associated with servicing each interrupt. The receive and transmit pulse interrupts contain a separate interrupt pacing sub-blocks. Each sub-block is disabled by default allowing the selected interrupt inputs to pass-through unaffected.

The interrupt pacing module counts the number of interrupts that occur over a 1 ms interval of time. At the end of each 1 ms interval, the current number of interrupts is compared with a target number of interrupts (specified by the associated EMAC control module interrupts per millisecond registers, CMTXINTMAX and CMRXINTMAX). Based on the results of the comparison, the length of time during which interrupts are blocked is dynamically adjusted. The 1 ms interval is derived from a 4 µs pulse that is created from a prescale counter whose value is set in the INTPRESCALE field of the EMAC control module interrupt control register (CMINTCTRL). This INTPRESCALE value should be written with the number of peripheral clock periods in 4 µs. The pacing timer determines the interval during which interrupts are blocked and decrements every 4 µs. It is reloaded each time a zero count is reached. The value loaded into the pacing timer is calculated by hardware every 1 ms, according to the dynamic algorithm in the hardware.

If the rate of transmit pulse interrupt inputs is much less than the target transmit pulse interrupt rate specified in CMTXINTMAX, then the interrupts are not blocked to the CPU. If the transmit pulse interrupt rate is greater than the specified target rate in CMTXINTMAX, the interrupt is paced at the rate specified in this register, which should be written with a value between 2 and 63 inclusive, indicating the target number of interrupts per 1 ms going to the CPU. Similarly, the number of receive interrupt pulses to the CPU is also separately controlled.

6.2.7 MDIO Module

The MDIO module is used to manage up to 32 physical layer (PHY) devices connected to the Ethernet Media Access Controller (EMAC). The device supports a single PHY being connected to the EMAC at any given time. The MDIO module is designed to allow almost transparent operation of the MDIO interface with little maintenance from the CPU.

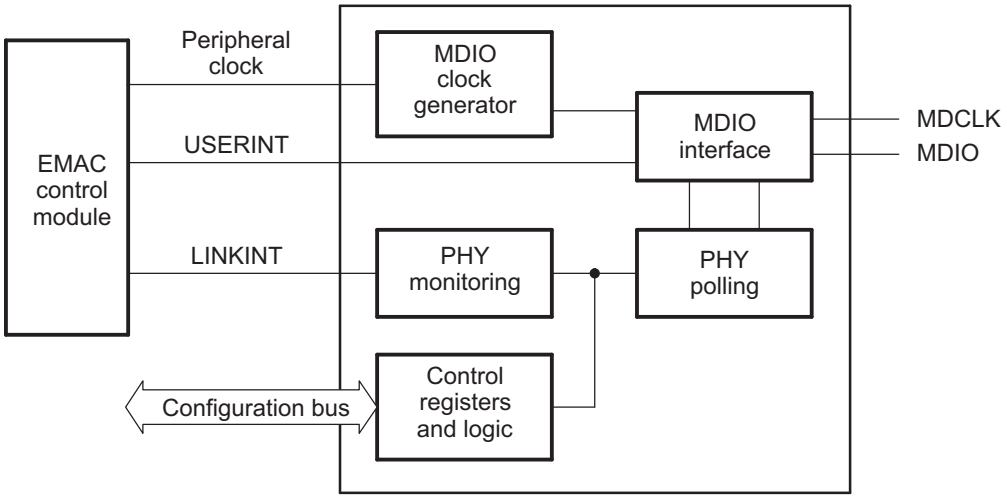
The MDIO module continuously polls 32 MDIO addresses in order to enumerate all PHY devices in the system. Once a PHY device has been detected, the MDIO module reads the MDIO PHY link status register (LINK) to monitor the PHY link state. Link change events are stored in the MDIO module, which can interrupt the CPU. This storing of the events allows the CPU to poll the link status of the PHY device without continuously performing MDIO module accesses. However, when the CPU must access the MDIO module for configuration and negotiation, the MDIO module performs the MDIO read or write operation independent of the CPU. This independent operation allows the processor to poll for completion or interrupt the CPU once the operation has completed.

6.2.7.1 MDIO Module Components

The MDIO module (Figure 6-9) interfaces to the PHY components through two MDIO pins (MDCLK and MDIO), and to the CPU through the EMAC control module and the configuration bus. The MDIO module consists of the following logical components:

- MDIO clock generator
- Global PHY detection and link state monitoring
- Active PHY monitoring
- PHY register user access

Figure 6-9. MDIO Module Block Diagram



6.2.7.1.1 MDIO Clock Generator

The MDIO clock generator controls the MDIO clock based on a divide-down of the peripheral clock (SYSCLK5) in the EMAC control module. The MDIO clock is specified to run up to 2.5 MHz, although typical operation would be 1.0 MHz. Since the peripheral clock frequency is variable (SYSCLK5), the application software or driver controls the divide-down amount.

6.2.7.1.2 Global PHY Detection and Link State Monitoring

The MDIO module continuously polls all 32 MDIO addresses in order to enumerate the PHY devices in the system. The module tracks whether or not a PHY on a particular address has responded, and whether or not the PHY currently has a link. Using this information allows the software application to quickly determine which MDIO address the PHY is using.

6.2.7.1.3 Active PHY Monitoring

Once a PHY candidate has been selected for use, the MDIO module transparently monitors its link state by reading the MDIO PHY link status register (LINK). Link change events are stored on the MDIO device and can optionally interrupt the CPU. This allows the system to poll the link status of the PHY device without continuously performing costly MDIO accesses.

6.2.7.1.4 PHY Register User Access

When the CPU must access MDIO for configuration and negotiation, the PHY access module performs the actual MDIO read or write operation independent of the CPU. This allows the CPU to poll for completion or receive an interrupt when the read or write operation has been performed. The user access registers USERACCESS*n* allows the software to submit the access requests for the PHY connected to the device.

6.2.7.2 MDIO Module Operational Overview

The MDIO module implements the 802.3 serial management interface to interrogate and control an Ethernet PHY, using a shared two-wired bus. It separately performs autodetection and records the current link status of up to 32 PHYs, polling all 32 MDIO addresses.

Application software uses the MDIO module to configure the autonegotiation parameters of the PHY attached to the EMAC, retrieve the negotiation results, and configure required parameters in the EMAC.

In this device, the Ethernet PHY attached to the system can be directly controlled and queried. The Media Independent Interface (MII) address of this PHY device is specified in one of the PHYADRMON bits in the MDIO user PHY select register (USERPHYSEL*n*). The MDIO module can be programmed to trigger a CPU interrupt on a PHY link change event, by setting the LINKINTENB bit in USERPHYSEL*n*. Reads and writes to registers in this PHY device are performed using the MDIO user access register (USERACCESS*n*).

The MDIO module powers-up in an idle state until specifically enabled by setting the ENABLE bit in the MDIO control register (CONTROL). At this time, the MDIO clock divider and preamble mode selection are also configured. The MDIO preamble is enabled by default, but can be disabled when the connected PHY does not require it. Once the MDIO module is enabled, the MDIO interface state machine continuously polls the PHY link status (by reading the generic status register) of all possible 32 PHY addresses and records the results in the MDIO PHY alive status register (ALIVE) and MDIO PHY link status register (LINK). The corresponding bit for the connected PHY (0-31) is set in ALIVE, if the PHY responded to the read request. The corresponding bit is set in LINK, if the PHY responded and also is currently linked. In addition, any PHY register read transactions initiated by the application software using USERACCESS*n* causes ALIVE to be updated.

The USERPHYSEL*n* is used to track the link status of the connected PHY address. A change in the link status of the PHY being monitored sets the appropriate bit in the MDIO link status change interrupt registers (LINKINTRAW and LINKINTMASKED), if enabled by the LINKINTENB bit in USERPHYSEL*n*.

While the MDIO module is enabled, the host issues a read or write transaction over the MII management interface using the DATA, PHYADR, REGADR, and WRITE bits in USERACCESS*n*. When the application sets the GO bit in USERACCESS*n*, the MDIO module begins the transaction without any further intervention from the CPU. Upon completion, the MDIO module clears the GO bit and sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (USERINTRAW) corresponding to USERACCESS*n* used. The corresponding USERINTMASKED bit (0 or 1) in the MDIO user command complete interrupt register (USERINTMASKED) may also be set, depending on the mask setting configured in the MDIO user command complete interrupt mask set register (USERINTMASKSET) and the MDIO user interrupt mask clear register (USERINTMASKCLEAR).

A round-robin arbitration scheme is used to schedule transactions that may be queued using both USERACCESS0 and USERACCESS1. The application software must check the status of the GO bit in USERACCESS*n* before initiating a new transaction, to ensure that the previous transaction has completed. The application software can use the ACK bit in USERACCESS*n* to determine the status of a read transaction.

6.2.7.2.1 Initializing the MDIO Module

The following steps are performed by the application software or device driver to initialize the MDIO device:

1. Configure the PREAMBLE and CLKDIV bits in the MDIO control register (CONTROL).
2. Enable the MDIO module by setting the ENABLE bit in CONTROL.
3. The MDIO PHY alive status register (ALIVE) can be read in polling fashion until a PHY connected to the system responded, and the MDIO PHY link status register (LINK) can determine whether this PHY already has a link.
4. Setup the appropriate PHY addresses in the MDIO user PHY select register (USERPHYSEL*n*), and set the LINKINTENB bit to enable a link change event interrupt if desirable.
5. If an interrupt on general MDIO register access is desired, set the corresponding bit in the MDIO user command complete interrupt mask set register (USERINTMASKSET) to use the MDIO user access register (USERACCESS*n*). Since only one PHY is used in this device, the application software can use one USERACCESS*n* to trigger a completion interrupt; the other USERACCESS*n* is not setup.

6.2.7.2.2 Writing Data To a PHY Register

The MDIO module includes a user access register (USERACCESS*n*) to directly access a specified PHY device. To write a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (USERACCESS*n*) is cleared.
2. Write to the GO, WRITE, REGADR, PHYADR, and DATA bits in USERACCESS*n* corresponding to the PHY and PHY register you want to write.
3. The write operation to the PHY is scheduled and completed by the MDIO module. Completion of the write operation can be determined by polling the GO bit in USERACCESS*n* for a 0.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (USERINTRAW) corresponding to USERACCESS*n* used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (USERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (USERINTMASKED) and an interrupt is triggered on the CPU.

6.2.7.2.3 Reading Data From a PHY Register

The MDIO module includes a user access register (USERACCESS*n*) to directly access a specified PHY device. To read a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (USERACCESS*n*) is cleared.
2. Write to the GO, REGADR, and PHYADR bits in USERACCESS*n* corresponding to the PHY and PHY register you want to read.

3. The read data value is available in the DATA bits in USERACCESS*n* after the module completes the read operation on the serial bus. Completion of the read operation can be determined by polling the GO and ACK bits in USERACCESS n . Once the GO bit has cleared, the ACK bit is set on a successful read.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (USERINTRAW) corresponding to USERACCESS*n* used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (USERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (USERINTMASKED) and an interrupt is triggered on the CPU.

6.2.7.2.4 Example of MDIO Register Access Code

The MDIO module uses the MDIO user access register (USERACCESS*n*) to access the PHY control registers. Software functions that implement the access process may simply be the following four macros:

- PHYREG_read(regadr, phyadr) — Start the process of reading a PHY register
- PHYREG_write(regadr, phyadr, data) — Start the process of writing a PHY register
- PHYREG_wait( ) — Synchronize operation (make sure read/write is idle)
- PHYREG_waitResults(results) — Wait for read to complete and return data read

Note that it is not necessary to wait after a write operation, as long as the status is checked before every operation to make sure the MDIO hardware is idle. An alternative approach is to call PHYREG_wait() after every write, and PHYREG_waitResults( ) after every read, then the hardware can be assumed to be idle when starting a new operation.

The implementation of these macros using the chip support library (CSL) is shown in Example 6-3 (USERACCESS0 is assumed).

Note that this implementation does not check the ACK bit in USERACCESS*n* on PHY register reads (does not follow the procedure outlined in Section 6.2.7.2.3). Since the MDIO PHY alive status register (ALIVE) is used to initially select a PHY, it is assumed that the PHY is acknowledging read operations. It is possible that a PHY could become inactive at a future point in time. An example of this would be a PHY that can have its MDIO addresses changed while the system is running. It is not very likely, but this condition can be tested by periodically checking the PHY state in ALIVE.

Example 6-3. MDIO Register Access Macros

```
#define PHYREG_read(regadr, phyadr)
    MDIO_REGS->USERACCESS0 =
        CSL_FMK(MDIO_USERACCESS0_GO,1u)                        | /
        CSL_FMK(MDIO_USERACCESS0_REGADR,regadr)                | /
        CSL_FMK(MDIO_USERACCESS0_PHYADR,phyadr)
#define PHYREG_write(regadr, phyadr, data)
    MDIO_REGS->USERACCESS0 =
        CSL_FMK(MDIO_USERACCESS0_GO,1u)                        | /
        CSL_FMK(MDIO_USERACCESS0_WRITE,1)                      | /
        CSL_FMK(MDIO_USERACCESS0_REGADR,regadr)                | /
        CSL_FMK(MDIO_USERACCESS0_PHYADR,phyadr)                | /
        CSL_FMK(MDIO_USERACCESS0_DATA, data)
#define PHYREG_wait()
    while( CSL_FEXT(MDIO_REGS->USERACCESS0,MDIO_USERACCESS0_GO) )
#define PHYREG_waitResults( results ) {
    while( CSL_FEXT(MDIO_REGS->USERACCESS0,MDIO_USERACCESS0_GO) );
results = CSL_FEXT(MDIO_REGS->USERACCESS0, MDIO_USERACCESS0_DATA); }
```

6.2.8 EMAC Module

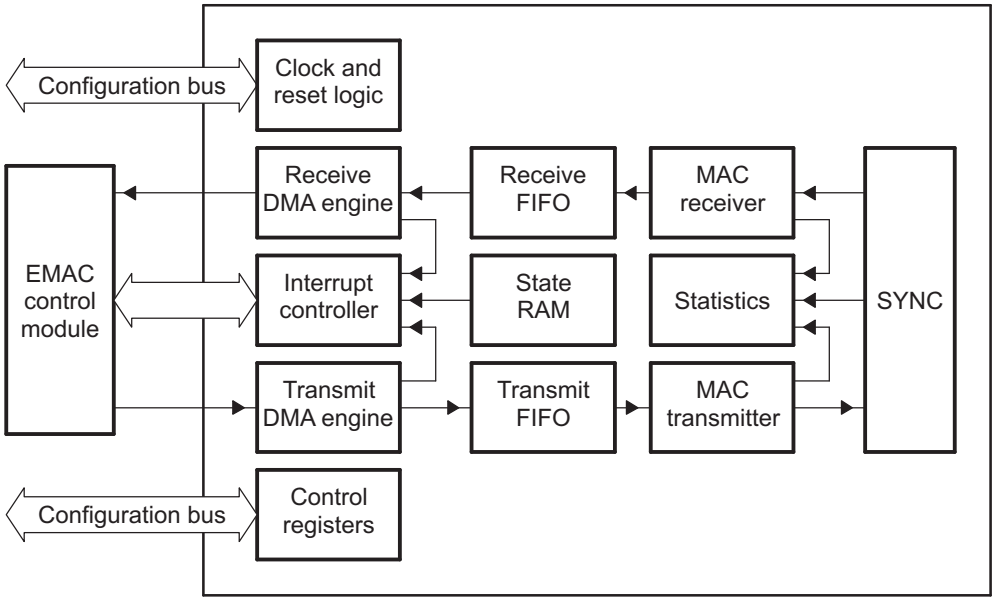
This section discusses the architecture and basic function of the EMAC module.

6.2.8.1 EMAC Module Components

The EMAC module (Figure 6-10) interfaces to the outside world through the Media Independent Interface (MII) and interfaces to the system core through the EMAC control module. The EMAC consists of the following logical components:

- The receive path includes: receive DMA engine, receive FIFO, and MAC receiver
- The transmit path includes: transmit DMA engine, transmit FIFO, and MAC transmitter
- Statistics logic
- State RAM
- Interrupt controller
- Control registers and logic
- Clock and reset logic

Figure 6-10. EMAC Module Block Diagram



6.2.8.1.1 Receive DMA Engine

The receive DMA engine is the interface between the receive FIFO and the system core. It interfaces to the CPU through the bus arbiter in the EMAC control module. This DMA engine is totally independent of the device DMA.

6.2.8.1.2 Receive FIFO

The receive FIFO consists of 68 cells of 64 bytes each and associated control logic. The FIFO buffers receive data in preparation for writing into packet buffers in device memory, and also enable receive FIFO flow control.

6.2.8.1.3 MAC Receiver

The MAC receiver detects and processes incoming network frames, de-frames them, and puts them into the receive FIFO. The MAC receiver also detects errors and passes statistics to the statistics RAM.

6.2.8.1.4 Receive Address

This sub-module performs address matching and address filtering based on the incoming packet's destination address. It contains a 32-by-53 bit two-port RAM, in which up to 32 addresses can be stored to be either matched or filtered by the EMAC. The RAM may contain multicast packet addresses, but the associated channel must have the unicast enable bit set, even though it is a multicast address. The unicast enable bits are used with multicast addresses in the receive address RAM (not the multicast hash enable bits). Therefore, hash matches can be disabled, but specific multicast addresses can be matched (or filtered) in the RAM. If a multicast packet hash matches, the packet may still be filtered in the RAM. Each packet can be sent to only a single channel.

6.2.8.1.5 Transmit DMA Engine

The transmit DMA engine is the interface between the transmit FIFO and the CPU. It interfaces to the CPU through the bus arbiter in the EMAC control module.

6.2.8.1.6 Transmit FIFO

The transmit FIFO consists of 24 cells of 64 bytes each and associated control logic. This enables a packet of 1518 bytes (standard Ethernet packet size) to be sent without the possibility of underrun. The FIFO buffers data in preparation for transmission.

6.2.8.1.7 MAC Transmitter

The MAC transmitter formats frame data from the transmit FIFO and transmits the data using the CSMA/CD access protocol. The frame CRC can be automatically appended, if required. The MAC transmitter also detects transmission errors and passes statistics to the statistics registers.

6.2.8.1.8 Statistics Logic

The Ethernet statistics are counted and stored in the statistics logic RAM. This statistics RAM keeps track of 36 different Ethernet packet statistics.

6.2.8.1.9 State RAM

State RAM contains the head descriptor pointers and completion pointers registers for both transmit and receive channels.

6.2.8.1.10 EMAC Interrupt Controller

The interrupt controller contains the interrupt related registers and logic. The 18 raw EMAC interrupts are input to this submodule and masked module interrupts are output.

6.2.8.1.11 Control Registers and Logic

The EMAC is controlled by a set of memory-mapped registers. The control logic also signals transmit, receive, and status related interrupts to the CPU through the EMAC control module.

6.2.8.1.12 Clock and Reset Logic

The clock and reset submodule generates all the EMAC clocks and resets. For more details on reset capabilities, see Section 14.2.9.1.

6.2.8.2 EMAC Module Operational Overview

After reset, initialization, and configuration, the application software running on the host may initiate transmit operations. Transmit operations are initiated by host writes to the appropriate transmit channel head descriptor pointer contained in the state RAM block. The transmit DMA controller then fetches the first packet in the packet chain from memory. The DMA controller writes the packet into the transmit FIFO in bursts of 64-byte cells. When the threshold number of cells, configurable using the TXCELLTHRESH bit in the FIFO control register (FIFOCONTROL), have been written to the transmit FIFO, or a complete packet, whichever is smaller, the MAC transmitter then initiates the packet transmission. The SYNC block transmits the packet over the MII interfaces in accordance with the 802.3 protocol. Transmit statistics are counted by the statistics block.

Receive operations are initiated by host writes to the appropriate receive channel head descriptor pointer after host initialization and configuration. The SYNC submodule receives packets and strips off the Ethernet related protocol. The packet data is input to the MAC receiver, which checks for address match and processes errors. Accepted packets are then written to the receive FIFO in bursts of 64-byte cells. The receive DMA controller then writes the packet data to memory. Receive statistics are counted by the statistics block.

The EMAC module operates independently of the CPU. It is configured and controlled by its register set mapped into device memory. Information about data packets is communicated by use of 16-byte descriptors that are placed in an 8K-byte block of RAM in the EMAC control module.

For transmit operations, each 16-byte descriptor describes a packet or packet fragment in the system's internal or external memory. For receive operations, each 16-byte descriptor represents a free packet buffer or buffer fragment. On both transmit and receive, an Ethernet packet is allowed to span one or more memory fragments, represented by one 16-byte descriptor per fragment. In typical operation, there is only one descriptor per receive buffer, but transmit packets may be fragmented, depending on the software architecture.

An interrupt is issued to the CPU whenever a transmit or receive operation has completed. However, it is not necessary for the CPU to service the interrupt while there are additional resources available. In other words, the EMAC continues to receive Ethernet packets until its receive descriptor list has been exhausted. On transmit operations, the transmit descriptors need only be serviced to recover their associated memory buffer. Thus, it is possible to delay servicing of the EMAC interrupt if there are real-time tasks to perform.

Eight channels are supplied for both transmit and receive operations. On transmit, the eight channels represent eight independent transmit queues. The EMAC can be configured to treat these channels as an equal priority "round-robin" queue or as a set of eight fixed-priority queues. On receive, the eight channels represent eight independent receive queues with packet classification. Packets are classified based on the destination MAC address. Each of the eight channels is assigned its own MAC address, enabling the EMAC module to act like eight virtual MAC adapters. Also, specific types of frames can be sent to specific channels. For example, multicast, broadcast, or other (promiscuous, error, etc.), can each be received on a specific receive channel queue.

The EMAC keeps track of 36 different statistics, plus keeps the status of each individual packet in its corresponding packet descriptor.

6.2.9 Media Independent Interface (MII)

The following sections discuss the operation of the Media Independent Interface (MII) in 10 Mbps and 100 Mbps mode. An IEEE 802.3 compliant Ethernet MAC controls the interface.

6.2.9.1 Data Reception

6.2.9.1.1 Receive Control

Data received from the PHY is interpreted and output to the EMAC receive FIFO. Interpretation involves detection and removal of the preamble and start-of-frame delimiter, extraction of the address and frame length, data handling, error checking and reporting, cyclic redundancy checking (CRC), and statistics control signal generation. Address detection and frame filtering is performed outside the MII interface.

6.2.9.1.2 Receive Inter-Frame Interval

The 802.3 standard requires an interpacket gap (IPG), which is 24 MII clocks (96 bit times). However, the EMAC can tolerate a reduced IPG (2 MII clocks or 8 bit times) with a correct preamble and start frame delimiter. This interval between frames must comprise (in the following order):

1. An Interpacket Gap (IPG).
2. A 7-byte preamble (all bytes 55h).
3. A 1-byte start of frame delimiter (5DH).

6.2.9.1.3 Receive Flow Control

When enabled and triggered, receive flow control is initiated to limit the EMAC from further frame reception. Two forms of receive flow control are implemented on the device:

- Receive buffer flow control
- Receive FIFO flow control

When enabled and triggered, receive buffer flow control prevents further frame reception based on the number of free buffers available. Receive buffer flow control issues flow control collisions in half-duplex mode and IEEE 802.3X pause frames for full-duplex mode. Receive buffer flow control is triggered when the number of free buffers in any enabled receive channel free buffer count register (RX*n* FREEBUFFER) is less than or equal to the receive channel flow control threshold register (RX*n* FLOWTHRESH) value. Receive flow control is independent of receive QOS, except that both use the free buffer values.

When enabled and triggered, receive FIFO flow control prevents further frame reception based on the number of cells currently in the receive FIFO. Receive FIFO flow control may be enabled only in full-duplex mode (FULLDUPLEX bit is set in the in the MAC control register, MACCONTROL). Receive flow control prevents reception of frames on the port until all of the triggering conditions clear, at which time frames may again be received by the port.

Receive FIFO flow control is triggered when the occupancy of the FIFO is greater than or equal to the RXFIFOFLOWTHRESH value in the FIFO control register (FIFOCONTROL). The RXFIFOFLOWTHRESH value must be greater than or equal to 1h and less than or equal to 42h (decimal 66). The RXFIFOFLOWTHRESH reset value is 2h.

Receive flow control is enabled by the RXBUFFERFLOWEN bit and the RXFIFOFLOWEN bit in MACCONTROL. The FULLDUPLEX bit in MACCONTROL configures the EMAC for collision or IEEE 802.3X flow control.

6.2.9.1.3.1 Collision-Based Receive Buffer Flow Control

Collision-based receive buffer flow control provides a means of preventing frame reception when the EMAC is operating in half-duplex mode (the FULLDUPLEX bit is cleared in MACCONTROL). When receive flow control is enabled and triggered, the EMAC generates collisions for received frames. The jam sequence transmitted is the 12-byte sequence C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3h. The jam sequence begins no later than approximately as the source address starts to be received. Note that these forced collisions are not limited to a maximum of 16 consecutive collisions, and are independent of the normal back-off algorithm.

Receive flow control does not depend on the value of the incoming frame destination address. A collision is generated for any incoming packet, regardless of the destination address, if any EMAC enabled channel's free buffer register value is less than or equal to the channel's flow threshold value.

6.2.9.1.3.2 IEEE 802.3x-Based Receive Buffer Flow Control

IEEE 802.3x-based receive buffer flow control provides a means of preventing frame reception when the EMAC is operating in full-duplex mode (the FULLDUPLEX bit is set in MACCONTROL). When receive flow control is enabled and triggered, the EMAC transmits a pause frame to request that the sending station stop transmitting for the period indicated within the transmitted pause frame.

The EMAC transmits a pause frame to the reserved multicast address at the first available opportunity (immediately if currently idle or following the completion of the frame currently being transmitted). The pause frame contains the maximum possible value for the pause time (FFFFh). The EMAC counts the receive pause frame time (decrements FF00h to 0) and retransmits an outgoing pause frame, if the count reaches 0. When the flow control request is removed, the EMAC transmits a pause frame with a zero pause time to cancel the pause request.

Note that transmitted pause frames are only a request to the other end station to stop transmitting. Frames that are received during the pause interval are received normally (provided the receive FIFO is not full).

Pause frames are transmitted if enabled and triggered, regardless of whether or not the EMAC is observing the pause time period from an incoming pause frame.

The EMAC transmits pause frames as described below:

- The 48-bit reserved multicast destination address 01.80.C2.00.00.01h.
- The 48-bit source address (set using the MACSRCADDRLO and MACSRCADDRHI registers).
- The 16-bit length/type field containing the value 88.08h.
- The 16-bit pause opcode equal to 00.01h.
- The 16-bit pause time value of FF.FFh. A pause-quantum is 512 bit-times. Pause frames sent to cancel a pause request have a pause time value of 00.00h.
- Zero padding to 64-byte data length (EMAC transmits only 64-byte pause frames).
- The 32-bit frame-check sequence (CRC word).

All quantities are hexadecimal and are transmitted most-significant byte first. The least-significant bit (LSB) is transferred first in each byte.

If the RXBUFFERFLOWEN bit in MACCONTROL is cleared to 0 while the pause time is nonzero, then the pause time is cleared to 0 and a zero count pause frame is sent.

6.2.9.2 Data Transmission

The EMAC passes data to the PHY from the transmit FIFO (when enabled). Data is synchronized to the transmit clock rate. Transmission begins when there are TXCELLTHRESH cells of 64 bytes each, or a complete packet, in the FIFO.

6.2.9.2.1 Transmit Control

A jam sequence is output if a collision is detected on a transmit packet. If the collision was late (after the first 64 bytes have been transmitted), the collision is ignored. If the collision is not late, the controller will back off before retrying the frame transmission. When operating in full-duplex mode, the carrier sense (MCRS) and collision-sensing (MCOL) modes are disabled.

6.2.9.2.2 CRC Insertion

If the SOP buffer descriptor PASSCRC flag is cleared, the EMAC generates and appends a 32-bit Ethernet CRC onto the transmitted data. For the EMAC-generated CRC case, a CRC (or placeholder) at the end of the data is allowed but not required. The buffer byte count value should not include the CRC bytes, if they are present.

If the SOP buffer descriptor PASSCRC flag is set, then the last four bytes of the transmit data are transmitted as the frame CRC. The four CRC data bytes should be the last four bytes of the frame and should be included in the buffer byte count value. The MAC performs no error checking on the outgoing CRC.

6.2.9.2.3 Adaptive Performance Optimization (APO)

The EMAC incorporates adaptive performance optimization (APO) logic that may be enabled by setting the TXPACE bit in the MAC control register (MACCONTROL). Transmission pacing to enhance performance is enabled when the TXPACE bit is set. Adaptive performance pacing introduces delays into the normal transmission of frames, delaying transmission attempts between stations, reducing the probability of collisions occurring during heavy traffic (as indicated by frame deferrals and collisions), thereby, increasing the chance of successful transmission.

When a frame is deferred, suffers a single collision, multiple collisions, or excessive collisions, the pacing counter is loaded with an initial value of 31. When a frame is transmitted successfully (without experiencing a deferral, single collision, multiple collision, or excessive collision), the pacing counter is decremented by 1, down to 0.

With pacing enabled, a new frame is permitted to immediately (after one interpacket gap) attempt transmission only if the pacing counter is 0. If the pacing counter is nonzero, the frame is delayed by the pacing delay of approximately four interpacket gap (IPG) delays. APO only affects the IPG preceding the first attempt at transmitting a frame; APO does not affect the back-off algorithm for retransmitted frames.

6.2.9.2.4 Interpacket-Gap (IPG) Enforcement

The measurement reference for the IPG of 96 bit times is changed depending on frame traffic conditions. If a frame is successfully transmitted without collision and MCRS is deasserted within approximately 48 bit times of MTXEN being deasserted, then 96 bit times is measured from MTXEN. If the frame suffered a collision or MCRS is not deasserted until more than approximately 48 bit times after MTXEN is deasserted, then 96 bit times (approximately, but not less) is measured from MCRS.

6.2.9.2.5 Back Off

The EMAC implements the 802.3 binary exponential back-off algorithm.

6.2.9.2.6 Transmit Flow Control

Incoming pause frames are acted upon, when enabled, to prevent the EMAC from transmitting any further frames. Incoming pause frames are only acted upon when the FULLDUPLEX and TXFLOWEN bits in the MAC control register (MACCONTROL) are set. Pause frames are not acted upon in half-duplex mode. Pause frame action is taken if enabled, but normally the frame is filtered and not transferred to memory. MAC control frames are transferred to memory, if the RXCMFEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) is set. The TXFLOWEN and FULLDUPLEX bits affect whether or not MAC control frames are acted upon, but they have no affect upon whether or not MAC control frames are transferred to memory or filtered.

Pause frames are a subset of MAC control frames with an opcode field of 0001h. Incoming pause frames are only acted upon by the EMAC if:

- TXFLOWEN bit is set in MACCONTROL
- The frame's length is 64 to RXMAXLEN bytes inclusive
- The frame contains no CRC error or align/code errors

The pause time value from valid frames is extracted from the two bytes following the opcode. The pause time is loaded into the EMAC transmit pause timer and the transmit pause time period begins. If a valid pause frame is received during the transmit pause time period of a previous transmit pause frame then:

- If the destination address is not equal to the reserved multicast address or any enabled or disabled unicast address, then the transmit pause timer immediately expires, or
- If the new pause time value is 0, then the transmit pause timer immediately expires, else
- The EMAC transmit pause timer immediately is set to the new pause frame pause time value. (Any remaining pause time from the previous pause frame is discarded).

If the TXFLOWEN bit in MACCONTROL is cleared, then the pause timer immediately expires.

The EMAC does not start the transmission of a new data frame any sooner than 512 bit-times after a pause frame with a nonzero pause time has finished being received (MRXDV going inactive). No transmission begins until the pause timer has expired (the EMAC may transmit pause frames in order to initiate outgoing flow control). Any frame already in transmission when a pause frame is received is completed and unaffected.

Incoming pause frames consist of:

- A 48-bit destination address equal to one of the following:
  - The reserved multicast destination address 01.80.C2.00.00.01h
  - Any EMAC 48-bit unicast address. Pause frames are accepted, regardless of whether the channel is enabled or not.
- The 16-bit length/type field containing the value 88.08h.
- The 48-bit source address of the transmitting device.
- The 16-bit pause opcode equal to 00.01h.
- The 16-bit pause time. A pause-quantum is 512 bit-times.
- Padding to 64-byte data length.
- The 32-bit frame-check sequence (CRC word).

All quantities are hexadecimal and are transmitted most-significant byte first. The least-significant bit (LSB) is transferred first in each byte.

The padding is required to make up the frame to a minimum of 64 bytes. The standard allows pause frames longer than 64 bytes to be discarded or interpreted as valid pause frames. The EMAC recognizes any pause frame between 64 bytes and RXMAXLEN bytes in length.

6.2.9.2.7 Speed, Duplex, and Pause Frame Support

The MAC operates at 10 Mbps or 100 Mbps, in half-duplex or full-duplex mode, and with or without pause frame support as configured by the host.

6.2.10 Packet Receive Operation

6.2.10.1 Receive DMA Host Configuration

To configure the receive DMA for operation the host must:

- Initialize the receive addresses.
- Initialize the receive channel *n* DMA head descriptor pointer registers (RX*n*HDP) to 0.
- Write the MAC address hash *n* registers (MACHASH1 and MACHASH2), if hash matching multicast addressing is desired.
- If flow control is to be enabled, initialize:
  - the receive channel *n* free buffer count registers (RX*n*FREEBUFFER)
  - the receive channel *n* flow control threshold register (RX*n*FLOWTHRESH)
  - the receive filter low priority frame threshold register (RXFILTERLOWTHRESH)
- Enable the desired receive interrupts using the receive interrupt mask set register (RXINTMASKSET) and the receive interrupt mask clear register (RXINTMASKCLEAR).
- Set the appropriate configuration bits in the MAC control register (MACCONTROL).
- Write the receive buffer offset register (RXBUFFEROFFSET) value (typically zero).
- Setup the receive channel(s) buffer descriptors and initialize RX*n*HDP.
- Enable the receive DMA controller by setting the RXEN bit in the receive control register (RXCONTROL).
- Configure and enable the receive operation, as desired, in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) and by using the receive unicast set register (RXUNICASTSET) and the receive unicast clear register (RXUNICASTCLEAR).

6.2.10.2 Receive Channel Enabling

Each of the eight receive channels has an enable bit (RXCH*n*EN) in the receive unicast enable set register (RXUNICASTSET) that is controlled using RXUNICASTSET and the receive unicast clear register (RXUNICASTCLEAR). The RXCH*n*EN bits determine whether the given channel is enabled (set to 1) to receive frames with a matching unicast or multicast destination address.

The RXBROADEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) determines if broadcast frames are enabled or filtered. If broadcast frames are enabled, then they are copied to only a single channel selected by the RXBROADCH field in RXMBPENABLE.

The RXMULTEN bit in RXMBPENABLE determines if hash matching multicast frames are enabled or filtered. Incoming multicast addresses (group addresses) are hashed into an index in the hash table. If the indexed bit is set, the frame hash will match and it will be transferred to the channel selected by the RXMULTCH field in RXMBPENABLE when multicast frames are enabled. The multicast hash bits are set in the MAC address hash n registers (MACHASH1 and MACHASH2).

The RXPROMCH bits in RXMBPENABLE select the promiscuous channel to receive frames selected by the RXCMFEN, RXCSFEN, RXCEFEN, and RXCAFEN bits. These four bits allow reception of MAC control frames, short frames, error frames, and all frames (promiscuous), respectively.

The address RAM can be configured to set multiple unicast and/or multicast addresses to a given channel (if the match bit is set in the RAM). Multicast addresses in the RAM are enabled by RXUNICASTSET and not by the RXMULTEN bit in RXMBPENABLE, the RXMULTEN bit enables the hash multicast match only. The address RAM takes precedence over the hash match.

If a multicast packet is received that hash matches (multicast packets enabled), but is filtered in the RAM, then the packet is filtered. If a multicast packet does not hash match, regardless of whether or not hash matching is enabled, but matches an enabled multicast address in the RAM, then the packet will be transferred to the associated channel.

6.2.10.3 Receive Address Matching

The receive address block can store up to 32 addresses to be filtered or matched. Before enabling packet reception, all the address RAM locations should be initialized, including locations to be unused. The system software is responsible for adding and removing addresses from the RAM.

A MAC address location in RAM is 53 bits wide and consists of:

- 48 bits of the MAC address.
- 3 bits for the channel to which a valid address match will be transferred. The channel is a don't care if MATCHFILT bit is cleared
- A valid bit
- A match or filter bit

First, write the index into the address RAM in the MACINDEX register to start writing a MAC address. Then write the upper 32 bits of the MAC address (MACADDRHI register), and then the lower 16 bits of MAC address with the VALID and MATCHFILT control bits (MACADDRLO). The valid bit should be cleared for the unused locations in the receive address RAM.

The most common uses for the receive address sub-module are:

- Set EMAC in promiscuous mode, using RXCAFEN and RXPROMCH bits in the RXMBPENABLE register. Then filter up to 32 individual addresses, which can be both unicast and/or multicast.
- Disable the promiscuous mode (RXCAFEN = 0) and match up to 32 individual addresses, multicast and/or unicast.

6.2.10.4 Hardware Receive QOS Support

Hardware receive quality of service (QOS) is supported, when enabled, by the Tag Protocol Identifier format and the associated Tag Control Information (TCI) format priority field. When the incoming frame length/type value is equal to 81.00h, the EMAC recognizes the frame as an Ethernet Encoded Tag Protocol Type. The two octets immediately following the protocol type contain the 16-bit TCI field. Bits 15-13 of the TCI field contain the received frames priority (0 to 7). The received frame is a low-priority frame, if the priority value is 0 to 3; the received frame is a high-priority frame, if the priority value is 4 to 7. All frames that have a length/type field value not equal to 81.00h are low-priority frames. Received frames that contain priority information are determined by the EMAC as:

- A 48-bit (6 bytes) destination address equal to:
  - The destination station's individual unicast address.
  - The destination station's multicast address (MACHASH1 and MACHASH2).
  - The broadcast address of all ones.
- A 48-byte (6 bytes) source address.
- The 16-bit (2 bytes) length/type field containing the value 81.00h.
- The 16-bit (2 bytes) TCI field with the priority field in the upper 3 bits.
- Data bytes
- The 4 bytes CRC.

The receive filter low priority frame threshold register (RXFILTERLOWTHRESH) and the receive channel *n* free buffer count registers (RX *n* FREEBUFFER) are used in conjunction with the priority information to implement receive hardware QOS. Low-priority frames are filtered if the number of free buffers (RX *n* FREEBUFFER) for the frame channel is less than or equal to the filter low threshold (RXFILTERLOWTHRESH) value. Hardware QOS is enabled by the RXQOSEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE).

6.2.10.5 Host Free Buffer Tracking

The host must track free buffers for each enabled channel (including unicast, multicast, broadcast, and promiscuous), if receive QOS or receive flow control is used. Disabled channel free buffer values are do not cares. During initialization, the host should write the number of free buffers for each enabled channel to the appropriate receive channel *n* free buffer count registers (RX*n*FREEBUFFER). The EMAC decrements the appropriate channel's free buffer value for each buffer used. When the host reclaims the frame buffers, the host should write the channel free buffer register with the number of reclaimed buffers (write to increment). There are a maximum of 65,535 free buffers available. RX*n*FREEBUFFER only needs to be updated by the host if receive QOS or flow control is used.

6.2.10.6 Receive Channel Teardown

The host commands a receive channel teardown by writing the channel number to the receive teardown register (RXTEARDOWN). When a teardown command is issued to an enabled receive channel, the following occurs:

- Any current frame in reception completes normally.
- The TDOWNCMPLT flag is set in the next buffer descriptor in the chain, if there is one.
- The channel head descriptor pointer is cleared to 0.
- A receive interrupt for the channel is issued to the host.
- The corresponding receive channel *n* completion pointer register (RX*n*CP) contains the value FFFF FFFCh.

Channel teardown may be commanded on any channel at any time. The host is informed of the teardown completion by the set teardown complete (TDOWNCMPLT) buffer descriptor bit. The EMAC does not clear any channel enables due to a teardown command. A teardown command to an inactive channel issues an interrupt that software should acknowledge with an FFFF FFFCh acknowledge value to RX*n*CP (note that there is no buffer descriptor in this case). Software may read RX*n*CP to determine if the interrupt was due to a commanded teardown. The read value is FFFF FFFCh, if the interrupt was due to a teardown command.

6.2.10.7 Receive Frame Classification

Received frames are proper (good) frames, if they are between 64 bytes and the value in the receive maximum length register (RXMAXLEN) bytes in length (inclusive) and contain no code, align, or CRC errors.

Received frames are long frames, if their frame count exceeds the value in RXMAXLEN. The RXMAXLEN reset (default) value is 5EEh (1518 in decimal). Long received frames are either oversized or jabber frames. Long frames with no errors are oversized frames; long frames with CRC, code, or alignment errors are jabber frames.

Received frames are short frames, if their frame count is less than 64 bytes. Short frames that address match and contain no errors are undersized frames; short frames with CRC, code, or alignment errors are fragment frames. If the frame length is less than or equal to 20, then the frame CRC is passed, regardless of whether the RXPASSCRC bit is set or cleared in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE).

A received long packet always contains RXMAXLEN number of bytes transferred to memory (if the RXCEFEN bit is set in RXMBPENABLE), regardless of the value of the RXPASSCRC bit. The following is an example with RXMAXLEN set to 1518:

- If the frame length is 1518, then the packet is not a long packet and there are 1514 or 1518 bytes transferred to memory depending on the value of the RXPASSCRC bit.
- If the frame length is 1519, there are 1518 bytes transferred to memory regardless of the RXPASSCRC bit value. The last three bytes are the first three CRC bytes.
- If the frame length is 1520, there are 1518 bytes transferred to memory regardless of the RXPASSCRC bit value. The last two bytes are the first two CRC bytes.
- If the frame length is 1521, there are 1518 bytes transferred to memory regardless of the RXPASSCRC bit value. The last byte is the first CRC byte.

- If the frame length is 1522, there are 1518 bytes transferred to memory. The last byte is the last data byte.

6.2.10.8 Promiscuous Receive Mode

When the promiscuous receive mode is enabled by setting the RXCAFEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE), nonaddress matching frames that would normally be filtered are transferred to the promiscuous channel. Address matching frames that would normally be filtered due to errors are transferred to the address match channel when the RXCAFEN and RXCEFEN bits in RXMBPENABLE are set. A frame is considered to be an address matching frame only if it is enabled to be received on a unicast, multicast, or broadcast channel. Frames received to disabled unicast, multicast, or broadcast channels are considered nonaddress matching.

MAC control frames address match only if the RXCMFEN bit in RXMBPENABLE is set. The RXCEFEN and RXCSFEN bits in RXMBPENABLE determine whether error frames are transferred to memory or not, but they do not determine whether error frames are address matching or not. Short frames are a special type of error frames.

A single channel is selected as the promiscuous channel by the RXPROMCH bit in RXMBPENABLE. The promiscuous receive mode is enabled by the RXCMFEN, RXCEFEN, RXCSFEN, and RXCAFEN bits in RXMBPENABLE. Table 6-6 shows the effects of the promiscuous enable bits. Proper frames are frames that are between 64 bytes and the value in the receive maximum length register (RXMAXLEN) bytes in length inclusive and contain no code, align, or CRC errors.

Table 6-6. Receive Frame Treatment Summary

Address Match	RXCAFEN	RXCEFEN	RXCMFEN	RXCSFEN	Receive Frame Treatment
0	0	X	X	X	No frames transferred.
0	1	0	0	0	Proper frames transferred to promiscuous channel.
0	1	0	0	1	Proper/undersized data frames transferred to promiscuous channel.
0	1	0	1	0	Proper data and control frames transferred to promiscuous channel.
0	1	0	1	1	Proper/undersized data and control frames transferred to promiscuous channel.
0	1	1	0	0	Proper/oversize/jabber/code/align/CRC data frames transferred to promiscuous channel. No control or undersized/fragment frames are transferred.
0	1	1	0	1	Proper/undersized/fragment/oversize/jabber/code/ align/CRC data frames transferred to promiscuous channel. No control frames are transferred.
0	1	1	1	0	Proper/oversize/jabber/code/align/CRC data and control frames transferred to promiscuous channel. No undersized frames are transferred.
0	1	1	1	1	All nonaddress matching frames with and without errors transferred to promiscuous channel.
1	X	0	0	0	Proper data frames transferred to address match channel.
1	X	0	0	1	Proper/undersized data frames transferred to address match channel.
1	X	0	1	0	Proper data and control frames transferred to address match channel.
1	X	0	1	1	Proper/undersized data and control frames transferred to address match channel.
1	X	1	0	0	Proper/oversize/jabber/code/align/CRC data frames transferred to address match channel. No control or undersized frames are transferred.
1	X	1	0	1	Proper/oversize/jabber/fragment/undersized/code/ align/CRC data frames transferred to address match channel. No control frames are transferred.

Table 6-6. Receive Frame Treatment Summary (continued)

Address Match	RXCAFEN	RXCEFEN	RXCMFEN	RXCSFEN	Receive Frame Treatment
1	X	1	1	0	Proper/oversize/jabber/code/align/CRC data and control frames transferred to address match channel. No undersized/fragment frames are transferred.
1	X	1	1	1	All address matching frames with and without errors transferred to the address match channel

6.2.10.9 Receive Overrun

The types of receive overrun are:

- FIFO start of frame overrun (FIFO_SOF)
- FIFO middle of frame overrun (FIFO_MOF)
- DMA start of frame overrun (DMA_SOF)
- DMA middle of frame overrun (DMA_MOF)

The statistics counters used to track these types of receive overrun are:

- Receive start of frame overruns register (RXSOFOVERRUNS)
- Receive middle of frame overruns register (RXMOFOVERRUNS)
- Receive DMA overruns register (RXDMAOVERRUNS)

Start of frame overruns happen when there are no resources available when frame reception begins. Start of frame overruns increment the appropriate overrun statistic(s) and the frame is filtered.

Middle of frame overruns happen when there are some resources to start the frame reception, but the resources run out during frame reception. In normal operation, a frame that overruns after starting the frame reception is filtered and the appropriate statistic(s) are incremented; however, the RXCEFEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) affects overrun frame treatment. Table 6-7 shows how the overrun condition is handled for the middle of frame overrun.

Table 6-7. Middle of Frame Overrun Treatment

Address Match	RXCAFEN	RXCEFEN	Middle of Frame Overrun Treatment
0	0	X	Overrun frame filtered.
0	1	0	Overrun frame filtered.
0	1	1	As much frame data as possible is transferred to the promiscuous channel until overrun. The appropriate overrun statistic(s) is incremented and the OVERRUN and NOMATCH flags are set in the SOP buffer descriptor. Note that the RXMAXLEN number of bytes cannot be reached for an overrun to occur (it would be truncated and be a jabber or oversize).
1	X	0	Overrun frame filtered with the appropriate overrun statistic(s) incremented.
1	X	1	As much frame data as possible is transferred to the address match channel until overrun. The appropriate overrun statistic(s) is incremented and the OVERRUN flag is set in the SOP buffer descriptor. Note that the RXMAXLEN number of bytes cannot be reached for an overrun to occur (it would be truncated).

6.2.11 Packet Transmit Operation

The transmit DMA is an eight channel interface. Priority between the eight queues may be either fixed or round-robin as selected by the TXPTYPE bit in the MAC control register (MACCONTROL). If the priority type is fixed, then channel 7 has the highest priority and channel 0 has the lowest priority. Round-robin priority proceeds from channel 0 to channel 7.

6.2.11.1 Transmit DMA Host Configuration

To configure the transmit DMA for operation the host must perform:

- Write the MAC source address low bytes register (MACSRCADDRLO) and the MAC source address high bytes register (MACSRCADDRHI) (used for pause frames on transmit).
- Initialize the transmit channel *n* DMA head descriptor pointer registers (TX*n*HDP) to 0.
- Enable the desired transmit interrupts using the transmit interrupt mask set register (TXINTMASKSET) and the transmit interrupt mask clear register (TXINTMASKCLEAR).
- Set the appropriate configuration bits in the MAC control register (MACCONTROL).
- Setup the transmit channel(s) buffer descriptors in host memory.
- Enable the transmit DMA controller by setting the TXEN bit in the transmit control register (TXCONTROL).
- Write the appropriate TX*n*HDP with the pointer to the first descriptor to start transmit operations.

6.2.11.2 Transmit Channel Teardown

The host commands a transmit channel teardown by writing the channel number to the transmit teardown register (TXTEARDOWN). When a teardown command is issued to an enabled transmit channel, the following occurs:

- Any frame currently in transmission completes normally.
- The TDOWNCMPLT flag is set in the next SOP buffer descriptor in the chain, if there is one.
- The channel head descriptor pointer is cleared to 0.
- A transmit interrupt is issued to inform the host of the channel teardown.
- The corresponding transmit channel *n* completion pointer register (TX*n*CP) contains the value FFFF FFFCh.
- The host should acknowledge a teardown interrupt with an FFFF FFFCh acknowledge value.

Channel teardown may be commanded on any channel at any time. The host is informed of the teardown completion by the set teardown complete (TDOWNCMPLT) buffer descriptor bit. The EMAC does not clear any channel enables due to a teardown command. A teardown command to an inactive channel issues an interrupt that software should acknowledge with an FFFF FFFCh acknowledge value to TX*n*CP (note that there is no buffer descriptor in this case). Software may read the interrupt acknowledge location (TX*n*CP) to determine if the interrupt was due to a commanded teardown. The read value is FFFF FFFCh, if the interrupt was due to a teardown command.

6.2.12 Receive and Transmit Latency

The transmit FIFO contains twenty-four 64-byte cells and the receive FIFO contains sixty-eight 64-byte cells. The EMAC begins transmission of a packet on the wire after TXCELLTHRESH cells (configurable through the FIFO control register, FIFOCONTROL) or a complete packet are available in the FIFO.

Transmit underrun cannot occur for packet sizes of TXCELLTHRESH × 64 bytes (or less). For larger packet sizes, transmit underrun can occur if the memory latency is greater than the time required to transmit a 64-byte cell on the wire; this is 0.512 ms in 1 Gbit mode, 5.12 ms in 100 Mbps mode, and 51.2 ms in 10 Mbps mode. The memory latency time includes all buffer descriptor reads for the entire cell data. The EMAC transmit FIFO uses 24 cells; thus, underrun cannot happen for a normal size packet (less than 1536 packet bytes). Cell transmission can be configured to start only after an entire packet is contained in the FIFO; for a maximum-size packet, set the TXCELLTHRESH field to the maximum possible value of 24.

Receive overrun is prevented if the receive memory cell latency is less than the time required to transmit a 64-byte cell on the wire (0.512 ms in 1 Gbps mode, 5.12 ms in 100 Mbps mode, or 51.2ms in 10 Mbps mode). The latency time includes any required buffer descriptor reads for the cell data.

Latency to descriptor RAM is low because RAM is local to the EMAC, as it is part of the EMAC control module.

6.2.13 Transfer Node Priority

The device contains a chip-level register, master priority register (MSTPRI), that is used to set the priority of the transfer node used in issuing memory transfer requests to system memory.

Although the EMAC has internal FIFOs to help alleviate memory transfer arbitration problems, the average transfer rate of data read and written by the EMAC to internal or external processor memory must be at least that of the Ethernet wire rate. In addition, the internal FIFO system can not withstand a single memory latency event greater than the time it takes to fill or empty a TXCELLTHRESH number of internal 64-byte FIFO cells.

For 100 Mbps operation, these restrictions translate into the following rules:

- The short-term average, each 64-byte memory read/write request from the EMAC must be serviced in no more than 5.12 µs.
- Any single latency event in request servicing can be no longer than (5.12 × TXCELLTHRESH) µs.

Bits 0-2 of the second chip-level master priority register (MSTPRI1) are used to set the transfer node priority within the Switched Central Resource (SCR5) for the EMAC master peripheral.

A value of 000b has the highest priority, while 111b has the lowest priority. The default priority assigned to the EMAC is 100b. It is important to have a balance between all peripherals. In most cases, the default priorities will not need adjustment. For more information on the master peripherals priorities, see the device-specific data manual.

6.2.14 Reset Considerations

6.2.14.1 Software Reset Considerations

Peripheral clock and reset control is done through the PRCM module included with the device. For more on how the EMAC, MDIO, and EMAC control module are disabled or placed in reset at runtime from the registers located in the PRCM module, see Section 14.2.13.

NOTE: For proper operation, both the EMAC and EMAC control module must be reset in the following sequence. First, the soft reset of the EMAC module should be commanded. After the reset takes effect (verified by reading back SOFTRESET), the soft reset of the EMAC control module should be commanded.

Within the peripheral there are two controls to separately reset the EMAC and the EMAC control module.

- The EMAC component of the Ethernet MAC peripheral can be placed in a reset state by writing to the soft reset register (SOFTRESET). Writing a 1 to the SOFTRESET bit, causes the EMAC logic to be reset and the register values to be set to their default values. Software reset occurs when the receive and transmit DMA controllers are in an idle state to avoid locking up the configuration bus; it is the responsibility of the software to verify that there are no pending frames to be transferred. After writing a 1 to the SOFTRESET bit, it may be polled to determine if the reset has occurred. If a 1 is read, the reset has not yet occurred; if a 0 is read, then a reset has occurred.
- The Software Reset Register (SOFT_RESET) is used to place the EMAC control module logic in soft reset. This resets the control logic, the EMAC registers, as well as, the EMAC control module 8KB internal memory that may be used for storing the transfer descriptors.

After a software reset operation, all the EMAC registers need to be reinitialized for proper data transmission.

Unlike the EMAC module, the MDIO and EMAC control modules cannot be placed in reset from a register inside their memory map.

6.2.14.2 Hardware Reset Considerations

When a hardware reset occurs, the EMAC peripheral has its register values reset and all the components return to their default state. After the hardware reset, the EMAC needs to be initialized before being able to resume its data transmission, as described in Section 6.2.15.

A hardware reset is the only means of recovering from the error interrupts (HOSTPEND), which are triggered by errors in packet buffer descriptors. Before doing a hardware reset, you should inspect the error codes in the MAC status register (MACSTATUS) that gives information about the type of software error that needs to be corrected. For detailed information on error interrupts, see Section 6.2.16.1.5.

6.2.15 Initialization

6.2.15.1 Enabling the EMAC/MDIO Peripheral

When the device is powered on, the EMAC peripheral is in a disabled state. Before any EMAC specific initialization can take place, the EMAC needs to be enabled; otherwise, its registers cannot be written and the reads will all return a value of zero.

The EMAC/MDIO is enabled through the PRCM registers. For information on how to enable the EMAC peripheral from the PRCM, see the PRCM Reference Guide.

When first enabled, the EMAC peripheral registers are set to their default values. After enabling the peripheral, you may proceed with the module specific initialization.

6.2.15.2 EMAC Control Module Initialization

The EMAC control module is used for global interrupt enable, and to pace back-to-back interrupts using an interrupt retrigger count based on the peripheral clock (SYSCLK5). There is also an 8K block of RAM local to the EMAC that is used to hold packet buffer descriptors.

Note that although the EMAC control module and the EMAC module have slightly different functions, in practice, the type of maintenance performed on the EMAC control module is more commonly conducted from the EMAC module software (as opposed to the MDIO module).

The initialization of the EMAC control module consists of two parts:

1. Configuration of the interrupt to the CPU.
2. Initialization of the EMAC control module:
   - Setting the registers related to interrupt pacing. This applies only to RXPulse and TXPulse interrupts. By default, interrupts pacing is disabled. If pacing is enabled by programming the EMAC control module interrupt control register (CMINTCTRL), then the CMTXINTMAX and CMRXINTMAX registers have to be programmed, to indicate the maximum number of TX_PULSE and RX_PULSE interrupts per millisecond.
   - Initializing the EMAC and MDIO modules.
   - Enabling interrupts in the EMAC control module using the EMAC control module interrupt control registers (CMRXTHRESHINTEN, CMRXINTEN, CMTXINTEN, and CMMISCINTEN).

When using the register-level CSL, the code to perform the actions associated with the second part may appear as in Example 6-4.

The process of mapping the EMAC interrupts to one of the CPU's interrupts is done using the ARM interrupt controller. Once the interrupt is mapped to a CPU interrupt, general masking and unmasking of the interrupt (to control reentrancy) should be done at the chip level by manipulating the interrupt enable mask.

Example 6-4. EMAC Control Module Initialization Code

```
Uint32 tmpval ;

/* Disable all the EMAC/MDIO interrupts in the control module */
EmacControlRegs->CONTROL.CMRXINTEN        = 0;
EmacControlRegs->CONTROL.C_TX_EN        = 0;
EmacControlRegs->CONTROL.CMRXTHRESHINTEN = 0;
EmacControlRegs->CONTROL.C_MISC_EN      = 0;


/* Wait about 100 cycles */
for( I=0; i<5; I++ )
  tmpval = ECTL_REGS->EWCTL ;


#ifdef INTT_PACING
/* Set the control related to pacing of TX and RX interrupts */
    EmacControlRegs->INTR_COUNT->C_RX_IMAX = 0x4; // 4 RX intt/ms
    EmacControlRegs->INTR_COUNT->C_TX_IMAX = 0x4; // 4 TX intt/ms
    EmacControlRegs->CMINTCTRL = 0x30000; //bit16,bit17 for enabling TX and Rx intt pacing.
    EmacControlRegs->CMINTCTRL |= 0x258; // 600 clocks of 150MHz in 4us time
#endif

/* Initialize MDIO and EMAC Module */
[Discussed later in this document]


/* Enable all the EMAC/MDIO interrupts in the control module */
EmacControlRegs->CONTROL.CMRXINTEN        = 0xff;
EmacControlRegs->CONTROL.C_TX_EN        = 0xff;
EmacControlRegs->CONTROL.CMRXTHRESHINTEN = 0xff;
EmacControlRegs->CONTROL.C_MISC_EN      = 0xf;
```

6.2.15.3 MDIO Module Initialization

The MDIO module is used to initially configure and monitor one or more external PHY devices. Other than initializing the software state machine (details on this state machine can be found in the IEEE 802.3 standard), all that needs to be done for the MDIO module is to enable the MDIO engine and to configure the clock divider. To set the clock divider, supply an MDIO clock of 1 MHz. For example, since the base clock used is the peripheral clock (SYSCLK5), for a SYSCLK5 at a frequency of 250 MHz the divider can be set to 99, with slower MDIO clocks for slower peripheral clock frequencies being perfectly acceptable.

Both the state machine enable and the MDIO clock divider are controlled through the MDIO control register (CONTROL). If none of the potentially connected PHYs require the access preamble, the PREAMBLE bit in CONTROL can also be set to speed up PHY register access. The code for this may appear as in Example 6-5.

Example 6-5. MDIO Module Initialization Code

```
#define PCLK 99
 ...
 /* Enable MDIO and setup divider */
 MDIO_REGS->CONTROL = CSL_FMKT( MDIO_CONTROL_ENABLE, YES) |
 CSL_FMK( MDIO_CONTROL_CLKDIV, PCLK ) ;
```

If the MDIO module is to operate on an interrupt basis, the interrupts can be enabled at this time using the MDIO user command complete interrupt mask set register (USERINTMASKSET) for register access and the MDIO user PHY select register (USERPHYSEL*n*) if a target PHY is already known.

Once the MDIO state machine has been initialized and enabled, it starts polling all 32 PHY addresses on the MDIO bus, looking for an active PHY. Since it can take up to 50 µs to read one register, it can be some time before the MDIO module provides an accurate representation of whether a PHY is available. Also, a PHY can take up to 3 seconds to negotiate a link. Thus, it is advisable to run the MDIO software off a time-based event rather than polling.

For more information on PHY control registers, see your PHY device documentation.

6.2.15.4 EMAC Module Initialization

The EMAC module is used to send and receive data packets over the network. This is done by maintaining up to eight transmit and receive descriptor queues. The EMAC module configuration must also be kept up-to-date based on PHY negotiation results returned from the MDIO module. Most of the work in developing an application or device driver for Ethernet is programming this module.

The following is the initialization procedure a device driver would follow to get the EMAC to the state where it is ready to receive and send Ethernet packets. Some of these steps are not necessary when performed immediately after device reset.

1. Program the VDD3P3V_PWDN register in the System module to power up the CPGMAC I/O cells (see the device-specific data manual). There are separate controls for MII I/O pins and GMII I/O pins, which are powered down by default at system reset, to save power.
2. If enabled, clear the device interrupt enable in the EMAC control module interrupt control registers (CMRXTHRESHINTEN, CMRXINTEN, CMTXINTEN, and CMMISCINTEN).
3. Clear the MAC control register (MACCONTROL), receive control register (RXCONTROL), and transmit control register (TXCONTROL) (not necessary immediately after reset).
4. Initialize all 16 header descriptor pointer registers (RX*n*HDP and TX*n*HDP) to 0.
5. Clear all 36 statistics registers by writing 0 (not necessary immediately after reset).
6. Setup the local Ethernet MAC address by programming the MAC index register (MACINDEX), MAC address high bytes register (MACADDRHI), and MAC address low bytes register (MACADDRLO). Be sure to program all eight MAC addresses - whether the receive channel is to be enabled or not. Duplicate the same MAC address across all unused channels. When using more than one receive channel, start with channel 0 and progress upwards.
7. Initialize the receive channel *n* free buffer count registers (RX*n*FREEBUFFER), receive channel *n* flow control threshold register (RX*n*FLOWTHRESH), and receive filter low priority frame threshold register (RXFILTERLOWTHRESH), if buffer flow control is to be enabled.
8. Most device drivers open with no multicast addresses, so clear the MAC address hash registers (MACHASH1 and MACHASH2) to 0.
9. Write the receive buffer offset register (RXBUFFEROFFSET) value (typically zero).
10. Initially clear all unicast channels by writing FFh to the receive unicast clear register (RXUNICASTCLEAR). If unicast is desired, it can be enabled now by writing the receive unicast set register (RXUNICASTSET). Some drivers will default to unicast on device open while others will not.
11. Setup the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) with an initial configuration. The configuration is based on the current receive filter settings of the device driver. Some drivers may enable things like broadcast and multicast packets immediately, while others may not.
12. Set the appropriate configuration bits in MACCONTROL (do not set the GMIIEN bit yet).
13. Clear all unused channel interrupt bits by writing the receive interrupt mask clear register (RXINTMASKCLEAR) and the transmit interrupt mask clear register (TXINTMASKCLEAR).
14. Enable the receive and transmit channel interrupt bits in the receive interrupt mask set register (RXINTMASKSET) and the transmit interrupt mask set register (TXINTMASKSET) for the channels to be used, and enable the HOSTMASK and STATMASK bits using the MAC interrupt mask set register (MACINTMASKSET).
15. Initialize the receive and transmit descriptor list queues.
16. Prepare receive by writing a pointer to the head of the receive buffer descriptor list to RX*n*HDP.
17. Enable the receive and transmit DMA controllers by setting the RXEN bit in RXCONTROL and the TXEN bit in TXCONTROL. Then set the GMIIEN bit in MACCONTROL.
18. Enable the device interrupt in the EMAC control module interrupt control registers (CMRXTHRESHINTEN, CMRXINTEN, CMTXINTEN, and CMMISCINTEN).

6.2.16 Interrupt Support

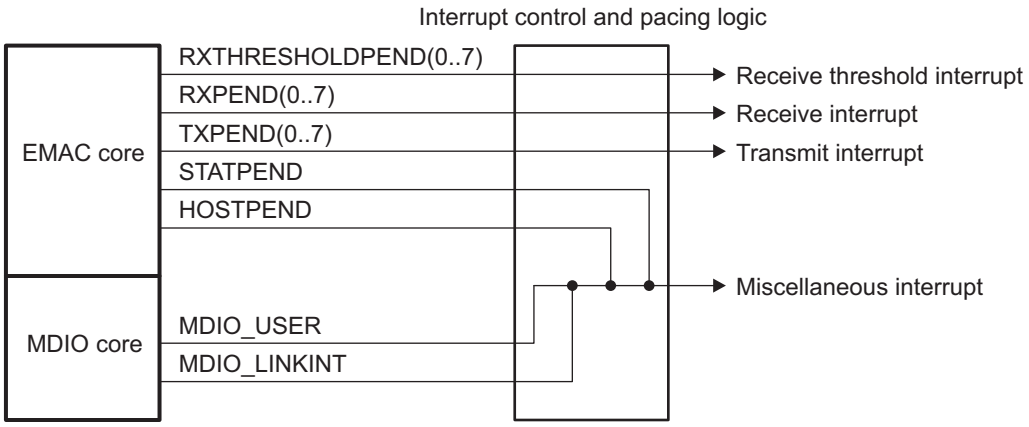
6.2.16.1 EMAC Module Interrupt Events and Requests

The EMAC module generates the following interrupt events:

- RXTHRESHOLDPEND*n*: Receive threshold interrupt for receive channels 0 through 7
- RXPEND*n*: Receive packet completion interrupt for receive channels 0 through 7
- TXPEND*n*: Transmit packet completion interrupt for transmit channels 0 through 7
- STATPEND: Statistics interrupt
- HOSTPEND: Host error interrupt
- USERINT: MDIO user Interrupt
- LINKINT: MDIO link Interrupt

As shown in Figure 6-11, the EMAC and MDIO interrupts are multiplexed on four interrupts lines going to the CPU.

Figure 6-11. EMAC Control Module Interrupt Logic Diagram



6.2.16.1.1 Receive Threshold Interrupts

Each of the eight receive channels have a corresponding receive threshold interrupt (RX_THRESH_PEND[0:7]). The receive threshold interrupts are level interrupts that remain asserted until the triggering condition is cleared by the host. Each of the eight threshold interrupts may be individually enabled by setting the corresponding bit in the receive interrupt mask set register (RXINTMASKSET) to 1. Each of the eight channel interrupts may be individually disabled by clearing the corresponding bit in the receive interrupt mask clear register (RXINTMASKCLEAR) to 0. The raw and masked receive interrupt status may be read from the receive interrupt status (unmasked) register (RXINTSTATRAW) and the receive interrupt status (masked) register (RXINTSTATMASKED), respectively. An RX_THRES_PEND[7:0] interrupt bit is asserted when enabled and when the channel's associated receive channel *n* free buffer count register (RX*n*FREEBUFFER) is less than or equal to the channel's associated receive channel *n* flow control threshold register (RX*n*FLOWTHRESH). The receive threshold interrupts use the same free buffer count and threshold logic as does flow control, but the interrupts are independently enabled from flow control. The threshold interrupts are intended to give the host an indication that resources are running low for a particular channel(s).

6.2.16.1.2 Transmit Packet Completion Interrupts

The transmit DMA engine has eight channels, with each channel having a corresponding interrupt (TXPEND*n*). The transmit interrupts are level interrupts that remain asserted until cleared by the CPU.

Each of the eight transmit channel interrupts may be individually enabled by setting the corresponding bit in the transmit interrupt mask set register (TXINTMASKSET) to 1. Each of the eight transmit channel interrupts may be individually disabled by clearing the corresponding bit in the transmit interrupt mask clear register (TXINTMASKCLEAR) to 0. The raw and masked transmit interrupt status may be read from the transmit interrupt status (unmasked) register (TXINTSTATRAW) and the transmit interrupt status (masked) register (TXINTSTATMASKED), respectively.

When the EMAC completes the transmission of a packet, the EMAC issues an interrupt to the CPU by writing the packet's last buffer descriptor address to the appropriate channel queue's transmit completion pointer located in the state RAM block. The interrupt is generated by the write when enabled by the interrupt mask, regardless of the value written.

Upon interrupt reception, the CPU processes one or more packets from the buffer chain and then acknowledges an interrupt by writing the address of the last buffer descriptor processed to the queue's associated transmit completion pointer in the transmit DMA state RAM.

The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the EMAC port (address of last buffer descriptor used by the EMAC). If the two values are not equal (which means that the EMAC has transmitted more packets than the CPU has processed interrupts for), the transmit packet completion interrupt signal remains asserted. If the two values are equal (which means that the host has processed all packets that the EMAC has transferred), the pending interrupt is cleared. The value that the EMAC is expecting is found by reading the transmit channel *n* completion pointer register (TX*n*CP).

The EMAC write to the completion pointer actually stores the value in the state RAM. The CPU written value does not actually change the register value. The host written value is compared to the register content (which was written by the EMAC) and if the two values are equal then the interrupt is removed; otherwise, the interrupt remains asserted. The host may process multiple packets prior to acknowledging an interrupt, or the host may acknowledge interrupts for every packet.

6.2.16.1.3 Receive Packet Completion Interrupts

The receive DMA engine has eight channels, which each channel having a corresponding interrupt (RXPEND*n*). The receive interrupts are level interrupts that remain asserted until cleared by the CPU.

Each of the eight receive channel interrupts may be individually enabled by setting the corresponding bit in the receive interrupt mask set register (RXINTMASKSET) to 1. Each of the eight receive channel interrupts may be individually disabled by clearing the corresponding bit in the receive interrupt mask clear register (RXINTMASKCLEAR) to 0. The raw and masked receive interrupt status may be read from the receive interrupt status (unmasked) register (RXINTSTATRAW) and the receive interrupt status (masked) register (RXINTSTATMASKED), respectively.

When the EMAC completes a packet reception, the EMAC issues an interrupt to the CPU by writing the packet's last buffer descriptor address to the appropriate channel queue's receive completion pointer located in the state RAM block. The interrupt is generated by the write when enabled by the interrupt mask, regardless of the value written.

Upon interrupt reception, the CPU processes one or more packets from the buffer chain and then acknowledges one or more interrupt(s) by writing the address of the last buffer descriptor processed to the queue's associated receive completion pointer in the receive DMA state RAM.

The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the EMAC (address of last buffer descriptor used by the EMAC). If the two values are not equal (which means that the EMAC has received more packets than the CPU has processed interrupts for), the receive packet completion interrupt signal remains asserted. If the two values are equal (which means that the host has processed all packets that the EMAC has received), the pending interrupt is de-asserted. The value that the EMAC is expecting is found by reading the receive channel *n* completion pointer register (RX*n*CP).

The EMAC write to the completion pointer actually stores the value in the state RAM. The CPU written value does not actually change the register value. The host written value is compared to the register content (which was written by the EMAC) and if the two values are equal then the interrupt is removed; otherwise, the interrupt remains asserted. The host may process multiple packets prior to acknowledging an interrupt, or the host may acknowledge interrupts for every packet.

6.2.16.1.4 Statistics Interrupt

The statistics level interrupt (STATPEND) is issued when any statistics value is greater than or equal to 8000 0000h, if enabled by setting the STATMASK bit in the MAC interrupt mask set register (MACINTMASKSET) to 1. The statistics interrupt is removed by writing to decrement any statistics value greater than 8000 0000h. As long as the most-significant bit of any statistics value is set, the interrupt remains asserted.

6.2.16.1.5 Host Error Interrupt

The host error interrupt (HOSTPEND) is issued, if enabled, under error conditions dealing with the handling of buffer descriptors, detected during transmit or receive DMA transactions. The failure of the software application to supply properly formatted buffer descriptors results in this error. The error bit can only be cleared by resetting the EMAC module in hardware.

The host error interrupt is enabled by setting the HOSTMASK bit in the MAC interrupt mask set register (MACINTMASKSET) to 1. The host error interrupt is disabled by clearing the appropriate bit in the MAC interrupt mask clear register (MACINTMASKCLEAR) to 0. The raw and masked host error interrupt status may be read by reading the MAC interrupt status (unmasked) register (MACINTSTATRAW) and the MAC interrupt status (masked) register (MACINTSTATMASKED), respectively.

The transmit host error conditions are:

- SOP error
- Ownership bit not set in SOP buffer
- Zero next buffer descriptor pointer with EOP
- Zero buffer pointer
- Zero buffer length
- Packet length error

The receive host error conditions are:

- Ownership bit not set in input buffer
- Zero buffer pointer

6.2.16.2 MDIO Module Interrupt Events and Requests

The MDIO module generates two interrupt events:

- LINKINT: Serial interface link change interrupt. Indicates a change in the state of the PHY link
- USERINT: Serial interface user command event complete interrupt

6.2.16.2.1 Link Change Interrupt

The MDIO module asserts a link change interrupt (LINKINT) if there is a change in the link state of the PHY corresponding to the address in the PHYADRMON bit in the MDIO user PHY select register *n* (USERPHYSEL*n*), and if the LINKINTENB bit is also set in USERPHYSEL*n*. This interrupt event is also captured in the LINKINTRAW bit in the MDIO link status change interrupt register (LINKINTRAW). LINKINTRAW bits 0 and 1 correspond to USERPHYSEL0 and USERPHYSEL1, respectively.

When the interrupt is enabled and generated, the corresponding LINKINTMASKED bit is also set in the MDIO link status change interrupt register (LINKINTMASKED). The interrupt is cleared by writing back the same bit to LINKINTMASKED (write to clear).

6.2.16.2.2 User Access Completion Interrupt

When the GO bit in one of the MDIO user access registers (USERACCESS*n*) transitions from 1 to 0 (indicating completion of a user access) and the corresponding USERINTMASKSET bit in the MDIO user command complete interrupt mask set register (USERINTMASKSET) corresponding to USERACCESS0 or USERACCESS1 is set, a user access completion interrupt (USERINT) is asserted. This interrupt event is also captured in the USERINTRAW bit in the MDIO user command complete interrupt register (USERINTRAW). USERINTRAW bits 0 and bit 1 correspond to USERACCESS0 and USERACCESS1, respectively.

When the interrupt is enabled and generated, the corresponding USERINTMASKED bit is also set in the MDIO user command complete interrupt register (USERINTMASKED). The interrupt is cleared by writing back the same bit to USERINTMASKED (write to clear).

6.2.16.3 Proper Interrupt Processing

All the interrupts signaled from the EMAC and MDIO modules are level driven, so if they remain active, their level remains constant; the CPU core requires edge-triggered interrupts. In order to properly convert the level-driven interrupt signal to an edge-triggered signal, the application software must make use of the interrupt control logic contained in the EMAC control module.

Section 6.2.6.3 discusses the interrupt control contained in the EMAC control module. For safe interrupt processing, upon entry to the ISR, the software application should disable interrupts using the EMAC control module interrupt control registers (CMRXTHRESHINTEN, CMRXINTEN, CMTXINTEN, and CMMISCINTEN), and then reenable them upon leaving the ISR. If any interrupt signals are active at that time, this creates another rising edge on the interrupt signal going to the CPU interrupt controller, thus triggering another interrupt. The EMAC control module also implements the optional interrupt pacing.

6.2.16.4 Interrupt Multiplexing

TThe EMAC control module combines all the different interrupt signals from both the EMAC and MDIO modules and generates four separate interrupt signals (Table 6-5) that are routed to the host processor(s).

Once this interrupt is generated, the reason for the interrupt can be read from the MAC input vector register (MACINVECTOR) located in the EMAC memory map. MACINVECTOR combines the status of the following 28 interrupt signals: TXPEND*n*, RXPEND*n*, RXTHRESHPEND*n*, STATPEND, HOSTPEND, LINKINT, and USERINT.

For more details on the ARM interrupt controller (AINTC), see the SoC Subsystem Reference Guide.

6.2.17 Power Management

Each of the three main components of the EMAC peripheral can independently be placed in reduced-power modes to conserve power during periods of low activity. The power management of the EMAC peripheral is controlled by the PRCM. The PRCM acts as a master controller for power management on behalf of all of the peripherals on the device.

The power conservation modes available for each of the three components of the EMAC/MDIO peripheral are:

- *Idle/Disabled state*. This mode stops the clocks going to the peripheral, and prevents all the register accesses. After reenabling the peripheral from this idle state, all the registers values prior to setting into the disabled state are restored, and data transmission can proceed. No reinitialization is required.
- *Synchronized reset.* This state is similar to the Power-on Reset (POR) state, when the processor is turned-on; reset to the peripheral is asserted, and clocks to the peripheral are gated after that. The registers are reset to their default value. When powering-up after a synchronized reset, all the EMAC submodules need to be reinitialized before any data transmission can happen.

For more information on the use of the PRCM, see the PRCM Reference Guide.

6.2.18 Emulation Considerations

NOTE: For correct operation, the EMAC and EMAC control module must both be suspended. Thus, the EMCONTROL and CMEMCONTROL registers must be configured alike.

EMAC emulation control is implemented for compatibility with other peripherals. The SOFT and FREE bits in the emulation control register (EMCONTROL) allow EMAC operation to be suspended. Additionally, emulation control is also implemented in the EMAC control module with the EMAC control module emulation control register (CMEMCONTROL) to allow the EMAC control module activity to be suspended.

When the emulation suspend state is entered, the EMAC stops processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission is completed normally without suspension. For transmission, any complete or partial frame in the transmit cell FIFO is transmitted. For receive, frames that are detected by the EMAC after the suspend state is entered are ignored. No statistics are kept for ignored frames.

Table 6-8 shows how the SOFT and FREE bits affect the operation of the emulation suspend.

Table 6-8. Emulation Control

SOFT	FREE	Description
0	0	Normal operation
1	0	Emulation suspend
X	1	Normal operation

6.3 EMAC/MDIO Registers

Table 6-9 shows the base address offset for the EMAC/MDIO module.

Table 6-9. EMAC/MDIO Registers

Description	Base Address Offset	Section
EMAC control module	0000 0900h	Section 6.3.1
EMAC module	0000 0000h	Section 6.3.2
MDIO module	0000 0800h	Section 6.3.3

6.3.1 EMAC Control Module Registers

Table 6-10 lists the memory-mapped registers for the EMAC control module. For the base address of these registers, see Table 1-13.

Table 6-10. EMAC Control Module Registers

Address Offset	Acronym	Register Description	Section
0h	CMIDVER	Identification and Version Register	Section 6.3.1.1
4h	CMSOFTRESET	Software Reset Register	Section 6.3.1.2
8h	CMEMCONTROL	Emulation Control Register	Section 6.3.1.3
Ch	CMINTCTRL	Interrupt Control Register	Section 6.3.1.4
10h	CMRXTHRESHINTEN	Receive Threshold Interrupt Enable Register	Section 6.3.1.5
14h	CMRXINTEN	Receive Interrupt Enable Register	Section 6.3.1.6
18h	CMTXINTEN	Transmit Interrupt Enable Register	Section 6.3.1.7
1Ch	CMMISCINTEN	Miscellaneous Interrupt Enable Register	Section 6.3.1.8
40h	CMRXTHRESHINTSTAT	Receive Threshold Interrupt Status Register	Section 6.3.1.9
44h	CMRXINTSTAT	Receive Interrupt Status Register	Section 6.3.1.10
48h	CMTXINTSTAT	Transmit Interrupt Status Register	Section 6.3.1.11
4Ch	CMMISCINTSTAT	Miscellaneous Interrupt Status Register	Section 6.3.1.12
70h	CMRXINTMAX	Receive Interrupts Per Millisecond Register	Section 6.3.1.13
74h	CMTXINTMAX	Transmit Interrupts Per Millisecond Register	Section 6.3.1.14

6.3.1.1 EMAC Control Module Identification and Version Register (CMIDVER)

The identification and version register (CMIDVER) is shown in Figure 6-12 and described in Table 6-11.

Figure 6-12. EMAC Control Module Identification and Version Register (CMIDVER)

31	16
CPGMACSS_S_IDENT	
4EC8h	

15	11	10	8	7	0
CPGMACSS_S_RTL_VER		CPGMACSS_S_MAJ_VER		CPGMACSS_S_MINOR_VER	
R-1h		R-1h		R-1h	

LEGEND: R = Read only; -*n* = value after reset

Table 6-11. EMAC Control Module Identification and Version Register (CMIDVER) Field Descriptions

Bit	Field	Value	Description
31-16	CPGMACSS_S_IDENT	0-FFFFh	CPGMACSS_S Identification value.
15-11	CPGMACSS_S_RTL_VER	0-1Fh	CPGMACSS_S RTL Version value.
10-8	CPGMACSS_S_MAJ_VER	0-7h	CPGMACSS_S Major Version Value.
7-0	CPGMACSS_S_MINOR_VER	0-FFh	CPGMACSS_S Minor Version Value.

6.3.1.2 EMAC Control Module Software Reset Register (CMSOFTRESET)

The software reset register (CMSOFTRESET) is shown in Figure 6-13 and described in Table 6-12.

Figure 6-13. EMAC Control Module Software Reset Register (CMSOFTRESET)

31	1	0
Reserved		SOFT_RESET
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-12. EMAC Control Module Software Reset Register (CMSOFTRESET) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	SOFT_RESET		Software reset – Writing a one to this bit causes the CPGMACSS_S logic to be reset (INT, REGS, CPPI). Software reset occurs on the clock following the register bit write.
		0	No software reset.
		1	Software reset occurs.

6.3.1.3 EMAC Control Module Emulation Control Register (CMEMCONTROL)

The emulation control register (CMEMCONTROL) is shown in Figure 6-14 and described in Table 6-13.

Figure 6-14. EMAC Control Module Emulation Control Register (CMEMCONTROL)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		SOFT	FREE
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-13. EMAC Control Module Emulation Control Register (CMEMCONTROL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	SOFT		Emulation soft bit. This bit is used in conjunction with FREE bit to determine the emulation suspend mode. This bit has no effect if FREE = 1.
		0	Soft mode is disabled. EMAC control module stops immediately during emulation halt.
		1	Soft mode is enabled. During emulation halt, EMAC control module stops after completion of current operation.
0	FREE		Emulation free bit. This bit is used in conjunction with SOFT bit to determine the emulation suspend mode.
		0	Free-running mode is disabled. During emulation halt, SOFT bit determines operation of the EMAC control module.
		1	Free-running mode is enabled. During emulation halt, the EMAC control module continues to operate.

6.3.1.4 EMAC Control Module Interrupt Control Register (CMINTCTRL)

The interrupt control register (CMINTCTRL) is shown in Figure 6-15 and described in Table 6-14.

Figure 6-15. EMAC Control Module Interrupt Control Register (CMINTCTRL)

31	30	22	21	16
INT_TEST	Reserved		INT_PACE_EN	
R/W-0	R-0		R/W-0	

15	12	11	0
Reserved		INTPRESCALE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-14. EMAC Control Module Interrupt Control Register (CMINTCTRL) Field Descriptions

Bit	Field	Value	Description
31	INT_TEST	0	Interrupt Test – Test bit to the interrupt pacing blocks.
30-22	Reserved	0	Reserved.
21-16	INT_PACE_EN		Interrupt Pacing Enable Bus.
		1h	C0_RX : Enables C0_Rx_Pulse Pacing
		2h	C0_TX : Enables C0_Tx_Pulse Pacing
		4h	C1_RX : Enables C1_Rx_Pulse Pacing
		8h	C1_TX : Enables C1_Tx_Pulse Pacing
		10h	C2_RX : Enables C2_Rx_Pulse Pacing
		20h	C2_TX : Enables C2_Tx_Pulse Pacing
15-12	Reserved	0	Reserved.
11-0	INTPRESCALE	0-7FFh	Interrupt Counter Prescaler – The number of VBUSP_CLK periods in 4 µs.

6.3.1.5 EMAC Control Module Receive Threshold Interrupt Enable Register (CMRXTHRESHINTEN)

The receive threshold interrupt enable register (CMRXTHRESHINTEN) is shown in Figure 6-16 and described in Table 6-15.

Figure 6-16. EMAC Control Module Receive Threshold Interrupt Enable Register (CMRXTHRESHINTEN)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		C0_RX_THRESH_EN	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-15. EMAC Control Module Receive Threshold Interrupt Enable Register (CMRXTHRESHINTEN) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	C0_RX_THRESH_EN	0-FFh	Core 0 Receive Threshold Enable. Each bit in this register corresponds to the bit in the receive threshold interrupt that is enabled to generate an interrupt on C0_RX_THRESH_PULSE.

6.3.1.6 EMAC Control Module Receive Interrupt Enable Register (CMRXINTEN)

The receive interrupt enable register (CMRXINTEN) is shown in Figure 6-17 and described in Table 6-16.

Figure 6-17. EMAC Control Module Receive Interrupt Enable Register (CMRXINTEN)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		C_RX_EN	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-16. EMAC Control Module Receive Interrupt Enable Register (CMRXINTEN) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	C_RX_EN	0-FFh	Core 0 Receive Enable. Each bit in this register corresponds to the bit in the RX interrupt that is enabled to generate an interrupt on C0_RX_PULSE.

6.3.1.7 EMAC Control Module Transmit Interrupt Enable Register (CMTXINTEN)

The transmit interrupt enable register (CMTXINTEN) is shown in Figure 6-18 and described in Table 6-17.

Figure 6-18. EMAC Control Module Transmit Interrupt Enable Register (CMTXINTEN)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		C_TX_EN	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-17. EMAC Control Module Transmit Interrupt Enable Register (CMTXINTEN) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	C_TX_EN	0-FFh	Core 0 Transmit Enable. Each bit in this register corresponds to the bit in the TX interrupt that is enabled to generate an interrupt on C0_TX_PULSE.

6.3.1.8 EMAC Control Module Miscellaneous Interrupt Enable Register (CMMISCINTEN)

The miscellaneous interrupt enable register (CMMISCINTEN) is shown in Figure 6-19 and described in Table 6-18.

Figure 6-19. EMAC Control Module Miscellaneous Interrupt Enable Register (CMMISCINTEN)

31	16
Reserved	
R-0	

15	4	3	0
Reserved		C_MISC_EN	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

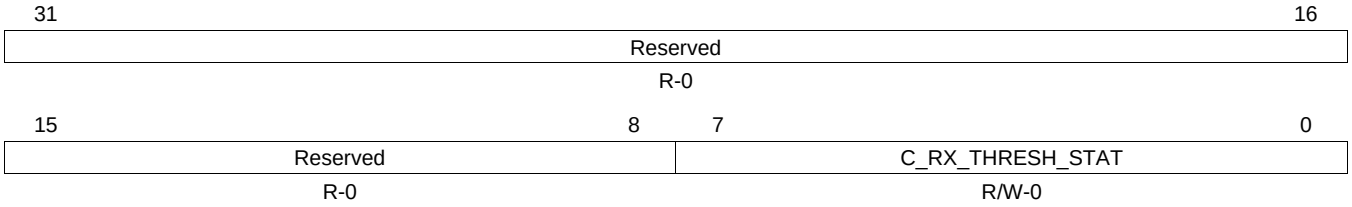
Table 6-18. EMAC Control Module Miscellaneous Interrupt Enable Register (CMMISCINTEN) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3-0	C_MISC_EN	0-Fh	Core 0 Misc Enable Each bit in this register corresponds to the miscellaneous interrupt (STAT_PEND, HOST_PEND, MDIO_LINKINT, MDIO_USERINT) that is enabled to generate an interrupt on C0_Misc_PULSE.

6.3.1.9 EMAC Control Module Receive Threshold Interrupt Status Register (CMRXTHRESHINTSTAT)

The receive threshold interrupt status register (CMRXTHRESHINTSTAT) is shown in Figure 6-20 and described in Table 6-19.

Figure 6-20. EMAC Control Module Receive Threshold Interrupt Status Register (CMRXTHRESHINTSTAT)



LEGEND: R = Read only; -*n* = value after reset

Table 6-19. EMAC Control Module Receive Threshold Interrupt Status Register (CMRXTHRESHINTSTAT) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	C_RX_THRESH_STAT	0-FFh	Core 0 Receive Threshold Masked Interrupt Status. Each bit in this read only register corresponds to the bit in the receive threshold interrupt that is enabled and generating an interrupt on C0_RX_THRESH_PULSE.

6.3.1.10 EMAC Control Module Receive Interrupt Status Register (CMRXINTSTAT)

The receive interrupt status register (CMRXINTSTAT) is shown in Figure 6-21and described in Table 6-20.

Figure 6-21. EMAC Control Module Receive Interrupt Status Register (CMRXINTSTAT)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		C_RX_STAT	
R-0		R/W-0	

LEGEND: R = Read only; -*n* = value after reset

Table 6-20. EMAC Control Module Receive Interrupt Status Register (CMRXINTSTAT) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	C_RX_STAT	0-FFh	Core 0 Receive Masked Interrupt Status. Each bit in this read only register corresponds to the bit in the Rx interrupt that is enabled and generating an interrupt on C0_RX_PULSE.

6.3.1.11 EMAC Control Module Transmit Interrupt Status Register (CMTXINTSTAT)

The transmit interrupt status register (CMTXINTSTAT) is shown in Figure 6-22 and described in Table 6-21.

Figure 6-22. EMAC Control Module Transmit Interrupt Status Register (CMTXINTSTAT)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		C_TX_STAT	
R-0		R/W-0	

LEGEND: R = Read only; -*n* = value after reset

Table 6-21. EMAC Control Module Transmit Interrupt Status Register (CMTXINTSTAT) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	C_TX_STAT	0-FFh	Core 0 Transmit Masked Interrupt Status. Each bit in this read only register corresponds to the bit in the Tx interrupt that is enabled and generating an interrupt on C0_TX_PULSE.

6.3.1.12 EMAC Control Module Miscellaneous Interrupt Status Register (CMMISCINTSTAT)

The miscellaneous interrupt status register (CMMISCINTSTAT) is shown in Figure 6-23 and described in Table 6-22.

Figure 6-23. EMAC Control Module Miscellaneous Interrupt Status Register (CMMISCINTSTAT)

31	16
Reserved	
R-0	

15	4	3	0
Reserved		C_MISC_STAT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-22. EMAC Control Module Miscellaneous Interrupt Status Register (CMMISCINTSTAT) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3-0	C_MISC_STAT	0-Fh	Core 0 Misc Masked Interrupt Status. Each bit in this register corresponds to the miscellaneous interrupt (STAT_PEND, HOST_PEND, MDIO_LINKINT, MDIO_USERINT) that is enabled and generating an interrupt on C0_MISC_PULSE.

6.3.1.13 EMAC Control Module Receive Interrupts per Millisecond Register (CMRXINTMAX)

The receive interrupts per millisecond register (CMRXINTMAX) is shown in Figure 6-24and described in Table 6-23.

Figure 6-24. EMAC Control Module Receive Interrupts per Millisecond Register (CMRXINTMAX)

31	16
Reserved	
R-0	

15	6	5	0
Reserved			
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-23. EMAC Control Module Receive Interrupts per Millisecond Register (CMRXINTMAX) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5-0	RXIMAX	0-3Fh	Core 0 Receive Interrupts per Millisecond. The maximum number of interrupts per millisecond generated on C0_RX_PULSE if pacing is enabled for this interrupt.

6.3.1.14 EMAC Control Module Transmit Interrupts per Millisecond Register (CMTXINTMAX)

The transmit interrupts per millisecond register (CMTXINTMAX) is shown in Figure 6-25and described in Table 6-24.

Figure 6-25. EMAC Control Module Transmit Interrupts per Millisecond Register (CMTXINTMAX)

31	16
Reserved	
R-0	

15	6	5	0
Reserved		C_TX_IMAX	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-24. EMAC Control Module Transmit Interrupts per Millisecond Register (CMTXINTMAX) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5-0	C_TX_IMAX	0-3Fh	Core 0 Transmit Interrupts per Millisecond. The maximum number of interrupts per millisecond generated on C0_TX_PULSE if pacing is enabled for this interrupt.

6.3.2 Ethernet Media Access Controller (EMAC) Registers

Table 6-25 lists the memory-mapped registers for the EMAC. For the base address of these registers, see Table 1-13.

Table 6-25. Ethernet Media Access Controller (EMAC) Registers

Address Offset	Acronym	Register Description	Section
0h	CPGMACIDVER	Identification and Version Register	Section 6.3.2.1
4h	TXCONTROL	Transmit Control Register	Section 6.3.2.2
8h	TXTEARDOWN	Transmit Teardown Register	Section 6.3.2.3
14h	RXCONTROL	Receive Control Register	Section 6.3.2.4
18h	RXTEARDOWN	Receive Teardown Register	Section 6.3.2.5
80h	TXINTSTATRAW	Transmit Interrupt Status (Unmasked) Register	Section 6.3.2.6
84h	TXINTSTATMASKED	Transmit Interrupt Status (Masked) Register	Section 6.3.2.7
88h	TXINTMASKSET	Transmit Interrupt Mask Set Register	Section 6.3.2.8
8Ch	TXINTMASKCLEAR	Transmit Interrupt Clear Register	Section 6.3.2.9
90h	MACINVECTOR	MAC Input Vector Register	Section 6.3.2.10
94h	MACEOIVECTOR	MAC End of Interrupt Vector Register	Section 6.3.2.11
A0h	RXINTSTATRAW	Receive Interrupt Status (Unmasked) Register	Section 6.3.2.12
A4h	RXINTSTATMASKED	Receive Interrupt Status (Masked) Register	Section 6.3.2.13
A8h	RXINTMASKSET	Receive Interrupt Mask Set Register	Section 6.3.2.14
ACh	RXINTMASKCLEAR	Receive Interrupt Mask Clear Register	Section 6.3.2.15
B0h	MACINTSTATRAW	MAC Interrupt Status (Unmasked) Register	Section 6.3.2.16
B4h	MACINTSTATMASKED	MAC Interrupt Status (Masked) Register	Section 6.3.2.17
B8h	MACINTMASKSET	MAC Interrupt Mask Set Register	Section 6.3.2.18
BCh	MACINTMASKCLEAR	MAC Interrupt Mask Clear Register	Section 6.3.2.19
100h	RXMBPENABLE	Receive Multicast/Broadcast/Promiscuous Channel Enable Register	Section 6.3.2.20
104h	RXUNICASTSET	Receive Unicast Enable Set Register	Section 6.3.2.21
108h	RXUNICASTCLEAR	Receive Unicast Clear Register	Section 6.3.2.22
10Ch	RXMAXLEN	Receive Maximum Length Register	Section 6.3.2.23
110h	RXBUFFEROFFSET	Receive Buffer Offset Register	Section 6.3.2.24
114h	RXFILTERLOWTHRESH	Receive Filter Low Priority Frame Threshold Register	Section 6.3.2.25
120h	RX0FLOWTHRESH	Receive Channel 0 Flow Control Threshold Register	Section 6.3.2.26
124h	RX1FLOWTHRESH	Receive Channel 1 Flow Control Threshold Register	Section 6.3.2.26
128h	RX2FLOWTHRESH	Receive Channel 2 Flow Control Threshold Register	Section 6.3.2.26
12Ch	RX3FLOWTHRESH	Receive Channel 3 Flow Control Threshold Register	Section 6.3.2.26
130h	RX4FLOWTHRESH	Receive Channel 4 Flow Control Threshold Register	Section 6.3.2.26
134h	RX5FLOWTHRESH	Receive Channel 5 Flow Control Threshold Register	Section 6.3.2.26
138h	RX6FLOWTHRESH	Receive Channel 6 Flow Control Threshold Register	Section 6.3.2.26
13Ch	RX7FLOWTHRESH	Receive Channel 7 Flow Control Threshold Register	Section 6.3.2.26
140h	RX0FREEBUFFER	Receive Channel 0 Free Buffer Count Register	Section 6.3.2.27
144h	RX1FREEBUFFER	Receive Channel 1 Free Buffer Count Register	Section 6.3.2.27
148h	RX2FREEBUFFER	Receive Channel 2 Free Buffer Count Register	Section 6.3.2.27
14Ch	RX3FREEBUFFER	Receive Channel 3 Free Buffer Count Register	Section 6.3.2.27
150h	RX4FREEBUFFER	Receive Channel 4 Free Buffer Count Register	Section 6.3.2.27
154h	RX5FREEBUFFER	Receive Channel 5 Free Buffer Count Register	Section 6.3.2.27
158h	RX6FREEBUFFER	Receive Channel 6 Free Buffer Count Register	Section 6.3.2.27
15Ch	RX7FREEBUFFER	Receive Channel 7 Free Buffer Count Register	Section 6.3.2.27
160h	MACCONTROL	MAC Control Register	Section 6.3.2.28

Table 6-25. Ethernet Media Access Controller (EMAC) Registers (continued)

Address Offset	Acronym	Register Description	Section
164h	MACSTATUS	MAC Status Register	Section 6.3.2.29
168h	EMCONTROL	Emulation Control Register	Section 6.3.2.30
16Ch	FIFOCONTROL	FIFO Control Register	Section 6.3.2.31
170h	MACCONFIG	MAC Configuration Register	Section 6.3.2.32
174h	SOFTRESET	Soft Reset Register	Section 6.3.2.33
1D0h	MACSRCADDRLO	MAC Source Address Low Bytes Register	Section 6.3.2.34
1D4h	MACSRCADDRHI	MAC Source Address High Bytes Register	Section 6.3.2.35
1D8h	MACHASH1	MAC Hash Address Register 1	Section 6.3.2.36
1DCh	MACHASH2	MAC Hash Address Register 2	Section 6.3.2.37
1E0h	BOFFTEST	Back Off Test Register	Section 6.3.2.38
1E4h	TPACETEST	Transmit Pacing Algorithm Test Register	Section 6.3.2.39
1E8h	RXPAUSE	Receive Pause Timer Register	Section 6.3.2.40
1ECh	TXPAUSE	Transmit Pause Timer Register	Section 6.3.2.41
500h	MACADDRLO	MAC Address Low Bytes Register, Used in Receive Address Matching	Section 6.3.2.42
504h	MACADDRHI	MAC Address High Bytes Register, Used in Receive Address Matching	Section 6.3.2.43
508h	MACINDEX	MAC Index Register	Section 6.3.2.44
600h	TX0HDP	Transmit Channel 0 DMA Head Descriptor Pointer Register	Section 6.3.2.45
604h	TX1HDP	Transmit Channel 1 DMA Head Descriptor Pointer Register	Section 6.3.2.45
608h	TX2HDP	Transmit Channel 2 DMA Head Descriptor Pointer Register	Section 6.3.2.45
60Ch	TX3HDP	Transmit Channel 3 DMA Head Descriptor Pointer Register	Section 6.3.2.45
610h	TX4HDP	Transmit Channel 4 DMA Head Descriptor Pointer Register	Section 6.3.2.45
614h	TX5HDP	Transmit Channel 5 DMA Head Descriptor Pointer Register	Section 6.3.2.45
618h	TX6HDP	Transmit Channel 6 DMA Head Descriptor Pointer Register	Section 6.3.2.45
61Ch	TX7HDP	Transmit Channel 7 DMA Head Descriptor Pointer Register	Section 6.3.2.45
620h	RX0HDP	Receive Channel 0 DMA Head Descriptor Pointer Register	Section 6.3.2.46
624h	RX1HDP	Receive Channel 1 DMA Head Descriptor Pointer Register	Section 6.3.2.46
628h	RX2HDP	Receive Channel 2 DMA Head Descriptor Pointer Register	Section 6.3.2.46
62Ch	RX3HDP	Receive Channel 3 DMA Head Descriptor Pointer Register	Section 6.3.2.46
630h	RX4HDP	Receive Channel 4 DMA Head Descriptor Pointer Register	Section 6.3.2.46
634h	RX5HDP	Receive Channel 5 DMA Head Descriptor Pointer Register	Section 6.3.2.46
638h	RX6HDP	Receive Channel 6 DMA Head Descriptor Pointer Register	Section 6.3.2.46
63Ch	RX7HDP	Receive Channel 7 DMA Head Descriptor Pointer Register	Section 6.3.2.46
640h	TX0CP	Transmit Channel 0 Completion Pointer Register	Section 6.3.2.47
644h	TX1CP	Transmit Channel 1 Completion Pointer Register	Section 6.3.2.47
648h	TX2CP	Transmit Channel 2 Completion Pointer Register	Section 6.3.2.47
64Ch	TX3CP	Transmit Channel 3 Completion Pointer Register	Section 6.3.2.47
650h	TX4CP	Transmit Channel 4 Completion Pointer Register	Section 6.3.2.47
654h	TX5CP	Transmit Channel 5 Completion Pointer Register	Section 6.3.2.47
658h	TX6CP	Transmit Channel 6 Completion Pointer Register	Section 6.3.2.47
65Ch	TX7CP	Transmit Channel 7 Completion Pointer Register	Section 6.3.2.47
660h	RX0CP	Receive Channel 0 Completion Pointer Register	Section 6.3.2.48
664h	RX1CP	Receive Channel 1 Completion Pointer Register	Section 6.3.2.48
668h	RX2CP	Receive Channel 2 Completion Pointer Register	Section 6.3.2.48
66Ch	RX3CP	Receive Channel 3 Completion Pointer Register	Section 6.3.2.48
670h	RX4CP	Receive Channel 4 Completion Pointer Register	Section 6.3.2.48

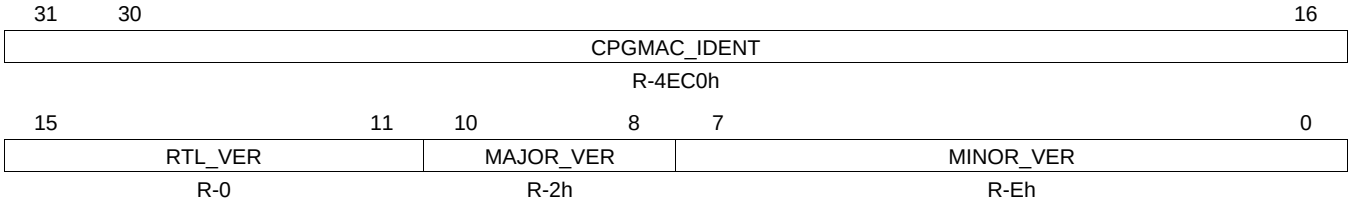
Table 6-25. Ethernet Media Access Controller (EMAC) Registers (continued)

Address Offset	Acronym	Register Description	Section
674h	RX5CP	Receive Channel 5 Completion Pointer Register	Section 6.3.2.48
678h	RX6CP	Receive Channel 6 Completion Pointer Register	Section 6.3.2.48
67Ch	RX7CP	Receive Channel 7 Completion Pointer Register	Section 6.3.2.48
		Network Statistics Registers	
200h	RXGOODFRAMES	Good Receive Frames Register	Section 6.3.2.49.1
204h	RXBCASTFRAMES	Broadcast Receive Frames Register	Section 6.3.2.49.2
208h	RXMCASTFRAMES	Multicast Receive Frames Register	Section 6.3.2.49.3
20Ch	RXPAUSEFRAMES	Pause Receive Frames Register	Section 6.3.2.49.4
210h	RXCRCERRORS	Receive CRC Errors Register	Section 6.3.2.49.5
214h	RXALIGNCODEERRORS	Receive Alignment/Code Errors Register	Section 6.3.2.49.6
218h	RXOVERSIZED	Receive Oversized Frames Register	Section 6.3.2.49.7
21Ch	RXJABBER	Receive Jabber Frames Register	Section 6.3.2.49.8
220h	RXUNDERSIZED	Receive Undersized Frames Register	Section 6.3.2.49.9
224h	RXFRAGMENTS	Receive Frame Fragments Register	Section 6.3.2.49.10
228h	RXFILTERED	Filtered Receive Frames Register	Section 6.3.2.49.11
22Ch	RXQOSFILTERED	Receive QOS Filtered Frames Register	Section 6.3.2.49.12
230h	RXOCTETS	Receive Octet Frames Register	Section 6.3.2.49.13
234h	TXGOODFRAMES	Good Transmit Frames Register	Section 6.3.2.49.14
238h	TXBCASTFRAMES	Broadcast Transmit Frames Register	Section 6.3.2.49.15
23Ch	TXMCASTFRAMES	Multicast Transmit Frames Register	Section 6.3.2.49.16
240h	TXPAUSEFRAMES	Pause Transmit Frames Register	Section 6.3.2.49.17
244h	TXDEFERRED	Deferred Transmit Frames Register	Section 6.3.2.49.18
248h	TXCOLLISION	Transmit Collision Frames Register	Section 6.3.2.49.19
24Ch	TXSINGLECOLL	Transmit Single Collision Frames Register	Section 6.3.2.49.20
250h	TXMULTICOLL	Transmit Multiple Collision Frames Register	Section 6.3.2.49.21
254h	TXEXCESSIVECOLL	Transmit Excessive Collision Frames Register	Section 6.3.2.49.22
258h	TXLATECOLL	Transmit Late Collision Frames Register	Section 6.3.2.49.23
25Ch	TXUNDERRUN	Transmit Underrun Error Register	Section 6.3.2.49.24
260h	TXCARRIERSENSE	Transmit Carrier Sense Errors Register	Section 6.3.2.49.25
264h	TXOCTETS	Transmit Octet Frames Register	Section 6.3.2.49.26
268h	FRAME64	Transmit and Receive 64 Octet Frames Register	Section 6.3.2.49.27
26Ch	FRAME65T127	Transmit and Receive 65 to 127 Octet Frames Register	Section 6.3.2.49.28
270h	FRAME128T255	Transmit and Receive 128 to 255 Octet Frames Register	Section 6.3.2.49.29
274h	FRAME256T511	Transmit and Receive 256 to 511 Octet Frames Register	Section 6.3.2.49.30
278h	FRAME512T1023	Transmit and Receive 512 to 1023 Octet Frames Register	Section 6.3.2.49.31
27Ch	FRAME1024TUP	Transmit and Receive 1024 to RXMAXLEN Octet Frames Register	Section 6.3.2.49.32
280h	NETOCTETS	Network Octet Frames Register	Section 6.3.2.49.33
284h	RXSOFOVERRUNS	Receive FIFO or DMA Start of Frame Overruns Register	Section 6.3.2.49.34
288h	RXMOFOVERRUNS	Receive FIFO or DMA Middle of Frame Overruns Register	Section 6.3.2.49.35
28Ch	RXDMAOVERRUNS	Receive DMA Overruns Register	Section 6.3.2.49.36

6.3.2.1 Identification and Version Register (CPGMACIDVER)

The identification and version register (CPGMACIDVER) is shown in Figure 6-26 and described in Table 6-26.

Figure 6-26. Identification and Version Register (CPGMACIDVER)



LEGEND: R = Read only; -*n* = value after reset

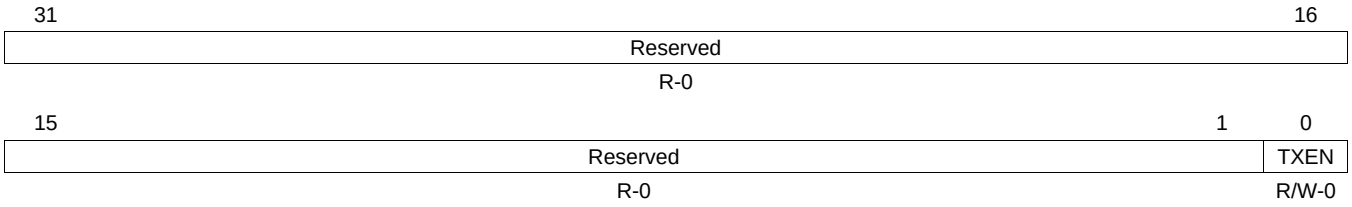
Table 6-26. Identification and Version Register (CPGMACIDVER) Field Descriptions

Bit	Field	Value	Description
31-16	CPGMAC_IDENT		Identification value.
		4EC0h	Current identification value.
15-11	RTL_VER		RTL version value.
		0	Current RTL version value.
10-8	MAJOR_VER		Major version value. Revisions are indicated by a revision code taking the format MAJOR_VER.MINOR_VER.
		2h	Current major version value.
7-0	MINOR_VER		Minor version value. Revisions are indicated by a revision code taking the format MAJOR_VER.MINOR_VER.
		Eh	Current minor version value.

6.3.2.2 Transmit Control Register (TXCONTROL)

The transmit control register (TXCONTROL) is shown in Figure 6-27 and described in Table 6-27.

Figure 6-27. Transmit Control Register (TXCONTROL)



LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 6-27. Transmit Control Register (TXCONTROL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	TXEN		Transmit enable
		0	Transmit is disabled.
		1	Transmit is enabled.

6.3.2.3 Transmit Teardown Register (TXTEARDOWN)

The transmit teardown register (TXTEARDOWN) is shown in Figure 6-28 and described in Table 6-28.

Figure 6-28. Transmit Teardown Register (TXTEARDOWN)

31	16
Reserved	
R-0	

15	3	2	0
Reserved		TXTDNCH	
R-0		R/W-0	

LEGEND: R = Read only; R/W = Read/Write; *-n* = value after reset

Table 6-28. Transmit Teardown Register (TXTEARDOWN) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	TXTDNCH	0-7h	Transmit teardown channel. The transmit channel teardown is commanded by writing the encoded value of the transmit channel to be torn down. The teardown register is read as 0.
		0	Teardown transmit channel 0
		1h	Teardown transmit channel 1
		2h	Teardown transmit channel 2
		3h	Teardown transmit channel 3
		4h	Teardown transmit channel 4
		5h	Teardown transmit channel 5
		6h	Teardown transmit channel 6
		7h	Teardown transmit channel 7

6.3.2.4 Receive Control Register (RXCONTROL)

The receive control register (RXCONTROL) is shown in Figure 6-29 and described in Table 6-29.

Figure 6-29. Receive Control Register (RXCONTROL)

31	16
Reserved	
R-0	

15	1	0
Reserved		RXEN
R-0		R/W-0

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 6-29. Receive Control Register (RXCONTROL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	RXEN		Receive enable
		0	Receive is disabled.
		1	Receive is enabled.

6.3.2.5 Receive Teardown Register (RXTEARDOWN)

The receive teardown register (RXTEARDOWN) is shown in Figure 6-30 and described in Table 6-30.

Figure 6-30. Receive Teardown Register (RXTEARDOWN)

31	16
Reserved	
R-0	

15	3	2	0
Reserved		RXTDNCH	
R-0		R/W-0	

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

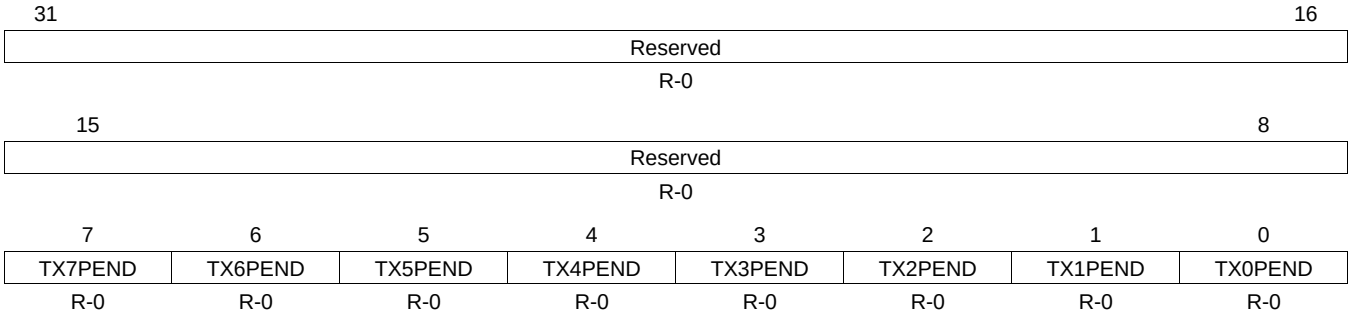
Table 6-30. Receive Teardown Register (RXTEARDOWN) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	RXTDNCH	0-7h	Receive teardown channel. The receive channel teardown is commanded by writing the encoded value of the receive channel to be torn down. The teardown register is read as 0.
		0	Teardown receive channel 0
		1h	Teardown receive channel 1
		2h	Teardown receive channel 2
		3h	Teardown receive channel 3
		4h	Teardown receive channel 4
		5h	Teardown receive channel 5
		6h	Teardown receive channel 6
		7h	Teardown receive channel 7

6.3.2.6 Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW)

The transmit interrupt status (unmasked) register (TXINTSTATRAW) is shown in Figure 6-31 and described in Table 6-31.

Figure 6-31. Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW)



LEGEND: R = Read only; -*n* = value after reset

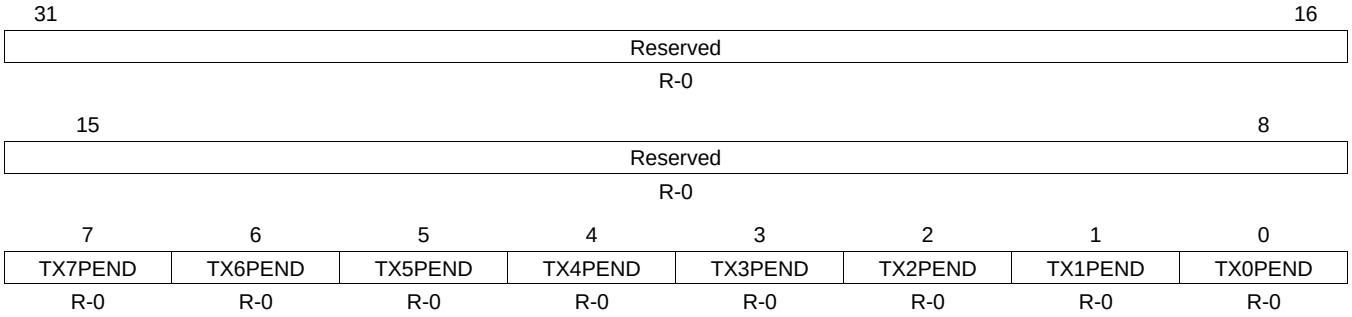
Table 6-31. Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TX7PEND	0-1	TX7PEND raw interrupt read (before mask)
6	TX6PEND	0-1	TX6PEND raw interrupt read (before mask)
5	TX5PEND	0-1	TX5PEND raw interrupt read (before mask)
4	TX4PEND	0-1	TX4PEND raw interrupt read (before mask)
3	TX3PEND	0-1	TX3PEND raw interrupt read (before mask)
2	TX2PEND	0-1	TX2PEND raw interrupt read (before mask)
1	TX1PEND	0-1	TX1PEND raw interrupt read (before mask)
0	TX0PEND	0-1	TX0PEND raw interrupt read (before mask)

6.3.2.7 Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED)

The transmit interrupt status (masked) register (TXINTSTATMASKED) is shown in Figure 6-32 and described in Table 6-32.

Figure 6-32. Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED)



LEGEND: R = Read only; -*n* = value after reset

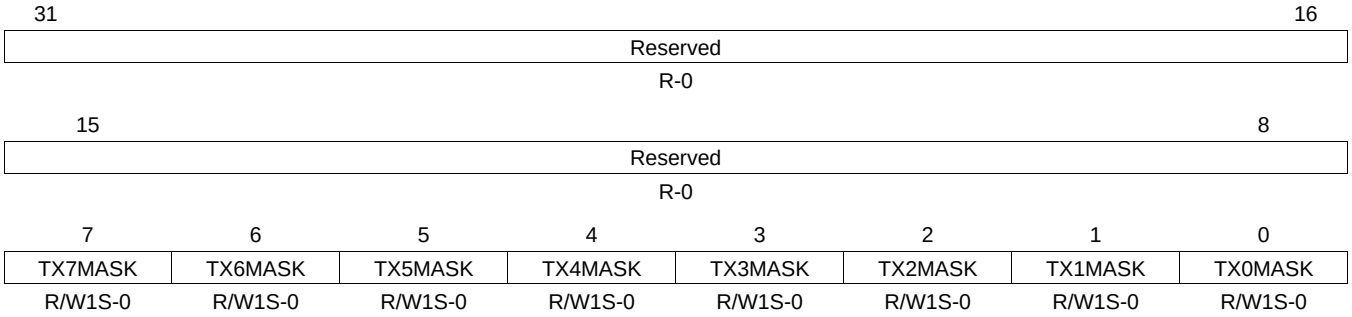
Table 6-32. Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TX7PEND	0-1	TX7PEND masked interrupt read
6	TX6PEND	0-1	TX6PEND masked interrupt read
5	TX5PEND	0-1	TX5PEND masked interrupt read
4	TX4PEND	0-1	TX4PEND masked interrupt read
3	TX3PEND	0-1	TX3PEND masked interrupt read
2	TX2PEND	0-1	TX2PEND masked interrupt read
1	TX1PEND	0-1	TX1PEND masked interrupt read
0	TX0PEND	0-1	TX0PEND masked interrupt read

6.3.2.8 Transmit Interrupt Mask Set Register (TXINTMASKSET)

The transmit interrupt mask set register (TXINTMASKSET) is shown in Figure 6-33 and described in Table 6-33.

Figure 6-33. Transmit Interrupt Mask Set Register (TXINTMASKSET)



LEGEND: R = Read only; R/W = Read/Write; W1S = Write 1 to set, write of 0 has no effect; -*n* = value after reset

Table 6-33. Transmit Interrupt Mask Set Register (TXINTMASKSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TX7MASK	0-1	Transmit channel 7 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
6	TX6MASK	0-1	Transmit channel 6 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
5	TX5MASK	0-1	Transmit channel 5 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
4	TX4MASK	0-1	Transmit channel 4 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
3	TX3MASK	0-1	Transmit channel 3 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
2	TX2MASK	0-1	Transmit channel 2 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
1	TX1MASK	0-1	Transmit channel 1 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
0	TX0MASK	0-1	Transmit channel 0 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.

6.3.2.9 Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR)

The transmit interrupt mask clear register (TXINTMASKCLEAR) is shown in Figure 6-34 and described in Table 6-34.

Figure 6-34. Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
TX7MASK	TX6MASK	TX5MASK	TX4MASK	TX3MASK	TX2MASK	TX1MASK	TX0MASK
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R = Read only; R/W = Read/Write; W1C = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

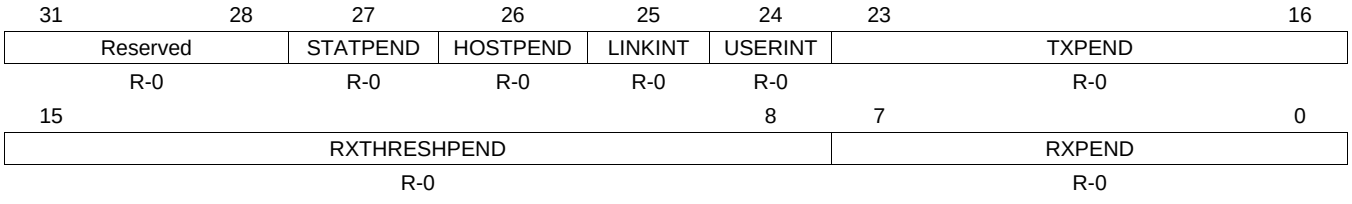
Table 6-34. Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TX7MASK	0-1	Transmit channel 7 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
6	TX6MASK	0-1	Transmit channel 6 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
5	TX5MASK	0-1	Transmit channel 5 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
4	TX4MASK	0-1	Transmit channel 4 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
3	TX3MASK	0-1	Transmit channel 3 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
2	TX2MASK	0-1	Transmit channel 2 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
1	TX1MASK	0-1	Transmit channel 1 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
0	TX0MASK	0-1	Transmit channel 0 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.

6.3.2.10 MAC Input Vector Register (MACINVECTOR)

The MAC input vector register (MACINVECTOR) is shown in Figure 6-35 and described in Table 6-35.

Figure 6-35. MAC Input Vector Register (MACINVECTOR)



LEGEND: R = Read only; -*n* = value after reset

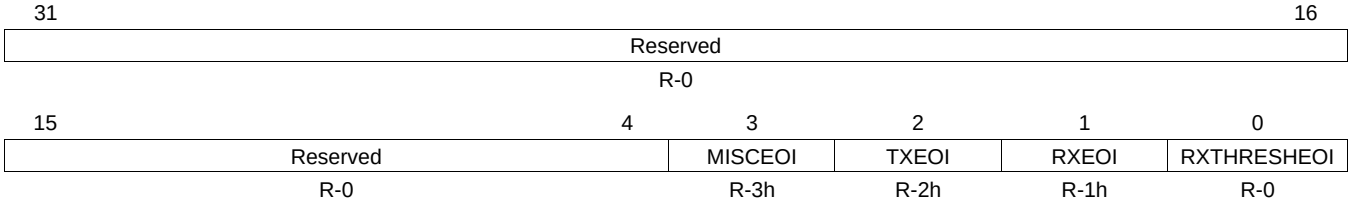
Table 6-35. MAC Input Vector Register (MACINVECTOR) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27	STATPEND	0-1	EMAC module statistics interrupt (STATPEND) pending status bit.
26	HOSTPEND	0-1	EMAC module host error interrupt (HOSTPEND) pending status bit.
25	LINKINT	0-1	MDIO module link change interrupt (LINKINT) pending status bit.
24	USERINT	0-1	MDIO module user interrupt (USERINT) pending status bit.
23-16	TXPEND	0-FFh	Transmit channels 0-7 interrupt pending (TXPEND*n*) status bit. Bit 16 is transmit channel 0.
15-8	RXTHRESHPEND	0-FFh	Receive threshold channels 0-7 interrupt pending (RXTHRESHPEND*n*) status bit. Bit 8 is receive channel 0.
7-0	RXPEND	0-FFh	Receive channels 0-7 interrupt pending (RXPEND*n*) status bit. Bit 0 is receive channel 0.

6.3.2.11 MAC End Of Interrupt Vector Register (MACEOIVECTOR)

The MAC end of interrupt vector register (MACEOIVECTOR) is shown in Figure 6-36 and described in Table 6-36.

Figure 6-36. MAC End Of Interrupt Vector Register (MACEOIVECTOR)



LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

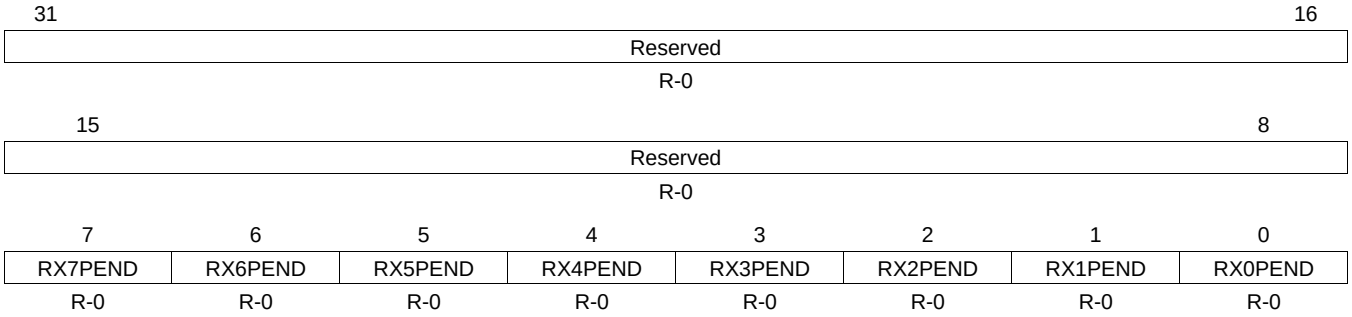
Table 6-36. MAC End Of Interrupt Vector Register (MACEOIVECTOR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	MISCEOI	3h	End of interrupt processing for Miscellaneous interrupt.
2	TXEOI	2h	End of interrupt processing for TXPULSE interrupt.
1	RXEOI	1h	End of interrupt processing for RXPULSE interrupt.
0	RXTHRESHEOI	0	End of interrupt processing for RXTHRESH interrupt.

6.3.2.12 Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW)

The receive interrupt status (unmasked) register (RXINTSTATRAW) is shown in Figure 6-37 and described in Table 6-37.

Figure 6-37. Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW)



LEGEND: R = Read only; -*n* = value after reset

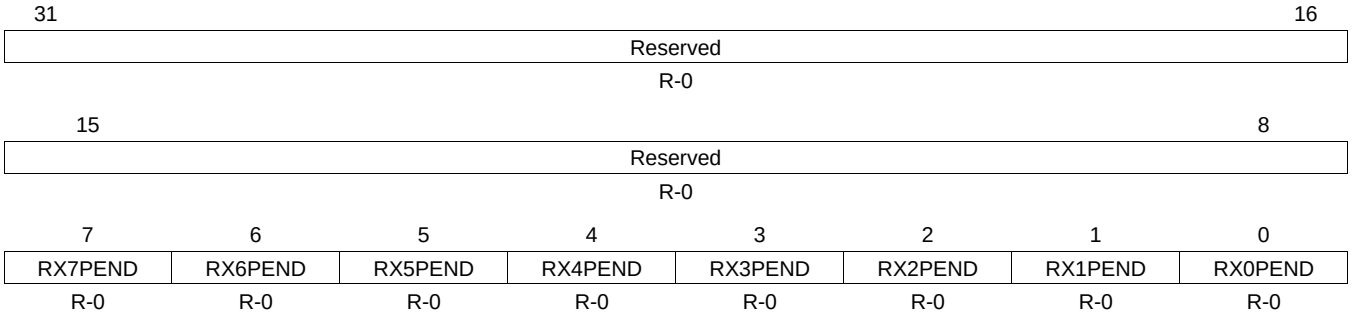
Table 6-37. Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RX7PEND	0-1	RX7PEND raw interrupt read (before mask)
6	RX6PEND	0-1	RX6PEND raw interrupt read (before mask)
5	RX5PEND	0-1	RX5PEND raw interrupt read (before mask)
4	RX4PEND	0-1	RX4PEND raw interrupt read (before mask)
3	RX3PEND	0-1	RX3PEND raw interrupt read (before mask)
2	RX2PEND	0-1	RX2PEND raw interrupt read (before mask)
1	RX1PEND	0-1	RX1PEND raw interrupt read (before mask)
0	RX0PEND	0-1	RX0PEND raw interrupt read (before mask)

6.3.2.13 Receive Interrupt Status (Masked) Register (RXINTSTATMASKED)

The receive interrupt status (masked) register (RXINTSTATMASKED) is shown in Figure 6-38 and described in Table 6-38.

Figure 6-38. Receive Interrupt Status (Masked) Register (RXINTSTATMASKED)



LEGEND: R = Read only; -*n* = value after reset

Table 6-38. Receive Interrupt Status (Masked) Register (RXINTSTATMASKED) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RX7PEND	0-1	RX7PEND masked interrupt read
6	RX6PEND	0-1	RX6PEND masked interrupt read
5	RX5PEND	0-1	RX5PEND masked interrupt read
4	RX4PEND	0-1	RX4PEND masked interrupt read
3	RX3PEND	0-1	RX3PEND masked interrupt read
2	RX2PEND	0-1	RX2PEND masked interrupt read
1	RX1PEND	0-1	RX1PEND masked interrupt read
0	RX0PEND	0-1	RX0PEND masked interrupt read

6.3.2.14 Receive Interrupt Mask Set Register (RXINTMASKSET)

The receive interrupt mask set register (RXINTMASKSET) is shown in Figure 6-39 and described in Table 6-39.

Figure 6-39. Receive Interrupt Mask Set Register (RXINTMASKSET)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
RX7MASK	RX6MASK	RX5MASK	RX4MASK	RX3MASK	RX2MASK	RX1MASK	RX0MASK
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R = Read only; R/W = Read/Write; W1S = Write 1 to set, write of 0 has no effect; -*n* = value after reset

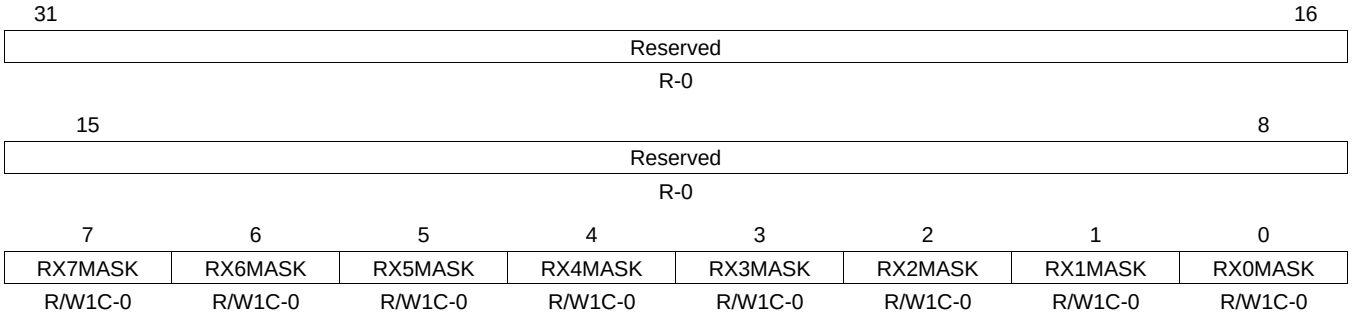
Table 6-39. Receive Interrupt Mask Set Register (RXINTMASKSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RX7MASK	0-1	Receive channel 7 mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
6	RX6MASK	0-1	Receive channel 6 mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
5	RX5MASK	0-1	Receive channel 5 mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
4	RX4MASK	0-1	Receive channel 4 mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
3	RX3MASK	0-1	Receive channel 3 mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
2	RX2MASK	0-1	Receive channel 2 mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
1	RX1MASK	0-1	Receive channel 1 mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
0	RX0MASK	0-1	Receive channel 0 mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.

6.3.2.15 Receive Interrupt Mask Clear Register (RXINTMASKCLEAR)

The receive interrupt mask clear register (RXINTMASKCLEAR) is shown in Figure 6-40 and described in Table 6-40.

Figure 6-40. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR)



LEGEND: R = Read only; R/W = Read/Write; W1C = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

Table 6-40. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RX7MASK	0-1	Receive channel 7 mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
6	RX6MASK	0-1	Receive channel 6 mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
5	RX5MASK	0-1	Receive channel 5 mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
4	RX4MASK	0-1	Receive channel 4 mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
3	RX3MASK	0-1	Receive channel 3 mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
2	RX2MASK	0-1	Receive channel 2 mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
1	RX1MASK	0-1	Receive channel 1 mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
0	RX0MASK	0-1	Receive channel 0 mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.

6.3.2.16 MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW)

The MAC interrupt status (unmasked) register (MACINTSTATRAW) is shown in Figure 6-41 and described in Table 6-41.

Figure 6-41. MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		HOSTPEND	STATPEND
R-0		R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 6-41. MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTPEND	0-1	Host pending interrupt (HOSTPEND); raw interrupt read (before mask)
0	STATPEND	0-1	Statistics pending interrupt (STATPEND); raw interrupt read (before mask)

6.3.2.17 MAC Interrupt Status (Masked) Register (MACINTSTATMASKED)

The MAC interrupt status (masked) register (MACINTSTATMASKED) is shown in Figure 6-42 and described in Table 6-42.

Figure 6-42. MAC Interrupt Status (Masked) Register (MACINTSTATMASKED)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		HOSTPEND	STATPEND
R-0		R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

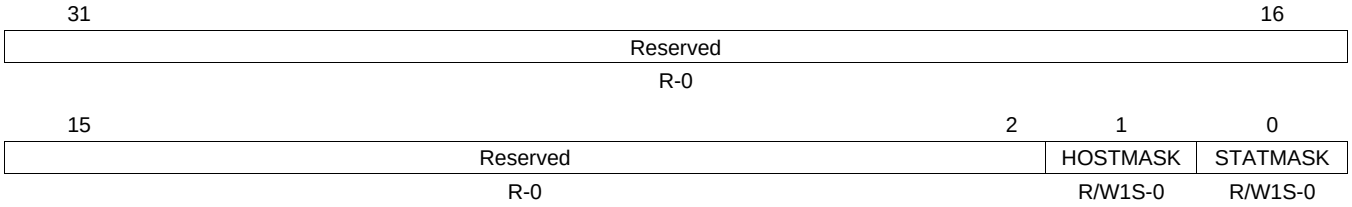
Table 6-42. MAC Interrupt Status (Masked) Register (MACINTSTATMASKED) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTPEND	0-1	Host pending interrupt (HOSTPEND); masked interrupt read
0	STATPEND	0-1	Statistics pending interrupt (STATPEND); masked interrupt read

6.3.2.18 MAC Interrupt Mask Set Register (MACINTMASKSET)

The MAC interrupt mask set register (MACINTMASKSET) is shown in Figure 6-43 and described in Table 6-43.

Figure 6-43. MAC Interrupt Mask Set Register (MACINTMASKSET)



LEGEND: R = Read only; R/W = Read/Write; W1S = Write 1 to set, write of 0 has no effect; -*n* = value after reset

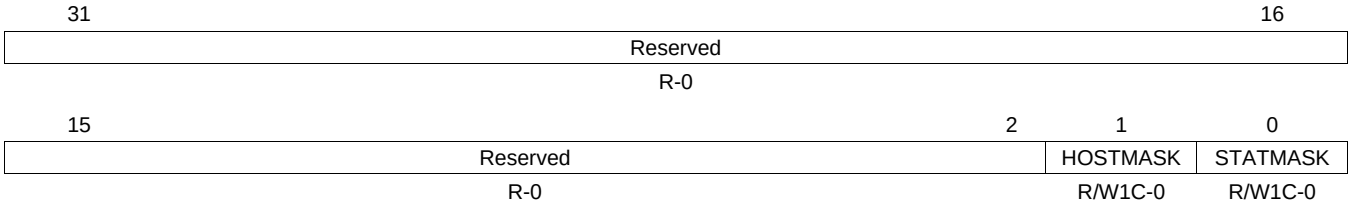
Table 6-43. MAC Interrupt Mask Set Register (MACINTMASKSET) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTMASK	0-1	Host error interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
0	STATMASK	0-1	Statistics interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.

6.3.2.19 MAC Interrupt Mask Clear Register (MACINTMASKCLEAR)

The MAC interrupt mask clear register (MACINTMASKCLEAR) is shown in Figure 6-44 and described in Table 6-44.

Figure 6-44. MAC Interrupt Mask Clear Register (MACINTMASKCLEAR)



LEGEND: R = Read only; R/W = Read/Write; W1C = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

Table 6-44. MAC Interrupt Mask Clear Register (MACINTMASKCLEAR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTMASK	0-1	Host error interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
0	STATMASK	0-1	Statistics interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.

6.3.2.20 Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE)

The receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) is shown in Figure 6-45 and described in Table 6-45.

Figure 6-45. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE)

31	30	29	28	27	25	24
Reserved	RXPASSCRC	RXQOSEN	RXNOCHAIN	Reserved		RXCMFEN
R-0	R/W-0	R/W-0	R/W-0	R-0		R/W-0

23	22	21	20	19	18	16
RXCSFEN	RXCEFEN	RXCAFEN	Reserved		RXPROMCH	
R/W-0	R/W-0	R/W-0	R-0		R/W-0	

15	14	13	12	11	10	8
Reserved		RXBROADEN	Reserved		RXBROADCH	
R-0		R/W-0	R-0		R/W-0	

7	6	5	4	3	2	0
Reserved		RXMULTEN	Reserved		RXMULTCH	
R-0		R/W-0	R-0		R/W-0	

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 6-45. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30	RXPASSCRC		Pass receive CRC enable bit
		0	Received CRC is discarded for all channels and is not included in the buffer descriptor packet length field.
		1	Received CRC is transferred to memory for all channels and is included in the buffer descriptor packet length.
29	RXQOSEN		Receive quality of service enable bit
		0	Receive QOS is disabled.
		1	Receive QOS is enabled.
28	RXNOCHAIN		Receive no buffer chaining bit
		0	Received frames can span multiple buffers.
		1	The Receive DMA controller transfers each frame into a single buffer, regardless of the frame or buffer size. All remaining frame data after the first buffer is discarded. The buffer descriptor buffer length field will contain the entire frame byte count (up to 65535 bytes).
27-25	Reserved	0	Reserved
24	RXCMFEN		Receive copy MAC control frames enable bit. Enables MAC control frames to be transferred to memory. MAC control frames are normally acted upon (if enabled), but not copied to memory. MAC control frames that are pause frames will be acted upon if enabled in MACCONTROL, regardless of the value of RXCMFEN. Frames transferred to memory due to RXCMFEN will have the CONTROL bit set in their EOP buffer descriptor.
		0	MAC control frames are filtered (but acted upon if enabled).
		1	MAC control frames are transferred to memory.
23	RXCSFEN		Receive copy short frames enable bit. Enables frames or fragments shorter than 64 bytes to be copied to memory. Frames transferred to memory due to RXCSFEN will have the FRAGMENT or UNDERSIZE bit set in their EOP buffer descriptor. Fragments are short frames that contain CRC / align / code errors and undersized are short frames without errors.
		0	Short frames are filtered.
		1	Short frames are transferred to memory.

Table 6-45. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) Field Descriptions (continued)

Bit	Field	Value	Description
22	RXCEFEN		Receive copy error frames enable bit. Enables frames containing errors to be transferred to memory. The appropriate error bit will be set in the frame EOP buffer descriptor.
		0	Frames containing errors are filtered.
		1	Frames containing errors are transferred to memory.
21	RXCAFEN		Receive copy all frames enable bit. Enables frames that do not address match (includes multicast frames that do not hash match) to be transferred to the promiscuous channel selected by RXPROMCH bits. Such frames will be marked with the NOMATCH bit in their EOP buffer descriptor.
		0	Frames that do not address match are filtered.
		1	Frames that do not address match are transferred to the promiscuous channel selected by RXPROMCH bits.
20-19	Reserved	0	Reserved
18-16	RXPROMCH	0-7h	Receive promiscuous channel select
		0	Select channel 0 to receive promiscuous frames
		1h	Select channel 1 to receive promiscuous frames
		2h	Select channel 2 to receive promiscuous frames
		3h	Select channel 3 to receive promiscuous frames
		4h	Select channel 4 to receive promiscuous frames
		5h	Select channel 5 to receive promiscuous frames
		6h	Select channel 6 to receive promiscuous frames
		7h	Select channel 7 to receive promiscuous frames
15-14	Reserved	0	Reserved
13	RXBROADEN		Receive broadcast enable. Enable received broadcast frames to be copied to the channel selected by RXBROADCH bits.
		0	Broadcast frames are filtered.
		1	Broadcast frames are copied to the channel selected by RXBROADCH bits.
12-11	Reserved	0	Reserved
10-8	RXBROADCH	0-7h	Receive broadcast channel select
		0	Select channel 0 to receive broadcast frames
		1h	Select channel 1 to receive broadcast frames
		2h	Select channel 2 to receive broadcast frames
		3h	Select channel 3 to receive broadcast frames
		4h	Select channel 4 to receive broadcast frames
		5h	Select channel 5 to receive broadcast frames
		6h	Select channel 6 to receive broadcast frames
		7h	Select channel 7 to receive broadcast frames
7-6	Reserved	0	Reserved
5	RXMULTEN		RX multicast enable. Enable received hash matching multicast frames to be copied to the channel selected by RXMULTCH bits.
		0	Multicast frames are filtered.
		1	Multicast frames are copied to the channel selected by RXMULTCH bits.
4-3	Reserved	0	Reserved

Table 6-45. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) Field Descriptions (continued)

Bit	Field	Value	Description
2-0	RXMULTCH	0-7h	Receive multicast channel select
		0	Select channel 0 to receive multicast frames
		1h	Select channel 1 to receive multicast frames
		2h	Select channel 2 to receive multicast frames
		3h	Select channel 3 to receive multicast frames
		4h	Select channel 4 to receive multicast frames
		5h	Select channel 5 to receive multicast frames
		6h	Select channel 6 to receive multicast frames
		7h	Select channel 7 to receive multicast frames

6.3.2.21 Receive Unicast Enable Set Register (RXUNICASTSET)

The receive unicast enable set register (RXUNICASTSET) is shown in Figure 6-46 and described in Table 6-46.

Figure 6-46. Receive Unicast Enable Set Register (RXUNICASTSET)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
RXCH7EN	RXCH6EN	RXCH5EN	RXCH4EN	RXCH3EN	RXCH2EN	RXCH1EN	RXCH0EN
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R = Read only; R/W = Read/Write; W1S = Write 1 to set, write of 0 has no effect; -*n* = value after reset

Table 6-46. Receive Unicast Enable Set Register (RXUNICASTSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7EN	0-1	Receive channel 7 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
6	RXCH6EN	0-1	Receive channel 6 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
5	RXCH5EN	0-1	Receive channel 5 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
4	RXCH4EN	0-1	Receive channel 4 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
3	RXCH3EN	0-1	Receive channel 3 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
2	RXCH2EN	0-1	Receive channel 2 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
1	RXCH1EN	0-1	Receive channel 1 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
0	RXCH0EN	0-1	Receive channel 0 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.

6.3.2.22 Receive Unicast Clear Register (RXUNICASTCLEAR)

The receive unicast clear register (RXUNICASTCLEAR) is shown in Figure 6-47 and described in Table 6-47.

Figure 6-47. Receive Unicast Clear Register (RXUNICASTCLEAR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
RXCH7EN	RXCH6EN	RXCH5EN	RXCH4EN	RXCH3EN	RXCH2EN	RXCH1EN	RXCH0EN
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R = Read only; R/W = Read/Write; W1C = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

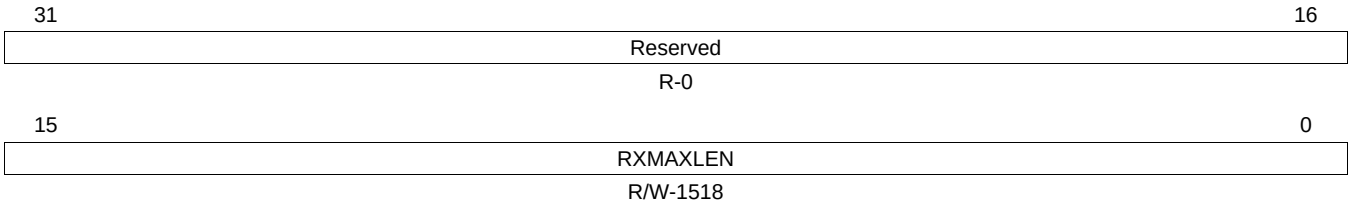
Table 6-47. Receive Unicast Clear Register (RXUNICASTCLEAR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7EN	0-1	Receive channel 7 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
6	RXCH6EN	0-1	Receive channel 6 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
5	RXCH5EN	0-1	Receive channel 5 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
4	RXCH4EN	0-1	Receive channel 4 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
3	RXCH3EN	0-1	Receive channel 3 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
2	RXCH2EN	0-1	Receive channel 2 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
1	RXCH1EN	0-1	Receive channel 1 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
0	RXCH0EN	0-1	Receive channel 0 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.

6.3.2.23 Receive Maximum Length Register (RXMAXLEN)

The receive maximum length register (RXMAXLEN) is shown in Figure 6-48 and described in Table 6-48.

Figure 6-48. Receive Maximum Length Register (RXMAXLEN)



LEGEND: R = Read only; R/W = Read/Write; *-n* = value after reset

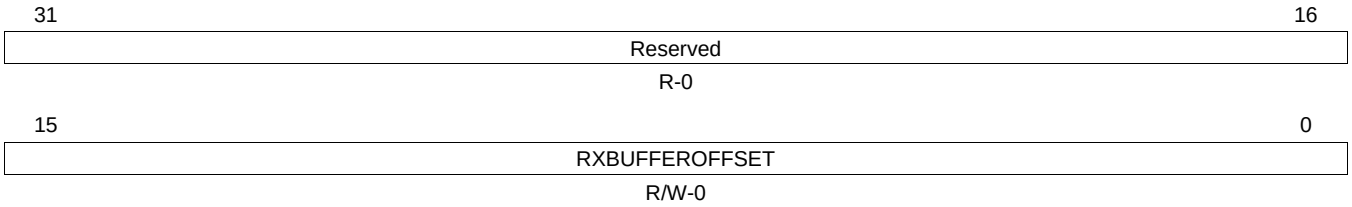
Table 6-48. Receive Maximum Length Register (RXMAXLEN) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	RXMAXLEN	0-FFFFh	Receive maximum frame length. These bits determine the maximum length of a received frame. The reset value is 5EEh (1518). Frames with byte counts greater than RXMAXLEN are long frames. Long frames with no errors are oversized frames. Long frames with CRC, code, or alignment error are jabber frames.

6.3.2.24 Receive Buffer Offset Register (RXBUFFEROFFSET)

The receive buffer offset register (RXBUFFEROFFSET) is shown in Figure 6-49 and described in Table 6-49.

Figure 6-49. Receive Buffer Offset Register (RXBUFFEROFFSET)



LEGEND: R = Read only; R/W = Read/Write; *-n* = value after reset

Table 6-49. Receive Buffer Offset Register (RXBUFFEROFFSET) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	RXBUFFEROFFSET	0-FFFFh	Receive buffer offset value. These bits are written by the EMAC into each frame SOP buffer descriptor Buffer Offset field. The frame data begins after the RXBUFFEROFFSET value of bytes. A value of 0 indicates that there are no unused bytes at the beginning of the data, and that valid data begins on the first byte of the buffer. A value of Fh (15) indicates that the first 15 bytes of the buffer are to be ignored by the EMAC and that valid buffer data starts on byte 16 of the buffer. This value is used for all channels.

6.3.2.25 Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH)

The receive filter low priority frame threshold register (RXFILTERLOWTHRESH) is shown in Figure 6-50 and described in Table 6-50.

Figure 6-50. Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		RXFILTERTHRESH	
R-0		R/W-0	

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 6-50. Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RXFILTERTHRESH	0-FFh	Receive filter low threshold. These bits contain the free buffer count threshold value for filtering low priority incoming frames. This field should remain 0, if no filtering is desired.

6.3.2.26 Receive Channel 0-7 Flow Control Threshold Register (RX*n*FLOWTHRESH)

The receive channel 0-7 flow control threshold register (RX*n*FLOWTHRESH) is shown in Figure 6-51 and described in Table 6-51.

Figure 6-51. Receive Channel *n* Flow Control Threshold Register (RX*n*FLOWTHRESH)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		RX*n*FLOWTHRESH	
R-0		R/W-0	

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 6-51. Receive Channel *n* Flow Control Threshold Register (RX*n*FLOWTHRESH) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RX*n*FLOWTHRESH	0-FFh	Receive flow threshold. These bits contain the threshold value for issuing flow control on incoming frames for channel *n* (when enabled).

6.3.2.27 Receive Channel 0-7 Free Buffer Count Register (RX*n*FREEBUFFER)

The receive channel 0-7 free buffer count register (RX*n*FREEBUFFER) is shown in Figure 6-52 and described in Table 6-52.

Figure 6-52. Receive Channel *n* Free Buffer Count Register (RX*n*FREEBUFFER)

31	16
Reserved	
R-0	

15	0
RX*n*FREEBUF	
WI-0	

LEGEND: R = Read only; WI = Write to increment; -*n* = value after reset

Table 6-52. Receive Channel *n* Free Buffer Count Register (RX*n*FREEBUFFER) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	RX*n*FREEBUF	0-FFh	Receive free buffer count. These bits contain the count of free buffers available. The RXFILTERTHRESH value is compared with this field to determine if low priority frames should be filtered. The RX*n*FLOWTHRESH value is compared with this field to determine if receive flow control should be issued against incoming packets (if enabled). This is a write-to-increment field. This field rolls over to 0 on overflow. If hardware flow control or QOS is used, the host must initialize this field to the number of available buffers (one register per channel). The EMAC decrements the associated channel register for each received frame by the number of buffers in the received frame. The host must write this field with the number of buffers that have been freed due to host processing.

6.3.2.28 MAC Control Register (MACCONTROL)

The MAC control register (MACCONTROL) is shown in Figure 6-53 and described in Table 6-53.

Figure 6-53. MAC Control Register (MACCONTROL)

31	18	17	16
Reserved		GIGFORCE	Reserved
R-0		R/W-0	R-0

15	14	13	12	11	10	9	8
Reserved	RXOFFLENBLOCK	RXOWNERSHIP	RXFIFOFLOWEN	CMDIDLE	Rsvd	TXPTYPE	Reserved
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R-0

7	6	5	4	3	2	1	0
GIG	TXPACE	GMIIEN	TXFLOWEN	RXBUFFERFLOWEN	Rsvd	LOOPBACK	FULLDUPLEX
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

LEGEND: R = Read only; R/W = Read/Write; *-n* = value after reset

Table 6-53. MAC Control Register (MACCONTROL) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved. Any writes to these bit(s) must always have a value of 0.
17	GIGFORCE	0-1	Gigabit force mode. This bit is used to force the EMAC into gigabit mode if the input MTCLK signal has been stopped by the PHY.
16-15	Reserved	0	Reserved. Any writes to these bit(s) must always have a value of 0.
14	RXOFFLENBLOCK		Receive offset/length word write block.
		0	Do not block the DMA writes to the receive buffer descriptor offset/buffer length word.
		1	Block all EMAC DMA controller writes to the receive buffer descriptor offset/buffer length words during packet processing. When this bit is set, the EMAC will never write the third word to any receive buffer descriptor.
13	RXOWNERSHIP		Receive ownership write bit value.
		0	EMAC writes the Receive ownership bit to 0 at the end of packet processing.
		1	EMAC writes the Receive ownership bit to 1 at the end of packet processing. If you do not use the ownership mechanism, you can set this mode to preclude the necessity of software having to set this bit each time the buffer descriptor is used.
12	RXFIFOFLOWEN		Receive FIFO flow control enable bit.
		0	Receive flow control is disabled. Full-duplex mode: no outgoing pause frames are sent.
		1	Receive flow control is enabled. Full-duplex mode: outgoing pause frames are sent when receive FIFO flow control is triggered.
11	CMDIDLE		Command Idle bit.
		0	Idle is not commanded.
		1	Idle is commanded (read the IDLE bit in the MACSTATUS register).
10	Reserved	0	Any writes to these bit(s) must always have a value of 0.
9	TXPTYPE		Transmit queue priority type.
		0	The queue uses a round-robin scheme to select the next channel for transmission.
		1	The queue uses a fixed-priority (channel 7 is highest priority) scheme to select the next channel for transmission.
8	Reserved	0	Any writes to these bit(s) must always have a value of 0.
7	GIG		Gigabit mode.
		0	Gigabit mode is disabled; 10/100 mode is in operation.
		1	Gigabit mode is enabled (full-duplex only).
6	TXPACE		Transmit pacing enable bit.
		0	Transmit pacing is disabled.
		1	Transmit pacing is enabled.

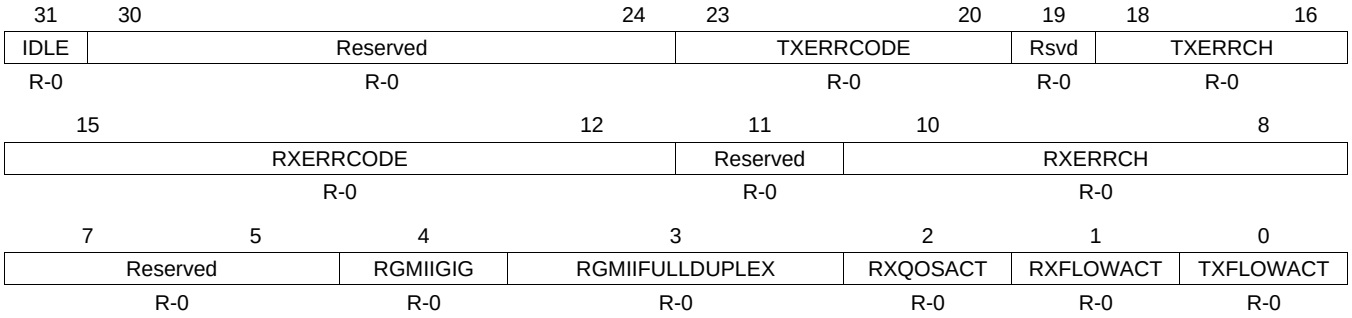
Table 6-53. MAC Control Register (MACCONTROL) Field Descriptions (continued)

Bit	Field	Value	Description
5	GMIIEN		GMII enable bit.
		0	GMII RX and TX are held in reset.
		1	GMII RX and TX are enabled for receive and transmit.
4	TXFLOWEN		Transmit flow control enable bit. This bit determines if incoming pause frames are acted upon in full-duplex mode. Incoming pause frames are not acted upon in half-duplex mode, regardless of the TXFLOWEN bit setting. The RXMBPENABLE bits determine whether or not received pause frames are transferred to memory.
		0	Transmit flow control is disabled. Full-duplex mode: incoming pause frames are not acted upon.
		1	Transmit flow control is enabled. Full-duplex mode: incoming pause frames are acted upon.
3	RXBUFFERFLOWEN		Receive buffer flow control enable bit.
		0	Receive flow control is disabled. Half-duplex mode: no flow control generated collisions are sent. Full-duplex mode: no outgoing pause frames are sent.
		1	Receive flow control is enabled. Half-duplex mode: collisions are initiated when receive buffer flow control is triggered. Full-duplex mode: outgoing pause frames are sent when receive flow control is triggered.
2	Reserved	0	Reserved.
1	LOOPBACK		Loopback mode. The loopback mode forces internal full-duplex mode regardless of the FULLDUPLEX bit setting. The loopback bit should be changed only when GMII bit is deasserted.
		0	Loopback mode is disabled.
		1	Loopback mode is enabled.
0	FULLDUPLEX		Full-duplex mode. The gigabit mode (GIG = 1) forces full-duplex mode regardless of the FULLDUPLEX bit setting.
		0	Half-duplex mode is enabled.
		1	Full-duplex mode is enabled.

6.3.2.29 MAC Status Register (MACSTATUS)

The MAC status register (MACSTATUS) is shown in Figure 6-54 and described in Table 6-54.

Figure 6-54. MAC Status Register (MACSTATUS)



LEGEND: R = Read only; R/W = Read/Write; *-n* = value after reset

Table 6-54. MAC Status Register (MACSTATUS) Field Descriptions

Bit	Field	Value	Description
31	IDLE		EMAC idle bit. This bit is set to 0 at reset; one clock after reset it goes to 1.
		0	The EMAC is not idle.
		1	The EMAC is in the idle state.
30-24	Reserved	0	Reserved
23-20	TXERRCODE	0-Fh	Transmit host error code. These bits indicate that EMAC detected transmit DMA related host errors. The host should read this field after a host error interrupt (HOSTPEND) to determine the error. Host error interrupts require hardware reset in order to recover. A 0 packet length is an error, but it is not detected.
		0	No error
		1h	SOP error; the buffer is the first buffer in a packet, but the SOP bit is not set in software.
		2h	Ownership bit not set in SOP buffer
		3h	Zero next buffer descriptor pointer without EOP
		4h	Zero buffer pointer
		5h	Zero buffer length
		6h	Packet length error (sum of buffers is less than packet length)
19	Reserved	0	Reserved
18-16	TXERRCH	0-7h	Transmit host error channel. These bits indicate which transmit channel the host error occurred on. This field is cleared to 0 on a host read.
		0	The host error occurred on transmit channel 0
		1h	The host error occurred on transmit channel 1
		2h	The host error occurred on transmit channel 2
		3h	The host error occurred on transmit channel 3
		4h	The host error occurred on transmit channel 4
		5h	The host error occurred on transmit channel 5
		6h	The host error occurred on transmit channel 6
		7h	The host error occurred on transmit channel 7
15-12	RXERRCODE	0-Fh	Receive host error code. These bits indicate that EMAC detected receive DMA related host errors. The host should read this field after a host error interrupt (HOSTPEND) to determine the error. Host error interrupts require hardware reset in order to recover.
		0	No error
		2h	Ownership bit not set in SOP buffer
		4h	Zero buffer pointer
11	Reserved	0	Reserved

Table 6-54. MAC Status Register (MACSTATUS) Field Descriptions (continued)

Bit	Field	Value	Description
10-8	RXERRCH	0-3h	Receive host error channel. These bits indicate which receive channel the host error occurred on. This field is cleared to 0 on a host read.
		0	The host error occurred on receive channel 0
		1h	The host error occurred on receive channel 1
		2h	The host error occurred on receive channel 2
		3h	The host error occurred on receive channel 3
		4h	The host error occurred on receive channel 4
		5h	The host error occurred on receive channel 5
		6h	The host error occurred on receive channel 6
		7h	The host error occurred on receive channel 7
7-5	Reserved	0	Reserved.
4	RGMIIGIG	0-1	RGMII gigabit. This is the value of RGMIIGIG input.
3	RGMIIFULLDUPLEX	0-1	RGMII full-duplex. This is the value of RGMIIFULLDUPLEX input.
2	RXQOSACT		Receive Quality of Service (QOS) active bit. When asserted, indicates that receive quality of service is enabled and that at least one channel freebuffer count (RX*n*FREEBUFFER) is less than or equal to the RXFILTERLOWTHRESH value.
		0	Receive quality of service is disabled.
		1	Receive quality of service is enabled.
1	RXFLOWACT		Receive flow control active bit. When asserted, at least one channel freebuffer count (RX*n*FREEBUFFER) is less than or equal to the channel's corresponding RX*n*FILTERTHRESH value.
		0	Receive flow control is inactive.
		1	Receive flow control is active.
0	TXFLOWACT		Transmit flow control active bit. When asserted, this bit indicates that the pause time period is being observed for a received pause frame. No new transmissions will begin while this bit is asserted, except for the transmission of pause frames. Any transmission in progress when this bit is asserted will complete.
		0	Transmit flow control is inactive.
		1	Transmit flow control is active.

6.3.2.30 Emulation Control Register (EMCONTROL)

The emulation control register (EMCONTROL) is shown in Figure 6-55 and described in Table 6-55.

Figure 6-55. Emulation Control Register (EMCONTROL)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		SOFT	FREE
R-0		R/W-0	R/W-0

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 6-55. Emulation Control Register (EMCONTROL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	SOFT	0-1	Emulation soft bit
0	FREE	0-1	Emulation free bit

6.3.2.31 FIFO Control Register (FIFOCONTROL)

The FIFO control register (FIFOCONTROL) is shown in Figure 6-56 and described in Table 6-56.

Figure 6-56. FIFO Control Register (FIFOCONTROL)

31	23	22	16
Reserved		RXFIFOFLOWTHRESH	
R-0		R/W-2h	

15	5	4	0
Reserved		TXCELLTHRESH	
R-0		R/W-24h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-56. FIFO Control Register (FIFOCONTROL) Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Reserved
22-16	RXFIFOFLOWTHRESH	0-7Fh	Receive FIFO flow control threshold. Occupancy of the receive FIFO when receive FIFO flow control is triggered (if enabled). The default value is 2h, which means that receive FIFO flow control is triggered when the occupancy of the FIFO reaches 2 cells.
15-5	Reserved	0	Reserved
4-0	TXCELLTHRESH	0-1Fh	Transmit FIFO cell threshold. Indicates the number of 64-byte packet cells required to be in the transmit FIFO before the packet transfer is initiated. Packets with fewer cells are initiated when the complete packet is contained in the FIFO. This value must be greater than or equal to 2 and less than or equal to 24 (2 ≥ TXCELLTHRESH ≤ 24).

6.3.2.32 MAC Configuration Register (MACCONFIG)

The MAC configuration register (MACCONFIG) is shown in Figure 6-57 and described in Table 6-57.

Figure 6-57. MAC Configuration Register (MACCONFIG)

31	24	23	16
TXCELLDEPTH		RXCELLDEPTH	
R-18h		R-44h	

15	8	7	0
ADDRESSTYPE		MACCFIG	
R-2h		R-3h	

LEGEND: R = Read only; -*n* = value after reset

Table 6-57. MAC Configuration Register (MACCONFIG) Field Descriptions

Bit	Field	Value	Description
31-24	TXCELLDEPTH	0-FFh	Transmit cell depth. Indicate the number of cells in the transmit FIFO.
23-16	RXCELLDEPTH	0-FFh	Receive cell depth. Indicate the number of cells in the receive FIFO.
15-8	ADDRESSTYPE	0-FFh	Address type.
7-0	MACCFIG	0-FFh	MAC configuration value.

6.3.2.33 Soft Reset Register (SOFTRESET)

The soft reset register (SOFTRESET) is shown in Figure 6-58 and described in Table 6-58.

Figure 6-58. Soft Reset Register (SOFTRESET)

31	16
Reserved	
R-0	

15	1	0
Reserved		SOFTRESET
R-0		R/W-0

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 6-58. Soft Reset Register (SOFTRESET) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	SOFTRESET		Software reset. Writing a 1 to this bit causes the EMAC logic to be reset. Software reset occurs when the receive and transmit DMA controllers are in an idle state to avoid locking up the Configuration bus. After writing a 1 to this bit, it may be polled to determine if the reset has occurred. If a 1 is read, the reset has not yet occurred. If a 0 is read, then a reset has occurred.
		0	A software reset has not occurred.
		1	A software reset has occurred.

6.3.2.34 MAC Source Address Low Bytes Register (MACSRCADDRLO)

The MAC source address low bytes register (MACSRCADDRLO) is shown in Figure 6-59 and described in Table 6-59.

Figure 6-59. MAC Source Address Low Bytes Register (MACSRCADDRLO)

31	16
Reserved	
R-0	

15	8	7	0
MACSRCADDR0		MACSRCADDR1	
R/W-0		R/W-0	

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 6-59. MAC Source Address Low Bytes Register (MACSRCADDRLO) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-8	MACSRCADDR0	0-FFh	MAC source address lower 8 bits (byte 0)
7-0	MACSRCADDR1	0-FFh	MAC source address bits 15-8 (byte 1)

6.3.2.35 MAC Source Address High Bytes Register (MACSRCADDRHI)

The MAC source address high bytes register (MACSRCADDRHI) is shown in Figure 6-60 and described in Table 6-60.

Figure 6-60. MAC Source Address High Bytes Register (MACSRCADDRHI)

31	24	23	16
MACSRCADDR2		MACSRCADDR3	
R/W-0		R/W-0	

15	8	7	0
MACSRCADDR4		MACSRCADDR5	
R/W-0		R/W-0	

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 6-60. MAC Source Address High Bytes Register (MACSRCADDRHI) Field Descriptions

Bit	Field	Value	Description
31-24	MACSRCADDR2	0-FFh	MAC source address bits 23-16 (byte 2)
23-16	MACSRCADDR3	0-FFh	MAC source address bits 31-24 (byte 3)
15-8	MACSRCADDR4	0-FFh	MAC source address bits 39-32 (byte 4)
7-0	MACSRCADDR5	0-FFh	MAC source address bits 47-40 (byte 5)

6.3.2.36 MAC Hash Address Register 1 (MACHASH1)

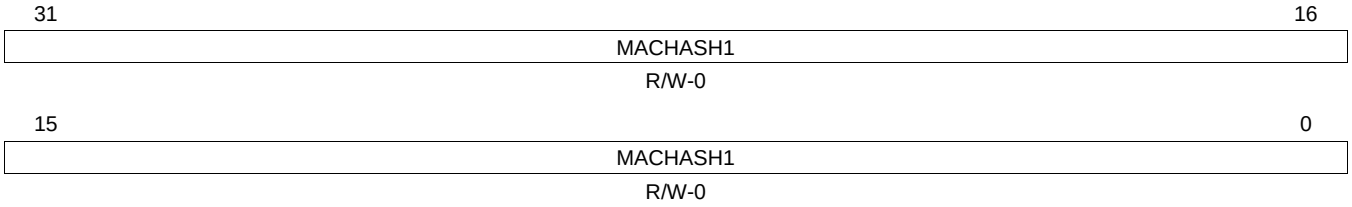
The MAC hash registers allow group addressed frames to be accepted on the basis of a hash function of the address. The hash function creates a 6-bit data value (Hash_fun) from the 48-bit destination address (DA) as follows:

```
Hash_fun(0)=DA(0) XOR DA(6) XOR DA(12) XOR DA(18) XOR DA(24) XOR DA(30) XOR DA(36) XOR DA(42);

Hash_fun(1)=DA(1) XOR DA(7) XOR DA(13) XOR DA(19) XOR DA(25) XOR DA(31) XOR DA(37) XOR DA(43);

Hash_fun(2)=DA(2) XOR DA(8) XOR DA(14) XOR DA(20) XOR DA(26) XOR DA(32) XOR DA(38) XOR DA(44);

Hash_fun(3)=DA(3) XOR DA(9) XOR DA(15) XOR DA(21) XOR DA(27) XOR DA(33) XOR DA(39) XOR DA(45);

Hash_fun(4)=DA(4) XOR DA(10) XOR DA(16) XOR DA(22) XOR DA(28) XOR DA(34) XOR DA(40) XOR DA(46);

Hash_fun(5)=DA(5) XOR DA(11) XOR DA(17) XOR DA(23) XOR DA(29) XOR DA(35) XOR DA(41) XOR DA(47);
```

This function is used as an offset into a 64-bit hash table stored in MACHASH1 and MACHASH2 that indicates whether a particular address should be accepted or not.

The MAC hash address register 1 (MACHASH1) is shown in Figure 6-61 and described in Table 6-61.

Figure 6-61. MAC Hash Address Register 1 (MACHASH1)



LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

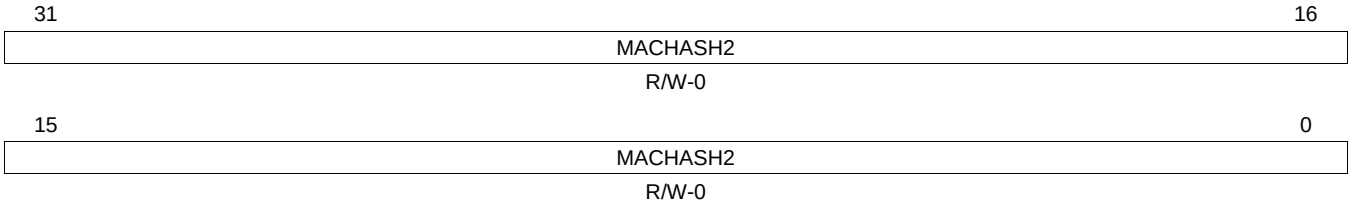
Table 6-61. MAC Hash Address Register 1 (MACHASH1) Field Descriptions

Bit	Field	Value	Description
31-0	MACHASH1	0-FFFF FFFFh	Least-significant 32 bits of the hash table corresponding to hash values 0 to 31. If a hash table bit is set, then a group address that hashes to that bit index is accepted.

6.3.2.37 MAC Hash Address Register 2 (MACHASH2)

The MAC hash address register 2 (MACHASH2) is shown in Figure 6-62 and described in Table 6-62.

Figure 6-62. MAC Hash Address Register 2 (MACHASH2)



LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 6-62. MAC Hash Address Register 2 (MACHASH2) Field Descriptions

Bit	Field	Value	Description
31-0	MACHASH2	0-FFFF FFFFh	Most-significant 32 bits of the hash table corresponding to hash values 32 to 63. If a hash table bit is set, then a group address that hashes to that bit index is accepted.

6.3.2.38 Back Off Test Register (BOFFTEST)

The back off test register (BOFFTEST) is shown in Figure 6-63 and described in Table 6-63.

Figure 6-63. Back Off Random Number Generator Test Register (BOFFTEST)

31	26	25	16
Reserved		RNDNUM	
R-0		R-0	

15	12	11	10	9	0
COLLCOUNT		Reserved		TXBACKOFF	
R-0		R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 6-63. Back Off Test Register (BOFFTEST) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	RNDNUM	0-3FFh	Backoff random number generator. This field allows the Backoff Random Number Generator to be read. Reading this field returns the generator's current value. The value is reset to 0 and begins counting on the clock after the deassertion of reset.
15-12	COLLCOUNT	0-Fh	Collision count. These bits indicate the number of collisions the current frame has experienced.
11-10	Reserved	0	Reserved
9-0	TXBACKOFF	0-3FFh	Backoff count. This field allows the current value of the backoff counter to be observed for test purposes. This field is loaded automatically according to the backoff algorithm, and is decremented by one for each slot time after the collision.

6.3.2.39 Transmit Pacing Algorithm Test Register (TPACETEST)

The transmit pacing algorithm test register (TPACETEST) is shown in Figure 6-64 and described in Table 6-64.

Figure 6-64. Transmit Pacing Algorithm Test Register (TPACETEST)

31	16
Reserved	
R-0	

15	5	4	0
Reserved		PACEVAL	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

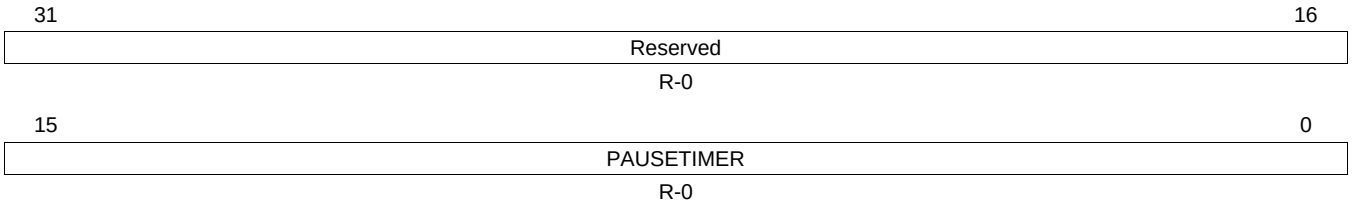
Table 6-64. Transmit Pacing Algorithm Test Register (TPACETEST) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	PACEVAL	0-1Fh	Pacing register current value. A nonzero value in this field indicates that transmit pacing is active. A transmit frame collision or deferral causes PACEVAL to be loaded with 1Fh (31); good frame transmissions (with no collisions or deferrals) cause PACEVAL to be decremented down to 0. When PACEVAL is nonzero, the transmitter delays four Inter Packet Gaps between new frame transmissions after each successfully transmitted frame that had no deferrals or collisions. If a transmit frame is deferred or suffers a collision, the IPG time is not stretched to four times the normal value. Transmit pacing helps reduce capture effects, which improves overall network bandwidth.

6.3.2.40 Receive Pause Timer Register (RXPAUSE)

The receive pause timer register (RXPAUSE) is shown in Figure 6-65 and described in Table 6-65.

Figure 6-65. Receive Pause Timer Register (RXPAUSE)



LEGEND: R = Read only; -*n* = value after reset

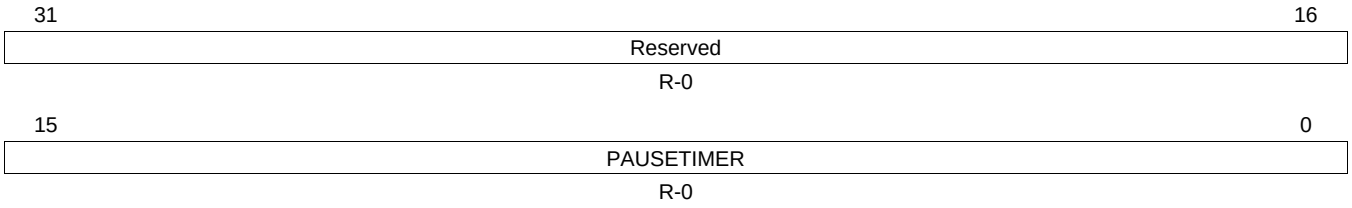
Table 6-65. Receive Pause Timer Register (RXPAUSE) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	PAUSETIMER	0-FFh	Receive pause timer value. These bits allow the contents of the receive pause timer to be observed. The receive pause timer is loaded with FF00h when the EMAC sends an outgoing pause frame (with pause time of FFFFh). The receive pause timer is decremented at slot time intervals. If the receive pause timer decrements to 0, then another outgoing pause frame is sent and the load/decrement process is repeated.

6.3.2.41 Transmit Pause Timer Register (TXPAUSE)

The transmit pause timer register (TXPAUSE) is shown in Figure 6-66 and described in Table 6-66.

Figure 6-66. Transmit Pause Timer Register (TXPAUSE)



LEGEND: R = Read only; -*n* = value after reset

Table 6-66. Transmit Pause Timer Register (TXPAUSE) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	PAUSETIMER	0-FFh	Transmit pause timer value. These bits allow the contents of the transmit pause timer to be observed. The transmit pause timer is loaded by a received (incoming) pause frame, and then decremented at slot time intervals down to 0, at which time EMAC transmit frames are again enabled.

6.3.2.42 MAC Address Low Bytes Register (MACADDRLO)

The MAC address low bytes register used in address matching (MACADDRLO), is shown in Figure 6-67 and described in Table 6-67.

Figure 6-67. MAC Address Low Bytes Register (MACADDRLO)

31	21	20	19	18	16
Reserved		VALID	MATCHFILT	CHANNEL	
R-0		R/W-x	R/W-x	R/W-x	

15	8	7	0
MACADDR0		MACADDR1	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; -x = value is indeterminate after reset

Table 6-67. MAC Address Low Bytes Register (MACADDRLO) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	VALID		Address valid bit. This bit should be cleared to zero for unused address RAM locations.
		0	Address location is not valid and will not be used in determining whether or not an incoming packet matches or is filtered.
		1	Address location is valid and will be used in determining whether or not an incoming packet matches or is filtered.
19	MATCHFILT		Match or filter bit
		0	The address will be used (if the VALID bit is set) to determine if the incoming packet address should be filtered.
		1	The address will be used (if the VALID bit is set) to determine if the incoming packet address is a match.
18-16	CHANNEL	0-7h	Channel bit. Determines which receive channel a valid address match will be transferred to. The channel is a don't care if the MATCHFILT bit is cleared to 0.
15-8	MACADDR0	0-FFh	MAC address lower 8 bits (byte 0)
7-0	MACADDR1	0-FFh	MAC address bits 15-8 (byte 1)

6.3.2.43 MAC Address High Bytes Register (MACADDRHI)

The MAC address high bytes register (MACADDRHI) is shown in Figure 6-68 and described in Table 6-68.

Figure 6-68. MAC Address High Bytes Register (MACADDRHI)

31	24	23	16
MACADDR2		MACADDR3	
R/W-0		R/W-0	

15	8	7	0
MACADDR4		MACADDR5	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 6-68. MAC Address High Bytes Register (MACADDRHI) Field Descriptions

Bit	Field	Value	Description
31-24	MACADDR2	0-FFh	MAC source address bits 23-16 (byte 2)
23-16	MACADDR3	0-FFh	MAC source address bits 31-24 (byte 3)
15-8	MACADDR4	0-FFh	MAC source address bits 39-32 (byte 4)
7-0	MACADDR5	0-FFh	MAC source address bits 47-40 (byte 5). Bit 40 is the group bit. It is forced to 0 and read as 0. Therefore, only unicast addresses are represented in the address table.

6.3.2.44 MAC Index Register (MACINDEX)

The MAC index register (MACINDEX) is shown in Figure 6-69 and described in Table 6-69.

Figure 6-69. MAC Index Register (MACINDEX)

31	16
Reserved	
R-0	

15	5	4	0
Reserved		MACINDEX	
R-0		R/W-0	

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

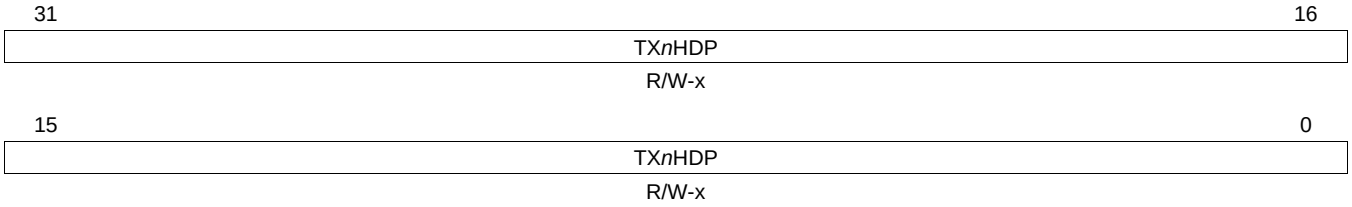
Table 6-69. MAC Index Register (MACINDEX) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	MACINDEX	0-1Fh	MAC address index. The host must write the index into the RX ADDR RAM in the MACINDEX field, followed by the upper 32-bits of address, followed by the lower 16-bits of address (with control bits). The 53-bit indexed RAM location is written when the low location is written. All 32 address RAM locations must be initialized prior to enabling packet reception.

6.3.2.45 Transmit Channel 0-7 DMA Head Descriptor Pointer Register (TX*n*HDP)

The transmit channel 0-7 DMA head descriptor pointer register (TX*n*HDP) is shown in Figure 6-70 and described in Table 6-70.

Figure 6-70. Transmit Channel *n* DMA Head Descriptor Pointer Register (TX*n*HDP)



LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

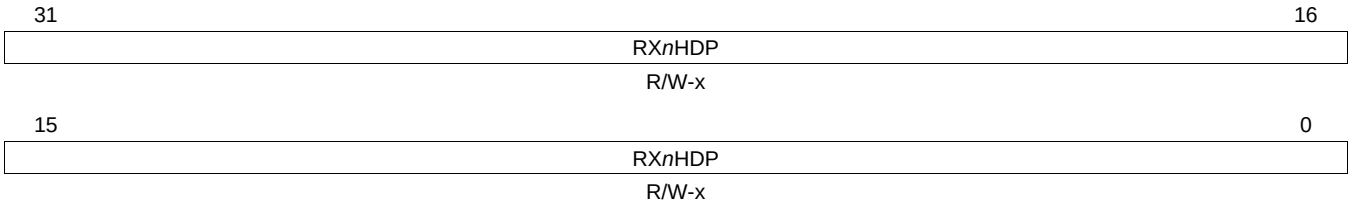
Table 6-70. Transmit Channel *n* DMA Head Descriptor Pointer Register (TX*n*HDP) Field Descriptions

Bit	Field	Value	Description
31-0	TX*n*HDP	0-FFFF FFFFh	Transmit channel *n* DMA Head Descriptor pointer. Writing a transmit DMA buffer descriptor address to a head pointer location initiates transmit DMA operations in the queue for the selected channel. Writing to these locations when they are nonzero is an error (except at reset). Host software must initialize these locations to 0 on reset.

6.3.2.46 Receive Channel 0-7 DMA Head Descriptor Pointer Register (RX*n*HDP)

The receive channel 0-7 DMA head descriptor pointer register (RX*n*HDP) is shown in Figure 6-71 and described in Table 6-71.

Figure 6-71. Receive Channel *n* DMA Head Descriptor Pointer Register (RX*n*HDP)



LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

Table 6-71. Receive Channel *n* DMA Head Descriptor Pointer Register (RX*n*HDP) Field Descriptions

Bit	Field	Value	Description
31-0	RX*n*HDP	0-FFFF FFFFh	Receive channel *n* DMA Head Descriptor pointer. Writing a receive DMA buffer descriptor address to this location allows receive DMA operations in the selected channel when a channel frame is received. Writing to these locations when they are nonzero is an error (except at reset). Host software must initialize these locations to 0 on reset.

6.3.2.47 Transmit Channel 0-7 Completion Pointer Register (TX*n*CP)

The transmit channel 0-7 completion pointer register (TX*n*CP) is shown in Figure 6-72 and described in Table 6-72.

Figure 6-72. Transmit Channel *n* Completion Pointer Register (TX*n*CP)

31	16
TX*n*CP	
R/W-x	

15	0
TX*n*CP	
R/W-x	

LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

Table 6-72. Transmit Channel *n* Completion Pointer Register (TX*n*CP) Field Descriptions

Bit	Field	Value	Description
31-0	TX*n*CP	0-FFFF FFFFh	Transmit channel *n* completion pointer register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The EMAC uses the value written to determine if the interrupt should be deasserted.

6.3.2.48 Receive Channel 0-7 Completion Pointer Register (RX*n*CP)

The receive channel 0-7 completion pointer register (RX*n*CP) is shown in Figure 6-73 and described in Table 6-73.

Figure 6-73. Receive Channel *n* Completion Pointer Register (RX*n*CP)

31	16
RX*n*CP	
R/W-x	

15	0
RX*n*CP	
R/W-x	

LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

Table 6-73. Receive Channel *n* Completion Pointer Register (RX*n*CP) Field Descriptions

Bit	Field	Value	Description
31-0	RX*n*CP	0-FFFF FFFFh	Receive channel *n* completion pointer register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The EMAC uses the value written to determine if the interrupt should be deasserted.

6.3.2.49 Network Statistics Registers

The EMAC has a set of statistics that record events associated with frame traffic. The statistics values are cleared to zero 38 clocks after the rising edge of reset. When the GMII bit in the MACCONTROL register is set, all statistics registers (see Figure 6-74) are write-to-decrement. The value written is subtracted from the register value with the result stored in the register. If a value greater than the statistics value is written, then zero is written to the register (writing FFFF FFFFh clears a statistics location). When the GMII bit is cleared, all statistics registers are read/write (normal write direct, so writing 0000 0000h clears a statistics location). All write accesses must be 32-bit accesses.

The statistics interrupt (STATPEND) is issued, if enabled, when any statistics value is greater than or equal to 8000 0000h. The statistics interrupt is removed by writing to decrement any statistics value greater than 8000 0000h. The statistics are mapped into internal memory space and are 32-bits wide. All statistics rollover from FFFF FFFFh to 0000 0000h.

Figure 6-74. Statistics Register

31	0
COUNT	
R/WD-0	

LEGEND: R/W = Read/Write; WD = Write to decrement; -*n* = value after reset

6.3.2.49.1 Good Receive Frames Register (RXGOODFRAMES)

The total number of good frames received on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See **Section 6.2.5.5** for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

6.3.2.49.2 Broadcast Receive Frames Register (RXBCASTFRAMES)

The total number of good broadcast frames received on the EMAC. A good broadcast frame is defined as having all of the following:

- Any data or MAC control frame that was destined for address FF-FF-FF-FF-FF-FFh only
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See Section 6.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

6.3.2.49.3 Multicast Receive Frames Register (RXMCASTFRAMES)

The total number of good multicast frames received on the EMAC. A good multicast frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any multicast address other than FF-FF-FF-FF-FF-FFh
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See Section 6.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

6.3.2.49.4 Pause Receive Frames Register (RXPAUSEFRAMES)

The total number of IEEE 802.3X pause frames received by the EMAC (whether acted upon or not). A pause frame is defined as having all of the following:

- Contained any unicast, broadcast, or multicast address
- Contained the length/type field value 88.08h and the opcode 0001h
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error
- Pause-frames had been enabled on the EMAC (TXFLOWEN bit is set in MACCONTROL).

The EMAC could have been in either half-duplex or full-duplex mode. See Section 6.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

6.3.2.49.5 Receive CRC Errors Register (RXCRCERRORS)

The total number of frames received on the EMAC that experienced a CRC error. A frame with CRC errors is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no alignment or code error
- Had a CRC error. A CRC error is defined as having all of the following:
  - A frame containing an even number of nibbles
  - Fails the frame check sequence test

See Section 6.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

6.3.2.49.6 Receive Alignment/Code Errors Register (RXALIGNCODEERRORS)

The total number of frames received on the EMAC that experienced an alignment error or code error. Such a frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had either an alignment error or a code error
  - An alignment error is defined as having all of the following:
    - A frame containing an odd number of nibbles
    - Fails the frame check sequence test, if the final nibble is ignored
  - A code error is defined as a frame that has been discarded because the EMAC_RXER pin is driven with a one for at least one bit-time's duration at any point during the frame's reception.

Overruns have no effect on this statistic.

CRC alignment or code errors can be calculated by summing receive alignment errors, receive code errors, and receive CRC errors.

6.3.2.49.7 Receive Oversized Frames Register (RXOVERSIZED)

The total number of oversized frames received on the EMAC. An oversized frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was greater than RXMAXLEN in bytes
- Had no CRC error, alignment error, or code error

See Section 6.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

6.3.2.49.8 Receive Jabber Frames Register (RXJABBER)

The total number of jabber frames received on the EMAC. A jabber frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was greater than RXMAXLEN bytes long
- Had a CRC error, alignment error, or code error

See Section 6.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

6.3.2.49.9 Receive Undersized Frames Register (RXUNDERSIZED)

The total number of undersized frames received on the EMAC. An undersized frame is defined as having all of the following:

- Was any data frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was less than 64 bytes long
- Had no CRC error, alignment error, or code error

See Section 6.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

6.3.2.49.10 Receive Frame Fragments Register (RXFRAGMENTS)

The total number of frame fragments received on the EMAC. A frame fragment is defined as having all of the following:

- Any data frame (address matching does not matter)
- Was less than 64 bytes long
- Had a CRC error, alignment error, or code error
- Was not the result of a collision caused by half duplex, collision based flow control

See Section 6.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

6.3.2.49.11 Filtered Receive Frames Register (RXFILTERED)

The total number of frames received on the EMAC that the EMAC address matching process indicated should be discarded. Such a frame is defined as having all of the following:

- Was any data frame (not MAC control frame) destined for any unicast, broadcast, or multicast address
- Did not experience any CRC error, alignment error, code error
- The address matching process decided that the frame should be discarded (filtered) because it did not match the unicast, broadcast, or multicast address, and it did not match due to promiscuous mode.

To determine the number of receive frames discarded by the EMAC for any reason, sum the following statistics (promiscuous mode disabled):

- Receive fragments
- Receive undersized frames
- Receive CRC errors
- Receive alignment/code errors
- Receive jabbers
- Receive overruns
- Receive filtered frames

This may not be an exact count because the receive overruns statistic is independent of the other statistics, so if an overrun occurs at the same time as one of the other discard reasons, then the above sum double-counts that frame.

6.3.2.49.12 Receive QOS Filtered Frames Register (RXQOSFILTERED)

The total number of frames received on the EMAC that were filtered due to receive quality of service (QOS) filtering. Such a frame is defined as having all of the following:

- Any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- The frame destination channel flow control threshold register (RX*n*FLOWTHRESH) value was greater than or equal to the channel's corresponding free buffer register (RX*n*FREEBUFFER) value
- Was of length 64 to RXMAXLEN
- RXQOSEN bit is set in RXMBPENABLE
- Had no CRC error, alignment error, or code error

See Section 6.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

6.3.2.49.13 Receive Octet Frames Register (RXOCTETS)

The total number of bytes in all good frames received on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See Section 6.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

6.3.2.49.14 Good Transmit Frames Register (TXGOODFRAMES)

The total number of good frames transmitted on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Was any length
- Had no late or excessive collisions, no carrier loss, and no underrun

6.3.2.49.15 Broadcast Transmit Frames Register (TXBCASTFRAMES)

The total number of good broadcast frames transmitted on the EMAC. A good broadcast frame is defined as having all of the following:

- Any data or MAC control frame destined for address FF-FF-FF-FF-FF-FFh only
- Was of any length
- Had no late or excessive collisions, no carrier loss, and no underrun

6.3.2.49.16 Multicast Transmit Frames Register (TXMCASTFRAMES)

The total number of good multicast frames transmitted on the EMAC. A good multicast frame is defined as having all of the following:

- Any data or MAC control frame destined for any multicast address other than FF-FF-FF-FF-FF-FFh
- Was of any length
- Had no late or excessive collisions, no carrier loss, and no underrun

6.3.2.49.17 Pause Transmit Frames Register (TXPAUSEFRAMES)

The total number of IEEE 802.3X pause frames transmitted by the EMAC. Pause frames cannot underrun or contain a CRC error because they are created in the transmitting MAC, so these error conditions have no effect on this statistic. Pause frames sent by software are not included in this count. Since pause frames are only transmitted in full-duplex mode, carrier loss and collisions have no effect on this statistic.

Transmitted pause frames are always 64-byte multicast frames so appear in the multicast transmit frames register and 64 octet frames register statistics.

6.3.2.49.18 Deferred Transmit Frames Register (TXDEFERRED)

The total number of frames transmitted on the EMAC that first experienced deferment. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced no collisions before being successfully transmitted
- Found the medium busy when transmission was first attempted, so had to wait.

CRC errors have no effect on this statistic.

6.3.2.49.19 Transmit Collision Frames Register (TXCOLLISION)

The total number of times that the EMAC experienced a collision. Collisions occur under two circumstances:

- When a transmit data or MAC control frame has all of the following:
  - Was destined for any unicast, broadcast, or multicast address
  - Was any size
  - Had no carrier loss and no underrun
  - Experienced a collision. A jam sequence is sent for every non-late collision, so this statistic increments on each occasion if a frame experiences multiple collisions (and increments on late collisions).
- When the EMAC is in half-duplex mode, flow control is active, and a frame reception begins.

CRC errors have no effect on this statistic.

6.3.2.49.20 Transmit Single Collision Frames Register (TXSINGLECOLL)

The total number of frames transmitted on the EMAC that experienced exactly one collision. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced one collision before successful transmission. The collision was not late.

CRC errors have no effect on this statistic.

6.3.2.49.21 Transmit Multiple Collision Frames Register (TXMULTICOLL)

The total number of frames transmitted on the EMAC that experienced multiple collisions. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 2 to 15 collisions before being successfully transmitted. None of the collisions were late.

CRC errors have no effect on this statistic.

6.3.2.49.22 Transmit Excessive Collision Frames Register (TXEXCESSIVECOLL)

The total number of frames when transmission was abandoned due to excessive collisions. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 16 collisions before abandoning all attempts at transmitting the frame. None of the collisions were late.

CRC errors have no effect on this statistic.

6.3.2.49.23 Transmit Late Collision Frames Register (TXLATECOLL)

The total number of frames when transmission was abandoned due to a late collision. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced a collision later than 512 bit-times into the transmission. There may have been up to 15 previous (non-late) collisions that had previously required the transmission to be reattempted. The late collisions statistic dominates over the single, multiple, and excessive collisions statistics. If a late collision occurs, the frame is not counted in any of these other three statistics.

CRC errors, carrier loss, and underrun have no effect on this statistic.

6.3.2.49.24 Transmit Underrun Error Register (TXUNDERRUN)

The number of frames sent by the EMAC that experienced FIFO underrun. Late collisions, CRC errors, carrier loss, and underrun have no effect on this statistic.

6.3.2.49.25 Transmit Carrier Sense Errors Register (TXCARRIERSENSE)

The total number of frames on the EMAC that experienced carrier loss. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- The carrier sense condition was lost or never asserted when transmitting the frame (the frame is not retransmitted)

CRC errors and underrun have no effect on this statistic.

6.3.2.49.26 Transmit Octet Frames Register (TXOCTETS)

The total number of bytes in all good frames transmitted on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Was any length
- Had no late or excessive collisions, no carrier loss, and no underrun

6.3.2.49.27 Transmit and Receive 64 Octet Frames Register (FRAME64)

The total number of 64-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was exactly 64-bytes long. (If the frame was being transmitted and experienced carrier loss that resulted in a frame of this size being transmitted, then the frame is recorded in this statistic).

CRC errors, alignment/code errors, and overruns do not affect the recording of frames in this statistic.

6.3.2.49.28 Transmit and Receive 65 to 127 Octet Frames Register (FRAME65T127)

The total number of 65-byte to 127-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 65-bytes to 127-bytes long

CRC errors, alignment/code errors, underruns, and overruns do not affect the recording of frames in this statistic.

6.3.2.49.29 Transmit and Receive 128 to 255 Octet Frames Register (FRAME128T255)

The total number of 128-byte to 255-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 128-bytes to 255-bytes long

CRC errors, alignment/code errors, underruns, and overruns do not affect the recording of frames in this statistic.

6.3.2.49.30 Transmit and Receive 256 to 511 Octet Frames Register (FRAME256T511)

The total number of 256-byte to 511-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 256-bytes to 511-bytes long

CRC errors, alignment/code errors, underruns, and overruns do not affect the recording of frames in this statistic.

6.3.2.49.31 Transmit and Receive 512 to 1023 Octet Frames Register (FRAME512T1023)

The total number of 512-byte to 1023-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 512-bytes to 1023-bytes long

CRC errors, alignment/code errors, and overruns do not affect the recording of frames in this statistic.

6.3.2.49.32 Transmit and Receive 1024 to RXMAXLEN Octet Frames Register (FRAME1024TUP)

The total number of 1024-byte to RXMAXLEN-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 1024-bytes to RXMAXLEN-bytes long

CRC/alignment/code errors, underruns, and overruns do not affect frame recording in this statistic.

6.3.2.49.33 Network Octet Frames Register (NETOCTETS)

The total number of bytes of frame data received and transmitted on the EMAC. Each frame counted has all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address (address match does not matter)
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)

Also counted in this statistic is:

- Every byte transmitted before a carrier-loss was experienced
- Every byte transmitted before each collision was experienced (multiple retries are counted each time)
- Every byte received if the EMAC is in half-duplex mode until a jam sequence was transmitted to initiate flow control. (The jam sequence is not counted to prevent double-counting).

Error conditions such as alignment errors, CRC errors, code errors, overruns, and underruns do not affect the recording of bytes in this statistic. The objective of this statistic is to give a reasonable indication of Ethernet utilization.

6.3.2.49.34 Receive FIFO or DMA Start of Frame Overruns Register (RXSOFOVERRUNS)

The total number of frames received on the EMAC that had either a FIFO or DMA start of frame (SOF) overrun. An SOF overrun frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)
- The EMAC was unable to receive it because it did not have the resources to receive it (cell FIFO full or no DMA buffer available at the start of the frame).

CRC errors, alignment errors, and code errors have no effect on this statistic.

6.3.2.49.35 Receive FIFO or DMA Middle of Frame Overruns Register (RXMOFOVERRUNS)

The total number of frames received on the EMAC that had either a FIFO or DMA middle of frame (MOF) overrun. An MOF overrun frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)
- The EMAC was unable to receive it because it did not have the resources to receive it (cell FIFO full or no DMA buffer available after the frame was successfully started - no SOF overrun).

CRC errors, alignment errors, and code errors have no effect on this statistic.

6.3.2.49.36 Receive DMA Overruns Register (RXDMAOVERRUNS)

The total number of frames received on the EMAC that had either a DMA start of frame (SOF) overrun or a DMA middle of frame (MOF) overrun. A receive DMA overrun frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)
- The EMAC was unable to receive it because it did not have the DMA buffer resources to receive it (zero head descriptor pointer at the start or during the middle of the frame reception).

CRC errors, alignment errors, and code errors have no effect on this statistic.

6.3.3 MDIO Registers

Table 6-74 lists the memory-mapped registers for the MDIO module. For the base address of these registers, see Table 1-13.

Table 6-74. Management Data Input/Output (MDIO) Registers

Address Offset	Acronym	Register Description	Section
0h	VERSION	MDIO Version Register	Section 6.3.3.1
4h	CONTROL	MDIO Control Register	Section 6.3.3.2
8h	ALIVE	PHY Alive Status register	Section 6.3.3.3
Ch	LINK	PHY Link Status Register	Section 6.3.3.4
10h	LINKINTRAW	MDIO Link Status Change Interrupt (Unmasked) Register	Section 6.3.3.5
14h	LINKINTMASKED	MDIO Link Status Change Interrupt (Masked) Register	Section 6.3.3.6
20h	USERINTRAW	MDIO User Command Complete Interrupt (Unmasked) Register	Section 6.3.3.7
24h	USERINTMASKED	MDIO User Command Complete Interrupt (Masked) Register	Section 6.3.3.8
28h	USERINTMASKSET	MDIO User Command Complete Interrupt Mask Set Register	Section 6.3.3.9
2Ch	USERINTMASKCLEAR	MDIO User Command Complete Interrupt Mask Clear Register	Section 6.3.3.10
80h	USERACCESS0	MDIO User Access Register 0	Section 6.3.3.11
84h	USERPHYSEL0	MDIO User PHY Select Register 0	Section 6.3.3.12
88h	USERACCESS1	MDIO User Access Register 1	Section 6.3.3.13
8Ch	USERPHYSEL1	MDIO User PHY Select Register 1	Section 6.3.3.14

6.3.3.1 MDIO Version Register (VERSION)

The MDIO version register (VERSION) is shown in Figure 6-75 and described in Table 6-75.

Figure 6-75. MDIO Version Register (VERSION)

31	16
MODID	
R-7h	

15	8	7	0
REVMAJ		REVMIN	
R-1h		R-4h	

LEGEND: R = Read only; -*n* = value after reset

Table 6-75. MDIO Version Register (VERSION) Field Descriptions

Bit	Field	Value	Description
31-16	MODID	0-FFFFh	Identifies type of peripheral.
15-8	REVMAJ	0-FFh	Management Interface Module major revision value.
7-0	REVMIN	0-FFh	Management Interface Module minor revision value.

6.3.3.2 MDIO Control Register (CONTROL)

The MDIO control register (CONTROL) is shown in Figure 6-76 and described in Table 6-76.

Figure 6-76. MDIO Control Register (CONTROL)

31	30	29	28 24	23 21	20	19	18	17 16
IDLE	ENABLE	Rsvd	HIGHEST_USER_CHANNEL	Reserved	PREAMBLE	FAULT	FAULTENB	Reserved
R-1	R/W-0	R-0	R-1	R-0	R/W-0	R/W1C-0	R/W-0	R-0

15 0
CLKDIV
R/W-FFh

LEGEND: R/W = R = Read only; R/W = Read/Write; W1C = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

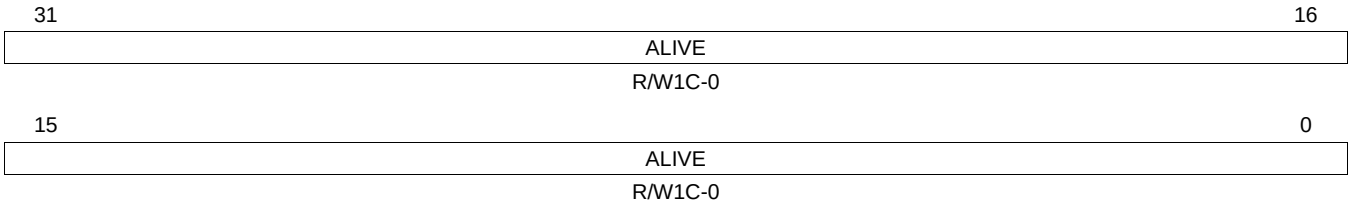
Table 6-76. MDIO Control Register (CONTROL) Field Descriptions

Bit	Field	Value	Description
31	IDLE		State machine IDLE status bit.
		0	State machine is not in idle state.
		1	State machine is in idle state.
30	ENABLE		State machine enable control bit. If the MDIO state machine is active at the time it is disabled, it will complete the current operation before halting and setting the idle bit.
		0	Disables the MDIO state machine.
		1	Enable the MDIO state machine.
29	Reserved	0	Reserved.
28-24	HIGHEST_USER_CHANNEL	0-1Fh	Highest user channel that is available in the module. It is currently set to 1. This implies that MDIOUserAccess1 is the highest available user access channel.
23-21	Reserved	0	Reserved.
20	PREAMBLE		Preamble disable.
		0	Standard MDIO preamble is used.
		1	Disables this device from sending MDIO frame preambles.
19	FAULT		Fault indicator. This bit is set to 1 if the MDIO pins fail to read back what the device is driving onto them. This indicates a physical layer fault and the module state machine is reset. Writing a 1 to it clears this bit, writing a 0 has no effect.
		0	No failure.
		1	Physical layer fault; the MDIO state machine is reset.
18	FAULTENB		Fault detect enable. This bit has to be set to 1 to enable the physical layer fault detection.
		0	Disables the physical layer fault detection.
		1	Enables the physical layer fault detection.
17-16	Reserved	0	Reserved.
15-0	CLKDIV	0-FFFFh	Clock Divider bits. This field specifies the division ratio between the peripheral clock and the frequency of MCLK. MCLK is disabled when CLKDIV is set to 0. MCLK frequency = peripheral clock frequency/(CLKDIV + 1).

6.3.3.3 PHY Acknowledge Status Register (ALIVE)

The PHY acknowledge status register (ALIVE) is shown in Figure 6-77 and described in Table 6-77.

Figure 6-77. PHY Acknowledge Status Register (ALIVE)



LEGEND: R/W = Read/Write; W1C = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

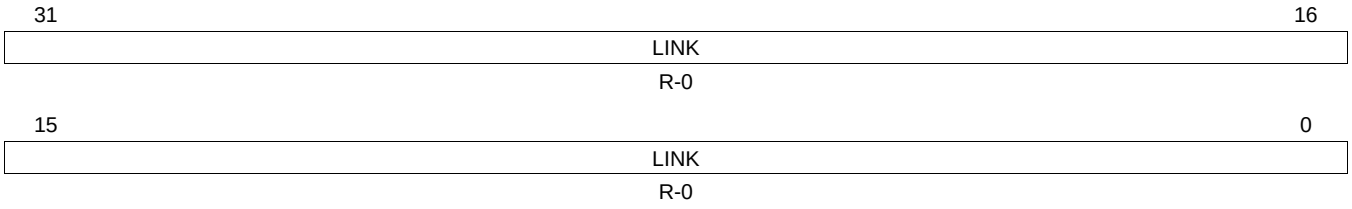
Table 6-77. PHY Acknowledge Status Register (ALIVE) Field Descriptions

Bit	Field	Value	Description
31-0	ALIVE	0-FFFF FFFFh	MDIO Alive bits. Each of the 32 bits of this register is set if the most recent access to the PHY with address corresponding to the register bit number was acknowledged by the PHY; the bit is reset if the PHY fails to acknowledge the access. Both the user and polling accesses to a PHY will cause the corresponding alive bit to be updated. The alive bits are only meant to be used to give an indication of the presence or not of a PHY with the corresponding address. Writing a 1 to any bit will clear it, writing a 0 has no effect.
		0	The PHY fails to acknowledge the access.
		1	The most recent access to the PHY with an address corresponding to the register bit number was acknowledged by the PHY.

6.3.3.4 PHY Link Status Register (LINK)

The PHY link status register (LINK) is shown in Figure 6-78 and described in Table 6-78.

Figure 6-78. PHY Link Status Register (LINK)



LEGEND: R = Read only; -*n* = value after reset

Table 6-78. PHY Link Status Register (LINK) Field Descriptions

Bit	Field	Value	Description
31-0	LINK	0-FFFF FFFFh	MDIO Link state bits. This register is updated after a read of the generic status register of a PHY. The bit is set if the PHY with the corresponding address has link and the PHY acknowledges the read transaction. The bit is reset if the PHY indicates it does not have link or fails to acknowledge the read transaction. Writes to the register have no effect.
		0	The PHY indicates it does not have a link or fails to acknowledge the read transaction.
		1	The PHY with the corresponding address has a link and the PHY acknowledges the read transaction.

6.3.3.5 MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW)

The MDIO link status change interrupt (unmasked) register (LINKINTRAW) is shown in Figure 6-79 and described in Table 6-79.

Figure 6-79. MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		LINKINTRAW	
R-0		R/W1C-0	

LEGEND: R = Read only; R/W = Read/Write; W1C = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

Table 6-79. MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved.
1-0	LINKINTRAW	0-3h	MDIO Link change event, raw value. When asserted, a bit indicates that there was an MDIO link change event (that is, change in the LINK register) corresponding to the PHY address in the USERPHYSEL register. LINKINTRAW[0] and LINKINTRAW[1] correspond to USERPHYSEL0 and USERPHYSEL1, respectively. Writing a 1 will clear the event and writing a 0 has no effect.
		0	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register *n* (USERPHYSEL*n*).

6.3.3.6 MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED)

The MDIO link status change interrupt (masked) register (LINKINTMASKED) is shown in Figure 6-80 and described in Table 6-80.

Figure 6-80. MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		LINKINTMASKED	
R-0		R/W1C-0	

LEGEND: R = Read only; R/W = Read/Write; W1C = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

Table 6-80. MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved.
1-0	LINKINTMASKED	0-3h	MDIO Link change interrupt, masked value. When asserted, a bit indicates that there was an MDIO link change event (that is, change in the LINK register) corresponding to the PHY address in the USERPHYSEL register and the corresponding LINKINTENB bit was set. LINKINTMASKED[0] and LINKINTMASKED[1] correspond to USERPHYSEL0 and USERPHYSEL1, respectively. Writing a 1 will clear the event and writing a 0 has no effect.
		0	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register *n* (USERPHYSEL*n*) and the LINKINTENB bit in USERPHYSEL*n* is set to 1.

6.3.3.7 MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW)

The MDIO user command complete interrupt (unmasked) register (USERINTRAW) is shown in Figure 6-81 and described in Table 6-81.

Figure 6-81. MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		USERINTRAW	
R-0		R/W1C-0	

LEGEND: R = Read only; R/W = Read/Write; W1C = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

Table 6-81. MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved.
1-0	USERINTRAW	0-3h	MDIO User command complete event bits. When asserted, a bit indicates that the previously scheduled PHY read or write command using that particular USERACCESS register has completed. USERINTRAW[0] and USERINTRAW[1] correspond to USERACCESS0 and USERACCESS1, respectively. Writing a 1 will clear the event and writing a 0 has no effect.
		0	No MDIO user command complete event.
		1	The previously scheduled PHY read or write command using MDIO user access register *n* (USERACCESS*n*) has completed.

6.3.3.8 MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED)

The MDIO user command complete interrupt (masked) register (USERINTMASKED) is shown in Figure 6-82 and described in Table 6-82.

Figure 6-82. MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		USERINTMASKED	
R-0		R/W1C-0	

LEGEND: R = Read only; R/W = Read/Write; W1C = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

Table 6-82. MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved.
1-0	USERINTMASKED	0-3h	Masked value of MDIO User command complete interrupt. When asserted, a bit indicates that the previously scheduled PHY read or write command using that particular USERACCESS register has completed and the corresponding USERINTMASKSET bit is set to 1. USERINTMASKED[0] and USERINTMASKED[1] correspond to USERACCESS0 and USERACCESS1, respectively. Writing a 1 will clear the interrupt and writing a 0 has no effect.
		0	No MDIO user command complete event.
		1	The previously scheduled PHY read or write command using MDIO user access register *n* (USERACCESS*n*) has completed and the corresponding bit in USERINTMASKSET is set to 1.

6.3.3.9 MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET)

The MDIO user command complete interrupt mask set register (USERINTMASKSET) is shown in Figure 6-83 and described in Table 6-83.

Figure 6-83. MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		USERINTMASKSET	
R-0		R/W1S-0	

LEGEND: R = Read only; R/W = Read/Write; W1S = Write 1 to set, write of 0 has no effect; -*n* = value after reset

Table 6-83. MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved.
1-0	USERINTMASKSET	0-3h	MDIO user interrupt mask set for USERINTMASKED[1:0], respectively. Setting a bit to 1 will enable MDIO user command complete interrupts for that particular USERACCESS register. MDIO user interrupt for a particular USERACCESS register is disabled if the corresponding bit is 0. USERINTMASKSET[0] and USERINTMASKSET[1] correspond to USERACCESS0 and USERACCESS1, respectively. Writing a 0 to this register has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register *n* (USERACCESS*n*) are disabled.
		1	MDIO user command complete interrupts for the MDIO user access register *n* (USERACCESS*n*) are enabled.

6.3.3.10 MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR)

The MDIO user command complete interrupt mask clear register (USERINTMASKCLEAR) is shown in Figure 6-84 and described in Table 6-84.

Figure 6-84. MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		USERINTMASKCLEAR	
R-0		R/W1C-0	

LEGEND: R = Read only; R/W = Read/Write; W1C = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

Table 6-84. MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved.
1-0	USERINTMASKCLEAR	0-3h	MDIO user command complete interrupt mask clear for USERINTMASKED[1:0], respectively. Setting a bit to 1 will disable further user command complete interrupts for that particular USERACCESS register. USERINTMASKCLEAR[0] and USERINTMASKCLEAR[1] correspond to USERACCESS0 and USERACCESS1, respectively. Writing a 0 to this register has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register *n* (USERACCESS*n*) are enabled.
		1	MDIO user command complete interrupts for the MDIO user access register *n* (USERACCESS*n*) are disabled.

6.3.3.11 MDIO User Access Register 0 (USERACCESS0)

The MDIO user access register 0 (USERACCESS0) is shown in Figure 6-85 and described in Table 6-85.

Figure 6-85. MDIO User Access Register 0 (USERACCESS0)

31	30	29	28 26	25 21	20 16
GO	WRITE	ACK	Reserved	REGADR	PHYADR
R/W1S-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

15 0
DATA
R/W-0

LEGEND: R = Read only; R/W = Read/Write; W1S = Write 1 to set, write of 0 has no effect; -*n* = value after reset

Table 6-85. MDIO User Access Register 0 (USERACCESS0) Field Descriptions

Bit	Field	Value	Description
31	GO	0-1	Go bit. Writing a 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so; this is not an instantaneous process. Writing a 0 to this bit has no effect. This bit is writeable only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to the USERACCESS0 register are blocked when the GO bit is 1.
30	WRITE		Write enable bit. Setting this bit to 1 causes the MDIO transaction to be a register write; otherwise, it is a register read.
		0	The user command is a read operation.
		1	The user command is a write operation.
29	ACK	0-1	Acknowledge bit. This bit is set if the PHY acknowledged the read transaction.
28-26	Reserved	0	Reserved.
25-21	REGADR	0-1Fh	Register address bits. This field specifies the PHY register to be accessed for this transaction.
20-16	PHYADR	0-1Fh	PHY address bits. This field specifies the PHY to be accessed for this transaction.
15-0	DATA	0-FFFFh	User data bits. These bits specify the data value read from or to be written to the specified PHY register.

6.3.3.12 MDIO User PHY Select Register 0 (USERPHYSEL0)

The MDIO user PHY select register 0 (USERPHYSEL0) is shown in Figure 6-86 and described in Table 6-86.

Figure 6-86. MDIO User PHY Select Register 0 (USERPHYSEL0)

31	16
Reserved	
R-0	

15	8	7	6	5	4	0
Reserved		LINKSEL	LINKINTENB	Reserved	PHYADRMON	
R-0		R/W-0	R/W-0	R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-86. MDIO User PHY Select Register 0 (USERPHYSEL0) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7	LINKSEL		Link status determination select bit. Default value is 0, which implies that the link status is determined by the MDIO state machine. This is the only option supported on this device.
		0	The link status is determined by the MDIO state machine.
		1	Not supported.
6	LINKINTENB		Link change interrupt enable. Set to 1 to enable link change status interrupts for PHY address specified in PHYADRMON. Link change interrupts are disabled if this bit is set to 0.
		0	Link change interrupts are disabled.
		1	Link change status interrupts for PHY address specified in PHYADDRMON bits are enabled.
5	Reserved	0	Reserved.
4-0	PHYADRMON	0-1Fh	PHY address whose link status is to be monitored.

6.3.3.13 MDIO User Access Register 1 (USERACCESS1)

The MDIO user access register 1 (USERACCESS1) is shown in Figure 6-87 and described in Table 6-87.

Figure 6-87. MDIO User Access Register 1 (USERACCESS1)

31	30	29	28 26	25 21	20 16
GO	WRITE	ACK	Reserved	REGADR	PHYADR
R/W1S-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

15 0
DATA
R/W-0

LEGEND: R = Read only; R/W = Read/Write; W1S = Write 1 to set, write of 0 has no effect; -*n* = value after reset

Table 6-87. MDIO User Access Register 1 (USERACCESS1) Field Descriptions

Bit	Field	Value	Description
31	GO	0-1	Go bit. Writing 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so; this is not an instantaneous process. Writing 0 to this bit has no effect. This bit is writeable only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to the USERACCESS0 register are blocked when the GO bit is 1.
30	WRITE		Write enable bit. Setting this bit to 1 causes the MDIO transaction to be a register write; otherwise, it is a register read.
		0	The user command is a read operation.
		1	The user command is a write operation.
29	ACK	0-1	Acknowledge bit. This bit is set if the PHY acknowledged the read transaction.
28-26	Reserved	0	Reserved.
25-21	REGADR	0-1Fh	Register address bits. This field specifies the PHY register to be accessed for this transaction.
20-16	PHYADR	0-1Fh	PHY address bits. This field specifies the PHY to be accessed for this transaction.
15-0	DATA	0-FFFFh	User data bits. These bits specify the data value read from or to be written to the specified PHY register.

6.3.3.14 MDIO User PHY Select Register 1 (USERPHYSEL1)

The MDIO user PHY select register 1 (USERPHYSEL1) is shown in Figure 6-88 and described in Table 6-88.

Figure 6-88. MDIO User PHY Select Register 1 (USERPHYSEL1)

31	16
Reserved	
R-0	

15	8	7	6	5	4	0
Reserved		LINKSEL	LINKINTENB	Reserved	PHYADRMON	
R-0		R/W-0	R/W-0	R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-88. MDIO User PHY Select Register 1 (USERPHYSEL1) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7	LINKSEL		Link status determination select bit. Default value is 0, which implies that the link status is determined by the MDIO state machine. This is the only option supported on this device.
		0	The link status is determined by the MDIO state machine.
		1	Not supported.
6	LINKINTENB		Link change interrupt enable. Set to 1 to enable link change status interrupts for the PHY address specified in PHYADRMON. Link change interrupts are disabled if this bit is set to 0.
		0	Link change interrupts are disabled.
		1	Link change status interrupts for PHY address specified in PHYADDRMON bits are enabled.
5	Reserved	0	Reserved.
4-0	PHYADRMON	0-1Fh	PHY address whose link status is to be monitored.

DDR2/DDR3 Memory Controller

This chapter describes the DDR2/DDR3 memory controller.

Topic **Page**

7.1 Introduction

7.1.1 Overview

The DDR2/3 memory controller is used to interface with JESD79-2E/JESD79-3C standard compliant DDR2/3 SDRAM devices. Memory types such as DDR1 SDRAM, SDR SDRAM, SBSRAM, and asynchronous memories are not supported. The DDR2/3 memory controller is the major memory location for program and data storage.

7.1.2 Features

The DDR2/3 memory controller supports the following features:

- JESD79-2E standard compliant DDR2 SDRAM
- JESD79-3C standard compliant DDR3 SDRAM
- 1024 Mbyte memory space
- Data bus width of 32 or 16 bits
- CAS latencies (DDR2): 3, 4, 5, 6, and 7
- CAS latencies (DDR3): 5, 6, 7, 8, 9, 10, and 11
- Internal banks:1,2,4,8
- Two Chip Select Signals (CS) (see your data manual for number of chip selects supported on your device)
- Burst length: 8
- Burst type: Sequential
- Page sizes: 256, 512, 1024, and 2048
- SDRAM initialization from configuration change
- Self Refresh and Power-Down modes for low power
- Periodic ZQ calibration (DDR3 only)
- Self-refresh mode
- Prioritized refresh
- Programmable refresh rate and backlog counter
- Programmable timing parameters
- Little endian
- Support for Fly-By-Topology board routing for DDR3 interfaces via leveling
- Software read/write leveling (calibration)

7.1.3 Features Not Supported

The following features are not supported on the DDR2/3 memory controller:

- Burst lengths other than 8
- Interleave Burst type
- OCD calibration for DDR2
- Burst chop for DDR3

7.1.4 Industry Standard(s) Compliance Statement

The DDR2/3 memory controller subsystem supports JEDEC standard compliant DDR2 (JESD79-2E) and DDR3 (JESD79-3C) devices. It does not support CAS latency of 2 for DDR2 due to data and command macro limitations. It supports a 128-bit wide OCP interface on the core side for programmability. The subsystem can be used to connect to 16-bit or 32-bit memory devices.

7.2 Architecture

7.2.1 Signal Descriptions

The DDR2/3 memory controller signals are shown in Figure 7-1 and described in Table 7-1. The following features are included:

- The maximum data bus width is 32-bits (DDR[x]_D[31:0]).
- The address bus (DDR[x]_A[14:0]) is 15-bits wide with an additional 3 bank address pins (DDR[x]_BA[2:0]).
- Two differential output clocks (DDR[x]_CLK[y] and $\overline{\text{DDR[x]\_CLK[y]}}$) driven by internal clock sources.
- Command signals: Row and column address strobes ($\overline{\text{DDR[x]\_RAS}}$ and $\overline{\text{DDR[x]\_CAS}}$), write enable strobe ($\overline{\text{DDR[x]\_WE}}$), data strobes (DDR[x]_DQS[3:0] and $\overline{\text{DDR[x]\_DQS[3:0]}}$), and data mask (DDR[x]_DQM[3:0]).
- Two chip select signals ($\overline{\text{DDR[x]\_CS[1:0]}}$) and one clock enable signal (DDR[x]_CKE).
- Two on-die termination output signals (DDR[x]_ODT[1:0]).

Where

x = 0 indicates support of one DDR2/3 memory controller

x = 1:0 indicates support of two DDR2/3 memory controllers

y = 0 indicates support of one chip select or one rank per DDR2/3 memory controller

y = 1:0 indicates support of two chip selects or two ranks per DDR2/3 memory controller

See your device-specific data manual for the number of DDR2/3 controllers and number of chip selects supported per DDR2/3 memory controller.

Figure 7-1. DDR2/3 Memory Controller Signals

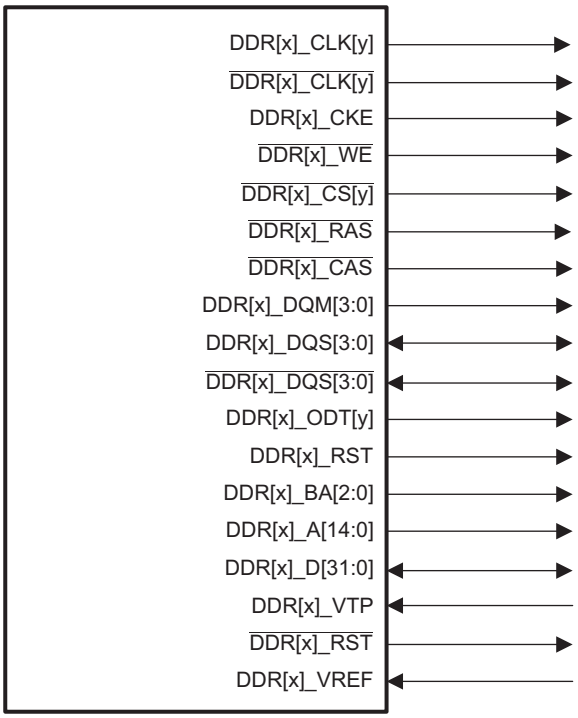


Table 7-1. DDR2/3 Memory Controller Signal Descriptions

Pin	Description
DDR[x]_D[31:0]	Bidirectional data bus. Input for data reads and output for data writes.
DDR[x]_A[14:0]	External address output
$\overline{\text{DDR[x]\_CS[y]}}$	Chip select output. Allows 2 ranks per DDR2/3 controller in $\overline{\text{DDR[x]\_CS[1:0]}}$
DDR[x]_DQM[3:0]	Active-low output data mask.
DDR[x]_CLK[y]/$\overline{\text{DDR[x]\_CLK[y]}}$	Differential clock outputs. All DDR2/3 interface signals are synchronous to these clocks
DDR[x]_CKE	Clock enable. Used to select Power-Down and Self-Refresh operations.
$\overline{\text{DDR[x]\_CAS}}$	Active-low column address strobe.
$\overline{\text{DDR[x]\_RAS}}$	Active-low row address strobe.
$\overline{\text{DDR[x]\_WE}}$	Active-low write enable.
DDR[x]_DQS[3:0]/$\overline{\text{DDR[x]\_DQS[3:0]}}$	Differential data strobe bidirectional signals. Edge-aligned inputs on reads and center-aligned outputs on writes
DDR[x]_ODT[1:0]	On-die termination signals to external DDR2/3 SDRAM
DDR[x]_BA[2:0]	Bank-address control outputs.
DDR[x]_VREF	Memory controller reference voltage. This voltage must be supplied externally. See the device-specific data manual for more details.
DDR[x]_VTP	DDR2/3 VTP Compensation Resistor Connection
$\overline{\text{DDR[x]\_RST}}$	Reset output. Asynchronous reset for DDR3 devices.

7.2.2 Memory Map

See your device-specific data manual for information describing the device memory map.

7.2.3 Clock Control

The DDR2/3 clock is derived directly from the DDR PLL's VCO output. The frequency of DDR[x]_CLK[y] can be determined by using the following formula:

DDR[x]_CLK[y] frequency = (DDRPLL input clock frequency × multiplier) / (pre-divider × post-divider)

The second output clock of the DDR2/3 memory controller $\overline{\text{DDR[x]\_CLK[y]}}$, is the inverse of DDR[x]_CLK[y]. You can change the multiplier, pre-divider and post-divider to get the desired DDR[x]_CLK[y] frequency.

For detailed information on the DDRPLL, see the *Chip Level Resources* chapter.

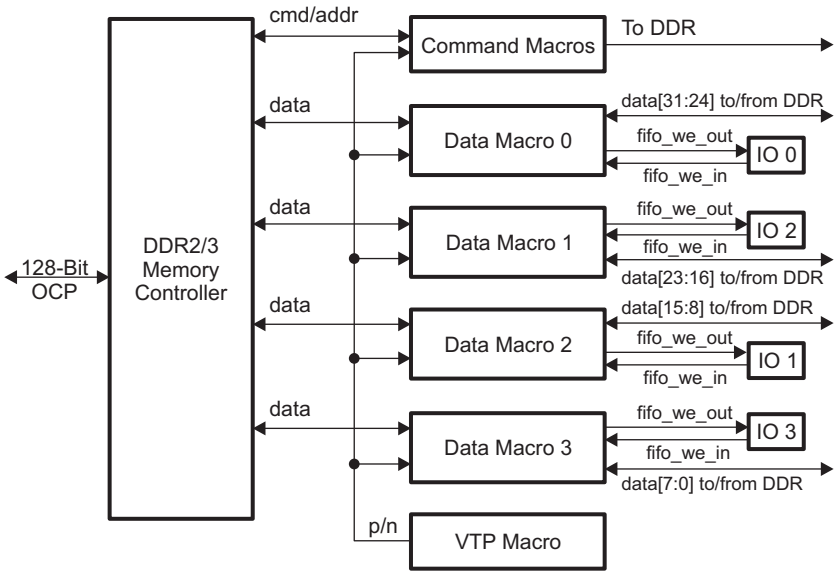
7.2.4 DDR2/3 Memory Controller Subsystem Overview

The DDR2/3 memory controller can gluelessly interface to most standard DDR2/3 SDRAM devices and supports such features as self-refresh mode and prioritized refresh. In addition, it provides flexibility through programmable parameters such as the refresh rate, CAS latency, and many SDRAM timing parameters. The DDR2/3 subsystem consists of:

- DDR2/3 memory controller
- Command macro
- Data macro
- VTP controller macro
- IOs for DQS gate

Figure 7-2 shows the DDR2/3 subsystem block diagram.

Figure 7-2. DDR2/3 Subsytem Block Diagram



Where:

fifo_we_out: DQS enable output for timing match between DQS and system (Memory) clock.

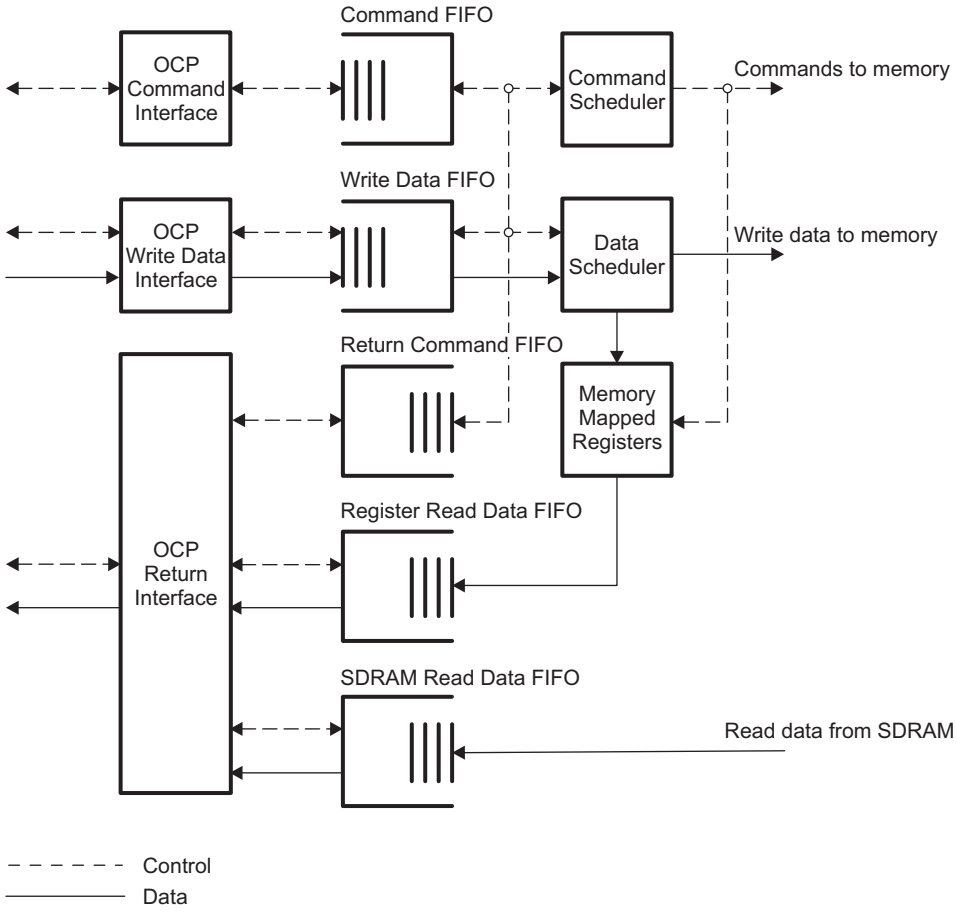
fifo_we_in: DQS enable input for timing match between DQS and system (Memory) clock.

7.2.4.1 DDR2/3 Memory Controller

To move data efficiently from on-chip resources to an external DDR2/3 SDRAM device, the DDR2/3 memory controller makes use of a command FIFO, a write data FIFO, a return command FIFO, and two read data FIFOs. Figure 7-3 shows the block diagram of the DDR2/3 memory controller FIFOs. Commands, write data, and read data arrive at the DDR2/3 memory controller parallel to each other. The same peripheral bus is used to write and read data from external memory as well as internal memory-mapped registers.

- The command FIFO stores all the commands coming in on the OCP command interface.
- The write data FIFO stores the write data for all the write transactions coming in on the OCP write data interface.
- The return command FIFO stores all the return transactions that are to be issued to the OCP return interface. These include the write status return and the read data return commands.
- There are two read data FIFOs that store the read data to be sent to the OCP return interface. One Read Data FIFO stores read data from the memory mapped registers and other Read Data FIFO stores read data from external memory.

Figure 7-3. DDR2/3 Memory Controller FIFO Block Diagram



7.2.4.2 Data Macro

The data macro consists of 8 data channels, one pair of complementary strobes (one pair for 8 bits of data), and one data mask channel (one for 8 bits of data).

The data macros consists of PHY Data Macro, DLLs, and IOs integrated into a macro. The data macro is a bidirectional interface. It is used to transmit data from the memory controller to the external memory chip during a write operation and receive data from memory and transmit it to the memory controller during a read operation.

During a write operation, the data macro translates 32/16-bit words from memory controller to 8-bit words and transmits them at double the bit rate to the memory along with the strobe. The strobe is center-aligned to the data. Data can be prevented from writing to the memory using data mask signal.

During a read operation, the data macro receives 8-bit DDR data along with the strobe and converts it to 32/16- bit words and transmits them to the memory controller along with the read-valid signals.

7.2.4.3 Command Macro

It consists of PHY Command Macro, DLLs, and the IOs integrated together. The command macro acts as a unidirectional macro that transmits address and control bits from memory controller to the memory chip. The clocks DDR[x]_CLK[y] and $\overline{\text{DDR[x]\_CLK[y]}}$ are used by the memory to register the command and address transmitted on the transmit channels. All address and control signals are transmitted clock-centered with respect to DDR[x]_CLK[y] and $\overline{\text{DDR[x]\_CLK[y]}}$. The memory, on the positive edge of DDR[x]_CLK[y] and the negative edge of $\overline{\text{DDR[x]\_CLK[y]}}$, samples all address and control signals.

7.2.4.4 VTP Controller Macro

The VTP controller macro evaluates silicon performance at current voltage, temperature, and process (VTP) in the chip to enable drivers to set constant predetermined output driver impedance. The controller operates by comparing driver impedances to the external reference resistor and adjusting driver impedance to obtain an impedance match. VTP Controller supports the following features

- The VTP controller generates information regarding the Voltage, temperature, and process (VTP) on a chip to be shared with the drivers on the periphery of an ASIC
- Requires a Clock input from the core running at 20 MHz or less.
- 110 clock cycles are needed to assure the VTP outputs are initially set after reset is removed.
- Can be used in static or dynamic update mode of operation.
- The VTP controller has internal noise filtering which allows it to provide continuous VTP bit updates (dynamic update) to the IO macros.

Impedance of the drivers and terminations must be updated often even while in operation, a system referred to as dynamic update. In contrast, static update requires the bus to stop sending data for the impedances to be updated.

Table 7-2. VTP Controller Macro

F2	F1	F0	Description
0	0	0	Off
0	0	1	Update on 5 consecutive update requests
0	1	0	Update on 3 consecutive update requests
0	1	1	Update on 7 consecutive update requests
1	0	0	Update on 2 consecutive update requests
1	0	1	Update on 6 consecutive update requests
1	1	0	Update on 4 consecutive update requests
1	1	1	Update on 8 consecutive update requests

7.2.4.5 DQS-Gate IOs

To effectively model the I/O delay on the DQS gating signal during a read request (the DQS receiver and the CLK driver I/Os), the signal is expected to be looped on a single I/O connecting fifo_we_in to fifo_we_out. The board and memory delay, being fairly constant across PTV variations, are calibrated within the data and command macros using a compensated delay line. The delay need not be modeled, and the loop-back can be done at the die level without bringing the signals out to the package level. A single I/O for fifo_we_in and fifo_we_out is required for every 8-bit data data and command macros.

The data and command macros are responsible for system-level flight time compensation. The following controls are supported by the DDR2/3 controller subsystem:

- Aligning DDR[x]_DQS with respect to DDR[x]_CLK[y] during Write Cycle: For DDR3 operation, initiate the write leveling state machine on each rank in turn to capture the proper delay settings to align DDR[x]_DQS with DDR[x]_CLK[y] clock for each memory chip. If you want to do this manually, write to the control register that controls the delay of DDR[x]_DQS versus DDR[x]_CLK[y] clock position.
- Aligning DDR[x]_DQ[31:0] with respect to DDR[x]_DQS during Write Operation

For each SDRAM part you need to align the DDR[x]_DQS to the center of the DDR[x]_D bits. Set the appropriate bits of the controller to change the delay of the DDR[x]_D signals such that DDR[x]_DQS is aligned in the center of the DQ eye to have proper setup and hold margin.

If you want to do this manually, write to the control register that control the delay of DDR[x]_D versus DDR[x]_DQS. To produce a given amount of skew to center the DDR[x]_D versus DDR[x]_DQS at the SDRAM, the following register is programmed:

- Align FIFO WE Window (Read DQS Gate)

The FIFO WE (Write Enable) signal is used as an input to gate the time where DDR[x]_DQS is valid into the controller on Read cycles to find a delay setting that allows the read data to be captured without errors. The alignment of FIFO WE to the valid Read cycle DDR[x]_DQS window is done internally in the controller PHY.

To adjust the delay of FIFO_WE to center the FIFO WE vs. DDR[x]_DQS valid window at the SDRAM, the data macro 0/1/2/3 DQS gate slave ratio register is programmed.

7.2.5 Address Mapping

The DDR2/3 memory controller views external DDR2/3 SDRAM as one continuous block of memory. This statement is true regardless of the number of memory devices located on the chip select space. The DDR2/3 memory controller receives DDR2/3 memory access requests along with a 32-bit logical address from the rest of the system. In turn, DDR2/3 memory controller uses the logical address to generate a row/page, column, and bank address for the DDR2/3 SDRAM. The number of column, row and bank address bits used is determined by the IBANK, RSIZE, and PAGESIZE fields (see Table 7-3). The DDR2/3 memory controller uses up to 15 bits for the row/page address.

Table 7-3. Address Mapping

Bit Field	Bit Value	Bit Description
RSIZE		Defines the number of address lines to be connected to DDR2/3 memory device
	0	9 row bits
	1h	10 row bits
	2h	11 row bits
	3h	12 row bits
	4h	13 row bits
	5h	14 row bits
	6h	15 row bits
PAGESIZE		Defines the page size of each page of the external DDR2/3 memory device
	0	256 words (requires 8 column address bits)
	1h	512 words (requires 9 column address bits)
	2h	1024 words (requires 10 column address bits)
	3h	2048 words (requires 11 column address bits)
IBANK		Defines the number of internal banks on the external DDR2/3 memory device
	0	1 bank
	1h	2 banks
	2h	4 banks
	3h	8 banks
EBANK		Defines whether DDR2/3 memory controller accesses use 1 or 2 chip selects
	0	CS0 only
	1h	CS0 and CS1

When addressing SDRAM, if the IBANK_POS field in the SDRCR is cleared to 0, the DDR2/3 memory controller uses 3 fields—IBANK, EBANK, and PAGESIZE in the SDRCR to determine the mapping from source address to DDR2/3 memory device row, column, bank and chip select. If IBANK_POS is set to 1, 2, or 3, the DDR2/3 controller uses 4 fields—IBANK, EBANK, PAGESIZE, and RSIZE in the SDRCR to determine the mapping from source address to DDR2/3 memory device row, column, bank, and chip select. In all cases, the DDR2/3 controller considers its DDR2/3 memory device address space to be a single logical block, regardless of the number of physical devices or whether the devices are mapped to 1 chip select or 2 DDR2/3 controller chip selects.

Since the DDR2/3 memory controller interleaves among less number of banks when IBANK_POS ≠ 0 or EBANK_POS = 1, these cases are lower in performance than the IBANK_POS = 0 case. Thus, these cases are only recommended to be used along with partial array self-refresh where performance can be traded off for power savings.

7.2.5.1 Address Mapping When IBANK_POS = 0 and EBANK_POS = 0

For IBANK_POS = 0 and EBANK_POS = 0 (see Table 7-4), the effect of the address mapping scheme is that as the source address increments across DDR2/3 memory device page boundaries, the DDR2/3 controller moves onto the same page in the next bank in the current device $\overline{\text{DDR[x]\_CS[0]}}$. This movement along the banks of the current proceeds to the same page in the next device (if EBANK = 1, $\overline{\text{DDR[x]\_CS[1]}}$) and proceeds through the same page in all its banks before moving over to the next page in the first device ($\overline{\text{DDR[x]\_CS[0]}}$). The DDR2/3 controller exploits this traversal across internal banks and chip selects while remaining on the same page to maximize the number of open DDR2/3 memory device banks within the overall DDR2/3 memory device space.

Thus, the DDR2/3 controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, and can interleave among all of them.

Table 7-4. OCP Address to DDR2/3 Address Mapping for IBANK_POS = 0 and EBANK_POS = 0

Logical Address			
Row Address	**Chip Select**	**Bank Address**	**Column Address**
15 bits	Number of bits defined by EBANK of SDRCR:	Number of bits defined by IBANK of SDRCR:	Number of bits defined by PAGESIZE of SDRCR:
	EBANK = 0 => 0 bits	IBANK = 0 => 0 bits	PAGESIZE = 0 => 8 bits
	EBANK = 1 => 1 bit	IBANK = 1 => 1 bit	PAGESIZE = 1 => 9 bits
		IBANK = 2 => 2 bits	PAGESIZE = 2 => 10 bits
		IBANK = 3 => 3 bits	PAGESIZE = 3 => 11 bits

7.2.5.2 Address Mapping when IBANK_POS = 1 and EBANK_POS = 0

For IBANK_POS = 1 and EBANK_POS = 0 (see Table 7-5), the interleaving of banks within a device (per chip select) is limited to 4 banks. However, it can still interleave banks between the two chip selects. Thus, the DDR2/3 controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among 8 of them.

Table 7-5. OCP Address to DDR2/3 Address Mapping for IBANK_POS = 1 and EBANK_POS = 0

Logical Address				
Bank Address[2]	**Row Address**	**Chip Select**	**Bank Address[1:0]**	**Column Address**
Number of bits defined by IBANK of SDRCR:	Number of bits defined by RSIZE of SDRCR:	Number of bits defined by EBANK of SDRCR:	Number of bits defined by IBANK of SDRCR:	Number of bits defined by PAGESIZE of SDRCR:
IBANK = 0 => 0 bits	RSIZE = 0 => 9 bits	EBANK = 0 => 0 bits	IBANK = 0 => 0 bits	PAGESIZE = 0 => 8 bits
IBANK = 1 => 0 bits	RSIZE = 1 => 10 bits	EBANK = 1 => 1 bit	IBANK = 1 => 1 bit	PAGESIZE = 1 => 9 bits
IBANK = 2 => 0 bits	RSIZE = 2 => 11 bits		IBANK = 2 => 2 bits	PAGESIZE = 2 => 10 bits
IBANK = 3 => 1 bit	RSIZE = 3 => 12 bits		IBANK = 3 => 2 bits	PAGESIZE = 3 => 11 bits
	RSIZE = 4 => 13 bits			
	RSIZE = 5 => 14 bits			
	RSIZE = 6 => 15 bits			

7.2.5.3 Address Mapping when IBANK_POS = 2 and EBANK_POS = 0

For IBANK_POS = 2 and EBANK_POS = 0 (see Table 7-6), the interleaving of banks within a device (per chip select) is limited to 2 banks. However, it can still interleave banks between the two chip selects. Thus, the DDR2/3 controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among 4 of them.

Table 7-6. OCP Address to DDR2/3 Address Mapping for IBANK_POS = 2 and EBANK_POS = 0

Logical Address				
Bank Address[2:1]	**Row Address**	**Chip Select**	**Bank Address[0]**	**Column Address**
Number of bits defined by IBANK of SDRCR:	Number of bits defined by RSIZE of SDRCR:	Number of bits defined by EBANK of SDRCR:	Number of bits defined by IBANK of SDRCR:	Number of bits defined by PAGESIZE of SDRCR:
IBANK = 0 => 0 bits	RSIZE = 0 => 9 bits	EBANK = 0 => 0 bits	IBANK = 0 => 0 bits	PAGESIZE = 0 => 8 bits
IBANK = 1 => 0 bits	RSIZE = 1 => 10 bits	EBANK = 1 => 1 bit	IBANK = 1 => 1 bit	PAGESIZE = 1 => 9 bits
IBANK = 2 => 1 bits	RSIZE = 2 => 11 bits		IBANK = 2 => 1 bits	PAGESIZE = 2 => 10 bits
IBANK = 3 => 2 bit	RSIZE = 3 => 12 bits		IBANK = 3 => 1 bits	PAGESIZE = 3 => 11 bits
	RSIZE = 4 => 13 bits			
	RSIZE = 5 => 14 bits			
	RSIZE = 6 => 15 bits			

7.2.5.4 Address Mapping when IBANK_POS = 3 and EBANK_POS = 0

For IBANK_POS = 3 and EBANK_POS = 0 (see Table 7-7), the DDR2/3 controller cannot interleave banks within a device (per chip select). However, it can still interleave banks between the two chip selects. Thus, the DDR2/3 controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among 2 of them.

Table 7-7. OCP Address to DDR2/3 Address Mapping for IBANK_POS = 3 and EBANK_POS = 0

Logical Address			
Bank Address	**Row Address**	**Chip Select**	**Column Address**
Number of bits defined by IBANK of SDRCR:	Number of bits defined by RSIZE of SDRCR:	Number of bits defined by EBANK of SDRCR:	Number of bits defined by PAGESIZE of SDRCR:
IBANK = 0 => 0 bits	RSIZE = 0 => 9 bits	EBANK = 0 => 0 bits	PAGESIZE = 0 => 8 bits
IBANK = 1 => 1 bit	RSIZE = 1 => 10 bits	EBANK = 1 => 1 bit	PAGESIZE = 1 => 9 bits
IBANK = 2 => 2 bits	RSIZE = 2 => 11 bits		PAGESIZE = 2 => 10 bits
IBANK = 3 => 3 bits	RSIZE = 3 => 12 bits		PAGESIZE = 3 => 11 bits
	RSIZE = 4 => 13 bits		
	RSIZE = 5 => 14 bits		
	RSIZE = 6 => 15 bits		

7.2.5.5 Address Mapping when IBANK_POS = 0 and EBANK_POS = 1

For IBANK_POS = 0 and EBANK_POS = 1 (see Table 7-8), the DDR2/3 memory controller interleaves among all the banks within a device (per chip select). However, the memory controller cannot interleave banks between the two chip selects. Thus, the DDR2/3 memory controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among 8 of them.

Table 7-8. OCP Address to DDR2/3 Address Mapping for IBANK_POS = 0 and EBANK_POS = 1

Logical Address			
Chip Select	**Row Address**	**Bank Address**	**Column Address**
Number of bits defined by EBANK of SDRCR:	Number of bits defined by RSIZE of SDRCR:	Number of bits defined by IBANK of SDRCR:	Number of bits defined by PAGESIZE of SDRCR:
EBANK = 0 => 0 bits	RSIZE = 0 => 9 bits	IBANK = 0 => 0 bits	PAGESIZE = 0 => 8 bits
EBANK = 1 => 1 bit	RSIZE = 1 => 10 bits	IBANK = 1 => 1 bit	PAGESIZE = 1 => 9 bits
	RSIZE = 2 => 11 bits	IBANK = 2 => 2 bits	PAGESIZE = 2 => 10 bits
	RSIZE = 3 => 12 bits	IBANK = 3 => 3 bits	PAGESIZE = 3 => 11 bits
	RSIZE = 4 => 13 bits		
	RSIZE = 5 => 14 bits		
	RSIZE = 6 => 15 bits		

7.2.5.6 Address Mapping when IBANK_POS = 1 and EBANK_POS = 1

For IBANK_POS = 1 and EBANK_POS = 1 (see Table 7-9), the interleaving of banks within a device (per chip select) is limited to 4 banks. Also, the DDR2/3 memory controller cannot interleave banks between the two chip selects. Thus, the memory controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among 4 of them.

Table 7-9. OCP Address to DDR2/3 Address Mapping for IBANK_POS = 1 and EBANK_POS = 1

Logical Address				
Chip Select	**Bank Address[2]**	**Row Address**	**Bank Address[0]**	**Column Address**
Number of bits defined by EBANK of SDRCR:	Number of bits defined by IBANK of SDRCR:	Number of bits defined by RSIZE of SDRCR:	Number of bits defined by IBANK of SDRCR:	Number of bits defined by PAGESIZE of SDRCR:
EBANK = 0 => 0 bits	IBANK = 0 => 0 bits	RSIZE = 0 => 9 bits	IBANK = 0 => 0 bits	PAGESIZE = 0 => 8 bits
EBANK = 1 => 1 bit	IBANK = 1 => 0 bits	RSIZE = 1 => 10 bits	IBANK = 1 => 1 bit	PAGESIZE = 1 => 9 bits
	IBANK = 2 => 0 bits	RSIZE = 2 => 11 bits	IBANK = 2 => 2 bits	PAGESIZE = 2 => 10 bits
	IBANK = 3 => 1 bit	RSIZE = 3 => 12 bits	IBANK = 3 => 2 bits	PAGESIZE = 3 => 11 bits
		RSIZE = 4 => 13 bits		
		RSIZE = 5 => 14 bits		
		RSIZE = 6 => 15 bits		

7.2.5.7 Address Mapping when IBANK_POS = 2 and EBANK_POS = 1

For IBANK_POS= 2 and EBANK_POS= 1 (see Table 7-10), the interleaving of banks within a device (per chip select) is limited to 2 banks. Also, the DDR2/3 memory controller cannot interleave banks between the two chip selects. Thus, the memory controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among 2 of them.

Table 7-10. OCP Address to DDR2/3 Address Mapping for IBANK_POS = 2 and EBANK_POS = 1

Logical Address				
Chip Select	**Bank Address[2]**	**Row Address**	**Bank Address[0]**	**Column Address**
Number of bits defined by EBANK of SDRCR:	Number of bits defined by IBANK of SDRCR:	Number of bits defined by RSIZE of SDRCR:	Number of bits defined by IBANK of SDRCR:	Number of bits defined by PAGESIZE of SDRCR:
EBANK = 0 => 0 bits	IBANK = 0 => 0 bits	RSIZE = 0 => 9 bits	IBANK = 0 => 0 bits	PAGESIZE = 0 => 8 bits
EBANK = 1 => 1 bit	IBANK = 1 => 0 bits	RSIZE = 1 => 10 bits	IBANK = 1 => 1 bit	PAGESIZE = 1 => 9 bits
	IBANK = 2 => 1 bit	RSIZE = 2 => 11 bits	IBANK = 2 => 1 bit	PAGESIZE = 2 => 10 bits
	IBANK = 3 => 2 bits	RSIZE = 3 => 12 bits	IBANK = 3 => 1 bit	PAGESIZE = 3 => 11 bits
		RSIZE = 4 => 13 bits		
		RSIZE = 5 => 14 bits		
		RSIZE = 6 => 15 bits		

7.2.5.8 Address Mapping when IBANK_POS = 3 and EBANK_POS = 1

For IBANK_POS= 3 and EBANK_POS= 1 (see Table 7-11), the DDR2/3 memory controller cannot interleave banks within a device (per chip select) or between the two chip selects. Thus, the memory controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but cannot interleave among any of them.

Table 7-11. OCP Address to DDR2/3 Address Mapping for IBANK_POS = 3 and EBANK_POS = 1

Logical Address			
Chip Select	**Bank Address**	**Row Address**	**Column Address**
Number of bits defined by EBANK of SDRCR:	Number of bits defined by IBANK of SDRCR:	Number of bits defined by RSIZE of SDRCR:	Number of bits defined by PAGESIZE of SDRCR:
EBANK = 0 => 0 bits	IBANK = 0 => 0 bits	RSIZE = 0 => 9 bits	PAGESIZE = 0 => 8 bits
EBANK = 1 => 1 bit	IBANK = 1 => 1 bit	RSIZE = 1 => 10 bits	PAGESIZE = 1 => 9 bits
	IBANK = 2 => 2 bits	RSIZE = 2 => 11 bits	PAGESIZE = 2 => 10 bits
	IBANK = 3 => 3 bits	RSIZE = 3 => 12 bits	PAGESIZE = 3 => 11 bits
		RSIZE = 4 => 13 bits	
		RSIZE = 5 => 14 bits	
		RSIZE = 6 => 15 bits	

7.2.6 Performance Management

7.2.6.1 Command Ordering and Scheduling

The DDR2/3 memory controller performs command re-ordering and scheduling in an attempt to achieve efficient transfers with maximum throughput. The goal is to maximize the utilization of the data, address, and command buses while hiding the overhead of opening and closing DDR2/3 SDRAM rows. Command re-ordering takes place within the command FIFO.

The DDR2/3 memory controller examines all the commands stored in the command FIFO to schedule commands to the external memory. For each master, the DDR2/3 memory controller reorders the commands based on the following rules:

- Selects the oldest command
- A read command is advanced before an older write command if the read is to a different block address (2048 bytes) and the read priority is equal to or greater than the write priority.

The second bullet above may be viewed as an exception to the first bullet. This means that for an individual master, all of its commands complete from oldest to newest, with the exception that a read may be advanced ahead of an older, lower or equal priority write. Following this scheduling, each master may have one command ready for execution.

Next, the DDR2/3 memory controller examines each of the commands selected by the individual masters and performs the following reordering:

- Among all pending reads, selects reads to rows already open. Among all pending writes, selects writes to rows already open.
- Selects the highest priority command from pending reads and writes to open rows. If multiple commands have the highest priority, then the DDR2 memory controller selects the oldest command.

The DDR2/3 memory controller may now have a final read and write command. If the Read FIFO is not full, then the read command is performed before the write command; otherwise, the write command is performed first.

Besides commands received from on-chip resources, the DDR2/3 memory controller also issues refresh commands. The DDR2/3 memory controller attempts to delay refresh commands as long as possible to maximize performance while meeting the SDRAM refresh requirements. As the DDR2/3 memory controller issues read, write, and refresh commands to DDR2/3 SDRAM device, it follows the following priority scheme:

1. (Highest priority) SDRAM refresh request due to Refresh Must level of refresh urgency reached (see Section 7.2.6.4)
2. Request for a read or a write.
3. SDRAM Activate commands.
4. SDRAM Deactivate commands.
5. SDRAM Deep Power-Down request.
6. SDRAM clock stop or Power-Down request.
7. SDRAM refresh request due to Refresh May or Release level of refresh urgency reached (see Section 7.2.6.4)
8. (Lowest priority) SDRAM self-refresh request.

7.2.6.2 Command Starvation

The reordering and scheduling rules listed in Section 7.2.6.1 may lead to command starvation, which is the prevention of certain commands from being processed by the DDR2/3 memory controller. Command starvation results from the following conditions:

- A continuous stream of high-priority read commands can block a low-priority write command
- A continuous stream of DDR2/3 SDRAM commands to a row in an open bank can block commands to the closed row in the same bank.

To avoid these conditions, the DDR2/3 memory controller can momentarily raise the priority of the oldest command in the command FIFO after a set number of transfers have been made. The PR_OLD_COUNT field in the peripheral bus burst priority register (PBBPR) sets the number of the transfers that must be made before the DDR2/3 memory controller raises the priority of the oldest command.

7.2.6.3 Possible Race Condition

A race condition may exist when certain masters write data to the DDR2/3 memory controller. For example, if master A passes a software message via a buffer in DDR2/3 memory and does not wait for indication that the write completes, when master B attempts to read the software message it may read stale data and therefore receive an incorrect message. In order to confirm that a write from master A has landed before a read from master B is performed, master A must wait for the write completion status from the DDR2/3 memory controller before indicating to master B that the data is ready to be read. If master A does not wait for indication that a write is complete, it must perform the following workaround:

1. Perform the required write.
2. Perform a dummy write to the DDR2 memory controller module ID and revision register.
3. Perform a dummy read to the DDR2 memory controller module ID and revision register.
4. Indicate to master B that the data is ready to be read after completion of the read in step 3.

The completion of the read in step 3 ensures that the previous write was done.

For a list of the master peripherals that need this workaround, see the device-specific data sheet.

7.2.6.4 Refresh Scheduling

The DDR2/3 memory controller issues AUTO REFRESH commands to DDR2/3 SDRAM devices at a rate defined in the refresh rate (RR) field in the SDRAM refresh control register (SDRRCR). The controller uses two counters to schedule AUTO REFRESH commands: a 13-bit decrementing refresh interval counter and a 4-bit refresh backlog counter. The refresh interval counter is loaded with the value of the RR field and decrements by 1 each cycle until it reaches zero. Once the interval counter reaches zero, it reloads with the value of the RR field. Each time the interval counter expires, a refresh backlog counter increments by 1. Conversely, each time the DDR2/3 memory controller performs a AUTO REFRESH command, the backlog counter decrements by 1. This means the refresh backlog counter records the number of AUTO REFRESH commands the memory controller currently has outstanding. For the range of values that the backlog counter can take, there are three levels of urgency with which the controller should perform an auto refresh cycle as listed in Table 7-12.

The refresh counters do not operate when the memory controller is in self-refresh mode.

Table 7-12. Refresh Scheduling

Urgency Level	Description
Refresh May	Whenever the backlog count is greater than 0, to indicate that there is a refresh backlog, so if the controller is not busy and none of the SDRAM banks are open, it should perform an auto refresh cycle.
Refresh Release	Whenever the backlog count is greater than 4, to indicate that the refresh backlog is getting high, so if the controller is not busy it should perform an auto refresh cycle even if any banks are open.
Refresh Must	Whenever the backlog count is greater than 7, to indicate that the refresh backlog is getting excessive and the controller should perform an auto refresh cycle before servicing any new memory access requests. The controller starts servicing new memory accesses after Refresh Release level is cleared.

7.2.6.5 Performance Counters

The PRFCNT1 and PRFCNT2 registers are used to monitor or calculate the DDR2/DDR3 Controller bandwidth and efficiency. These counters are able to count events such as total SDRAM accesses, SDRAM activates, reads, writes, and other events. Each counter counts independent of the other counters. In addition to the ability to count events, the counters can also filter the events from a particular master or address space. The events counting and filter enabling are configured using the PRFCNTCFG register. The filter value used is configured through the PRFCNTSEL register. Each counter can be configured independently.

Table 7-13 lists all the events that can be counted and whether a filter can be applied to a particular event. A filter is applied to an event if the following bits are set to 0x1 for that event:

- For Performance Counter 1: PRFCNTCFG[15] CNTR1_MCONNID_EN and PRFCNTCFG[14] CNTR1_REGION_EN;
- For Performance Counter 2: PRFCNTCFG[31] CNTR2_MCONNID_EN and PRFCNTCFG[30] CNTR2_REGION_EN.

Table 7-13. Performance Counter Filter Configuration

CNTRn_CFG	CNTRn_REGION_EN	CNTRn_MCONNID_EN	Description
0x0	0x0	0x0 or 0x1	Count total SDRAM accesses
0x1	0x0	0x0 or 0x1	Count total SDRAM activates
0x2	0x0 or 0x1	0x0 or 0x1	Count total reads
0x3	0x0 or 0x1	0x0 or 0x1	Count total writes
0x4	0x0	0x0	Count the number of DDR2/DDR3 Controller functional clock cycles during which the L3 Command FIFO is full
0x5	0x0	0x0	Count the number of DDR2/DDR3 Controller functional clock cycles during which the L3 Write Data FIFO is full
0x6	0x0	0x0	Count the number of DDR2/DDR3 Controller functional clock cycles during which the L3 Read Data FIFO is full
0x7	0x0	0x0	Count the number of DDR2/DDR3 Controller functional clock cycles during which the L3 Return Command FIFO is full
0x8	0x0 or 0x1	0x0 or 0x1	Count the number of priotity elevations
0x9	0x0	0x0	Count the number of DDR2/DDR3 Controller functional clock cycles that a command was pending
0xA	0x0	0x0	Count the number of DDR2/DDR3 Controller functional clock cycles for which the memory data bus was transferring data.
0xB - 0xF	0x0	0x0	Reserved for future use.

7.2.6.5.1 Performance Counters General Examples

- **General Example for Counting All Write Accesses**
  PRFCNT1 register needs to count all write accesses from master with connection ID equal to 0xA (this is the system MMU). To enable the writes counting, the PRFCNTCFG[3:0] CNTR1_CFG bit field must be set to 0x3. The PRFCNTSEL[15:8] MCONNID1 bit field must be set to 0xA. Finally, to enable filtering, the PRFCNTCFG[15] CNTR1_MCONNID_EN bit must be set to 0x1. With this configuration PRFCNT1 will count every write made to the SDRAM from master 0xA to any address space. This will not include accesses from other masters and will not include commands other than writes.

- **General Example for Counting Total Access**
  PRFCNT2 register needs to count total accesses to the SDRAM regardless of the address space or master. To enable the counting of all accesses to the SDRAM, the PRFCNTCFG[19:16] CNTR2_CFG bit field must be set to 0x0. Finally, to disable filtering, both the PRFCNTCFG[31] CNTR2_MCONNID_EN and PRFCNTCFG[30] CNTR2_REGION_EN bits must be set to 0x0. With this configuration PRFCNT2 will count every access made to the SDRAM. This will include all accesses from all masters and to any address space.
- **General Example for Counting All Read Accesses**
  PRFCNT1 register needs to count all read accesses from master with connection ID equal to 0xA (this is the system MMU) to address space 0x0. To enable the reads counting, the PRFCNTCFG[3:0] CNTR1_CFG bit field must be set to 0x2. The PRFCNTSEL[15:8] MCONNID1bit field must be set to 0xA and the PRFCNTSEL[1:0] REGION_SEL1 bit field must be set to 0x0. Finally, to enable filtering, both the PRFCNTCFG[15] CNTR1_MCONNID_EN and the PRFCNTCFG[14] CNTR1_REGION_EN bits must be set to 0x1. With this configuration, PRFCNT1 will count every read made to the SDRAM from master 0xA to address space 0x0. This will not include accesses from other masters or accesses to other address spaces and will not include commands other than reads.

7.2.7 DDR3 Software Read-Write Leveling

The memory controller supports only software leveling. Software leveling is a procedure by which the time delays between DDR3 signals can be compensated by appropriate values programmed in the corresponding slave ratio registers of the DDR PHY. For description of these registers, see Table 7-35. Software leveling can be done independently for each data and command macro. When software leveling is used there is no need to enable the leveling feature of the external DDR3 SDRAM, which is controlled by one of its MR registers. However, during write leveling the ODT function must be on. Proper ODT values must be turned on at the external memory side by setting Rtt_Nom (A9,A6,A2) bits in external DDR3 MR1 register.

For more information about the concept of the software leveling, see http://processors.wiki.ti.com/index.php/TI814x-DDR3-Init-U-Boot.

7.2.8 PRCM Sequence for DDR2/3 Memory Controller

Following is the power, reset, and clock management (PRCM) module sequence to enable the clocks to the DDR2/3 memory controller:

1. Enable DDR2/3 clock by writing 2h to PRCM CM_DEFAULT_FW_CLKCTRL register.
2. Enable L3 interconnect fast clock by writing 2h to PRCM CM_DEFAULT_L3_FAST_CLKSTCTRL register.
3. Enable DDR2/3 controller 0 interface clocks by writing 2h to PRCM CM_DEFAULT_EMIF_0_CLKCTRL register.
4. Enable DDR2/3 controller 1 interface clocks by writing 2h to PRCM CM_DEFAULT_EMIF_1_CLKCTRL register.
5. Poll for 1 in CLKACTIVITY_L3_FAST_GCLK and CLKACTIVITY_DDR_GCLK bits of PRCM CM_DEFAULT_L3_FAST_CLKSTCTRL register (indicates clocks are active).
6. Poll for 0 in IDLEST bit of PRCM CM_DEFAULT_EMIF_0_CLKCTRL register (indicates module is fully functional including OCP).
7. Poll for 0 in IDLEST bit of PRCM CM_DEFAULT_EMIF_1_CLKCTRL register (indicates module is fully functional including OCP).
8. Enable DMM module clocks by writing 2h to PRCM CM_DEFAULT_DMM_CLKCTRL register.
9. Poll for 0 in IDLEST bit of PRCM CM_DEFAULT_DMM_CLKCTRL register (indicates module is fully functional including OCP).

Refer to the *Power, Reset, and Clock Management (PRCM) Module* chapter for the PRCM register details.

7.2.9 Interrupt Support

The DDR2/3 controller is compliant with Open Core Protocol Specification (OCP-IP 2.2). The controller supports only Idle, Write, Read, and WriteNonPost command types. Also, the controller supports only incrementing, wrapping, and 2-dimensional block addressing modes. The controller supports generations of an error interrupt, if an unsupported command or a command with unsupported addressing mode is received.

7.2.10 EDMA Event Support

The DDR2/3 memory controller is a DMA slave peripheral and therefore does not generate EDMA events. Data read and write requests may be made directly by masters including the EDMA controller.

7.2.11 Emulation Considerations

The DDR2/3 memory controller remains fully functional during emulation halts to allow emulation access to external memory.

7.2.12 Power Management

This section defines the power management capabilities and requirements.

7.2.12.1 Self-Refresh Mode

The DDR2/3 memory controller supports self-refresh mode for low power. The memory controller automatically puts the SDRAM into self-refresh after the memory controller is idle for SR_TIM number of DDR clock cycles and the LP_MODE field is set to 2h. The LP_MODE and SR_TIM fields are programmed in the power management control register (PMCR). The memory controller completes all pending refreshes before it puts the DRAM into self-refresh. Therefore, after the expiration of reg_sr_tim, the memory controller starts issuing refreshes to complete the refresh backlog, and then issues a SELF-REFRESH command to the SDRAM.

In self-refresh mode, the memory controller automatically stops the clocks DDR[x]_CLK[y] to the SDRAM. The memory controller maintains DDR[x]_CKE low to maintain the self-refresh state. When the SDRAM is in self-refresh, the memory controller services register accesses as normal. If the LP_MODE field is not equal to 2 or an SDRAM access is requested while it is in self-refresh, and T_CKE + 1 cycles have elapsed since the SELF-REFRESH command was issued, the memory controller brings the SDRAM out of self-refresh. The value of T_CKE is taken from the SDRAM timing 2 register (SDRTIM2).

The exit sequence of self-refresh mode for DDR2 devices. The memory controller:

- Enables clocks.
- Drives DDR[x]_CKE high.
- Waits for T_XSNR + 1 cycles. The value of T_XSNR is taken from SDRAM timing 2 register (SDRTIM2).
- If the DLL bit in the SDRAM configuration register (SDRCR) is 1, issues a LOAD MODE REGISTER command to the extended mode register 1 with the DDR[x]_A[15:0] bits set as:

Bits	Value	Description
DDR[x]_A[15:13]	0	Reserved
DDR[x]_A[12]	0	Output buffer enabled
DDR[x]_A[11]	0	Reserved
DDR[x]_A[10]	!DDQS	Differential DQS enable value from SDRAM configuration register
DDR[x]_A[9:7]	0	Exit OCD calibration mode
DDR[x]_A[6]	DDRTERM[1]	DDR2 termination resistor value from SDRAM configuration register
DDR[x]_A[5:3]	0	Additive latency = 0
DDR[x]_A[2]	DDRTERM[0]	DDR2 termination resistor value from SDRAM configuration register
DDR[x]_A[1]	DRIVE	SDRAM drive strength from SDRAM configuration register
DDR[x]_A[0]	1	Disable DLL

- Starts an auto-refresh cycle in the next cycle.
- Enters its idle state and can issue any other commands except a write or a read. A write or a read is only issued after T_XSRD + 1 clock cycles have elapsed since DDR[x]_CKE is driven high. The value of T_XSRD is taken from SDRTIM2.

The exit sequence of self-refresh mode for DDR3 devices. The memory controller:

- Enables clocks.
- Drives DDR[x]_CKE high.
- Waits for T_XSNR + 1 cycles. The value of T_XSNR is taken from SDRAM timing 2 register (SDRTIM2).
- If the DLL bit in the SDRAM configuration register (SDRCR) is 1, issues a LOAD MODE REGISTER command to the extended mode register 1 with the pad_a_o bits set as:

Bits	Value	Description
DDR[x]_A[15:13]	0	Reserved
DDR[x]_A[12]	0	Output buffer enabled
DDR[x]_A[11]	0	Reserved
DDR[x]_A[10]	0	Reserved
DDR[x]_A[9]	DDRTERM[2]	DDR3 termination resistor value from SDRAM configuration register
DDR[x]_A[8]	0	Reserved
DDR[x]_A[7]	0	Write leveling disabled
DDR[x]_A[6]	DDRTERM[1]	DDR3 termination resistor value from SDRAM configuration register
DDR[x]_A[5]	DRIVE[1]	SDRAM drive strength from SDRAM configuration register
DDR[x]_A[4:3]	0	Additive latency = 0
DDR[x]_A[2]	DDRTERM[0]	DDR3 termination resistor value from SDRAM configuration register
DDR[x]_A[1]	DRIVE[0]	SDRAM drive strength from SDRAM configuration register
DDR[x]_A[0]	1	Disable DLL

- Starts an auto-refresh cycle in the next cycle.
- Performs one write incremental leveling.
- Performs read DQS incremental training.
- Performs read data-eye incremental training.
- Enters its idle state and can issue any other commands except a write or a read. A write or a read is only issued after T_XSRD + 1 clock cycles have elapsed since DDR[x]_CKE is driven high. The value of T_XSRD is taken from SDRTIM2.

7.2.12.2 Power Down Mode

The memory controller also supports Power-Down mode for low power. The memory controller automatically puts the SDRAM into Power-Down after the memory controller is idle for PD_TIM number of DDR clock cycles and the LP_MODE field is set to 4h. The LP_MODE and PD_TIM fields are programmed in the power management control register (PMCR). If the Refresh Must Level is not reached before the entry into Power-Down, the memory controller does not precharge all banks before issuing the POWER-DOWN command. This results in SDRAM entering Active Power-Down mode.

If the Refresh Must Level is reached before the entry into Power-Down, the memory controller precharges all banks and issues refreshes until the Refresh Release Level is reached before issuing the POWER-DOWN command. This results in SDRAM entering Precharge Power-Down mode.

In Power-Down mode, the memory controller does not stop the clocks DDR[x]_CLK[y] to the SDRAM. The memory controller maintains DDR[x]_CKE low to maintain the Power-Down state.

When the SDRAM is in Power-Down, the memory controller services register accesses as normal. If the LP_MODE field is set not equal to 4h, or an SDRAM access is requested, or the Refresh Must level is reached while the SDRAM is in Power-Down, the memory controller brings the SDRAM out of Power-Down. For DDR3, the memory controller also exits Power-Down to perform incremental leveling.

Exit sequence of Power-Down mode for DDR2 and DDR3: The memory controller:

- Drives DDR[x]_CKE high after T_CKE + 1 cycles have elapsed since the POWER-DOWN command was issued. The value of T_CKE is taken from SDRAM timing 2 register (SDRTIM2).
- Waits for T_XP + 1 cycles. The value of T_XP is taken from SDRTIM2.
- Enters its idle state and can issue any commands.

7.2.12.3 Save and Restore Mode

The DDR2/3 memory controller supports a save and restore mechanism to completely switch off power to the DDR2/3 memory controller. The following operation puts the DDR2/3 memory controller in off mode:

- An external master reads the following memory-mapped registers and saves their value external to the DDR2/3 memory controller:
  - SDRAM configuration register (SDRCR)
  - SDRAM configuration register 2 (SDRCR2)
  - SDRAM refresh control shadow register (SDRRCSR)
  - SDRAM timing 1 register (SDRTIM1)
  - SDRAM timing 1 shadow register (SDRTIM1SR)
  - SDRAM timing 2 register (SDRTIM2)
  - SDRAM timing 2 shadow register (SDRTIM2SR)
  - SDRAM timing 3 register (SDRTIM3)
  - SDRAM timing 3 shadow register (SDRTIM3SR)
  - Power management control register (PMCR)
  - Power management control shadow register (PMCSR)
  - Peripheral bus burst priority register (PBBPR)
  - System OCP interrupt enable set register (SOIESR)
  - DDR PHY control register (DDRPHYCR)
  - DDR PHY control shadow register (DDRPHYCSR)
- Memory controller completes all pending transactions and drains all its FIFOs.
- Memory controller puts the SDRAM in self-refresh mode.
- Memory controller copies all shadow memory-mapped registers to its main registers. It is assumed that the shadow register always has the same value as its corresponding main register.
- Memory controller waits for all interrupts to be serviced.
- Memory controller acknowledges assertion of internal power down request.
- The internal module reset signal is asserted.
- The clocks and power to the memory controller can now be switched off.

To restore power to the memory controller, the following operation is followed:

- The power and clocks to the memory controller are switched on.
- The internal module reset signal is deasserted, indicating to the memory controller that it is waking up from off mode.
- The memory controller does not perform SDRAM initialization and forces its state machine to be in self-refresh mode.
- The external master restores all of the above memory-mapped registers.
- The memory controller exits self-refresh state.
- The system can now perform access to the external memory.

7.3 DDR PHY

7.3.1 Functional Overview

DDR PHY is a fully digital, flexible and advanced solution to get the ultimate performance from their memory interface. The DDR PHY supports:

- Bi-directional DQS
- Software read/write leveling.

For DDR3, while maintaining a backward compatibility with the DDR2 standard.

DDR PHY consists of 3 separate blocks:

- Address and command block
- Byte-wide data block
- Master DLL block

The data slices or macros contain the data signal path and the slave DLLs for write data, write DQS, and read DQS. The ratio logic functions are used to adjust the signal timing for each signal edge and these are controlled by the master DLL.

7.3.2 Data Slice Block

The data slice block contains all the logic required to support read and write interface for a single memory data slice of 8 bits. It handles the DQ, DM, DQS, and FIFO_WE signals. The block consists of the following modules:

- Data Macro
- Write Data/DQS Training FSM
- Read Data Eye Training FSM
- Read Gate Training FSM
- Multi-Rank Ratio Logic

7.3.3 Master DLL

The Master DLL measures the cycle period in terms of a number of taps and passes this number through the ratio logic to the slave delay lines. The DLL is comprised of a delay line, an FSM, a clock divider, a phase detector and output filter. The FSM sends different control values to the delay line until phase detector detects transition of crossed from 0 to 1. When cross is detected, control values are provided to output filter which generates dll lock as well as final full cycle delay value. The master DLL manages On-Chip Variation (OCV) and continuously monitors core clock over PVT.

7.3.4 Ratio Logic Block

The Ratio Logic takes the output from the master DLL, which indicates the number of delay taps required to form a full-cycle shift, and scales it by an input ratio value in the range of 0 to 511 in units of 1/256th of a clock cycle. The result from Ratio Logic is the number of delay taps for slave Delay line.

7.3.5 Address Command Write, Write and Read Operation

7.3.5.1 Address Command Write

DDR[x]_CLK[y] must be properly delayed with respect to DDR[x]_DQS[3:0] to start write leveling; otherwise, the write-leveling operation cannot be guaranteed. To ensure proper functionality, there is a feature for pushing out the DDR[x]_CLK[y] and the corresponding address/command by one half cycle, enabled by asserting CMD0/1/2_REG_PHY_INVERT_CLKOUT_0. There is no need configuring this register, if:

- there is no fly-by routing on the board
- there is no multiple fanout on DDR[x]_CLK[y] (multiple ranks)

7.3.5.2 Write Operation

On a write operation, the IDID accepts signals synchronous to the core or memory clock from controller. DDR[x]_CLK[y]/$\overline{\text{DDR[x]\_CLK[y]}}$ are generated inside the IDID and sent out edge-aligned with DDR[x]_CLK[y]/$\overline{\text{DDR[x]\_CLK[y]}}$ for DDR2 operation, and delayed from core clock by the write-leveling delay for DDR3 operation. DDR[x]_D[31:0] and DDR[x]_DQM[3:0] are phase shifted by approximately 90 degrees with respect to DDR[x]_DQS[3:0].

7.3.5.3 Read Operation

For a read operation, the DDR[x]_D[31:0] signal sent by memory is edge-aligned with DDR[x]_DQS[3:0], so DDR[x]_DQS[3:0] must be shifted by PHY data slice to center-align DDR[x]_D[31:0] with DDR[x]_DQS[3:0]. This is approximately a 90-degree phase shift, with the exact shift dependent on the result of read DQS eye training in DDR3 applications, or register programming for DDR2 applications. After DDR[x]_D[31:0] is sampled by DDR[x]_DQS[3:0] and converted from DDR to SDR domain, the sampled data is re-synchronized by the core clock through a FIFO. Read data is then sent to the controller along with a data-valid signal.

7.4 Electrical Characteristics Control

7.4.1 Output Impedance Calibration

The DDR2/3 controller supports automatic output impedance calibration for DDR3. The ZQ calibration is enabled per chip select by setting the CS0EN and CS1EN bits in the SDRAM output impedance calibration configuration register (ZQCR). The DDR2/3 memory controller supports the following three types of ZQ calibration commands:

- ZQINIT – ZQ calibration command during initialization
- ZQCS – ZQ calibration short command
- ZQCL – ZQ calibration long command

For DDR3, the memory controller automatically issues a ZQINIT command during initialization. The memory controller waits and blocks any other command for (ZQINITMULT + 1) × (ZQCLMULT + 1) × (T_ZQCS + 1) number of clock DDR3 clock cycles every time a ZQINIT command is issued.

The memory controller periodically issues a ZQCS command every time REFINTERVAL expires. In other words, the REFINTERVAL field defines the interval between ZQCS commands. The memory controller waits and blocks any other command for (T_ZQCS + 1) number of DDR3 clock cycles every time a ZQCS command is issued.

If SEFXITEN field is set to a 1, the memory controller issues a ZQCL command every time it exits Self-Refresh, Active Power-Down, and Precharge Power-Down. The memory controller waits and blocks any other command for (ZQCLMULT + 1) × (T_ZQCS + 1) number of clock DDR clock cycles every time a ZQCL command is issued.

If a separate calibration resistor is used per device, the ZQ calibration can be performed simultaneously over both the chip selects. To enable ZQ calibration to be performed simultaneously over both chip selects, the DUALCALEN bit must be set to 1. If the DUALCALEN bit is cleared to a 0, the EMIF performs ZQ calibration per chip select serially.

7.4.2 IO Configuration and Slew Rate Control

On this device, programmability is given to control the output driver strength/impedance and slew rate. Table 7-14 shows the programmable impedance settings, where Rext is the external resistor between device VTP pin and ground. Recommended value is 50 ohms with 1% tolerance.

The recommended IO buffer impedance settings are:

1. For all the command signals DDR[x]_A[14:0], DDR[x]_BA[2:0], $\overline{\text{DDR[x]\_CAS}}$, $\overline{\text{DDR[x]\_WE}}$, $\overline{\text{DDR[x]\_CS[1:0]}}$, and DDR[x]_CLK/$\overline{\text{DDR[x]\_CLK}}$: programmed to a constant value of 5h and are independent of board and DDR2/3 memory chip.
2. For all data signals DDR[x]_D[31:0] , DDR[x]_DQM[3:0] , DDR[x]_DQS[3:0]/$\overline{\text{DDR[x]\_DQS[3:0]}}$, and DQSGATE (internal signal) : programmed to a constant value of 4h and are independent of board and DDR2/3 memory chip.

Refer to Section 7.7.2.1 and Section 7.7.2.2 for address/command pad and clock pad configuration registers.

Refer to Section 7.7.2.7 and Section 7.7.2.8 for data pad and data strobe configuration registers.

Table 7-14. Programmable Impedance Settings

I2	I1	I0	Output Impedance (Ron)	Ron for Rext 50 Ohms	Drive Strength
0	0	0	2 × Rext	100 ohms	4 mA
0	0	1	Rext	50 ohms	8 mA
0	1	0	1.33 × Rext	66.7 ohms	6 mA
0	1	1	0.8 × Rext	40 ohms	10 mA
1	0	0	1.6 × Rext	80 ohms	5 mA
1	0	1	0.88 × Rext	44.4 ohms	9 mA
1	1	0	1.14 × Rext	57.1 ohms	7 mA
1	1	1	0.72 × Rext	36.4 ohms	11 mA

To achieve optimal noise/frequency trade off, the slew rate of the output signal can also be programmed using the slew rate control registers. The Fasted Slew Rate bit setting is recommended. Table 7-15 shows the programmable slew-rate settings.

Table 7-15. Programmable Slew-rate Settings

SR[1]	SR[0]	Description
0	0	Fastest Slew Rate
0	1	Fast Slew Rate
1	0	Slow Slew Rate
1	1	Slowest Slew Rate

7.4.3 ODT Control

The DDR2/3 controller supports ODT programming. ODT may be required for better system-level signal integrity and performance. ODT strength level is a function of output driver strength. 50-ohms termination programming is recommended for better system signal integrity. Table 7-16 shows the ODT settings.

Table 7-16. ODT Settings

ODT[1]	ODT[0]	Description
0	0	ODT off
0	1	ODT enable (50 ohms). Recommended for better signal integrity. For control lines: Rext For other than control lines: (0.88 x Rext)
1	0	ODT off
1	1	ODT enable (100 ohms) For control lines: 2 × Rext For other than control lines: (2 × 0.88 x Rext)

Refer to DDRPHYCR (Section 7.7.1.26) and DDRPHYCSR (Section 7.7.1.27) for register programming.

7.5 DDR2/3 SDRAM Memory Initialization

7.5.1 DDR2 SDRAM Memory Initialization

The DQSGATE signal on earlier devices (such as the TMS320DM648 processor) were looped back through the board. However, on this device the DQSGATE signal is looped back internally within the device. A delay line is used to adjust for the board round-trip delay. This is captured in the register. The delay value is represented in terms of DDR2 clock cycles and is 10 bits. A value of 100h implies a delay equal to one DDR2 clock cycle period. For example, if the round-trip delay on board is 1.25 ns for 400-MHz clock DDR2 operation (2.5 ns period), this would mean half a cycle delay and hence the value programmed would be 80h. This is board dependent.

The recommended IO buffer impedance settings are:

1. For all the command signals DDR[x]_A[14:0], DDR[x]_BA[2:0], $\overline{\text{DDR[x]\_CAS}}$, $\overline{\text{DDR[x]\_WE}}$, $\overline{\text{DDR[x]\_CS[1:0]}}$, and DDR[x]_CLK[y]/$\overline{\text{DDR[x]\_CLK[y]}}$: programmed to a constant value of 5h and are independent of board and DDR2 memory chip.
2. For all data signals DDR[x]_D[31:0], DDR[x]_DQM[3:0], DDR[x]_DQS[3:0]/$\overline{\text{DDR[x]\_DQS[3:0]}}$, and DQSGATE (internal signal): programmed to a constant value of 4h and are independent of board and DDR2 memory chip.

DDR2 SDRAM devices contain mode and extended mode registers that configure the mode of operation for the device. These registers control parameters such as burst type, burst length, and CAS latency. The DDR2/3 memory controller programs the mode and extended mode registers of the DDR2 memory by issuing MRS and EMRS commands during the initialization sequence. The initialization sequence performed by the DDR2/3 memory controller is compliant with the JESD79-2E specification for DDR2 operation.

The DDR2/3 memory controller performs the DDR2 initialization sequence under the following condition:

- DDR2 auto-initialization is not done after cold reset and is triggered by programming the INITREF_DIS bit to 0 in the SDRRCR.

At the end of the initialization sequence, the DDR2 memory controller performs an auto-refresh cycle, leaving the DDR2 memory controller in an idle state with all banks deactivated.

7.5.1.1 DDR2 SDRAM Device Mode Register Configuration Values

The DDR2 memory controller initializes the mode register, extended mode register 1, and extended mode register 2 of the memory device with the values shown in Table 7-17, Table 7-18, and Table 7-19. The DDR2 SDRAM extended mode registers 3 is configured with a value of 0.

Table 7-17. DDR2 SDRAM Mode Register Configuration

Mode Register Bit	Mode Register Field	Initial Value	Description
12	Power-down Mode	0	Active power-down exit time bit. Configured for Fast exit.
11-9	Write Recovery	SDRTIM1.T_WR	Write recovery bits for auto-precharge. Initialized using the T_WR bit if the SDRAM timing 1 register (SDRTIM1).
8	DLL Reset	1h	DLL reset bit
7	Mode	0	Operating mode bit. Normal operating mode is always selected.
6-4	CAS Latency	SDRCR.CL	CAS latency bit. Initialized using the CL bit in the SDRAM configuration register (SDRCR).
3	Burst Type	0	Burst type bit. Sequential burst mode is always used.
2-0	Burst Length	3h	Burst length bit. A burst length of 8 is always used.

Table 7-18. DDR2 SDRAM Extended Mode Register 1 Configuration

Mode Register Bit	Mode Register Field	Initial Value	Description
12	Output Buffer Enable	0	Output buffer enable bits. Output buffer is always enabled.
11	RDQS Enable	0	Always initialized to 0.
10	!DQS Enable	SDRCR.DDQS	DQS enable
9-7	OCD Operation	0	Off-chip driver impedance calibration bits. This bit is always initialized to 0.
6	ODT Value(Rtt) or DDRTERM[1]	SDRCR.DDRTERM[1]	On-die termination effective resistance (Rtt) bit.
5-3	Additive Latency	0	Additive latency bits. Always initialized to 0 (no additive latency).
2	ODT Value(Rtt) or DDRTERM[0]	SDRCR.DDRTERM[0]	On-die termination effective resistance (Rtt) bit.
1	Output Driver Impedance	SDRCR.DRIVE	Output driver impedance control bits. Initialized using the DRIVE bit in the SDRAM configuration register (SDRCR).
0	Enable DLL	0	DLL enable/disable bit. DLL is always enabled.

Table 7-19. DDR2 SDRAM Extended Mode Register 2 Configuration

Mode Register Bit	Mode Register Field	Initial Value	Description
7	Self-Refresh Temperature	0	Normal Operating Temperature range
6-4	Reserved	0	Reserved
3	DCC	0	DCC disable
2-0	Partial array self-refresh	0	Always initialized to 0

7.5.1.2 After Register Configuration

The initialization sequence can be initiated by programming INITREF_DIS bit in the SDRAM refresh control register (SDRRCR) to 0. Perform the following steps to start the initialization sequence:

1. Program the DDRPLL for the required DDR2 memory target frequency.
2. Write 1 to DDR_RCD to bring Reset Clock Distribution module out of reset.
3. Program PRCM to enable DDR2/3 memory controller clocks. See Section 7.2.8.
4. Program DDR PHY:
   (a) Write 1 to DATA0/1/2/3_REG_PHY_USE_RANK0_DELAYS register to use rank 0 delays for all ranks during read/write operation.
   (b) Write 0 to CMD0/1/2_REG_PHY_INVERT_CLKOUT_0 register to send non-inverted DDR Clock to DRAM.
   (c) Program CMD0/1/2_REG_PHY_CTRL_SLAVE_RATIO_0 register. If CMD0/1/2_REG_PHY_INVERT_CLKOUT_0 is cleared to 0, program to 80h; else, program to 100h.
   (d) Write 0 to WR_LEVEL bit of DDR_VTP_CTRL_0 register to select slave ratio support.
   (e) Calculate the Read DQS Gate, Read DQS, and Write DQS parameters (see the wiki page: DDR3 Initialization with SW Leveling):
      (i) Program DATA0/1/2/3_REG_PHY_FIFO_WE_SLAVE_RATIO_0 register with Read DQS Gate parameter.
      (ii) Program DATA0/1/2/3_REG_PHY_WR_DQS_SLAVE_RATIO_0 register with Write DQS parameter.
      (iii) Program DATA0/1/2/3_REG_PHY_WR_DATA_SLAVE_RATIO_0 register with (Write DQS parameter + 40h)
      (iv) Program DATA0/1/2/3_REG_PHY_RD_DQS_SLAVE_RATIO_0 register with Read DQS parameter.
   (f) Write 5h to CMD0/1/2_IO_CONFIG_I_0 register.
   (g) Write 5h to CMD0/1/2_IO_CONFIG_I_CLK_0 register.
   (h) Write 4h to DATA0/1/2/3_IO_CONFIG_I_0 register.
   (i) Write 4h to DATA0/1/2/3_IO_CONFIG_I_CLK_0 register.
5. Setup DMM registers DMM_LISA_MAP_0/1/2/3 to access DDR.
6. Program SDRTIM1 and SDRTIM1SR registers with the value needed to meet the DDR2 SDRAM device timings.
7. Program SDRTIM2 and SDRTIM2SR registers with the value needed to meet the DDR2 SDRAM device timings.
8. Program SDRTIM3 and SDRTIM3SR registers with the value needed to meet the DDR2 SDRAM device timings.
9. Program SDCFG register with the value needed to meet the DDR3 SDRAM.
10. Program the RL and RD_LODT bits in DDRPHYCR register to the desired value.
11. Program the RR bit in SDRRCR register to a value that meets the refresh requirements of the DDR3 SDRAM device.
12. Write 1 to the ASR bit in SDRRCR register to enable DDR3 auto self refresh.
13. Write 0 to the INITREF_DIS bit in SDRRCR register to enable DDR3 initialization.

7.5.2 DDR3 SDRAM Memory Initialization

7.5.2.1 Introduction

DDR3 supports the following modes to compute the slave delay lines:

- User Defined Delays as a Ratio of the DDR3 Clock. This mode ensures that the delays track the clock automatically. To calculate the Read DQS Gate, Read DQS, and Write DQS parameters, see the wiki page: DDR3 Initialization with SW Leveling.
- Automatic Mode: Use the results from Leveling state machines.

7.5.2.2 DDR3 SDRAM Device Mode Register Configuration Values

The DDR3 memory controller initializes the mode register, extended mode register 1, and extended mode register 2 of the memory device with the values shown in Table 7-20, Table 7-21, and Table 7-22.

Table 7-20. DDR3 SDRAM Mode Register Configuration

Mode Register Bit	Mode Register Field	Initial Value	Description
12	Power-down Mode	0	Active power-down exit time bit. Configured for Fast exit.
11-9	Write Recovery	SDRTIM1.T_WR	Write recovery bits for auto-precharge. Initialized using the T_WR bit in the SDRAM timing 1 register (SDRTIM1).
8	DLL Reset	1h	DLL reset bit
7	Mode	0	Operating mode bit. Normal operating mode is always selected.
6-4	CAS Latency	SDRCR.CL[3:1]	CAS latency bit. Initialized using the CL bit in the SDRAM configuration register (SDRCR).
3	Burst Type	0	Burst type bit. Sequential burst mode is always used.
2	CAS Latency	SDRCR.CL[0]	CAS latency bit. Initialized using the CL bit in the SDRAM configuration register (SDRCR).
1-0	Burst Length	0	Burst length bits. A burst length of 8 is always used.

Table 7-21. DDR3 SDRAM Extended Mode Register 1 Configuration

Mode Register Bit	Mode Register Field	Initial Value	Description
12	Output Buffer Enable	0	Output buffer enable bit. Output buffer is always enabled.
11	TDQS Enable	0	TDQS disable
10	Reserved	0	Reserved
9	ODT Value(Rtt) or DDRTERM[2	SDRCR.DDRTERM[2]	On-die termination effective resistance (Rtt) bit.
8	Reserved	0	Reserved
7	Write Leveling	1h	Write Leveling Enabled
6	ODT Value(Rtt) or DDRTERM[1]	SDRCR.DDRTERM[1]	On-die termination effective resistance (Rtt) bit.
5	Output Driver Impedance	SDRCR.DRIVE[1]	Output driver impedance control bits. Initialized using the DRIVE bit in the SDRAM configuration register (SDRCR).
4-3	Additive Latency	0	Additive latency bits. Always initialized to 0 (no additive latency).
2	ODT Value(Rtt) or DDRTERM[0]	SDRCR.DDRTERM[0]	On-die termination effective resistance (Rtt) bit.
1	Output Driver Impedance	SDRCR.DRIVE	Output driver impedance control bit. Initialized using the DRIVE bit in the SDRAM configuration register (SDRCR).
0	Enable DLL	0	DLL enable/disable bits. DLL is always enabled.

Table 7-22. DDR3 SDRAM Extended Mode Register 2 Configuration

Mode Register Bit	Mode Register Field	Initial Value	Description
10-9	Dynamic ODT	SDRCR.DYNODT	Dynamic ODT value from SDRAM configuration register (SDRCR).
8	Reserved	0	Reserved
7	Self-Refresh Range	0	Normal Self Refresh
6	Auto Self-Refresh Enable	SDRCR.ASR	Auto Self Refresh enable from SDRAM refresh control register (SDRRCR).
5	Reserved	0	Reserved
4-3	CAS Write Latency	SDRCR.CWL	CAS Write Latency from SDRAM configuration register (SDRCR).
2-0	Partial array self-refresh	0	Always initialized to 0

7.5.2.3 After Register Configuration

1. Program the DDRPLL for the required DDR3 memory target frequency.
2. Write 1 to DDR_RCD to bring Reset Clock Distribution module out of reset.
3. Program PRCM to enable DDR2/3 memory controller clocks. See Section 7.2.8.
4. Program DDR PHY:
   (a) Write 1 to DATA0/1/2/3_REG_PHY_USE_RANK0_DELAYS register to use rank 0 delays for all ranks during read/write operation.
   (b) Write 1 to CMD0/1/2_REG_PHY_INVERT_CLKOUT_0 register to send inverted DDR Clock to DRAM.
   (c) Program CMD0/1/2_REG_PHY_CTRL_SLAVE_RATIO_0 register. If CMD0/1/2_REG_PHY_INVERT_CLKOUT_0 is cleared to 0, program 80h; else, program to 100h.
   (d) Write 5h to CMD0/1/2_IO_CONFIG_I_0 register.
   (e) Write 5h to CMD0/1/2_IO_CONFIG_I_CLK_0 register.
   (f) Write 4h to DATA0/1/2/3_IO_CONFIG_I_0 register.
   (g) Write 4h to DATA0/1/2/3_IO_CONFIG_I_CLK_0 register.
5. Setup DMM registers DMM_LISA_MAP_0/1/2/3 to access DDR.
6. Program SDRTIM1 and SDRTIM1SR registers with the value needed to meet the DDR3 SDRAM device timings.
7. Program SDRTIM2 and SDRTIM2SR registers with the value needed to meet the DDR3 SDRAM device timings.
8. Program SDRTIM3 and SDRTIM3SR registers with the value needed to meet the DDR3 SDRAM device timings.
9. Program SDCFG register with the value needed to meet the DDR3 SDRAM.
10. Program the RL and RD_LODT bits in DDRPHYCR register to the desired value.
11. Program the RR bit in the SDRAM refresh control register (SDRRCR) to a value that meets the refresh requirements of the DDR3 SDRAM device.
12. Write 1 to the ASR bit in SDRRCR register to enable DDR3 auto self refresh.
13. Write 0 to the INITREF_DIS bit in SDRRCR register to enable DDR3 initialization.
14. Write 0 to the WR_LEVEL bit in DDR_VTP_CTRL_0 register to select slave ratio support.

15. Calculate the Read DQS Gate, Read DQS, and Write DQS parameters (see the wiki page: *DDR3 Initialization with SW Leveling*):
    (a) Program DATA0/1/2/3_REG_PHY_FIFO_WE_SLAVE_RATIO_0 register with Read DQS Gate parameter.
    (b) Program DATA0/1/2/3_REG_PHY_WR_DQS_SLAVE_RATIO_0 register with Write DQS parameter.
    (c) Program DATA0/1/2/3_REG_PHY_WR_DATA_SLAVE_RATIO_0 register with (Write DQS parameter + 40h)
    (d) Program DATA0/1/2/3_REG_PHY_RD_DQS_SLAVE_RATIO_0 register with Read DQS parameter.

7.6 Using the DDR2/3 Memory Controller

The following sections describe the architecture of the DDR2/3 memory controller and how to configure it to perform read and write operations to DDR2/3 SDRAM devices.

Not all memory topologies may be supported by your device. See your device-specific data manual for details of supported configurations and topologies.

Printed circuit board (PCB) layout rules and connection requirements between the processor and the memory device are described in the device-specific data manual.

7.6.1 Configuring DDR2/3 Memory Controller Registers to Meet DDR2 SDRAM Specifications

The DDR2/3 memory controller allows a high degree of programmability for shaping DDR2 accesses. This provides the DDR2/3 memory controller with the flexibility to interface with a variety of DDR2 devices. By programming the SDRAM configuration register (SDRCR), SDRAM refresh control register (SDRRCR), SDRAM timing 1 register (SDRTIM1), SDRAM timing 2 register (SDRTIM2), and SDRAM timing 3 register (SDRTIM3), the DDR2/3 memory controller can be configured to meet the data sheet specification for JESD79-2E compliant DDR2 SDRAM devices.

As an example, the following sections describe how to configure each of these registers for access to two 1Gb, 16-bit wide DDR2 SDRAM devices, where each device has the following configuration:

- Maximum data rate: 800 MHz
- Number of banks: 8
- Page size: 1024 words
- CAS latency: 5

Consider a data rate of 800 MHz for DDR2 controller configurations in the following example.

7.6.1.1 Programming the SDRAM Configuration Register (SDRCR)

The SDRAM configuration register (SDRCR) contains register fields that configure the DDR2 memory controller to match the data bus width, CAS latency, number of banks, and page size of the attached DDR2 memory. Table 7-23 shows the SDRCR configuration.

Table 7-23. SDRCR Configuration

Field	Value	Function Selection
MEMTYPE	2h	To select DDR2 Memory Type
DDRTERM	3h	To select 50 ohm termination register
DQS	1h	To select Differential Ended DQS
DLL	0h	To enable DLL for Normal Operation
DRIVE	0h	To select full drive strength
NM	0h	To configure the DDR2 memory controller for a 32-bit data bus width.
CL	5h	To select CAS latency of 5
IBANK	8h	To select 8 internal DDR2 banks
EBANK	0h	To select CS0
PAGESIZE	2h	To select 1024-word page size

7.6.1.2 Programming the SDRAM Refresh Control Register (SDRRCR)

The SDRAM refresh control register (SDRRCR) configures the DDR2 memory controller to meet the refresh requirements of the attached DDR2 device. SDRRCR also allows the DDR2 memory controller to enter and exit self refresh. In this example, we assume that the DDR2 memory controller is not is in self-refresh mode.

The RR field in SDRRCR is defined as the rate at which the attached DDR2 device is refreshed in DDR2 cycles. The value of this field may be calculated using the following equation:

RR = DDR2 clock frequency (in MHz) × DDR2 memory fresh rate (in µs)

The DDR2-800 refresh rate specification:

Symbol	Description	Value
tREF	Average Periodic Refresh Interval	7.8 µs

Therefore, the value for the refresh rate (RR) can be calculated as:

RR = 400 MHz x 7.8 µs = 2078.7 = C30h

Table 7-24 shows the SDRRCR configuration.

Table 7-24. SDRRCR Configuration

Field	Value	Function Selection
INITREF_DIS	0	To Enable DDR2 Initialization and Refreshes
ASR	1	To Enable auto self refresh
RR	C30h	Set to C30h DDR2 clock cycles to meet the DDR2 memory refresh rate requirement

7.6.1.3 Configuring SDRAM Timing Registers (SDRTIM1, SDRTIM2, and SDTIM3)

The SDRAM timing 1 register (SDRTIM1), SDRAM timing 2 register (SDRTIM2), and SDRAM timing 3 register (SDRTIM3) configure the DDR2 memory controller to meet the data sheet timing parameters of the attached DDR2 device. Each field in SDRTIM1, SDRTIM2, and SDTIM3 corresponds to a timing parameter in the DDR2 data sheet specification. Table 7-25, Table 7-26, and Table 7-27 lists the register field name and corresponding DDR2 data sheet parameter name along with the data sheet value. These tables also provide a formula to calculate the register field value and displays the resulting calculation. Each of the equations includes a minus 1 because the register fields are defined in terms of DDR2 clock cycles minus 1.

Table 7-25. SDRTIM1 Configuration

Register Field	DDR2 SDRAM Data Sheet Parameter	Description	Data Sheet Value	Unit	Formula (Register Field Must be ≥)	Field Value
T_RP	tRP	Precharge command to refresh or activate command	13.125	nS	(tRP × fDDR[x]_CLK) - 1	5
T_RCD	tRCD	Activate command to read/write command	13.125	nS	(tRCD × fDDR[x]_CLK) - 1	5
T_WR	tWR	Write recovery time	15	nS	(tWR × fDDR[x]_CLK) - 1	5
T_RAS	tRAS	Active to precharge command	45	nS	(tRAS × fDDR[x]_CLK) - 1	17
T_RC	tRC	Activate to Activate command in the same bank	58.125	nS	(tRC × fDDR[x]_CLK) - 1	23
T_RRD	tRRD/tFAW	Activate to Activate command in a different bank	45	nS	(tFAW × fDDR[x]_CLK/4) - 1 for 8 bank DDR2 memory	4
T_WTR	tWTR	Write to read command delay	7.5	nS	(tWTR × fDDR[x]_CLK) - 1	3

Table 7-26. SDRTIM2 Configuration

Register Field	DDR2 SDRAM Data Sheet Parameter	Description	Data Sheet Value	Unit	Formula (Register Field Must be ≥)	Field Value
T_XP	tXP	Exit precharge power-down to any non-read command	3	nS	(tXP × fDDR[y]_CLK) - 1	2
T_XSNR	tXSNR	Exit self-refresh to a non-read command	137.5	nS	(tXSNR × fDDR[y]_CLK) - 1	54
T_XSRD	tXSRD	Exit self-refresh to a read command	200	tCK	(tXSRD - 1)	199
T_RTP	tRTP	Read to precharge command delay	7.5	nS	(tRTP × fDDR[y]_CLK) - 1	2
T_CKE	tCKE	CKE minimum pulse width (high and low pulse width)	3	tCK	(tCKE - 1)	2

Table 7-27. SDRTIM3 Configuration

Register Field	DDR2 SDRAM Data Sheet Parameter	Description	Data Sheet Value	Unit	Formula (Register Field Must be ≥)	Field Value
T_RFC	tRFC	Auto-refresh to active/auto-refresh command time	127.5	nS	(tRFC × fDDR[y]_CLK) - 1	43
T_RASMAX		Maximum number of refresh rate intervals from Active to Precharge command				15

7.6.1.4 Configuring DDR PHY Control Register

The DDR2/3 memory controller control register (DDRPHYCR) contains a read latency (RL) field that helps the DDR2 memory controller determine when to sample read data.

Table 7-28. DDRPHYCR Configuration

Field	Value	Description
RD_LODT	3	Termination resistance during read access. For DDR2 control lines (Rext) For other than DDR2 control lines (0.88 x Rext). Where Rext is the VTP resistor value.
RL	11	Maximum RL = CL + 7 - 1 Minimum RL = CL + 1 - 1 CL = 5 is being used for 400 MHz DDR2 clock

7.6.2 Configuring DDR2/3 Memory Controller Registers to Meet DDR3 SDRAM Specifications

The DDR2/3 memory controller allows a high degree of programmability for shaping DDR3 accesses. This provides the DDR2/3 memory controller with the flexibility to interface with a variety of DDR3 devices. By programming the SDRAM Configuration Register (SDRCR), SDRAM Refresh Control Register (SDRRCR), SDRAM Timing 1 Register (SDRTIM1), SDRAM Timing 2 Register (SDRTIM2), and SDRAM Timing 3 Register (SDRTIM3), the DDR2/3 memory controller can be configured to meet the data sheet specification for JESD79-3C compliant DDR3 SDRAM devices.

As an example, the following sections describe how to configure each of these registers for access to two 1Gb, 16-bit wide DDR3 SDRAM devices, where each device has the following configuration:

- Maximum data rate: 1593 MHz
- Number of banks: 8
- Page size: 1024 words
- CAS latency: 11

Consider a data rate of 1593 MHz for DDR3 controller configurations in the following example.

7.6.2.1 Programming the SDRAM Configuration Register (SDRCR)

The SDRAM configuration register (SDRCR) contains register fields that configure the DDR3 memory controller to match the data bus width, CAS latency, number of banks, and page size of the attached DDR3 memory. Table 7-29 shows the resulting SDRCR configuration.

Table 7-29. SDRCR Configuration

Field	Value	Description
MEMTYPE	3h	To select DDR3 Operation
IBANKPOS	0h	If 0, RSIZE field value is ignored
DDRTERM	2h	240 (ZQ Value)/2, 120 Ohm termination resistance
DDQS	1h	Differential DQS. DDR3 supports only differential DQS
DYNODT	1h	To select 60 Ohm Dynamic ODT
DLL	0h	To enable DLL for Normal Operation
DRIVE	1h	To select 34 Ohm Output Driver Impedance
CWL	3h	To select CAS write latency of 8
NW	0h	To configure DDR2/3 memory controller for a 32-bit data bus width.
CL	Eh	To select CAS latency of 11
RSIZE	0h	If IBANKPOS = 0, RSIZE field does not have any effect
IBANK	3h	To select 8 internal DDR3 banks
EBANK	0h	To select 1 chip
PAGESIZE	2h	To select 1024-word page size

7.6.2.2 Programming the SDRAM Refresh Control Register (SDRRCR)

The SDRAM refresh control register (SDRRCR) configures the DDR3 memory controller to meet the refresh requirements of the attached DDR3 device. SDRRCR also allows the DDR3 memory controller to enter and exit self refresh. In this example, we assume that the DDR3 memory controller is not is in self-refresh mode.

The RR field in SDRRCR is defined as the rate at which the attached DDR3 device is refreshed in DDR3 cycles. The value of this field may be calculated using the following equation:

RR = DDR3 clock frequency (in MHz) x DDR3 memory fresh rate (in µs)

The DDR3-1600 refresh rate specification:

Symbol	Description	Value
tREF	Average Periodic Refresh Interval	7.8 µs

Therefore, the value for the Refresh Rate (RR) can be calculated as follows:

RR = 796.5 MHz x 7.8 µs = 6212.7= 1844h

Table 7-30 shows the resulting SDRRCR configuration.

Table 7-30. SDRRCR Configuration

Field	Value	Function Selection
INITREF_DIS	0	To Enable DDR3 Initialization and Refreshes
ASR	1	To Enable auto self refresh
RR	1844h	Set to 1844h DDR3 clock cycles to meet the DDR3 memory refresh rate requirement

7.6.2.3 Configuring SDRAM Timing Registers (SDRTIM1, SDRTIM2, and SDTIM3)

The SDRAM timing 1 register (SDRTIM1), SDRAM timing 2 register (SDRTIM2), and SDRAM timing 3 register (SDRTIM3) configure the DDR3 memory controller to meet the data sheet timing parameters of the attached DDR3 device. Each field in SDRTIM1, SDRTIM2, and SDTIM3 corresponds to a timing parameter in the DDR3 data sheet specification. Table 7-31, Table 7-32, and Table 7-33 lists the register field name and corresponding DDR3 data sheet parameter name along with the data sheet value. These tables also provide a formula to calculate the register field value and displays the resulting calculation. Each of the equations includes a minus 1 because the register fields are defined in terms of DDR3 clock cycles minus 1.

Table 7-31. SDRTIM1 Configuration

Register Field	DDR2 SDRAM Data Sheet Parameter	Description	Data Sheet Value	Unit	Formula (Register Field Must be ≥)	Field Value
T_RP	tRP	Precharge command to refresh or activate command	13.125	nS	(tRP × fDDR[x]_CLK) - 1	10
T_RCD	tRCD	Activate command to read/write command	13.125	nS	(tRCD x fDDR[x]_CLK) - 1	10
T_WR	tWR	Write recovery time	15	nS	(tWR x fDDR[x]_CLK) - 1	11
T_RAS	tRAS	Active to precharge command	35	nS	(tRAS x fDDR[x]_CLK) - 1	27
T_RC	tRC	Activate to Activate command in the same bank	48.125	nS	(tRC x fDDR[x]_CLK) - 1	38
T_RRD	tRRD/tFAW	Activate to Activate command in a different bank	30 (x8) 40 (x16)	nS	(tFAW x fDDR[x]_CLK/4) - 1 for 8 Bank DDR3	5 (×8) 7 (×16)

Table 7-31. SDRTIM1 Configuration (continued)

Register Field	DDR2 SDRAM Data Sheet Parameter	Description	Data Sheet Value	Unit	Formula (Register Field Must be ≥)	Field Value
T_WTR	tWTR	Write to read command delay	7.5	nS	(Max((T_WTR x fDDR[x]_CLK),4)) - 1	5

Table 7-32. SDRTIM2 Configuration

Register Field	DDR2 SDRAM Data Sheet Parameter	Description	Data Sheet Value	Unit	Formula (Register Field Must be ≥)	Field Value
T_XP	tXP	Exit precharge power-down to any non-read command	6	nS	MAX((T_XP x fDDR[x]_CLK),3) - 1	4
T_XSNR	tXSNR	Exit self-refresh to a non-read command	120	nS	(tXSNR x fDDR[x]_CLK) - 1	95
T_XSRD	tXSRD	Exit self-refresh to a read command	512	tCK	(tXSRD - 1)	511
T_RTP	tRTP	Read to precharge command delay	7.5	nS	(tRTP x fDDR[x]_CLK) - 1	5
T_CKE	tCKE	CKE minimum pulse width (high and low pulse width)	5	tCK	MAX((T_CKE × fDDR[x]_CLK), 3) - 1	3

Table 7-33. SDRTIM3 Configuration

Register Field	DDR2 SDRAM Data Sheet Parameter	Description	Data Sheet Value	Unit	Formula (Register Field Must be ≥)	Field Value
T_CKESR	tCKESR	Minimum CKE low pulse width for self refresh entry to self refresh exit timing	6.25	nS	MAX((T_CKESR × fDDR[x]_CLK), 3) - 1	5
T_ZQCS	tZQCS	Normal operation short calibration time	64	tCK	(T_ZQCS - 1)	63
T_RFC	tRFC	Auto-refresh to active/auto-refresh command time	110	nS	(T_RFC x fDDR[x]_CLK) - 1	87
T_RAS_MAX		Maximum number of refresh rate intervals from Active to Precharge command. This field should be programmed to 15.				15

7.6.2.4 Configuring DDR PHY Control Register

The DDR2/3 memory controller control register (DDRPHYCR) contains a read latency (RL) field that helps the DDR3 memory controller determine when to sample read data.

Table 7-34. DDRPHYCR Configuration

Field	Value	Description
RD_LODT	1	Termination resistance during read access. For DDR3 control lines (Rext) For other than DDR3 control lines (0.88 x Rext). Where Rext is the VTP resistor value.
RL	17	Maximum RL = CL + 7 - 1 Minimum RL = CL + 1 - 1 CL = 11 is being used for 800 MHz DDR3 clock

7.7 DDR2/3 Memory Controller Registers

7.7.1 DDR2/3 Memory Controller Registers

Table 7-35 lists the memory-mapped registers for the DDR2/3 memory controller. For the base address of these registers, see Table 1-11.

Table 7-35. DDR2/3 Memory Controller Registers

Address Offset	Acronym	Register Description	Section
4h	SDRSTAT	SDRAM Status Register	Section 7.7.1.1
8h	SDRCR	SDRAM Configuration Register	Section 7.7.1.2
Ch	SDRCR2	SDRAM Configuration Register 2	Section 7.7.1.3
10h	SDRRCR	SDRAM Refresh Control Register	Section 7.7.1.4
14h	SDRRCSR	SDRAM Refresh Control Shadow Register	Section 7.7.1.5
18h	SDRTIM1	SDRAM Timing 1 Register	Section 7.7.1.6
1Ch	SDRTIM1SR	SDRAM Timing 1 Shadow Register	Section 7.7.1.7
20h	SDRTIM2	SDRAM Timing 2 Register	Section 7.7.1.8
24h	SDRTIM2SR	SDRAM Timing 2 Shadow Register	Section 7.7.1.9
28h	SDRTIM3	SDRAM Timing 3 Register	Section 7.7.1.10
2Ch	SDRTIM3SR	SDRAM Timing 3 Shadow Register	Section 7.7.1.11
38h	PMCR	Power Management Control Register	Section 7.7.1.12
3Ch	PMCSR	Power Management Control Shadow Register	Section 7.7.1.13
54h	PBBPR	Peripheral Bus Burst Priority Register	Section 7.7.1.14
80h	PRFCNT1	Performance Counter 1 Register	Section 7.7.1.15
84h	PRFCNT2	Performance Counter 2 Register	Section 7.7.1.16
88h	PRFCNTCFG	Performance Counter Config Register	Section 7.7.1.17
8Ch	PRFCNTSEL	Performance Counter Master Region Select Register	Section 7.7.1.18
90h	PRFCNTTIM	Performance Counter Time Register	Section 7.7.1.19
A0h	EOI	End of Interrupt Register	Section 7.7.1.20
A4h	SOIRSR	System OCP Interrupt Raw Status Register	Section 7.7.1.21
ACh	SOISR	System OCP Interrupt Status Register	Section 7.7.1.22
B4h	SOIESR	System OCP Interrupt Enable Set Register	Section 7.7.1.23
BCh	SOIECR	System OCP Interrupt Enable Clear Register	Section 7.7.1.24
C8h	ZQCR	SDRAM Output Impedance Calibration Configuration Register	Section 7.7.1.25
E4h	DDRPHYCR	DDR PHY Control Register	Section 7.7.1.26
E8h	DDRPHYCSR	DDR PHY Control Shadow Register	Section 7.7.1.27

7.7.1.1 SDRAM Status Register (SDRSTAT)

The SDRSTAT is shown in Figure 7-4 and described in Table 7-36.

Figure 7-4. SDRAM Status Register (SDRSTAT)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		PHYRDY	Reserved	
R-0		R-0	R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 7-36. SDRAM Status Register (SDRSTAT) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	PHYRDY	0	DDR PHY is not ready.
		1	DDR PHY is ready
1-0	Reserved	0	Reserved

7.7.1.2 SDRAM Configuration Register (SDRCR)

The SDRCR is shown in Figure 7-5 and described in Table 7-37.

Figure 7-5. SDRAM Configuration Register (SDRCR)

31	29	28	27	26	24
MEMTYPE		IBANKPOS		DDRTERM	
R/W-3h		R/W-0		R/W-2h	

23	22	21	20	19	18	17	16
DDQS	DYNODT		DLL	DRIVE		CWL	
R/W-1	R/W-0		R/W-0	R/W-1h		R/W-3h	

15	14	13	10	9	8
NM		CL		RSIZE	
R/W-0		R/W-Eh		R/W-5h	

7	6	4	3	2	0
RSIZE	IBANK		EBANK	PAGESIZE	
R/W-5h	R/W-3h		R/W-0	R/W-2h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-37. SDRAM Configuration Register (SDRCR) Field Descriptions

Bit	Field	Value	Description
31-29	MEMTYPE		SDRAM Type Selection
		0-1h	Reserved
		2h	DDR2
		3h	DDR3
		4h-7h	Reserved
28-27	IBANKPOS		Internal Bank position selection.
		0	DDR2/3 memory controller uses 3 fields, IBANK, EBANK and PAGESIZE in the SDRCR to determine the mapping from source address to DDR2/3 memory device row, column, bank and chip select
		1h-3h	DDR2/3 controller uses 4 fields, IBANK, EBANK, PAGESIZE and RSIZE in the SDRCR to determine the mapping from source address to DDR2/3 memory device row, column, bank, and chip select
26-24	DDRTERM		DDR2 and DDR3 termination resistor value.
		0	Disable
			For DDR2 only:
		1h	75 Ohm
		2h	150 Ohm
		3h	50 Ohm (Recommended)
		4h-7h	Reserved
			For DDR3 only:
		1h	RZQ/4
		2h	RZQ/2 (Recommended)
		3h	RZQ/6
		4h	RZQ/12
		5h	RZQ/8
		6h-7h	Reserved
23	DDQS		DDR2 Differential DQS Enable. DDR3 supports only Differential DQS.
		0	Single Ended DQS
		1	Differential Ended DQS

Table 7-37. SDRAM Configuration Register (SDRCR) Field Descriptions (continued)

Bit	Field	Value	Description
22-21	DYNODT		DDR3 Dynamic ODT
		0	Off
		1h	RZQ/4 (Recommended)
		2h	RZQ/2
		3h	Reserved
20	DLL		Disable DLL Select
		0	Enable DLL for Normal Operation
		1	Disable DLL
19-18	DRIVE		DDR2/3 Drive strength. Configures output drive impedance control value of the DDR2/3 SDRAM memory.
		2h-3h	Reserved
			For DDR2:
		0	Full Drive Strength
		1h	Reduced Drive Strength
			For DDR3:
		0	RZQ/6
		1h	RZQ/7
17-16	CWL		DDR3 CAS Write Latency. ForDDR2, this bit has no effect.
		0	CAS Write Latency of 5
		1h	CAS Write Latency of 6
		2h	CAS Write Latency of 7
		3h	CAS Write Latency of 8
15-14	NM		Data Bus Width
		0	32 Bits
		1	16 Bits
13-10	CL		CAS Latency
			For DDR2 CAS Latency:
		0-1h	Reserved
		2h	CAS latency of 2
		3h	CAS latency of 3
		4h	CAS latency of 4
		5h	CAS latency of 5
		6h	CAS latency of 6
		7h	CAS latency of 7
		8h-Fh	Reserved
			For DDR3 CAS Latency:
		0-1h	Reserved
		2h	CAS latency of 5
		4h	CAS latency of 6
		6h	CAS latency of 7
		8h	CAS latency of 8
		Ah	CAS latency of 9
		Ch	CAS latency of 10
		Eh	CAS latency of 11
		3h, 5h, 7h, 9h, Bh, Dh, Fh	Reserved

Table 7-37. SDRAM Configuration Register (SDRCR) Field Descriptions (continued)

Bit	Field	Value	Description
9-7	RSIZE		Row Size Selection
		0	9 Rows bits
		1h	10 Rows bits
		2h	11 Rows bits
		3h	12 Rows bits
		4h	13 Rows bits
		5h	14 Rows bits
		6h	15 Rows bits
		7h	Reserved
6-4	IBANK		Bank Selection. Defines number of internal banks on the external DDR2/3 memory.
		0	1 Bank
		1h	2 Bank
		2h	4 Bank
		3h	8 Bank
		4h-7h	Reserved
3	EBANK		Chip Selection. Defines whether SDRAM accesses use 1 or 2 chip selects.
		0	CS0 only
		1	CS0 and CS1
2-0	PAGESIZE		Page Size Selection. Defines the page size of each page of the external DDR2/3 memory.
		0	256 word page requiring 8 column address bits
		1h	512 word page requiring 9 column address bits
		2h	1024 word page requiring 10 column address bits
		3h	2048 word page requiring 11 column address bits

7.7.1.3 SDRAM Configuration Register 2 (SDRCR2)

The SDRCR2 is shown in Figure 7-6 and described in Table 7-38.

Figure 7-6. SDRAM Configuration Register 2 (SDRCR2)

31	28	27	26	16
Reserved		EBANK_POS	Reserved	
R-0		R/W-0	R-0	

15	0
Reserved	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-38. SDRAM Configuration Register 2 (SDRCR2) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27	EBANK_POS		External Bank Position
		0	Assign external bank address bits from lower OCP address bits as shown in tables
		1	Assign external bank address bits from higher OCP address bits as shown in tables
26-0	Reserved	0	Reserved

7.7.1.4 SDRAM Refresh Control Register (SDRRCR)

The SDRRCR is shown in Table 7-39 and described in Figure 7-7.

Figure 7-7. SDRAM Refresh Control Register (SDRRCR)

31	30	29	28	27	26 24	23 16
INITREF_DIS	Rsvd	SRT	ASR	Rsvd	PASR	Reserved
R/W-1	R-0	R/W-0	R/W-0	R-0	R/W-0	R-0

15 0
RR
R/W-613Bh

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-39. SDRAM Refresh Control Register (SDRRCR) Field Descriptions

Bit	Field	Value	Description
31	INITREF_DIS		Initialization and Refresh Disable
		0	Enable SDRAM Initialization and Refreshes.
		1	Disable SDRAM Initialization and Refreshes but carries out SDRAM Write/Read transactions.
30	Reserved	0	Reserved
29	SRT		DDR2 and DDR3 Self Refresh Temperature Range. For DDR3, this bit must be cleared to 0 if ASR is set to 1. A write to this field causes the DDR2/3 to start the initialization sequence.
		0	Normal Operating Temperature range.
		1	Extended Operating Temperature range.
28	ASR		DDR3 Auto Self Refresh Enable. A write to this field causes the DDR3 to start the SDRAM initialization sequence.
		0	Reserved
		1	Auto Self Refresh Enable
27	Reserved	0	Reserved
26-24	PASR		Partial Array Self Refresh. Determines what portion of the array is refreshed with the external memory in partial array self refresh mode. These bits gets loaded into the EMR of the external DDR2/3 memory during initialization. If the external memory does not support partial array self refresh, this field should be cleared to 0. A write to this field causes the memory controller to start the SDRAM initialization sequence.
		0	Full array
		1h	1/2 array
		2h	1/4 array
		3h	1/8 array
		4h	3/4 array
		5h	1/2 array
		6h	1/4 array
		7h	1/8 array
23-16	Reserved	0	Reserved
15-0	RR	0-FFFFh	Refresh rate. Defines the rate at which the attached DDR2/3 devices are refreshed. The value of this field is calculated with the following equation:
			RR = DDR2/3 clock frequency (in MHz) x DDR2/3 fresh rate (in µs)
			Where DDR2/3 refresh rate is derived from DDR2/3 data sheet.

7.7.1.5 SDRAM Refresh Control Shadow Register (SDRRCSR)

The SDRRCSR is shown in Figure 7-8 and described in Table 7-40.

Figure 7-8. SDRAM Refresh Control Shadow Register (SDRRCSR)

31	16	15	0
Reserved		RR_SHDW	
R-0		R/W-613Bh	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 7-40. SDRAM Refresh Control Shadow Register (SDRRCSR) Field Descriptions

Bits	Field	Value	Description
31-16	Reserved	0	
15-0	RR_SHDW	0-FFFFh	Shadow field for RR in SDRRCR. This field is loaded into RR field in SDRAM Refresh Control Register when SIdleAck is asserted.

7.7.1.6 SDRAM Timing 1 Register (SDRTIM1)

The SDRTIM1 is shown in Figure 7-9 and described in Table 7-41.

Figure 7-9. SDRAM Timing 1 Register (SDRTIM1)

31 29	28 25	24 21	20 17
Reserved	T_RP	T_RCD	T_WR
R-0	R/W-Ah	R/W-Ah	R/W-Bh

16 12	11 6	5 3	2 0
T_RAS	T_RC	T_RRD	T_WTR
R/W-1Bh	R/W-26h	R/W-5h	R/W-5h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-41. SDRAM Timing 1 Register (SDRTIM1) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved
28-25	T_RP	0-Fh	Specifies the minimum number of DDR[x]_CLK[y] cycles from a precharge command to a refresh or activate command, minus 1. Corresponds to tRP AC timing parameter in the DDR2/3 data sheet. Calculated by T_RP = (tRP/DDR[x]_CLK[y] period) - 1
24-21	T_RCD	0-Fh	Specifies the minimum number of DDR[x]_CLK[y] cycles from an activate command to read or write command, minus 1. Corresponds to tRCD AC timing parameter in the DDR2/3 data sheet. Calculated by T_RCD = (tRCD/DDR[x]_CLK[y] period) - 1
20-17	T_WR	0-Fh	Specifies the minimum number of DDR[x]_CLK[y] cycles from the last write transfer to a precharge command, minus 1. Corresponds to tWR AC timing parameter in the DDR2/3 data sheet. Calculated by T_WR = (tWR/DDR[x]_CLK[y] period) - 1
16-12	T_RAS	0-1Fh	Specifies the minimum number of DDR[x]_CLK[y] cycles from an active command to a precharge command, minus 1. Corresponds to tras AC timing parameter in the DDR2/3 data sheet. Calculated by T_RAS = (tRAS/DDR[x]_CLK[y] period) - 1
11-6	T_RC	0-3Fh	Specifies the minimum number of DDR[x]_CLK[y] cycles from an active command to an active command, minus 1. Corresponds to trc AC timing parameter in the DDR2/3 data sheet. Calculated by T_RC = (tRC/DDR[x]_CLK[y] period) - 1
5-3	T_RRD	0-7h	Specifies the minimum number of DDR[x]_CLK[y] cycles from an activate command to an activate command in a different bank, minus 1. Corresponds to tRRD AC timing parameter in the DDR2/3 data sheet. Calculated by T_RRD = (tRRD/DDR[x]_CLK[y] period) - 1 For an 8-ban DDR2 and DDR3, this field must be equal to ((tFAW/(4 × DDR[x]_CLK[y] period)) - 1)
2-0	T_WTR	0-7h	Specifies the minimum number of DDR[x]_CLK[y] cycles from the last write to a read command, minus 1. Corresponds to the tWTR AC timing parameter in the DDR2/3 data sheet. Calculated by T_WTR = (tWTR/DDR[x]_CLK[y] period) - 1

7.7.1.7 SDRAM Timing 1 Shadow Register (SDRTIM1SR)

The SDRTIM1SR is shown in Figure 7-10 and described in Table 7-42.

Figure 7-10. SDRAM Timing 1 Shadow Register (SDRTIM1SR)

31 29	28 25	24 21	20 17
Reserved	T_RP_SHDW	T_RCD_SHDW	T_WR_SHDW
R-0	R/W-Ah	R/W-Ah	R/W-Bh

16 12	11 6	5 3	2 0
T_RAS_SHDW	T_RC_SHDW	T_RRD_SHDW	T_WTR_SHDW
R/W-1Bh	R/W-26h	R/W-5h	R/W-5h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-42. SDRAM Timing 1 Shadow Register (SDRTIM1SR) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved
28-25	T_RP_SHDW	0-Fh	Shadow field for T_RP in SDRTIMR1. This field is loaded into T_RP field in SDRAM Timing 1 Register when SIdleAck is asserted.
24-21	T_RCD_SHDW	0-Fh	Shadow field for T_RCD in SDRTIMR1. This field is loaded into T_RCD field in SDRAM Timing 1 Register when SIdleAck is asserted.
20-17	T_WR_SHDW	0-Fh	Shadow field for T_WR in SDRTIMR1. This field is loaded into T_WR field in SDRAM Timing 1 Register when SIdleAck is asserted.
16-12	T_RAS_SHDW	0-1Fh	Shadow field for T_RAS in SDRTIMR1. This field is loaded into T_RAS field in SDRAM Timing 1 Register when SIdleAck is asserted.
11-6	T_RC_SHDW	0-3Fh	Shadow field for T_RC in SDRTIMR1. This field is loaded into T_RC field in SDRAM Timing 1 Register when SIdleAck is asserted.
5-3	T_RRD_SHDW	0-7h	Shadow field for T_RRD in SDRTIMR1. This field is loaded into T_RRD field in SDRAM Timing 1 Register when SIdleAck is asserted.
2-0	T_WTR_SHDW	0-7h	Shadow field for T_WTR in SDRTIMR1. This field is loaded into T_WTR field in SDRAM Timing 1 Register when SIdleAck is asserted.

7.7.1.8 SDRAM Timing 2 Register (SDRTIM2)

SDRTIM2 is shown in Figure 7-11 and described in Table 7-43.

Figure 7-11. SDRAM Timing 2 Register (SDRTIM2)

31	30 28	27 25	24 16
Rsvd	T_XP	Reserved	T_XSNR
R-0	R/W-4h	R-0	R/W-5Fh

15 6	5 3	2 0
T_XSRD	T_RTP	T_CKE
R/W-1FFh	R/W-5h	R/W-3h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-43. SDRAM Timing 2 Register (SDRTIM2) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	T_XP	0-7h	Specifies the minimum number of DDR[x]_CLK[y] cycles from power down exit to any other command except read command, minus 1. For DDR2, this field must be greater than tXP or tCKE. Corresponds to tXP or tCKE AC timing parameter in the DDR2/3 data sheet.
27-25	Reserved	0	Reserved
24-16	T_XSNR	0-1FFh	Specifies the minimum number of DDR[x]_CLK[y] cycles from a self-refresh exit to any other command except a read command, minus 1. Corresponds to tXSNR/tXS AC timing parameter in the DDR2/3 data sheet. Calculated by T_XSNR = (tXSNR/DDR[x]_CLK[y] period) - 1
15-6	T_XSRD	0-3FFh	Specifies the mimimum number of DDR[x]_CLK[y] cycles from a self-refresh exit to a read command, minus 1. Corresponds to tXSRD/tXSDLL AC timing parameter in the DDR2/3 data sheet. Calculated by T_XSRD = (tXSRD or tXSDLL) - 1
5-3	T_RTP	0-7h	Specifies the mimimum number of DDR[x]_CLK[y] cycles from a last read command to a precharge command, minus 1. Corresponds to tRTP AC timing parameter in the DDR2/3 data sheet. Calculated by T_RTP = (tRTP/DDR[x]_CLK[y] period) - 1
2-0	T_CKE	0-7h	Specifies the minimum number of DDR[x]_CLK[y] cycles between transitions on the DDR[x]_CKE[y] pin, minus 1. Corresponds to the tCKE AC timing parameter in the DDR2 data sheet. Calculated by T_CKE = (tCKE/DDR[x]_CLK[y] period) - 1

7.7.1.9 SDRAM Timing 2 Shadow Register (SDRTIM2SR)

The SDRTIM2SR is shown in Figure 7-12 and described in Table 7-44.

Figure 7-12. SDRAM Timing 2 Shadow Register (SDRTIM2SR)

31	30 28	27 25	24 16
Rsvd	T_XP_SHDW	Reserved	T_XSNR_SHDW
R-0	R/W-4h	R-0	R/W-5Fh

15 6	5 3	2 0
T_XSRD_SHDW	T_RTP_SHDW	T_CKE_SHDW
R/W-1FFh	R/W-5h	R/W-3h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-44. SDRAM Timing 2 Shadow Register (SDRTIM2SR) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-29	T_XP_SHDW	0-7h	Shadow field for T_XP in SDRTIMR2. This field is loaded into T_XP field in SDRAM Timing 2 Register when SIdleAck is asserted.
27-25	Reserved	0	Reserved
24-16	T_XSNR_SHDW	0-1FFh	Shadow field for T_XSNR in SDRTIMR2. This field is loaded into T_XSNR field in SDRAM Timing 2 Register when SIdleAck is asserted.
15-6	T_XSRD_SHDW	0-3FFh	Shadow field for T_XSRD in SDRTIMR2. This field is loaded into T_XSRD field in SDRAM Timing 2 Register when SIdleAck is asserted.
5-3	T_RTP_SHDW	0-7h	Shadow field for T_RTP in SDRTIMR2. This field is loaded into T_RTP field in SDRAM Timing 2 Register when SIdleAck is asserted.
2-0	T_CKE_SHDW	0-7h	Shadow field for T_CKE in SDRTIMR2. This field is loaded into T_CKE field in SDRAM Timing 2 Register when SIdleAck is asserted.

7.7.1.10 SDRAM Timing 3 Register (SDRTIM3)

The SDRTIM3 is shown in Figure 7-13 and described in Table 7-45.

Figure 7-13. SDRAM Timing 3 Register (SDRTIM3)

31	21	20	15
Reserved		T_ZQCS	
R/W-4h		R/W-3Fh	

14	13	12	4	3	0
Reserved		T_RFC		T_RASMAX	
R-0		R/W-57h		R/W-8h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-45. SDRAM Timing 3 Register (SDRTIM3) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	4h	Reserved
20-15	T_ZQCS	0-3Fh	Specifies the minimum number of DDR[x]_CLK[y] cycles for ZQCS command, minus 1. This is applicable only for DDR3. Corresponds to T_ZQCS = (tZQCS - 1), where tZQCS is DDR Clock Cycles.
14-13	Reserved	0	Reserved
12-4	T_RFC	0-1FFh	Specifies the minimum number of DDR[x]_CLK[y] cycles from Refresh or Load mode to Refresh or Activate, minus 1. Corresponds to tRFC AC timing parameter in the DDR2/3 data sheet. Calculated by T_RFC = (tRFC/DDR[x]_CLK[y] period) - 1
3-0	T_RASMAX	0-Fh	Specifies the maximum number of refresh rate intervals from Active to Precharge command. This field must be programmed to Fh for DDR2/3 SDRAM types.

7.7.1.11 SDRAM Timing 3 Shadow Register (SDRTIM3SR)

The SDRTIM3SR is shown in Figure 7-14 and described in Table 7-46.

Figure 7-14. SDRAM Timing 3 Shadow Register (SDRTIM3SR)

31	21	20	15
Reserved		T_ZQCS_SHDW	
R/W-4h		R/W-3Fh	

14	13	12	4	3	0
Reserved		T_RFC _SHDW		T_RASMAX _SHDW	
R-0		R/W-57h		R/W-8h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-46. SDRAM Timing 3 Shadow Register (SDRTIM3SR) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	4h	Reserved
20-15	T_ZQCS _SHDW	0-3Fh	Shadow field for T_ZQCS in SDRTIMR3. This field is loaded into T_ZQCS field in SDRAM Timing 3 Register when SIdleAck is asserted.
14-13	Reserved	0	Reserved
12-4	T_RFC _SHDW	0-1FFh	Shadow field for T_RFC in SDRTIMR3. This field is loaded into T_RFC field in SDRAM Timing 3 Register when SIdleAck is asserted.
3-0	T_RASMAX _SHDW	0-Fh	Shadow field for T_RASMAX in SDRTIMR3. This field is loaded into T_RASMAX field in SDRAM Timing 3 Register when SIdleAck is asserted.

7.7.1.12 Power Management Control Register (PMCR)

The PMCR is shown in Figure 7-15 and described in Table 7-47.

Figure 7-15. Power Management Control Register (PMCR)

31	16
Reserved	
R-0	

15 12	11	10 8	7 4	3 0
PD_TIM	DP_DEN	LP_MODE	SR_TIM	CS_TIM
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-47. Power Management Control Register (PMCR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-12	PD_TIM		Power Management timer for Power Down. The DDR controller puts the external SDRAM in Power-Down mode after the DDR controller is idle for these number of DDR clock cycles and if LP_MODE field is set to 4h.
		0	Immediately enter Power down Mode
		1h	After 16 DDR cycles
		2h	After 32 DDR cycles
		3h	After 64 DDR cycles
		4h	After 128 DDR cycles
		5h	After 256 DDR cycles
		6h	After 512 DDR cycles
		7h	After 1024 DDR cycles
		8h	After 2048 DDR cycles
		9h	After 4096 DDR cycles
		Ah	After 8192 DDR cycles
		Bh	After 16384 DDR cycles
		Ch	After 32768 DDR cycles
		Dh	After 65536 DDR cycles
		Eh	After 131072 DDR cycles
		Fh	After 262144 DDR cycles
11	DP_DEN		Deep Power Down Enable
		0	Normal Operation
		1	Overrides LP_MODE field setting
10-8	LP_MODE		Automatic Power Management Enable
		1h	Clock Stop Mode
		2h	Self Refresh Mode
		4h	Power Down Mode
		0, 3h, 5h-7h	Disable Automatic power management

Table 7-47. Power Management Control Register (PMCR) Field Descriptions (continued)

Bit	Field	Value	Description
7-4	SR_TIM		Power Management timer for Self Refresh. The DDR controller puts the external SDRAM in Self Refresh Mode after the DDR controller is idle for these number of DDR clock cycles and if LP_MODE is set to 2h.
		0	Immediately enter Self Refresh Mode
		1h	Enter Self Refresh Mode after 16 DDR Clocks
		2h	Enter Self Refresh Mode after 32 DDR Clocks
		3h	Enter Self Refresh Mode after 64 DDR Clocks
		4h	Enter Self Refresh Mode after 128 DDR Clocks
		5h	Enter Self Refresh Mode after 256 DDR Clocks
		6h	Enter Self Refresh Mode after 512 DDR Clocks
		7h	Enter Self Refresh Mode after 1024 DDR Clocks
		8h	Enter Self Refresh Mode after 2048 DDR Clocks
		9h	Enter Self Refresh Mode after 4096 DDR Clocks
		Ah	Enter Self Refresh Mode after 8192 DDR Clocks
		Bh	Enter Self Refresh Mode after 16384 DDR Clocks
		Ch	Enter Self Refresh Mode after 32768 DDR Clocks
		Dh	Enter Self Refresh Mode after 65536 DDR Clocks
		Eh	Enter Self Refresh Mode after 131072 DDR Clocks
		Fh	Enter Self Refresh Mode after 262144 DDR Clocks
3-0	CS_TIM		Power Management timer for Clock Stop. The DDR controller puts the external SDRAM in clock stop mode after the DDR controller is idle for these number of DDR clock cycles and if LP_MODE field is set to 1.
		0	Immediately enter Clock Stop Mode
		1h	Enter Clock Stop Mode after 16 clocks
		2h	Enter Clock Stop Mode after 32 clocks
		3h	Enter Clock Stop Mode after 64 clocks
		4h	Enter Clock Stop Mode after 128 clocks
		5h	Enter Clock Stop Mode after 256 clocks
		6h	Enter Clock Stop Mode after 512 clocks
		7h	Enter Clock Stop Mode after 1024 clocks
		8h	Enter Clock Stop Mode after 2048 clocks
		9h	Enter Clock Stop Mode after 4096 clocks
		Ah	Enter Clock Stop Mode after 8192 clocks
		Bh	Enter Clock Stop Mode after 16384 clocks
		Ch	Enter Clock Stop Mode after 32768 clocks
		Dh	Enter Clock Stop Mode after 65536 clocks
		Eh	Enter Clock Stop Mode after 131072 clocks
		Fh	Enter Clock Stop Mode after 262144 clocks

7.7.1.13 Power Management Control Shadow Register (PMCSR)

The PMCSR is shown in Figure 7-16 and described in Table 7-48.

Figure 7-16. Power Management Control Shadow Register (PMCSR)

31	16
Reserved	
R-0	

15 12	11 8	7 4	3 0
PD_TIM_SHDW	Reserved	SR_TIM_SHDW	CS_TIM_SHDW
R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-48. Power Management Control Shadow Register (PMCSR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-12	PD_TIM_SHDW	0-Fh	Shadow field for PD_TIM in PMCR. This field is loaded into PD_TIM field in Power Management Control Register when SidleAck is asserted.
11-8	Reserved	0	Reserved
7-4	SR_TIM_SHDW	0-Fh	Shadow field for SR_TIM in PMCR. This field is loaded into SR_TIM field in Power Management Control Register when SidleAck is asserted.
3-0	CS_TIM_SHDW	0-Fh	Shadow field for CS_TIM in PMCR. This field is loaded into CS_TIM field in Power Management Control Register when SidleAck is asserted.

7.7.1.14 Peripheral Bus Burst Priority Register (PBBPR)

The PBBPR is shown in Figure 7-17 and described in Table 7-49.

Figure 7-17. Peripheral Bus Burst Priority Register (PBBPR)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		PR_OLD_COUNT	
R-0		R/W-FFh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-49. Peripheral Bus Burst Priority Register (PBBPR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Any writes to these bit(s) must always have a value of 0.
7-0	PR_OLD_COUNT	0-FFh	Priority raise old counter. Specifies the number of memory transfers after which the DDR2/3 memory controller elevates the priority of the oldest command in the command FIFO.
		0	1 memory transfer
		1h	2 memory transfers
		2h	3 memory transfers
		3h-FFh	4 to 256 memory transfers

7.7.1.15 Performance Counter 1 Register (PRFCNT1)

The PRFCNT1 is shown in Figure 7-18 and described in Table 7-50.

Figure 7-18. Performance Counter 1 Register (PRFCNT1)

31	0
COUNTER1	
R-00000000h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-50. Performance Counter 1 Register (PRFCNT1) Field Descriptions

Bit	Field	Value	Description
31- 0	COUNTER1	0-FFFFFFFFh	32-bit counter that can be configured as specified in the Performance Counter Config Register (PRFCNTCFG) and Performance Counter Master Region Select Register (PRFCNTSEL).

7.7.1.16 Performance Counter 2 Register (PRFCNT2)

The PRFCNT2 is shown in Figure 7-19 and described in Table 7-51.

Figure 7-19. Performance Counter 2 Register (PRFCNT2)

31	0
COUNTER2	
R-00000000h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-51. Performance Counter 2 Register (PRFCNT2) Field Descriptions

Bit	Field	Value	Description
31- 0	COUNTER2	0-FFFFFFFFh	32-bit counter that can be configured as specified in the Performance Counter Config Register (PRFCNTCFG) and Performance Counter Master Region Select Register (PRFCNTSEL).

7.7.1.17 Performance Counter Config Register (PRFCNTCFG)

The PRFCNTCFG is shown in Figure 7-20 and described in Table 7-52.

Figure 7-20. Performance Counter Config Register (PRFCNTCFG)

31	30	29	20	19	16
CNTR2_MCONNID_EN	CNTR2_REGION_EN	RESERVED		CNTR2_CFG	
R/W-0h	R/W-0h	R-0h		R/W-1h	

15	14	13	4	3	0
CNTR1_MCONNID_EN	CNTR1_REGION_EN	RESERVED		CNTR1_CFG	
R/W-0h	R/W-0h	R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-52. Performance Counter Config Register (PRFCNTCFG) Field Descriptions

Bit	Field	Value	Description
31	CNTR2_MCONNID_EN	0-1	MConnID filter enable for Performance Counter 2 register.
30	CNTR2_REGION_EN	0-1	Chip Select filter enable for Performance Counter 2 register.
29-20	RESERVED	0	Reserved.
19-16	CNTR2_CFG	0-Ah	Filter configuration for Performance Counter 2. For more details, see Section 7.7.1.16.
15	CNTR1_MCONNID_EN	0-1	MConnID filter enable for Performance Counter 1 register.
14	CNTR1_REGION_EN	0-1	Chip Select filter enable for Performance Counter 1 register.
13-4	RESERVED	0	Reserved.
3-0	CNTR1_CFG	0-Ah	Filter configuration for Performance Counter 1. For more details, see Section 7.7.1.15.

7.7.1.18 Performance Counter Master Region Select Register (PRFCNTSEL)

The PRFCNTSEL is shown in Figure 7-21 and described in Table 7-53.

Figure 7-21. Performance Counter Master Region Select Register (PRFCNTSEL)

31	24	23	18	17	16
MCONNID2		RESERVED		REGION_SEL2	
R/W-0h		R-0h		R/W-0h	

15	8	7	2	1	0
MCONNID1		RESERVED		REGION_SEL1	
R/W-0h		R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-53. Performance Counter Master Region Select Register (PRFCNTSEL) Field Descriptions

Bit	Field	Value	Description
31-24	MCONNID2	0-3Fh	MConnID for Performance Counter 2 register.
23-18	RESERVED	0	Reserved.
17-16	REGION_SEL2	0-3h	MAddrSpace for Performance Counter 2 register.
15-8	MCONNID1	0-3Fh	MConnID for Performance Counter 1 register.
7-2	RESERVED	0	Reserved.
1-0	REGION_SEL1	0-3h	MAddrSpace for Performance Counter 1 register.

7.7.1.19 Performance Counter Time Register (PRFCNTTIM)

The PRFCNTTIM is shown in Figure 7-22 and described in Table 7-54.

Figure 7-22. Performance Counter Time Register (PRFCNTTIM)

31	0
TOTAL_TIME	
R-00000000h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-54. Performance Counter Time Register (PRFCNTTIM) Field Descriptions

Bit	Field	Value	Description
31- 0	TOTAL_TIME	0-FFFFFFFFh	32-bit counter that continuously counts the number of DDR2/DDR3 Controller functional clock cycles elapsed after the DDR2/DDR3 Controller is brought out of reset.

7.7.1.20 End of Interrupt Register (EOI)

The EOI is shown in Figure 7-23 and described in Table 7-55.

Figure 7-23. End of Interrupt Register (EOI)

31	16
Reserved	
R-0	

15	1	0
Reserved		EOI
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-55. End of Interrupt Register (EOI) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	EOI		Software End of Interrupt Control.
		0	System OCP Interrupt
		1	Reserved

7.7.1.21 System OCP Interrup RAW Status Register (SOIRSR)

The SOIRSR is shown in Figure 7-24 and described in Table 7-56.

Figure 7-24. System OCP Interrup RAW Status Register (SOIRSR)

31	16
Reserved	
R-0	

15	1	0
Reserved		ERRSYS
R-0		W1S-0

LEGEND: R = Read only; W1S = Write 1 to set, write of 0 has no effect; *-n* = value after reset

Table 7-56. System OCP Interrup RAW Status Register (SOIRSR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	ERRSYS		Raw status of system OCP interrupt for command or address error.
		0	Writing a 0 has no effect.
		1	Write 1 to set the raw status.

7.7.1.22 Sytem OCP Interrupt Status Register (SOISR)

The SOISR is shown in Figure 7-25 and described in Table 7-57.

Figure 7-25. Sytem OCP Interrupt Status Register (SOISR)

31	16
Reserved	
R-0	

15	1	0
Reserved		ERRSYS
R-0		W1C-0

LEGEND: R = Read only; W1C = Write 1 to clear, write of 0 has no effect; *-n* = value after reset

Table 7-57. Sytem OCP Interrupt Status Register (SOISR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	ERRSYS		Enabled status of system OCP interrupt for SDRAM command or address error.
		0	Writing a 0 has no effect.
		1	Write 1 to clear the status after interrupt has been serviced (raw status gets cleared, that is, even if not enabled).

7.7.1.23 Sytem OCP Interrupt Enable Set Register (SOIESR)

The SOIESR is shown in Figure 7-26 and described in Table 7-58.

Figure 7-26. Sytem OCP Interrupt Enable Set Register (SOIESR)

31	16
Reserved	
R-0	

15	1	0
Reserved		ERRSYSSET
R-0		W1S-0

LEGEND: R = Read only; W1S = Write 1 to set, a write of 0 has no effect; -*n* = value after reset

Table 7-58. Sytem OCP Interrupt Enable Set Register (SOIESR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	ERRSYSSET		Enable set for system OCP interrupt for SDRAM command or address error. Write 1 to set ERRSYSSET and ERRSYSCLR bit in the System OCP Interrupt Enable Clear Register.
		0	Interrupt is not Enabled
		1	Interrupt is enabled.

7.7.1.24 Sytem OCP Interrupt Enable Clear Register (SOIECR)

The SOIECR is shown in Figure 7-27 and described in Table 7-59.

Figure 7-27. Sytem OCP Interrupt Enable Clear Register (SOIECR)

31	16
Reserved	
R-0	

15	1	0
Reserved		ERRSYSCLR
R-0		W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear, a write of 0 has no effect; -*n* = value after reset

Table 7-59. Sytem OCP Interrupt Enable Clear Register (SOIECR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	ERRSYSCLR		Enable clear for system OCP interrupt for SDRAM command or address error. Write 1 to clear ERRSYSCLR and ERRSYSSET bit in System OCP Interrupt Enable Set Register.
		0	No effect
		1	Interrupt is enabled.

7.7.1.25 SDRAM Output Impedance Calibration Configuration Register (ZQCR)

The ZQCR is shown in Figure 7-28 and described in Table 7-60.

Figure 7-28. SDRAM Output Impedance Calibration Configuration Register (ZQCR)

31	30	29	28	27	24
CS1EN	CS0EN	DUALCALEN	SEFXITEN	Reserved	
R/W-0	R/W-1h	R/W-0	R/W-1h	R-0	

23	20	19	18	17	16
Reserved		ZQINITMULT		ZQCLMULT	
R-0		R/W-1h		R/W-3h	

15	0
REFINTERVAL	
R/W-4C00h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-60. SDRAM Output Impedance Calibration Configuration Register (ZQCR) Field Descriptions

Bit	Field	Value	Description
31	CS1EN	0-1	Writing a 1 enables ZQ calibration for CS1
30	CS0EN	0-1	Writing a 1 enables ZQ calibration for CS0
29	DUALCALEN	0-1	ZQ Dual Calibration Enable. Allows both ranks (chip selects) to be ZQ calibrated simultaneously. Setting this bit requires both chip selects to have a separate calibration resistor per device.
28	SEFXITEN	0-1	ZQCL on Self-Refresh, Active Power Down and Prechare Power-Down exit enable. Writing a 1 enables the issuing of ZQCL on Self-Refresh, Active Power-Down and Precharge Power Down exit.
27-20	Reserved	0	Reserved
19-18	ZQINITMULT	0-3h	Indicates number of ZQCL intervals that make up a ZQINIT interval, minus 1
17-16	ZQCLMULT	0-3h	Indicates number of ZQCS intervals that make up a ZQCL interval, minus 1
15-0	REFINTERVAL	0-FFFFh	Number of refresh periods between ZQCS commands. This field supports between 1 refresh period to 256 ms between ZQCS calibration commands. Refresh period is defined by RR field in SDRAM Refresh Control Register.

7.7.1.26 DDR PHY Control Register (DDRPHYCR)

The DDRPHYCR is shown in Figure 7-29 and described in Table 7-61.

Figure 7-29. DDR PHY Control Register (DDRPHYCR)

31	21	20	19	16
Reserved		EN_DYN _PWRDN	Reserved	
R-0		R/W-0	R-0	

15	10	9	8	7	5	4	0
Reserved		RD_LODT		Reserved		RL	
R-0		R/W-1		R-0		R/W-Bh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-61. DDR PHY Control Register (DDRPHYCR) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	EN_DYN_PWRDN		Enables dynamically powering down the IO Receiver when not performing a read.
			For PG1.x devices:
		0	IO Receiver for DQ, DQS always powered up
		1	IO Receiver for DQ, DQS powered up during a read
			For PG2.x devices:
		0	IO Receiver always powered up
		1	IO Receiver powered up during a read
19-10	Reserved	0	Reserved
9-8	RD_LODT		Select controller termination during read accesses.
		0	ODT off
		1h	ODT enable (50 ohms). Recommended for better signal integrity.
			For control lines: Rext
			For other than control lines: (0.88 x Rext)
		2h	ODT off
		3h	ODT enable (100 ohms)
			For control lines: 2 × Rext
			For other than control lines: (2 × 0.88 x Rext)
7-5	Reserved	0	Reserved
4-0	RL	0-1Fh	Defines the latency for read data from DDR SDRAM in number of 1x cycles. The value applied should be equal to the required value minus one. The maximum read latency supported by the DDR PHY is equal to CAS latency plus 7 clock cycles. The minimum read latency must be equal to CAS latency plus 1 clock cycles.

7.7.1.27 DDR PHY Control Shadow Register (DDRPHYCSR)

The DDRPHYCSR is shown in Figure 7-30 and described in Table 7-62.

Figure 7-30. DDR PHY Control Shadow Register (DDRPHYCSR)

31	21	20	19	16
Reserved		EN_DYN _PWRDN _SHDW	Reserved	
R-0		R/W-0	R-0	

15	10	9	8	7	5	4	0
Reserved		RD_LODT_SHDW		Reserved		RL_SHDW	
R-0		R/W-1		R-0		R/W-Bh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-62. DDR PHY Control Shadow Register (DDRPHYCSR) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	EN_DYN_PWRDN_SHDW	0-1	Shadow field for EN_DYN_PWRDN in DDRPHYCR. This bit is loaded into EN_DYN_PWRDN bit in DDR PHY Control Register when SIdleAck is asserted.
19-10	Reserved	0	Reserved
9-8	RD_LODT_SHDW	0-3h	Shadow field for RD_LODT in DDRPHYCR. This field is loaded into RD_LODT field in DDR PHY Control Register when SIdleAck is asserted.
7-5	Reserved	0	Reserved
4-0	RL_SHDW	0-1Fh	Shadow field for RL in DDRPHYCR. This field is loaded into RL field in DDR PHY Control Register when SIdleAck is asserted.

7.7.2 DDR2/3 PHY Registers

Table 7-63 lists the memory-mapped registers for the DDR2/3 PHY. For the base address of these registers, see Table 1-12.

Table 7-63. Memory Mapped Registers for DDR2/3 PHY

Address Offset	Acronym	Register Description	Section
Ch	CMD0_IO_CONFIG_I_0	Command 0 Address/Command Pad Configuration Register	Section 7.7.2.1
10h	CMD0_IO_CONFIG_I_CLK_0	Command 0 Clock Pad Configuration Register	Section 7.7.2.2
14h	CMD0_IO_CONFIG_SR_0	Command 0 Address/Command Slew Rate Configuration Register	Section 7.7.2.3
18h	CMD0_IO_CONFIG_SR_CLK_0	Command 0 Clock Pad Slew Rate Configuration Register	Section 7.7.2.4
1Ch	CMD0_REG_PHY_CTRL_SLAVE_RATIO_0	Command 0 Address/Command Slave Ratio Register	Section 7.7.2.5
2Ch	CMD0_REG_PHY_INVERT_CLKOUT_0	Command 0 Invert Clockout Selection Register	Section 7.7.2.6
40h	CMD1_IO_CONFIG_I_0	Command 1 Address/Command Pad Configuration Register	Section 7.7.2.1
44h	CMD1_IO_CONFIG_I_CLK_0	Command 1 Clock Pad Configuration Register	Section 7.7.2.2
48h	CMD1_IO_CONFIG_SR_0	Command 1 Address/Command Slew Rate Configuration Register	Section 7.7.2.3
4Ch	CMD1_IO_CONFIG_SR_CLK_0	Command 1 Clock Pad Slew Rate Configuration Register	Section 7.7.2.4
50h	CMD1_REG_PHY_CTRL_SLAVE_RATIO_0	Command 1 Address/Command Slave Ratio Register	Section 7.7.2.5
60h	CMD1_REG_PHY_INVERT_CLKOUT_0	Command 1 Invert Clockout Selection Register	Section 7.7.2.6
74h	CMD2_IO_CONFIG_I_0	Command 2 Address/Command Pad Configuration Register	Section 7.7.2.1
78h	CMD2_IO_CONFIG_I_CLK_0	Command 2 Clock Pad Configuration Register	Section 7.7.2.2
7Ch	CMD2_IO_CONFIG_SR_0	Command 2 Address/Command Slew Rate Configuration Register	Section 7.7.2.3
80h	CMD2_IO_CONFIG_SR_CLK_0	Command 2 Clock Pad Slew Rate Configuration Register	Section 7.7.2.4
84h	CMD2_REG_PHY_CTRL_SLAVE_RATIO_0	Command 2 Address/Command Slave Ratio Register	Section 7.7.2.5
94h	CMD2_REG_PHY_INVERT_CLKOUT_0	Command 2 Invert Clockout Selection Register	Section 7.7.2.6
A8h	DATA0_IO_CONFIG_I_0	Data Macro 0 Data Pad Configuration Register	Section 7.7.2.7
ACh	DATA0_IO_CONFIG_I_CLK_0	Data Macro 0 Data Strobe Pad Configuration Register	Section 7.7.2.8
B0h	DATA0_IO_CONFIG_SR_0	Data Macro 0 Data Slew Rate Configuration Register	
B4h	DATA0_IO_CONFIG_SR_CLK_0	Data Macro 0 Data Strobe Slew Rate Configuration Register	
C8h	DATA0_REG_PHY_RD_DQS_SLAVE_RATIO_0	Data Macro 0 Read DQS Slave Ratio Register	Section 7.7.2.9
DCh	DATA0_REG_PHY_WR_DQS_SLAVE_RATIO_0	Data Macro 0 Write DQS Slave Ratio Register	Section 7.7.2.10
108h	DATA0_REG_PHY_FIFO_WE_SLAVE_RATIO_0	Data Macro 0 DQS Gate Slave Ratio Register	Section 7.7.2.11

Table 7-63. Memory Mapped Registers for DDR2/3 PHY (continued)

Address Offset	Acronym	Register Description	Section
120h	DATA0_REG_PHY_WR_DATA_SLAVE_RATIO_0	Data Macro 0 Write Data Slave Ratio Register	Section 7.7.2.12
134h	DATA0_REG_PHY_USE_RANK0_DELAYS	Data Macro 0 Delay Selection Register	Section 7.7.2.13
14Ch	DATA1_IO_CONFIG_I_0	Data Macro 1 Data Pad Configuration Register	Section 7.7.2.7
150h	DATA1_IO_CONFIG_I_CLK_0	Data Macro 1 Data Strobe Pad Configuration Register	Section 7.7.2.8
154h	DATA1_IO_CONFIG_SR_0	Data Macro 1 Data Slew Rate Configuration Register	
158h	DATA1_IO_CONFIG_SR_CLK_0	Data Macro 1 Data Strobe Slew Rate Configuration Register	
16Ch	DATA1_REG_PHY_RD_DQS_SLAVE_RATIO_0	Data Macro 1 Read DQS Slave Ratio Register	Section 7.7.2.9
180h	DATA1_REG_PHY_WR_DQS_SLAVE_RATIO_0	Data Macro 1 Write DQS Slave Ratio Register	Section 7.7.2.10
1ACh	DATA1_REG_PHY_FIFO_WE_SLAVE_RATIO_0	Data Macro 1 DQS Gate Slave Ratio Register	Section 7.7.2.11
1C4h	DATA1_REG_PHY_WR_DATA_SLAVE_RATIO_0	Data Macro 1 Write Data Slave Ratio Register	Section 7.7.2.12
1D8h	DATA1_REG_PHY_USE_RANK0_DELAYS	Data Macro 1 Delay Selection Register	Section 7.7.2.13
1F0h	DATA2_IO_CONFIG_I_0	Data Macro 2 Data Pad Configuration Register	Section 7.7.2.7
1F4h	DATA2_IO_CONFIG_I_CLK_0	Data Macro 2 Data Strobe Pad Configuration Register	Section 7.7.2.8
1F8h	DATA2_IO_CONFIG_SR_0	Data Macro 2 Data Slew Rate Configuration Register	
1FCh	DATA2_IO_CONFIG_SR_CLK_0	Data Macro 2 Data Strobe Slew Rate Configuration Register	
210h	DATA2_REG_PHY_RD_DQS_SLAVE_RATIO_0	Data Macro 2 Read DQS Slave Ratio Register	Section 7.7.2.9
224h	DATA2_REG_PHY_WR_DQS_SLAVE_RATIO_0	Data Macro 2 Write DQS Slave Ratio Register	Section 7.7.2.10
250h	DATA2_REG_PHY_FIFO_WE_SLAVE_RATIO_0	Data Macro 2 DQS Gate Slave Ratio Register	Section 7.7.2.11
268h	DATA2_REG_PHY_WR_DATA_SLAVE_RATIO_0	Data Macro 2 Write Data Slave Ratio Register	Section 7.7.2.12
27Ch	DATA2_REG_PHY_USE_RANK0_DELAYS	Data Macro 2 Delay Selection Register	Section 7.7.2.13
294h	DATA3_IO_CONFIG_I_0	Data Macro 3 Data Pad Configuration Register	Section 7.7.2.7
298h	DATA3_IO_CONFIG_I_CLK_0	Data Macro 3 Data Strobe Pad Configuration Register	Section 7.7.2.8
29Ch	DATA3_IO_CONFIG_SR_0	Data Macro 3 Data Slew Rate Configuration Register	
2A0h	DATA3_IO_CONFIG_SR_CLK_0	Data Macro 3 Data Strobe Slew Rate Configuration Register	
2B4h	DATA3_REG_PHY_RD_DQS_SLAVE_RATIO_0	Data Macro 3 Read DQS Slave Ratio Register	Section 7.7.2.9
2C8h	DATA3_REG_PHY_WR_DQS_SLAVE_RATIO_0	Data Macro 3 Write DQS Slave Ratio Register	Section 7.7.2.10
2F4h	DATA3_REG_PHY_FIFO_WE_SLAVE_RATIO_0	Data Macro 3 DQS Gate Slave Ratio Register	Section 7.7.2.11

Table 7-63. Memory Mapped Registers for DDR2/3 PHY (continued)

Address Offset	Acronym	Register Description	Section
30Ch	DATA3_REG_PHY_WR_DATA_SLAVE_RATIO_0	Data Macro 3 Write Data Slave Ratio Register	Section 7.7.2.12
320h	DATA3_REG_PHY_USE_RANK0_DELAYS	Data Macro 3 Delay Selection Register	Section 7.7.2.13
358h	DDR_VTP_CTRL_0	DDR VTP Control Register	Section 7.7.2.14

7.7.2.1 Command 0/1/2 Address/Command Pad Configuration Register (CMD0/1/2_IO_CONFIG_I_0)

The CMD0/1/2_IO_CONFIG_I_0 is shown in Figure 7-31 and described in Table 7-64. See *IO Configuration and Slew Rate Control*.

Figure 7-31. Command 0/1/2 Address/Command Pad Configuration Register (CMD0/1/2_IO_CONFIG_I_0)

31	16
Reserved	
R-0	

15	3	2	0
Reserved		ADDR_CMD_CFG	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-64. Command 0/1/2 Address/Command Pad Configuration Register (CMD0/1/2_IO_CONFIG_I_0) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	ADDR_CMD_CFG	0-7h	To configure Address/Command I/O Pad for pull up/pull down output impedance.

7.7.2.2 Command 0/1/2 Clock Pad Configuration Register (CMD0/1/2_IO_CONFIG_I_CLK_0)

The CMD0/1/2_IO_CONFIG_I_CLK_0 is shown in Figure 7-32 and described in Table 7-65. See *IO Configuration and Slew Rate Control*.

Figure 7-32. Command 0/1/2 Clock Pad Configuration Register (CMD0/1/2_IO_CONFIG_I_CLK_0)

31	16
Reserved	
R-0	

15	3	2	0
Reserved		CLK_CFG	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-65. Command 0/1/2 Clock Pad Configuration Register (CMD0/1/2_IO_CONFIG_I_CLK_0) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLK_CFG	0-7h	To configure Clock output Pad for pull up/pull down output impedance.

7.7.2.3 Command 0/1/2 Address/Command Slew Rate Configuration Register (CMD0/1/2_IO_CONFIG_SR_0)

The CMD0/1/2_IO_CONFIG_SR_0 is shown in Figure 7-33 and described in Table 7-66.

Figure 7-33. Command 0/1/2 Address/Command Slew Rate Configuration Register (CMD0/1/2_IO_CONFIG_SR_0)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		SR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-66. Command 0/1/2 Address/Command Slew Rate Configuration Register (CMD0/1/2_IO_CONFIG_SR_0) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	SR		2 bit to program addr/cmd IO Pads output Slew Rate
		0	Fastest Slew Rate
		1h	Fast Slew Rate
		2h	Slow Slew Rate
		3h	Slowest Slew Rate

7.7.2.4 Command 0/1/2 Clock Pad Slew Rate Configuration Register (CMD0/1/2_IO_CONFIG_SR_CLK_0)

The CMD0/1/2_IO_CONFIG_SR_CLK_0 is shown in Figure 7-34 and described in Table 7-67.

Figure 7-34. Command 0/1/2 Clock Pad Slew Rate Configuration Register (CMD0/1/2_IO_CONFIG_SR_CLK_0)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		SR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-67. Command 0/1/2 Clock Pad Slew Rate Configuration Register (CMD0/1/2_IO_CONFIG_SR_CLK_0) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	SR		2 bit to program clock IO Pads (CK/CK#) output Slew Rate
		0	Fastest Slew Rate
		1h	Fast Slew Rate
		2h	Slow Slew Rate
		3h	Slowest Slew Rate

7.7.2.5 Command 0/1/2 Address/Command Slave Ratio Register (CMD0/1/2_REG_PHY_CTRL_SLAVE_RATIO_0)

The CMD0/1/2_REG_PHY_CTRL_SLAVE_RATIO_0 is shown in Command 0/1/2 Address/Command Slave Ratio Register (CMD0/1/2_REG_PHY_CTRL_SLAVE_RATIO_0) and described in Table 7-68.

Command 0/1/2 Address/Command Slave Ratio Register (CMD0/1/2_REG_PHY_CTRL_SLAVE_RATIO_0)

31	16
Reserved	
R-0	

15	10	9	0
Reserved		CMD_SLAVE_RATIO	
R-0		R/W-80h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-68. Command 0/1/2 Address/Command Slave Ratio Register (CMD0/1/2_REG_PHY_CTRL_SLAVE_RATIO_0) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-0	CMD_SLAVE_RATIO	0-3FFh	Ratio value for address/command launch timing in DDR PHY macro. This is the fraction of a clock cycle represented by the shift to be applied to the read DQS in units of 256ths. In other words, the full-cycle tap value from the master DLL is scaled by this number over 256 to get the delay value for the slave delay line.

7.7.2.6 Command 0/1/2 Invert Clockout Selection Register (CMD0/1/2_REG_PHY_INVERT_CLKOUT_0)

The CMD0/1/2_REG_PHY_INVERT_CLKOUT_0 is shown in Figure 7-35 and described in Table 7-69.

Figure 7-35. Command 0/1/2 Invert Clockout Selection Register (CMD0/1/2_REG_PHY_INVERT_CLKOUT_0)

31	16
Reserved	
R-0	

15	1	0
Reserved		INVERT_CLK _SEL
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-69. Command 0/1/2 Invert Clockout Selection Register (CMD0/1/2_REG_PHY_INVERT_CLKOUT_0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	INVERT_CLK_SEL		Inverts the polarity of DRAM clock.
		0	Core clock is passed on to DRAM
		1	Inverted core clock is passed on to DRAM

7.7.2.7 Data Macro 0/1/2/3 Data Pad Configuration Register (DATA0/1/2/3_IO_CONFIG_I_0)

The DATA0/1/2/3_IO_CONFIG_I_0 is shown in Figure 7-36 and described in Table 7-70. See *IO Configuration and Slew Rate Control*.

Figure 7-36. Data Macro 0/1/2/3 Data Pad Configuration Register (DATA0/1/2/3_IO_CONFIG_I_0)

31	16
Reserved	
R-0	

15	3	2	0
Reserved		DQ_DM_CFG	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-70. Data Macro 0/1/2/3 Data Pad Configuration Register (DATA0/1/2/3_IO_CONFIG_I_0) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	DQ_DM_CFG	0-7h	To configure DQ/DM IO Pad Pull up/Pull down output impedance.

7.7.2.8 Data Macro 0/1/2/3 Data Strobe Pad Configuration Register (DATA0/1/2/3_IO_CONFIG_I_CLK_0)

The DATA0/1/2/3_IO_CONFIG_I_CLK_0 is shown in Figure 7-37 and described in Table 7-71. See *IO Configuration and Slew Rate Control*.

Figure 7-37. Data Macro 0/1/2/3 Data Strobe Pad Configuration Register (DATA0/1/2/3_IO_CONFIG_I_CLK_0)

31	16
Reserved	
R-0	

15	3	2	0
Reserved		DQS_CFG	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-71. Data Macro 0/1/2/3 Data Strobe Pad Configuration Register (DATA0/1/2/3_IO_CONFIG_I_CLK_0) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	DQS_CFG	0-7h	To configure strobe IO Pads (DQS/DQS#) Pull up/Pull down output impedance.

7.7.2.9 Data Macro 0/1/2/3 Read DQS Slave Ratio Register (DATA0/1/2/3_REG_PHY_RD_DQS_SLAVE_RATIO_0)

The DATA0/1/2/3_REG_PHY_RD_DQS_SLAVE_RATIO_0 is shown in Figure 7-38 and described in Table 7-72.

Figure 7-38. Data Macro 0/1/2/3 Read DQS Slave Ratio Register (DATA0/1/2/3_REG_PHY_RD_DQS_SLAVE_RATIO_0)

31	20	19	16
Reserved		RD_DQS_SLAVE_RATIO_CS1	
R-0		R/W-40h	

15	10	9	0
RD_DQS_SLAVE_RATIO_CS1		RD_DQS_SLAVE_RATIO_CS0	
R/W-40h		R/W-40h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

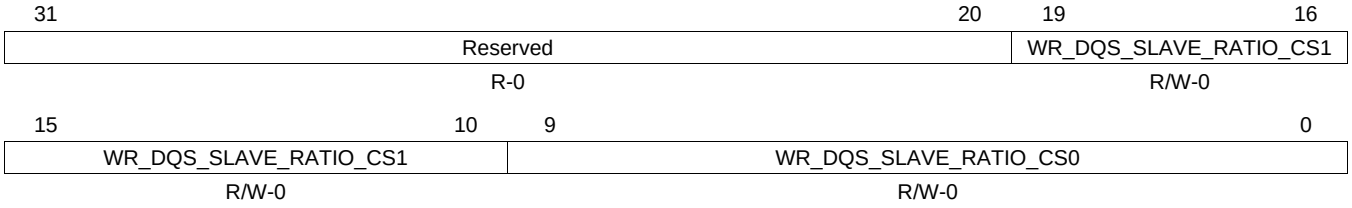
Table 7-72. Data Macro 0/1/2/3 Read DQS Slave Ratio Register (DATA0/1/2/3_REG_PHY_RD_DQS_SLAVE_RATIO_0) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reserved
19-10	RD_DQS_SLAVE_RATIO_CS1	0-3FFh	Ratio value for Read DQS slave DLL for CS1. This is the fraction of a clock cycle represented by the shift to be applied to the read DQS in units of 256ths. In other words, the full-cycle tap value from the master DLL is scaled by this number over 256 to get the delay value for the slave delay line.
9-0	RD_DQS_SLAVE_RATIO_CS0	0-3FFh	Ratio value for Read DQS slave DLL for CS0. This is the fraction of a clock cycle represented by the shift to be applied to the read DQS in units of 256ths. In other words, the full-cycle tap value from the master DLL is scaled by this number over 256 to get the delay value for the slave delay line.

7.7.2.10 Data Macro 0/1/2/3 Write DQS Slave Ratio Register (DATA0/1/2/3_REG_PHY_WR_DQS_SLAVE_RATIO_0)

The DATA0/1/2/3_REG_PHY_WR_DQS_SLAVE_RATIO_0 is shown in Figure 7-39 and described in Table 7-73.

Figure 7-39. Data Macro 0/1/2/3 Write DQS Slave Ratio Register (DATA0/1/2/3_REG_PHY_WR_DQS_SLAVE_RATIO_0)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

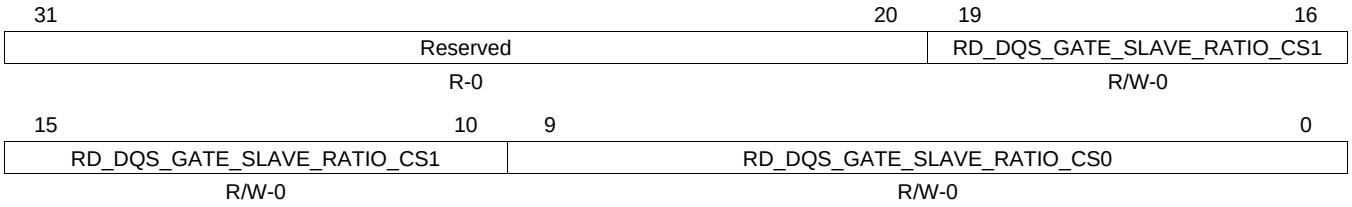
Table 7-73. Data Macro 0/1/2/3 Write DQS Slave Ratio Register (DATA0/1/2/3_REG_PHY_WR_DQS_SLAVE_RATIO_0) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reserved
19-10	WR_DQS_SLAVE_RATIO_CS1	0-3FFh	Ratio value for Write DQS slave DLL for CS1. This is the fraction of a clock cycle represented by the shift to be applied to the write DQS in units of 256ths. In other words, the full-cycle tap value from the master DLL is scaled by this number over 256 to get the delay value for the slave delay line.
9-0	WR_DQS_SLAVE_RATIO_CS0	0-3FFh	Ratio value for Write DQS slave DLL for CS0. This is the fraction of a clock cycle represented by the shift to be applied to the write DQS in units of 256ths. In other words, the full-cycle tap value from the master DLL is scaled by this number over 256 to get the delay value for the slave delay line.

7.7.2.11 Data Macro 0/1/2/3 DQS Gate Slave Ratio Register (DATA0/1/2/3_REG_PHY_FIFO_WE_SLAVE_RATIO_0)

The DATA0/1/2/3_REG_PHY_FIFO_WE_SLAVE_RATIO_0 is shown in Figure 7-40 and described in Table 7-74.

Figure 7-40. Data Macro 0/1/2/3 DQS Gate Slave Ratio Register (DATA0/1/2/3_REG_PHY_FIFO_WE_SLAVE_RATIO_0)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-74. Data Macro 0/1/2/3 DQS Gate Slave Ratio Register (DATA0/1/2/3_REG_PHY_FIFO_WE_SLAVE_RATIO_0) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reserved

Table 7-74. Data Macro 0/1/2/3 DQS Gate Slave Ratio Register (DATA0/1/2/3_REG_PHY_FIFO_WE_SLAVE_RATIO_0) Field Descriptions (continued)

Bit	Field	Value	Description
19-10	RD_DQS_GATE _SLAVE_RATIO_CS1	0-3FFh	Ratio value for Read DQS Gate slave DLL for CS1. This is the fraction of a clock cycle represented by the shift to be applied to the Read DQS Gate in units of 256ths. In other words, the full-cycle tap value from the master DLL is scaled by this number over 256 to get the delay value for the slave delay line.
9-0	RD_DQS_GATE _SLAVE_RATIO_CS0	0-3FFh	Ratio value for Read DQS Gate slave DLL for CS0. This is the fraction of a clock cycle represented by the shift to be applied to the Read DQS Gate in units of 256ths. In other words, the full-cycle tap value from the master DLL is scaled by this number over 256 to get the delay value for the slave delay line.

7.7.2.12 Data Macro 0/1/2/3 Write Data Slave Ratio Register (DATA0/1/2/3_REG_PHY_WR_DATA_SLAVE_RATIO_0)

The DATA0/1/2/3_REG_PHY_WR_DATA_SLAVE_RATIO_0 is shown in Figure 7-41 and described in Table 7-75.

Figure 7-41. Data Macro 0/1/2/3 Write Data Slave Ratio Register (DATA0/1/2/3_REG_PHY_WR_DATA_SLAVE_RATIO_0)

31	20	19	16
Reserved		WR_DATA_SLAVE_RATIO_CS1	
R-0		R/W-0	

15	10	9	0
WR_DATA_SLAVE_RATIO_CS1		WR_DATA_SLAVE_RATIO_CS0	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

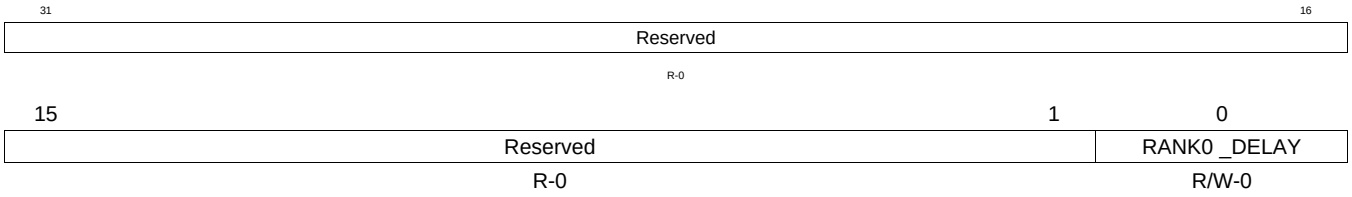
Table 7-75. Data Macro 0/1/2/3 Write Data Slave Ratio Register (DATA0/1/2/3_REG_PHY_WR_DATA_SLAVE_RATIO_0) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reserved
19-10	WR_DATA _SLAVE_RATIO_CS1	0-3FFh	Ratio value for write data slave DLL for CS1. This is the fraction of a clock cycle represented by the shift to be applied to the write DQ multiplexers in units of 256ths. In other words, the full-cycle tap value from the master DLL is scaled by this number over 256 to get the delay value for the slave delay line.
9-0	WR_DATA_SLAVE_RATIO_CS0	0-3FFh	Ratio value for write data slave DLL for CS0. This is the fraction of a clock cycle represented by the shift to be applied to the write DQ multiplexers in units of 256ths. In other words, the full-cycle tap value from the master DLL is scaled by this number over 256 to get the delay value for the slave delay line.

7.7.2.13 Data Macro 0/1/2/3 Delay Selection Register (DATA0/1/2/3_REG_PHY_USE_RANK0_DELAYS)

The DATA0/1/2/3_REG_PHY_USE_RANK0_DELAYS is shown in Figure 7-42 and described in Table 7-76.

Figure 7-42. Data Macro 0/1/2/3 Delay Selection Register (DATA0/1/2/3_REG_PHY_USE_RANK0_DELAYS)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-76. Data Macro 0/1/2/3 Delay Selection Register (DATA0/1/2/3_REG_PHY_USE_RANK0_DELAYS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	RANK0_DELAY		Delay Selection
		0	Rank 0 delays are used for all ranks during read/write.
		1	Each Rank uses its own delay. (Recommended)

7.7.2.14 DDR VTP Control Register (DDR_VTP_CTRL_0)

The DDR_VTP_CTRL_0 is shown in Figure 7-43 and described in Table 7-77.

Figure 7-43. DDR VTP Control Register (DDR_VTP_CTRL_0)

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		WR_LEVEL	LOCK	VTP_CTRL		
R-0		R/W-1h	R-0	R/W-6h		

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-77. DDR VTP Control Register (DDR_VTP_CTRL_0) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	WR_LEVEL		Selects Leveling or Slave ratio
		0	Slave Ratio. Supported for DDR2 or DDR3.
		1	Leveling Ratio. Auto leveling supported only for DDR3 operation.
3	LOCK		Freeze Dynamic Update
		0	Normal operation Dynamic update of driver impedance based on PVT variation.
		1	Freeze dynamic update of driver impedance based on PVT variation.
2-0	VTP_CTRL	0	Off
		1h	Update on 5 consecutive update requests
		2h	Update on 3 consecutive update requests
		3h	Update on 7 consecutive update requests
		4h	Update on 2 consecutive update requests
		5h	Update on 6 consecutive update requests
		6h	Update on 4 consecutive update requests
		7h	Update on 5 consecutive update requests

General-Purpose I/O (GPIO) Interface

This chapter describes the general-purpose I/O (GPIO) interface.

Topic **Page**

8.1 Introduction

8.1.1 Purpose of the Peripheral

The general-purpose interface combines two general-purpose input/output (GPIO) banks. Each GPIO module provides 32 dedicated general-purpose pins with input and output capabilities; thus, the general-purpose interface supports up to 64 (2 × 32) pins. These pins can be configured for the following applications:

- Data input (capture)/output (drive)
- Keyboard interface with a debounce cell
- Interrupt generation in active mode upon the detection of external events. Detected events are processed by two parallel independent interrupt-generation submodules to support biprocessor operations.

8.1.2 Features

Each channel in the GPIO module has the following features:

- The output enable register (GPIO_OE) controls the output capability for each pin.
- The output line level reflects the value written in the data output register (GPIO_DATAOUT) through the peripheral bus.
- The input line can be fed in to the GPIO module through an optional and configurable debouncing cell.
- The input line value is sampled into the data input register (GPIO_DATAIN) and can be read from the peripheral bus.
- In Active mode, the input line can be used through level and edge detectors to trigger synchronous interrupts. The edge (rising, falling, or both) or the level (logical 0, logical 1, or both) to be used can be configured.

The global features of the GPIO module are:

- Two identical interrupt generation sub-modules process synchronous interrupt requests from each channel in order to be used independently in a bi-processor environment. Each sub-module controls its own synchronous interrupt request line and has its own interrupt enable and interrupt status registers. The interrupt enable register (GPIO_IRQENABLE_SET_x) selects the channel(s) considered for the interrupt request generation, and the interrupt status register (GPIO_IRQSTATUS_RAW_x) determines which channel(s) has/have activated the interrupt request. Event detection on GPIO channels is reflected into the interrupt status registers independently from the interrupt enable registers content.

The module provides an alternative to the atomic ‘Test and Set’ operations for the data output and interrupt enable registers. For these registers, the module implements the “Set and Clear protocol register update”.

8.1.3 Block Diagram

Figure 8-1 details the GPIO block diagram with its configuration registers and its main functional paths:

- The synchronous path (for Active mode operations), Figure 8-2, used to generate a synchronous interrupt request upon expected event detection on any input GPIO; the synchronous interrupt request lines 0 and 1 are active according to their respective interrupt enable 0 and 1 registers.
- The blocks handling the internal clock (clock gating) and managing the Sleep mode request/acknowledge protocol (enabling the Synchronous path in Active mode).

Figure 8-1. GPIO Block Diagram

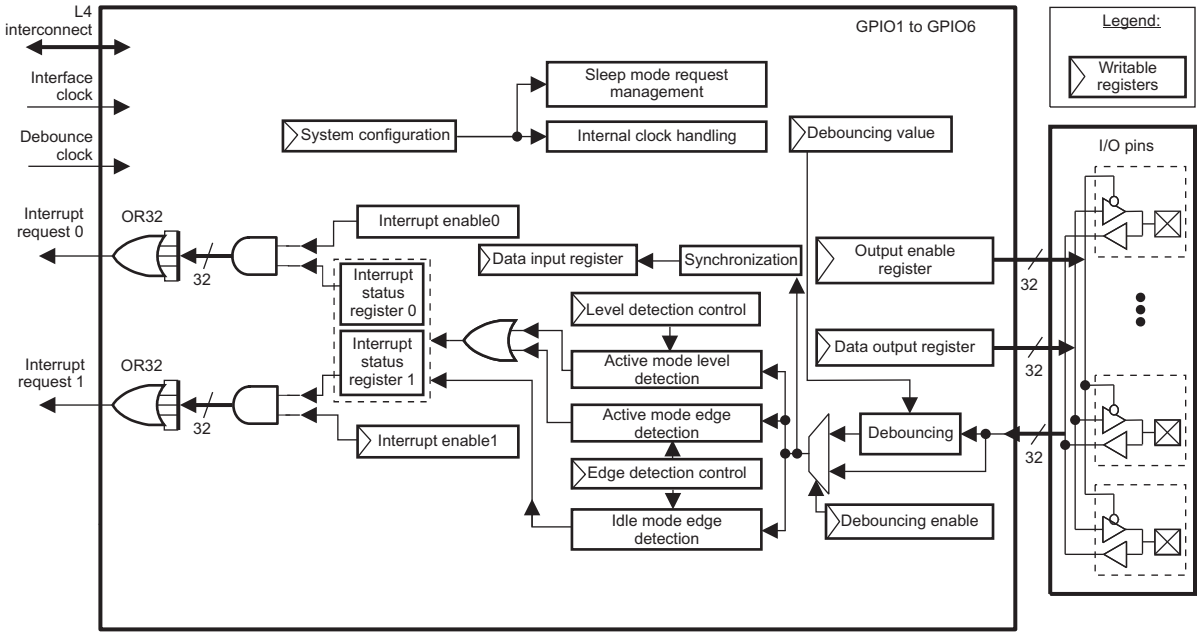
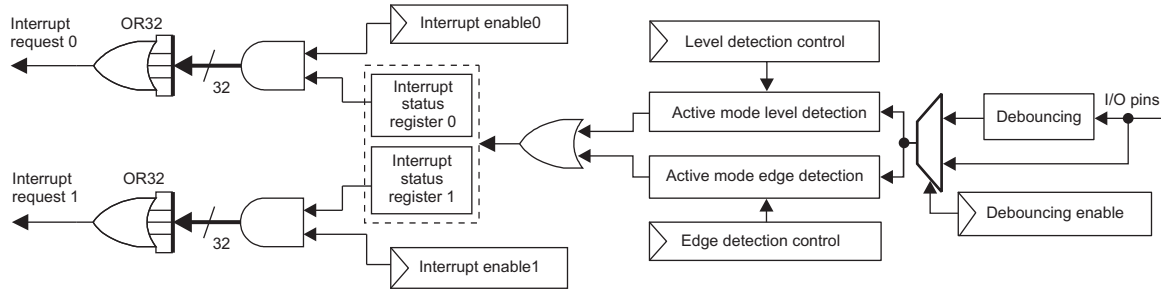


Figure 8-2. Synchronous Path Block Diagram



8.2 Architecture

This section discusses the operational details and basic functions of the GPIO peripheral.

8.2.1 Operating Modes

Four operating modes are defined for the module:

- **Active mode:** the module is running synchronously on the interface clock, interrupt can be generated according to the configuration and the external signals.
- **Idle mode:** the module is in a waiting state, interface clock can be stopped , no interrupt can be generated. Check the chip top-level functional specification for the availability of the debouncing clock while in Idle mode.
- **Inactive mode:** the module has no activity, interface clock can be stopped, no interrupt can be generated.
- **Disabled mode:** the module is not used, internal clock paths are gated, no interrupt request can be generated.

The Idle and Inactive modes are configured within the module and activated on request by the host processor through system interface sideband signals. The Disabled mode is set by software through a dedicated configuration bit. It unconditionally gates the internal clock paths not use for the system interface. All module registers are 8, 16 or 32-bit accessible through the OCP compatible interface (little endian encoding). In active mode, the event detection (level or transition) is performed in the GPIO module using the interface clock. The detection's precision is set by the frequency of this clock and the selected internal gating scheme.

8.2.2 Clocking and Reset Strategy

8.2.2.1 Clocks

GPIO module runs using two clocks:

- The debouncing clock is used for the debouncing sub-module logic (without the corresponding configuration registers). This module can sample the input line and filters the input level using a programmed delay.
- The interface clock provided by the peripheral bus (OCP compatible system interface). It is used through the entire GPIO module (except within the debouncing sub-module logic). It clocks the OCP interface and the internal logic. Clock gating features allow adapting the module power consumption to the activity.

8.2.2.2 Clocks, Gating and Active Edge Definitions

The interface clock provided by the peripheral bus (OCP compatible system interface) is used through the entire GPIO module. Two clock domains are defined: the OCP interface and the internal logic. Each clock domain can be controlled independently. Sampling operations for the data capture and for the events detection are done using the rising edge. The data loaded in the data output register (GPIO_DATAOUT) is set at the output GPIO pins synchronously with the rising edge of the interface clock.

Five clock gating features are available:

- Clock for the system interface logic can be gated when the module is not accessed, if the AUTOIDLE configuration bit in the system configuration register (GPIO_SYSCONFIG) is set. Otherwise, this logic is free running on the interface clock.
- Clock for the input data sample logic can be gated when the data in register is not accessed.
- Four clock groups are used for the logic in the synchronous events detection. Each 8 input GPIO_V2 pins group will have a separate enable signal depending on the edge/level detection register setting. If a group requires no detection, then the corresponding clock will be gated. All channels are also gated using a 'one out of N' scheme. N can take the values 1, 2, 4 or 8. The interface clock is enabled for this logic one cycle every N cycles. When N is equal to 1, there is no gating and this logic is free running on the interface clock. When N is between 2 to 8, this logic is running at the equivalent frequency of interface clock frequency divided by N.

- In Inactive mode, all internal clock paths are gated.
- In Disabled mode, all internal clock paths not used for the system interface are gated. All GPIO registers are accessible synchronously with the interface clock.

8.2.2.3 Sleep Mode Request and Acknowledge

Upon a Sleep mode request issued by the host processor, the GPIO module goes to the Idle mode according to the IDLEMODE field in the system configuration register (GPIO_SYSCONFIG).

- IDLEMODE = 0 (Force-Idle mode): the GPIO goes in Inactive mode independently of the internal module state and the Idle acknowledge is unconditionally sent. In Force-Idle mode, the module is in Inactive mode.
- IDLEMODE = 1h (No-Idle mode): the GPIO does not go to the Idle mode and the Idle acknowledge is never sent.
- IDLEMODE = 2h (Smart-Idle mode) or IDLEMODE = 3h (Smart-Idle mode): the GPIO module evaluates its internal capability to have the interface clock switched off. Once there is no more internal activity (the data input register completed to capture the input GPIO pins, there is no pending interrupt, all interrupt status bits are cleared, and there is no write access to GPIO_DEBOUNCINGTIME register pending to be synchronized), the Idle acknowledge is asserted and the GPIO enters Idle mode. When the system is awake, the Idle Request goes inactive, the Idle acknowledge signals are immediately de-asserted.

NOTE: Idle mode request and Idle acknowledge are system interface sideband signals. Once the GPIO acknowledges the Sleep mode request (Idle acknowledge has been sent), the interface clock can be stopped anytime.

Upon a Sleep mode request issued by the host processor, the GPIO module goes to the Idle mode only if there is no active bit in GPIO_IRQSTATUS_RAW_n registers.

8.2.2.4 Reset

The OCP hardware Reset signal has a global reset action on the GPIO. All configuration registers, all DFFs clocked with the Interface clock or Debouncing clock and all internal state machines are reset when the OCP hardware Reset is active (low level). The RESETDONE bit in the system status register (GPIO_SYSSTATUS) monitors the internal reset status: it is set when the Reset is completed on both OCP and Debouncing clock domains. The software Reset (SOFTRESET bit in the system configuration register) has the same effect as the OCP hardware Reset signal, and the RESETDONE bit in GPIO_SYSSTATUS is updated in the same condition.

8.2.3 Interrupt Features

8.2.3.1 Functional Description

In order to generate an interrupt request to a host processor upon a defined event (level or logic transition) occurring on a GPIO pin, the GPIO configuration registers have to be programmed as follows:

- Interrupts for the GPIO channel must be enabled in the GPIO_IRQENABLE_SET_0 and/or GPIO_IRQENABLE_SET_1 registers.
- The expected event(s) on input GPIO to trigger the interrupt request has to be selected in the GPIO_LEVELDETECT0, GPIO_LEVELDETECT1, GPIO_RISINGDETECT, and GPIO_FALLINGDETECT registers.

For instance, interrupt generation on both edges on input k is configured by setting to 1 the *k*th bit in registers GPIO_RISINGDETECT and GPIO_FALLINGDETECT along with the interrupt enabling for one or both interrupt lines (GPIO_IRQENABLE_SET_n).

NOTE: All interrupt sources (the 32 input GPIO channels) are merged together to issue two synchronous interrupt requests 0 and 1.

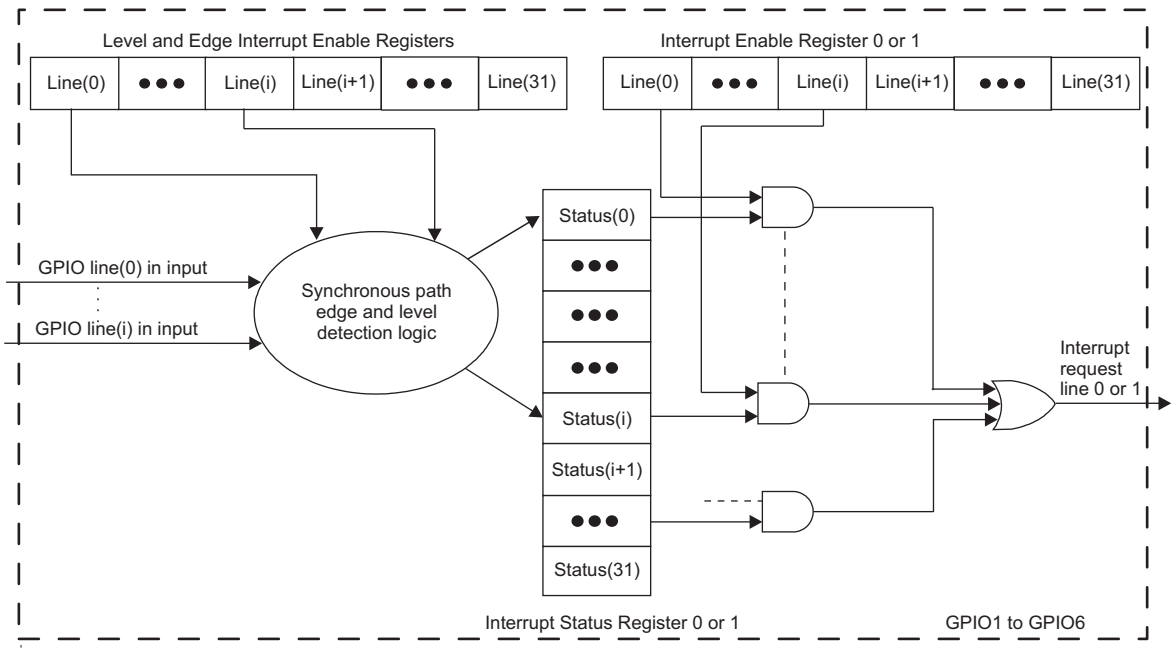
8.2.3.2 Synchronous Path: Interrupt Request Generation

In Active mode, once the GPIO configuration registers have been set to enable the interrupt generation, a synchronous path (Figure 8-3) samples the transitions and levels on the input GPIO with the internally gated interface clock. When an event matches the programmed settings, the corresponding bit in the GPIO_IRQSTATUS_RAW_n registers is set to 1 and, on the following interface clock cycle, the interrupt lines 0 and/or 1 are activated (depending on the GPIO_IRQENABLE_SET_n registers).

Due to the sampling operation, the minimum pulse width on the input GPIO to trigger a synchronous interrupt request is two times the internally gated interface clock period (the internally gated interface clock period is equal to N times the interface clock period). This minimum pulse width has to be met before and after any expected level transition detection. Level detection requires the selected level to be stable for at least two times the internally gated interface clock period to trigger a synchronous interrupt.

As the module is synchronous, latency is minimal between the expected event occurrence and the activation of the interrupt line(s). This should not exceed 3 internally gated interface clock cycles + 2 interface clock cycles when the debouncing feature is not used. When the debouncing feature is active, the latency depends on the GPIO_DEBOUNCINGTIME register value and should be less than 3 internally gated interface clock cycles + 2 interface clock cycles + GPIO_DEBOUNCINGTIME value debouncing clock cycles + 3 debouncing clock cycles.

Figure 8-3. Interrupt Request Generation



8.2.3.3 Interrupt Line Release

When the host processor receives an interrupt request issued by the GPIO module, it can read the corresponding GPIO_IRQSTATUS_n register to find out which input GPIO has triggered the interrupt. After servicing the interrupt, the processor resets the status bit and releases the interrupt line by writing a 1 in the corresponding bit of the GPIO_IRQSTATUS_n register. If there is still a pending interrupt request to serve (all bits in the GPIO_IRQSTATUS_RAW_n register not masked by the GPIO_IRQENABLE_SET_n, which are not cleared by setting the GPIO_IRQENABLE_CLR_n), the interrupt line will be re-asserted.

8.2.4 General-Purpose Interface Basic Programming Model

8.2.4.1 Power Saving by Grouping the Edge/Level Detection

Each GPIO module implements four gated clocks used by the edge/level detection logic to save power. Each group of eight input GPIO pins generates a separate enable signal depending on the edge/level detection register setting (because the input is 32 bits, four groups of eight inputs are defined for each GPIO module). If a group requires no edge/level detection, then the corresponding clock is gated (cut off). Grouping the edge/level enable can save the power consumption of the module as described in the following example.

If any of the registers:

- GPIO_LEVELDETECT0
- GPIO_LEVELDETECT1
- GPIO_RISINGDETECT
- GPIO_FALLINGDETECT

are set to 0101 0101h, then all clocks are active (power consumption is high);

are set to 0000 00FFh, then a single clock is active.

NOTE: When the clocks are enabled by writing to the GPIO_LEVELDETECT0, GPIO_LEVELDETECT1, GPIO_RISINGDETECT, and GPIO_FALLINGDETECT registers, the detection starts after 5 clock cycles. This period is required to clean the synchronization edge/level detection pipeline.

The mechanism is independent of each clock group. If the clock has been started before a new setting is performed, the following is recommended: first, set the new detection required; second, disable the previous setting (if necessary). In this way, the corresponding clock is not gated and the detection starts immediately.

8.2.4.2 Set and Clear Instructions

The GPIO module implements the set-and-clear protocol register update for the data output and interrupt enable registers. This protocol is an alternative to the atomic test and set operations and consists of writing operations at dedicated addresses (one address for setting bit[s] and one address for clearing bit[s]). The data to write is 1 at bit position(s) to clear (or to set) and 0 at unaffected bit(s).

Registers can be accessed in two ways:

- Standard: Full register read and write operations at the primary register address
- Set and clear (recommended): Separate addresses are provided to set (and clear) bits in registers. Writing 1 at these addresses sets (or clears) the corresponding bit into the equivalent register; writing a 0 has no effect.

Therefore, for these registers, three addresses are defined for one unique physical register. Reading these addresses has the same effect and returns the register value.

8.2.4.2.1 Clear Instruction

8.2.4.2.1.1 Clear Interrupt Enable Registers (GPIO_IRQENABLE_CLR_0 and GPIO_IRQENABLE_CLR_1):

- A write operation in the clear interrupt enable0 (or enable1) register clears the corresponding bit in the interrupt enable0 (or enable1) register when the written bit is 1; a written bit at 0 has no effect.
- A read of the clear interrupt enable0 (or enable1) register returns the value of the interrupt enable0 (or enable1) register.

8.2.4.2.1.2 Clear Data Output Register (GPIO_CLEARDATAOUT):

- A write operation in the clear data output register clears the corresponding bit in the data output register when the written bit is 1; a written bit at 0 has no effect.
- A read of the clear data output register returns the value of the data output register.

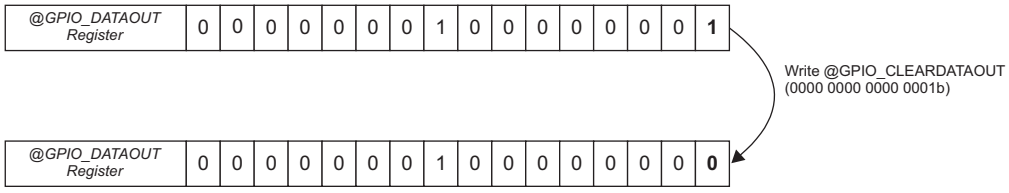
8.2.4.2.1.3 Clear Instruction Example

Assume the data output register (or one of the interrupt enable registers) contains the binary value, 0000 0001 0000 0001h, and you want to clear bit 0.

With the clear instruction feature, write 0000 0000 0000 0001h at the address of the clear data output register (or at the address of the clear interrupt enable register). After this write operation, a reading of the data output register (or the interrupt enable register) returns 0000 0001 0000 0000h; bit 0 is cleared.

NOTE: Although the general-purpose interface registers are 32-bits wide, only the 16 least-significant bits are represented in this example.

Figure 8-4. Write @ GPIO_CLEARDATAOUT Register Example



8.2.4.2.2 Set Instruction

8.2.4.2.2.1 Set Interrupt Enable Registers (GPIO_IRQENABLE_SET_0 and GPIO_IRQENABLE_SET_1):

- A write operation in the set interrupt enable0 (or enable1) register sets the corresponding bit in the interrupt enable0 (or enable1) register when the written bit is 1; a written bit at 0 has no effect.
- A read of the set interrupt enable0(or enable1) register returns the value of the interrupt enable0 (or enable1) register.

8.2.4.2.2.2 Set Data Output Register (GPIO_SETDATAOUT):

- A write operation in the set data output register sets the corresponding bit in the data output register when the written bit is 1; a written bit at 0 has no effect.
- A read of the set data output register returns the value of the data output register.

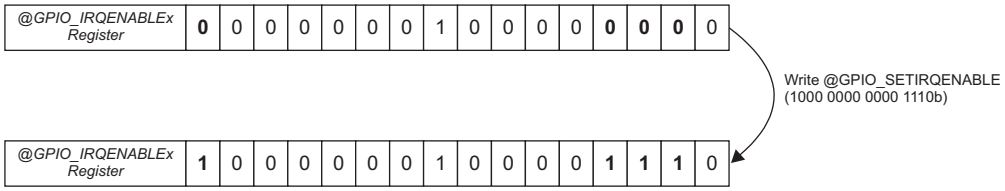
8.2.4.2.2.3 Set Instruction Example

Assume the interrupt enable0 (or enable1) register (or the data output register) contains the binary value, 0000 0001 0000 0000h, and you want to set bits 15, 3, 2, and 1.

With the set instruction feature, write 1000 0000 0000 1110h at the address of the set interrupt enable0 (or enable1) register (or at the address of the set data output register). After this write operation, a reading of the interrupt enable0 (or enable1) register (or the data output register) returns 1000 0001 0000 1110h; bits 15, 3, 2, and 1 are set.

NOTE: Although the general-purpose interface registers are 32-bits wide, only the 16 least-significant bits are represented in this example.

Figure 8-5. Write @ GPIO_SETIRQENABLEx Register Example



8.2.4.3 Data Input (Capture)/Output (Drive)

The output enable register (GPIO_OE) controls the output/input capability for each pin. At reset, all the GPIO-related pins are configured as input and output capabilities are disabled. This register is not used within the module; its only function is to carry the pads configuration.

When configured as an output (the desired bit reset in GPIO_OE), the value of the corresponding bit in the GPIO_DATAOUT register is driven on the corresponding GPIO pin. Data is written to the data output register synchronously with the interface clock. This register can be accessed with read/write operations or by using the alternate set and clear protocol register update feature. This feature lets you set or clear specific bits of this register with a single write access to the set data output register (GPIO_SETDATAOUT) or to the clear data output register (GPIO_CLEARDATAOUT) address. If the application uses a pin as an output and does not want interrupt generation from this pin, the application must properly configure the interrupt enable registers.

When configured as an input (the desired bit set to 1 in GPIO_OE), the state of the input can be read from the corresponding bit in the GPIO_DATAIN register. The input data is sampled synchronously with the interface clock and then captured in the data input register synchronously with the interface clock. When the GPIO pin levels change, they are captured into this register after two interface clock cycles (the required cycles to synchronize and to write data). If the application uses a pin as an input, the application must properly configure the interrupt enable registers to the interrupt as needed.

8.2.4.4 Debouncing Time

To enable the debounce feature for a pin, the GPIO configuration registers must be programmed as follows:

- The GPIO pin must be configured as input in the output enable register (write 1 to the corresponding bit of the GPIO_OE register).
- The debouncing time must be set in the debouncing value register (GPIO_DEBOUNCINGTIME). The GPIO_DEBOUNCINGTIME register is used to set the debouncing time for all input lines in the GPIO module. The value is global for all the ports of one GPIO module, so up to six different debouncing values are possible. The debounce cell is running with the debounce clock (32 kHz). This register represents the number of the clock cycle(s) (one cycle is 31 microseconds long) to be used.

The following formula describes the required input stable time to be propagated to the debounced output:

Debouncing time = (DEBOUNCETIME + 1) × 31 µs

Where the DEBOUNCETIME field value in the GPIO_DEBOUNCINGTIME register is from 0 to 255.

- The debouncing feature must be enabled in the debouncing enable register (write 1 to the corresponding DEBOUNCEENABLE bit in the GPIO_DEBOUNCENABLE register).

8.2.4.5 GPIO as a Keyboard Interface

The general-purpose interface can be used as a keyboard interface (Figure 8-6). You can dedicate channels based on the keyboard matrix size. Figure 8-6 shows row channels configured as inputs with the input debounce feature enabled. The row channels are driven high with an external pull-up. Column channels are configured as outputs and drive a low level.

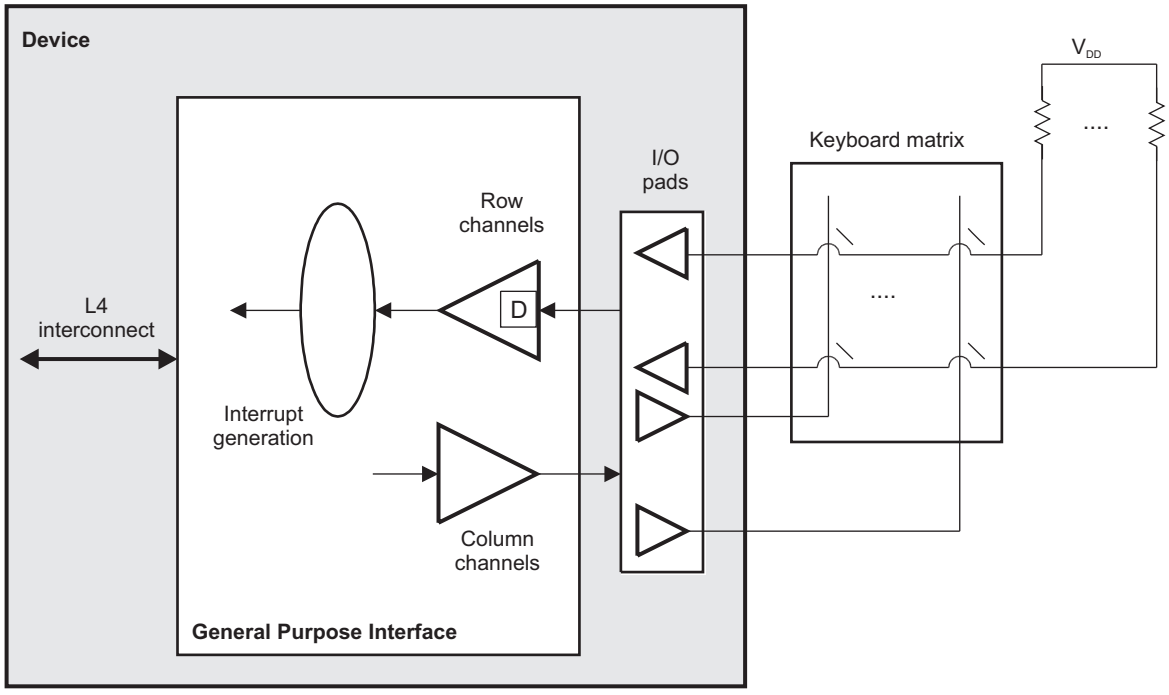
When a keyboard matrix key is pressed, the corresponding row and column lines are shorted together and a low level is driven on the corresponding row channel. This generates an interrupt based on the proper configuration (see Section 8.2.3).

When the keyboard interrupt is received, the processor can disable the keyboard interrupt and scan the column channels for the key coordinates.

- The scanning sequence has as many states as column channels: For each step in the sequence, the processor drives one column channel low and the others high.
- The processor reads the values of the row channels and thus detects which keys in the column are pressed.

At the end of the scanning sequence, the processor establishes which keys are pressed. The keyboard interface can then be reconfigured in the interrupt waiting state.

Figure 8-6. General-Purpose Interface Used as a Keyboard Interface



8.3 GPIO Registers

All module registers are 8-, 16-, or 32-bit accessible through the L4 interconnect (little-endian encoding). Access to registers is direct; no shadow registers are implemented. For the base address of these registers, see .

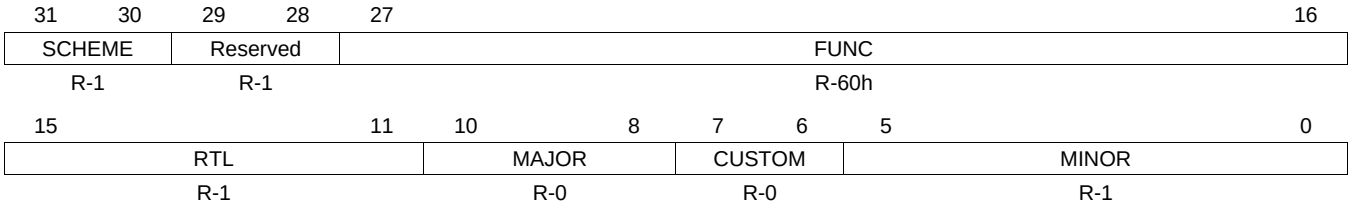
Table 8-1. GPIO Registers

Offset	Acronym	Register Name	Section
0h	GPIO_REVISION	GPIO Revision Register	Section 8.3.1
10h	GPIO_SYSCONFIG	System Configuration Register	Section 8.3.2
20h	GPIO_EOI	End of Interrupt Register	Section 8.3.3
24h	GPIO_IRQSTATUS_RAW_0	Status Raw Register for Interrupt 0	Section 8.3.4
28h	GPIO_IRQSTATUS_RAW_1	Status Raw Register for Interrupt 1	Section 8.3.5
2Ch	GPIO_IRQSTATUS_0	Status Register for Interrupt 0	Section 8.3.6
30h	GPIO_IRQSTATUS_1	Status Register for Interrupt 1	Section 8.3.7
34h	GPIO_IRQENABLE_SET_0	Enable Set Register for Interrupt 0	Section 8.3.8
38h	GPIO_IRQENABLE_SET_1	Enable Set Register for Interrupt 1	Section 8.3.9
3Ch	GPIO_IRQENABLE_CLR_0	Enable Clear Register for Interrupt 0	Section 8.3.10
40h	GPIO_IRQENABLE_CLR_1	Enable Clear Register for Interrupt 1	Section 8.3.11
114h	GPIO_SYSSTATUS	System Status Register	Section 8.3.12
130h	GPIO_CTRL	Module Control Register	Section 8.3.13
134h	GPIO_OE	Output Enable Register	Section 8.3.14
138h	GPIO_DATAIN	Data Input Register	Section 8.3.15
13Ch	GPIO_DATAOUT	Data Output Register	Section 8.3.16
140h	GPIO_LEVELDETECT0	Low-level Detection Enable Register	Section 8.3.17
144h	GPIO_LEVELDETECT1	High-level Detection Enable Register	Section 8.3.18
148h	GPIO_RISINGDETECT	Rising-edge Detection Enable Register	Section 8.3.19
14Ch	GPIO_FALLINGDETECT	Falling-edge Detection Enable Register	Section 8.3.20
150h	GPIO_DEBOUNCENABLE	Debounce Enable Register	Section 8.3.21
154h	GPIO_DEBOUNCINGTIME	Debouncing Time Register	Section 8.3.22
190h	GPIO_CLEARDATAOUT	Clear Data Output Register	Section 8.3.23
194h	GPIO_SETDATAOUT	Set Data Output Register	Section 8.3.24

8.3.1 GPIO_REVISION Register

The GPIO revision register is a read only register containing the revision number of the GPIO module. A write to this register has no effect, the same as the reset.

Figure 8-7. GPIO_REVISION Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-2. GPIO_REVISION Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	SCHEME	R	1h	Used to distinguish between old Scheme and current.
29-28	Reserved	R	1h	Reserved
27-16	FUNC	R	60h	Indicates a software compatible module family.
15-11	RTL	R	1Fh	RTL version
10-8	MAJOR	R	0h	Major Revision
7-6	CUSTOM	R	0h	Indicates a special version for a particular device.
5-0	MINOR	R	1Fh	Minor Revision

8.3.2 GPIO_SYSCONFIG Register

The GPIO_SYSCONFIG register controls the various parameters of the L4 interconnect.

NOTE: When the AUTOIDLE bit is set, the GPIO_DATAIN read command has a 3 OCP cycle latency due to the data in sample gating mechanism. When the AUTOIDLE bit is not set, the GPIO_DATAIN read command has a 2 OCP cycle latency.

Figure 8-8. GPIO_SYSCONFIG Register

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		IDLEMODE		Reserved	SOFTRESET	AUTOIDLE
R-0		R/W-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-3. GPIO_SYSCONFIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	Reserved	R	0h	Reserved
4-3	IDLEMODE	R/W	0h	Power Management, Req/Ack control 0 = Force-idle. An idle request is acknowledged unconditionally 1 = No-idle. An idle request is never acknowledged 2 = Smart-idle. Acknowledgment to an idle request is given based on the internal activity of the module 3 = Smart-idle.
2	Reserved	R/W	0h	Reserved. Always write the default value for future device compatibility.
1	SOFTRESET	R/W	0h	Software reset. This bit is automatically reset by the hardware. During reads, it always returns 0. 0 = Normal mode 1 = The module is reset
0	AUTOIDLE	R/W	0h	Internal interface clock gating strategy 0 = Internal Interface OCP clock is free-running 1 = Automatic internal OCP clock gating, based on the OCP interface activity

8.3.3 GPIO_EOI Register

The GPIO_EOI register provides software end of interrupt (EOI) control.

Figure 8-9. GPIO_EOI Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															LINE_NUMBER
R-0															R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-4. GPIO_EOI Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved
0	LINE_NUMBER	R/W	0h	Software End Of Interrupt (EOI) control. Write number of interrupt output. Write 0 = EOI for interrupt output line #0 Write 1 = EOI for interrupt output line #1 Read 0 = Reads always 0 (no EOI memory)

8.3.4 GPIO_IRQSTATUS_RAW_0 Register

The GPIO_IRQSTATUS_RAW_0 register provides core status information for the interrupt handling, showing all active events (enabled and not enabled). The fields are read-write. Writing a 1 to a bit sets it to 1, that is, triggers the IRQ (mostly for debug). Writing a 0 has no effect, that is, the register value is not be modified. Only enabled, active events trigger an actual interrupt request on the IRQ output line.

Figure 8-10. GPIO_IRQSTATUS_RAW_0 Register

31	0
INTLINE[n]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 8-5. GPIO_IRQSTATUS_RAW_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INTLINE[*n*]	R/W	0h	Interrupt n status. 0 = No effect 1 = IRQ is triggered.

8.3.5 GPIO_IRQSTATUS_RAW_1 Register

The GPIO_IRQSTATUS_RAW_1 register provides core status information for the interrupt handling, showing all active events (enabled and not enabled). The fields are read-write. Writing a 1 to a bit sets it to 1, that is, triggers the IRQ (mostly for debug). Writing a 0 has no effect, that is, the register value is not be modified. Only enabled, active events trigger an actual interrupt request on the IRQ output line.

Figure 8-11. GPIO_IRQSTATUS_RAW_1 Register

31	0
INTLINE[*n*]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

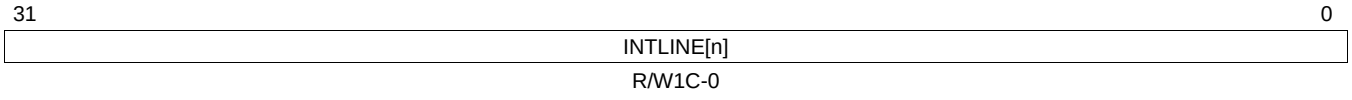
Table 8-6. GPIO_IRQSTATUS_RAW_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INTLINE[*n*]	R/W	0h	Interrupt n status. 0 = No effect 1 = IRQ is triggered.

8.3.6 GPIO_IRQSTATUS_0 Register

The GPIO_IRQSTATUS_0 register provides core status information for the interrupt handling, showing all enabled events (enabled interrupts only). The fields are read-write. Writing a 1 to a bit clears the bit to 0, that is, clears the IRQ. Writing a 0 has no effect, that is, the register value is not modified. Only enabled, active events trigger an actual interrupt request on the IRQ output line.

Figure 8-12. GPIO_IRQSTATUS_0 Register



LEGEND: R/W = Read/Write; W1C = Write a 1 to clear to 0, a write of 0 has no effect; -*n* = value after reset

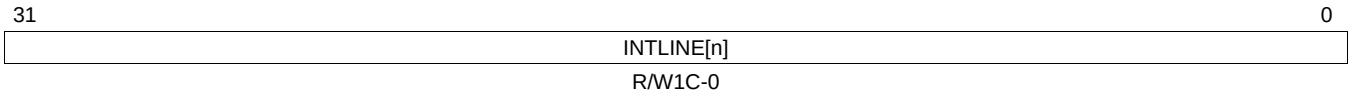
Table 8-7. GPIO_IRQSTATUS_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INTLINE[*n*]	R/W1C	0h	Interrupt n status. 0 = No effect 1 = IRQ is triggered.

8.3.7 GPIO_IRQSTATUS_1 Register

The GPIO_IRQSTATUS_1 register provides core status information for the interrupt handling, showing all enabled events (enabled interrupts only). The fields are read-write. Writing a 1 to a bit clears the bit to 0, that is, clears the IRQ. Writing a 0 has no effect, that is, the register value is not modified. Only enabled, active events trigger an actual interrupt request on the IRQ output line.

Figure 8-13. GPIO_IRQSTATUS_1 Register



LEGEND: R/W = Read/Write; W1C = Write a 1 to clear to 0, a write of 0 has no effect; -*n* = value after reset

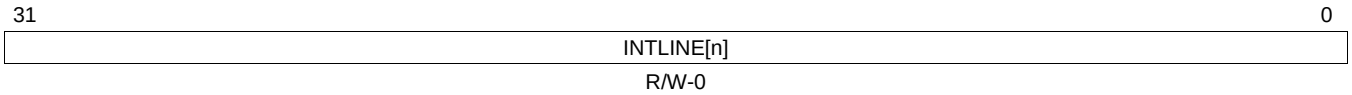
Table 8-8. GPIO_IRQSTATUS_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INTLINE[*n*]	R/W1C	0h	Interrupt n status. 0 = No effect 1 = IRQ is triggered.

8.3.8 GPIO_IRQENABLE_SET_0 Register

All 1-bit fields in the GPIO_IRQENABLE_SET_0 register enable a specific interrupt event to trigger an interrupt request. Writing a 1 to a bit enables the interrupt field. Writing a 0 has no effect, that is, the register value is not modified.

Figure 8-14. GPIO_IRQENABLE_SET_0 Register



LEGEND: R/W = Read/Write; -*n* = value after reset

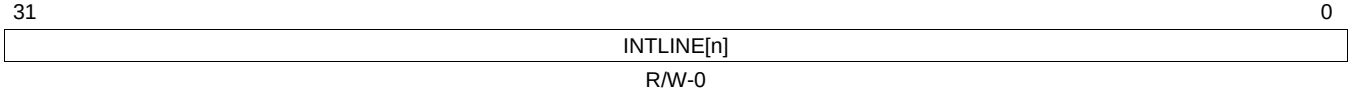
Table 8-9. GPIO_IRQENABLE_SET_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INTLINE[*n*]	R/W	0h	Interrupt n enable 0 = No effect 1 = Enable IRQ generation.

8.3.9 GPIO_IRQENABLE_SET_1 Register

All 1-bit fields in the GPIO_IRQENABLE_SET_1 register enable a specific interrupt event to trigger an interrupt request. Writing a 1 to a bit enables the interrupt field. Writing a 0 has no effect, that is, the register value is not modified.

Figure 8-15. GPIO_IRQENABLE_SET_1 Register



LEGEND: R/W = Read/Write; -*n* = value after reset

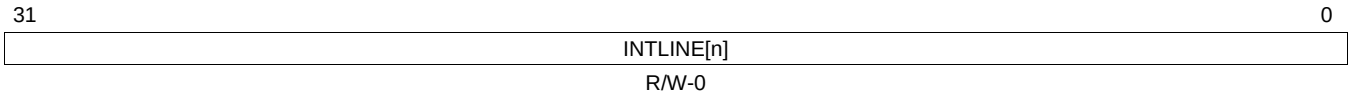
Table 8-10. GPIO_IRQENABLE_SET_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INTLINE[*n*]	R/W	0h	Interrupt n enable 0 = No effect 1 = Enable IRQ generation.

8.3.10 GPIO_IRQENABLE_CLR_0 Register

All 1-bit fields in the GPIO_IRQENABLE_CLR_0 register clear a specific interrupt event. Writing a 1 to a bit disables the interrupt field. Writing a 0 has no effect, that is, the register value is not modified.

Figure 8-16. GPIO_IRQENABLE_CLR_0 Register



LEGEND: R/W = Read/Write; -*n* = value after reset

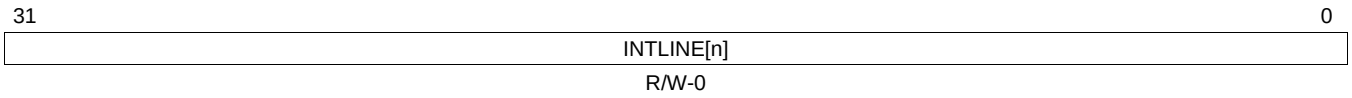
Table 8-11. GPIO_IRQENABLE_CLR_0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INTLINE[*n*]	R/W	0h	Interrupt n enable 0 = No effect 1 = Disable IRQ generation.

8.3.11 GPIO_IRQENABLE_CLR_1 Register

All 1-bit fields in the GPIO_IRQENABLE_CLR_1 register clear a specific interrupt event. Writing a 1 to a bit disables the interrupt field. Writing a 0 has no effect, that is, the register value is not modified.

Figure 8-17. GPIO_IRQENABLE_CLR_1 Register



LEGEND: R/W = Read/Write; -*n* = value after reset

Table 8-12. GPIO_IRQENABLE_CLR_1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INTLINE[*n*]	R/W	0h	Interrupt n enable 0 = No effect 1 = Disable IRQ generation.

8.3.12 GPIO_SYSSTATUS Register

The GPIO_SYSSTATUS register provides the reset status information about the GPIO module. It is a read-only register; a write to this register has no effect.

Figure 8-18. GPIO_SYSSTATUS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RESETDONE
R-0															R-0

LEGEND: R = Read only; -*n* = value after reset

Table 8-13. GPIO_SYSSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved
0	RESETDONE	R	0h	Reset status information. 0 = Internal Reset is on-going 1 = Reset completed

8.3.13 GPIO_CTRL Register

The GPIO_CTRL register controls the clock gating functionality. The DISABLEMODULE bit controls a clock gating feature at the module level. When set, this bit forces the clock gating for all internal clock paths. Module internal activity is suspended. System interface is not affected by this bit. System interface clock gating is controlled with the AUTOIDLE bit in the system configuration register (GPIO_SYSCONFIG). This bit is to be used for power saving when the module is not used because of the multiplexing configuration selected at the chip level. This bit has precedence over all other internal configuration bits.

Figure 8-19. GPIO_CTRL Register

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		GATINGRATIO		DISABLEMODULE
R-0		R/W-1		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-14. GPIO_CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	Reserved	R	0h	Reserved
2-1	GATINGRATIO	R/W	1h	Gating Ratio. Controls the clock gating for the event detection logic. 0 = Functional clock is interface clock. 1 = Functional clock is interface clock divided by 2. 2 = Functional clock is interface clock divided by 4. 3 = Functional clock is interface clock divided by 8.
0	DISABLEMODULE	R/W	0h	Module Disable 0 = Module is enabled, clocks are not gated. 1 = Module is disabled, clocks are gated.

8.3.14 GPIO_OE Register

The GPIO_OE register is used to enable the pins output capabilities. At reset, all the GPIO related pins are configured as input and output capabilities are disabled. This register is not used within the module, its only function is to carry the pads configuration. When the application is using a pin as an output and does not want interrupt generation from this pin, the application can/has to configure properly the Interrupt Enable registers.

Figure 8-20. GPIO_OE Register

31	0
OUTPUTEN[n]	
R/W-FFFF FFFFh	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 8-15. GPIO_OE Register Field Descriptions

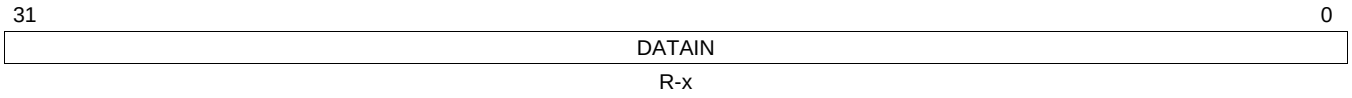
Bit	Field	Type	Reset	Description
31-0	OUTPUTEN[*n*]	R/W	FFFF FFFFh	Output Data Enable 0 = The corresponding GPIO port is configured as an output. 1 = The corresponding GPIO port is configured as an input.

8.3.15 GPIO_DATAIN Register

The GPIO_DATAIN register is used to register the data that is read from the GPIO pins. The GPIO_DATAIN register is a read-only register. The input data is sampled synchronously with the interface clock and then captured in the GPIO_DATAIN register synchronously with the interface clock. So, after changing, GPIO pin levels are captured into this register after two interface clock cycles (the required cycles to synchronize and to write the data).

NOTE: When the AUTOIDLE bit in the system configuration register (GPIO_SYSCONFIG) is set, the GPIO_DATAIN read command has a 3 OCP cycle latency due to the data in sample gating mechanism. When the AUTOIDLE bit is not set, the GPIO_DATAIN read command has a 2 OCP cycle latency.

Figure 8-21. GPIO_DATAIN Register



LEGEND: R = Read only; -*n* = value after reset

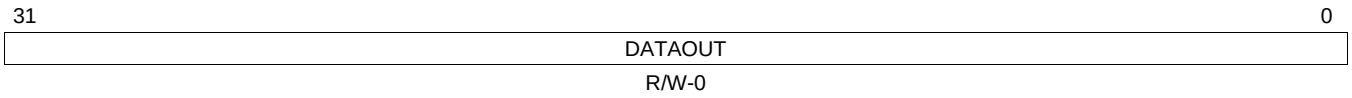
Table 8-16. GPIO_DATAIN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DATAIN	R	x	Sampled Input Data

8.3.16 GPIO_DATAOUT Register

The GPIO_DATAOUT register is used for setting the value of the GPIO output pins. Data is written to the GPIO_DATAOUT register synchronously with the interface clock. This register can be accessed with direct read/write operations or using the alternate 'Set/Clear' feature. This feature enables to set or clear specific bits of this register with a single write access to the set data output register (GPIO_SETDATAOUT) or to the clear data output register (GPIO_CLEARDATAOUT) address.

Figure 8-22. GPIO_DATAOUT Register



LEGEND: R/W = Read/Write; -*n* = value after reset

Table 8-17. GPIO_DATAOUT Register Field Descriptions

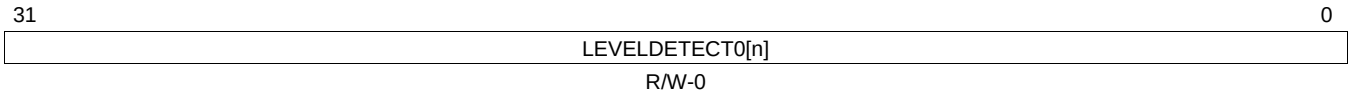
Bit	Field	Type	Reset	Description
31-0	DATAOUT	R/W	0	Data to set on output pins

8.3.17 GPIO_LEVELDETECT0 Register

The GPIO_LEVELDETECT0 register is used to enable/disable for each input lines the low-level (0) detection to be used for the interrupt request generation.

NOTE: Enabling at the same time high-level detection and low-level detection for one given pin makes a constant interrupt generator.

Figure 8-23. GPIO_LEVELDETECT0 Register



LEGEND: R/W = Read/Write; -*n* = value after reset

Table 8-18. GPIO_LEVELDETECT0 Register Field Descriptions

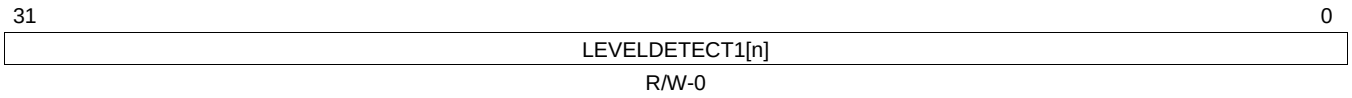
Bit	Field	Type	Reset	Description
31-0	LEVELDETECT0[*n*]	R/W	0h	Low Level Interrupt Enable 0 = Disable the IRQ assertion on low-level detect. 1 = Enable the IRQ assertion on low-level detect.

8.3.18 GPIO_LEVELDETECT1 Register

The GPIO_LEVELDETECT1 register is used to enable/disable for each input lines the high-level (1) detection to be used for the interrupt request generation.

NOTE: Enabling at the same time high-level detection and low-level detection for one given pin makes a constant interrupt generator.

Figure 8-24. GPIO_LEVELDETECT1 Register



LEGEND: R/W = Read/Write; -*n* = value after reset

Table 8-19. GPIO_LEVELDETECT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LEVELDETECT1[*n*]	R/W	0h	High Level Interrupt Enable 0 = Disable the IRQ assertion on high-level detect. 1 = Enable the IRQ assertion on high-level detect.

8.3.19 GPIO_RISINGDETECT Register

The GPIO_RISINGDETECT register is used to enable/disable for each input lines the rising-edge (transition 0 to 1) detection to be used for the interrupt request generation.

Figure 8-25. GPIO_RISINGDETECT Register

31	0
RISINGDETECT[n]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 8-20. GPIO_RISINGDETECT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RISINGDETECT[*n*]	R/W	0h	Rising Edge Interrupt Enable 0 = Disable IRQ on rising-edge detect. 1 = Enable IRQ on rising-edge detect.

8.3.20 GPIO_FALLINGDETECT Register

The GPIO_FALLINGDETECT register is used to enable/disable for each input lines the falling-edge (transition 1 to 0) detection to be used for the interrupt request generation.

Figure 8-26. GPIO_FALLINGDETECT Register

31	0
FALLINGDETECT[n]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 8-21. GPIO_FALLINGDETECT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	FALLINGDETECT[*n*]	R/W	0h	Falling Edge Interrupt Enable 0 = Disable IRQ on falling-edge detect. 1 = Enable IRQ on falling-edge detect.

8.3.21 GPIO_DEBOUNCENABLE Register

The GPIO_DEBOUNCENABLE register is used to enable/disable the debouncing feature for each input line.

Figure 8-27. GPIO_DEBOUNCENABLE Register

31	0
DEBOUNCEENABLE[n]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 8-22. GPIO_DEBOUNCENABLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	DEBOUNCEENABLE[*n*]	R/W	0h	Input Debounce Enable 0 = Disable debouncing feature on the corresponding input port. 1 = Enable debouncing feature on the corresponding input port.

8.3.22 GPIO_DEBOUNCINGTIME Register

The GPIO_DEBOUNCINGTIME register controls debouncing time (the value is global for all ports). The debouncing cell is running with the debouncing clock (32 kHz), this register represents the number of the clock cycle(s) (31µs long) to be used.

Figure 8-28. GPIO_DEBOUNCINGTIME Register

31	16
Reserved	
R-0	

15	8	7	0
Reserved		DEBOUNCETIME	
R-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 8-23. GPIO_DEBOUNCINGTIME Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	Reserved	R/W	0h	Reserved
7-0	DEBOUNCETIME	R/W	0h	Input Debouncing Value in 31 microsecond steps. Debouncing time = (DEBOUNCETIME + 1) × 31 microseconds

8.3.23 GPIO_CLEARDATAOUT Register

Writing a 1 to a bit in the GPIO_CLEARDATAOUT register clears to 0 the corresponding bit in the GPIO_DATAOUT register; writing a 0 has no effect. A read of the GPIO_CLEARDATAOUT register returns the value of the data output register (GPIO_DATAOUT).

Figure 8-29. GPIO_CLEARDATAOUT Register

31	0
INTLINE [n]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 8-24. GPIO_CLEARDATAOUT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INTLINE [*n*]	R/W	0h	Clear Data Output Register 0 = No effect 1 = Clear the corresponding bit in the GPIO_DATAOUT register.

8.3.24 GPIO_SETDATAOUT Register

Writing a 1 to a bit in the GPIO_SETDATAOUT register sets to 1 the corresponding bit in the GPIO_DATAOUT register; writing a 0 has no effect. A read of the GPIO_SETDATAOUT register returns the value of the data output register (GPIO_DATAOUT).

Figure 8-30. GPIO_SETDATAOUT Register

31	0
INTLINE [n]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 8-25. GPIO_SETDATAOUT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INTLINE [*n*]	R/W	0h	Set Data Output Register 0 = No effect 1 = Set the corresponding bit in the GPIO_DATAOUT register.

General-Purpose Memory Controller (GPMC)

This chapter describes the general-purpose memory controller (GPMC).

Topic **Page**

9.1 Introduction

9.1.1 Overview

The general-purpose memory controller (GPMC) is an unified memory controller dedicated to interfacing external memory devices:

- Asynchronous SRAM-like memories and application-specific integrated circuit (ASIC) devices
- Asynchronous, synchronous, and page mode (only available in non-multiplexed mode) burst NOR flash devices
- NAND Flash
- Pseudo-SRAM devices

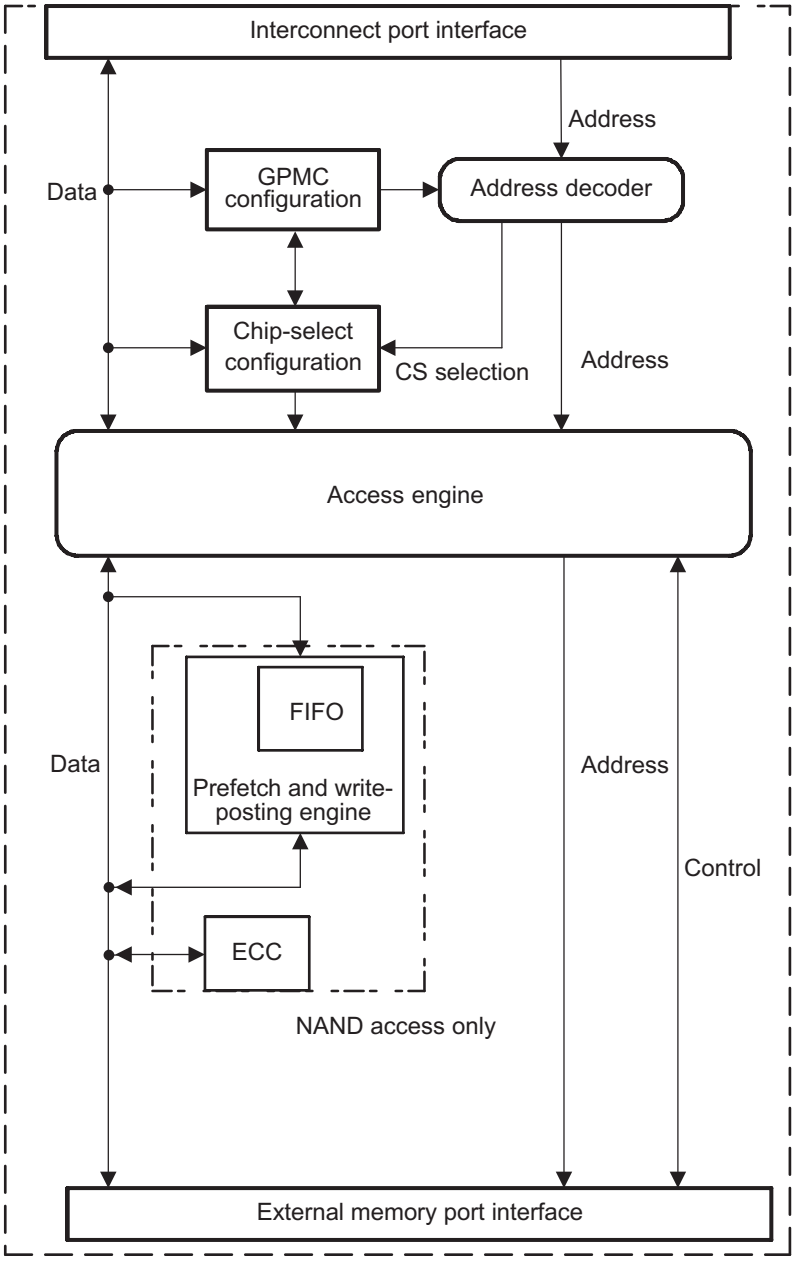
9.1.2 Block Diagram

The GPMC can access various external devices through the L3 Slow Interconnect. The flexible programming model allows a wide range of attached device types and access schemes. Based on the programmed configuration bit fields stored in the GPMC registers, the GPMC is able to generate all control signals timing depending on the attached device and access type. Given the chip-select decoding and its associated configuration registers, the GPMC selects the appropriate device type control signals timing.

Figure 9-1 shows the GPMC functional block diagram. The GPMC consists of six blocks:

- Interconnect port interface
- Address decoder, GPMC configuration, and chip-select configuration register file
- Access engine
- Prefetch and write-posting engine
- Error correction code engine (ECC)
- External device/memory port interface

Figure 9-1. GPMC Block Diagram



9.2 Architecture

9.2.1 GPMC Signals

Table 9-1 lists the GPMC subsystem I/O pins.

NOTE: In this chapter, the i in GPMC_CONFIGx_i represents GPMC chip-select i, where i = 0 to 5.

Table 9-1. GPMC I/O Description

Pin Name	I/O	Description
GPMC_A[27:0]	O	Address
GPMC_D[15:0]	I/O	Data
GPMC_CS[5:0]	O	Chip-selects (active low).
GPMC_CLK	O	Clock generated for the external memory or device
GPMC_ADV_ALE	O	Address valid (active low). Also used as address latch enable (active high) for NAND protocol memories.
GPMC_OE_RE	O	Output enable (active low). Also used as read enable (active low) for NAND protocol memories.
GPMC_WE	O	Write enable (active low)
GPMC_BE0_CLE	O	Lower-byte enable (active low). Also used as command latch enable for NAND protocol memories.
GPMC_BE1	O	Upper-byte enable (active low)
GPMC_WP	O	Write protect (active low). Not all devices pin out the Write protect signal. See the device specific data manual to determine if the Write protect pin is supported.
GPMC_WAIT[1:0]	I	External wait signals for NOR and NAND protocol memories. The wait signals can be mapped on any of the chip-selects. Not all devices pin out both External wait signals. See the device specific data manual to determine the number of External wait signals supported.
GPMC_DIR	O	GPMC_D[15:0] signals direction control: Low during transmit (for write access: data OUT from GPMC to memory) High during receive (for read access: data IN from memory to GPMC) Not all devices pin out the direction control signal. See the device specific data manual to determine if the direction control signal is supported.

Table 9-2 shows the use of address and data GPMC controller pins based on the type of external device.

Table 9-2. GPMC Pin Multiplexing Options

GPMC Pin	Non Multiplexed Address Data 16-Bit Device	Non Multiplexed Address Data 8-Bit Device	Multiplexed Address Data 16-Bit Device	16-Bit NAND Device	8-Bit NAND Device
GPMC_A[27]	A26	A27	A26	Not Used	Not Used
GPMC_A[26]	A25	A26	A25	Not Used	Not Used
GPMC_A[25]	A24	A25	A24	Not Used	Not Used
GPMC_A[24]	A23	A24	A23	Not Used	Not Used
GPMC_A[23]	A22	A23	A22	Not Used	Not Used
GPMC_A[22]	A21	A22	A21	Not Used	Not Used
GPMC_A[21]	A20	A21	A20	Not Used	Not Used
GPMC_A[20]	A19	A20	A19	Not Used	Not Used
GPMC_A[19]	A18	A19	A18	Not Used	Not Used
GPMC_A[18]	A17	A18	A17	Not Used	Not Used
GPMC_A[17]	A16	A17	A16	Not Used	Not Used
GPMC_A[16]	A15	A16	Not Used	Not Used	Not Used
GPMC_A[15]	A14	A15	Not Used	Not Used	Not Used

Table 9-2. GPMC Pin Multiplexing Options (continued)

GPMC Pin	Non Multiplexed Address Data 16-Bit Device	Non Multiplexed Address Data 8-Bit Device	Multiplexed Address Data 16-Bit Device	16-Bit NAND Device	8-Bit NAND Device
GPMC_A[14]	A13	A14	Not Used	Not Used	Not Used
GPMC_A[13]	A12	A13	Not Used	Not Used	Not Used
GPMC_A[12]	A11	A12	Not Used	Not Used	Not Used
GPMC_A[11]	A10	A11	Not Used	Not Used	Not Used
GPMC_A[10]	A9	A10	Not Used	Not Used	Not Used
GPMC_A[9]	A8	A9	Not Used	Not Used	Not Used
GPMC_A[8]	A7	A8	Not Used	Not Used	Not Used
GPMC_A[7]	A6	A7	Not Used	Not Used	Not Used
GPMC_A[6]	A5	A6	Not Used	Not Used	Not Used
GPMC_A[5]	A4	A5	Not Used	Not Used	Not Used
GPMC_A[4]	A3	A4	Not Used	Not Used	Not Used
GPMC_A[3]	A2	A3	Not Used	Not Used	Not Used
GPMC_A[2]	A1	A2	Not Used	Not Used	Not Used
GPMC_A[1]	A0	A1	Not Used	Not Used	Not Used
GPMC_A[0]	Not Used	A0	Not Used	Not Used	Not Used
GPMC_D[15]	D15	Not Used	A16/D15	D15	Not Used
GPMC_D[14]	D14	Not Used	A15/D14	D14	Not Used
GPMC_D[13]	D13	Not Used	A14/D13	D13	Not Used
GPMC_D[12]	D12	Not Used	A13/D12	D12	Not Used
GPMC_D[11]	D11	Not Used	A12/D11	D11	Not Used
GPMC_D[10]	D10	Not Used	A11/D10	D10	Not Used
GPMC_D[9]	D9	Not Used	A10/D9	D9	Not Used
GPMC_D[8]	D8	Not Used	A9/D8	D8	Not Used
GPMC_D[7]	D7	D7	A8/D7	D7	D7
GPMC_D[6]	D6	D6	A7/D6	D6	D6
GPMC_D[5]	D5	D5	A6/D5	D5	D5
GPMC_D[4]	D4	D4	A5/D4	D4	D4
GPMC_D[3]	D3	D3	A4/D3	D3	D3
GPMC_D[2]	D2	D2	A3/D2	D2	D2
GPMC_D[1]	D1	D1	A2/D1	D1	D1
GPMC_D[0]	D0	D0	A1/D0	D0	D0

With all device types, the GPMC does not drive unnecessary address lines. They stay at their reset value of 00.

Address mapping supports address/data-multiplexed 16-bit wide devices:

- The NOR flash memory controller still supports non-multiplexed address and data memory devices.
- Multiplexing mode can be selected through the GPMC_CONFIG1_i MUXADDDATA field (i = 0 to 5).
- Asynchronous page mode is not supported for multiplexed address and data devices.

9.2.2 GPMC Modes

This section shows three GPMC external connections options:

- Figure 9-2 shows a connection between the GPMC and a 16-bit synchronous address/data-multiplexed (or AAD-multiplexed, but this protocol use less address pins) external memory device.
- Figure 9-3 shows a connection between the GPMC and a 16-bit synchronous nonmultiplexed external memory device .
- Figure 9-4 shows a connection between the GPMC and a 8-bit NAND device

NOTE: If ROM boot from a NOR device is required, some high-order address lines must be driven low by external hardware during ROM operation. See Section 25.7.2 for details.

Figure 9-2. GPMC to 16-Bit Address/Data-Multiplexed Memory

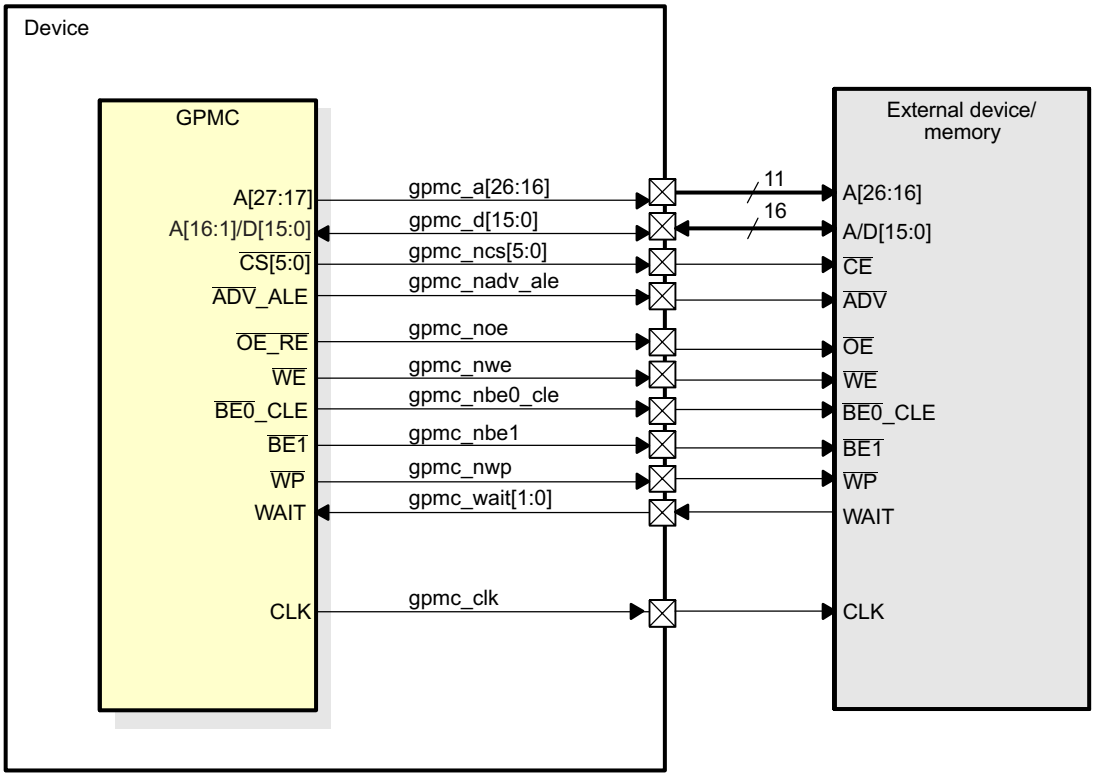


Figure 9-3. GPMC to 16-Bit Nonmultiplexed Memory

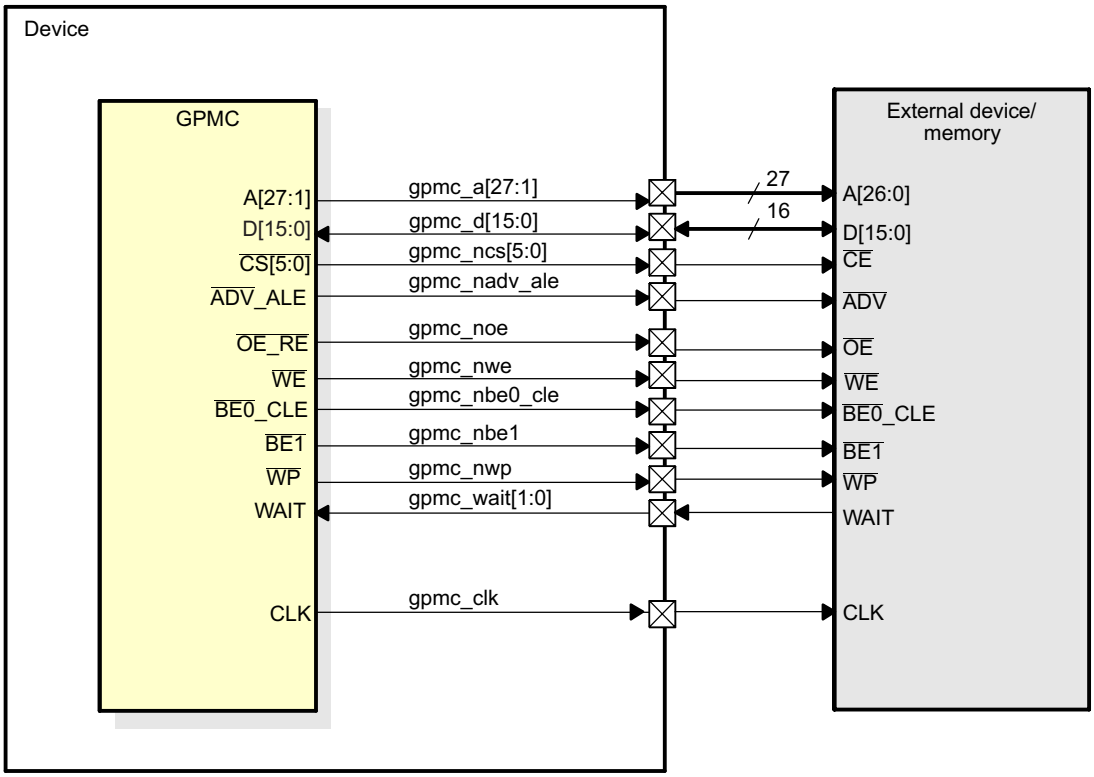
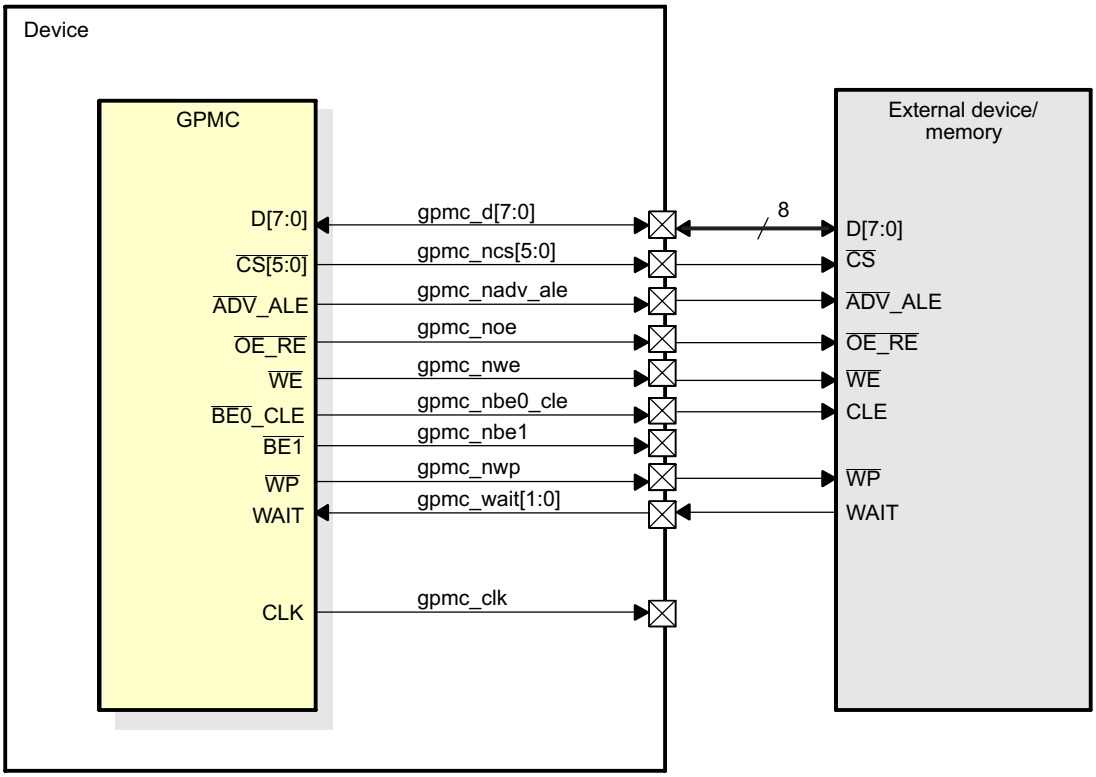


Figure 9-4. GPMC to 8-Bit NAND Device



9.2.3 GPMC Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

- No STANDBY hardware handshake
- No wake-up request
- One system direct memory access (eDMA) request
- One interrupt request to the Cortex-A8 MPU Interrupt Controller (MA_IRQ)
- One clock for functional and interface domains

Figure 9-5 shows GPMC integration. Table 9-3 through Table 9-5 summarize the integration of the module in the device.

Figure 9-5. GPMC Integration

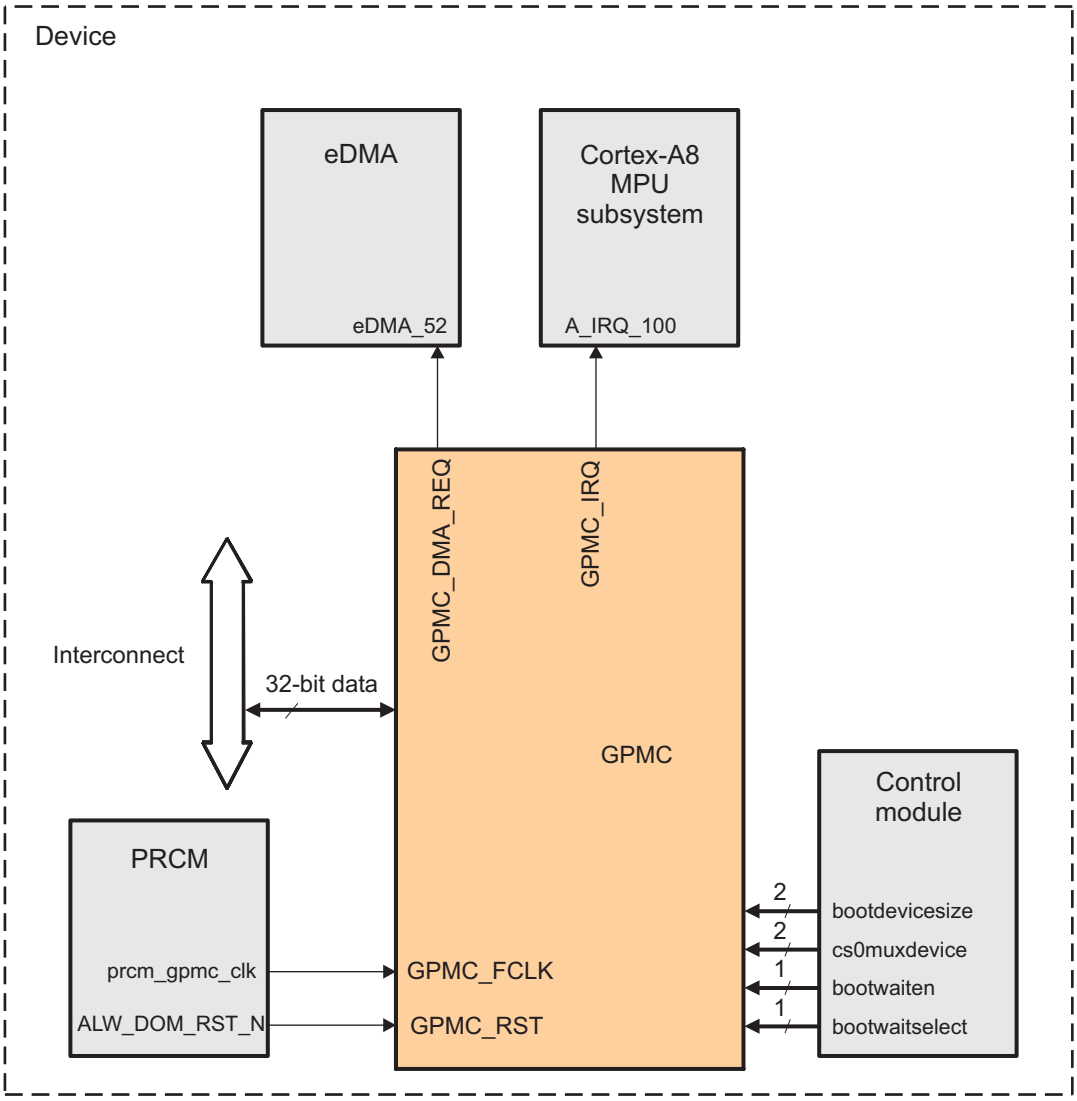


Table 9-3. GPMC Integration Attributes

Module Instance	Attribute		
	Power Domain	Wake-up Capability	Interconnect
GPMC	ALWAYS_ON	No	L3_Slow

Table 9-4. GPMC Clocks and Resets

Module Instance	Destination Signal Name	Source Signal Name	Source	Description
Clocks				
GPMC	GPMC_FCLK	prcm_GPMC_CLK	PRCM	Functional clock
Resets				
GPMC	GPMC_RST	ALW_DOM_RST	PRCM	GPMC reset

Table 9-5. GPMC Hardware Requests

Module Instance	Source Signal Name	Destination Signal Name	Destination	Description
Interrupt Requests				
GPMC	GPMC_IRQ	A_IRQ_100	Cortex-A8	GPMC interrupt to Cortex-A8 MPU subsystem
DMA Requests				
GPMC	GPMC_DMA_REQ	e_DMA_52	eDMA	GPMC request from Prefetch Engine to eDMA

9.2.4 GPMC Functional Description

The GPMC basic programming model offers maximum flexibility to support various access protocols for each of the eight configurable chip-selects. Use optimal chip-select settings, based on the characteristics of the external device:

- Different protocols can be selected to support generic asynchronous or synchronous random-access devices (NOR flash, SRAM) or to support specific NAND devices.
- The address and the data bus can be multiplexed on the same external bus.
- Read and write access can be independently defined as asynchronous or synchronous.
- System requests (byte, 16-bit word, burst) are performed through single or multiple accesses. External access profiles (single, multiple with optimized burst length, native- or emulated-wrap) are based on external device characteristics (supported protocol, bus width, data buffer size, native-wrap support).
- System burst read or write requests are synchronous-burst (multiple-read or multiple-write). When neither burst nor page mode is supported by external memory or ASIC devices, system burst read or write requests are translated to successive single synchronous or asynchronous accesses (single reads or single writes). 8-bit wide devices are supported only in single synchronous or single asynchronous read or write mode.
- To simulate a programmable internal-wait state, an external wait pin can be monitored to dynamically control external access at the beginning (initial access time) of and during a burst access.

Each control signal is controlled independently for each chip-select. The internal functional clock of the GPMC (GPMC_FCLK) is used as a time reference to specify the following:

- Read- and write-access duration
- Most GPMC external interface control-signal assertion and deassertion times
- Data-capture time during read access
- External wait-pin monitoring time
- Duration of idle time between accesses, when required

9.2.4.1 GPMC Clock Configuration

Table 9-6 describes the GPMC clocks.

Table 9-6. GPMC Clocks

Signal	I/O	Description
GPMC_FCLK	I	Functional and interface clock
GPMC_CLK	O	External clock provided to synchronous external memory devices.

The GPMC_CLK is generated by the GPMC from the internal GPMC_FCLK clock. The source of the GPMC_FCLK is described in Table 9-4. The GPMC_CLK is configured via the GPMC_CONFIG1_i GPMCFCLKDIVIDER field (for i = 0 to 3) as shown in Table 9-7.

Table 9-7. GPMC_CONFIG1_i Configuration

Source Clock	GPMC_CONFIG1_i GPMCFCLKDIVIDER Field	GPMC_CLK Generated Clock Provided to External Memory Device
GPMC_FCLK	00	GPMC_FCLK
	01	GPMC_FCLK/2
	10	GPMC_FCLK/3
	11	GPMC_FCLK/4

9.2.4.2 GPMC Software Reset

The GPMC can be reset by software through the GPMC_SYSCONFIG SOFTRESET bit. Setting the bit to 1 enables an active software reset that is functionally equivalent to a hardware reset. Hardware and software resets initialize all GPMC registers and the finite state-machine (FSM) immediately and unconditionally. The GPMC_SYSSTATUS RESETDONE bit indicates that the software reset is complete when its value is 1. The software must ensure that the software reset completes before doing GPMC operations.

9.2.4.3 GPMC Power Management

GPMC power is supplied by the CORE power domain, and GPMC power management complies with system power-management guidelines. Table 9-8 describes power-management features available for the GPMC module.

Table 9-8. GPMC Local Power Management Features

Feature	GPMC_SYSCONFIG Register	Description
Clock Auto Gating	AUTOIDLE bit	This bit allows a local power optimization inside the module, by gating the GPMC_FCLK clock upon the internal activity.
Slave Idle Modes	SIDLEMODE field	Force-idle, No-idle and Smart-idle wakeup modes are available
Clock Activity	N/A	Feature not available
Master Standby Modes	N/A	Feature not available
Global Wake-up Enable	N/A	Feature not available
Wake-up Sources Enable	N/A	Feature not available

9.2.4.4 GPMC Interrupt Requests

The GPMC generates one interrupt event as shown in Figure 9-5.

- The interrupt request goes from GPMC (GPMC_IRQ) to the Cortex-A8 MPU subsystem: A_IRQ_100

Table 9-9 lists the event flags, and their mask, that can cause module interrupts.

Table 9-9. GPMC Interrupt Events

GPMC_IRQSTATUS Event Flag	GPMC_IRQENABLE Event Mask	Sensitivity	Map to	Description
WAIT1EDGEDETECTION STATUS	WAIT1EDGEDETECTION ENABLE	Edge	A_IRQ_100	Wait1 edge detection interrupt: Triggered if a rising or falling edge is detected on the GPMC_WAIT1 signal. The rising or falling edge detection of Wait1 is selected through GPMC_CONFIG WAIT1PINPOLARITY bit.
WAIT0EDGEDETECTION STATUS	WAIT0EDGEDETECTION ENABLE	Edge	A_IRQ_100	Wait0 edge detection interrupt: Triggered if a rising or falling edge is detected on the GPMC_WAIT0 signal. The rising or falling edge detection of Wait0 is selected through GPMC_CONFIG WAIT0PINPOLARITY bit.
TERMINALCOUNT STATUS	TERMINALCOUNT ENABLE	Level	A_IRQ_100	Terminal count event: Triggered on prefetch process completion, that is when the number of currently remaining data to be requested reaches 0.
FIFOEVENT STATUS	FIFOEVENT ENABLE	Level	A_IRQ_100	FIFO event interrupt: Indicates FIFO levels availability for in Write-Posting mode and prefetch mode. GPMC_PREFETCH_CONFIG DMAMODE bit shall be cleared to 0.

9.2.4.5 GPMC DMA Requests

The GPMC generates one DMA event as shown in Figure 9-5.

- From GPMC (GPMC_DMA_REQ) to the eDMA: e_DMA_53

9.2.4.6 L3 Slow Interconnect Interface

The GPMC L3 Slow interconnect interface is a pipelined interface including an 16 × 32-bit word write buffer. Any system host can issue external access requests through the GPMC. The device system can issue the following requests through this interface:

- One 8-bit / 16-bit / 32-bit interconnect access (read/write)
- Two incrementing 32-bit interconnect accesses (read/write)
- Two wrapped 32-bit interconnect accesses (read/write)
- Four incrementing 32-bit interconnect accesses (read/write)
- Four wrapped 32-bit interconnect accesses (read/write)
- Eight incrementing 32-bit interconnect accesses (read/write)
- Eight wrapped 32-bit interconnect accesses (read/write)

Only linear burst transactions are supported; interleaved burst transactions are not supported. Only power-of-two-length precise bursts 2 × 32, 4 × 32, 8 × 32 or 16 × 32 with the burst base address aligned on the total burst size are supported (this limitation applies to incrementing bursts only).

This interface also provides one interrupt and one DMA request line, for specific event control.

It is recommended to program the GPMC_CONFIG1_i ATTACHEDDEVICEPAGELENGTH field according to the effective attached device page length and to enable the GPMC_CONFIG1_i WRAPBURST bit if the attached device supports wrapping burst. However, it is possible to emulate wrapping burst on a non-wrapping memory by providing relevant addresses within the page or splitting transactions. Bursts larger than the memory page length are chopped into multiple bursts transactions. Due to the alignment requirements, a page boundary is never crossed.

9.2.4.7 GPMC Address and Data Bus

The current application supports GPMC connection to NAND devices and to address/data-multiplexed memories or devices. Connection to address/data-nonmultiplexed memories Depending on the GPMC configuration of each chip-select, address and data bus lines that are not required for a particular access protocol are not updated (changed from current value) and are not sampled when input (input data bus).

- For address/data-multiplexed and AAD-multiplexed NOR devices, the address is multiplexed on the data bus.
- 8-bit wide NOR devices do not use GPMC I/O: GPMC_D[15-8] for data (they are used for address if needed).
- 16-bit wide NAND devices do not use GPMC I/O: GPMC_A[27-0].
- 8-bit wide NAND devices do not use GPMC I/O: GPMC_A[27-0] and GPMC I/O: GPMC_D[15-8].

9.2.4.7.1 GPMC I/O Configuration Setting

To select a NAND device, program the following register fields:

- GPMC_CONFIG1_i DEVICETYPE field = 10
- GPMC_CONFIG1_i MUXADDDATA field = 00

To select an address/data-multiplexed device, program the following register fields:

- GPMC_CONFIG1_i DEVICETYPE field = 00
- GPMC_CONFIG1_i MUXADDDATA field = 10

To select an address/address/data-multiplexed device, program the following register fields:

- GPMC_CONFIG1_i DEVICETYPE field = 00
- GPMC_CONFIG1_i MUXADDDATA field = 01

To select an address/data-nonmultiplexed device , program the following register fields:

- GPMC_CONFIG1_i DEVICETYPE field = 00
- GPMC_CONFIG1_i MUXADDDATA field = 00

9.2.4.8 Address Decoder and Chip-Select Configuration

Addresses are decoded accordingly with the address request of the chip-select and the content of the chip-select base address register file, which includes a set of global GPMC configuration registers and eight sets of chip-select configuration registers.

The GPMC configuration register file is memory-mapped and can be read or written with byte, 16-bit word, or 32-bit word accesses. The register file should be configured as a noncacheable, nonbufferable region to prevent any desynchronization between host execution (write request) and the completion of register configuration (write completed with register updated). Section 9.5 provides the GPMC register locations. For the map of GPMC memory locations, see Table 9-55.

After the chip-select is configured, the access engine accesses the external device, drives the external interface control signals, and applies the interface protocol based on user-defined timing parameters and settings.

9.2.4.8.1 Chip-Select Base Address and Region Size

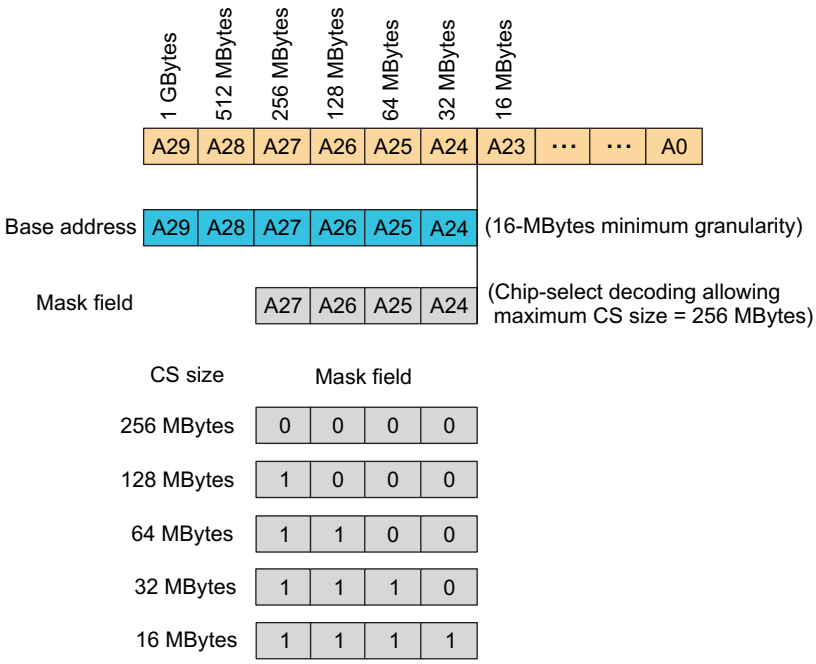
Any external memory or ASIC device attached to the GPMC external interface can be accessed by any device system host within the GPMC 512-Mbyte contiguous address space. For details, see Table 9-55.

The GPMC 512 Mbyte address space can be divided into a maximum of eight chip-select regions with programmable base address and programmable CS size. The CS size is programmable from 16 Mbytes to 256 Mbytes (must be a power-of-2) and is defined by the mask field. Attached memory smaller than the programmed CS region size is accessed through the entire CS region (aliasing).

Each chip-select has a 6-bit base address encoding and a 4-bit decoding mask, which must be programmed according to the following rules:

- The programmed chip-select region base address must be aligned on the chip-select region size address boundary and is limited to a power-of-2 address value. During access decoding, the register base address value is used for address comparison with the address-bit line mapping as described in Figure 9-6 (with A0 as the device system byte-address line). Base address is programmed through the GPMC_CONFIG7_i BASEADDRESS field.
- The register mask is used to exclude some address lines from the decoding. A register mask bit field cleared to 0 suppresses the associated address line from the address comparison (incoming address bit line is don't care). The register mask value must be limited to the subsequent value, based on the desired chip-select region size. Any other value has an undefined result. When multiple chip-select regions with overlapping addresses are enabled concurrently, access to these chip-select regions is cancelled and a GPMC access error is posted. The mask field is programmed through the GPMC_CONFIG7_i MASKADDRESS field.

Figure 9-6. Chip-Select Address Mapping and Decoding Mask



Chip-select configuration (base and mask address or any protocol and timing settings) must be performed while the associated chip-select is disabled through the GPMC_CONFIG7_i CSVALID bit (i = 0 to 5). In addition, a chip-select configuration can only be disabled if there is no ongoing access to that chip-select. This requires activity monitoring of the prefetch or write-posting engine if the engine is active on the chip-select. Also, the write buffer state must be monitored to wait for any posted write completion to the chip-select.

Any access attempted to a nonvalid GPMC address region (CSVALID disabled or address decoding outside a valid chip-select region) is not propagated to the external interface and a GPMC access error is posted. In case of chip-selects overlapping, an error is generated and no access will occur on either chip-select. Chip-select 0 is the only chip-select region enabled after either a power-up or a GPMC reset.

Although the GPMC interface can drive up to eight chip-selects, the frequency specified for this interface is for a specific load. If this load is exceeded, the maximum frequency cannot be reached. One solution is to implement a board with buffers, to allow the slowest device to maintain the total load on the lines.

9.2.4.8.2 Access Protocol

9.2.4.8.2.1 Supported Devices

The access protocol of each chip-select can be independently specified through the GPMC_CONFIG1_i DEVICETYPE parameter (where i = 0 to 5) for:

- Random-access synchronous or asynchronous memory like NOR flash, SRAM
- NAND flash asynchronous devices

For more information about the NAND flash GPMC basic programming model and NAND support, see Section 9.2.4.12 and Section 9.2.4.12.1.

9.2.4.8.2.2 Access Size Adaptation and Device Width

Each chip-select can be independently configured through the GPMC_CONFIG1_i DEVICESIZE field (where i = 0 to 5) to interface with a 16-bit wide device or an 8-bit wide device. System requests with data width greater than the external device data bus width are split into successive accesses according to both the external device data-bus width and little-endian data organization.

The device does not provide the A0 byte address line required for random-byte addressable 8-bit wide device interfacing (for both multiplexed and nonmultiplexed protocol). It limits the use of 8-bit wide device interfacing to byte-alias accesses. This limitation is not applicable to NAND device interfacing (8-bit wide or 16-bit wide devices).

9.2.4.8.2.3 Address/Data-Multiplexing Interface

For random synchronous or asynchronous memory interfacing (DEVICETYPE = 00), an address- and data-multiplexing protocol can be selected through the GPMC_CONFIG1_i MUXADDDATA field (where i = 0 to 5). The $\overline{\text{ADV}}$ signal must be used as the external device address latch control signal. For the associated chip-select configuration, $\overline{\text{ADV}}$ assertion and deassertion time and $\overline{\text{OE}}$ assertion time must be set to the appropriate value to meet the address latch setup/hold time requirements of the external device (see Section 9.2.3).

This address/data-multiplexing interface is not applicable to NAND device interfacing. NAND devices require a specific address, command, and data multiplexing protocol (see Section 9.2.4.12).

9.2.4.8.3 External Signals

9.2.4.8.3.1 WAIT Pin Monitoring Control

GPMC access time can be dynamically controlled using an external gpmc_wait pin when the external device access time is not deterministic and cannot be defined and controlled only using the GPMC internal RDACCESSTIME, WRACCESSTIME and PAGEBURSTACCESSTIME wait state generator.

The GPMC features two input wait pin:gpmc_wait1, and gpmc_wait0. This pin allow control of external devices with different wait-pin polarity. They also allow the overlap of wait-pin assertion from different devices without affecting access to devices for which the wait pin is not asserted.

- The GPMC_CONFIG1_i WAITPINSELECT field (where i = 0 to 5) selects which input gpmc_wait pin is used for the device attached to the corresponding chip-select.
- The polarity of the wait pin is defined through the WAITxPINPOLARITY bit of the GPMC_CONFIG register. A wait pin configured to be active low means that low level on the WAIT signal indicates that the data is not ready and that the data bus is invalid. When WAIT is inactive, data is valid.

The GPMC access engine can be configured per CS to monitor the wait pin of the external memory device or not, based on the access type: read or write.

- The GPMC_CONFIG1_i WAITREADMONITORING bit defines whether the wait pin should be monitored during read accesses or not.
- The GPMC_CONFIG1_i WAITWRITEMONITORING bit defines whether the wait pin should be monitored during write accesses or not.

The GPMC access engine can be configured to monitor the wait pin of the external memory device asynchronously or synchronously with the GPMC_CLK clock, depending on the access type: synchronous or asynchronous (the GPMC_CONFIG1_i READTYPE and GPMC_CONFIG1_i WRITETYPE bits).

9.2.4.8.3.2 Wait Monitoring During an Asynchronous Read Access

When wait-pin monitoring is enabled for read accesses (WAITREADMONITORING), the effective access time is a logical AND combination of the RDACCESSTIME timing completion and the wait-deasserted state.

During asynchronous read accesses with wait-pin monitoring enabled, the wait pin must be at a valid level (asserted or deasserted) for at least two GPMC clock cycles before RDACCESSTIME completes, to ensure correct dynamic access-time control through wait-pin monitoring. The advance pipelining of the two GPMC clock cycles is the result of the internal synchronization requirements for the WAIT signal.

In this context, RDACCESSTIME is used as a WAIT invalid timing window and is set to such a value that the wait pin is at a valid state two GPMC clock cycles before RDACCESSTIME completes.

Similarly, during a multiple-access cycle (for example, asynchronous read page mode), the effective access time is a logical AND combination of PAGEBURSTACCESSTIME timing completion and the wait-deasserted state. Wait-monitoring pipelining is also applicable to multiple accesses (access within a page).

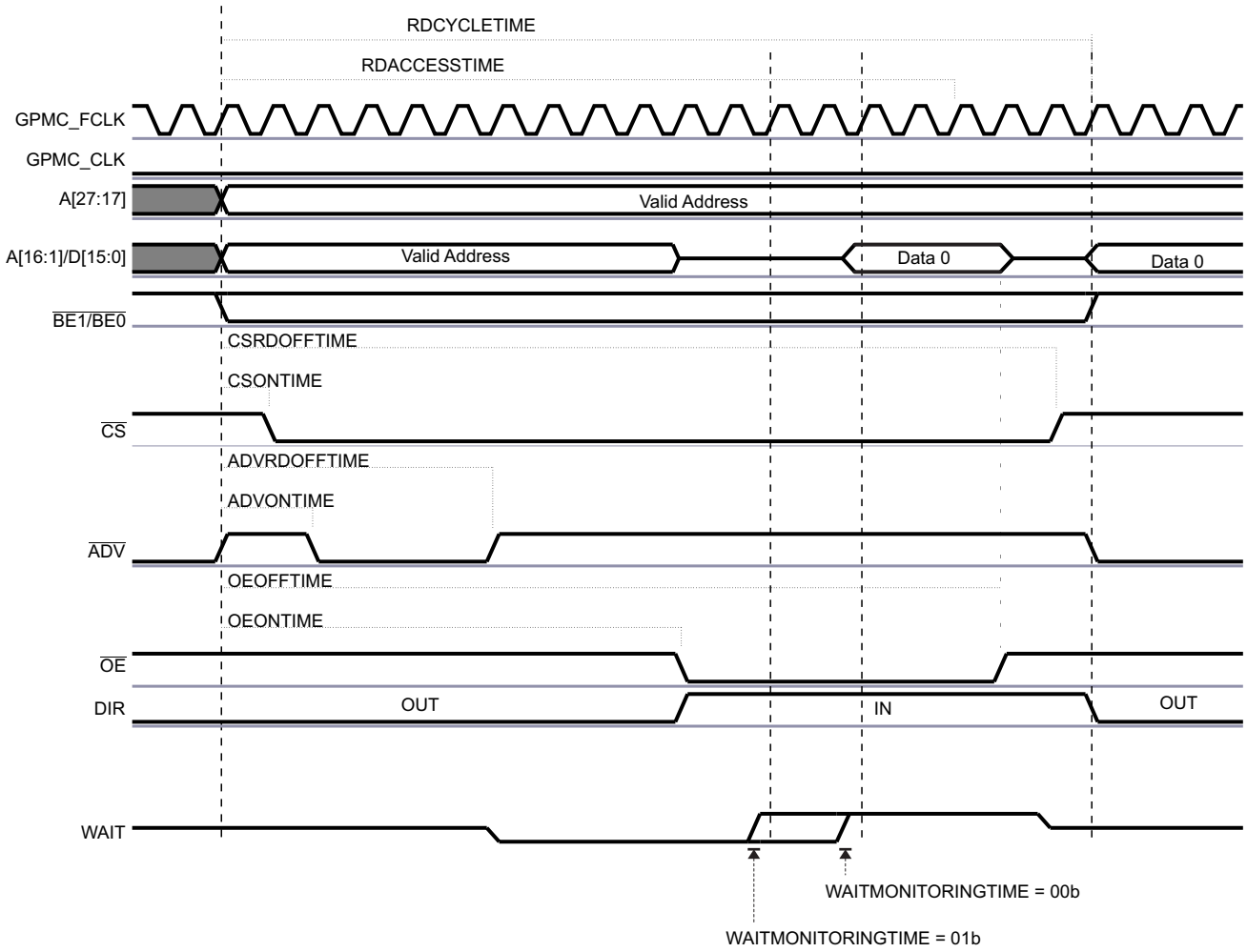
- WAIT monitored as active freezes the CYCLETIME counter. For an access within a page, when the CYCLETIME counter is by definition in a lock state, WAIT monitored as asserted extends the current access time in the page. Control signals are kept in their current state. The data bus is considered invalid, and no data are captured during this clock cycle.
- WAIT monitored as inactive unfreezes the CYCLETIME counter. For an access within a page, when the CYCLETIME counter is by definition in a lock state, WAIT monitored as inactive completes the current access time and starts the next access phase in the page. The data bus is considered valid, and data are captured during this clock cycle. In case of a single access or if this was the last access in a multiple-access cycle, all signals are controlled according to their related control timing value and according to the CYCLETIME counter status.

When a delay larger than two GPMC clocks must be observed between wait-pin deactivation time and data valid time (including the required GPMC and the device data setup time), an extra delay can be added between wait-pin deassertion time detection and effective data-capture time and the effective unlock of the CYCLETIME counter. This extra delay can be programmed in the GPMC_CONFIG1_i WAITMONITORINGTIME field (i = 0 to 5).

- The WAITMONITORINGTIME parameter does not delay the wait-pin active or inactive detection, nor does it modify the two GPMC clocks pipelined detection delay.
- This extra delay is expressed as a number of GPMC_CLK clock cycles, even though the access is defined as asynchronous, and no GPMC_CLK clock is provided to the external device. Still, GPMCFCLKDIVIDER is used as a divider for the GPMC clock, so it must be programmed to define the correct WAITMONITORINGTIME delay.

Figure 9-7 shows wait behavior during an asynchronous single read access.

Figure 9-7. Wait Behavior During an Asynchronous Single Read Access (GPMCFCLKDivider = 1)



The WAIT signal is active low. GPMC_CONFIG1_i WAITMONITORINGTIME field = 00 or 01.

9.2.4.8.3.3 Wait Monitoring During an Asynchronous Write Access

When wait-pin monitoring is enabled for write accesses (GPMC_CONFIG1_i WAITWRITEMONITORING bit = 1), the WAIT-invalid timing window is defined by the WRACCESSTIME field. WRACCESSTIME must be set so that the wait pin is at a valid state two GPMC clock cycles before WRACCESSTIME completes. The advance pipelining of the two GPMC clock cycles is the result of the internal synchronization requirements for the WAIT signal.

- WAIT monitored as active freezes the CYCLETIME counter. This informs the GPMC that the data bus is not captured by the external device. The control signals are kept in their current state. The data bus still drives the data.
- WAIT monitored as inactive unfreezes the CYCLETIME counter. This informs that the data bus is correctly captured by the external device. All signals, including the data bus, are controlled according to their related control timing value and to the CYCLETIME counter status.

When a delay larger than two GPMC clock cycles must be observed between wait-pin deassertion time and the effective data write into the external device (including the required GPMC data setup time and the device data setup time), an extra delay can be added between wait-pin deassertion time detection and effective data write time into the external device and the effective unfreezing of the CYCLETIME counter. This extra delay can be programmed in the GPMC_CONFIG1_i WAITMONITORINGTIME field (i = 0 to 5).

- The WAITMONITORINGTIME parameter does not delay the wait-pin assertion or deassertion detection, nor does it modify the two GPMC clock cycles pipelined detection delay.
- This extra delay is expressed as a number of GPMC_CLK clock cycles, even though the access is defined as asynchronous, and even though no clock is provided to the external device. Still, GPMC_CONFIG1_i GPMCFCLKDIVIDER field is used as a divider for the GPMC clock and so it must be programmed to define the correct WAITMONITORINGTIME delay.

9.2.4.8.3.4 Wait Monitoring During a Synchronous Read Access

During synchronous accesses with wait-pin monitoring enabled, the wait pin is captured synchronously with GPMC_CLK, using the rising edge of this clock.

The WAIT signal can be programmed to apply to the same clock cycle it is captured in. Alternatively, it can be sampled one or two GPMC_CLK cycles ahead of the clock cycle it applies to. This pipelining is applicable to the entire burst access, and to all data phase in the burst access. This WAIT pipelining depth is programmed in the GPMC_CONFIG1_i WAITMONITORINGTIME field (i = 0 to 5), and is expressed as a number of GPMC_CLK clock cycles.

In synchronous mode, when wait-pin monitoring is enabled (GPMC_CONFIG1_i WAITREADMONITORING bit), the effective access time is a logical AND combination of the RDACCESSTIME timing completion and the WAIT deasserted-state detection.

Depending on the programmed WAITMONITORINGTIME value, the wait pin should be at a valid level, either asserted or deasserted:

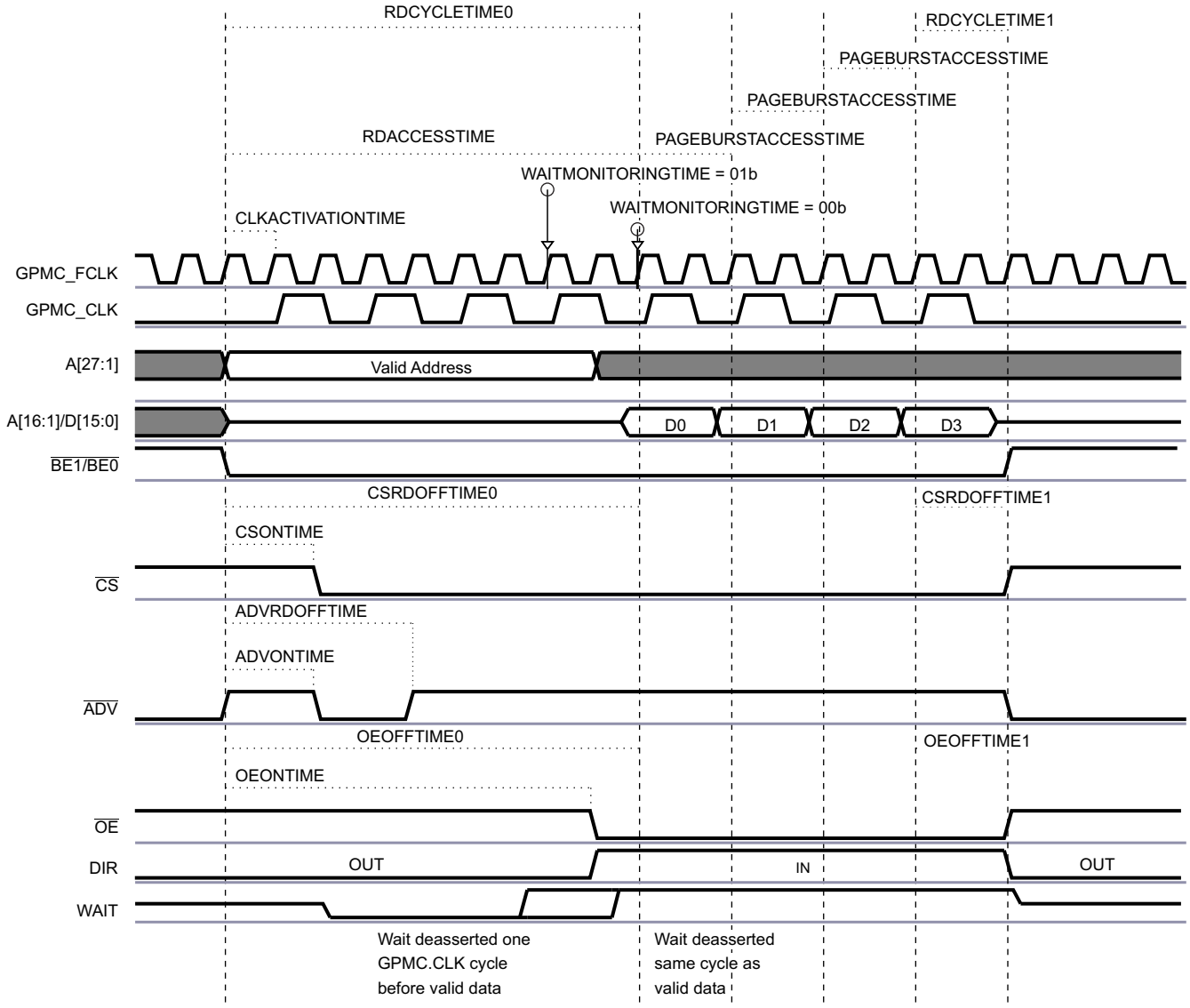
- In the same clock cycle the data is valid if WAITMONITORINGTIME = 0 (at RDACCESSTIME completion)
- In the WAITMONITORINGTIME × (GPMCFCLKDIVIDER + 1) GPMC_FCLK clock cycles before RDACCESSTIME completion if WAITMONITORINGTIME is not equal to 0

Similarly, during a multiple-access cycle (burst mode), the effective access time is a logical AND combination of PAGEBURSTACCESSTIME timing completion and the wait-inactive state. The Wait pipelining depth programming applies to the whole burst access.

- WAIT monitored as active freezes the CYCLETIME counter. For an access within a burst (when the CYCLETIME counter is by definition in a lock state), WAIT monitored as active extends the current access time in the burst. Control signals are kept in their current state. The data bus is considered invalid, and no data are captured during this clock cycle.
- WAIT monitored as inactive unfreezes the CYCLETIME counter. For an access within a burst (when the CYCLETIME counter is by definition in lock state), WAIT monitored as inactive completes the current access time and starts the next access phase in the burst. The data bus is considered valid, and data are captured during this clock cycle. In a single access or if this was the last access in a multiple-access cycle, all signals are controlled according to their relative control timing value and the CYCLETIME counter status.

Figure 9-8 shows wait behavior during a synchronous read burst access.

Figure 9-8. Wait Behavior During a Synchronous Read Burst Access



The WAIT signal is active low. WAITMONITORINGTIME field = 00 or 01.

9.2.4.8.3.5 Wait Monitoring During a Synchronous Write Access

During synchronous accesses with wait-pin monitoring enabled (the WAITWRITEMONITORING bit), the wait pin is captured synchronously with GPMC_CLK, using the rising edge of this clock.

If enabled, external wait-pin monitoring can be used in combination with WRACCESSTIME to delay the effective memory device GPMC_CLK capture edge.

Wait-monitoring pipelining depth is similar to synchronous read access:

- At WRACCESSTIME completion if WAITMONITORINGTIME = 0
- In the WAITMONITORINGTIME × (GPMCFCLKDIVIDER + 1) GPMC_FCLK cycles before WRACCESSTIME completion if WAITMONITORINGTIME is not equal to 0.

Wait-monitoring pipelining definition applies to whole burst accesses:

- WAIT monitored as active freezes the CYCLETIME counter. For accesses within a burst, when the CYCLETIME counter is by definition in a lock state, WAIT monitored as active indicates that the data bus is not being captured by the external device. Control signals are kept in their current state. The data bus is kept in its current state.
- WAIT monitored as inactive unfreezes the CYCLETIME counter. For accesses within a burst, when the CYCLETIME counter is by definition in a lock state, WAIT monitored as inactive indicates the effective data capture of the bus by the external device and starts the next access of the burst. In case of a single access or if this was the last access in a multiple access cycle, all signals, including the data bus, are controlled according to their related control timing value and the CYCLETIME counter status.

Wait monitoring is supported for all configurations except for GPMC_CONFIG1_i WAITMONITORINGTIME field = 0 (where i = 0 to 3) for write bursts with a clock divider of 1 or 2 (GPMC_CONFIG1_i GPMCFCLKDIVIDER field is equal to 0 or 1, respectively).

9.2.4.8.3.6 WAIT with NAND Device

For details about the use of the wait pin for communication with a NAND flash external device, see Section 9.2.4.12.2.

9.2.4.8.3.7 Idle Cycle Control Between Successive Accesses

9.2.4.8.3.7.1 Bus Turnaround (BUSTURNAROUND)

To prevent data-bus contention, an access that follows a read access to a slow memory/device must be delayed (in other words, control the $\overline{CS}$/$\overline{OE}$ de-assertion to data bus in high-impedance delay).

The bus turnaround is a time-out counter starting after $\overline{CS}$ or $\overline{OE}$ de-assertion time, whichever occurs first, and delays the next access start-cycle time. The counter is programmed through the GPMC_CONFIG6_i BUSTURNAROUND field (i = 0 to 5).

After a read access to a chip-select with a nonzero BUSTURNAROUND, the next access is delayed until the BUSTURNAROUND delay completes, if the next access is one of the following:

- A write access to any chip-select (same or different from the chip-select data was read from)
- A read access to a different chip-select from the chip-select data was read access from
- A read or write access to a chip-select associated with an address/data-multiplexed device

Bus keeping starts after bus turnaround completion so that DIR changes from IN to OUT after bus turnaround. The bus will not have enough time to go into high-impedance even though it could be driven with the same value before bus turnaround timing.

BUSTURNAROUND delay runs in parallel with GPMC_CONFIG6_i CYCLE2CYCLEDELAY delays. It should be noted that BUSTURNAROUND is a timing parameter for the ending chip-select access while CYCLE2CYCLEDELAY is a timing parameter for the following chip-select access. The effective minimum delay between successive accesses is driven by these delay timing parameters and by the access type of the following access. See Figure 9-9 to Figure 9-11.

Another way to prevent bus contention is to define an earlier $\overline{CS}$ or $\overline{OE}$ deassertion time for slow devices or to extend the value of RDCYCLETIME. Doing this prevents bus contention, but affects all accesses of this specific chip-select.

Figure 9-9. Read to Read for an Address-Data Multiplexed Device, On Different CS, Without Bus Turnaround ($\overline{CS0}$ Attached to Fast Device)

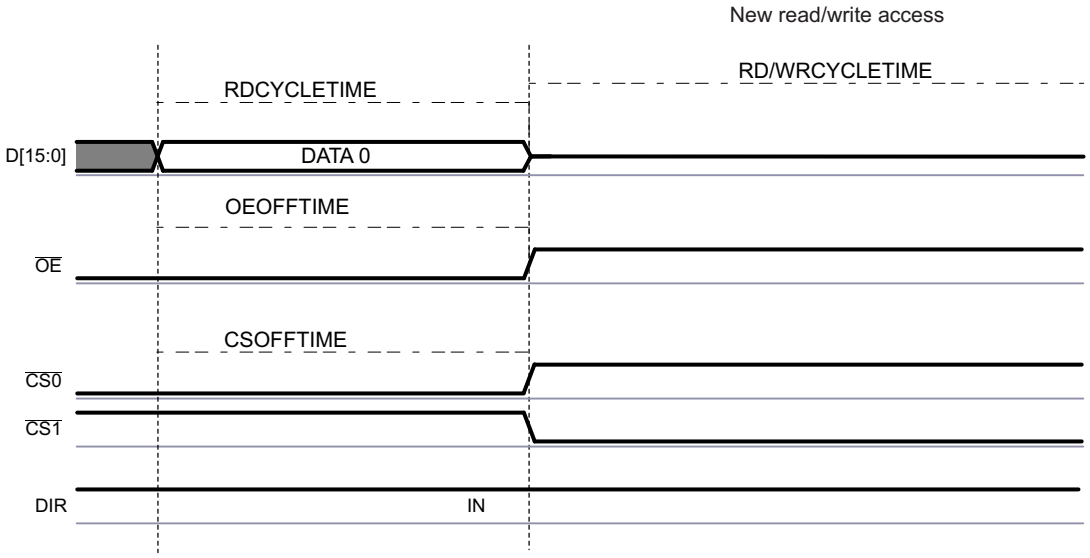


Figure 9-10. Read to Read / Write for an Address-Data Multiplexed Device, On Different CS, With Bus Turnaround

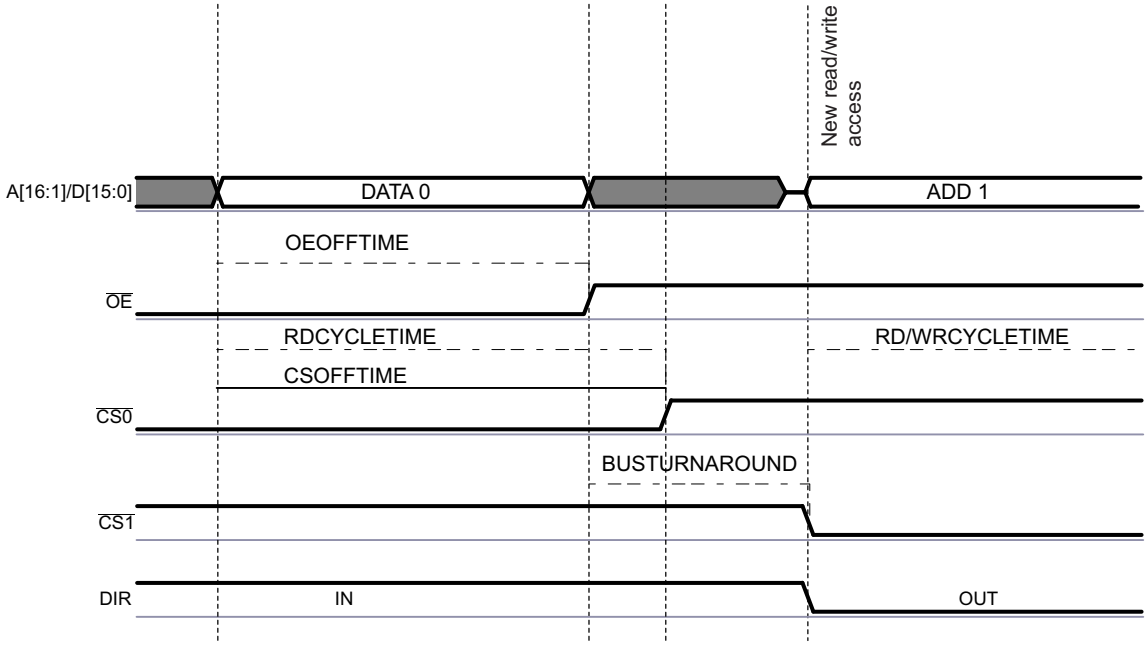
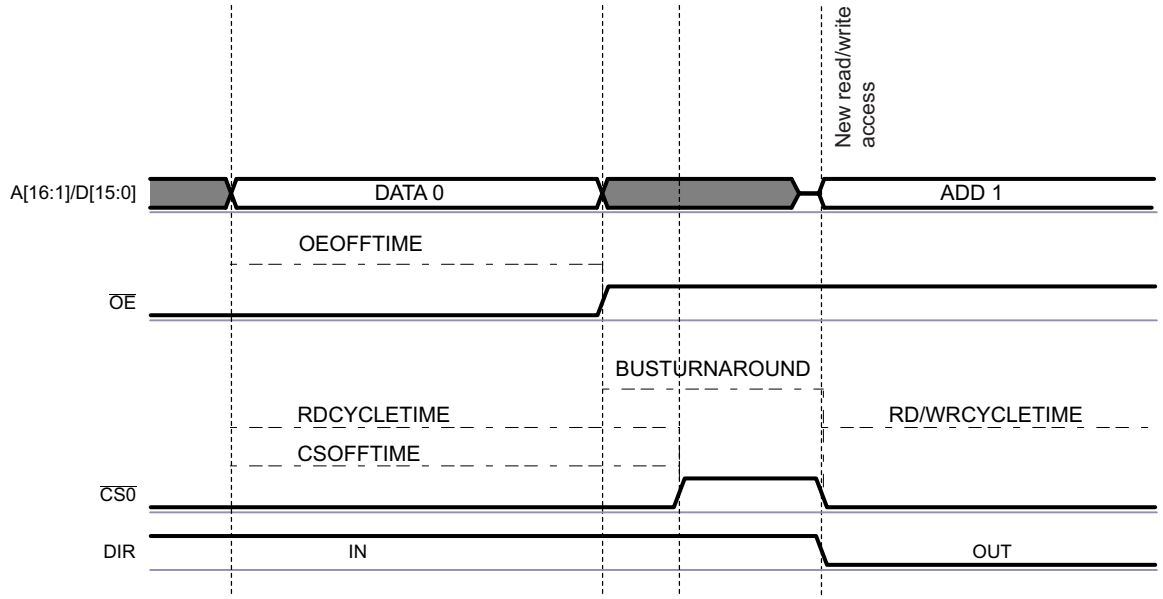


Figure 9-11. Read to Read / Write for a Address-Data or AAD-Multiplexed Device, On Same CS, With Bus Turnaround



9.2.4.8.3.7.2 Idle Cycles Between Accesses to Same Chip-Select (CYCLE2CYCLESAMECSEN, CYCLE2CYCLEDELAY)

Some devices require a minimum chip-select signal inactive time between accesses. The GPMC_CONFIG6_i CYCLE2CYCLESAMECSEN bit (i = 0 to 5) enables insertion of a minimum number of GPMC_FCLK cycles, defined by the GPMC_CONFIG6_i CYCLE2CYCLEDELAY field, between successive accesses of any type (read or write) to the same chip-select.

If CYCLE2CYCLESAMECSEN is enabled, any subsequent access to the same chip-select is delayed until its CYCLE2CYCLEDELAY completes. The CYCLE2CYCLEDELAY counter starts when CSRDOFFTIME/CSWROFFTIME completes.

The same applies to successive accesses occurring during 32-bit word or burst accesses split into successive single accesses when the single-access mode is used (GPMC_CONFIG1_i READMULTIPLE = 0 or GPMC_CONFIG1_i WRITEMULTIPLE = 0).

All control signals are kept in their default states during these idle GPMC_FCLK cycles. This prevents back-to-back accesses to the same chip-select without idle cycles between accesses.

9.2.4.8.3.7.3 Idle Cycles Between Accesses to Different Chip-Select (CYCLE2CYCLEDIFFCSEN, CYCLE2CYCLEDELAY)

Because of the pipelined behavior of the system, successive accesses to different chip-selects can occur back-to-back with no idle cycles between accesses. Depending on the control signals ($\overline{CS}$, $\overline{ADV}$_ALE, $\overline{BE0}$_CLE, $\overline{OE\_RE}$, $\overline{WE}$) assertion and de-assertion timing parameters and on the IC timing parameters, some control signals assertion times may overlap between the successive accesses to different CS. Similarly, some control signals ($\overline{WE}$, $\overline{OE\_RE}$) may not respect required transition times.

To work around the overlapping and to observe the required control-signal transitions, a minimum of CYCLE2CYCLEDELAY inactive cycles is inserted between the access being initiated to this chip-select and the previous access ending for a different chip-select. This applies to any type of access (read or write).

If GPMC_CONFIG6_i CYCLE2CYCLEDIFFCSEN is enabled, the chip-select access is delayed until CYCLE2CYCLEDELAY cycles have expired since the end of a previous access to a different chip-select. CYCLE2CYCLEDELAY count starts at CSRDOFFTIME/CSWROFFTIME completion. All control signals are kept inactive during the idle GPMC_FCLK cycles.

CYCLE2CYCLESAMECSEN and CYCLE2CYCLEDIFFCSEN should be set in registers to respectively get idle cycles inserted between accesses on this chip-select and after accesses to a different chip-select.

The CYCLE2CYCLEDELAY delay runs in parallel with the BUSTURNAROUND delay. It should be noted that BUSTURNAROUND is a timing parameter defined for the ending chip-select access, whereas CYCLE2CYCLEDELAY is a timing parameter defined for the starting chip-select access. The effective minimum delay between successive accesses is based on the larger delay timing parameter and on access type combination, since bus turnaround does not apply to all access types. See Section 9.2.4.8.3.7.1 for more details on bus turnaround.

Table 9-10 describes the configuration required for idle cycle insertion.

Table 9-10. Idle Cycle Insertion Configuration

First Access Type	BUSTURNAROUND Timing Parameter	Second Access Type	Chip-Select	Addr/Data Multiplexed	CYCLE2CYCLE SAMECSEN Parameter	CYCLE2CYCLE DIFFCSEN Parameter	Idle Cycle Insertion Between the Two Accesses
R/W	0	R/W	Any	Any	0	x	No idle cycles are inserted if the two accesses are well pipelined.
R	>0	R	Same	Nonmuxed	x	0	No idle cycles are inserted if the two accesses are well pipelined.
R	>0	R	Different	Nonmuxed	0	0	BUSTURNAROUND cycles are inserted.
R	>0	R/W	Any	Muxed	0	0	BUSTURNAROUND cycles are inserted.
R	>0	W	Any	Any	0	0	BUSTURNAROUND cycles are inserted.
W	>0	R/W	Any	Any	0	0	No idle cycles are inserted if the two accesses are well pipelined.
R/W	0	R/W	Same	Any	1	x	CYCLE2CYCLEDELAY cycles are inserted.
R/W	0	R/W	Different	Any	x	1	CYCLE2CYCLEDELAY cycles are inserted.
R/W	>0	R/W	Same	Any	1	x	CYCLE2CYCLEDELAY cycles are inserted. If BTA idle cycles already apply on these two back-to-back accesses, the effective delay is max (BUSTURNAROUND, CYCLE2CYCLEDELAY).
R/W	>0	R/W	Different	Any	x	1	CYCLE2CYCLEDELAY cycles are inserted. If BTA idle cycles already apply on these two back-to-back accesses, the effective delay is maximum (BUSTURNAROUND, CYCLE2CYCLEDELAY).

9.2.4.8.3.8 Slow Device Support (TIMEPARAGRANULARITY Parameter)

All access-timing parameters can be multiplied by 2 by setting the GPMC_CONFIG1_i TIMEPARAGRANULARITY bit (where i = 0 to 5). Increasing all access timing parameters allows support of slow devices.

9.2.4.8.3.9 GPMC_DIR Pin

The GPMC_DIR pin is used to control I/O direction on the GPMC data bus GPMC_D[15-0]. Depending on top-level pad multiplexing, this signal can be output and used externally to the device, if required. The GPMC_DIR pin is low during transmit (OUT) and high during receive (IN).

For write accesses, the GPMC_DIR pin stays OUT from start-cycle time to end-cycle time.

For read accesses, the GPMC_DIR pin goes from OUT to IN at $\overline{OE}$ assertion time and stays IN until:

- BUSTURNAROUND is enabled
  - The GPMC_DIR pin goes from IN to OUT at end-cycle time plus programmable bus turnaround time.
- BUSTURNAROUND is disabled
  - After an asynchronous read access, the GPMC_DIR pin goes from IN to OUT at RDACCESSTIME + 1 GPMC_FCLK cycle or when RDCYCLETIME completes, whichever occurs last.
  - After a synchronous read access, the GPMC_DIR pin goes from IN to OUT at RDACCESSTIME + 2 GPMC_FCLK cycles or when RDCYCLETIME completes, whichever occurs last.

Because of the bus-keeping feature of the GPMC, after a read or write access and with no other accesses pending, the default value of the GPMC_DIR pin is OUT (see Section 9.2.4.9.10). In nonmultiplexed devices, the GPMC_DIR pin stays IN between two successive read accesses to prevent unnecessary toggling.

9.2.4.8.3.10 Reset

No reset signal is sent to the external memory device by the GPMC. For more information about external-device reset, see *Power, Reset, and Clock Management (PRCM) Module* chapter.

The PRCM module provides an input pin, global_rst_n, to the GPMC:

- The global_rst_n pin is activated during device warm reset and cold reset.
- The global_rst_n pin initializes the internal state-machine and the internal configuration registers.

9.2.4.8.3.11 Write Protect Signal ($\overline{WP}$)

When connected to the attached memory device, the write protect signal can enable or disable the lockdown function of the attached memory. The $\overline{GPMC\_WP}$ output pin value is controlled through the GPMC_CONFIG WRITEPROTECT bit, which is common to all CS.

9.2.4.8.3.12 Byte Enable ($\overline{BE1}$/$\overline{BE0}$)

Byte enable signals ($\overline{BE1}$/$\overline{BE0}$) are:

- Valid (asserted or nonasserted according to the incoming system request) from access start to access completion for asynchronous and synchronous single accesses
- Asserted low from access start to access completion for asynchronous and synchronous multiple read accesses
- Valid (asserted or nonasserted, according to the incoming system request) synchronously to each written data for synchronous multiple write accesses

9.2.4.8.4 Error Handling

When an error occurs in the GPMC, the error information is stored in the GPMC_ERR_TYPE register and the address of the illegal access is stored in the GPMC_ERR_ADDRESS register. The GPMC keeps only the first error abort information until the GPMC_ERR_TYPE register is reset. Subsequent accesses that cause errors are not logged until the error is cleared by hardware with the GPMC_ERR_TYPE ERRORVALID bit.

- ERRORNOTSUPPADD occurs when an incoming system request address decoding does not match any valid chip-select region, or if two chip-select regions are defined as overlapped, or if a register file access is tried outside the valid address range of 1KB.

- ERRORNOTSUPPMCMD occurs when an unsupported command request is decoded at the L3 Slow interconnect interface
- ERRORTIMEOUT: A time-out mechanism prevents the system from hanging. The start value of the

9-bit time-out counter is defined in the GPMC_TIMEOUT_CONTROL register and enabled with the GPMC_TIMEOUT_CONTROL TIMEOUTENABLE bit. When enabled, the counter starts at start-cycle time until it reaches 0 and data is not responded to from memory, and then a time-out error occurs. When data are sent from memory, this counter is reset to its start value. With multiple accesses (asynchronous page mode or synchronous burst mode), the counter is reset to its start value for each data access within the burst.

The GPMC does not generate interrupts on these errors. True abort to the MPU or interrupt generation is handled at the interconnect level.

9.2.4.9 Timing Setting

The GPMC offers the maximum flexibility to support various access protocols. Most of the timing parameters of the protocol access used by the GPMC to communicate with attached memories or devices are programmable on a chip-select basis. Assertion and deassertion times of control signals are defined to match the attached memory or device timing specifications and to get maximum performance during accesses. For more information on GPMC_CLK and GPMC_FCLK see Section 9.2.4.9.6.

In the following sections, the start access time refer to the time at which the access begins.

9.2.4.9.1 Read Cycle Time and Write Cycle Time (RDCYCLETIME / WRCYCLETIME)

The GPMC_CONFIG5_i RDCYCLETIME and GPMC_CONFIG5_i WRCYCLETIME fields (i = 0 to 5) define the address bus and byte enables valid times for read and write accesses. To ensure a correct duty cycle of GPMC_CLK between accesses, RDCYCLETIME and WRCYCLETIME are expressed in GPMC_FCLK cycles and must be multiples of the GPMC_CLK cycle. RDCYCLETIME and WRCYCLETIME fields can be set with a granularity of 1 or 2 throught GPMC_CONFIG1_i TIMEPARAGRANULARITY.

When either RDCYCLETIME or WRCYCLETIME completes, if they are not already deasserted, all control signals ($\overline{CS}$, $\overline{ADV}$_ALE, $\overline{OE}$_$\overline{RE}$, $\overline{WE}$, and $\overline{BE0}$_CLE) are deasserted to their reset values, regardless of their deassertion time parameters.

An exception to this forced deassertion occurs when a pipelined request to the same chip-select or to a different chip-select is pending. In such a case, it is not necessary to deassert a control signal with deassertion time parameters equal to the cycle-time parameter. This exception to forced deassertion prevents any unnecessary glitches. This requirement also applies to BE signals, thus avoiding an unnecessary BE glitch transition when pipelining requests.

If no inactive cycles are required between successive accesses to the same or to a different chip-select (GPMC_CONFIG6_i CYCLE2CYCLESAMECSEN = 0 or GPMC_CONFIG6_i CYCLE2CYCLEDIFFCSEN = 0, where i = 0 to 3), and if assertion-time parameters associated with the pipelined access are equal to 0, asserted control signals ($\overline{CS}$, $\overline{ADV}$_ALE, $\overline{BE0}$_CLE, $\overline{WE}$, and $\overline{OE}$_$\overline{RE}$) are kept asserted. This applies to any read/write to read/write access combination.

If inactive cycles are inserted between successive accesses, that is, CYCLE2CYCLESAMECSEN = 1 or CYCLE2CYCLEDIFFCSEN = 1, the control signals are forced to their respective default reset values for the number of GPMC_FCLK cycles defined in CYCLE2CYCLEDELAY.

9.2.4.9.2 $\overline{CS}$: Chip-Select Signal Control Assertion/Deassertion Time (CSONTIME / CSRDOFFTIME / CSWROFFTIME / CSEXTRADELAY)

The GPMC_CONFIG2_i CSONTIME field (where i = 0 to 5) defines the $\overline{CS}$ signal-assertion time relative to the start access time. It is common for read and write accesses.

The GPMC_CONFIG2_i CSRDOFFTIME (read access) and GPMC_CONFIG2_i CSWROFFTIME (write access) fields define the $\overline{CS}$ signal deassertion time relative to start access time.

CSONTIME, CSRDOFFTIME, and CSWROFFTIME parameters are applicable to synchronous and asynchronous modes. CSONTIME can be used to control an address and byte enable setup time before chip-select assertion. CSRDOFFTIME and CSWROFFTIME can be used to control an address and byte enable hold time after chip-select deassertion.

$\overline{CS}$ signal transitions as controlled through CSONTIME, CSRDOFFTIME, and CSWROFFTIME can be delayed by half a GPMC_FCLK period by enabling the GPMC_CONFIG2_i CSEXTRADELAY bit. This half of a GPMC_FCLK period provides more granularity on the $\overline{CS}$ assertion and deassertion time to guarantee proper setup and hold time relative to GPMC_CLK. CSEXTRADELAY is especially useful in configurations where GPMC_CLK and GPMC_FCLK have the same frequency, but can be used for all GPMC configurations. If enabled, CSEXTRADELAY applies to all parameters controlling $\overline{CS}$ transitions.

The CSEXTRADELAY bit must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the RDCYCLETIME and WRCYCLETIME fields to be greater than the $\overline{CS}$ signal-deassertion time, including the extra half-GPMC_FCLK-period delay.

9.2.4.9.3 $\overline{ADV}$_ALE: Address Valid/Address Latch Enable Signal Control Assertion/Deassertion Time (ADVONTIME / ADVRDOFFTIME / ADVWROFFTIME / ADVEXTRADELAY / ADVAADMUXONTIME / ADVAADMUXRDOFFTIME / ADVAADMUXWROFFTIME)

The GPMC_CONFIG3_i ADVONTIME field (where i = 0 to 5) defines the $\overline{ADV}$_ALE signal-assertion time relative to start access time. It is common to read and write accesses.

The GPMC_CONFIG3_i ADVRDOFFTIME (read access) and GPMC_CONFIG3_i ADVWROFFTIME (write access) fields define the $\overline{ADV}$_ALE signal-deassertion time relative to start access time.

ADVONTIME can be used to control an address and byte enable valid setup time control before $\overline{ADV}$_ALE assertion. ADVRDOFFTIME and ADVWROFFTIME can be used to control an address and byte enable valid hold time control after $\overline{ADV}$_ALE de-assertion. ADVRDOFFTIME and ADVWROFFTIME are applicable to both synchronous and asynchronous modes.

$\overline{ADV}$_ALE signal transitions as controlled through ADVONTIME, ADVRDOFFTIME, and ADVWROFFTIME can be delayed by half a GPMC_FCLK period by enabling the GPMC_CONFIG3_i ADVEXTRADELAY bit. This half of a GPMC_FCLK period provides more granularity on $\overline{ADV}$_ALE assertion and deassertion time to assure proper setup and hold time relative to GPMC_CLK. The ADVEXTRADELAY configuration parameter is especially useful in configurations where GPMC_CLK and GPMC_FCLK have the same frequency, but can be used for all GPMC configurations. If enabled, ADVEXTRADELAY applies to all parameters controlling $\overline{ADV}$_ALE transitions.

ADVEXTRADELAY must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the RDCYCLETIME and WRCYCLETIME fields to be greater than $\overline{ADV}$_ALE signal-deassertion time, including the extra half-GPMC_FCLK-period delay.

The GPMC_CONFIG3_i ADVAADMUXONTIME, GPMC_CONFIG3_i ADVAADMUXRDOFFTIME, and GPMC_CONFIG3_i ADVAADMUXWROFFTIME parameters have the same functions as ADVONTIME, ADVRDOFFTIME, and ADVWROFFTIME, but apply to the first address phase in the AAD-multiplexed protocol. It is the user responsibility to make sure ADVAADMUXxxOFFTIME is programmed to a value lower than or equal to ADVxxOFFTIME. Functionality in AAD-mux mode is undefined if the settings do not comply with this requirement. ADVAADMUXxxOFFTIME can be programmed to the same value as ADVONTIME if no high $\overline{ADV}$ pulse is needed between the two AAD-mux address phases, which is the typical case in synchronous mode. In this configuration, $\overline{ADV}$ is kept low until it reaches the correct ADVxxOFFTIME.

See Section 9.2.4.12 for more details on ADVONTIME, ADVRDOFFTIME, ADVWROFFTIME, and ADVAADMUXRDOFFTIME, ADVAADMUXWROFFTIME usage for CLE and ALE (Command / Address Latch Enable) usage for a NAND Flash interface.

9.2.4.9.4 $\overline{OE\_RE}$: Output Enable / Read Enable Signal Control Assertion / Deassertion Time (OEONTIME / OEOFFTIME / OEEXTRADELAY / OEAADMUXONTIME / OEAADMUXOFFTIME)

The GPMC_CONFIG4_i OEONTIME field (where i = 0 to 5) defines the $\overline{OE\_RE}$ signal assertion time relative to start access time. It is applicable only to read accesses.

The GPMC_CONFIG4_i OEOFFTIME field defines the $\overline{OE\_RE}$ signal deassertion time relative to start access time. It is applicable only to read accesses. $\overline{OE\_RE}$ is not asserted during a write cycle.

OEONTIME, OEOFFTIME, OEAADMUXONTIME, and OEAADMUXOFFTIME parameters are applicable to synchronous and asynchronous modes. OEONTIME can be used to control an address and byte enable valid setup time control before $\overline{OE\_RE}$ assertion. OEOFFTIME can be used to control an address and byte enable valid hold time control after $\overline{OE\_RE}$ assertion.

OEAADMUXONTIME and OEAADMUXOFFTIME parameters have the same functions as OEONTIME and OEOFFTIME, but apply to the first OE assertion in the AAD-multiplexed protocol for a read phase, or to the only OE assertion for a write phase. It is the user responsibility to make sure OEAADMUXOFFTIME is programmed to a value lower than OEONTIME. Functionality in AAD-mux mode is undefined if the settings do not comply with this requirement. OEAADMUXOFFTIME shall never be equal to OEONTIME because the AAD-mux protocol requires a second address phase with the $\overline{OE}$ signal de-asserted before $\overline{OE}$ can be asserted again to define a read command.

The $\overline{OE\_RE}$ signal transitions as controlled through OEONTIME, OEOFFTIME, OEAADMUXONTIME, and OEAADMUXOFFTIME can be delayed by half a GPMC_FCLK period by enabling the GPMC_CONFIG4_i OEEXTRADELAY bit. This half of a GPMC_FCLK period provides more granularity on $\overline{OE\_RE}$ assertion and deassertion time to assure proper setup and hold time relative to GPMC_CLK. If enabled, OEEXTRADELAY applies to all parameters controlling $\overline{OE\_RE}$ transitions.

OEEXTRADELAY must be used carefully, to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program RDCYCLETIME and WRCYCLETIME to be greater than $\overline{OE\_RE}$ signal-deassertion time, including the extra half-GPMC_FCLK-period delay.

When the GPMC generates a read access to an address-/data-multiplexed device, it drives the address bus until $\overline{OE}$ assertion time.

9.2.4.9.5 $\overline{WE}$: Write Enable Signal Control Assertion / Deassertion Time (WEONTIME / WEOFFTIME / WEEXTRADELAY)

The GPMC_CONFIG4_i WEONTIME field (where i = 0 to 3) defines the $\overline{WE}$ signal-assertion time relative to start access time. The GPMC_CONFIG4_i WEOFFTIME field defines the $\overline{WE}$ signal-deassertion time relative to start access time. These bit fields only apply to write accesses. $\overline{WE}$ is not asserted during a read cycle.

WEONTIME can be used to control an address and byte enable valid setup time control before $\overline{WE}$ assertion. WEOFFTIME can be used to control an address and byte enable valid hold time control after $\overline{WE}$ assertion.

$\overline{WE}$ signal transitions as controlled through WEONTIME, and WEOFFTIME can be delayed by half a GPMC_FCLK period by enabling the GPMC_CONFIG4_i WEEXTRADELAY bit. This half of a GPMC_FCLK period provides more granularity on $\overline{WE}$ assertion and deassertion time to guaranty proper setup and hold time relative to GPMC_CLK. If enabled, WEEXTRADELAY applies to all parameters controlling $\overline{WE}$ transitions.

The WEEXTRADELAY bit must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the WRCYCLETIME bit field to be greater than the $\overline{WE}$ signal-deassertion time, including the extra half-GPMC_FCLK-period delay.

9.2.4.9.6 GPMC_CLK

GPMC_CLK is the external clock provided to the attached synchronous memory or device.

- The GPMC_CLK clock frequency is the GPMC_FCLK functional clock frequency divided by 1, 2, 3, or 4, depending on the GPMC_CONFIG1_i GPMCFCLKDIVIDER field (where i = 0 to 5), with an assured 50-percent duty cycle.
- The GPMC_CLK clock is only activated when the access in progress is defined as synchronous (read or write access).
- The GPMC_CONFIG1_i CLKACTIVATIONTIME field (i = 0 to 5) defines the number of GPMC_FCLK cycles from start access time to GPMC_CLK activation.
- The GPMC_CLK clock is stopped when cycle time completes and is asserted low between accesses.
- The GPMC_CLK clock is kept low when access is defined as asynchronous.

When cycle time completes, the GPMC_CLK may be high because of the GPMCFCLKDIVIDER field. To ensure correct stoppage of the GPMC_CLK clock within the 50-percent required duty cycle, it is the user's responsibility to extend the RDCYCLETIME or WRCYCLETIME value.

To ensure a correct external clock cycle, the following rules must be applied:

- (RDCYCLETIME - CLKACTIVATIONTIME) must be a multiple of (GPMCFCLKDIVIDER + 1).
- The PAGEBURSTACCESSTIME value must be a multiple of (GPMCFCLKDIVIDER + 1).

9.2.4.9.7 GPMC_CLK and Control Signals Setup and Hold

Control-signal transition (assertion and deassertion) setup and hold values with respect to the GPMC_CLK edge can be controlled in the following ways:

- For the GPMC_CLK signal, the GPMC_CONFIG1_i CLKACTIVATIONTIME field (i = 0 to 5) allows setup and hold control of control-signal assertion time.
- The use of a divided GPMC_CLK allows setup and hold control of control-signal assertion and deassertion times.
- When GPMC_CLK runs at the GPMC_FCLK frequency so that GPMC_CLK edge and control-signal transitions refer to the same GPMC_FCLK edge, the control-signal transitions can be delayed by half of a GPMC_FCLK period to provide minimum setup and hold times. This half-GPMC_FCLK delay is enabled with the CSEXTRADELAY, ADVEXTRADELAY, OEEXTRADELAY, or WEEXTRADELAY parameter. This delay must be used carefully to prevent control-signal overlap between successive accesses to different chip-selects. This implies that the RDCYCLETIME and WRCYCLETIME are greater than the last control-signal deassertion time, including the extra half-GPMC_FCLK cycle.

9.2.4.9.8 Access Time (RDACCESSTIME / WRACCESSTIME)

The read access time and write access time durations can be programmed independently through GPMC_CONFIG5_i RDACCESSTIME and GPMC_CONFIG6_i WRACCESSTIME fields (i = 0 to 5). This allows $\overline{OE}$ and GPMC data capture timing parameters to be independent of $\overline{WE}$ and memory device data capture timing parameters. RDACCESSTIME and WRACCESSTIME fields can be set with a granularity of 1 or 2 through GPMC_CONFIG1_i TIMEPARAGRANULARITY.

9.2.4.9.8.1 Access Time on Read Access

In asynchronous read mode, for single and paged accesses, GPMC_CONFIG5_i RDACCESSTIME field (i = 0 to 5) defines the number of GPMC_FCLK cycles from start access time to the GPMC_FCLK rising edge used for the first data capture. RDACCESSTIME must be programmed to the rounded greater value (in GPMC_FCLK cycles) of the read access time of the attached memory device.

In synchronous read mode, for single or burst accesses, RDACCESSTIME defines the number of GPMC_FCLK cycles from start access time to the GPMC_FCLK rising edge corresponding to the GPMC_CLK rising edge used for the first data capture.

GPMC_CLK, which is sent to the memory device for synchronization with the GPMC controller, is internally timed to correctly latch the returned data. GPMC_CONFIG5_i RDCYCLETIME must be greater than RDACCESSTIME in order to let the GPMC latch the last return data using the internally timed GPMC_CLK.

The external WAIT signal can be used in conjunction with RDACCESSTIME to control the effective GPMC data-capture GPMC_FCLK edge on read access in both asynchronous mode and synchronous mode. For details about wait monitoring, see Section 9.2.4.8.1.

9.2.4.9.8.2 Access Time on Write Access

In asynchronous write mode, the GPMC_CONFIG6_i WRACCESSTIME timing parameter is not used to define the effective write access time. Instead, it is used as a WAIT invalid timing window, and must be set to a correct value so that the gpmc_wait pin is at a valid state two GPMC_CLK cycles before WRACCESSTIME completes. For details about wait monitoring, see Section 9.2.4.8.1.

In synchronous write mode , for single or burst accesses, WRACCESSTIME defines the number of GPMC_FCLK cycles from start access time to the GPMC_CLK rising edge used by the memory device for the first data capture.

The external WAIT signal can be used in conjunction with WRACCESSTIME to control the effective memory device data capture GPMC_CLK edge for a synchronous write access. For details about wait monitoring, see Section 9.2.4.8.1.

9.2.4.9.9 Page Burst Access Time (PAGEBURSTACCESSTIME)

GPMC_CONFIG5_i PAGEBURSTACCESSTIME field (i = 0 to 5) can be set with a granularity of 1 or 2 through the GPMC_CONFIG1_i TIMEPARAGRANULARITY.

9.2.4.9.9.1 Page Burst Access Time on Read Access

In asynchronous page read mode, the delay between successive word captures in a page is controlled through the GPMC_CONFIG5_i PAGEBURSTACCESSTIME field. The PAGEBURSTACCESSTIME parameter must be programmed to the rounded greater value (in GPMC_FCLK cycles) of the read access time of the attached device.

In synchronous burst read mode, the delay between successive word captures in a burst is controlled through the PAGEBURSTACCESSTIME field.

The external WAIT signal can be used in conjunction with PAGEBURSTACCESSTIME to control the effective GPMC data capture GPMC_FCLK edge on read access. For details about wait monitoring, see Section 9.2.4.8.1.

9.2.4.9.9.2 Page Burst Access Time on Write Access

Asynchronous page write mode is not supported. GPMC_CONFIG5_i PAGEBURSTACCESSTIME is irrelevant in this case.

In synchronous burst write mode, PAGEBURSTACCESSTIME controls the delay between successive memory device word captures in a burst.

The external WAIT signal can be used in conjunction with PAGEBURSTACCESSTIME to control the effective memory-device data capture GPMC_CLK edge in synchronous write mode. For details about wait monitoring, see Section 9.2.4.8.1.

9.2.4.9.10 Bus Keeping Support

At the end-cycle time of a read access, if no other access is pending, the GPMC drives the bus with the last data read after RDCYCLETIME completion time to prevent bus floating and reduce power consumption.

After a write access, if no other access is pending, the GPMC keeps driving the data bus after WRCYCLETIME completes with the same data to prevent bus floating and power consumption.

9.2.4.10 NOR Access Description

For each chip-select configuration, the read access can be specified as either asynchronous or synchronous access through the GPMC_CONFIG1_i READTYPE bit (i = 0 to 5). For each chip-select configuration, the write access can be specified as either synchronous or asynchronous access through the GPMC_CONFIG1_i WRITETYPE bit (i = 0 to 5).

Asynchronous and synchronous read and write access time and related control signals are controlled through timing parameters that refer to GPMC_FCLK. The primary difference of synchronous mode is the availability of a configurable clock interface (GPMC_CLK) to control the external device. Synchronous mode also affects data-capture and wait-pin monitoring schemes in read access.

For details about asynchronous and synchronous access, see the descriptions of GPMC_CLK, RdAccessTime, WrAccessTime, and wait-pin monitoring.

For more information about timing-parameter settings, see the sample timing diagrams in this chapter.

The address bus and $\overline{\text{BE}}$[1:0] are fixed for the duration of a synchronous burst read access, but they are updated for each beat of an asynchronous page-read access.

9.2.4.10.1 Asynchronous Access Description

This section describes:

- Asynchronous single read operation on an address/data multiplexed device
- Asynchronous single write operation on an address/data-multiplexed device
- Asynchronous single read operation on an AAD-multiplexed device
- Asynchronous single write operation on an AAD-multiplexed device
- Asynchronous multiple (page) read operation on a non-multiplexed device

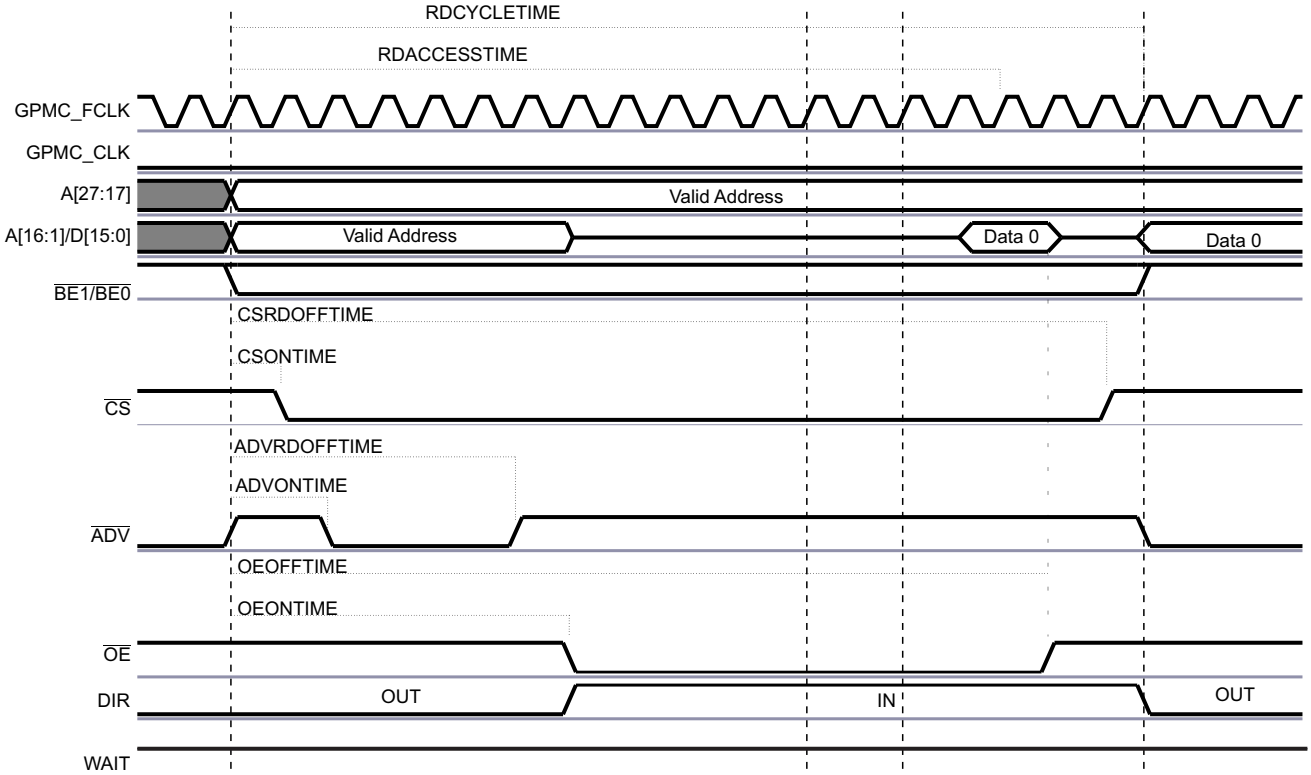
In asynchronous operations GPMC_CLK is not provided outside the GPMC and is kept low.

9.2.4.10.1.1 Access on Address/Data Multiplexed Devices

9.2.4.10.1.1.1 Asynchronous Single-Read Operation on an Address/Data Multiplexed Device

Figure 9-12 shows an asynchronous single read operation on an address/data-multiplexed device.

Figure 9-12. Asynchronous Single Read Operation on an Address/Data Multiplexed Device



9.2.4.10.1.1.2 Asynchronous Single Read on an Address/Data-Multiplexed Device

See Section 9.3.6.1 for formulas to calculate timing parameters.

Table 9-41 lists the timing bit fields to set up in order to configure the GPMC in asynchronous single read mode.

When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until $\overline{OE}$ assertion time. For details, see Section 9.2.4.8.2.3.

Address bits (A[16:1] from a GPMC perspective, A[15:0] from an external device perspective) are placed on the address/data bus, and the remaining address bits GPMC_A[25:16] are placed on the address bus. The address phase ends at $\overline{OE}$ assertion, when the DIR signal goes from OUT to IN.

- Chip-select signal $\overline{CS}$
  - $\overline{CS}$ assertion time is controlled by the GPMC_CONFIG2_i CSONTIME field. It controls the address setup time to $\overline{CS}$ assertion.
  - $\overline{CS}$ deassertion time is controlled by the GPMC_CONFIG2_i CSRDOFFTIME field. It controls the address hold time from $\overline{CS}$ deassertion
- Address valid signal $\overline{ADV}$
  - $\overline{ADV}$ assertion time is controlled by the GPMC_CONFIG3_i ADVONTIME field.
  - $\overline{ADV}$ deassertion time is controlled by the GPMC_CONFIG3_i ADVRDOFFTIME field.

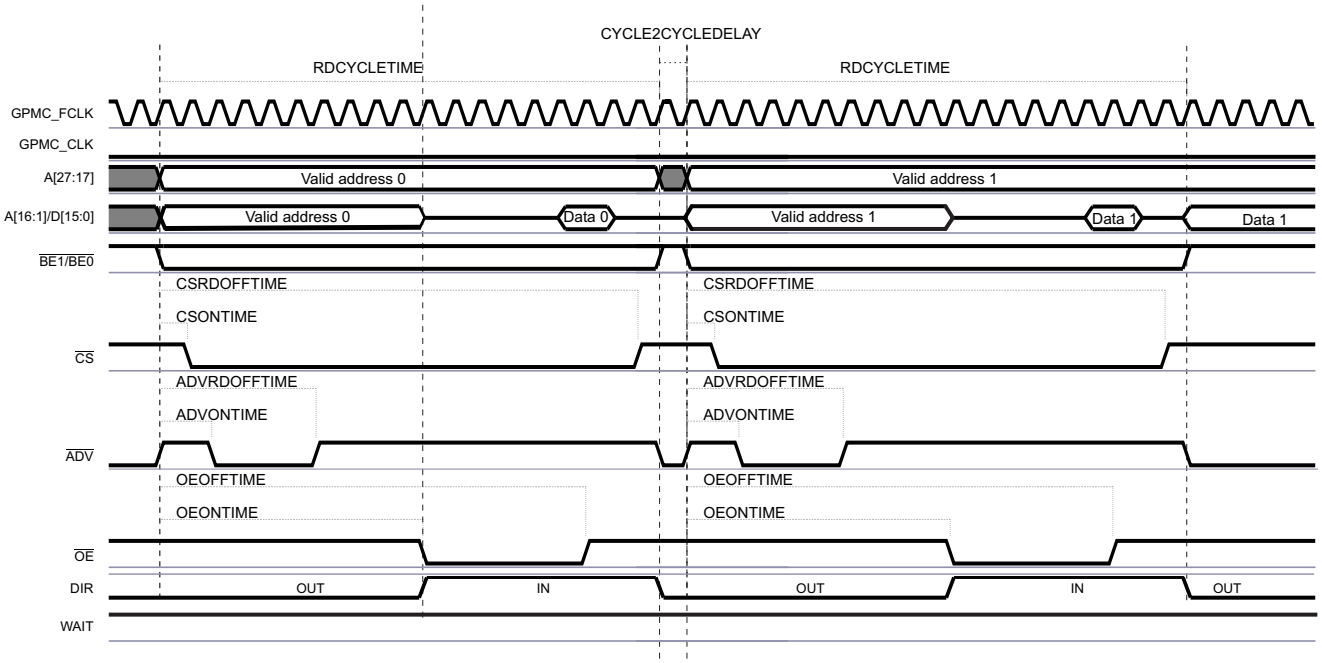
- Output enable signal $\overline{OE}$
  - $\overline{OE}$ assertion indicates a read cycle.
  - $\overline{OE}$ assertion time is controlled by the GPMC_CONFIG4_i OEONTIME field.
  - $\overline{OE}$ deassertion time is controlled by the GPMC_CONFIG4_i OEOFFTIME field.
- Read data is latched when RDACCESSTIME completes. Access time is defined in the GPMC_CONFIG5_i RDACCESSTIME field.
- Direction signal DIR: DIR goes from OUT to IN at the same time that $\overline{OE}$ is asserted.
- The end of the access is defined by the GPMC_CONFIG5_i RDCYCLETIME parameter.

In the GPMC, when a 16-bit wide device is attached to the controller, a 32-bit word write access is split into two 16-bit word write accesses. For more information about GPMC access size and type adaptation, see Section 9.2.4.10.5. Between two successive accesses, if an $\overline{CS}$ pulse is needed:

- The GPMC_CONFIG6_i CYCLE2CYCLEDELAY field can be programmed with GPMC_CONFIG6_i CYCLE2CYCLESAMECSEN bit enabled.
- The CSWROFFTIME and CSONTIME parameters also allow a chip-select pulse, but this affects all other types of access.

Figure 9-13 shows two asynchronous single-read accesses on an address/data-multiplexed devices.

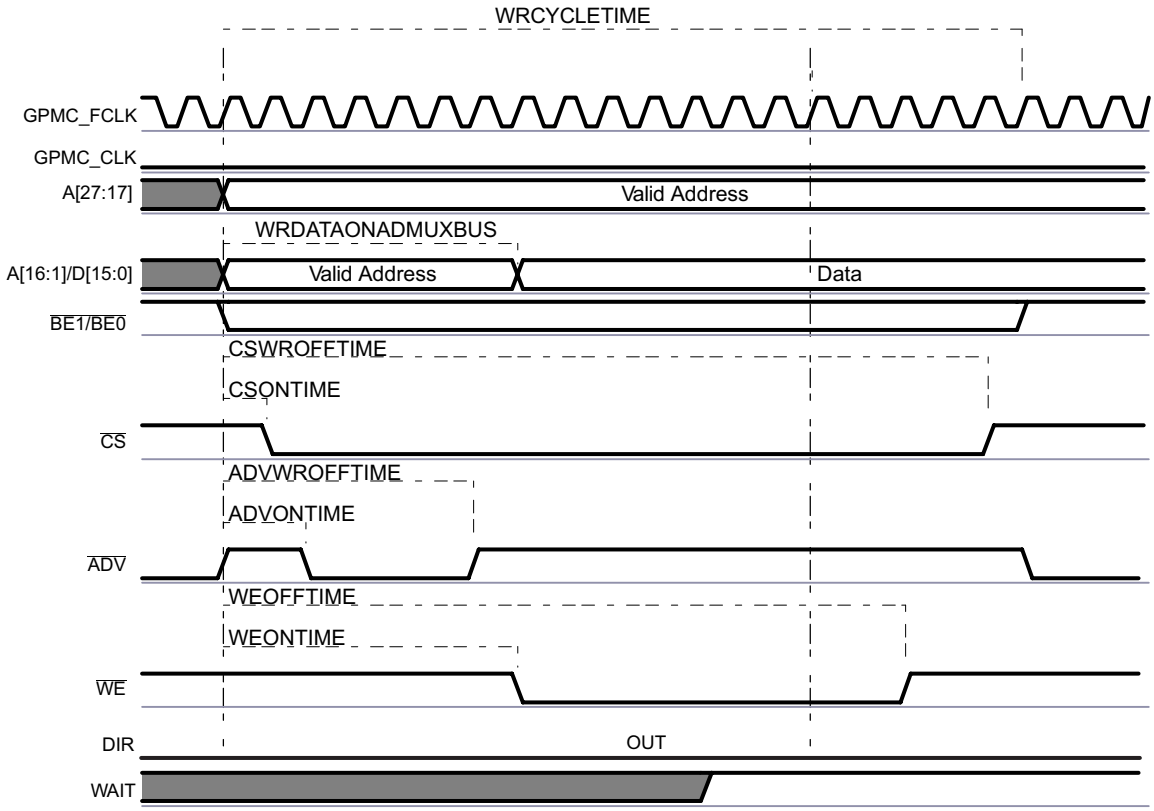
Figure 9-13. Two Asynchronous Single Read Accesses on an Address/Data Multiplexed Device (32-Bit Read Split Into 2 × 16-Bit Read)



9.2.4.10.1.1.3 Asynchronous Single Write Operation on an Address/Data-Multiplexed Device

Figure 9-14 shows an asynchronous single write operation on an address/data-multiplexed device.

Figure 9-14. Asynchronous Single Write on an Address/Data-Multiplexed Device



9.2.4.10.1.1.4 Asynchronous Single Write on an Address/Data-Multiplexed Device

See Section 9.3.6.1 for formulas to calculate timing parameters.

Table 9-41 lists the timing bit fields to set up in order to configure the GPMC in asynchronous single write mode. When the GPMC generates a write access to an address/data-multiplexed device, it drives the address bus until $\overline{WE}$ assertion time. For more information, see Section 9.2.4.8.2.3.

The $\overline{CS}$ and $\overline{ADV}$ signals are controlled in the same way as for asynchronous single read operation on an address/data-multiplexed device.

- Write enable signal $\overline{WE}$
  - $\overline{WE}$ assertion indicates a write cycle.
  - $\overline{WE}$ assertion time is controlled by the GPMC_CONFIG4_i WEONTIME field.
  - $\overline{WE}$ deassertion time is controlled by the GPMC_CONFIG4_i WEOFFTIME field.
- Direction signal DIR: DIR signal is OUT during the entire access.
- The end of the access is defined by the GPMC_CONFIG5_i WRCYCLETIME parameter.

Address bits A[16:1] (GPMC point of view) are placed on the address/data bus at the start of cycle time, and the remaining address bits A[26:17] are placed on the address bus.

Data is driven on the address/data bus at a GPMC_CONFIG6_i WRDATAONADMUXBUS time.

Write multiple access in asynchronous mode is not supported. If WRITEMULTIPLE is enabled with WRITETYPE as asynchronous, the GPMC processes single asynchronous accesses.

After a write operation, if no other access (read or write) is pending, the data bus keeps its previous value. See Section 9.2.4.9.10.

9.2.4.10.1.1.5 Asynchronous Multiple (Page) Write Operation on an Address/Data-Multiplexed Device

Write multiple (page) access in asynchronous mode is not supported for address/data-multiplexed devices. If GPMC_CONFIG1_i WRITEMULTIPLE bit is enabled (1) with GPMC_CONFIG1_i WRITETYPE bit as asynchronous (0), the GPMC processes single asynchronous accesses.

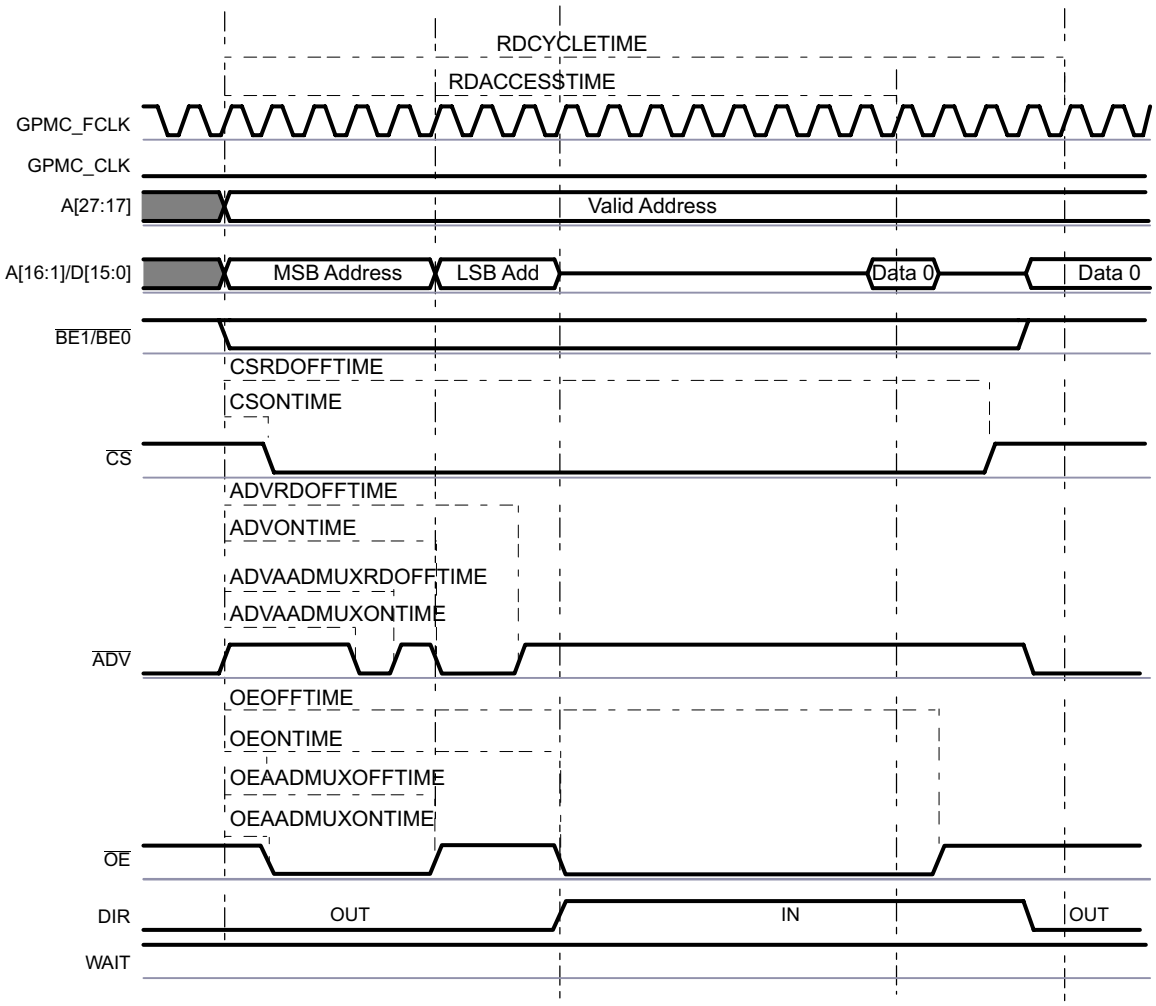
For accesses on non-multiplexed devices, see Section 9.2.4.10.3.

9.2.4.10.1.2 Access on Address/Address/Data (AAD) Multiplexed Devices

9.2.4.10.1.2.1 Asynchronous Single Read Operation on an AAD-Multiplexed Device

Figure 9-15 shows an asynchronous single read operation on an AAD-multiplexed device.

Figure 9-15. Asynchronous Single-Read on an AAD-Multiplexed Device



9.2.4.10.1.2.2 Asynchronous Single Read on an AAD-Multiplexed Device

See Section 9.3.6.1 for formulas to calculate timing parameters.

Table 9-41 lists the timing bit fields to set up in order to configure the GPMC in asynchronous single write mode.

When the GPMC generates a read access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with $\overline{OE}$ driven low. The first address phase ends at the first $\overline{OE}$ deassertion time. The second phase for LSB address is qualified with $\overline{OE}$ driven high. The second address phase ends at the second $\overline{OE}$ assertion time, when the DIR signal goes from OUT to IN.

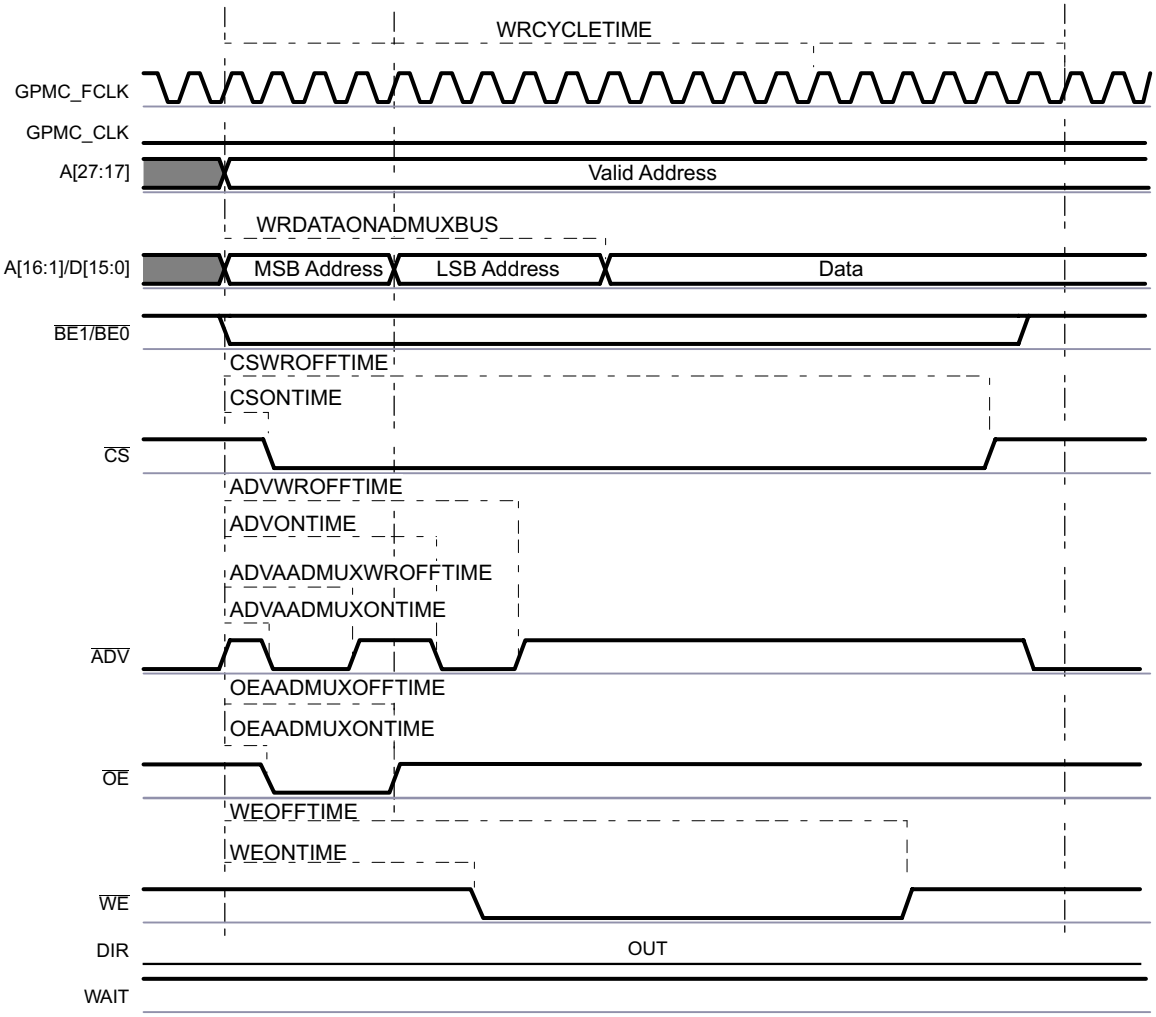
The $\overline{CS}$ and DIR signals are controlled in the same way as for asynchronous single read operation on an address/data-multiplexed device.

- Address valid signal $\overline{ADV}$. $\overline{ADV}$ is asserted and deasserted twice during a read transaction:
  - $\overline{ADV}$ first assertion time is controlled by the GPMC_CONFIG3_i ADVAADMUXONTIME field.
  - $\overline{ADV}$ first deassertion time is controlled by the GPMC_CONFIG3_i ADVAADMUXRDOFFTIME field.
  - $\overline{ADV}$ second assertion time is controlled by the GPMC_CONFIG3_i ADVONTIME field.
  - $\overline{ADV}$ second deassertion time is controlled by the GPMC_CONFIG3_i ADVRDOFFTIME field.
- Output Enable signal $\overline{OE}$. $\overline{OE}$ is asserted and deasserted twice during a read transaction ($\overline{OE}$ second assertion indicates a read cycle):
  - $\overline{OE}$ first assertion time is controlled by the GPMC_CONFIG4_i OEAADMUXONTIME field.
  - $\overline{OE}$ first deassertion time is controlled by the GPMC_CONFIG4_i OEAADMUXOFFTIME field.
  - $\overline{OE}$ second assertion time is controlled by the GPMC_CONFIG4_i OEONTIME field.
  - $\overline{OE}$ second deassertion time is controlled by the GPMC_CONFIG4_i OEOFFTIME field.

9.2.4.10.1.2.3 Asynchronous Single Write Operation on an AAD-Multiplexed Device

Figure 9-16 shows an asynchronous single write operation on an AAD-multiplexed device.

Figure 9-16. Asynchronous Single Write on an AAD-Multiplexed Device



See Section 9.3.6.1 for formulas to calculate timing parameters.

Table 9-41 lists the timing bit fields to set up to configure the GPMC in asynchronous single write mode.

When the GPMC generates a write access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with $\overline{OE}$ driven low. The second phase for LSB address is qualified with $\overline{OE}$ driven high. The address phase ends at $\overline{WE}$ assertion time.

The $\overline{CS}$, $\overline{WE}$, and DIR signals are controlled in the same way as for asynchronous single write operation on an address/data-multiplexed device.

- Address valid signal $\overline{ADV}$ is asserted and deasserted twice during a write transaction
  - $\overline{ADV}$ first assertion time is controlled by the GPMC_CONFIG3_i ADVAADMUXONTIME field.
  - $\overline{ADV}$ first deassertion time is controlled by the GPMC_CONFIG3_i ADVAADMUXWROFFTIME field.
  - $\overline{ADV}$ second assertion time is controlled by the GPMC_CONFIG3_i ADVONTIME field.
  - $\overline{ADV}$ second deassertion time is controlled by the GPMC_CONFIG3_i ADVWROFFTIME field.
- Output Enable signal $\overline{OE}$ is asserted during the address phase of a write transaction
  - $\overline{OE}$ assertion time is controlled by the GPMC_CONFIG4_i OEAADMUXONTIME field.
  - $\overline{OE}$ deassertion time is controlled by the GPMC_CONFIG4_i OEAADMUXOFFTIME field.

The address bits for the first address phase are driven onto the data bus until $\overline{OE}$ deassertion. Data is driven onto the address/data bus at the clock edge defined by the GPMC_CONFIG6_i WRDATAONADMUXBUS parameter.

9.2.4.10.1.2.4 Asynchronous Multiple (Page) Read Operation on an AAD-Multiplexed Device

Write multiple (page) access in asynchronous mode is not supported for AAD-multiplexed devices.

If GPMC_CONFIG1_i WRITEMULTIPLE bit is enabled (1) with GPMC_CONFIG1_i WRITETYPE bit as asynchronous (0), the GPMC processes single asynchronous accesses.

For accesses on non-multiplexed devices, see Section 9.2.4.10.3.

9.2.4.10.2 Synchronous Access Description

This section details read and write synchronous accesses on address/data multiplexed. All information in this section can be applied to any type of memory - non-multiplexed, address and data multiplexed or AAD-multiplexed - with a difference limited to the address phase. For accesses on non-multiplexed devices, see Section 9.2.4.10.3.

In synchronous operations:

- The GPMC_CLK clock is provided outside the GPMC when accessing the memory device.
- The GPMC_CLK clock is derived from the GPMC_FCLK clock using the GPMC_CONFIG1_i GPMCFCLKDIVIDER field. In the following section, i stands for the chip-select number, i = 0 to 3.
- The GPMC_CONFIG1_i CLKACTIVATIONTIME field specifies that the GPMC_CLK is provided outside the GPMC 0, 1, or 2 GPMC_FCLK cycles after start access time until RDCYCLETIME or WRCYCLETIME completion.

9.2.4.10.2.1 Synchronous Single Read

Figure 9-17 and Figure 9-18 show a synchronous single-read operation with GPMCFCLKDIVIDER equal to 0 and 1, respectively.

Figure 9-17. Synchronous Single Read (GPMCFCLKDIVIDER = 0)

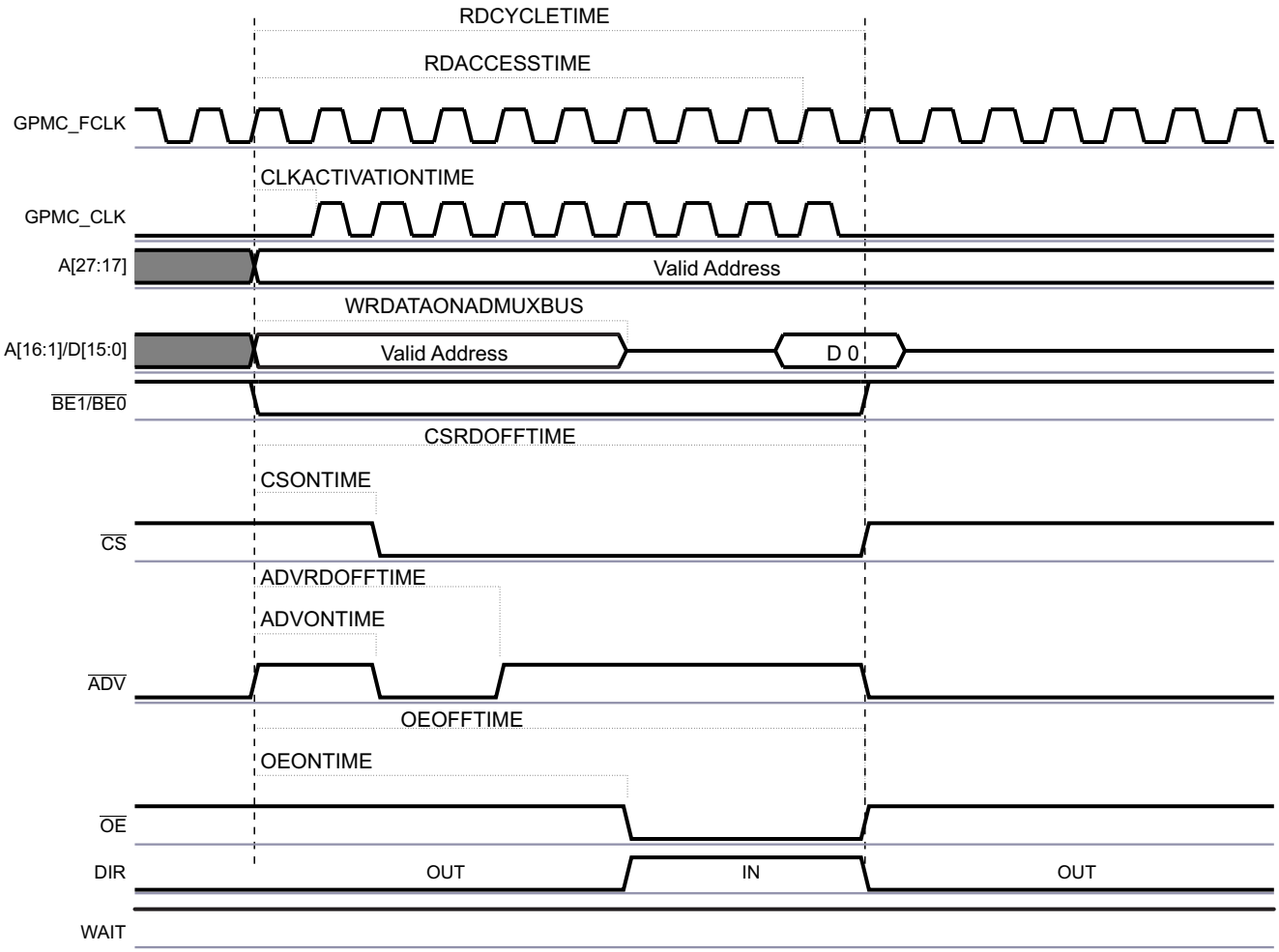
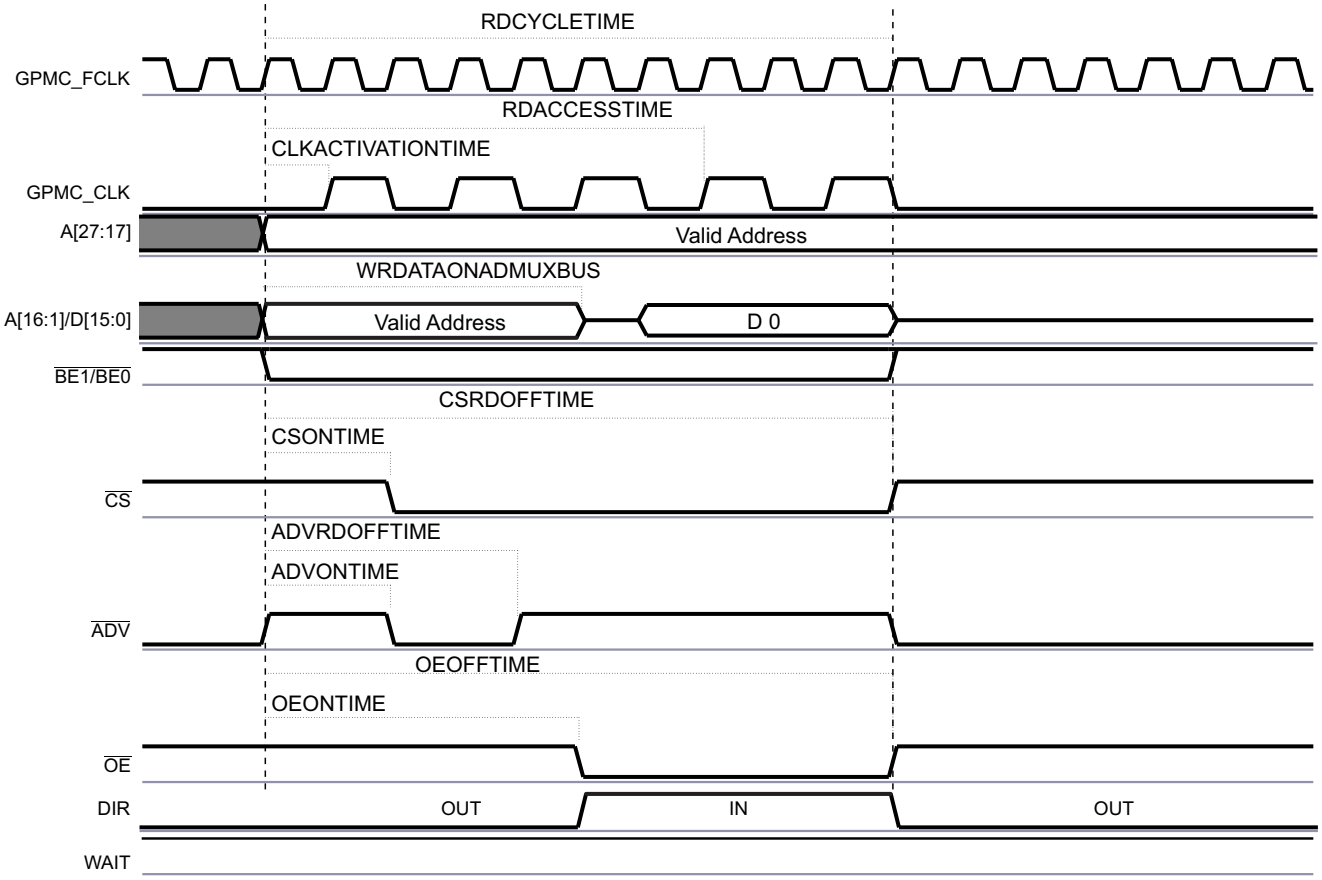


Figure 9-18. Synchronous Single Read (GPMCFCLKDIVIDER = 1)



See Section 9.3.6.1 for formulas to calculate timing parameters.

Table 9-41 lists the timing bit fields to set up in order to configure the GPMC in asynchronous single read mode.

When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until $\overline{OE}$ assertion time. For details, see Section 9.2.4.8.2.3.

- Chip-select signal $\overline{CS}$
  - $\overline{CS}$ assertion time is controlled by the GPMC_CONFIG2_i CSONTIME field and ensures address setup time to $\overline{CS}$ assertion.
  - $\overline{CS}$ deassertion time is controlled by the GPMC_CONFIG2_i CSRDOFFTIME field and ensures address hold time to $\overline{CS}$ deassertion.
- Address valid signal $\overline{ADV}$
  - $\overline{ADV}$ assertion time is controlled by the GPMC_CONFIG3_i ADVONTIME field.
  - $\overline{ADV}$ deassertion time is controlled by the GPMC_CONFIG3_i ADVRDOFFTIME field.
- Output enable signal $\overline{OE}$
  - $\overline{OE}$ assertion indicates a read cycle.
  - $\overline{OE}$ assertion time is controlled by the GPMC_CONFIG4_i OEONTIME field.
  - $\overline{OE}$ deassertion time is controlled by the GPMC_CONFIG4_i OEOFFTIME field.
- Initial latency for the first read data is controlled by GPMC_CONFIG5_i RDACCESSTIME field or by monitoring the WAIT signal.
- Total access time (GPMC_CONFIG5_i RDCYCLETIME) corresponds to RDACCESSTIME plus the address hold time from $\overline{CS}$ deassertion, plus time from RDACCESSTIME to CSRDOFFTIME.

- Direction signal DIR: DIR goes from OUT to IN at the same time as $\overline{OE}$ assertion.

When the GPMC generates a write access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with $\overline{OE}$ driven low. The second phase for LSB address is qualified with $\overline{OE}$ driven high. The address phase ends at $\overline{WE}$ assertion time.

The $\overline{CS}$ and DIR signals are controlled in the same way as for synchronous single read operation on an address/data-multiplexed device.

- Address valid signal $\overline{ADV}$ is asserted and deasserted twice during a read transaction
  - $\overline{ADV}$ first assertion time is controlled by the GPMC_CONFIG3_i ADVAADMUXONTIME field.
  - $\overline{ADV}$ first deassertion time is controlled by the GPMC_CONFIG3_i ADVAADMUXRDOFFTIME field.
  - $\overline{ADV}$ second assertion time is controlled by the GPMC_CONFIG3_i ADVONTIME field.
  - $\overline{ADV}$ second deassertion time is controlled by the GPMC_CONFIG3_i ADVRDOFFTIME field.
- Output Enable signal $\overline{OE}$ is asserted and deasserted twice during a read transaction ($\overline{OE}$ second assertion indicates a read cycle)
  - $\overline{OE}$ first assertion time is controlled by the GPMC_CONFIG4_i OEAADMUXONTIME field.
  - $\overline{OE}$ first deassertion time is controlled by the GPMC_CONFIG4_i OEAADMUXOFFTIME field.
  - $\overline{OE}$ second assertion time is controlled by the GPMC_CONFIG4_i OEONTIME field.
  - $\overline{OE}$ second deassertion time is controlled by the GPMC_CONFIG4_i OEOFFTIME field.

After a read operation, if no other access (read or write) is pending, the data bus is driven with the previous read value. See Section 9.2.4.9.10.

9.2.4.10.2.2 Synchronous Multiple (Burst) Read (4-, 8-, 16-Word16 Burst With Wraparound Capability)

Figure 9-19 and Figure 9-20 show a synchronous multiple read operation with GPMCFCLKDivider equal to 0 and 1, respectively.

Figure 9-19. Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 0)

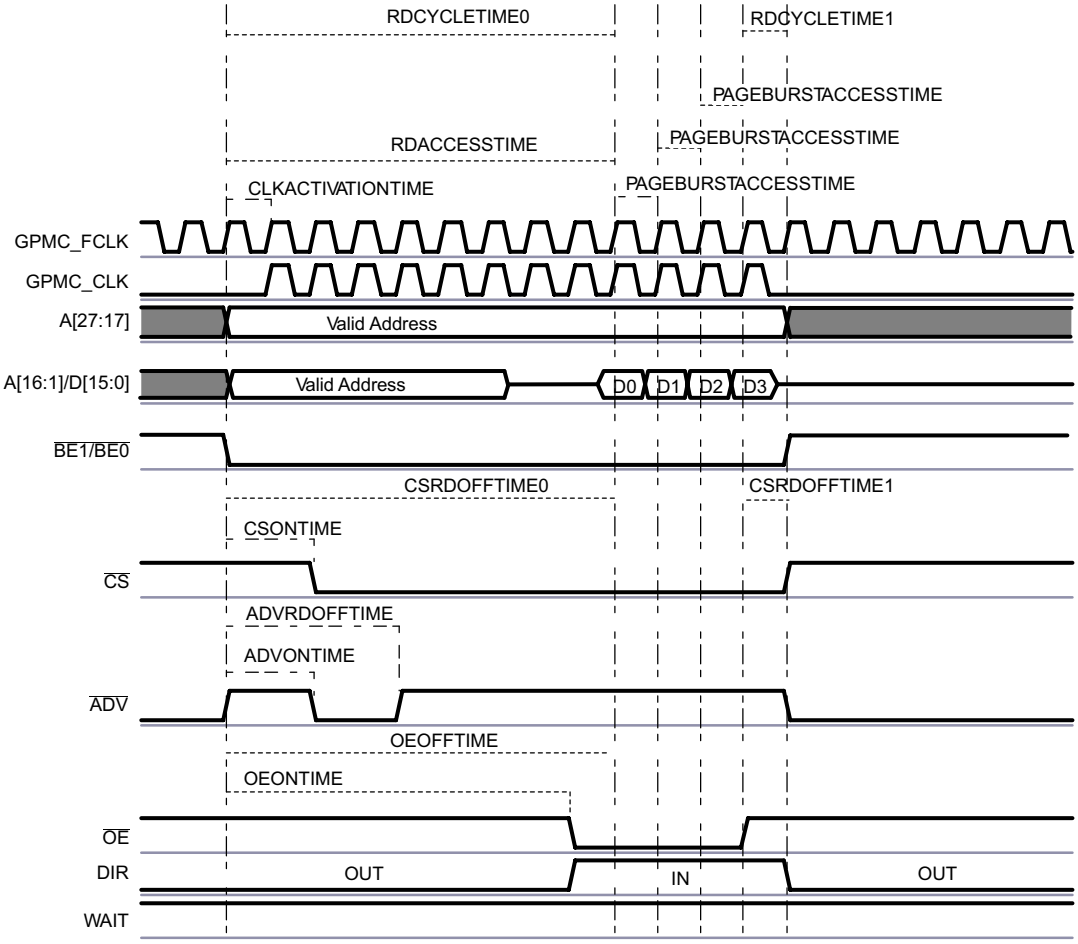
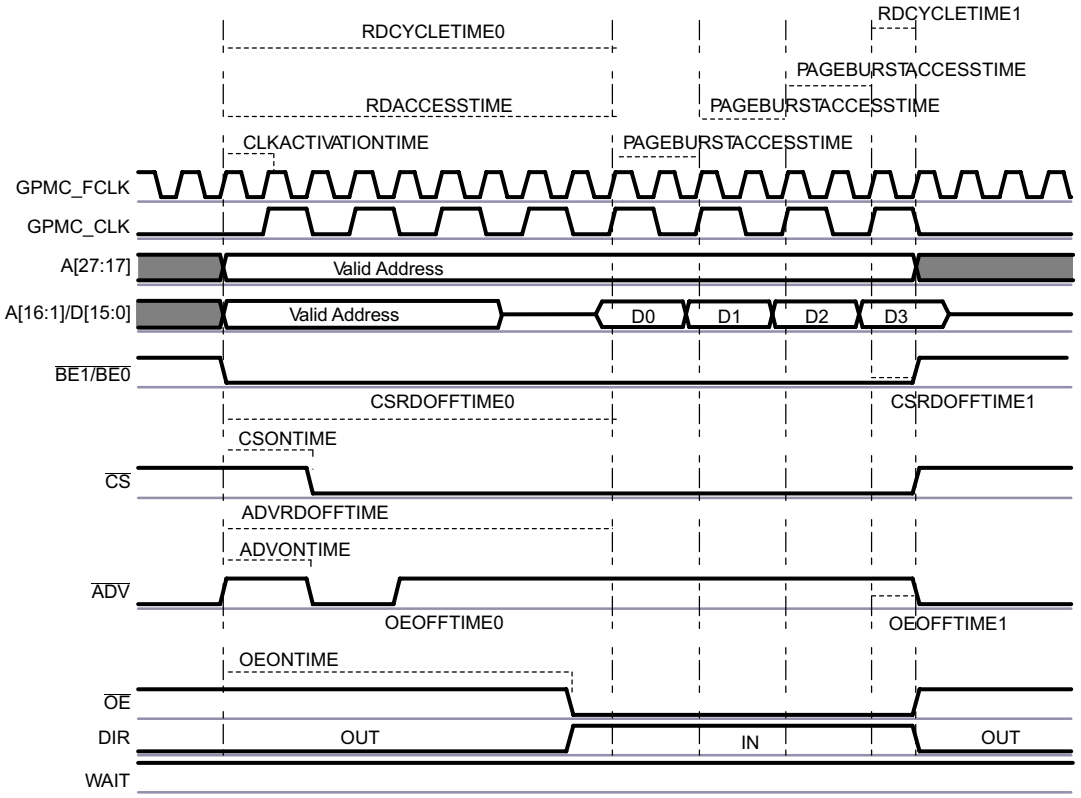


Figure 9-20. Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 1)



When GPMC_CONFIG5_i RDACCESSTIME completes, control-signal timings are frozen during the multiple data transactions, corresponding to GPMC_CONFIG5_i PAGEBURSTACCESSTIME multiplied by the number of remaining data transactions.

The $\overline{CS}$, $\overline{ADV}$, $\overline{OE}$ and DIR signals are controlled in the same way as for synchronous single read operation. See Section 9.2.4.10.2.1.

Initial latency for the first read data is controlled by RDACCESSTIME or by monitoring the WAIT signal. Successive read data are provided by the memory device each one or two GPMC_CLK cycles. The PAGEBURSTACCESSTIME parameter must be set accordingly with GPMC_CONFIG1_i GPMCFCLKDIVIDER and the memory-device internal configuration. Depending on the device page length, the GPMC checks device page crossing during a new burst request and purposely insert initial latency (of RDACCESSTIME) when required.

Total access time GPMC_CONFIG5_i RDCYCLETIME corresponds to RDACCESSTIME plus the address hold time from $\overline{CS}$ deassertion. In Figure 9-19, RDCYCLETIME programmed value equals to RDCYCLETIME0 + RDCYCLETIME1.

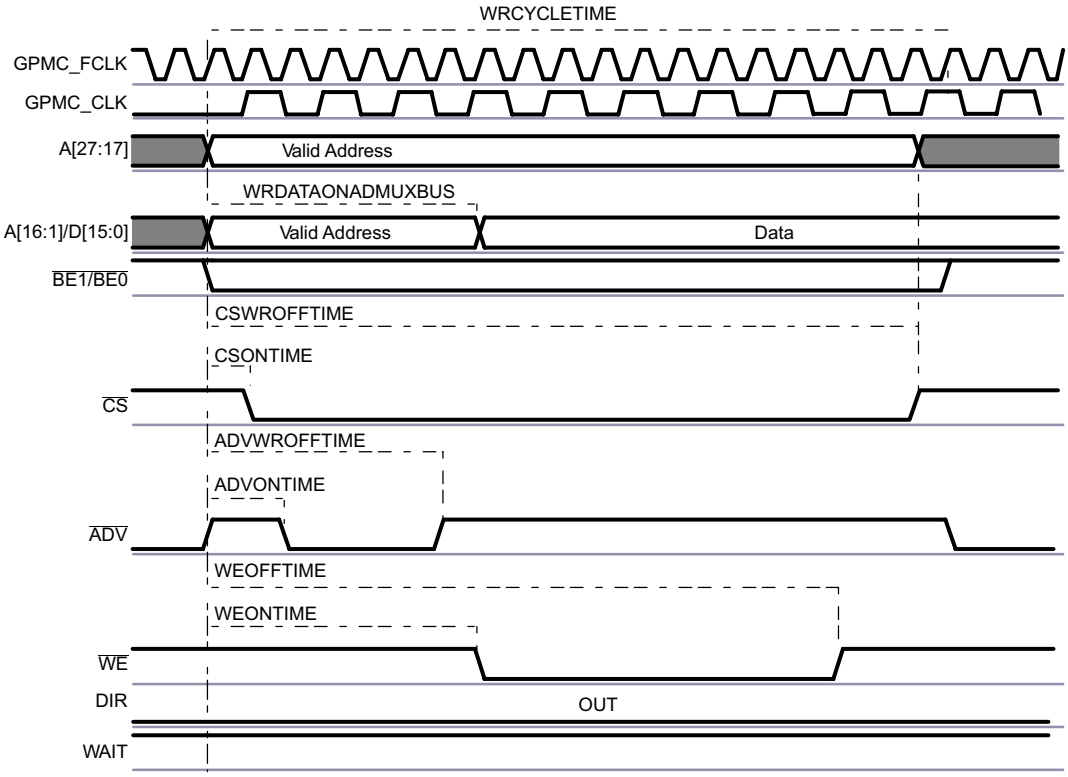
After a read operation, if no other access (read or write) is pending, the data bus is driven with the previous read value. See Section 9.2.4.9.10.

Burst wraparound is enabled through the GPMC_CONFIG1_i WRAPBURST bit and allows a 4-, 8-, or 16-Word16 linear burst access to wrap within its burst-length boundary through GPMC_CONFIG1_i ATTACHEDDEVICEPAGELENGTH.

9.2.4.10.2.3 Synchronous Single Write

Burst write mode is used for synchronous single or burst accesses (see Figure 9-21).

Figure 9-21. Synchronous Single Write on an Address/Data-Multiplexed Device



When the GPMC generates a write access to an address/data-multiplexed device, it drives the data bus (with address bits A[16:1]) until GPMC_CONFIG6_i WRDATAONADMUXBUS time. First data of the burst is driven on the address/data bus at WRDATAONADMUXBUS time.

9.2.4.10.2.4 Synchronous Multiple (Burst) Write

Synchronous burst write mode provides synchronous single or consecutive accesses. Figure 9-22 shows a synchronous burst write access when the chip-select is configured in address/data-multiplexed mode.

Figure 9-22. Synchronous Multiple Write (Burst Write) in Address/Data-Multiplexed Mode

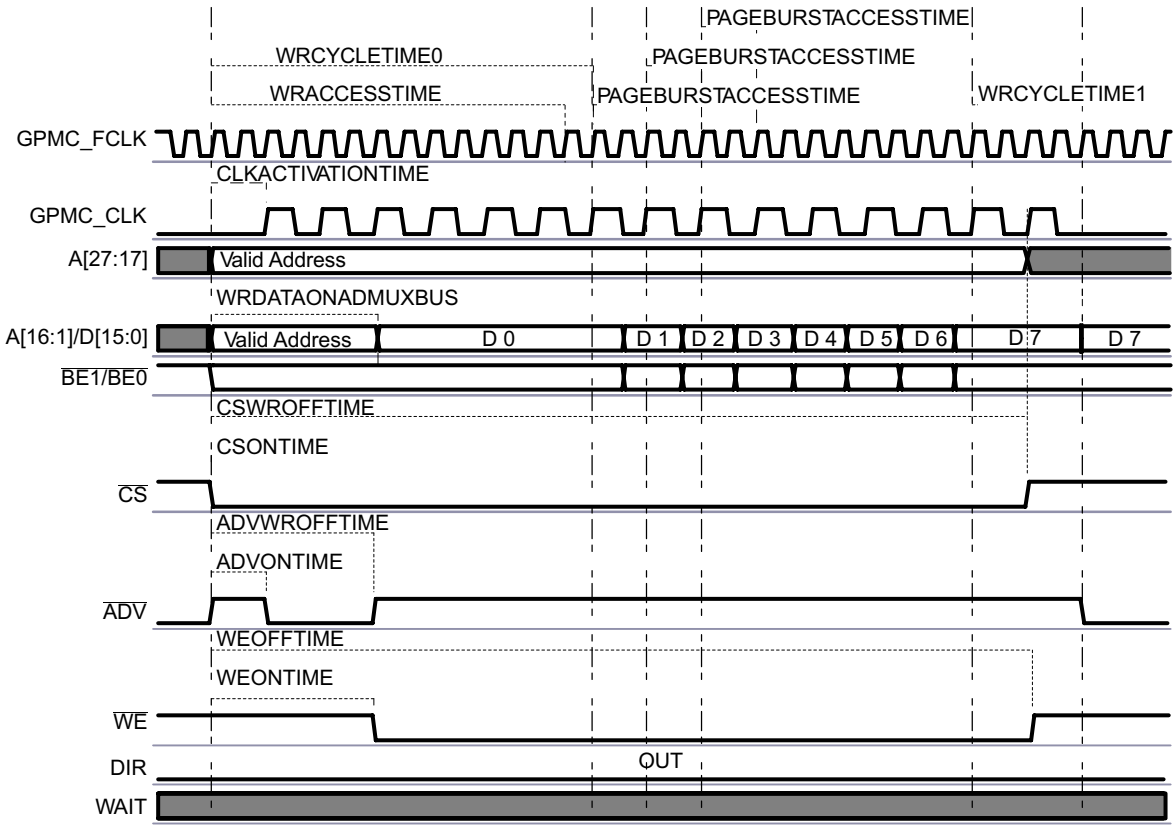
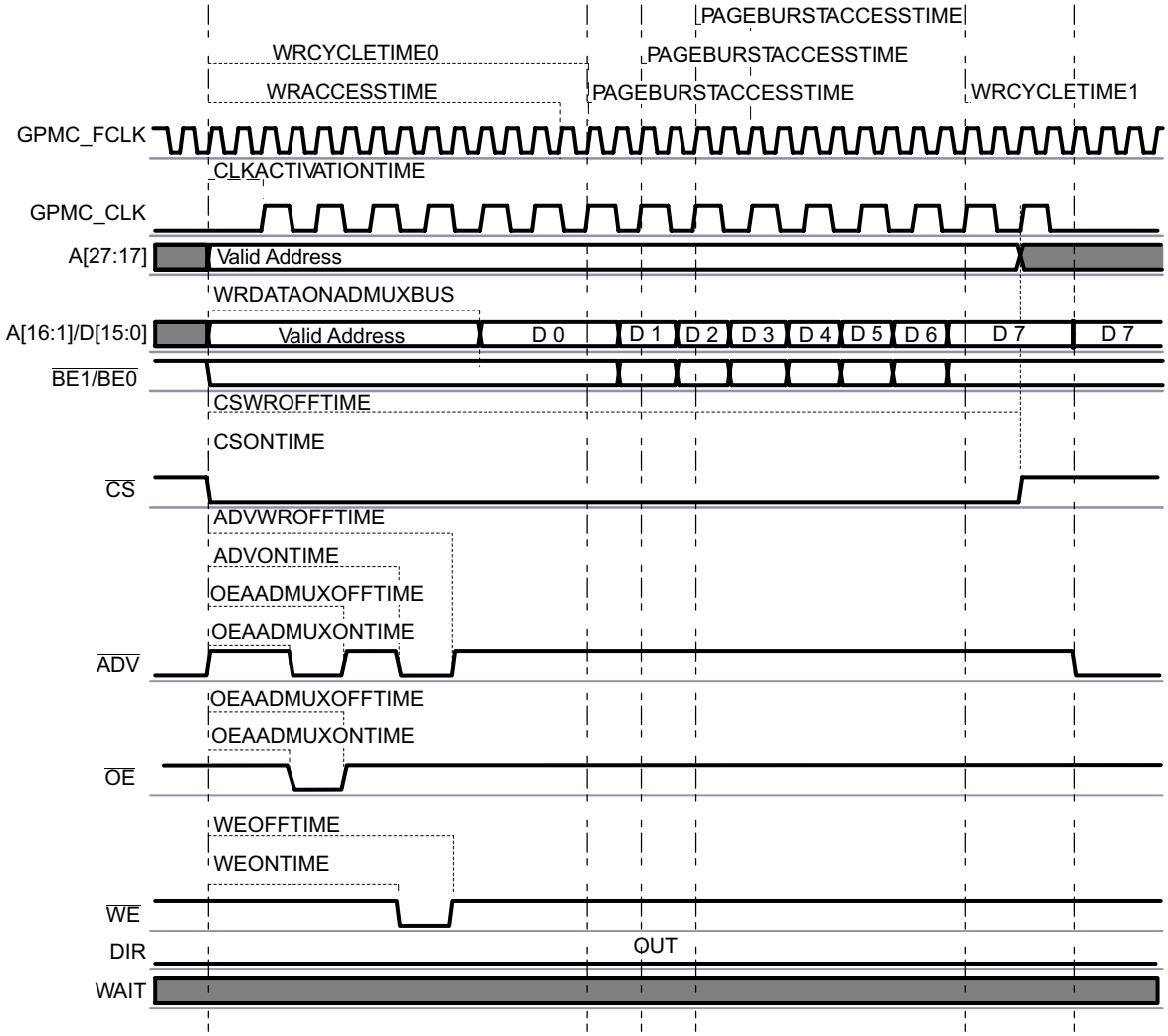


Figure 9-23 shows the same synchronous burst write access when the chip-select is configured in address/address/data-multiplexed (AAD-multiplexed) mode.

Figure 9-23. Synchronous Multiple Write (Burst Write) in Address/Address/Data-Multiplexed Mode



The first data of the burst is driven on the A/D bus at GPMC_CONFIG6_i WRDATAONADMUXBUS.

When WRACCESSTIME completes, control-signal timings are frozen during the multiple data transactions, corresponding to the GPMC_CONFIG5_i PAGEBURSTACCESSTIME multiplied by the number of remaining data transactions.

When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until $\overline{OE}$ assertion time. For details, see Section 9.2.4.8.2.3.

- Chip-select signal $\overline{CS}$
  - $\overline{CS}$ assertion time is controlled by the GPMC_CONFIG2_i CSONTIME field (i = 0 to 5) and ensures address setup time to $\overline{CS}$ assertion.
  - $\overline{CS}$ deassertion time controlled by the GPMC_CONFIG2_i CSWROFFTIME field and ensures address hold time to $\overline{CS}$ deassertion.
- Address valid signal $\overline{ADV}$
  - $\overline{ADV}$ assertion time is controlled by the GPMC_CONFIG3_i ADVONTIME field.
  - $\overline{ADV}$ deassertion time is controlled by the GPMC_CONFIG3_i ADVWROFFTIME field.

- Write enable signal $\overline{WE}$
  - $\overline{WE}$ assertion indicates a read cycle.
  - $\overline{WE}$ assertion time is controlled by the GPMC_CONFIG4_i WEONTIME field.
  - $\overline{WE}$ deassertion time is controlled by the GPMC_CONFIG4_i WEOFFTIME field.

  The $\overline{WE}$ falling edge must not be used to control the time when the burst first data is driven in the address/data bus because some new devices require the $\overline{WE}$ signal at low during the address phase.
- Direction signal DIR is OUT during the entire access.

When the GPMC generates a write access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with $\overline{OE}$ driven low. The second phase for LSB address is qualified with $\overline{OE}$ driven high. The address phase ends at $\overline{WE}$ assertion time.

The $\overline{CS}$ and DIR signals are controlled as detailed above.

- Address valid signal $\overline{ADV}$ is asserted and deasserted twice during a read transaction
  - $\overline{ADV}$ first assertion time is controlled by the GPMC_CONFIG3_i ADVAADMUXONTIME field.
  - $\overline{ADV}$ first deassertion time is controlled by the GPMC_CONFIG3_i ADVAADMUXRDOFFTIME field.
  - $\overline{ADV}$ second assertion time is controlled by the GPMC_CONFIG3_i ADVONTIME field.
  - $\overline{ADV}$ second deassertion time is controlled by the GPMC_CONFIG3_i ADVRDOFFTIME field.
- Output Enable signal $\overline{OE}$ is asserted and deasserted twice during a read transaction ($\overline{OE}$ second assertion indicates a read cycle)
  - $\overline{OE}$ first assertion time is controlled by the GPMC_CONFIG4_i OEAADMUXONTIME field.
  - $\overline{OE}$ first deassertion time is controlled by the GPMC_CONFIG4_i OEAADMUXOFFTIME field.
  - $\overline{OE}$ second assertion time is controlled by the GPMC_CONFIG4_i OEONTIME field.
  - $\overline{OE}$ second deassertion time is controlled by the GPMC_CONFIG4_i OEOFFTIME field.

First write data is driven by the GPMC at GPMC_CONFIG6_i WRDATAONADMUXBUS, when in address/data mux configuration. The next write data of the burst is driven on the bus at WRACCESSTIME + 1 during GPMC_CONFIG5_i PAGEBURSTACCESSTIME GPMC_FCLK cycles. The last data of the synchronous burst write is driven until GPMC_CONFIG5_i WRCYCLETIME completes.

- WRACCESSTIME is defined in the GPMC_CONFIG6_i register.
- The PAGEBURSTACCESSTIME parameter must be set accordingly with GPMCFCLKDIVIDER and the memory-device internal configuration.

Total access time GPMC_CONFIG5_i WRCYCLETIME corresponds to WRACCESSTIME plus the address hold time from $\overline{CS}$ deassertion. In Figure 9-23 the WRCYCLETIME programmed value equals WRCYCLETIME0 + WRCYCLETIME1. WRCYCLETIME0 and WRCYCLETIME1 delays are not actual parameters and are only a graphical representation of the full WRCYCLETIME value.

After a write operation, if no other access (read or write) is pending, the data bus keeps the previous value. See Section 9.2.4.9.10.

9.2.4.10.3 Asynchronous and Synchronous Accesses in Nonmultiplexed Mode

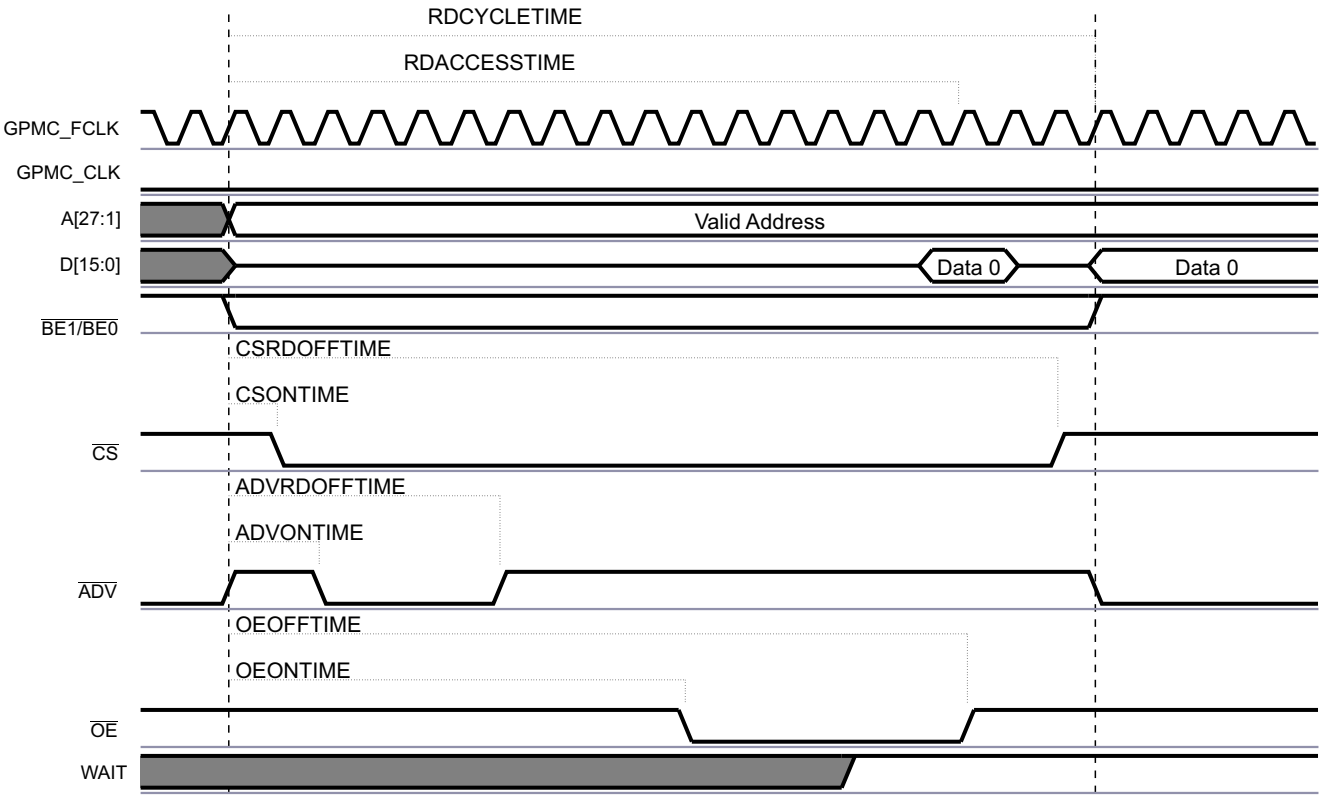
Page mode is only available in non-multiplexed mode.

- Asynchronous single read operation on a nonmultiplexed device
- Asynchronous single write operation on a nonmultiplexed device
- Asynchronous multiple (page mode) read operation on a nonmultiplexed device
- Synchronous operations on a nonmultiplexed device

9.2.4.10.3.1 Asynchronous Single Read Operation on a Nonmultiplexed Device

Figure 9-24 shows an asynchronous single read operation on a nonmultiplexed device.

Figure 9-24. Asynchronous Single Read on an Address/Data-Nonmultiplexed Device



The 27-bit address is driven onto the address bus A[27:1] and the 16-bit data is driven onto the data bus D[15:0].

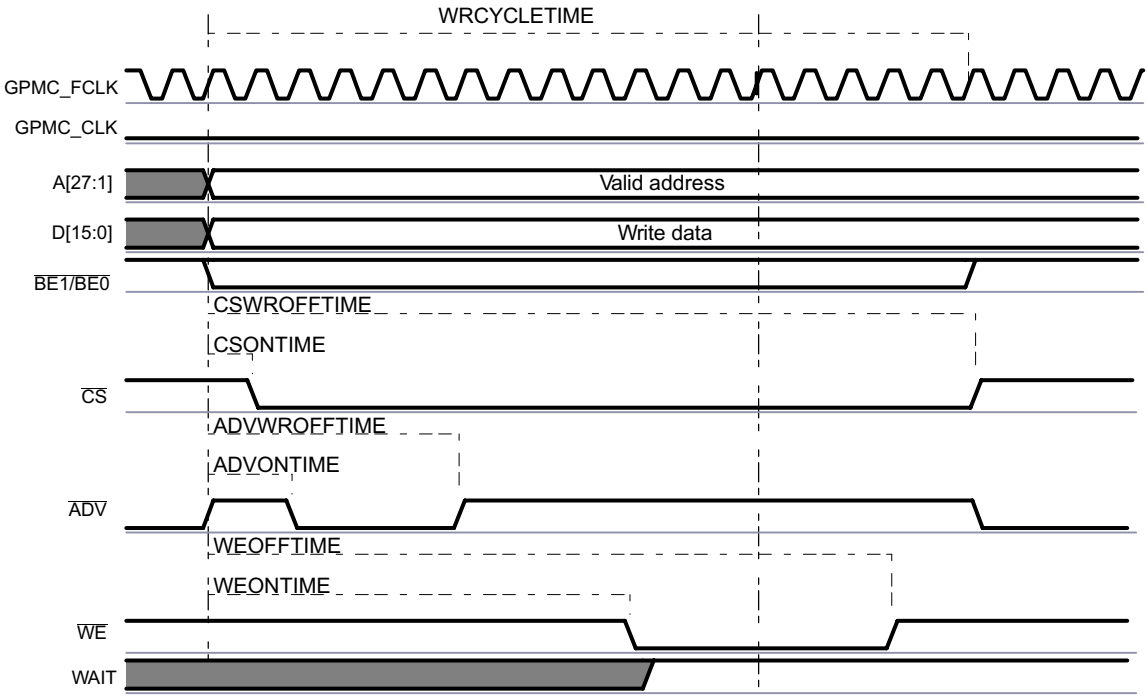
Read data is latched at GPMC_CONFIG1_5 RDACCESSTIME completion time. The end of the access is defined by the GPMC_CONFIG1_5 RDCYCLETIME parameter.

$\overline{CS}$, $\overline{ADV}$, $\overline{OE}$ and DIR signals are controlled in the same way as address/data multiplexed accesses, see Section 9.2.4.10.1.1.2.

9.2.4.10.3.2 Asynchronous Single Write Operation on a Nonmultiplexed Device

Figure 9-25 shows an asynchronous single write operation on a nonmultiplexed device.

Figure 9-25. Asynchronous Single Write on an Address/Data-Nonmultiplexed Device



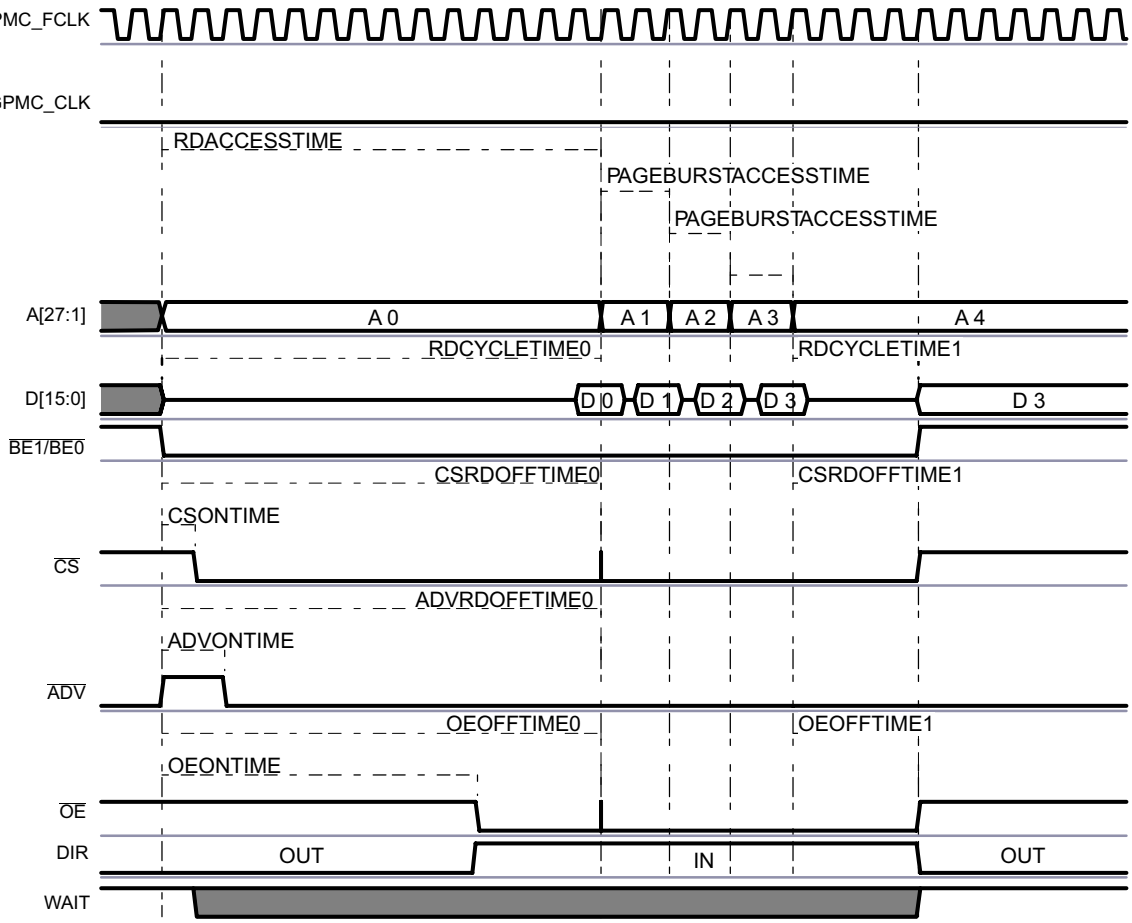
The 27-bit address is driven onto the address bus A[27:1] and the 16-bit data is driven onto the data bus D[15:0].

$\overline{CS}$, $\overline{ADV}$, $\overline{WE}$ and DIR signals are controlled in the same way as address/data multiplexed accesses, see Section 9.2.4.10.1.1.3.

9.2.4.10.3.3 Asynchronous Multiple (Page Mode) Read Operation on a Nonmultiplexed Device

Figure 9-26 shows an asynchronous multiple read operation on a Nonmultiplexed Device, in which two 32-word host read accesses to the GPMC are split into one multiple (page mode of four 16-word) read access to the attached device.

Figure 9-26. Asynchronous Multiple (Page Mode) Read



The WAIT signal is active low.

CS, ADV, OE and DIR signals are controlled in the same way as address/data multiplexed accesses, see Section 9.2.4.10.1.1.2.

When RDACCESSTIME completes, control-signal timings are frozen during the multiple data transactions, corresponding to PAGEBURSTACCESSTIME multiplied by the number of remaining data transactions.

Read data is latched at GPMC_CONFIG5_i RDACCESSTIME completion time (i = 0 to 5). The end of the access is defined by the GPMC_CONFIG5_i RDCYCLETIME parameter.

During consecutive accesses, the GPMC increments the address after each data read completes.

Delay between successive read data in the page is controlled by the GPMC_CONFIG5_i PAGEBURSTACCESSTIME parameter. Depending on the device page length, the GPMC can control device page crossing during a burst request and insert initial RDACCESSTIME latency. Note that page crossing is only possible with a new burst access, meaning a new initial access phase is initiated.

Total access time RDCYCLETIME corresponds to RDACCESSTIME plus the address hold time starting from the CS deassertion.

- The read cycle time is defined in the GPMC_CONFIG5_i RDCYCLETIME field.
- In Figure 9-26, the RDCYCLETIME programmed value equals RDCYCLETIME0 (before paged accesses) + RDCYCLETIME1 (after paged accesses).

9.2.4.10.3.4 Synchronous Operations on a Nonmultiplexed Device

All information for this section is equivalent to similar operations for address/data- or AAD-multiplexed accesses. The only difference resides in the address phase. See Section 9.3.3.

9.2.4.10.4 Page and Burst Support

Each chip-select can be configured to process system single or burst requests into successive single accesses or asynchronous page/synchronous burst accesses, with appropriate access size adaptation.

Depending on the external device page or burst capability, read and write accesses can be independently configured through the GPMC. The GPMC_CONFIG1_i READMULTIPLE and GPMC_CONFIG1_i WRITMULTIPLE bits (i = 0 to 5) are associated with the READTYPE and WRITETYPE parameters.

- Asynchronous write page mode is not supported.
- 8-bit wide device support is limited to nonburstable devices (READMULTIPLE and WRITEMULTIPLE are ignored).
- Not applicable to NAND device interfacing.

9.2.4.10.5 System Burst Versus External Device Burst Support

The device system can issue the following requests to the GPMC:

- Byte, 16-bit word, 32-bit word requests (byte enable controlled). This is always a single request from the interconnect point of view.
- Incrementing fixed-length bursts of two words, four words, and eight words
- Wrapped (critical word access first) fixed-length burst of two, four, or eight words

To process a system request with the optimal protocol, the READMULTIPLE (and READTYPE) and WRITEMULTIPLE (and WRITETYPE) parameters must be set according to the burstable capability (synchronous or asynchronous) of the attached device.

The GPMC access engine issues only fixed-length burst. The maximum length that can be issued is defined per CS by the GPMC_CONFIG1_i ATTACHEDDEVICEPAGELENGTH field (i = 0 to 5). When the ATTACHEDDEVICEPAGELENGTH value is less than the system burst request length (including the appropriate access size adaptation according to the device width), the GPMC splits the system burst request into multiple bursts. Within the specified 4-, 8-, or 16-word value, the ATTACHEDDEVICEPAGELENGTH field value must correspond to the maximum-length burst supported by the memory device configured in fixed-length burst mode (as opposed to continuous burst mode).

To get optimal performance from memory devices that natively support 16 Word16-length-wrapping burst capability (critical word access first), the ATTACHEDDEVICEPAGELENGTH parameter must be set to 16 words and the GPMC_CONFIG1_i WRAPBURST bit (i = 0 to 5) must be set to 1. Similarly, DEVICEPAGELENGTH is set to 4 and 8 for memories supporting 4 and 8 Word 16-length-wrapping burst, respectively.

When the memory device does not offer (or is not configured to offer) native 16 Word 16-length-wrapping burst, the WRAPBURST parameter must be cleared, and the GPMC access engine emulates the wrapping burst by issuing the appropriate burst sequences according to the ATTACHEDDEVICEPAGELENGTH value.

When the memory device does not support native-wrapping burst, there is usually no difference in behavior between a fixed burst length mode and a continuous burst mode configuration (except for a potential power increase from a memory-speculative data prefetch in a continuous burst read). However, even though continuous burst mode is compatible with GPMC behavior, because the GPMC access engine issues only fixed-length burst and does not benefit from continuous burst mode, it is best to configure the memory device in fixed-length burst mode.

The memory device maximum-length burst (configured in fixed-length burst wrap or nonwrap mode) usually corresponds to the memory device data buffer size. Memory devices with a minimum of 16 half-word buffers are the most appropriate (especially with wrap support), but memory devices with smaller buffer size (4 or 8) are also supported, assuming that the GPMC_CONFIG1_i ATTACHEDDEVICEPAGELENGTH field is set accordingly to 4 or 8 words.

The device system issues only requests with addresses or starting addresses for nonwrapping burst requests; that is, the request size boundary is aligned. In case of an eight-word-wrapping burst, the wrapping address always occurs on the eight-words boundary. As a consequence, all words requested must be available from the memory data buffer when the buffer size is equal to or greater than the ATTACHEDDEVICEPAGELENGTH value. This usually means that data can be read from or written to the buffer at a constant rate (number of cycles between data) without wait states between data accesses. If the memory does not behave this way (nonzero wait state burstable memory), wait-pin monitoring must be enabled to dynamically control data-access completion within the burst.

When the system burst request length is less than the ATTACHEDDEVICEPAGELENGTH value, the GPMC proceeds with the required accesses.

9.2.4.11 pSRAM Access Specificities

pSRAM devices are SRAM-pin-compatible low-power memories that contain a self-refreshed DRAM memory array. The GPMC_CONFIG1_i DEVICETYPE field (i = 0 to 3) shall be cleared to 00.

The pSRAM devices uses the NOR protocol. It support the following operations:

- Asynchronous single read
- Asynchronous page read
- Asynchronous single write
- Synchronous single read and write
- Synchronous burst read
- Synchronous burst write (not supported by NOR Flash memory)

pSRAM devices must be powered up and initialized in a predefined manner according to the specifications of the attached device.

pSRAM devices can be programmed to use either mode: fixed or variable latency. pSRAM devices can either automatically schedule autorefresh operations, which force the GPMC to use its WAIT signal capability when read or write operations occur during an internal self-refresh operation, or pSRAM devices automatically include the autorefresh operation in the access time. These devices do not require additional WAIT signal capability or a minimum $\overline{CS}$ high pulse width between consecutive accesses to ensure that the correct internal refresh operation is scheduled.

9.2.4.12 NAND Access Description

NAND (8-bit and 16-bit) memory devices using a standard NAND asynchronous address/data-multiplexing scheme can be supported on any chip-select with the appropriate asynchronous configuration settings

As for any other type of memory compatible with the GPMC interface, accesses to a chip-select allocated to a NAND device can be interleaved with accesses to chip-selects allocated to other external devices. This interleaved capability limits the system to *chip enable don't care* NAND devices, because the chip-select allocated to the NAND device must be de-asserted if accesses to other chip-selects are requested.

9.2.4.12.1 NAND Memory Device in Byte or 16-Bit Word Stream Mode

NAND devices require correct command and address programming before data array read or write accesses. The GPMC does not include specific hardware to translate a random address system request into a NAND-specific multiphase access. In that sense, GPMC NAND support, as opposed to random memory-map device support, is data-stream-oriented (byte or 16-bit word).

The GPMC NAND programming model relies on a software driver for address and command formatting with the correct data address pointer value according to the block and page structure. Because of NAND structure and protocol interface diversity, the GPMC does not support automatic command and address phase programming, and software drivers must access the NAND device ID to ensure that correct command and address formatting are used for the identified device.

NAND device data read and write accesses are achieved through an asynchronous read or write access. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

Any chip-select region can be qualified as a NAND region to constrain the $\overline{ADV}$_ALE signal as Address Latch Enable (ALE active high, default state value at low) during address program access, and the $\overline{BE0}$_CLE signal as Command Latch Enable (CLE active high, default state value at low) during command program access. GPMC address lines are not used (the previous value is not changed) during NAND access.

9.2.4.12.1.1 Chip-Select Configuration for NAND Interfacing in Byte or Word Stream Mode

The GPMC_CONFIG7_i register (where i = 0 to 5) associated with a NAND device region interfaced in byte or word stream mode can be initialized with a minimum size of 16 Mbytes, because any address location in the chip-select memory region can be used to access a NAND data array. The NAND Flash protocol specifies an address sequence where address bits are passed through the data bus in a series of write accesses with the ALE pin asserted. After this address phase, all operations are streamed and the system requests address is irrelevant.

To allow correct command, address, and data-access controls, the GPMC_CONFIG1_i register associated with a NAND device region must be initialized in asynchronous read and write modes with the parameters shown in Table 9-11. Failure to comply with these settings corrupts the NAND interface protocol.

The GPMC_CONFIG1_i to GPMC_CONFIG4_i register (where i = 0 to 5) associated with a NAND device region must be initialized with the correct control-signal timing value according to the NAND device timing parameters.

Table 9-11. Chip-Select Configuration for NAND Interfacing

GPMC_CONFIG1_i Bit Field	Value	Comments
WRAPBURST	0	No wrap
READMULTIPLE	0	Single access
READTYPE	0	Asynchronous mode
WRITEMULTIPLE	0	Single access
WRITETYPE	0	Asynchronous mode
CLKACTIVATIONTIME	0b00	
ATTACHEDDEVICEPAGELENGTH	Don't care	Single-access mode
WAITREADMONITORING	0	Wait not monitored by GPMC access engine
WAITWRITEMONITORING	0	Wait not monitored by GPMC access engine
WAITMONITORINGTIME	Don't care	Wait not monitored by GPMC access engine
WAITPINSELECT		Select which wait is monitored by edge detectors
DEVICESIZE	0b00 or 0b01	8- or 16-bit interface
DEVICETYPE	0b10	NAND device in stream mode
MUXADDDATA	0b00	Nonmultiplexed mode
TIMEPARAGRANULARITY	0	Timing achieved with best GPMC clock granularity
GPMCFCLKDIVIDER	Don't care	Asynchronous mode

9.2.4.12.1.2 NAND Device Command and Address Phase Control

NAND devices require multiple address programming phases. The MPU software driver is responsible for issuing the correct number of command and address program accesses, according to the device command set and the device address-mapping scheme.

NAND device-command and address-phase programming is achieved through write requests to the GPMC_NAND_COMMAND_i and GPMC_NAND_ADDRESS_i (where i = 0 to 5) register locations with the correct command and address values. These locations are mapped in the associated chip-select register region. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

Command and address values are not latched during the access and cannot be read back at the register location.

- Only write accesses must be issued to these locations, but the GPMC does not discard any read access. Accessing a NAND device with $\overline{OE}$ and CLE or ALE asserted (read access) can produce undefined results.
- Write accesses to the GPMC_NAND_COMMAND_i and GPMC_NAND_ADDRESS_i (where i = 0 to 5) register locations must be posted for faster operations. The GPMC_CONFIG NANDFORCEPOSTEDWRITE bit enables write accesses to these locations as posted, even if they are defined as nonposted.

A write buffer is used to store write transaction information before the external device is accessed:

- Up to eight consecutive posted write accesses can be accepted and stored in the write buffer.
- For nonposted write, the pipeline is one deep.
- A GPMC_STATUS EMPTYWRITEBUFFERSTATUS bit stores the empty status of the write buffer.

The GPMC_NAND_COMMAND_i and GPMC_NAND_ADDRESS_i (where i = 0 to 5) registers are 32-bit word locations, which means any 32-bit word or 16-bit word access is split into 4- or 2-byte accesses if an 8-bit wide NAND device is attached. For multiple-command phase or multiple-address phase, the software driver can use 32-bit word or 16-bit word access to these registers, but it must account for the splitting and little-endian ordering scheme. When only one byte command or address phase is required, only byte write access to a GPMC_NAND_COMMAND_i and GPMC_NAND_ADDRESS_i can be used, and any of the four byte locations of the registers are valid.

The same applies to GPMC_NAND_COMMAND_i and GPMC_NAND_ADDRESS_i (where i = 0 to 5) 32-bit word write access to a 16-bit wide NAND device (split into two 16-bit word accesses). In the case of a 16-bit word write access, the MSByte of the 16-bit word value must be set according to the NAND device requirement (usually 0). Either 16-bit word location or any one of the four byte locations of the registers is valid

9.2.4.12.1.3 Command Latch Cycle

Writing data at the GPMC_NAND_COMMAND_i (where i = 0 to 5) register location places the data as the NAND command value on the bus, using a regular asynchronous write access.

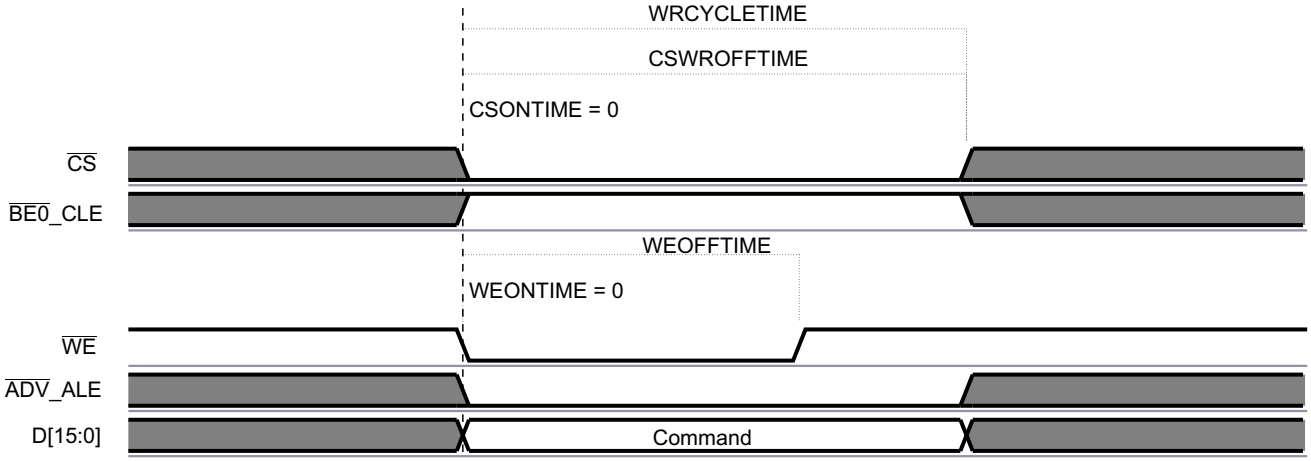
- $\overline{CE}$ is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- CLE is controlled by the ADVONTIME and ADVWROFFTIME timing parameters.
- WE is controlled by the WEONTIME and WEOFFTIME timing parameters.
- ALE and $\overline{RE}$ ($\overline{OE}$) are maintained inactive.

Figure 9-27 shows the NAND command latch cycle.

CLE is shared with the $\overline{BE0}$ output signal and has an inverted polarity from $\overline{BE0}$. The NAND qualifier deals with this. During the asynchronous NAND data access cycle, $\overline{BE0}$ (also $\overline{BE1}$) must not toggle, because it is shared with CLE.

NAND Flash memories do not use byte enable signals at all.

Figure 9-27. NAND Command Latch Cycle



9.2.4.12.1.4 Address Latch Cycle

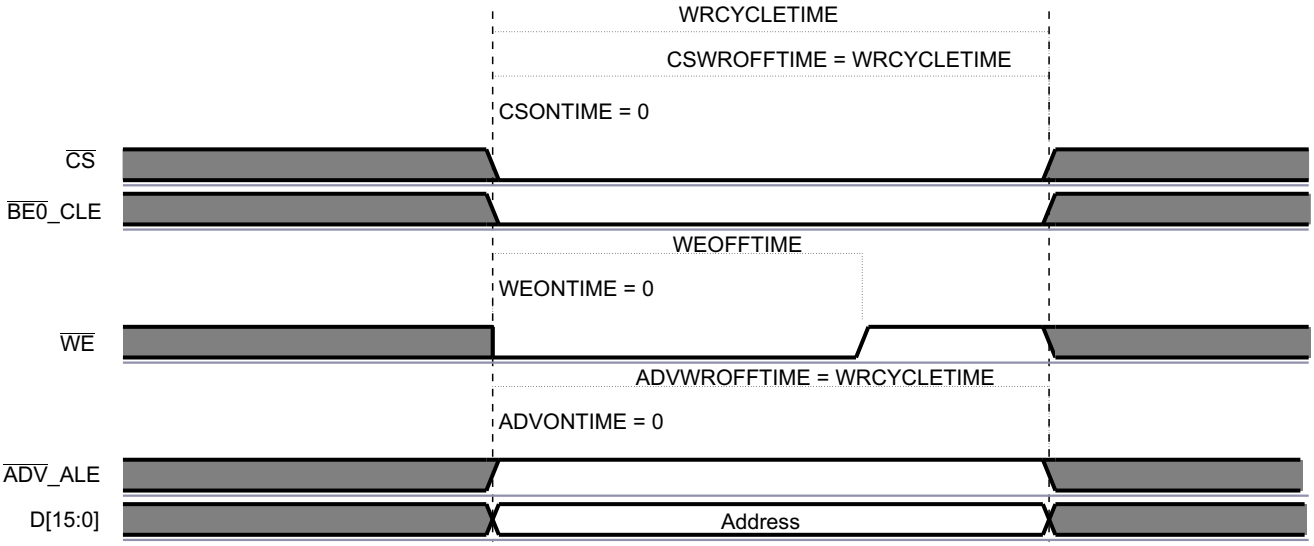
Writing data at the GPMC_NAND_ADDRESS_i (where i = 0 to 5) register location places the data as the NAND partial address value on the bus, using a regular asynchronous write access.

- $\overline{CS}$ is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- ALE is controlled by the ADVONTIME and ADVWROFFTIME timing parameters.
- $\overline{WE}$ is controlled by the WEONTIME and WEOFFTIME timing parameters.
- CLE and $\overline{RE}$ ($\overline{OE}$) are maintained inactive.

Figure 9-28 shows the NAND address latch cycle.

ALE is shared with the $\overline{ADV}$ output signal and has an inverted polarity from $\overline{ADV}$. The NAND qualifier deals with this. During the asynchronous NAND data access cycle, ALE is kept stable.

Figure 9-28. NAND Address Latch Cycle



9.2.4.12.1.5 NAND Device Data Read and Write Phase Control in Stream Mode

NAND device data read and write accesses are achieved through a read or write request to the chip-select-associated memory region at any address location in the region or through a read or write request to the GPMC_NAND_DATA_i (where i = 0 to 5) register location mapped in the chip-select-associated control register region. GPMC_NAND_DATA_i is not a true register, but an address location to enable $\overline{RE}$ or $\overline{WE}$ signal control. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

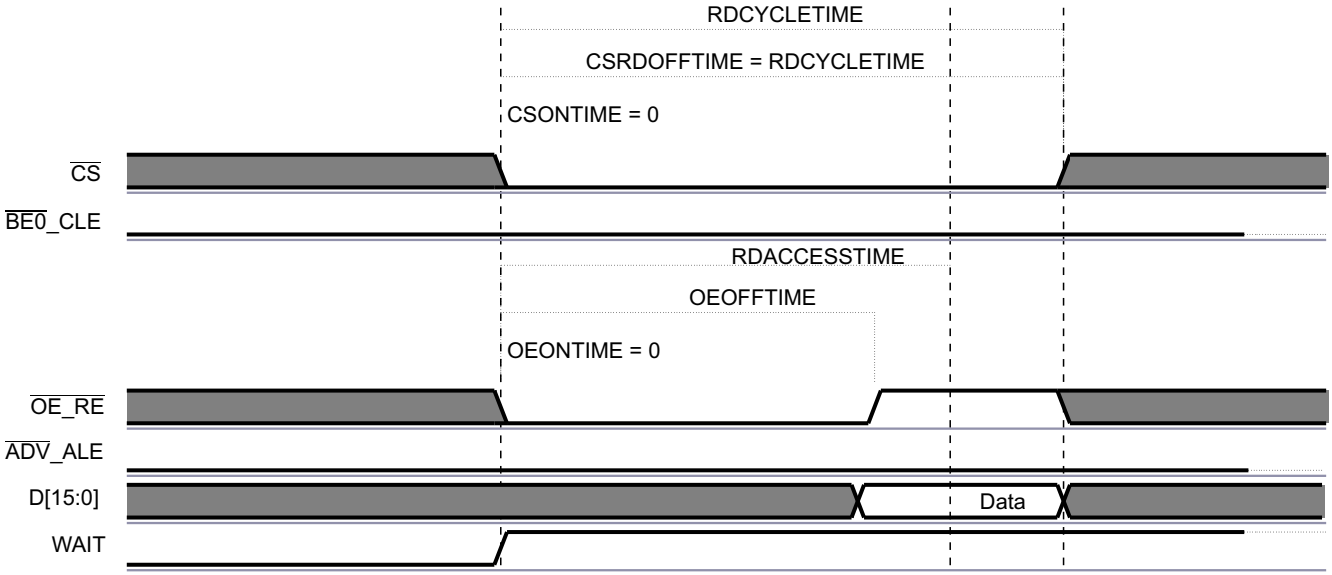
Reading data from the GPMC_NAND_DATA_i location or from any location in the associated chip-select memory region activates an asynchronous read access.

- $\overline{CS}$ is controlled by the CSONTIME and CSRDOFFTIME timing parameters.
- $\overline{RE}$ is controlled by the OEONTIME and OEOFFTIME timing parameters.
- To take advantage of $\overline{RE}$ high-to-data invalid minimum timing value, the RDACCESSTIME can be set so that data are effectively captured after $\overline{RE}$ deassertion. This allows optimization of NAND read access cycle time completion. For optimal timing parameter settings, see the NAND device and the device IC timing parameters.

ALE, CLE, and $\overline{WE}$ are maintained inactive.

Figure 9-29 shows the NAND data read cycle.

Figure 9-29. NAND Data Read Cycle

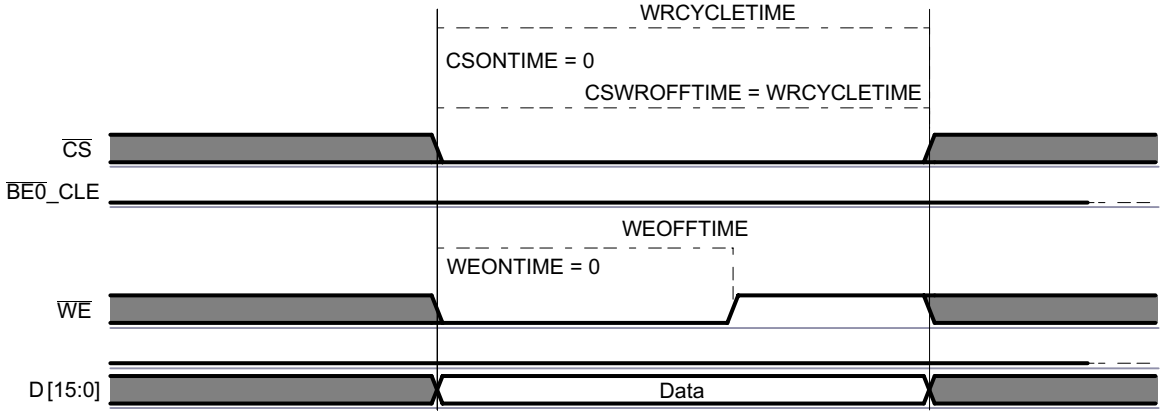


Writing data to the GPMC_NAND_DATA_i location or to any location in the associated chip-select memory region activates an asynchronous write access.

- $\overline{CS}$ is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- $\overline{WE}$ is controlled by the WEONTIME and WEOFFTIME timing parameters.
- ALE, CLE, and $\overline{RE}$ ($\overline{OE}$) are maintained inactive.

Figure 9-30 shows the NAND data write cycle.

Figure 9-30. NAND Data Write Cycle



9.2.4.12.1.6 NAND Device General Chip-Select Timing Control Requirement

For most NAND devices, read data access time is dominated by $\overline{CS}$-to-data-valid timing and has faster $\overline{RE}$-to-data-valid timing. Successive accesses with $\overline{CS}$ deassertions between accesses are affected by this timing constraint. Because accesses to a NAND device can be interleaved with other chip-select accesses, there is no certainty that $\overline{CS}$ always stays low between two accesses to the same chip-select. Moreover, an $\overline{CS}$ deassertion time between the same chip-select NAND accesses is likely to be required as follows: the $\overline{CS}$ deassertion requires programming CYCLETIME and RDACCESSTIME according to the $\overline{CS}$-to-data-valid critical timing.

To get full performance from NAND read and write accesses, the prefetch engine can dynamically reduce RDCYCLETIME, WRCYCLETIME, RDACCESSTIME, WRACCESSTIME, CSRDOFFTIME, CSWROFFTIME, ADVRDOFFTIME, ADVWROFFTIME, OEOFFTIME, and WEOFFTIME on back-to-back NAND accesses (to the same memory) and suppress the minimum $\overline{CS}$ high pulse width between accesses. For more information about optimal prefetch engine access, see Section 9.2.4.12.4.

Some NAND devices require minimum write-to-read idle time, especially for device-status read accesses following status-read command programming (write access). If such write-to-read transactions are used, a minimum $\overline{CS}$ high pulse width must be set. For this, CYCLE2CYCLESAMECSEN and CYCLE2CYCLEDELAY must be set according to the appropriate timing requirement to prevent any timing violation.

NAND devices usually have an important $\overline{RE}$ high to data bus in tristate mode. This requires a bus turnaround setting (BUSTURNAROUND = 1), so that the next access to a different chip-select is delayed until the BUSTURNAROUND delay completes. Back-to-back NAND read accesses to the same NAND Flash are not affected by the programmed bus turnaround delay.

9.2.4.12.1.7 Read and Write Access Size Adaptation

9.2.4.12.1.7.1 8-Bit Wide NAND Device

Host 16-bit word and 32-bit word read and write access requests to a chip-select associated with an 8-bit wide NAND device are split into successive read and write byte accesses to the NAND memory device. Byte access is ordered according to little-endian organization. A NAND 8-bit wide device must be interfaced on the D0D7 interface bus lane. GPMC data accesses are justified on this bus lane when the chip-select is associated with an 8-bit wide NAND device.

9.2.4.12.1.7.2 16-Bit Wide NAND Device

Host 32-bit word read and write access requests to a chip-select associated with a 16-bit wide NAND device are split into successive read and write 16-bit word accesses to the NAND memory device. 16-bit word access is ordered according to little-endian organization.

Host byte read and write access requests to a 16-bit wide NAND device are completed as 16-bit accesses on the device itself, because there is no byte-addressing capability on 16-bit wide NAND devices. This means that the NAND device address pointer is incremented on a 16-bit word basis and not on a byte basis. For a read access, only the requested byte is given back to the host, but the remaining byte is not stored or saved by the GPMC, and the next byte or 16-bit word read access gets the next 16-bit word NAND location. For a write access, the invalid byte part of the 16-bit word is driven to FF, and the next byte or 16-bit word write access programs the next 16-bit word NAND location.

Generally, byte access to a 16-bit wide NAND device should be avoided, especially when ECC calculation is enabled. 8-bit or 16-bit ECC-based computations are corrupted by a byte read to a 16-bit wide NAND device, because the nonrequested byte is considered invalid on a read access (not captured on the external data bus; FFh is applied to the ECC engine) and is set to FFh on a write access.

Host requests (read/write) issued in the chip-select memory region are translated in successive single or split accesses (read/write) to the attached device. Therefore, incrementing 32-bit burst requests are translated in multiple 32-bit sequential accesses following the access adaptation of the 32-bit to 8- or 16-bit device.

9.2.4.12.2 NAND Device-Ready Pin

The NAND memory device provides a ready pin to indicate data availability after a block/page opening and to indicate that data programming is complete. The ready pin can be connected to one of the WAIT GPMC input pins; data read accesses must not be tried when the ready pin is sampled inactive (device is not ready) even if the associated chip-select WAITREADMONITORING bit field is set. The duration of the NAND device busy state after the block/page opening is so long (up to 50 µs) that accesses occurring when the ready pin is sampled inactive can stall GPMC access and eventually cause a system time-out.

If a read access to a NAND flash is done using the wait monitoring mode, the device is blocked during a page opening, and so is the GPMC. If the correct settings are used, other chip-selects can be used while the memory processes the page opening command.

To avoid a time-out caused by a block/page opening delay in NAND flash, disable the wait pin monitoring for read and write accesses (that is, set the GPMC_CONFIG1_i WAITWRITEMONITORING and GPMC_CONFIG1_i WAITREADMONITORING bits to 0, where i = 0 to 5), and use one of the following methods instead:

- Use software to poll the WAITnSTATUS bit (n = 0 to 1) of the GPMC_STATUS register.
- Configure an interrupt that is generated on the WAIT signal change (through the GPMC_IRQENABLE [9-8] bits).

Even if the READWAITMONITORING bit is not set, the external memory nR/B pin status is captured in the programmed WAIT bit in the GPMC_STATUS register.

The READWAITMONITORING bit method must be used for other memories than NAND flash, if they require the use of a WAIT signal.

9.2.4.12.2.1 Ready Pin Monitored by Software Polling

The ready signal state can be monitored through the GPMC_STATUS WAITxSTATUS bit (x = 0 or 1). The software must monitor the ready pin only when the signal is declared valid. Refer to the NAND device timing parameters to set the correct software temporization to monitor ready only after the invalid window is complete from the last read command written to the NAND device.

9.2.4.12.2.2 Ready Pin Monitored by Hardware Interrupt

Each gpmc_wait input pin can generate an interrupt when a wait-to-no-wait transition is detected. Depending on whether the GPMC_CONFIG WAITxPINPOLARITY bits (x = 0 or 1) is active low or active high, the wait-to-no-wait transition is a low-to-high external WAIT signal transition or a high-to-low external WAIT signal transition, respectively.

The wait transition pin detector must be cleared before any transition detection. This is done by writing 1 to the WAITxEDGEDETECTIONSTATUS bit (x = 0 or 1) of the GPMC_IRQSTATUS register according to the gpmc_wait pin used for the NAND device-ready signal monitoring. To detect a wait-to-no-wait transition, the transition detector requires a wait active time detection of a minimum of two GPMC_FCLK cycles. Software must incorporate precautions to clear the wait transition pin detector before wait (busy) time completes.

A wait-to-no-wait transition detection can issue a GPMC interrupt if the WAITxEDGEDETECTIONENABLE bit in the GPMC_IRQENABLE register is set and if the WAITxEDGEDETECTIONSTATUS bit field in the GPMC_IRQSTATUS register is set.

The WAITMONITORINGTIME field does not affect wait-to-no-wait transition time detection.

It is also possible to poll the WAITxEDGEDETECTIONSTATUS bit field in the GPMC_IRQSTATUS register according to the gpmc_wait pin used for the NAND device ready signal monitoring.

9.2.4.12.3 ECC Calculator

The General Purpose Memory Controller includes an Error Code Correction (ECC) calculator circuitry that enables on the fly ECC calculation during data read or data program (that is, write) operations. The page size supported by the ECC calculator in one calculation/context is 512 bytes.

The user can choose from two different algorithms with different error correction capabilities through the GPMC_ECC_CONFIG ECCALGORITHM bit:

- Hamming code for 1-bit error code correction on 8- or 16-bit NAND Flash organized with page size greater than 512 bytes
- BCH (Bose-Chaudhurl-Hocquenghem) code for 4- to 16-bit error correction

The GPMC does not directly handle the error code correction itself. During writes, the GPMC computes parity bits. During reads, the GPMC provides enough information for the processor to correct errors without reading the data buffer all over again.

The Hamming code ECC is based on a 2-dimensional (row and column) bit parity accumulation. This parity accumulation is either accomplished on the programmed number of bytes or 16-bit words read from the memory device, or written to the memory device in stream mode.

Because the ECC engine includes only one accumulation context, it can be allocated to only one chip-select at a time through the GPMC_ECC_CONFIG ECCCS field. Even if two CS use different ECC algorithms, one the Hamming code and the other a BCH code, they must define separate ECC contexts because some of the ECC registers are common to all types of algorithms.

9.2.4.12.3.1 Hamming Code

All references to Error Code Correction (ECC) in this subsection refer to the 1-bit error correction Hamming code.

The ECC is based on a two-dimensional (row and column) bit parity accumulation known as Hamming Code. The parity accumulation is done for a programmed number of bytes or 16-bit word read from the memory device or written to the memory device in stream mode.

There is no automatic error detection or correction, and it is the software NAND driver responsibility to read the multiple ECC calculation results, compare them to the expected code value, and take the appropriate corrective actions according to the error handling strategy (ECC storage in spare byte, error correction on read, block invalidation).

The ECC engine includes a single accumulation context. It can be allocated to a single designated chip-select at a time and parallel computations on different chip-selects are not possible. Since it is allocated to a single chip-select, the ECC computation is not affected by interleaved GPMC accesses to other chip-selects and devices. The ECC accumulation is sequentially processed in the order of data read from or written to the memory on the designated chip-select. The ECC engine does not differentiate read accesses from write accesses and does not differentiate data from command or status information. It is the software responsibility to make sure only relevant data are passed to the NAND flash memory while the ECC computation engine is active.

The starting NAND page location must be programmed first, followed by an ECC accumulation context reset with an ECC enabling, if required. The NAND device accesses discussed in the following sections must be limited to data read or write until the specified number of ECC calculations is completed.

9.2.4.12.3.1.1 ECC Result Register and ECC Computation Accumulation Size

The GPMC includes up to nine ECC result registers (GPMC_ECCj_RESULT, j = 1 to 9) to store ECC computation results when the specified number of bytes or 16-bit words has been computed.

The ECC result registers are used sequentially; one ECC result is stored in one ECC result register on the list, the next ECC result is stored in the next ECC result register on the list, and so forth, until the last ECC computation. The value of the GPMC_ECCj_RESULT register value is valid only when the programmed number of bytes or 16-bit words has been accumulated, which means that the same number of bytes or 16-bit words has been read from or written to the NAND device in sequence.

The GPMC_ECC_CONTROL ECCPOINTER field must be set to the correct value to select the ECC result register to be used first in the list for the incoming ECC computation process. The ECCPOINTER can be read to determine which ECC register is used in the next ECC result storage for the ongoing ECC computation. The value of the GPMC_ECCj_RESULT register (j = 1 to 9) can be considered valid when ECCPOINTER equals j + 1. When the GPMC_ECCj_RESULT (where j = 9) is updated, ECCPOINTER is frozen at 10, and ECC computing is stopped (ECCENABLE = 0).

The ECC accumulator must be reset before any ECC computation accumulation process. The GPMC_ECC_CONTROL[8] ECCCLEAR bit must be set to 1 (nonpersistent bit) to clear the accumulator and all ECC result registers.

For each ECC result (each register, j = 1 to 9), the number of bytes or 16-bit words used for ECC computing accumulation can be selected from between two programmable values.

The ECCjRESULTSIZE bits (j = 1 to 9) in the GPMC_ECC_SIZE_CONFIG register select which programmable size value (ECCSIZE0 or ECCSIZE1) must be used for this ECC result (stored in GPMC_ECCj_RESULT register).

The ECCSIZE0 and ECCSIZE1 fields allow selection of the number of bytes or 16-bit words used for ECC computation accumulation. Any even values from 2 to 512 are allowed.

Flexibility in the number of ECCs computed and the number of bytes or 16-bit words used in the successive ECC computations enables different NAND page error-correction strategies. Usually based on 256 or 512 bytes and on 128 or 256 16-bit word, the number of ECC results required is a function of the NAND device page size. Specific ECC accumulation size can be used when computing the ECC on the NAND spare byte.

For example, with a 2 Kbyte data page 8-bit wide NAND device, eight ECCs accumulated on 256 bytes can be computed and added to one extra ECC computed on the 24 spare bytes area where the eight ECC results used for comparison and correction with the computed data page ECC are stored. The GPMC then provides nine GPMC_ECCj_RESULT registers (j = 1 to 9) to store the results. In this case, ECCSIZE0 is set to 256, and ECCSIZE1 is set to 24; the ECC[1-8]RESULTSIZE bits are cleared to 0, and the ECC9RESULTSIZE bit is set to 1.

9.2.4.12.3.1.2 ECC Enabling

The GPMC_ECC_CONFIG ECCCS field selects the allocated chip-select. The GPMC_ECC_CONFIG ECCENABLE bit enables ECC computation on the next detected read or write access to the selected chip-select.

The ECCPOINTER, ECCCLEAR, ECCSIZE, ECCjRESULTSIZE (where j = 1 to 9), ECC16B, and ECCCS fields must not be changed or cleared while an ECC computation is in progress.

The ECC accumulator and ECC result register must not be changed or cleared while an ECC computation is in progress.

Table 9-12 describes the ECC enable settings.

Table 9-12. ECC Enable Settings

Bit Field	Register	Value	Comments
ECCCS	GPMC_ECC_CONFIG	0-3h	Selects the chip-select where ECC is computed
ECC16B	GPMC_ECC_CONFIG	0-1	Selects column number for ECC calculation
ECCCLEAR	GPMC_ECC_CONTROL	0-7h	Clears all ECC result registers
ECCPOINTER	GPMC_ECC_CONTROL	0-7h	A write to this bit field selects the ECC result register where the first ECC computation is stored. Set to 1 by default.
ECCSIZE1	GPMC_ECC_SIZE_CONFIG	0-FFh	Defines ECCSIZE1
ECCSIZE0	GPMC_ECC_SIZE_CONFIG	0-FFh	Defines ECCSIZE0
ECCjRESULTSIZE	GPMC_ECC_SIZE_CONFIG	0-1	Selects the size of ECCn result register
ECCENABLE	GPMC_ECC_CONFIG	1	Enables the ECC computation

9.2.4.12.3.1.3 ECC Computation

The ECC algorithm is a multiple parity bit accumulation computed on the odd and even bit streams extracted from the byte or Word 16 streams. The parity accumulation is split into row and column accumulations, as shown in Figure 9-31 and Figure 9-32. The intermediate row and column parities are used to compute the upper level row and column parities. Only the final computation of each parity bit is used for ECC comparison and correction.

P1o = bit7 XOR bit5 XOR bit3 XOR bit1 on each byte of the data stream

P1e = bit6 XOR bit4 XOR bit2 XOR bit0 on each byte of the data stream

P2o = bit7 XOR bit6 XOR bit3 XOR bit2 on each byte of the data stream

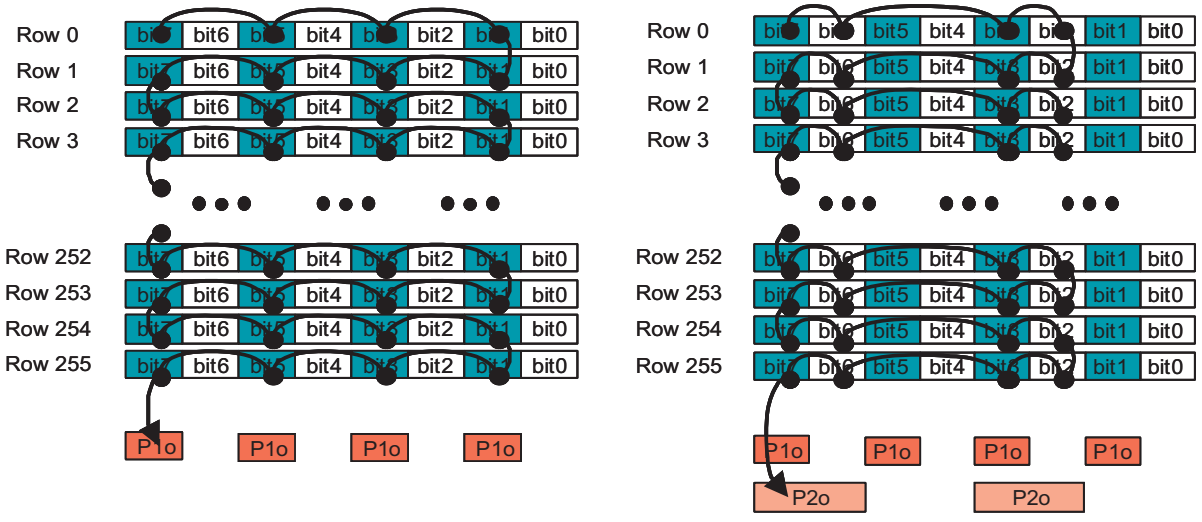
P2e = bit5 XOR bit4 XOR bit1 XOR bit0 on each byte of the data stream

P4o = bit7 XOR bit6 XOR bit5 XOR bit4 on each byte of the data stream

P4e = bit3 XOR bit2 XOR bit1 XOR bit0 on each byte of the data stream

Each column parity bit is XORed with the previous accumulated value.

Figure 9-31. Hamming Code Accumulation Algorithm (1 of 2)



For line parities, the bits of each new data are XORed together, and line parity bits are computed as:

P8e = row0 XOR row2 XOR row4 XOR ... XOR row254

P8o = row1 XOR row3 XOR row5 XOR ... XOR row255

P16e = row0 XOR row1 XOR row4 XOR row5 XOR ... XOR row252 XOR row 253

P16o = row2 XOR row3 XOR row6 XOR row7 XOR ... XOR row254 XOR row 255

Unused parity bits in the result registers are cleared to 0.

Figure 9-32. Hamming Code Accumulation Algorithm (2 of 2)

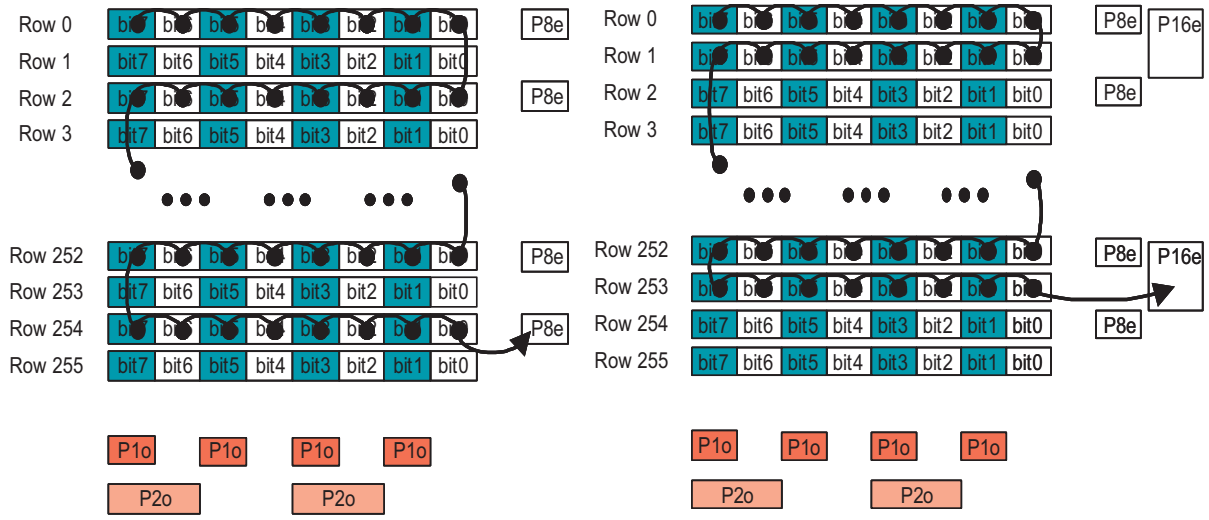


Figure 9-33 shows ECC computation for a 256-byte data stream (read or write). The result includes six column parity bits (P1o-P2o-P4o for odd parities, and P1e-P2e-P4e for even parities) and sixteen row parity bits (P8o-P16o-P32o--P1024o for odd parities, and P8e-P16e-P32e--P1024e for even parities).

Figure 9-33. ECC Computation for a 256-Byte Data Stream (Read or Write)

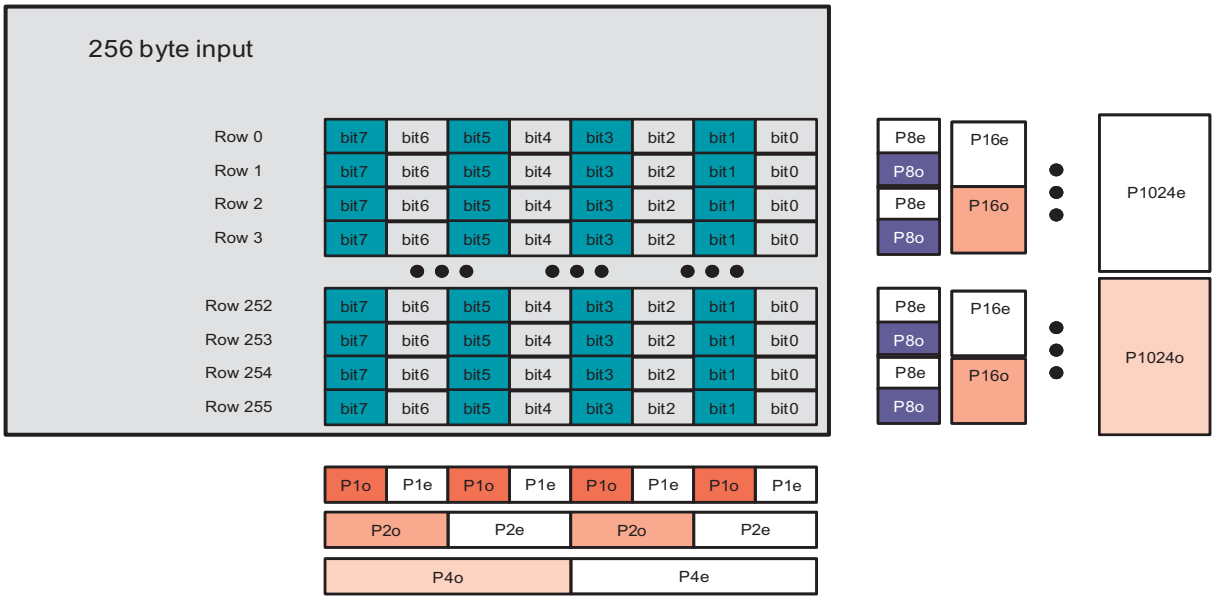
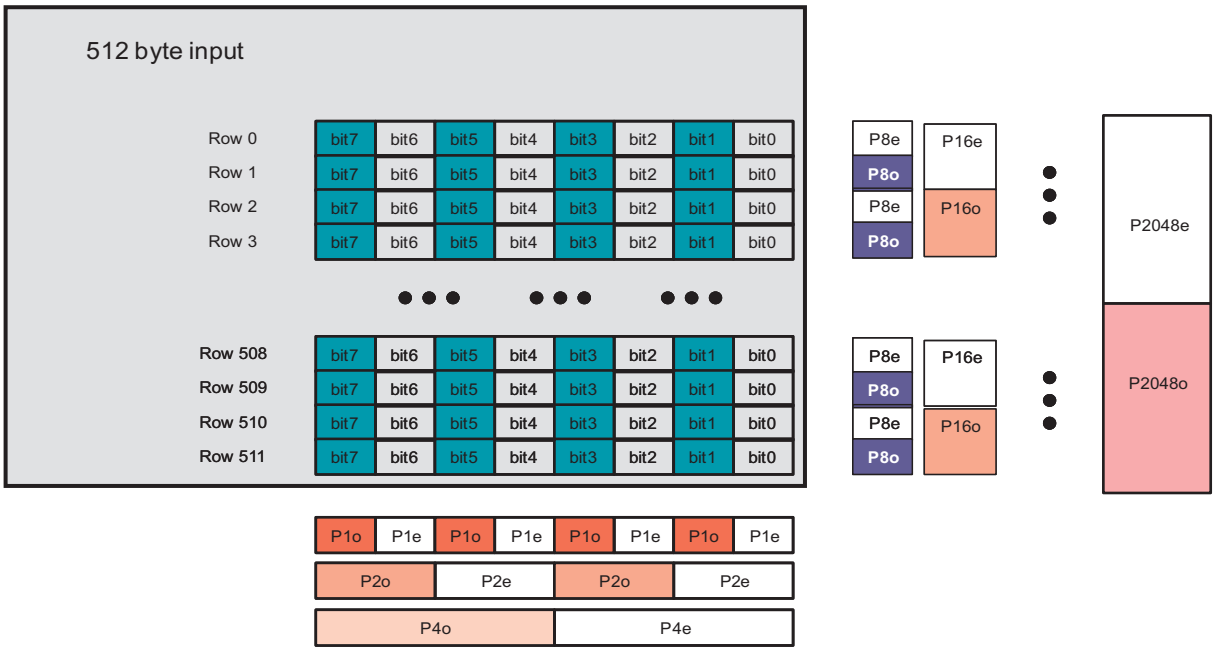


Figure 9-34 shows ECC computation for a 512-byte data stream (read or write). The result includes six column parity bits (P1o-P2o-P4o for odd parities, and P1e-P2e-P4e for even parities) and eighteen row parity bits (P8o-P16o-P32o--P1024o- - P2048o for odd parities, and P8e-P16e-P32e--P1024e- P2048e for even parities).

For a 2 Kbytes page, four 512 bytes ECC calculations plus one for the spare area are required. Results are stored in the GPMC_ECCj_RESULT registers (j = 1 to 9).

Figure 9-34. ECC Computation for a 512-Byte Data Stream (Read or Write)



9.2.4.12.3.1.4 ECC Comparison and Correction

To detect an error, the computed ECC result must be XORed with the parity value stored in the spare area of the accessed page.

- If the result of this logical XOR is all 0s, no error is detected and the read data is correct.
- If every second bit in the parity result is a 1, one bit is corrupted and is located at bit address (P2048o, P1024o, P512o, P256o, P128o, P64o, P32o, P16o, P8o, P4o, P2o, P1o). The software must correct the corresponding bit.
- If only one bit in the parity result is 1, it is an ECC error and the read data is correct.

9.2.4.12.3.1.5 ECC Calculation Based on 8-Bit Word

The 8-bit based ECC computation is used for 8-bit wide NAND device interfacing.

The 8-bit based ECC computation can be used for 16-bit wide NAND device interfacing to get backward compatibility on the error-handling strategy used with 8-bit wide NAND devices. In this case, the 16-bit wide data read from or written to the NAND device is fragmented into 2 bytes. According to little-endian access, the least significant bit (LSB) of the 16-bit wide data is ordered first in the byte stream used for 8-bit based ECC computation.

9.2.4.12.3.1.6 ECC Calculation Based on 16-Bit Word

ECC computation based on a 16-bit word is used for 16-bit wide NAND device interfacing. This ECC computation is not supported when interfacing an 8-bit wide NAND device, and the GPMC_ECC_CONFIG ECC16B bit must be cleared to 0 when interfacing an 8-bit wide NAND device.

The parity computation based on 16-bit words affects the row and column parity mapping. The main difference is that the odd and even parity bits P8o and P8e are computed on rows for an 8-bit based ECC while there are computed on columns for a 16-bit based ECC. Figure 9-35 and Figure 9-36.

Figure 9-35. 128 Word16 ECC Computation

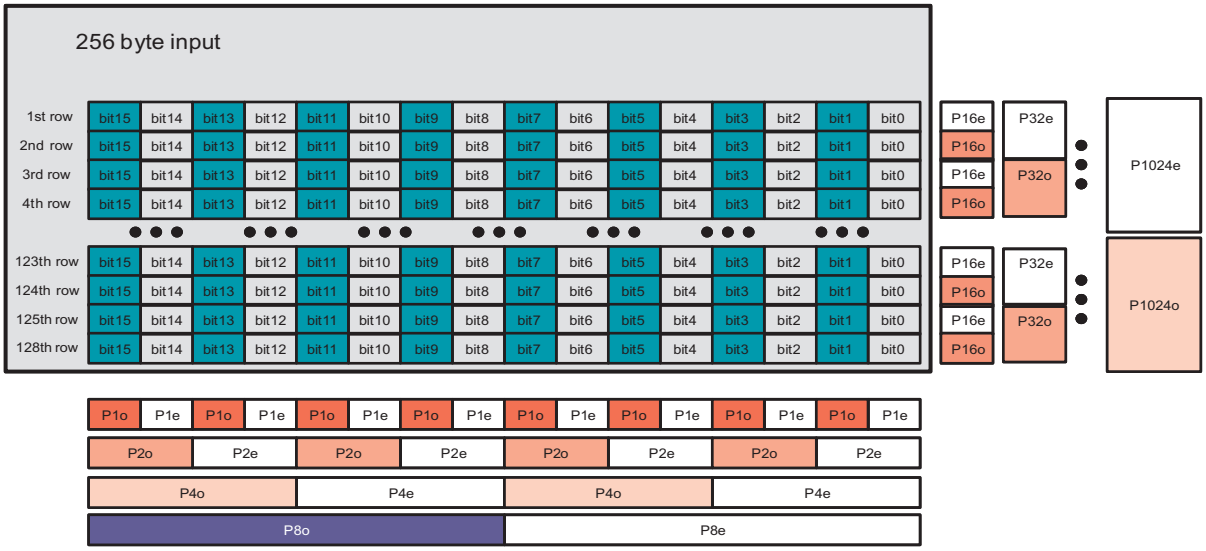
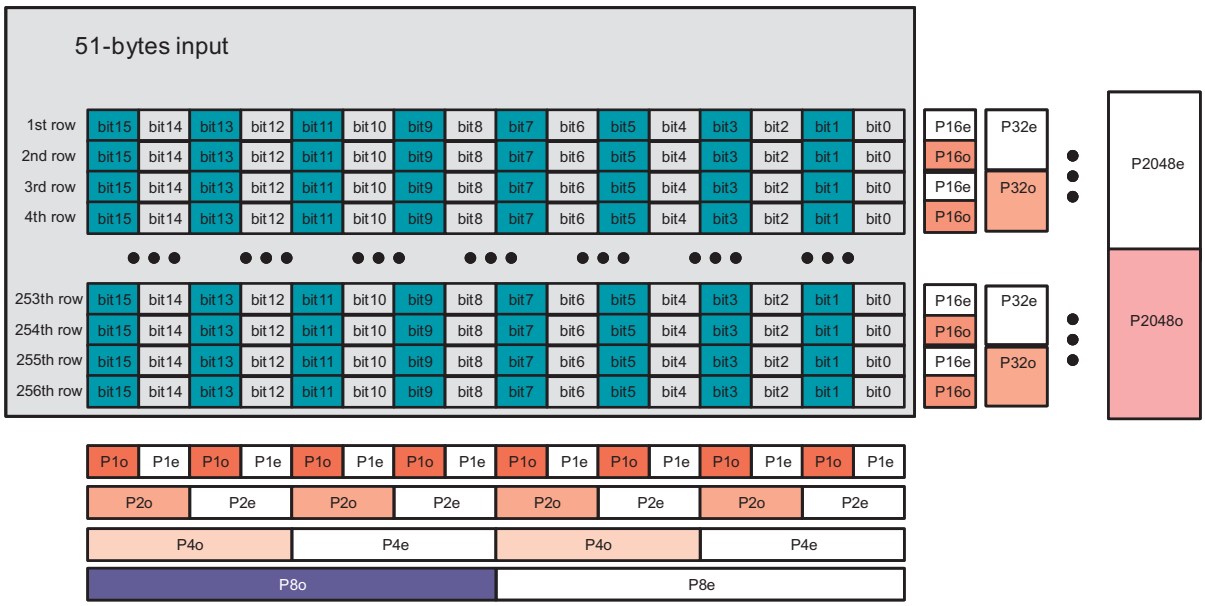


Figure 9-36. 256 Word16 ECC Computation



9.2.4.12.3.2 BCH Code (Bose-Chaudhurl-Hocquenghem)

All references to Error Code Correction (ECC) in this subsection refer to the 4- to 16-bit error correction BCH code.

9.2.4.12.3.2.1 Requirements

Read and write accesses to a NAND flash take place by whole pages, in a predetermined sequence: first the data byte page itself, then some spare bytes, including the BCH ECC (and other information). The NAND IC can cache a full page, including spares, for read and write accesses.

Typical page write sequence:

- Sequential write to NAND cache of main data + spare data, for a page. ECC is calculated on the fly. Calculated ECC may be inserted on the fly in the spares, or replaced by dummy accesses.
- When the calculated ECC is replaced by dummy accesses, it must be written to the cache in a second, separate phase. The ECC module is disabled during that time.
- NAND writes its cache line (page) to the array

Typical page read sequence:

- Sequential read of a page. ECC is calculated on the fly.
- ECC module buffers status determines the presence of errors.
- Accesses to several memories may be interleaved by the GPMC, but only one of those memories can be a NAND using the BCH engine at a time; in other words, only one BCH calculation (for example, for a single page) can be on-going at any time. Note also that the sequential nature of NAND accesses guarantees that the data is always written / read out in the same order. BCH-relevant accesses are selected by the GPMCs chip-select.
- Each page may hold up to 4 Kbytes of data, spare bytes not included. This means up to 8 × 512-byte BCH messages. Since all the data is written / read out first, followed by the BCH ECC, this means that the BCH engine must be able to hold 8 104-bit remainders or syndromes (or smaller, 52-bit ones) at the same time.

The BCH module has the capacity to store all remainders internally. After the page start, an internal counter is used to detect the 512-byte sector boundaries. On those boundaries, the current remainder is stored and the divider reset for the next calculation. At the end of the page, the BCH module contains all remainders.

- NAND access cycles hold 8 or 16 bits of data each (1 or 2 bytes); Each NAND cycle takes at least 4 cycles of the GPMCs internal clock. This means the NAND flash timing parameters must define a RDCYCLETIME and a WRCYCLETIME of at least 4 clock cycles after optimization when using the BCH calculator.
- The spare area is assumed to be large enough to hold the BCH ECC, that is, to have at least a message of 13 bytes available per 512-byte sector of data. The zone of unused spare area by the ECC may or may not be protected by the same ECC scheme, by extending the BCH message beyond 512 bytes (maximum codeword is 1023-byte long, ECC included, which leaves a lot of space to cover some spares bytes).

9.2.4.12.3.2.2 Memory-Mapping of the BCH Codeword

BCH encoding considers a block of data to protect as a polynomial message M(x). In our standard case, 512 bytes of data (that is, 2 bits = 4096 bits) are seen as a polynomial of degree 2 - 1 = 4095, with parameters ranging from M0 to M4095. For 512 bytes of data, 52 bits are required for 4-bit error correction, and 104 bits are required for 8-bit error correction and 207 bits are required for 16-bit error correction. The ECC is a remainder polynomial R(x) of degree 103 (or 51, depending on the selected mode). The complete codeword C(x) is the concatenation of M(x) and R(x) as shown in Table 9-13.

Table 9-13. Flattened BCH Codeword Mapping (512 Bytes + 104 Bits)

	Message M(x)	ECC R(x)				
Bit number	M4095	...	M0	R103	...	R0

If the message is extended by the addition of spare bytes to be protected by the same ECC, the principle is still valid. For example, a 3-byte extension of the message gives a polynomial message M(x) of degree ((512 + 3) × 8) - 1 = 4119, for a total of 3 + 13 = 16 spare bytes of spare, all protected as part of the same codeword.

The message and the ECC bits are manipulated and mapped in the GPMC byte-oriented system. The ECC bits are stored in , , , and (where i = 0 to 5).

- GPMC_BCH_RESULT0_i
- GPMC_BCH_RESULT1_i
- GPMC_BCH_RESULT2_i
- GPMC_BCH_RESULT3_i

9.2.4.12.3.2.3 Memory Mapping of the Data Message

The data message mapping shall follow the following rules:

- Bit endianness within a byte is little-endian, that is, the bytes LS bit is also the lowest-degree polynomial parameter: a byte b7-b0 (with b0 the LS bit) represents a segment of polynomial (b7 × x) + (b6 × x) + ... + (b0 × x
- The message is mapped in the NAND starting with the highest-order parameters, that is, in the lowest addresses of a NAND page.
- Byte endianness within the NANDs 16-bit words is big endian. This means that the same message mapped in 8- and 16-bit memories has the same content at the same byte address.

The BCH module has no visibility over actual addresses. The most important point is the sequence of data word the BCH sees. However, the NAND page is always scanned incrementally in read and write accesses, and this produces the mapping patterns described in the following.

Table 9-14 and Table 9-15 show the mapping of the same 512-byte vector (typically a BCH message) in the NAND memory space. Note that the byte 'address' is only an offset modulo 512 (200h), since the same page may contain several contiguous 512-byte sectors (BCH blocks). The LSB and MSB are respectively the bits M0 and M(2^12-1) of the codeword mapping given above. In both cases the data vectors are aligned, that is, their boundaries coincide with the RAMs data word boundaries.

Table 9-14. Aligned Message Byte Mapping in 8-bit NAND

Byte Offset	8-Bit Word
0	(msb) Byte 511 (1FFh)
1h	Byte 510 (1FEh)
⋮	⋮
1FFh	Byte 0 (0) (LSB)

Table 9-15. Aligned Message Byte Mapping in 16-bit NAND

Byte Offset	16-Bit Words MSB	16-Bit Words LSB
0	Byte 510 (1FEh)	(msb) Byte 511 (1FFh)
2h	Byte 508 (1FCh)	Byte 509 (1FDh)
⋮	⋮	⋮
1FEh	Byte 0 (0)	(lsb) Byte 1 (1)

Table 9-16 and Table 9-17 show the mapping in memory of arbitrarily-sized messages, starting on access (byte or 16-bit word) boundaries for more clarity. Note that message may actually start and stop on arbitrary nibbles. A nibble is a 4-bit entity. The unused nibbles are not discarded, and they can still be used by the BCH module, but as part of the next message section (for example, on another sectors ECC).

Table 9-16. Aligned Nibble Mapping of Message in 8-bit NAND

Byte Offset	8-Bit Word	
	4-Bit Most Significant Nibble	4-Bit Less Significant Nibble
1	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-3	Nibble S-4
⋮	⋮	⋮
S/2 - 2	Nibble 3	Nibble 2
S/2 - 1	Nibble 1	Nibble 0 (LSB)

Table 9-17. Misaligned Nibble Mapping of Message in 8-bit NAND

Byte Offset	8-Bit Word	
	4-Bit Most Significant Nibble	4-Bit Less Significant Nibble
1	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-3	Nibble S-4
⋮	⋮	⋮
(S+1)/2 - 2	Nibble 2	Nibble 1
(S+1)/2 - 1	Nibble 0 (LSB)	

Table 9-18. Aligned Nibble Mapping of Message in 16-bit NAND

Byte Offset	16-Bit Word			
	4-Bit Most Significant Nibble		4-Bit Less Significant Nibble	
0	Nibble S-3	Nibble S-4	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
⋮	⋮	⋮	⋮	⋮
S/2 - 4	Nibble 5	Nibble 4	Nibble 7	Nibble 6
S/2 - 2	Nibble 1	Nibble 0 (LSB)	Nibble 3	Nibble 2

Table 9-19. Misaligned Nibble Mapping of Message in 16-bit NAND (1 Unused Nibble)

Byte Offset	16-Bit Word			
	4-Bit Most Significant Nibble		4-Bit Less Significant Nibble	
0	Nibble S-3	Nibble S-4	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
⋮	⋮	⋮	⋮	⋮
(S+1)/2 - 4	Nibble 4	Nibble 3	Nibble 6	Nibble 5
(S+1)/2 - 2	Nibble 0 (LSB)		Nibble 2	Nibble 1

Table 9-20. Misaligned Nibble Mapping of Message in 16-bit NAND (2 Unused Nibble)

Byte Offset	16-Bit Word			
	4-Bit Most Significant Nibble		4-Bit Less Significant Nibble	
0	Nibble S-3	Nibble S-4	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
⋮	⋮	⋮	⋮	⋮
(S+2)/2 - 4	Nibble 3	Nibble 2	Nibble 5	Nibble 4
(S+2)/2 - 2			Nibble 1	Nibble 0 (LSB)

Table 9-21. Misaligned Nibble Mapping of Message in 16-bit NAND (3 Unused Nibble)

Byte Offset	16-Bit Word			
	4-Bit Most Significant Nibble		4-Bit Less Significant Nibble	
0	Nibble S-3	Nibble S-4	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
⋮	⋮	⋮	⋮	⋮
(S+3)/2 - 4	Nibble 2	Nibble 1	Nibble 4	Nibble 3
(S+3)/2 - 2			Nibble 0 (LSB)	

Note that many other cases exist than the ones represented above, for example, where the message does not start on a word boundary.

9.2.4.12.3.2.4 Memory Mapping of the ECC

The ECC (or remainder) is presented by the BCH module as a single 104-bit (or 52-bit), little-endian vector. It is up to the software to fetch those 13 bytes (or 6 bytes) from the modules interface, then store them to the NANDs spare area (page write) or to an intermediate buffer for comparison with the stored ECC (page read). There are no constraints on the ECC mapping inside the spare area: it is a software controlled operation.

However, it is advised to maintain a coherence in the respective formats of the message or the ECC remainder once they have been read out of the NAND. The error correction algorithm works from the complete codeword (concatenated message and remainder) once an error as been detected. The creation of this codeword should be made as straightforward as possible.

There are cases where the same NAND access contains both data and the ECC protecting that data. This is the case when the data/ECC boundary (which can be on any nibble) does not coincide with an access boundary. The ECC is calculated on-the-fly following the write. In that case, the write must also contain part of the ECC because it is impossible to insert the ECC on-the-fly. Instead:

- During the initial page write (BCH encoding), the ECC is replaced by dummy bits. The BCH encoder is by definition turned OFF during the ECC section, so the BCH result is unmodified.
- During a second phase, the ECC is written to the correct location, next to the actual data.
- The completed line buffer is then written to the NAND array.

9.2.4.12.3.2.5 Wrapping Modes

For a given wrapping mode, the module automatically goes through a specific number of sections, as data is being fed into the module. For each section, the BCH core can be enabled (in which case the data is fed to the BCH divider) or not (in which case the BCH simply counts to the end of the section). When enabled, the data is added to the ongoing calculation for a given sector number (for example, number 0).

Wrapping modes are described below. To get a better understanding and see the real-life read and write sequences implemented with each mode, see Section 9.2.4.12.3.3.

For each mode:

- A sequence describes the mode in pseudo-language, with for each section the size and the buffer used for ECC processing (if ON). The programmable lengths are size, size0 and size1.
- A checksum condition is given. If the checksum condition is not respected for a given mode, the modules behavior is unpredictable. S is the number of sectors in the page; size0 and size1 are the section sizes programmed for the mode, in nibbles.

Note that wrapping modes 8, 9, 10, and 11 insert a 1-nibble padding where the BCH processing is OFF. This is intended for t = 4 ECC, where ECC is 6 bytes long and the ECC area is expected to include (at least) 1 unused nibble to remain byte-aligned.

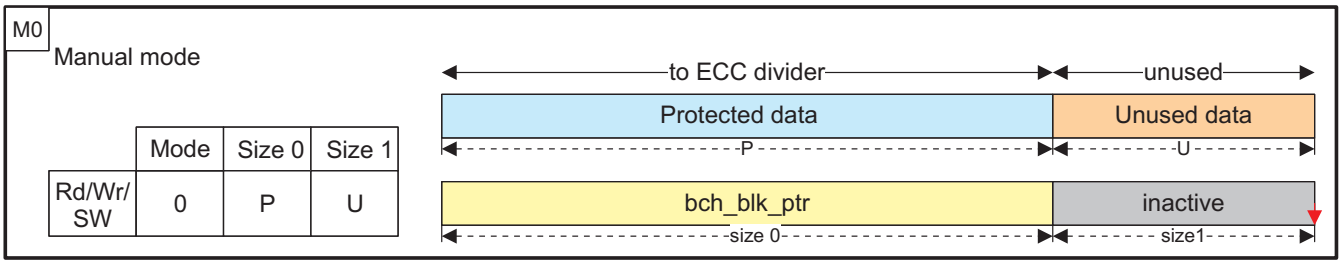
9.2.4.12.3.2.6 Manual Mode (0x0)

This mode is intended for short sequences, added manually to a given buffer through the software data port input. A complete page may be built out of several such sequences.

To process an arbitrary sequence of 4-bit nibbles, accesses to the software data port shall be made, containing the appropriate data. If the sequence end does not coincide with an access boundary (for example, to process 5 nibbles = 20 bits in 16-bit access mode) and those nibbles need to be skipped, a number of unused nibbles shall be programmed in size1 (in the same example: 5 nibbles to process + 3 to discard = 8 nibbles = exactly 2 × 16-bit accesses: we must program size0 = 5, size1 = 3).

Figure 9-37 shows the manual mode sequence and mapping. In this figure, size and size0 are the same parameter.

Figure 9-37. Manual Mode Sequence and Mapping



Section processing sequence:

- One time with buffer
  - size0 nibbles of data, processing ON
  - size1 nibbles of unused data, processing OFF

Checksum: size0 + size1 nibbles must fit in a whole number of accesses.

In the following sections, S is the number of sectors in the page.

9.2.4.12.3.2.7 Mode 0x1

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S - (size0 + size1)

9.2.4.12.3.2.8 Mode 0xA (10)

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
  - 1 nibble pad spare, processing OFF
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S - (size0 + 1 + size1)

9.2.4.12.3.2.9 Mode 0x2

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S - (size0 + size1)

9.2.4.12.3.2.10 Mode 0x3

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S - size1)

9.2.4.12.3.2.11 Mode 0x7

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat S times (no buffer used)
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = size0 + (S - size1)

9.2.4.12.3.2.12 Mode 0x8

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S - (1 + size1))

9.2.4.12.3.2.13 Mode 0x4

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time (no buffer used)
  - size0 nibbles spare, processing OFF
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S - size1)

9.2.4.12.3.2.14 Mode 0x9

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time (no buffer used)
  - size0 nibbles spare, processing OFF
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S - (1 + size1))

9.2.4.12.3.2.15 Mode 0x5

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S - (size0 + size1)

9.2.4.12.3.2.16 Mode 0xB (11)

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S - (size0 + 1 + size1)

9.2.4.12.3.2.17 Mode 0x6

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat S times (no buffer used)
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S - (size0 + size1)

9.2.4.12.3.3 Supported NAND Page Mappings and ECC Schemes

The following rules apply throughout the entire mapping description:

- Main data area (sectors) size is hardcoded to 512 bytes.
- Spare area size is programmable.
- All page sections (of main area data bytes, protected spare bytes, unprotected spare bytes, and ECC) are defined as explained in Section 9.2.4.12.3.2.3.

Each one of the following sections shows a NAND page mapping example (per-sector spare mappings, pooled spare mapping, per-sector spare mapping, with ECC separated at the end of the page).

In the mapping diagrams, sections that belong to the same BCH codeword have the same color (blue or green); unprotected sections are not covered (orange) by the BCH scheme.

Below each mapping diagram, a write (encoding) and read (decoding: syndrome generation) sequence is given, with the number of the active buffers at each point in time (yellow). In the inactive zones (grey), no computing is taking place but the data counter is still active.

In Figure 9-38 to Figure 9-40, tables on the left summarize the mode, size0, size1 parameters to program for respectively write and read processing of a page, with the given mapping, where:

- P is the size of spare byte section Protected by the ECC (in nibbles)
- U is the size of spare byte section Unprotected by the ECC (in nibbles)
- E is the size of the ECC itself (in nibbles)
- S is the number of Sectors per page (2 in the current diagrams)

Each time the processing of a BCH block is complete (ECC calculation for write/encoding, syndrome generation for read/decoding, indicated by red arrows), the update pointer is pulsed. Note that the processing for block 0 can be the first or the last to complete, depending on the NAND page mapping and operation (read or write). All examples show a page size of 1kByte + spares, that is, S = 2 sectors of 512 bytes. The same principles can be extended to larger pages by adding more sectors.

The actual BCH codeword size is used during the error location work to restrict the search range: by definition, errors can only happen in the codeword that was actually written to the NAND, and not in the mathematical codeword of n = 2 - 1 = 8191 bits. That codeword (higher-order bits) is all-zero and implicit during computations.

The actual BCH codeword size depends on the mode, on the programmed sizes and on the sector number (all sizes in nibbles):

- Spares mapped and protected per sector (Figure 9-38: see M1-M2-M3-M9-M10):
  - all sectors: (512) + P + E
- Spares pooled and protected by sector 0 (Figure 9-38: see M5-M6):
  - sector 0 codeword: (512) + P + E
  - other sectors: (512) + E
- Unprotected spares (Figure 9-38: see M4-M7-M8-M11-M12):
  - all codewords (512) + E

9.2.4.12.3.3.1 Per-Sector Spare Mappings

In these schemes (Figure 9-38), each 512-byte sector of the main area has its own dedicated section of the spare area. The spare area of each sector is composed of:

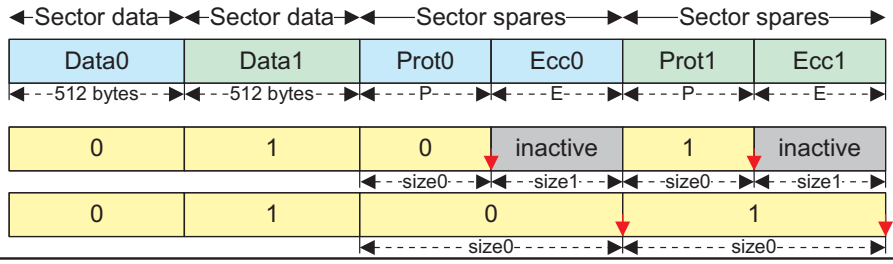
- ECC, which must be located after the data it protects
- other data, which may or may not be protected by the sectors ECC

Figure 9-38. NAND Page Mapping and ECC: Per-Sector Schemes

M1

Per-sector spares
Spares covered by sector ECC per sector ECC mapping.

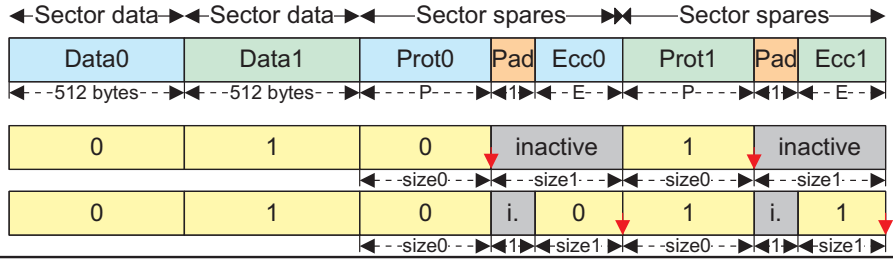
	Mode	Size0	Size1
Write	1	P	E
Read	1	P+E	0



M2

Per-sector spares
Spares covered by sector ECC per sector, left-padded ECC.

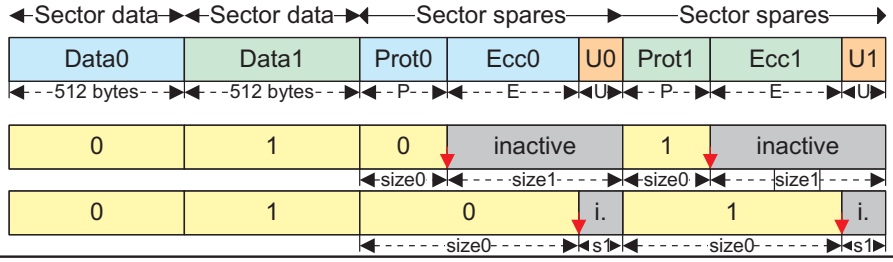
	Mode	Size0	Size1
Write	1	P	1+E
Read	10	P	E



M3

Per-sector spares
Spares covered by sector ECC, ECC not right-aligned.

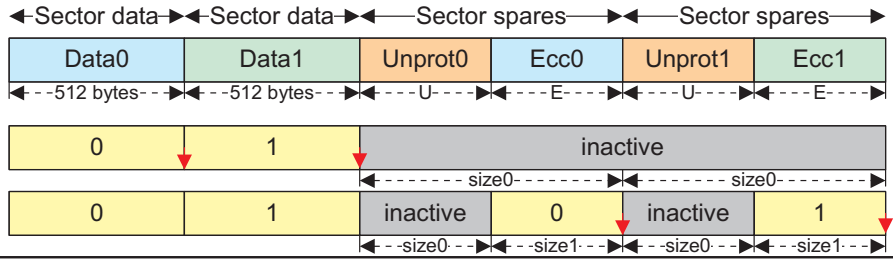
	Mode	Size0	Size1
Write	1	P	E+U
Read	1	P+E	U



M4

Per-sector spares
Spares not covered by ECC, ECC right-aligned per sector.

	Mode	Size0	Size1
Write	2	U+E	0
Read	2	U	E



9.2.4.12.3.3.2 Pooled Spare Mapping

In these schemes (Figure 9-39), the spare area is pooled for the page.

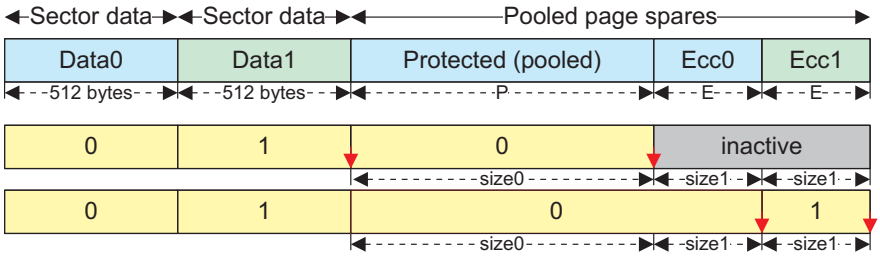
- The ECC of each sector is aligned at the end of the spare area.
- The non-ECC spare data may or may not be covered by the ECC of sector 0

Figure 9-39. NAND Page Mapping and ECC: Pooled Spare Schemes

M5

Pooled spares
Spares covered by ECC0.
All ECC at the end (of page).

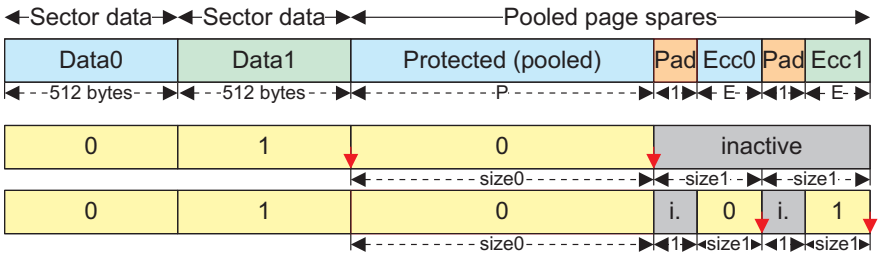
	Mode	Size0	Size1
Write	7	P	E
Read	3	P	E



M6

Pooled spares
Spares covered by ECC0.
All ECC at the end, left-padded.

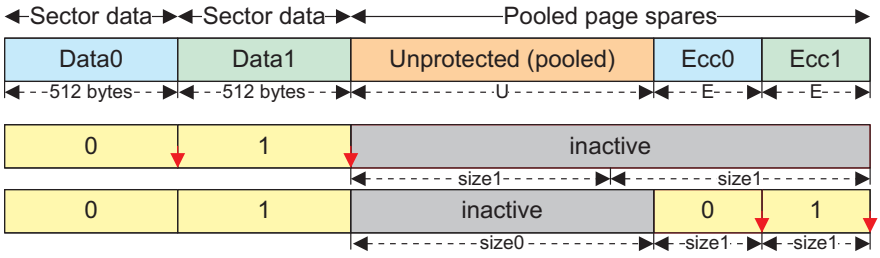
	Mode	Size0	Size1
Write	7	P	1+E
Read	8	P	E



M7

Pooled spares
Spares not covered by ECC.
All ECC at the end.

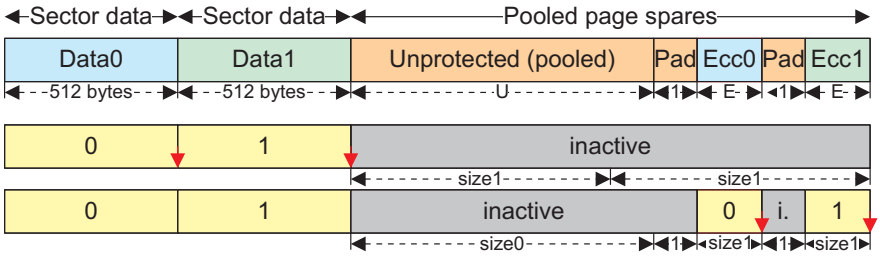
	Mode	Size0	Size1
Write	6	0	U/S +E
Read	4	U	E



M8

Pooled spares
Spares not covered by ECC.
All ECC at the end, left-padded.

	Mode	Size0	Size1
Write	6	0	U/S +1+E
Read	9	U	E

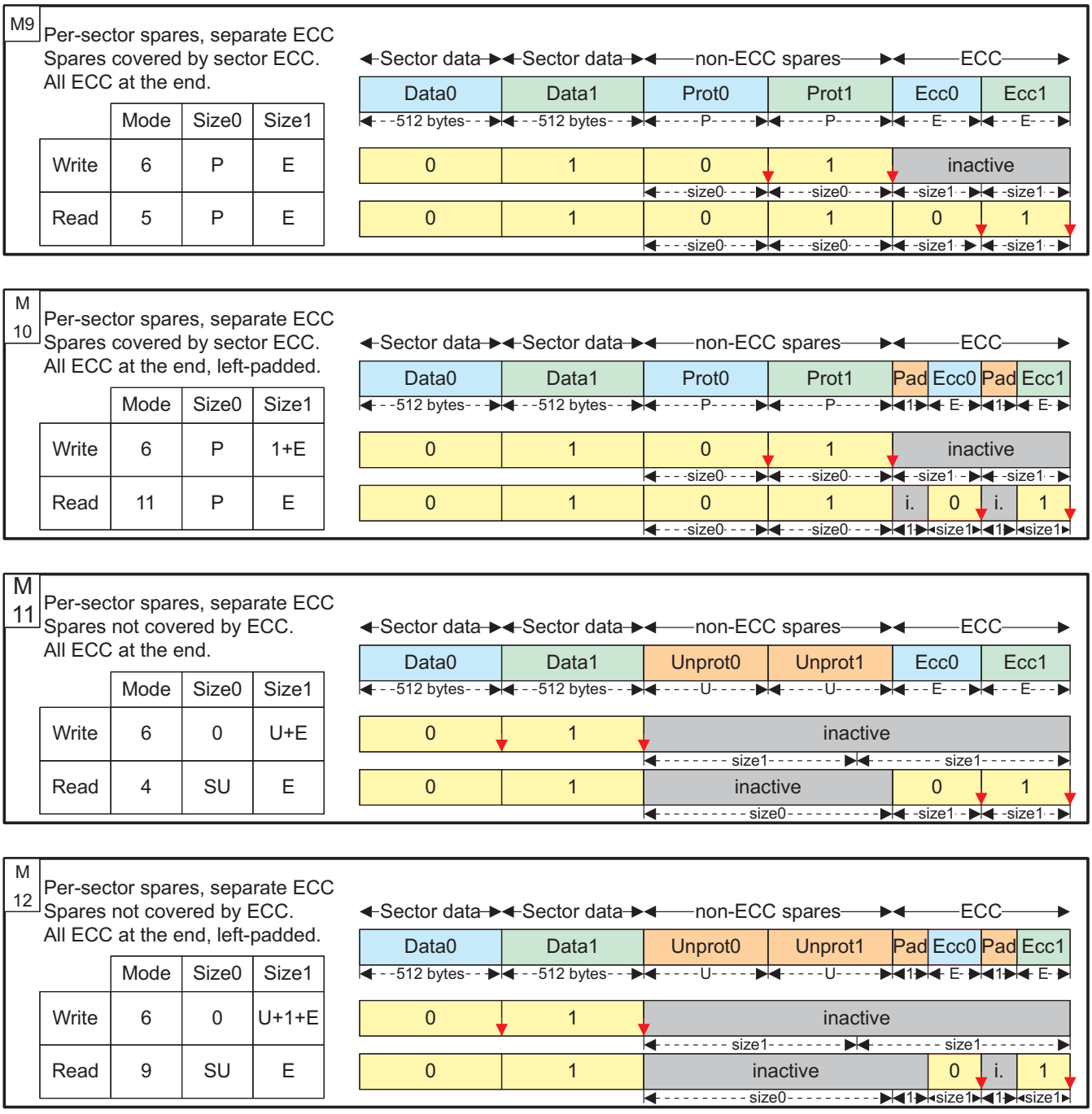


9.2.4.12.3.3.3 Per-Sector Spare Mapping, With ECC Separated at the End of the Page

In these schemes (Figure 9-40), each 512-byte sector of the main area is associated with two sections of the spare area.

- ECC section, all aligned at the end of the page
- other data section, aligned before the ECCs, each of which may or may not be protected by its sectors ECC

Figure 9-40. NAND Page Mapping and ECC: Per-Sector Schemes, with Separate ECC



9.2.4.12.4 Prefetch and Write-Posting Engine

NAND device data access cycles are usually much slower than the MPU system frequency; such NAND read or write accesses issued by the processor will impact the overall system performance, especially considering long read or write sequences required for NAND page loading or programming. To minimize this effect on system performance, the GPMC includes a prefetch and write-posting engine, which can be used to read from or write to any chip-select location in a buffered manner.

The prefetch and write-posting engine is a simplified embedded-access requester that presents requests to the access engine on a user-defined chip-select target. The access engine interleaves these requests with any request coming from the L3 interface; as a default the prefetch and write-posting engine has the lowest priority.

The prefetch and write-posting engine is dedicated to data-stream access (as opposed to random data access); thus, it is primarily dedicated to NAND support. The engine does not include an address generator; the request is limited to chip-select target identification. It includes a 64-byte FIFO associated with a DMA request synchronization line, for optimal DMA-based use.

The prefetch and write-posting engine uses an embedded 64 bytes (32 16-bit word) FIFO to prefetch data from the NAND device in read mode (prefetch mode) or to store host data to be programmed into the NAND device in write mode (write-posting mode). The FIFO draining and filling (read and write) can be controlled either by the MPU through interrupt synchronization (an interrupt is triggered whenever a programmable threshold is reached) or the sDMA through DMA request synchronization, with a programmable request byte size in both prefetch or posting mode.

The prefetch and write-posting engine includes a single memory pool. Therefore, only one mode, read or write, can be used at any given time. In other words, the prefetch and write-posting engine is a single-context engine that can be allocated to only one chip-select at a time for a read prefetch or a write-posting process.

The engine does not support atomic command and address phase programming and is limited to linear memory read or write access. In consequence, it is limited to NAND data-stream access. The engine relies on the MPU NAND software driver to control block and page opening with the correct data address pointer initialization, before the engine can read from or write to the NAND memory device.

Once started, the engine data reads and writes sequencing is solely based on FIFO location availability and until the total programmed number of bytes is read or written.

Any host-concurrent accesses to a different chip-select are correctly interleaved with ongoing engine accesses. The engine has the lowest priority access so that host accesses to a different chip-select do not suffer a large latency.

A round-robin arbitration scheme can be enabled to ensure minimum bandwidth to the prefetch and write-posting engine in the case of back-to-back direct memory requests to a different chip-select. If the GPMC_PREFETCH_CONFIG1 PFPWENROUNDROBIN bit is enabled, the arbitration grants the prefetch and write posting engine access to the GPMC bus for a number of requests programmed in the GPMC_PREFETCH_CONFIG1 PFPWWEIGHTEDPRIO field.

The prefetch/write-posting engine read or write request is routed to the access engine with the chip-select destination ID. After the required arbitration phase, the access engine processes the request as a single access with the data access size equal to the device size specified in the corresponding chip-select configuration.

The destination chip-select configuration must be set to the NAND protocol-compatible configuration for which address lines are not used (the address bus is not changed from its current value). Selecting a different chip-select configuration can produce undefined behavior.

9.2.4.12.4.1 General Facts About the Engine Configuration

The engine can be configured only if the GPMC_PREFETCH_CONTROL STARTENGINE bit is de-asserted.

The engine must be correctly configured in prefetch or write-posting mode and must be linked to a NAND chip-select before it can be started. The chip-select is linked using the GPMC_PREFETCH_CONFIG1 ENGINECSSELECTOR field.

In both prefetch and write-posting modes, the engine, respectively, uses byte or 16-bit word access requests for an 8- or 16-bit wide NAND device attached to the linked chip-select. The FIFOTHRESHOLD and TRANSFERCOUNT fields must be programmed accordingly as a number of bytes or a number of 16-bit word.

When the GPMC_PREFETCH_CONFIG1 ENABLEENGINE bit is set, the FIFO entry on the L3 interconnect port side is accessible at any address in the associated chip-select memory region. When the ENABLEENGINE bit is set, any host access to this chip-select is rerouted to the FIFO input. Directly accessing the NAND device linked to this chip-select from the host is still possible through these registers (where i = 0 to 5):

- GPMC_NAND_COMMAND_i
- GPMC_NAND_ADDRESS_i
- GPMC_NAND_DATA_i

The FIFO entry on the L3 interconnect port can be accessed with Byte, 16-bit word, or 32-bit word access size, according to little-endian format, even though the FIFO input is 32-bit wide.

The FIFO control is made easier through the use of interrupts or DMA requests associated with the FIFOTHRESHOLD bit field. The GPMC_PREFETCH_STATUS FIFOPOINTER field monitors the number of available bytes to be read in prefetch mode or the number of free empty slots which can be written in write-posting mode. The GPMC_PREFETCH_STATUS COUNTVALUE field monitors the number of remaining bytes to be read or written by the engine according to the TRANSFERCOUNT value. The FIFOPOINTER and COUNTVALUE bit fields are always expressed as a number of bytes even if a 16-bit wide NAND device is attached to the linked chip-select.

In prefetch mode, when the FIFOPOINTER equals 0, that is, the FIFO is empty, a host read access receives the byte last read from the FIFO as its response. In case of 32-bit word or 16-bit word read accesses, the last byte read from the FIFO is copied the required number of times to fit the requested word size. In write-posting mode, when the FIFOPOINTER equals 0, that is, the FIFO is full, a host write overwrites the last FIFO byte location. There is no underflow or overflow error reporting in the GPMC.

9.2.4.12.4.2 Prefetch Mode

The prefetch mode is selected when the GPMC_PREFETCH_CONFIG1 ACCESSMODE bit is cleared.

The MPU NAND software driver must issue the block and page opening (READ) command with the correct data address pointer initialization before the engine can be started to read from the NAND memory device. The engine is started by asserting the GPMC_PREFETCH_CONTROL STARTENGINE bit. The STARTENGINE bit automatically clears when the prefetch process completes.

If required, the ECC calculator engine must be initialized (reset, configured, and enabled) before the prefetch engine is started, so that the ECC is correctly computed on all data read by the prefetch engine.

When the GPMC_PREFETCH_CONFIG1 SYNCHROMODE bit is cleared, the prefetch engine starts requesting data as soon as the STARTENGINE bit is set. If using this configuration, the host must monitor the NAND device-ready pin so that it only sets the STARTENGINE bit when the NAND device is in a ready state, meaning data is valid for prefetching.

When the SYNCHROMODE bit is set, the prefetch engine starts requesting data when an active to inactive wait signal transition is detected. The transition detector must be cleared before any transition detection; see Section 9.2.4.12.2.2. The GPMC_PREFETCH_CONFIG1 WAITPINSELECTOR field selects which gpmc_wait pin edge detector triggers the prefetch engine in this synchronized mode.

If the STARTENGINE bit is set after the NAND address phase (page opening command), the engine is effectively started only after the actual NAND address phase completion. To prevent GPMC stall during this NAND address phase, set the STARTENGINE bit field before NAND address phase completion when in synchronized mode. The prefetch engine will start when an active to inactive wait signal transition is detected. The STARTENGINE bit is automatically cleared on prefetch process completion.

The prefetch engine issues a read request to fill the FIFO with the amount of data specified by GPMC_PREFETCH_CONFIG2 TRANSFERCOUNT field.

Table 9-22 describes the prefetch mode configuration.

Table 9-22. Prefetch Mode Configuration

Bit Field	Register	Value	Comments
STARTENGINE	GPMC_PREFETCH_CONTROL	0	Prefetch engine can be configured only if STARTENGINE is cleared to 0.
ENGINECSSELECTOR	GPMC_PREFETCH_CONFIG1	0 to 3h	Selects the chip-select associated with a NAND device where the prefetch engine is active.
ACCESSMODE	GPMC_PREFETCH_CONFIG1	0	Selects prefetch mode
FIFOTHRESHOLD	GPMC_PREFETCH_CONFIG1		Selects the maximum number of bytes read or written by the host on DMA or interrupt request
TRANSFERCOUNT	GPMC_PREFETCH_CONFIG1		Selects the number of bytes to be read or written by the engine to the selected chip-select
SYNCHROMODE	GPMC_PREFETCH_CONFIG1	0/1	Selects when the engine starts the access to the chip-select
WAITPINSELECT	GPMC_PREFETCH_CONFIG1	0 to 1	Selects wait pin edge detector (if GPMC_PREFETCH_CONFIG1 SYNCHROMODE = 1)
ENABLEOPTIMIZEDACCESS	GPMC_PREFETCH_CONFIG1	0/1	See Section 9.2.4.12.4.6
CYCLEOPTIMIZATION	GPMC_PREFETCH_CONFIG1		Number of clock cycle removed to timing parameters
ENABLEENGINE	GPMC_PREFETCH_CONFIG1	1	Engine enabled
STARTENGINE	GPMC_PREFETCH_CONFIG1	1	Starts the prefetch engine

9.2.4.12.4.3 FIFO Control in Prefetch Mode

The FIFO can be drained directly by the MPU or by an eDMA channel.

In MPU draining mode, the FIFO status can be monitored through the GPMC_PREFETCH_STATUS FIFOPOINTER field or through the GPMC_PREFETCH_STATUS FIFOTHRESHOLDSTATUS bit. The FIFOPOINTER indicates the current number of available data to be read; FIFOTHRESHOLDSTATUS set to 1 indicates that at least FIFOTHRESHOLD bytes are available from the FIFO.

An interrupt can be triggered by the GPMC if the GPMC_IRQENABLE FIFOEVENTENABLE bit is set. The FIFO interrupt event is logged, and the GPMC_IRQSTATUS FIFOEVENTSTATUS bit is set. To clear the interrupt, the MPU must read all the available bytes, or at least enough bytes to get below the programmed FIFO threshold, and the FIFOEVENTSTATUS bit must be cleared to enable further interrupt events. The FIFOEVENTSTATUS bit must always be reset prior to asserting the FIFOEVENTENABLE bit to clear any out-of-date logged interrupt event. This interrupt generation must be enabled after enabling the STARTENGINE bit.

Prefetch completion can be monitored through the GPMC_PREFETCH_STATUS COUNTVALUE field. COUNTVALUE indicates the number of currently remaining data to be requested according to the TRANSFERCOUNT value. An interrupt can be triggered by the GPMC when the prefetch process is complete (that is, COUNTVALUE equals 0) if the GPMC_IRQENABLE TERMINALCOUNTEVENTENABLE bit is set. At prefetch completion, the TERMINALCOUNT interrupt event is also logged, and the GPMC_IRQSTATUS TERMINALCOUNTSTATUS bit is set. To clear the interrupt, the MPU must clear the TERMINALCOUNTSTATUS bit. The TERMINALCOUNTSTATUS bit must always be cleared prior to asserting the TERMINALCOUNTEVENTENABLE bit to clear any out-of-date logged interrupt event.

NOTE: The COUNTVALUE value is only valid when the prefetch engine is active (started), and an interrupt is only triggered when COUNTVALUE reaches 0, that is, when the prefetch engine automatically goes from an active to an inactive state.

The number of bytes to be prefetched (programmed in TRANSFERCOUNT) must be a multiple of the programmed FIFOTHRESHOLD to trigger the correct number of interrupts allowing a deterministic and transparent FIFO control. If this guideline is respected, the number of ISR accesses is always required and the FIFO is always empty after the last interrupt is triggered. In other cases, the TERMINALCOUNT interrupt must be used to read the remaining bytes in the FIFO (the number of remaining bytes being lower than the FIFOTHRESHOLD value).

In DMA draining mode, the GPMC_PREFETCH_CONFIG1 DMAMODE bit must be set so that the GPMC issues a DMA hardware request when at least FIFOTHRESHOLD bytes are ready to be read from the FIFO. The DMA channel owning this DMA request must be programmed so that the number of bytes programmed in FIFOTHRESHOLD is read from the FIFO during the DMA request process. The DMA request is kept active until this number of bytes has effectively been read from the FIFO, and no other DMA request can be issued until the ongoing active request is complete.

In prefetch mode, the TERMINALCOUNT event is also a source of DMA requests if the number of bytes to be prefetched is not a multiple of FIFOTHRESHOLD, the remaining bytes in the FIFO can be read by the DMA channel using the last DMA request. This assumes that the number of remaining bytes to be read is known and controlled through the DMA channel programming model.

Any potentially active DMA request is cleared when the prefetch engine goes from inactive to active prefetch (the STARTENGINE bit is set to 1). The associated DMA channel must always be enabled by the MPU after setting the STARTENGINE bit so that the out-of-date active DMA request does not trigger spurious DMA transfers.

9.2.4.12.4.4 Write-Posting Mode

The write-posting mode is selected when the GPMC_PREFETCH_CONFIG1 ACCESSMODE bit is set.

The MPU NAND software driver must issue the correct address pointer initialization command (page program) before the engine can start writing data into the NAND memory device. The engine starts when the GPMC_PREFETCH_CONTROL STARTENGINE bit is set to 1. The STARTENGINE bit clears automatically when posting completes. When all data have been written to the NAND memory device, the MPU NAND software driver must issue the second cycle program command and monitor the status for programming process completion (adding ECC handling, if required).

If used, the ECC calculator engine must be started (configured, reset, and enabled) before the posting engine is started so that the ECC parities are properly calculated on all data written by the prefetch engine to the associated chip-select.

In write-posting mode, the GPMC_PREFETCH_CONFIG1 SYNCHROMODE bit must be cleared so that posting starts as soon as the STARTENGINE bit is set and the FIFO is not empty.

If the STARTENGINE bit is set after the NAND address phase (page program command), the STARTENGINE setting is effective only after the actual NAND command completion. To prevent GPMC stall during this NAND command phase, set the STARTENGINE bit field before the NAND address completion and ensure that the associated DMA channel is enabled after the NAND address phase.

The posting engine issues a write request when valid data are available from the FIFO and until the programmed GPMC_PREFETCH_CONFIG2 TRANSFERCOUNT accesses have been completed.

The STARTENGINE bit clears automatically when posting completes. When all data have been written to the NAND memory device, the MPU NAND software driver must issue the second cycle program command and monitor the status for programming process completion. The closing program command phase must only be issued when the full NAND page has been written into the NAND flash write buffer, including the spare area data and the ECC parities, if used.

Table 9-23. Write-Posting Mode Configuration

Bit Field	Register	Value	Comments
STARTENGINE	GPMC_PREFETCH_CONTROL	0	Write-posting engine can be configured only if STARTENGINE is cleared to 0.
ENGINECSSELECTOR	GPMC_PREFETCH_CONFIG1	0 to 3h	Selects the chip-select associated with a NAND device where the prefetch engine is active
ACCESSMODE	GPMC_PREFETCH_CONFIG1	1	Selects write-posting mode
FIFOTHRESHOLD	GPMC_PREFETCH_CONFIG1		Selects the maximum number of bytes read or written by the host on DMA or interrupt request
TRANSFERCOUNT	GPMC_PREFETCH_CONFIG2		Selects the number of bytes to be read or written by the engine from/to the selected chip-select
SYNCHROMODE	GPMC_PREFETCH_CONFIG1	0	Engine starts the access to chip-select as soon as STARTENGINE is set.
ENABLEOPTIMIZEDACCESS	GPMC_PREFETCH_CONFIG1	0/1	See Section 9.2.4.12.4.6
CYCLEOPTIMIZATION	GPMC_PREFETCH_CONFIG		
ENABLEENGINE	GPMC_PREFETCH_CONFIG1	1	Engine enabled
STARTENGINE	GPMC_PREFETCH_CONTROL	1	Starts the prefetch engine

9.2.4.12.4.5 FIFO Control in Write-Posting Mode

The FIFO can be filled directly by the MPU or by an sDMA channel.

In MPU filling mode, the FIFO status can be monitored through the FIFOPOINTER or through the GPMC_PREFETCH_STATUS FIFOTHRESHOLDSTATUS bit. FIFOPOINTER indicates the current number of available free byte places in the FIFO, and the FIFOTHRESHOLDSTATUS bit, when set, indicates that at least FIFOTHRESHOLD free byte places are available in the FIFO.

An interrupt can be issued by the GPMC if the GPMC_IRQENABLE FIFOEVENTENABLE bit is set. When the interrupt is fired, the GPMC_IRQSTATUS FIFOEVENTSTATUS bit is set. To clear the interrupt, the MPU must write enough bytes to fill the FIFO, or enough bytes to get below the programmed threshold, and the FIFOEVENTSTATUS bit must be cleared to get further interrupt events. The FIFOEVENTSTATUS bit must always be cleared prior to asserting the FIFOEVENTENABLE bit to clear any out-of-date logged interrupt event. This interrupt must be enabled after enabling the STARTENGINE bit

The posting completion can be monitored through the GPMC_PREFETCH_STATUS COUNTVALUE field. COUNTVALUE indicates the current number of remaining data to be written based on the TRANSFERCOUNT value. An interrupt is issued by the GPMC when the write-posting process completes (that is, COUNTVALUE equal to 0) if the GPMC_IRQENABLE TERMINALCOUNTEVENTENABLE bit is set. When the interrupt is fired, the GPMC_IRQSTATUS TERMINALCOUNTSTATUS bit is set. To clear the interrupt, the MPU must clear the TERMINALCOUNTSTATUS bit. The TERMINALCOUNTSTATUS bit must always be cleared prior to asserting the TERMINALCOUNTEVENTENABLE bit to clear any out-of-date logged interrupt event.

NOTE: The COUNTVALUE value is only valid if the write-posting engine is active and started, and an interrupt is only issued when COUNTVALUE reaches 0, that is, when the posting engine automatically goes from active to inactive.

In DMA filling mode, the GPMC_PREFETCH_CONFIG1 DMAMODE bit must be set so that the GPMC issues a DMA hardware request when at least FIFOTHRESHOLD bytes-free places are available in the FIFO. The DMA channel owning this DMA request must be programmed so that a number of bytes equal to the value programmed in the FIFOTHRESHOLD bit field are written into the FIFO during the DMA access. The DMA request remains active until the associated number of bytes has effectively been written into the FIFO, and no other DMA request can be issued until the ongoing active request has been completed.

Any potentially active DMA request is cleared when the prefetch engine goes from inactive to active prefetch (STARTENGINE set to 1). The associated DMA channel must always be enabled by the MPU after setting the STARTENGINE bit so that an out-of-date active DMA request does not trigger spurious DMA transfers.

In write-posting mode, the DMA or the MPU fill the FIFO with no consideration to the associated byte enables. Any byte stored in the FIFO is written into the memory device.

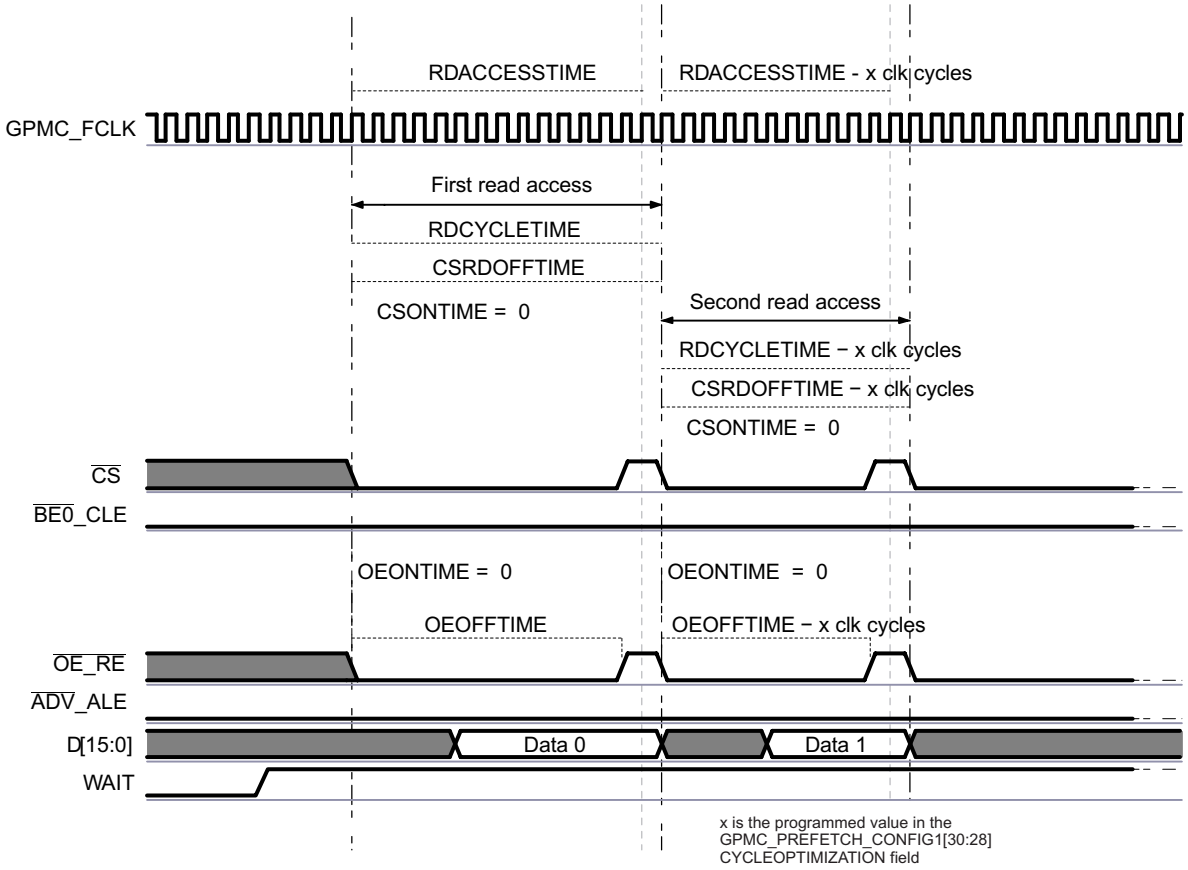
9.2.4.12.4.6 Optimizing NAND Access Using the Prefetch and Write-Posting Engine

Access time to a NAND memory device can be optimized for back-to-back accesses if the associated $\overline{CS}$ signal is not deasserted between accesses. The GPMC access engine can track prefetch engine accesses to optimize the access timing parameter programmed for the allocated chip-select, if no accesses to other chip-selects (that is, interleaved accesses) occur. Similarly, the access engine also eliminates the CYCLE2CYCLEDELAY even if CYCLE2CYCLESAMECSEN is set. This capability is limited to the prefetch and write-posting engine accesses, and MPU accesses to a NAND memory device (through the defined chip-select memory region or through the GPMC_NAND_DATA_i (where , i = 0 to 5) are never optimized.

The GPMC_PREFETCH_CONFIG1 ENABLEOPTIMIZEDACCESS bit must be set to enable optimized accesses. To optimize access time, the GPMC_PREFETCH_CONFIG1 CYCLEOPTIMIZATION field defines the number of GPMC_FCLK cycles to be suppressed from the RDCYCLETIME, WRCYCLETIME, RDACCESSTIME, WRACCESSTIME, CSOFFTIME, ADVOFFTIME, OEOFFTIME, and WEOFFTIME timing parameters.

Figure 9-41, in the case of back-to-back accesses to the NAND flash through the prefetch engine, CYCLE2CYCLESAMECSEN is forced to 0 when using optimized accesses. The first access uses the regular timing settings for this chip-select. All accesses after this one use settings reduced by x clock cycles, x being defined by the GPMC_PREFETCH_CONFIG1 CYCLEOPTIMIZATION field.

Figure 9-41. NAND Read Cycle Optimization Timing Description



9.2.4.12.4.7 Interleaved Accesses Between Prefetch and Write-Posting Engine and Other Chip-Selects

Any on-going read or write access from the prefetch and write-posting engine is completed before an access to any other chip-select can be initiated. As a default, the arbiter uses a fixed-priority algorithm, and the prefetch and write-posting engine has the lowest priority. The maximum latency added to access starting time in this case equals the RDCYCLETIME or WRCYCLETIME (optimized or not) plus the requested BUSTURNAROUND delay for bus turnaround completion programmed for the chip-select to which the NAND device is connected to.

Alternatively, a round-robin arbitration can be used to prioritize accesses to the external bus. This arbitration scheme is enabled by setting the GPMC_PREFETCH_CONFIG1 PFPWENROUNDROBIN bit. When a request to another chip-select is received while the prefetch and write-posting engine is active, priority is given to the new request. The request processed thereafter is the prefetch and write-posting engine request, even if another interconnect request is passed in the mean time. The engine keeps control of the bus for an additional number of requests programmed in the GPMC_PREFETCH_CONFIG1 PFPWWEIGHTEDPRIO field. Control is then passed to the direct interconnect request.

As an example, the round-robin arbitration scheme is selected with PFPWWEIGHTEDPRIO set to 2h. Considering the prefetch and write-posting engine and the interconnect interface are always requesting access to the external interface, the GPMC grants priority to the direct interconnect access for one request. The GPMC then grants priority to the engine for three requests, and finally back to the direct interconnect access, until the arbiter is reset when one of the two initiators stops initiating requests.

9.3 Basic Programming Model

9.3.1 GPMC High-Level Programming Model Overview

The high-level programming model introduces a top-down approach for users that need to configure the GPMC module. Figure 9-42 shows a programming model top-level diagram for the GPMC. Each block of the diagram is described in one of the following subsections through a set of registers to configure. Table 9-24 and Table 9-25 list each step in the model.

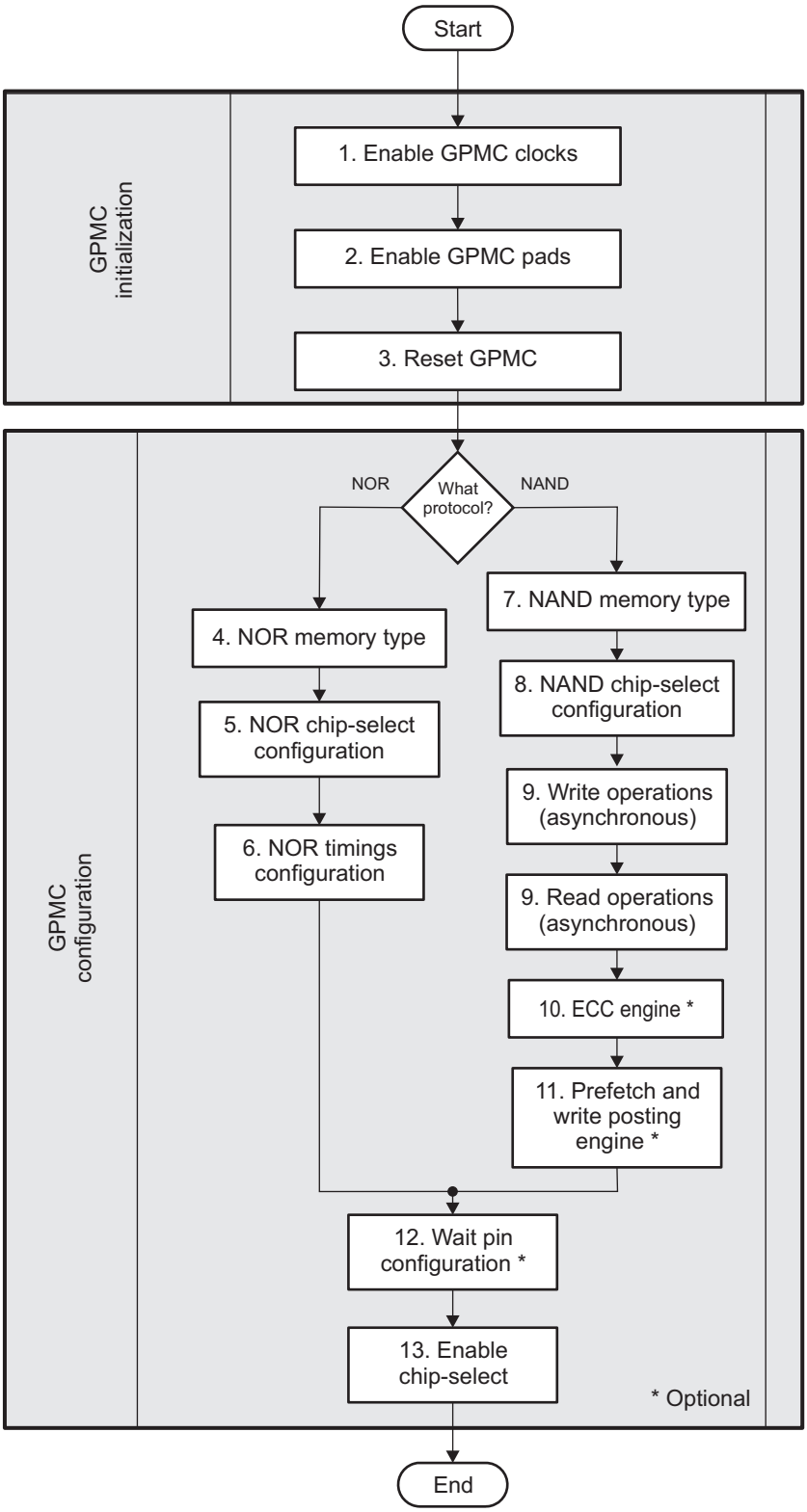
Table 9-24. GPMC Configuration in NOR Mode

Step	Description
NOR Memory Type	See Table 9-27
NOR Chip-Select Configuration	See Table 9-28
NOR Timings Configuration	See Table 9-29
Wait Pin Configuration	See Table 9-30
Enable Chip-Select	See Table 9-31

Table 9-25. GPMC Configuration in NAND Mode

Step	Description
NAND Memory Type	See Table 9-32
NAND Chip-Select Configuration	See Table 9-33
Write Operations (Asynchronous)	See Table 9-34
Read Operations (Asynchronous)	See Table 9-34
ECC Engine	See Table 9-35
Prefetch and Write-Posting Engine	See Table 9-36
Wait Pin Configuration	See Table 9-37
Enable Chip-Select	See Table 9-38

Figure 9-42. Programming Model Top-Level Diagram



9.3.2 GPMC Initialization

Table 9-26 describes the settings required to reset the GPMC.

Table 9-26. Reset GPMC

Sub-process Name	GPMC_SYSCONFIG Register Bit	Value
Start a software reset	SOFTRESET	1
Wait until	RESETDONE	1

9.3.3 GPMC Configuration in NOR Mode

This section gives a generic configuration for parameters related to the NOR memory connected to the GPMC. Table 9-27 through Table 9-31 list the steps to configure the GPMC in NOR mode.

NOTE: In the tables of this section, 'x' in Value column stands for 'depends on configuration'.

Table 9-27. NOR Memory Type

Sub-process Name	GPMC_CONFIG1_i Register Bitfield	Value
Set the NOR protocol	DEVICETYPE	0
Set a device size	DEVICESIZE	x
Select an address and data multiplexing protocol	MUXADDDATA	x
Set the attached device page length	ATTACHEDDEVICEPAGELENGTH	x
Set the wrapping burst capabilities	WRAPBURST	x
Select a timing signals latencies factor	TIMEPARAGRANULARITY	x
Select an output clock frequency	GPMCFCLKDIVIDER	x
Choose an output clock activation time	CLKACTIVATIONTIME	x
Set a single or multiple access for read operations	READMULTIPLE	x
Set a synchronous or asynchronous mode for read operations	READTYPE	x
Set a single or multiple access for write operations	WRITEMULTIPLE	x
Set a synchronous or asynchronous mode for write operations	WRITETYPE	x

Table 9-28. NOR Chip-Select Configuration

Sub-process Name	GPMC_CONFIG7_i Register Bitfield	Value
Select the chip-select base address	BASEADDRESS	x
Select the chip-select mask address	MASKADDRESS	x

Table 9-29. NOR Timings Configuration

Sub-process Name	Register Bitfield	Value
Configure adequate timing parameters in various memory modes	See Section 9.3.6	

Table 9-30. WAIT Pin Configuration

Sub-process Name	GPMC_CONFIG1_i Register Bitfield	Value
Enable or disable wait pin monitoring for read operations	WAITREADMONITORING	x
Enable or disable wait pin monitoring for write operations	WAITWRITEMONITORING	x
Select a wait pin monitoring time	WAITMONITORINGTIME	x
Choose the input wait pin for the chip-select	WAITPINSELECT	x

Table 9-31. Enable Chip-Select

Sub-process Name	GPMC_CONFIG7_i Register Bitfield	Value
When all parameters are configured, enable the chip-select	CSVALID	x

9.3.4 GPMC Configuration in NAND Mode

This section gives a generic configuration for parameters related to NAND memory connected to the GPMC.

Table 9-32. NAND Memory Type

Sub-process Name	GPMC_CONFIG1_i Register Bitfield	Value
Set the NAND protocol	DEVICETYPE	2h
Set a device size	DEVICESIZE	x
Set the address and data multiplexing protocol to non-multiplexed attached device	MUXADDDATA	0
Select a timing signals latencies factor	TIMEPARAGRANULARITY	x
Set a synchronous or asynchronous mode and a single or multiple access for read and write operations	See Section 9.3.5	x

Table 9-33. NAND Chip-Select Configuration

Sub-process Name	GPMC_CONFIG7_i Register Bitfield	Value
Select the chip-select base address	BASEADDRESS	x
Select the chip-select minimum granularity (16M bytes)	MASKADDRESS	x

Table 9-34. Asynchronous Read and Write Operations

Sub-process Name	Register Bitfield	Value
Configure adequate timing parameters in asynchronous modes	See Section 9.3.6	

Table 9-35. ECC Engine

Sub-process Name	Register Bitfield	Value
Select the ECC result register where the first ECC computation is stored (Only applies to Hamming)	GPMC_ECC_CONTROL ECCPOINTER	x
Clear all ECC result registers	GPMC_ECC_CONTROL ECCCLEAR	Write 1 to clear
Define ECCSIZE0 and ECCSIZE1	GPMC_ECC_SIZE_CONFIG ECCSIZE0 and ECCSIZE1	x
Select the size of each of the 9 result registers (size specified by ECCSIZE0 or ECCSIZE1)	GPMC_ECC_SIZE_CONFIG[j-1] ECCjRESULTSIZE, where j = 1 to 9	x
Select the chip-select where ECC is computed	GPMC_ECC_SIZE_CONFIG ECCCS	x
Select the Hamming code or BCH code ECC algorithm in use	GPMC_ECC_SIZE_CONFIG ECCALGORITHM	x
Select word size for ECC calculation	GPMC_ECC_SIZE_CONFIG ECC16B	x
If the BCH code is used, Set an error correction capability and Select a number of sectors to process	GPMC_ECC_SIZE_CONFIG ECCBCHTSEL and ECCTOPSECTOR	x
Enable the ECC computation	GPMC_ECC_SIZE_CONFIG ECCENABLE	1

Table 9-36. Prefetch and Write-Posting Engine

Sub-process Name	Register Bitfield	Value
Disable the engine before configuration	GPMC_PREFETCH_CONTROL STARTENGINE	0
Select the chip-select associated with a NAND device where the prefetch engine is active	GPMC_PREFETCH_CONFIG1 ENGINECSSELECTOR	x
Select access direction through prefetch engine, read or write.	GPMC_PREFETCH_CONFIG1 ACCESSMODE	x
Select the threshold used to issue a DMA request	GPMC_PREFETCH_CONFIG1 FIFOTHRESHOLD	x
Select either DMA synchronized mode or SW manual mode.	GPMC_PREFETCH_CONFIG1 DMAMODE	x
Select if the engine immediately starts accessing the memory upon STARTENGINE assertion or if hardware synchronization based on a WAIT signal is used.	GPMC_PREFETCH_CONFIG1 SYNCHROMODE	x
Select which wait pin edge detector should start the engine in synchronized mode	GPMC_PREFETCH_CONFIG1 WAITPINSELECTOR	x
Enter a number of clock cycles removed to timing parameters (For all back-to-back accesses to the NAND flash but not the first one)	GPMC_PREFETCH_CONFIG1 CYCLEOPTIMIZATION	x
Enable the prefetch postwite engine	GPMC_PREFETCH_CONFIG1 ENABLEENGINE	1
Select the number of bytes to be read or written by the engine to the selected chip-select	GPMC_PREFETCH_CONFIG2 TRANSFERCOUNT	x
Start the prefetch engine	GPMC_PREFETCH_CONTROL STARTENGINE	1

Table 9-37. WAIT Pin Configuration

Sub-process Name	GPMC_PREFETCH_CONFIG1 Register Bitfield	Value
Selects when the engine starts the access to CS	SYNCHROMODE	x
Select which wait pin edge detector should start the engine in synchronized mode	WAITPINSELECTOR	x

Table 9-38. Enable Chip-Select

Sub-process Name	GPMC_CONFIG7_i Register Bitfield	Value
When all parameters are configured, enable the chip-select	CSVALID	x

9.3.5 Set Memory Access

This section details the bit field to configure to set the GPMC in various memory modes.

Table 9-39. Mode Parameters Check List Table

GPMC_CONFIG1_i Register Bitfield	Asynchronous				Synchronous			
	Single Read Access	Single Write Access	Multiple Read (Page) Access	Multiple Write (Page) Access	Single Read Access	Single Write Access	Multiple Read (Burst) Access	Multiple Write (Burst) Access
READMULTIPLE	0	-	1	N/S	0	-	1	-
READTYPE	0	-	0	N/S	1	-	1	-
WRITEMULTIPLE	-	0		N/S	-	0	-	1
WRITETYPE	-	0		N/S	-	1	-	1

Table 9-40. Access Type Parameters Check List Table

GPMC_CONFIG1_i Register Bitfield	Access Type		
	Non-Mux	Address/Data Mux	AAD Mux
MUXADDDATA	0	2h	1

9.3.6 GPMC Timing Parameters

Figure 9-43 shows a programming model diagram for the NOR interfacing timing parameters.

Table 9-41 lists bit fields to configure adequate timing parameter in various memory modes.

Figure 9-43. NOR Interfacing Timing Parameters Diagram

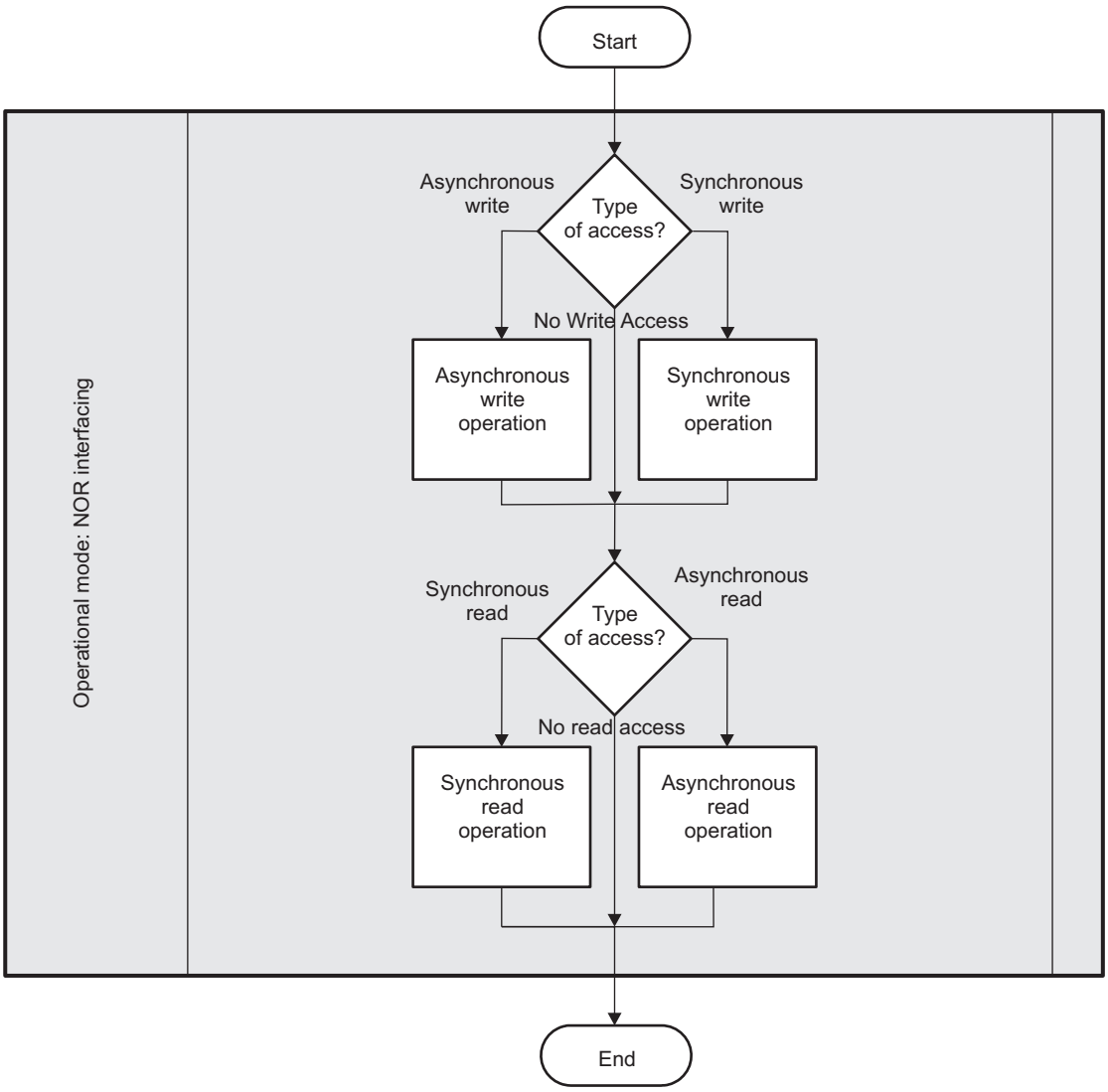


Table 9-41. Timing Parameters

Register	Bit	Bit Field Name	Asynchronous			Synchronous				Access Type		
			Single Read Access	Single Write Access	Multiple Read (Page) access	Single Read Access	Single Write Access	Multiple Read (Burst) Access	Multiple Write (Burst) Access	Non-multiplexed	Address /Data-multiplexed	AAD-multiplexed
GPMC_CONFIG1_i	9-8	MUXADDDATA	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG1_i	29	READTYPE	y		y	y		y		y	y	y
GPMC_CONFIG1_i	30	READMULTIPLE	y		y	y		y		y	y	y
GPMC_CONFIG1_i	27	WRITETYPE		y			y		y	y	y	y
GPMC_CONFIG1_i	28	WRITEMULTIPLE		y			y		y	y	y	y
GPMC_CONFIG1_i	31	WRAPBURST						y	y	y	y	y
GPMC_CONFIG1_i	26-25	CLKACTIVATIONTIME				y	y	y	y	y	y	y
GPMC_CONFIG1_i	19-18	WAITMONITORINGTIME	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG1_i	4	TIMEPARAGRANULARITY	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG2_i	20-16	CSWROFFTIME		y			y		y	y	y	y
GPMC_CONFIG2_i	12-8	CSRDOFFTIME	y		y	y		y		y	y	y
GPMC_CONFIG2_i	7	CSEXTRADELAY	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG2_i	3-0	CSONTIME	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG3_i	30-28	ADVAADMUXWROFFTIME		y			y		y			y
GPMC_CONFIG3_i	26-24	ADVAADMUXRDOFFTIME	y		y	y		y				y
GPMC_CONFIG3_i	6-4	ADVAADMUXONTIME	y	y	y	y	y	y	y			y
GPMC_CONFIG3_i	20-16	ADVWROFFTIME		y			y		y	y	y	y
GPMC_CONFIG3_i	12-8	ADVRDOFFTIME	y		y	y		y		y	y	y
GPMC_CONFIG3_i	7	ADVEXTRADELAY	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG3_i	3-0	ADVONTIME	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG4_i	15-13	OEAADMUXOFFTIME	y	y	y	y	y	y	y			y
GPMC_CONFIG4_i	6-4	OEAADMUXONTIME	y	y	y	y	y	y	y			y
GPMC_CONFIG4_i	28-24	WEOFFTIME		y			y		y	y	y	y
GPMC_CONFIG4_i	23	WEEXTRADELAY		y			y		y	y	y	y
GPMC_CONFIG4_i	19-16	WEONTIME		y			y		y	y	y	y
GPMC_CONFIG4_i	12-8	OEOFFTIME	y		y	y		y		y	y	y
GPMC_CONFIG4_i	7	OEEXTRADELAY	y		y	y		y		y	y	y
GPMC_CONFIG4_i	3-0	OEONTIME	y		y	y		y		y	y	y
GPMC_CONFIG5_i	27-24	PAGEBURSTACCESSTIME			y			y	y	y	y	y
GPMC_CONFIG5_i	20-16	RDACCESSTIME	y		y	y		y		y	y	y
GPMC_CONFIG5_i	12-8	WRCYCLETIME		y			y		y	y	y	y
GPMC_CONFIG5_i	4-0	RDCYCLETIME	y		y	y		y		y	y	y
GPMC_CONFIG6_i	28-24	WRACCESSTIME		y			y		y	y	y	y
GPMC_CONFIG6_i	19-16	WRDATAONADMUXBUS		y			y		y		y	y

Table 9-41. Timing Parameters (continued)

Register	Bit	Bit Field Name	Asynchronous			Synchronous				Access Type		
			Single Read Access	Single Write Access	Multiple Read (Page) access	Single Read Access	Single Write Access	Multiple Read (Burst) Access	Multiple Write (Burst) Access	Non-multiplexed	Address /Data-multiplexed	AAD-multiplexed
GPMC_CONFIG6_i	11-8	CYCLE2CYCLEDELAY	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG6_i	7	CYCLE2CYCLESAMECSEN	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG6_i	6	CYCLE2CYCLEDIFFCSEN	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG6_i	3-0	BUSTURNAROUND	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG7_i	6	CSVALID	y	y	y	y	y	y	y	y	y	y

9.3.6.1 GPMC Timing Parameters Formulas

This section intends to help the user to calculate the GPMC timing bit fields values. Formulas are not listed exhaustively.

The section details:

- NAND Flash Interface Timing Parameters Formulas
- Synchronous NOR Flash Timing Parameters Formulas
- Asynchronous NOR Flash Timing Parameters Formulas

9.3.6.1.1 NAND Flash Interface Timing Parameters Formulas

This section lists formulas to use in order to calculate NAND timing parameters. This is the case when GPMC_CONFIG1_i DEVICETYPE = 2h. Table 9-42 details NAND timing parameters.

Table 9-42. NAND Formulas Description Table

Configuration Parameter	Unit	Description
A	ns	Pulse duration - GPMC_WE valid time
B	ns	Delay time - GPMC_CS valid to GPMC_WE valid
C	ns	Delay time - GPMC_BE0_CLE/GPMC_ADV_ALE high to GPMC_WE valid
D	ns	Delay time - GPMC_AD[15:0] valid to GPMC_WE valid
E	ns	Delay time - GPMC_WE invalid to GPMC_AD[15:0] invalid
F	ns	Delay time - GPMC_WE invalid to GPMC_BE0_CLE/GPMC_ADV_ALE invalid
G	ns	Delay time - GPMC_WE invalid to GPMC_CS invalid
H	ns	Cycle time - Write cycle time
I	ns	Delay time - GPMC_CS valid to GPMC_OE valid
J	ns	Setup time - GPMC_AD[15:0] valid to GPMC_OE invalid
K	ns	Pulse duration - GPMC_OE valid time
L	ns	Cycle time - Read cycle time
M	ns	Delay time - GPMC_OE invalid to GPMC_CS invalid

The configuration parameters are calculated through the following formulas.

A = (WEOffTime - WEOnTime) × (TimeParaGranularity + 1) × GPMC_FCLK period

B = ((WEOnTime - CSOnTime) × (TimeParaGranularity + 1) + 0.5 × (WEExtraDelay - CSExtraDelay)) × GPMC_FCLK period

C = ((WEOnTime - ADVOnTime) × (TimeParaGranularity + 1) + 0.5 × (WEExtraDelay - ADVExtraDelay)) × GPMC_FCLK period

D = (WEOnTime × (TimeParaGranularity + 1) + 0.5 × WEExtraDelay) × GPMC_FCLK period

E = (WrCycleTime - WEOffTime × (TimeParaGranularity + 1) - 0.5 × WEExtraDelay) × GPMC_FCLK period

F = (ADVWrOffTime - WEOffTime × (TimeParaGranularity + 1) + 0.5 × (ADVExtraDelay - WEExtraDelay) × GPMC_FCLK period

G = (CSWrOffTime - WEOffTime × (TimeParaGranularity + 1) + 0.5 × (CSExtraDelay - WEExtraDelay) × GPMC_FCLK period

H = WrCycleTime × (1 + TimeParaGranularity) × GPMC_FCLK period

I = ((OEOnTime - CSOnTime) × (TimeParaGranularity + 1) + 0.5 × (OEExtraDelay - CSExtraDelay)) × GPMC_FCLK period

J = ((RdAccessTime - OEOffTime) × (TimeParaGranularity + 1) - 0.5 × OEExtraDelay)) × GPMC_FCLK period

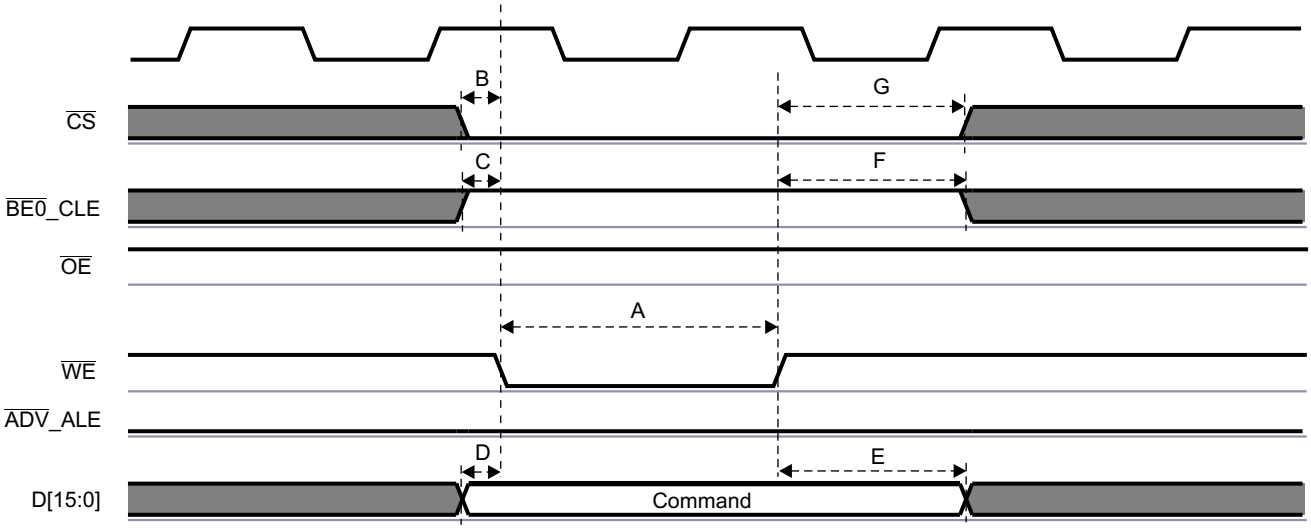
K = (OEOffTime - OEOnTime) × (1 + TimeParaGranularity) × GPMC_FCLK period

L = RdCycleTime × (1 + TimeParaGranularity) × GPMC_FCLK period

M = (CSRdOffTime - OEOffTime × (TimeParaGranularity + 1) + 0.5 × (CSExtraDelay - OEExtraDelay) × GPMC_FCLK period

Figure 9-44 shows a command latch cycle timing simplified example where formulas are associated with signal waves.

Figure 9-44. NAND Command Latch Cycle Timing Simplified Example



9.3.6.1.2 Synchronous NOR Flash Timing Parameters Formulas

This section lists all formulas to use in order to calculate synchronous NOR timing parameters. This is the case when GPMC_CONFIG1_i DEVICETYPE = 0 and when READTYPE or WRITETYPE are set to synchronous mode.

Table 9-43. Synchronous NOR Formulas Description Table

Configuration Parameter	Unit	Description
A	ns	Pulse duration - GPMC_CS low
B	ns	Delay time - address bus valid to GPMC_CLK first edge
		Delay time - GPMC_BE0_CLE/GPMC_BE1 valid to GPMC_CLK first edge
C	ns	Pulse duration - GPMC_BE0_CLE/GPMC_BE1 low
D	ns	Delay time - GPMC_CLK rising edge to GPMC_BE0_CLE/GPMC_BE1 invalid
		Delay time - GPMC_CLK rising edge to GPMC_ADV_ALE invalid
E	ns	Delay time - GPMC_CLK rising edge to GPMC_CS invalid
		Delay time - GPMC_CLK rising edge to GPMC_OE invalid
F	ns	Delay time - GPMC_CLK rising edge to GPMC_CS transition
G	ns	Delay time - GPMC_CLK rising edge to GPMC_ADV_ALE transition
H	ns	Delay time - GPMC_CLK rising edge to GPMC_OE transition
I	ns	Delay time - GPMC_CLK rising edge to GPMC_WE transition
J	ns	Delay time - GPMC_CLK rising edge to GPMC_AD data bus transition
		Delay time - GPMC_CLK rising edge to GPMC_BE0_CLE/GPMC_BE1 transition
K	ns	Pulse duration - GPMC_ADV_ALE low
L	ns	Delay time - GPMC_WAIT invalid to first data latching GPMC_CLK edge

The configuration parameters are calculated through the following formulas.

For single read accesses:

A = (CSRDOFFTIME - CSONTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period

C = RDCYCLETIME × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period

D = (RDCYCLETIME - RDACCESSTIME) × GPMC_FCLK period

E = (CSRDOFFTIME - RDACCESSTIME) × GPMC_FCLK period

For burst read accesses (where n is the page burst access number):

A = (CSRDOFFTIME - CSONTIME + (n - 1) × PAGEBURSTACCESSTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period

C = (RDCYCLETIME + (n - 1) × PAGEBURSTACCESSTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period

D = (RDCYCLETIME - (RDACCESSTIME + (n - 1) × PAGEBURSTACCESSTIME)) × GPMC_FCLK period

E = (CSRDOFFTIME - (RDACCESSTIME + (n - 1) × PAGEBURSTACCESSTIME)) × GPMC_FCLK period

For burst write accesses (where n is the page burst access number):

A = (CSWROFFTIME - CSONTIME + (n - 1) × PAGEBURSTACCESSTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period

C = (WRCYCLETIME + (n - 1) × PAGEBURSTACCESSTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period

D = (WRCYCLETIME - (WRACCESSTIME + (n - 1) × PAGEBURSTACCESSTIME)) × GPMC_FCLK period

E = (CSWROFFTIME - (WRACCESSTIME + (n - 1) × PAGEBURSTACCESSTIME)) × GPMC_FCLK period

For all accesses:

For $\overline{CS}$ falling edge ($\overline{CS}$ activated):

- Case where GPMC_CONFIG1_i GPMCFCLKDIVIDER = 0

  F = 0.5 × CSEXTRADELAY × GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 1h

  F = 0.5 × CSEXTRADELAY × GPMC_FCLK period, when (CLKACTIVATIONTIME and CSONTIME are odd) or (CLKACTIVATIONTIME and CSONTIME are even)

  F = (1 + 0.5 × CSEXTRADELAY) × GPMC_FCLK period, otherwise
- Case where GPMCFCLKDIVIDER = 2h

  F = 0.5 × CSEXTRADELAY × GPMC_FCLK period, when (CSONTIME - CLKACTIVATIONTIME) is a multiple of 3

  F = (1 + 0.5 × CSEXTRADELAY) × GPMC_FCLK period, when (CSONTIME - CLKACTIVATIONTIME - 1) is a multiple of 3

  F = (2 + 0.5 × CSEXTRADELAY) × GPMC_FCLK period, when (CSONTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For $\overline{CS}$ rising edge ($\overline{CS}$ de-activated) in reading mode:

- Case where GPMC_CONFIG1_i GPMCFCLKDIVIDER = 0

  F = 0.5 × CSEXTRADELAY × GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 1h

  F = 0.5 × CSEXTRADELAY × GPMC_FCLK period, when (CLKACTIVATIONTIME and CSRDOFFTIME are odd) or (CLKACTIVATIONTIME and CSRDOFFTIME are even)

  F = (1 + 0.5 × CSEXTRADELAY) × GPMC_FCLK period, otherwise
- Case where GPMCFCLKDIVIDER = 2h

  F = 0.5 × CSEXTRADELAY × GPMC_FCLK period, when (CSRDOFFTIME - CLKACTIVATIONTIME) is a multiple of 3

  F = (1 + 0.5 × CSEXTRADELAY) × GPMC_FCLK period, when (CSRDOFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3

  F = (2 + 0.5 × CSEXTRADELAY) × GPMC_FCLK period, when (CSRDOFFTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For $\overline{CS}$ rising edge ($\overline{CS}$ de-activated) in writing mode:

- Case where GPMC_CONFIG1_i GPMCFCLKDIVIDER = 0

  F = 0.5 × CSEXTRADELAY × GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 1h

  F = 0.5 × CSEXTRADELAY × GPMC_FCLK period, when (CLKACTIVATIONTIME and CSWROFFTIME are odd) or (CLKACTIVATIONTIME and CSWROFFTIME are even)

  F = (1 + 0.5 × CSEXTRADELAY) × GPMC_FCLK period, otherwise
- Case where GPMCFCLKDIVIDER = 2h

  F = 0.5 × CSEXTRADELAY × GPMC_FCLK period, when (CSWROFFTIME - CLKACTIVATIONTIME) is a multiple of 3

  F = (1 + 0.5 × CSEXTRADELAY) × GPMC_FCLK period, when (CSWROFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3

  F = (2 + 0.5 × CSEXTRADELAY) × GPMC_FCLK period, when (CSWROFFTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For $\overline{ADV}$ falling edge ($\overline{ADV}$ activated):

- Case where GPMC_CONFIG1_i GPMCFCLKDIVIDER = 0

  G = 0.5 × ADVEXTRADELAY × GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 1h

  G = 0.5 × ADVEXTRADELAY × GPMC_FCLK period, when (CLKACTIVATIONTIME and ADVONTIME are odd) or (CLKACTIVATIONTIME and ADVONTIME are even)

  G = (1 + 0.5 × ADVEXTRADELAY) × GPMC_FCLK period, otherwise
- Case where GPMCFCLKDIVIDER = 2h

  G = 0.5 × ADVEXTRADELAY × GPMC_FCLK period, when (ADVONTIME - CLKACTIVATIONTIME) is a multiple of 3

  G = (1 + 0.5 × ADVEXTRADELAY) × GPMC_FCLK period, when (ADVONTIME - CLKACTIVATIONTIME - 1) is a multiple of 3

  G = (2 + 0.5 × ADVEXTRADELAY) × GPMC_FCLK period, when (ADVONTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For $\overline{ADV}$ rising edge ($\overline{ADV}$ de-activated) in reading mode:

- Case where GPMC_CONFIG1_i GPMCFCLKDIVIDER = 0

  G = 0.5 × ADVEXTRADELAY × GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 1h

  G = 0.5 × ADVEXTRADELAY × GPMC_FCLK period, when (CLKACTIVATIONTIME and ADVRDOFFTIME are odd) or (CLKACTIVATIONTIME and ADVRDOFFTIME are even)

  G = (1 + 0.5 × ADVEXTRADELAY) × GPMC_FCLK period, otherwise
- Case where GPMCFCLKDIVIDER = 2h

  G = 0.5 × ADVEXTRADELAY × GPMC_FCLK period, when (ADVRDOFFTIME - CLKACTIVATIONTIME) is a multiple of 3

  G = (1 + 0.5 × ADVEXTRADELAY) × GPMC_FCLK period, when (ADVRDOFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3

  G = (2 + 0.5 × ADVEXTRADELAY) × GPMC_FCLK period, when (ADVRDOFFTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For $\overline{ADV}$ rising edge ($\overline{ADV}$ de-activated) in writing mode:

- Case where GPMC_CONFIG1_i GPMCFCLKDIVIDER = 0

  G = 0.5 × ADVEXTRADELAY × GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 1h

  G = 0.5 × ADVEXTRADELAY × GPMC_FCLK period, when (CLKACTIVATIONTIME and ADVWROFFTIME are odd) or (CLKACTIVATIONTIME and ADVWROFFTIME are even)

  G = (1 + 0.5 × ADVEXTRADELAY) × GPMC_FCLK period, otherwise
- Case where GPMCFCLKDIVIDER = 2h

  G = 0.5 × ADVEXTRADELAY × GPMC_FCLK period, when (ADVWROFFTIME - CLKACTIVATIONTIME) is a multiple of 3

  G = (1 + 0.5 × ADVEXTRADELAY) × GPMC_FCLK period, when (ADVWROFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3

  G = (2 + 0.5 × ADVEXTRADELAY) × GPMC_FCLK period, when (ADVWROFFTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For $\overline{OE}$ falling edge ($\overline{OE}$ activated):

- Case where GPMC_CONFIG1_i GPMCFCLKDIVIDER = 0

  H = 0.5 × OEEXTRADELAY × GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 1h

  H = 0.5 × OEEXTRADELAY × GPMC_FCLK period, when (CLKACTIVATIONTIME and OEONTIME are odd) or (CLKACTIVATIONTIME and OEONTIME are even)

  H = (1 + 0.5 × OEEXTRADELAY) × GPMC_FCLK period, otherwise
- Case where GPMCFCLKDIVIDER = 2h

  H = 0.5 × OEEXTRADELAY × GPMC_FCLK period, when (OEONTIME - CLKACTIVATIONTIME) is a multiple of 3

  H = (1 + 0.5 × OEEXTRADELAY) × GPMC_FCLK period, when (OEONTIME - CLKACTIVATIONTIME - 1) is a multiple of 3

  H = (2 + 0.5 × OEEXTRADELAY) × GPMC_FCLK period, when (OEONTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For $\overline{OE}$ rising edge ($\overline{OE}$ de-activated):

- Case where GPMC_CONFIG1_i GPMCFCLKDIVIDER = 0

  H = 0.5 × OEEXTRADELAY × GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 1h

  H = 0.5 × OEEXTRADELAY × GPMC_FCLK period, when (CLKACTIVATIONTIME and OEOFFTIME are odd) or (CLKACTIVATIONTIME and OEOFFTIME are even)

  H = (1 + 0.5 × OEEXTRADELAY) × GPMC_FCLK period, otherwise
- Case where GPMCFCLKDIVIDER = 2h

  H = 0.5 × OEEXTRADELAY × GPMC_FCLK period, when (OEOFFTIME - CLKACTIVATIONTIME) is a multiple of 3

  H = (1 + 0.5 × OEEXTRADELAY) × GPMC_FCLK period, when (OEOFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3

  H = (2 + 0.5 × OEEXTRADELAY) × GPMC_FCLK period, when (OEOFFTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For $\overline{WE}$ falling edge ($\overline{WE}$ activated):

- Case where GPMC_CONFIG1_i GPMCFCLKDIVIDER = 0

  I = 0.5 × WEEXTRADELAY × GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 1h

  I = 0.5 × WEEXTRADELAY × GPMC_FCLK period, when (CLKACTIVATIONTIME and WEONTIME are odd) or (CLKACTIVATIONTIME and WEONTIME are even)

  I = (1 + 0.5 × WEEXTRADELAY) × GPMC_FCLK period, otherwise
- Case where GPMCFCLKDIVIDER = 2h

  I = 0.5 × WEEXTRADELAY × GPMC_FCLK period, when (WEONTIME - CLKACTIVATIONTIME) is a multiple of 3

  I = (1 + 0.5 × WEEXTRADELAY) × GPMC_FCLK period, when (WEONTIME - CLKACTIVATIONTIME - 1) is a multiple of 3

  I = (2 + 0.5 × WEEXTRADELAY) × GPMC_FCLK period, when (WEONTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For $\overline{WE}$ rising edge ($\overline{WE}$ de-activated):

- Case where GPMC_CONFIG1_i GPMCFCLKDIVIDER = 0

  I = 0.5 × WEEXTRADELAY × GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 1h

  I = 0.5 × WEEXTRADELAY × GPMC_FCLK period, when (CLKACTIVATIONTIME and WEOFFTIME are odd) or (CLKACTIVATIONTIME and WEOFFTIME are even)

  I = (1 + 0.5 × WEEXTRADELAY) × GPMC_FCLK period otherwise
- Case where GPMCFCLKDIVIDER = 2h

  I = 0.5 × WEEXTRADELAY × GPMC_FCLK period, when (WEOFFTIME - CLKACTIVATIONTIME) is a multiple of 3

  I = (1 + 0.5 × WEEXTRADELAY) × GPMC_FCLK period, when (WEOFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3

  I = (2 + 0.5 × WEEXTRADELAY) × GPMC_FCLK period, when (WEOFFTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For GPMC_$\overline{ADV}$ low pulse duration:

- Read operation

  K = (ADVRDOFFTIME - ADVONTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period
- Write operation

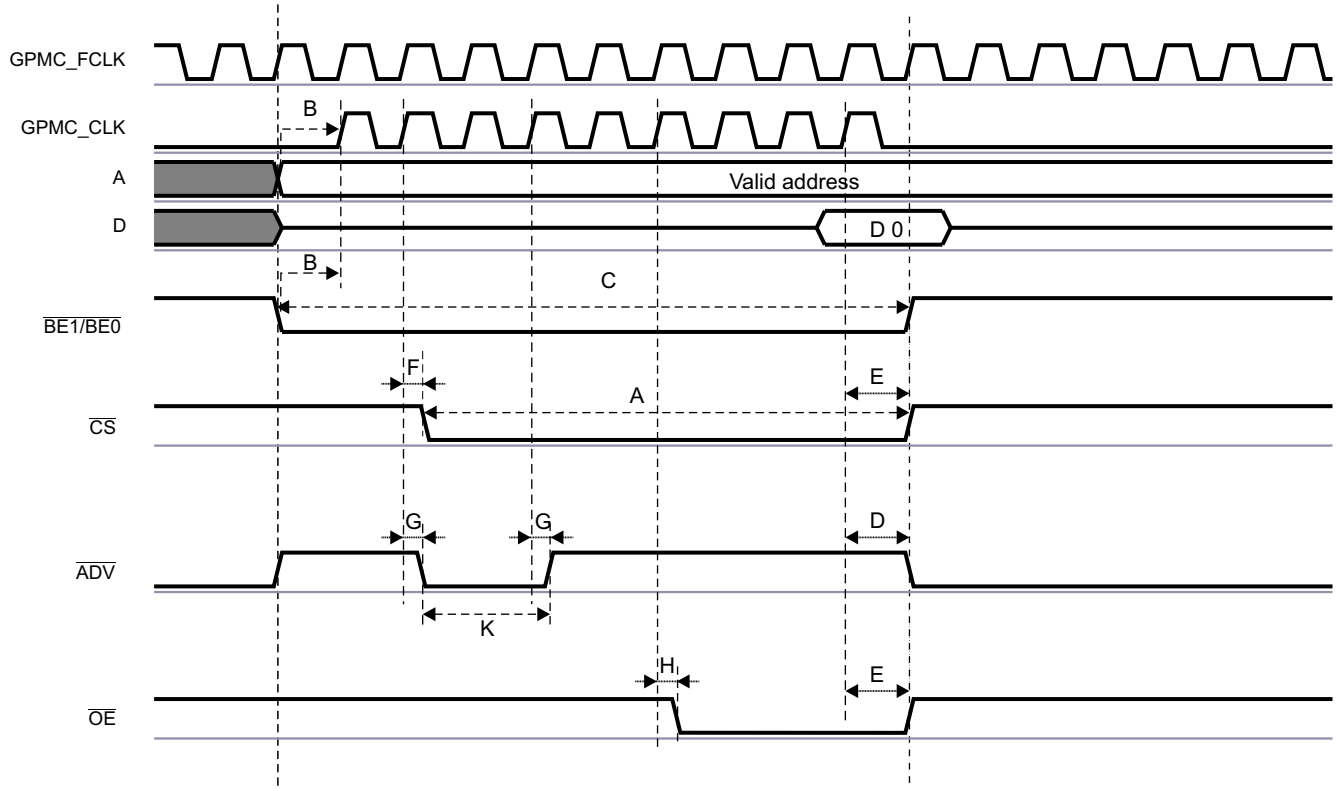
  K = (ADVWROFFTIME - ADVONTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period

For GPMC_WAIT invalid to first data latching GPMC_CLK edge:

L = WAITMONITORINGTIME × (GPMCFCLKDIVIDER + 1) × GPMC_FCLK period + GPMC_CLK period

Figure 9-45 shows a synchronous NOR single read simplified example where formulas are associated with signal waves.

Figure 9-45. Synchronous NOR Single Read Simplified Example



9.3.6.1.3 Asynchronous NOR Flash Timing Parameters Formulas

This section lists all formulas to use in order to calculate asynchronous NOR timing parameters. This is the case when [11-10] DEVICETYPE = 0x0 and when READTYPE or WRITETYPE are set to asynchronous mode.

Table 9-44. Asynchronous NOR Formulas Description Table

Configuration Parameter	Unit	Description
A	ns	Pulse duration - GPMC_CS low
B	ns	Delay time - GPMC_CS valid to GPMC_ADV_ALE invalid
C	ns	Delay time - GPMC_CS valid to GPMC_OE invalid (single read)
D	ns	Pulse duration - address bus valid - 2nd, 3rd and 4th accesses
E	ns	Delay time - GPMC_CS valid to GPMC_WE valid
F	ns	Delay time - GPMC_CS valid to GPMC_WE invalid
G	ns	Address invalid duration between 2 successive R/W accesses
H	ns	Setup time - read data valid before GPMC_OE high
I	ns	Delay time - GPMC_CS valid to GPMC_OE invalid (burst read)
J	ns	Delay time - address bus valid to GPMC_CS valid
		Delay time - data bus valid to GPMC_CS valid
		Delay time - GPMC_BE0_CLE/GPMC_BE1 valid to GPMC_CS valid
K	ns	Delay time - GPMC_CS valid to GPMC_ADV_ALE valid
L	ns	Delay time - GPMC_CS valid to GPMC_OE valid
M	ns	Delay time - GPMC_CS valid to first data latching edge
N	ns	Pulse duration - GPMC_BE0_CLE/GPMC_BE1 valid time
O	ns	Delay time - GPMC_CS valid to GPMC_ADV_ALE valid

The configuration parameters are calculated through the following formulas. Note that these formulas are not exhaustive.

GPMC_CS low pulse:

For single read: A = (CSRDOFFTIME - CSONTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period

For burst read: A = (CSRDOFFTIME - CSONTIME + (N - 1) × PAGEBURSTACCESSTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period, where N = page burst access number

For single write: A = (CSWROFFTIME - CSONTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period

For burst write: A = (CSWROFFTIME - CSONTIME + (N - 1) × PAGEBURSTACCESSTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period, where N = page burst access number

GPMC_CS valid to GPMC_ADV_ALE invalid delay:

For reading: B = ((ADVRDOFFTIME - CSONTIME) × (TIMEPARAGRANULARITY + 1) + 0.5 × (ADVEXTRADELAY - CSEXTRADELAY)) × GPMC_FCLK period

For writing: B = ((ADVWROFFTIME - CSONTIME) × (TIMEPARAGRANULARITY + 1) + 0.5 × (ADVEXTRADELAY - CSEXTRADELAY)) × GPMC_FCLK period

C = ((OEOFFTIME - CSONTIME) × (TIMEPARAGRANULARITY + 1) + 0.5 × (OEEXTRADELAY - CSEXTRADELAY)) × GPMC_FCLK period

D = PAGEBURSTACCESSTIME × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period

E = ((WEONTIME - CSONTIME) × (TIMEPARAGRANULARITY + 1) + 0.5 × (WEEXTRADELAY - CSEXTRADELAY)) × GPMC_FCLK period

F = ((WEOFFTIME - CSONTIME) × (TIMEPARAGRANULARITY + 1) + 0.5 × (WEEXTRADELAY - CSEXTRADELAY)) × GPMC_FCLK period

G = CYCLE2CYCLEDELAY × GPMC_FCLK period

H = ((OEOFFTIME - RDACCESSTIME) × (TIMEPARAGRANULARITY + 1) + 0.5 × OEEXTRADELAY) × GPMC_FCLK period

I = ((OEOFFTIME + (N - 1) × PAGEBURSTACCESSTIME - CSONTIME) × (TIMEPARAGRANULARITY + 1) + 0.5 × (OEEXTRADELAY - CSEXTRADELAY)) × GPMC_FCLK period, where N = page burst access number

J = (CSONTIME × (TIMEPARAGRANULARITY + 1) + 0.5 × CSEXTRADELAY) × GPMC_FCLK period

K = ((ADVONTIME - × (TIMEPARAGRANULARITY + 1) + 0.5 × (ADVEXTRADELAY - CSEXTRADELAY)) × GPMC_FCLK period

L = ((OEONTIME - CSONTIME) × (TIMEPARAGRANULARITY + 1) + 0.5 × (OEEXTRADELAY - CSEXTRADELAY)) × GPMC_FCLK period M = ((RDACCESSTIME - CSONTIME) × (TIMEPARAGRANULARITY + 1) - 0.5 × CSEXTRADELAY) × GPMC_FCLK period

GPMC_$\overline{BE0}$_CLE/$\overline{GPMC_BE1}$ pulse:

For single read: N = RDCYCLETIME × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period

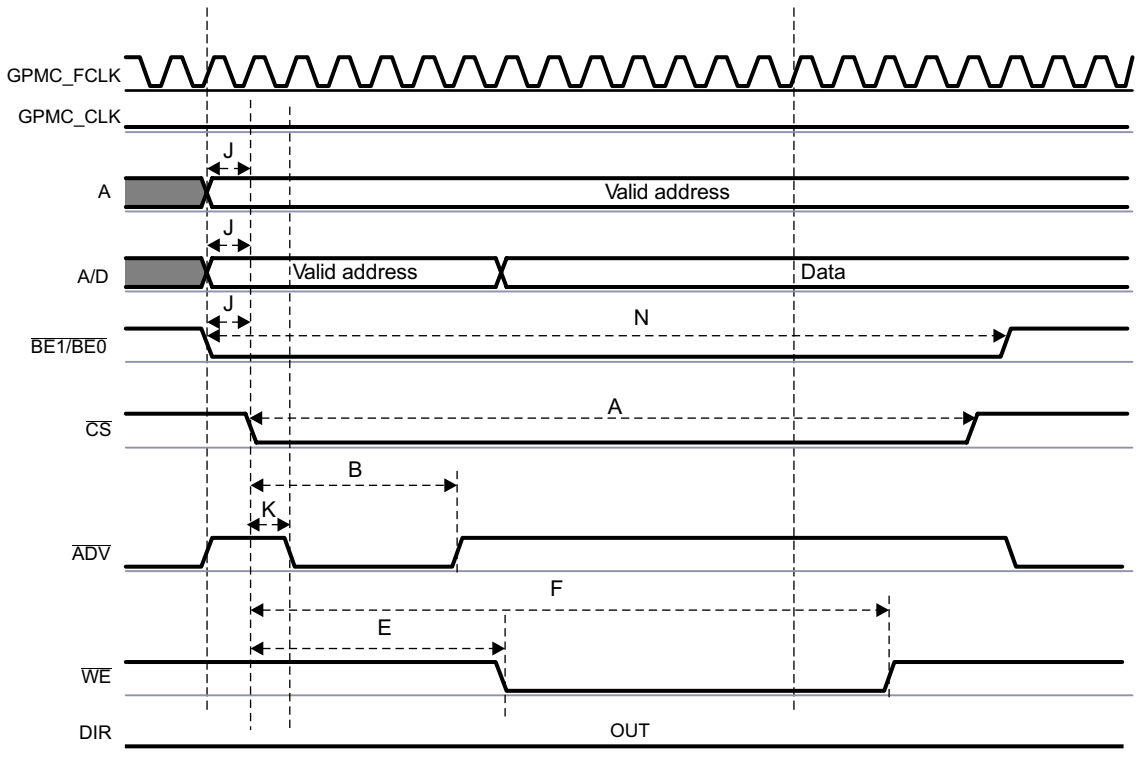
For burst read: N = (RDCYCLETIME + (N - 1) × PAGEBURSTACCESSTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period, where N = page burst access number

For burst write: N = (WRCYCLETIME + (N - 1) × PAGEBURSTACCESSTIME) × (TIMEPARAGRANULARITY + 1) × GPMC_FCLK period, where N = page burst access number

O = ((WRCYCLETIME + (N - 1) × PAGEBURSTACCESSTIME - CSONTIME) × (TIMEPARAGRANULARITY + 1) + 0.5 × (ADVEXTRADELAY - CSEXTRADELAY)) × GPMC_FCLK period

Figure 9-46 shows an asynchronous NOR single write simplified example where formulas are associated with signal waves.

Figure 9-46. Asynchronous NOR Single Write Simplified Example



Write multiple access is not supported in asynchronous mode. If WRITEMULTIPLE is enabled with WRITETYPE as asynchronous, the GPMC processes single asynchronous accesses.

9.4 Use Cases And Tips

9.4.1 How to Set GPMC Timing Parameters for Typical Accesses

9.4.1.1 External Memory Attached to the GPMC Module

As discussed in the introduction to this chapter, the GPMC module supports the following external memory types:

- Asynchronous or synchronous, 8-bit or 16-bit-width memory or device
- 16-bit address/data-multiplexed or not multiplexed NOR flash device
- 8- or 16-bit NAND flash device

The following examples show how to calculate GPMC timing parameters by showing a typical parameter setup for the access to be performed.

The example is based on a 512-Mb multiplexed NOR flash memory with the following characteristics:

- Type: NOR flash (address/data-multiplexed mode)
- Size: 512M bits
- Data Bus: 16 bits wide
- Speed: 125 MHz clock frequency
- Read access time: 80 ns

9.4.1.2 Typical GPMC Setup

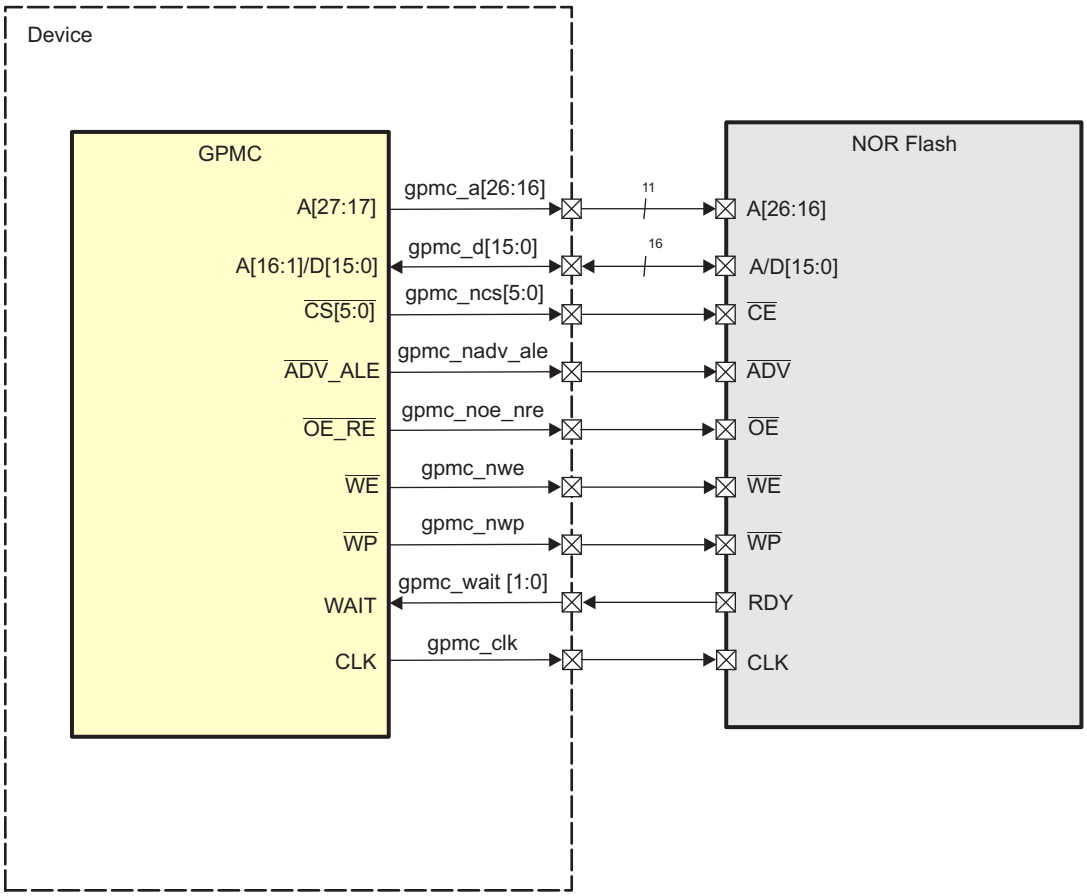
Table 9-45 lists some of the I/Os of the GPMC module.

Table 9-45. GPMC Signals

Signal Name	I/O	Description
GPMC_FCLK	Internal	Functional and interface clock. Acts as the time reference.
GPMC_CLK	O	External clock provided to the external device for synchronous operations
GPMC_A[27:17]	O	Address
GPMC_D[15: 0]	I/O	Data-multiplexed with addresses A[16:1] on memory side
$\overline{\text{GPMC\_CSx}}$	O	Chip-select (where x = 0, or 1)
GPMC_$\overline{\text{ADV}}$_ALE	O	Address valid enable
$\overline{\text{GPMC\_OE\_RE}}$	O	Output enable (read access only)
$\overline{\text{GPMC\_WE}}$	O	Write enable (write access only)
GPMC_WAIT	I	Ready signal from memory device. Indicates when valid burst data is ready to be read

Figure 9-47 shows the typical connection between the GPMC module and an attached NOR Flash memory.

Figure 9-47. GPMC Connection to an External NOR Flash Memory



The following sections demonstrate how to calculate GPMC parameters for three access types:

- Synchronous burst read
- Asynchronous read
- Asynchronous single write

9.4.1.3 GPMC Configuration for Synchronous Burst Read Access

The clock runs at 125 MHz (f = 125 MHz; T = 8 ns).

Table 9-46 shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

Table 9-47 shows how to calculate timings for the GPMC using the memory parameters.

Figure 9-48 shows the synchronous burst read access.

Table 9-46. Useful Timing Parameters on the Memory Side

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tCES	$\overline{CS}$ setup time to clock	0
tACS	Address setup time to clock	3
tIACC	Synchronous access time	80
tBACC	Burst access time valid clock to output delay	5,2
tCEZ	Chip-select to High-Impedance	7
tOEZ	Output enable to High-Impedance	7
tAVC	$\overline{ADV}$ setup time	6
tAVD	$\overline{AVD}$ pulse	6
tACH	Address hold time from clock	3

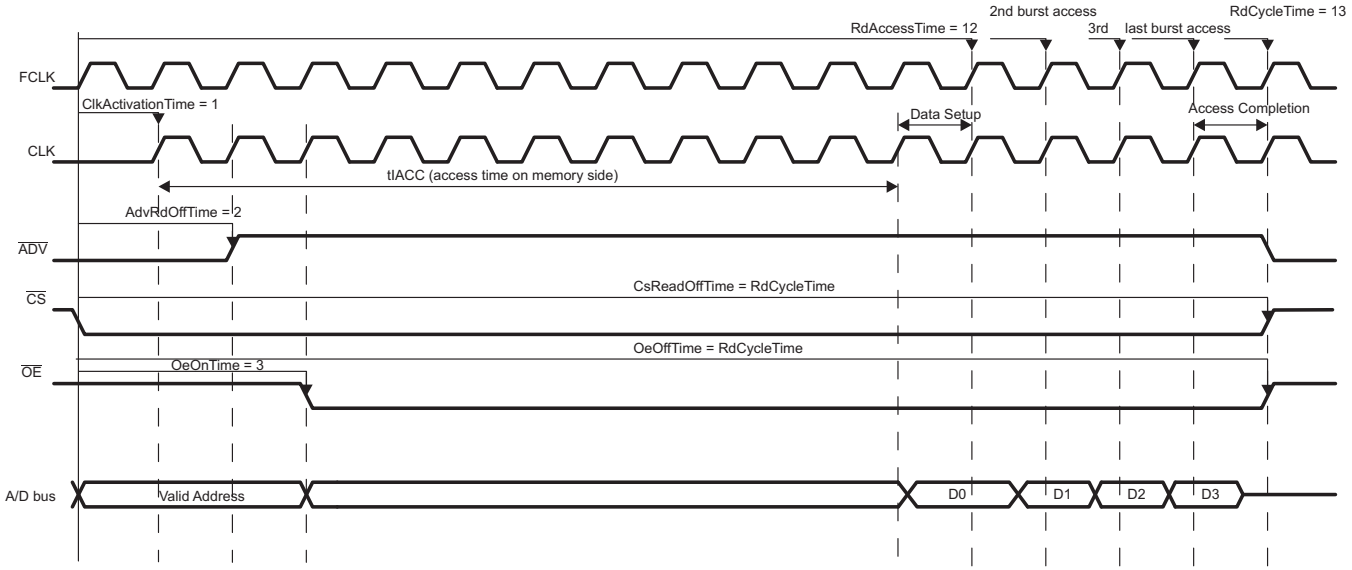
The following terms, which describe the timing interface between the controller and its attached device, are used to calculate the timing parameters on the GPMC side:

- Read Access time (GPMC side): Time required to activate the clock + read access time requested on the memory side + data setup time required for optimal capture of a burst of data
- Data setup time (GPMC side): Ensures a good capture of a burst of data (as opposed to taking a burst of data out). One word of data is processed in one clock cycle (T = 8 ns). The read access time between 2 bursts of data is tBACC = 5,2 ns. Therefore, data setup time is a clock period - tBACC = 2,8 ns of data setup.
- Access completion (GPMC side): (Different from page burst access time) Time required between the last burst access and access completion: $\overline{CS}$/$\overline{OE}$ hold time ($\overline{CS}$ and $\overline{OE}$ must be released at the end of an access. These signals are held to allow the access to complete).
- Read cycle time (GPMC side): Read Access time + access completion
- Write cycle time for burst access: Not supported for NOR flash memory

Table 9-47. Calculating GPMC Timing Parameters

Parameter Name on GPMC Side	Formula	Duration (ns)	Number of Clock Cycles (F = 125 MHz)	GPMC Register Configurations
GPMC FCLK Divider	-	-	-	GPMCFCLKDIVIDER = 0
ClkActivationTime	min (tCES, tACS)	3	1	CLKACTIVATIONTIME = 1
RdAccessTime	roundmax (ClkActivationTime + tIACC + DataSetupTime)	90,8: (8 + 80 + 2,8)	12 : roundmax (90,8 / 8)	RDACCESSTIME = Ch
PageBurstAccessTime	roundmax (tBACC)	roundmax (5,2)	1	PAGEBURSTACCESSTIME = 1
RdCycleTime	RdAccessTime + max (tCEZ, tOEZ)	97,8: (90,8 + 7)	13	RDCYCLETIME = Dh
CsOnTime	tCES	0	0	CSONTIME = 0
CsReadOffTime	RdCycleTime	-	13	CSRDOFFTIME = Dh
AdvOnTime	tAVC	0	0	ADVONTIME = 0
AdvRdOffTime	tAVD + tAVC	12	2	ADVRDOFFTIME = 2h
OeOnTime	(ClkActivationTime + tACH) < OeOnTime < (ClkActivationTime + tIACC)	-	3, for instance	OEONTIME = 3h
OeOffTime	RdCycleTime	-	13	OEOFFTIME = Dh

Figure 9-48. Synchronous Burst Read Access (Timing Parameters in Clock Cycles)



9.4.1.4 GPMC Configuration for Asynchronous Read Access

The clock runs at 125 MHz (f = 125 MHz; T = 8 ns).

Table 9-48 shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

Table 9-49 shows how to calculate timings for the GPMC using the memory parameters.

Figure 9-49 shows the asynchronous read access.

Table 9-48. AC Characteristics for Asynchronous Read Access

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tCE	Read Access time from $\overline{CS}$ low	80
tAAVDS	Address setup time to rising edge of $\overline{ADV}$	3
tAVDP	$\overline{ADV}$ low time	6
tCAS	$\overline{CS}$ setup time to $\overline{ADV}$	0
tOE	Output enable to output valid	6
tOEZ	Output enable to High-Impedance	7

Use the following formula to calculate the RdCycleTime parameter for this typical access:

RdCycleTime = RdAccessTime + AccessCompletion = RdAccessTime + 1 clock cycle + tOEZ

- First, on the memory side, the external memory makes the data available to the output bus. This is the memory-side read access time defined in Table 9-49: the number of clock cycles between the address capture ($\overline{ADV}$ rising edge) and the data valid on the output bus.

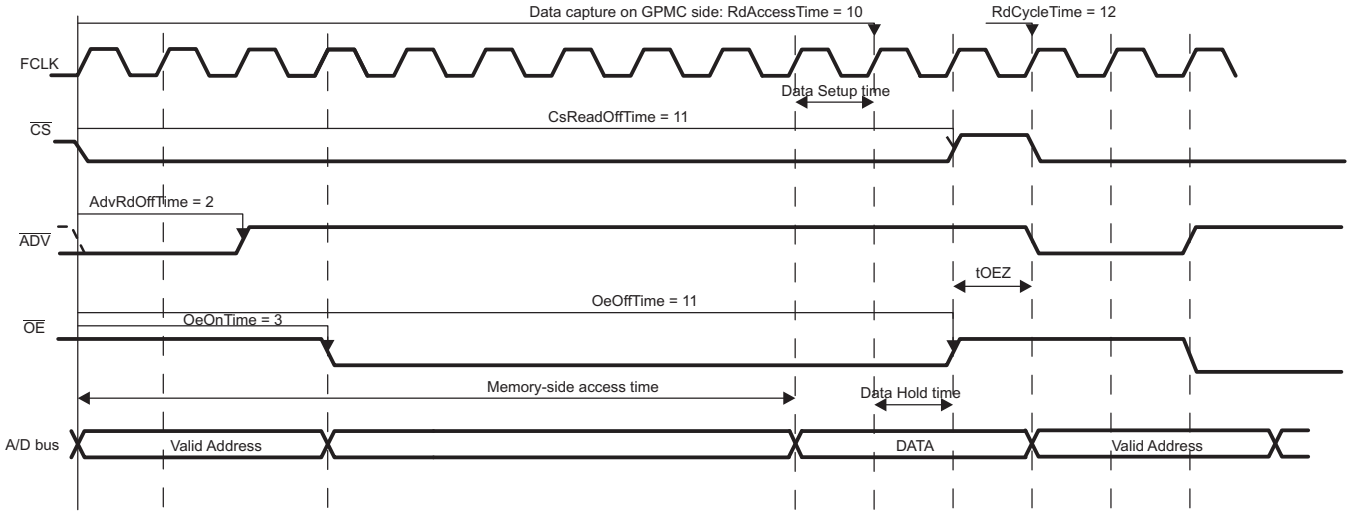
The GPMC requires some hold time to allow the data to be captured correctly and the access to be finished.

- To read the data correctly, the GPMC must be configure to meet the data setup time requirement of the memory; the GPMC module captures the data on the next rising edge. This is access time on the GPMC side.
- There must also be a data hold time for correctly reading the data (checking that there is no $\overline{OE}$/$\overline{CS}$ deassertion while reading the data). This data hold time is 1 clock cycle (that is, RdAccessTime + 1).
- To complete the access, $\overline{OE}$/$\overline{CS}$ signals are driven to high-impedance. RdAccessTime + 1 + tOEZ is the read cycle time.
- Addresses can now be relatched and a new read cycle begun.

Table 9-49. GPMC Timing Parameters for Asynchronous Read Access

Parameter Name on GPMC side	Formula	Duration (ns)	Number of Clock Cycles (F = 125 MHz)	GPMC Register Configurations
ClkActivationTime	n/a (asynchronous mode)			
RdAccessTime	round max (tCE)	80	10	RDACCESSTIME = Ah
PageBurstAccessTime	n/a (single access)			
RdCycleTime	RdAccessTime + 1cycle + tOEZ	95	12	RDCYCLETIME = Ch
CsOnTime	tCAS	0	0	CSONTIME = 0
CsReadOffTime	RdAccessTime + 1 cycle	88	11	CSRDOFFTIME = Bh
AdvOnTime	tAAVDS	3	1	ADVONTIME = 1
AdvRdOffTime	tAAVDS + tAVDP	9	2	ADVRDOFFTIME = 2
OeOnTime	OeOnTime ≥ AdvRdOffTime (multiplexed mode)	-	3, for instance	OEONTIME = 3h
OeOffTime	RdAccessTime + 1cycle	88	11	OEOFFTIME = Bh

Figure 9-49. Asynchronous Single Read Access (Timing Parameters in Clock Cycles)



9.4.1.5 GPMC Configuration for Asynchronous Single Write Access

The clock runs at 125 MHz: (f = 125 MHz; T = 8 ns).

Table 9-50 shows how to calculate timings for the GPMC using the memory parameters.

Table 9-51 shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

Figure 9-50 shows the synchronous burst write access.

Table 9-50. AC Characteristics for Asynchronous Single Write (Memory Side)

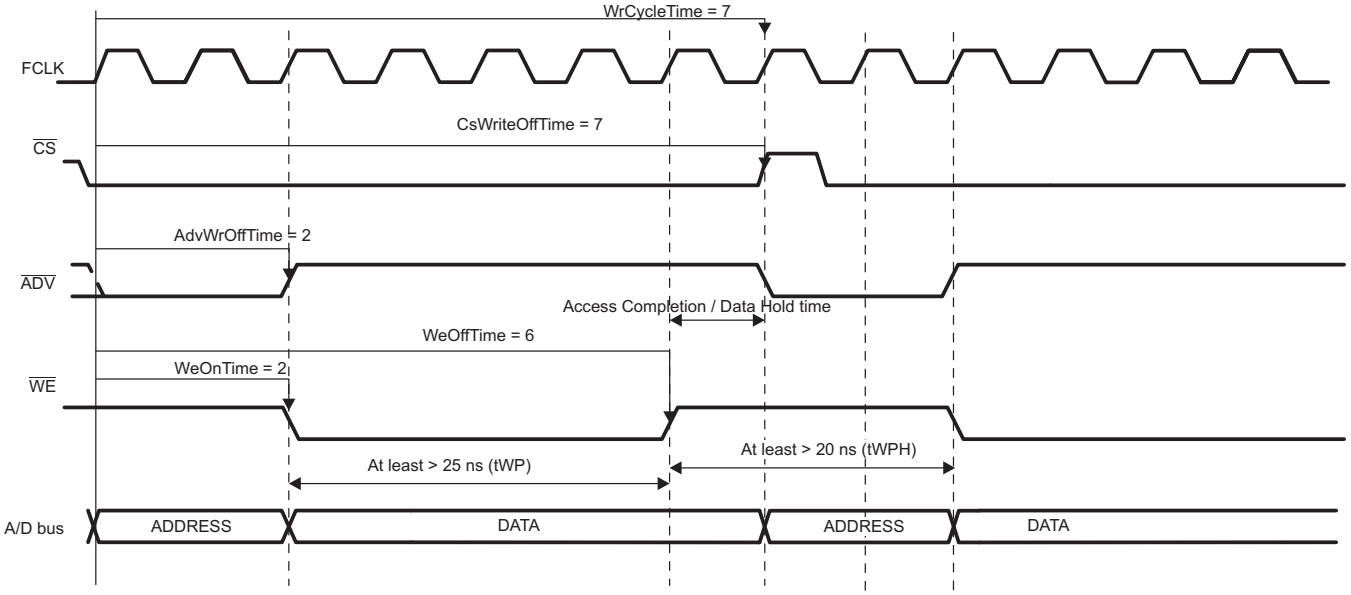
AC Characteristics on the Memory Side	Description	Duration (ns)
tWC	Write cycle time	60
tAVDP	$\overline{ADV}$ low time	6
tWP	Write pulse width	25
tWPH	Write pulse width high	20
tCS	$\overline{CS}$ setup time to $\overline{WE}$	3
tCAS	$\overline{CS}$ setup time to $\overline{ADV}$	0
tAVSC	$\overline{ADV}$ setup time	3

For asynchronous single write access, write cycle time is WrCycleTime = WeOffTime + AccessCompletion = WeOffTime + 1. For the AccesCompletion, the GPMC requires 1 cycle of data hold time ($\overline{CS}$ de-assertion).

Table 9-51. GPMC Timing Parameters for Asynchronous Single Write

Parameter Name on GPMC side	Formula	Duration (ns)	Number of Clock Cycles (F = 125 MHz)	GPMC Register Configurations
ClkActivationTime	n/a (asynchronous mode)			
WdAccessTime	Applicable only to WAITMONITORING (the value is the same as for read access)			
PageBurstAccessTime	n/a (single access)			
WrCycleTime	WeOffTime + AccessCompletion	56	7	WRCYCLETIME = 7h
CsOnTime	tCAS	0	0	CSONTIME = 0
CsWrOffTime	WeOffTime + 1	56	7	CSWROFFTIME = 7h
AdvOnTime	tAVSC	3	1	ADVONTIME = 1
AdvWrOffTime	tAVSC + tAVDP	9	2	ADVWROFFTIME = 2
WeOnTime	tCS	3	1	WEONTIME = 2
WeOffTime	tCS + tWP + tWPH	48	6	WEOFFTIME = 6h

Figure 9-50. Asynchronous Single Write Access (Timing Parameters in Clock Cycles)



9.4.2 How to Choose a Suitable Memory to Use With the GPMC

This section is intended to help the user select a suitable memory device to interface with the GPMC controller.

9.4.2.1 Supported Memories or Devices

NAND flash and NOR flash architectures are the two flash technologies. The GPMC supports various types of external memory or device, basically any one that supports NAND or NOR protocols:

- 8- and 16-bit width asynchronous or synchronous memory or device (8-bit: non burst device only)
- 16-bit address and data multiplexed NOR flash devices (pSRAM, , ...)
- 8- and 16-bit NAND flash device

9.4.2.2 Memory Pin Multiplexing

This section highlights the interfacing differences of the GPMC supported memories (see Table 9-52).

Table 9-52. Supported Memory Interfaces

Function	16-bit Address/Data Non-Muxed Flash	16-bit Address/Data Muxed pSRAM or NOR Flash	OneNAND	16-bit NAND	8-bit NAND
GPMC_A[27]	A28	A28			
GPMC_A[26]	A27	A27			
GPMC_A[25]	A26	A26			
GPMC_A[24]	A25	A25			
GPMC_A[23]	A24	A24			
GPMC_A[22]	A23	A23			
GPMC_A[21]	A22	A22			
GPMC_A[20]	A21	A21			
GPMC_A[19]	A20	A20			
GPMC_A[18]	A10	A10			
GPMC_A[17]	A18	A18			
GPMC_A[16]	A17	A17			
GPMC_A[15]	A16				
GPMC_A[14]	A15				
GPMC_A[13]	A14				
GPMC_A[12]	A13				
GPMC_A[11]	A12				
GPMC_A[10]	A11				
GPMC_A[9]	A10				
GPMC_A[8]	A9				
GPMC_A[7]	A8				
GPMC_A[6]	A7				
GPMC_A[5]	A6				
GPMC_A[4]	A5				
GPMC_A[3]	A4				
GPMC_A[2]	A3				
GPMC_A[1]	A2				
GPMC_A[0]	A1				
GPMC_D[15]	D15 or A16	D15 or A16	D15 or A16	IO15	
GPMC_D[14]	D14 or A15	D14 or A15	D14 or A15	IO14	
GPMC_D[13]	D13 or A14	D13 or A14	D13 or A14	IO13	

Table 9-52. Supported Memory Interfaces (continued)

Function	16-bit Address/Data Non-Muxed Flash	16-bit Address/Data Muxed pSRAM or NOR Flash	OneNAND	16-bit NAND	8-bit NAND
GPMC_D[12]	D12 or A13	D12 or A13	D12 or A13	IO12	
GPMC_D[11]	D11 or A12	D11 or A12	D11 or A12	IO11	
GPMC_D[10]	D10 or A11	D10 or A11	D10 or A11	IO10	
GPMC_D[9]	D9 or A10	D9 or A10	D9 or A10	IO9	
GPMC_D[8]	D8 or A9	D8 or A9	D8 or A9	IO8	
GPMC_D[7]	D7 or A8	D7 or A8	D7 or A8	IO7	IO7
GPMC_D[6]	D6 or A7	D6 or A7	D6 or A7	IO6	IO6
GPMC_D[5]	D5 or A6	D5 or A6	D5 or A6	IO5	IO5
GPMC_D[4]	D4 or A5	D4 or A5	D4 or A5	IO4	IO4
GPMC_D[3]	D3 or A4	D3 or A4	D3 or A4	IO3	IO3
GPMC_D[2]	D2 or A3	D2 or A3	D2 or A3	IO2	IO2
GPMC_D[1]	D1 or A2	D1 or A2	D1 or A2	IO1	IO1
GPMC_D[0]	D0 or A1	D0 or A1	D0 or A1	IO0	IO0
GPMC_CLK	CLK	CLK	CLK		
GPMC_CS[0]	CS0 (Chip Select)	CS0 (Chip Select)	CS0 (Chip Select)	CE0 (Chip Enable)	CE0 (Chip Enable)
GPMC_CS[1]	CS1	CS1	CS1	CE1	CE1
GPMC_CS[2]	CS2	CS2	CS2	CE2	CE2
GPMC_CS[3]	CS3	CS3	CS3	CE3	CE3
GPMC_CS[4]	CS4	CS4	CS4	CE4	CE4
GPMC_CS[5]	CS5	CS5	CS5	CE5	CE5
GPMC_ADV_ALE	ADV (Address Value)	ADV (Address Value)	ADV (Address Value)	ALE (address latch enable)	ALE (address latch enable)
GPMC_OE_RE	OE (Output Enable)	OE (Output Enable)	OE (Output Enable)	RE (read enable)	RE (read enable)
GPMC_WE	WE (Write Enable)	WE (Write Enable)	WE (Write Enable)	WE (write enable)	WE (write enable)
GPMC_BE0_CLE	BE0 (Byte Enable)	BE0 (Byte Enable)	BE0 (Byte Enable)	CLE (command latch enable)	CLE (command latch enable)
GPMC_BE1	BE1	BE1	BE1		
GPMC_WP	WP (Write Protect)	WP (Write Protect)	WP (Write Protect)	WP (write protect)	WP (write protect)
GPMC_WAIT	WAIT0	WAIT0	WAIT0	R/B0 (ready/busy)	R/B0 (ready/busy)

9.4.2.3 NAND Interface Protocol

NAND flash architecture, introduced in 1989, is a flash technology. NAND is a page-oriented memory device, that is, read and write accesses are done by pages. NAND achieves great density by sharing common areas of the storage transistor, which creates strings of serially connected transistors (in NOR devices, each transistor stands alone). Thanks to its high density NAND is best suited to devices requiring high capacity data storage, such as pictures, music, or data files. NAND non-volatility, makes of it a good storage solution for many applications where mobility, low power, and speed are key factors. Low pin count and simple interface are other advantages of NAND.

Table 9-53 summarizes the NAND interface signals level applied to external device or memories.

Table 9-53. NAND Interface Bus Operations Summary

Bus Operation	CLE	ALE	$\overline{CE}$	$\overline{WE}$	$\overline{RE}$	$\overline{WP}$
Read (cmd input)	H	L	L	RE	H	x
Read (add input)	L	H	L	RE	H	x
Write (cmd input)	H	L	L	RE	H	H
Write (add input)	L	H	L	RE	H	H
Data input	L	L	L	RE	H	H
Data output	L	L	L	H	FE	x
Busy (during read)	x	x	H	H	H	x
Busy (during program)	x	x	x	x	x	H
Busy (during erase)	x	x	x	x	x	H
Write protect	x	x	x	x	x	L
Stand-by	x	x	H	x	x	H/L

9.4.2.4 NOR Interface Protocol

NOR flash architecture, introduced in 1988, is a flash technology. Unlike NAND, which is a sequential access device, NOR is directly addressable; it is designed to be a random access device. NOR is best suited to devices used to store and run code or firmware, usually in small capacities. While NOR has fast read capabilities it has slow write and erase functions compared to NAND architecture.

Table 9-54 summarizes the NOR interface signals level applied to external device or memories.

Table 9-54. NOR Interface Bus Operations Summary

Bus Operation	CLK	$\overline{ADV}$	$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	WAIT	DQ[15:0]
Read (asynchronous)	x	L	L	L	H	Asserted	Output
Read (synchronous)	Running	L	L	L	H	Driven	Output
Read (burst suspend)	Halted	x	L	H	H	Active	Output
Write	x	L	L	H	L	Asserted	Input
Output disable	x	x	L	H	H	Asserted	High-Z
Standby	x	x	H	x	x	High-Z	High-Z

9.4.2.5 Other Technologies

Other supported device type interact with the GPMC through the NOR interface protocol.

OneNAND Flash is a high-density and low-power memory device. It is based on single- or multi-level-cell NAND core with SRAM and logic, and interfaces as a synchronous NOR Flash, plus has synchronous write capability. It reads faster than conventional NAND and writes faster than conventional NOR flash. Hence, it is appropriate for both mass storage and code storage.

pSRAM stands for pseudo-static random access memory. pSRAM is a low-power memory device for mobile applications. pSRAM is based on the DRAM cell with internal refresh and address control features, and interfaces as a synchronous NOR Flash, plus has synchronous write capability.

9.4.2.6 Supported Protocols

The GPMC supports the following interface protocols when communicating with external memory or external devices:

- Asynchronous read/write access
- Asynchronous read page access (4-8-16 Word16)
- Synchronous read/write access
- Synchronous read burst access without wrap capability (4-8-16 Word16)
- Synchronous read burst access with wrap capability (4-8-16 Word16)

9.4.2.7 GPMC Features and Settings

This section lists GPMC features and settings:

- Supported device type: up to four NAND or NOR protocol external memories or devices
- Operating Voltage: 3.3V
- Maximum GPMC addressing capability: 256 MBytes divided into eight chip-selects
- Maximum supported memory size: 256 MBytes (must be a power-of-2)
- Minimum supported memory size: 16 MBytes (must be a power-of-2). Aliasing occurs when addressing smaller memories.
- Data path to external memory or device: 8- and 16-bit wide
- Burst and page access: burst of 4-8-16 Word16
- Supports bus keeping
- Supports bus turn around

9.5 GPMC Registers

Table 9-55 provides a summary of the GPMC registers. For the base address of these registers, see Table 1-11. All GPMC registers are aligned to 32-bit address boundaries. All register file accesses, except the GPMC_NAND_DATA_i register, are little-endian. If the GPMC_NAND_DATA_i register is accessed, the endianness is access-dependent.

In this section, i corresponds to the chip-select number, i = 0 to 5.

Table 9-55. GPMC Registers

Address Offset	Register Name	Section
0h	GPMC_REVISION	Section 9.5.1
10h	GPMC_SYSCONFIG	Section 9.5.2
14h	GPMC_SYSSTATUS	Section 9.5.3
18h	GPMC_IRQSTATUS	Section 9.5.4
1Ch	GPMC_IRQENABLE	Section 9.5.5
40h	GPMC_TIMEOUT_CONTROL	Section 9.5.6
44h	GPMC_ERR_ADDRESS	Section 9.5.7
48h	GPMC_ERR_TYPE	Section 9.5.8
50h	GPMC_CONFIG	Section 9.5.9
54h	GPMC_STATUS	Section 9.5.10
60h + (30h × i)	GPMC_CONFIG1_i[1]	Section 9.5.11
64h + (30h × i)	GPMC_CONFIG2_i[1]	Section 9.5.12
68h + (30h × i)	GPMC_CONFIG3_i[1]	Section 9.5.13
6Ch + (30h × i)	GPMC_CONFIG4_i[1]	Section 9.5.14
70h + (30h × i)	GPMC_CONFIG5_i[1]	Section 9.5.15
74h + (30h × i)	GPMC_CONFIG6_i[1]	Section 9.5.16
78h + (30h × i)	GPMC_CONFIG7_i[1]	Section 9.5.17
7Ch + (30h × i)	GPMC_NAND_COMMAND_i[1]	Section 9.5.18
80h + (30h × i)	GPMC_NAND_ADDRESS_i[1]	Section 9.5.19
84h + (30h × i)	GPMC_NAND_DATA_i[1]	Section 9.5.20
1E0h	GPMC_PREFETCH_CONFIG1	Section 9.5.21
1E4h	GPMC_PREFETCH_CONFIG2	Section 9.5.22
1ECh	GPMC_PREFETCH_CONTROL	Section 9.5.23
1F0h	GPMC_PREFETCH_STATUS	Section 9.5.24
1F4h	GPMC_ECC_CONFIG	Section 9.5.25
1F8h	GPMC_ECC_CONTROL	Section 9.5.26
1FCh	GPMC_ECC_SIZE_CONFIG	Section 9.5.27
200h + (4h × j)	GPMC_ECCj_RESULT[2]	Section 9.5.28
240h + (10h × i)	GPMC_BCH_RESULT0_i[1]	Section 9.5.29
244h + (10h × i)	GPMC_BCH_RESULT1_i[1]	Section 9.5.30
248h + (10h × i)	GPMC_BCH_RESULT2_i[1]	Section 9.5.31
24Ch + (10h × i)	GPMC_BCH_RESULT3_i[1]	Section 9.5.32
300h + (10h × i)	GPMC_BCH_RESULT4_i[1]	Section 9.5.34
304h + (10h × i)	GPMC_BCH_RESULT5_i[1]	Section 9.5.35
308h + (10h × i)	GPMC_BCH_RESULT6_i[1]	Section 9.5.36
2D0h	GPMC_BCH_SWDATA	Section 9.5.33

(1) i = 0 to 5 for GPMC
(2) j = 0 to 8 for GPMC

9.5.1 GPMC_REVISION

This register contains the IP revision code.

Figure 9-51. GPMC_REVISION

31	16
Reserved	
R-0	

15	8	7	0
Reserved		REV	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-56. GPMC_REVISION Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	REV	0-FFh	IP revision. Major revision is [7-4]. Minor revision is [3-0]. Examples: 10h for revision 1.0, 21h for revision 2.1.

9.5.2 GPMC_SYSCONFIG

This register controls the various parameters of the OCP interface.

Figure 9-52. GPMC_SYSCONFIG

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		SIDLEMODE		Rsvd	SOFTRESET	AUTOIDLE
R-0		R/W		R-0	R/W	R/W

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-57. GPMC_SYSCONFIG Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-3	SIDLEMODE		Idle mode
		0	Force-idle. An idle request is acknowledged unconditionally
		1h	No-idle. An idle request is never acknowledged
		2h	Smart-idle. Acknowledgement to an idle request is given based on the internal activity of the module
		3h	Reserved
2	Reserved	0	Reserved
1	SOFTRESET		Software reset (Set 1 to this bit triggers a module reset. This bit is automatically reset by hardware. During reads, it always returns 0)
		0	Normal mode
		1	The module is reset
0	AUTOIDLE		Internal OCP clock gating strategy
		0	Interface clock is free-running
		1	Automatic Interface clock gating strategy is applied, based on the Interconnect activity

9.5.3 GPMC_SYSSTATUS

This register provides status information about the module, excluding the interrupt status information.

Figure 9-53. GPMC_SYSSTATUS

31	16
Reserved	
R-0	

15	1	0
Reserved		RESETDONE
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-58. GPMC_SYSSTATUS Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	RESETDONE		Internal reset monitoring
		R0	Internal module reset in on-going
		R1	Reset completed

9.5.4 GPMC_IRQSTATUS

This interrupt status register regroups all the status of the module internal events that can generate an interrupt.

Figure 9-54. GPMC_IRQSTATUS

31	16
Reserved	
R-0	

15	10	9	8
Reserved		WAIT1EDGE DETECTIONSTATUS	WAIT0EDGE DETECTIONSTATUS
R-0		R/W-0	R/W-0

7	2	1	0
Reserved		TERMINAL COUNTSTATUS	FIFOEVENT STATUS
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-59. GPMC_IRQSTATUS Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	WAIT1EDGEDETECTIONSTATUS		Status of the Wait1 Edge Detection interrupt
		R0	A transition on WAIT1 input pin has not been detected
		W0	WAIT1EDGEDETECTIONSTATUS bit unchanged
		R1	A transition on WAIT1 input pin has been detected
		W1	WAIT1EDGEDETECTIONSTATUS bit is reset
8	WAIT0EDGEDETECTIONSTATUS		Status of the Wait0 Edge Detection interrupt
		R0	A transition on WAIT0 input pin has not been detected
		W0	WAIT0EDGEDETECTIONSTATUS bit unchanged
		R1	A transition on WAIT0 input pin has been detected
		W1	WAIT0EDGEDETECTIONSTATUS bit is reset
7-2	Reserved	0	Reserved
1	TERMINALCOUNTSTATUS		Status of the TerminalCountEvent interrupt
		R0	Indicates that CountValue is greater than 0
		W0	TERMINALCOUNTSTATUS bit unchanged
		R1	Indicates that CountValue is equal to 0
		W1	TERMINALCOUNTSTATUS bit is reset
0	FIFOEVENTSTATUS		Status of the FIFOEvent interrupt
		R0	Indicates than less than GPMC_PREFETCH_STATUS[16] FIFOTHRESHOLDSTATUS bytes are available in prefetch mode and less than FIFOTHRESHOLD bytes free places are available in write-posting mode.
		W0	FIFOEVENTSTATUS bit unchanged
		R1	Indicates than at least GPMC_PREFETCH_STATUS[16] FIFOTHRESHOLDSTATUS bytes are available in prefetch mode and at least FIFOTHRESHOLD bytes free places are available in write-posting mode.
		W1	FIFOEVENTSTATUS bit is reset

9.5.5 GPMC_IRQENABLE

The interrupt enable register allows to mask/unmask the module internal sources of interrupt, on a event-by-event basis.

Figure 9-55. GPMC_IRQENABLE

31	16
Reserved	
R-0	

15	10	9	8
Reserved		WAIT1EDGE DETECTIONENABLE	WAIT0EDGE DETECTIONENABLE
R-0		R/W-0	R/W-0

7	2	1	0
Reserved		TERMINAL COUNTENABLE	FIFOEVENT ENABLE
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-60. GPMC_IRQENABLE Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	WAIT1EDGEDETECTIONENABLE		Enables the Wait1 Edge Detection interrupt
		0	Wait1EdgeDetection interrupt is masked
		1	Wait1EdgeDetection event generates an interrupt if occurs
8	WAIT0EDGEDETECTIONENABLE		Enables the Wait0 Edge Detection interrupt
		0	Wait0EdgeDetection interrupt is masked
		1	Wait0EdgeDetection event generates an interrupt if occurs
7-2	Reserved	0	Reserved
1	TERMINALCOUNTEVENTENABLE		Enables TerminalCountEvent interrupt issuing in pre-fetch or write posting mode
		0	TerminalCountEvent interrupt is masked
		1	TerminalCountEvent interrupt is not masked
0	FIFOEVENTENABLE		Enables the FIFOEvent interrupt
		0	FIFOEvent interrupt is masked
		1	FIFOEvent interrupt is not masked

9.5.6 GPMC_TIMEOUT_CONTROL

The GPMC_TIMEOUT_CONTROL register allows the user to set the start value of the timeout counter

Figure 9-56. GPMC_TIMEOUT_CONTROL

31	16
Reserved	
R-0	

15	13	12	4	3	1	0
Reserved		TIMEOUTSTARTVALUE		Reserved		TIMEOUTENABLE
R-0		R/W-0		R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-61. GPMC_TIMEOUT_CONTROL Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved
12-4	TIMEOUTSTARTVALUE	0-1FFh	Start value of the time-out counter (000 corresponds to 0 GPMC.FCLK cycle, 1h corresponds to 1 GPMC.FCLK cycle, and 1FFh corresponds to 511 GPMC.FCLK cycles)
3-1	Reserved	0	Reserved
0	TIMEOUTENABLE		Enable bit of the TimeOut feature
		0	TimeOut feature is disabled
		1	TimeOut feature is enabled

9.5.7 GPMC_ERR_ADDRESS

The GPMC_ERR_ADDRESS register stores the address of the illegal access when an error occurs.

Figure 9-57. GPMC_ERR_ADDRESS

31	30	0
Rsvd	ILLEGALADD	
R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-62. GPMC_ERR_ADDRESS Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-0	ILLEGALADD	0-7FFF FFFFh	Address of illegal access: A30 (0 for memory region, 1 for GPMC register region) and A29-A0 (1GByte maximum)

9.5.8 GPMC_ERR_TYPE

The GPMC_ERR_TYPE register stores the type of error when an error occurs.

Figure 9-58. GPMC_ERR_TYPE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved					ILLEGALMCMD		
R-0					R/W-0		

7	6	5	4	3	2	1	0
Reserved			ERRORNOT SUPPADD	ERRORNOT SUPPMCMD	ERROR TIMEOUT	Reserved	ERRORVALID
R-0			R/W-0	R/W-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-63. GPMC_ERR_TYPE Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10-8	ILLEGALMCMD	0-7h	System Command of the transaction that caused the error
7-5	Reserved	0	Reserved
4	ERRORNOTSUPPADD		Not supported Address error
		0	No error occurs
		1	The error is due to a non supported Address
3	ERRORNOTSUPPMCMD		Not supported Command error
		0	No error occurs
		1	The error is due to a non supported Command
2	ERRORTIMEOUT		Time-out error
		0	No error occurs
		1	The error is due to a time out
1	Reserved	0	Reserved
0	ERRORVALID		Error validity status - Must be explicitly cleared with a write 1 transaction
		0	All error fields no longer valid
		1	Error detected and logged in the other error fields

9.5.9 GPMC_CONFIG

The configuration register allows global configuration of the GPMC.

Figure 9-59. GPMC_CONFIG

31	16
Reserved	
R-0	

15	10	9	8
Reserved		WAIT1PIN POLARITY	WAIT0PIN POLARITY
R-0		R/W-0	R/W-0

7	5	4	3	2	1	0
Reserved		WRITE PROTECT	Reserved		LIMITED ADDRESS	NANDFORCE POSTEDWRITE
R-0		R/W-0	R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-64. GPMC_CONFIG Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	WAIT1PINPOLARITY		Selects the polarity of input pin WAIT1
		0	WAIT1 active low
		1	WAIT1 active high
8	WAIT0PINPOLARITY		Selects the polarity of input pin WAIT0
		0	WAIT0 active low
		1	WAIT0 active high
7-5	Reserved	0	Reserved
4	WRITEPROTECT		Controls the $\overline{WP}$ output pin level
		0	$\overline{WP}$ output pin is low
		1	$\overline{WP}$ output pin is high
3-2	Reserved	0	Reserved
1	LIMITEDADDRESS	0-1	Limited Address device support
0	NANDFORCEPOSTEDWRITE		Enables the Force Posted Write feature to NAND Cmd/Add/Data location
		0	Disables Force Posted Write
		1	Enables Force Posted Write

9.5.10 GPMC_STATUS

The status register provides global status bits of the GPMC.

Figure 9-60. GPMC_STATUS

31	16
Reserved	
R-0	

15	10	9	8
Reserved		WAIT1STATUS	WAIT0STATUS
R-0		R/W-0	R/W-0

7	1	0
Reserved		EMPTYWRITEBUFFERSTATUS
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-65. GPMC_STATUS Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	WAIT1STATUS		Is a copy of input pin WAIT1. (Reset value is WAIT1 input pin sampled at IC reset)
		0	WAIT1 asserted (inactive state)
		1	WAIT1 de-asserted
8	WAIT0STATUS		Is a copy of input pin WAIT0. (Reset value is WAIT0 input pin sampled at IC reset)
		0	WAIT0 asserted (inactive state)
		1	WAIT0 de-asserted
7-1	Reserved	0	Reserved
0	EMPTYWRITEBUFFERSTATUS		Stores the empty status of the write buffer
		0	Write Buffer is not empty
		1	Write Buffer is empty

9.5.11 GPMC_CONFIG1_i

The configuration 1 register sets signal control parameters per chip select.

Figure 9-61. GPMC_CONFIG1_i

31	30	29	28	27	26	25	24
WRAPBURST	READMULTIPLE	READTYPE	WRITEMULTIPLE	WRITETYPE	CLKACTIVATIONTIME		ATTACHEDDEVICE PAGELENGTH
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		R/W-0

23	22	21	20	19	18	17	16
ATTACHEDDEVICE PAGELENGTH	WAITREAD MONITORING	WAITWRITE MONITORING	Reserved	WAITMONITORINGTIME		WAITPINSELECT	
R/W-0	R/W-0	R/W-0	R-0	R/W-0		R/W-0	

15	14	13	12	11	10	9	8
Reserved		DEVICESIZE		DEVICETYPE		MUXADDDATA	
R-0		R/W-0		R/W-0		R/W-0	

7	5	4	3	2	1	0
Reserved		TIMEPARA GRANULARITY	Reserved		GPMCFCLKDIVIDER	
R-0		R/W-0	R-0		R/W-0	

LEGEND: R = Read only; W1C = Write 1 to clear bit; *-n* = value after reset

Table 9-66. GPMC_CONFIG1_i Field Descriptions

Bit	Field	Value	Description
31	WRAPBURST		Enables the wrapping burst capability. Must be set if the attached device is configured in wrapping burst
		0	Synchronous wrapping burst not supported
		1	Synchronous wrapping burst supported
30	READMULTIPLE		Selects the read single or multiple access
		0	single access
		1	multiple access (burst if synchonous, page if asynchronous)
29	READTYPE		Selects the read mode operation
		0	Read Asynchronous
		1	Read Synchronous
28	WRITEMULTIPLE		Selects the write single or multiple access
		0	Single access
		1	Multiple access (burst if synchronous, considered as single if asynchronous)
27	WRITETYPE		Selects the write mode operation
		0	Write Asynchronous
		1	Write Synchronous
26-25	CLKACTIVATIONTIME		Output GPMC.CLK activation time
		0	First rising edge of GPMC_CLK at start access time
		1h	First rising edge of GPMC_CLK one GPMC_FCLK cycle after start access time
		2h	First rising edge of GPMC_CLK two GPMC_FCLK cycles after start access time
		3h	Reserved
24-23	ATTACHEDDEVICEPAGELENGTH		Specifies the attached device page (burst) length (1 Word = Interface size)
		0	4 Words
		1h	8 Words
		2h	16 Words
		3h	Reserved

Table 9-66. GPMC_CONFIG1_i Field Descriptions (continued)

Bit	Field	Value	Description
22	WAITREADMONITORING		Selects the Wait monitoring configuration for Read accesses (Reset value is BOOTWAITEN input pin sampled at IC reset)
		0	WAIT pin is not monitored for read accesses
		1	WAIT pin is monitored for read accesses
21	WAITWRITEMONITORING		Selects the Wait monitoring configuration for Write accesses
		0	WAIT pin is not monitored for write accesses
		1	WAIT pin is monitored for write accesses
20	Reserved	0	Reserved
19-18	WAITMONITORINGTIME		Selects input pin Wait monitoring time
		0	WAIT pin is monitored with valid data
		1h	WAIT pin is monitored one GPMC_CLK cycle before valid data
		2h	WAIt pin is monitored two GPMC_CLK cycle before valid data
		3h	Reserved
17-16	WAITPINSELECT		Selects the input WAIT pin for this chip select (Reset value is BOOTWAITSELECT input pin sampled at IC reset for CS0 and 0 for CS1-5)
		0	WAIT input pin is WAIT0
		1h	WAIT input pin is WAIT1
		2h	Reserved
		3h	Reserved
15-14	Reserved	0	Reserved
13-12	DEVICESIZE		Selects the device size attached (Reset value is BOOTDEVICESIZE input pin sampled at IC reset for $\overline{CS}$[0] and 01 for $\overline{CS}$[1-5])
		0	8 bit
		1h	16 bit
		2h	Reserved
		3h	Reserved
11-10	DEVICETYPE		Selects the attached device type
		0	NOR Flash like, asynchronous and synchronous devices
		1h	Reserved
		2h	NAND Flash like devices, stream mode
		3h	Reserved
9-8	MUXADDDATA		Enables the Address and data multiplexed protocol (Reset value is CS0MUXDEVICE input pin sampled at IC reset for $\overline{CS}$[0] and 0 for $\overline{CS}$[1-5])
		0	Non-multiplexed attached device
		1h	AAD-multiplexed protocol device
		2h	Address and data multiplexed attached device
		3h	Reserved
7-5	Reserved	0	Reserved
4	TIMEPARAGRANULARITY		Signals timing latencies scalar factor (Rd/WRCycleTime, RdAccessTime, PageBurstAccessTime, CSOnTime, CSRd/WrOffTime, ADVOnTime, ADVRd/WrOffTime, OEOnTime, OEOffTime, WEOnTime, WEOffTime, Cycle2CycleDelay, BusTurnAround, TimeOutStartValue)
		0	×1 latencies
		1	×2 latencies
3-2	Reserved	0	Reserved

Table 9-66. GPMC_CONFIG1_i Field Descriptions (continued)

Bit	Field	Value	Description
1-0	GPMCFCLKDIVIDER		Divides the GPMC.FCLK clock
		0	GPMC_CLK frequency = GPMC_FCLK frequency
		1h	GPMC_CLK frequency = GPMC_FCLK frequency/2
		2h	GPMC_CLK frequency = GPMC_FCLK frequency/3
		3h	GPMC_CLK frequency = GPMC_FCLK frequency/4

9.5.12 GPMC_CONFIG2_i

Chip-select signal timing parameter configuration.

Figure 9-62. GPMC_CONFIG2_i

31	21	20	16
Reserved		CSWROFFTIME	
R-0		R/W-0	

15	13	12	8	7	6	4	3	0
Reserved		CSRDOFFTIME		CSEXTRADELAY	Reserved		CSONTIME	
R-0		R/W-0		R/W-0	R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-67. GPMC_CONFIG2_i Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20-16	CSWROFFTIME		CS# de-assertion time from start cycle time for write accesses
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		1Fh	31 GPMC_FCLK cycles
15-13	Reserved	0	Reserved
12-8	CSRDOFFTIME		CS# de-assertion time from start cycle time for read accesses
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		1Fh	31 GPMC_FCLK cycles
7	CSEXTRADELAY		CS# Add Extra Half GPMC.FCLK cycle
		0	CS i Timing control signal is not delayed
		1	CS i Timing control signal is delayed of half GPMC_FCLK clock cycle
6-4	Reserved	0	Reserved
3-0	CSONTIME		CS# assertion time from start cycle time
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		Fh	15 GPMC_FCLK cycles

9.5.13 GPMC_CONFIG3_i

ADV# signal timing parameter configuration.

Figure 9-63. GPMC_CONFIG3_i

31	30	28	27	26	24
Reserved	ADVAADMUXWROFFTIME		Reserved	ADVAADMUXRDOFFTIME	
R-0	R/W-0		R-0	R/W-0	

23	21	20	16
Reserved		ADVWROFFTIME	
R/W-0		R/W-0	

15	13	12	8
Reserved		ADVRDOFFTIME	
R-0		R/W-0	

7	6	4	3	0
ADVEXTRA DELAY	ADVAADMUXONTIME		ADVONTIME	
R/W-0	R/W-0		R/W-0	

LEGEND: R = Read only; W1C = Write 1 to clear bit; -*n* = value after reset

Table 9-68. GPMC_CONFIG3_i Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	ADVAADMUXWROFFTIME		ADV# de-assertion for first address phase when using the AAD-Mux protocol
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		7h	7 GPMC_FCLK cycles
27	Reserved	0	Reserved
26-24	ADVAADMUXRDOFFTIME		ADV# assertion for first address phase when using the AAD-Mux protocol
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		7h	7 GPMC_FCLK cycles
23-21	Reserved	0	Reserved
20-16	ADVWROFFTIME		ADV# de-assertion time from start cycle time for write accesses
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		1Fh	31 GPMC_FCLK cycles
15-13	Reserved	0	Reserved
12-8	ADVRDOFFTIME		ADV# de-assertion time from start cycle time for read accesses
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		1Fh	31 GPMC_FCLK cycles
7	ADVEXTRADELAY		ADV# Add Extra Half GPMC.FCLK cycle
		0	$\overline{\text{ADV}}$ Timing control signal is not delayed
		1	$\overline{\text{ADV}}$ Timing control signal is delayed of half GPMC_FCLK clock cycle

Table 9-68. GPMC_CONFIG3_i Field Descriptions (continued)

Bit	Field	Value	Description
6-4	ADVAADMUXONTIME		ADV# assertion for first address phase when using the AAD-Multiplexed protocol
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		7h	7 GPMC_FCLK cycles
3-0	ADVONTIME		ADV# assertion time from start cycle time
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		Fh	15 GPMC_FCLK cycles

9.5.14 GPMC_CONFIG4_i

WE# and OE# signals timing parameter configuration.

Figure 9-64. GPMC_CONFIG4_i

31	29	28	24
Reserved		WEOFFTIME	
R-0		R/W-0	

23	22	20	19	16
WEEXTRADELAY	Reserved		WEONTIME	
R/W-0	R-0		R/W-0	

15	13	12	8
OEAADMUXOFFTIME		OEOFFTIME	
R/W-0		R/W-0	

7	6	4	3	0
OEEXTRA DELAY	OEAADMUXONTIME		OEONTIME	
R/W-0	R/W-0		R/W-0	

LEGEND: R = Read only; W1C = Write 1 to clear bit; -*n* = value after reset

Table 9-69. GPMC_CONFIG4_i Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved
28-24	WEOFFTIME		WE# de-assertion time from start cycle time
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		1Fh	31 GPMC_FCLK cycles
23	WEEXTRADELAY		WE# Add Extra Half GPMC.FCLK cycle
		0	$\overline{WE}$ Timing control signal is not delayed
		1	$\overline{WE}$ Timing control signal is delayed of half GPMC_FCLK clock cycle
22-20	Reserved	0	Reserved
19-16	WEONTIME		WE# assertion time from start cycle time
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		Fh	15 GPMC_FCLK cycles
15-13	OEAADMUXOFFTIME		OE# de-assertion time for the first address phase in an AAD-Multiplexed access
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		7h	7 GPMC_FCLK cycles
12-8	OEOFFTIME		OE# de-assertion time from start cycle time
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		1Fh	31 GPMC_FCLK cycles
7	OEEXTRADELAY		OE# Add Extra Half GPMC.FCLK cycle
		0	$\overline{OE}$ Timing control signal is not delayed
		1	$\overline{OE}$ Timing control signal is delayed of half GPMC_FCLK clock cycle

Table 9-69. GPMC_CONFIG4_i Field Descriptions (continued)

Bit	Field	Value	Description
6-4	OEAADMUXONTIME		OE# assertion time for the first address phase in an AAD-Multiplexed access
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		7h	7 GPMC_FCLK cycles
3-0	OEONTIME		OE# assertion time from start cycle time
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		Fh	15 GPMC_FCLK cycles

9.5.15 GPMC_CONFIG5_i

RdAccessTime and CycleTime timing parameters configuration.

Figure 9-65. GPMC_CONFIG5_i

31 28	27 24	23 21	20 16
Reserved	PAGEBURSTACCESSTIME	Reserved	RDACCESSTIME
R-0	R/W-0	R-0	R/W-0

15 13	12 8	7 5	4 0
Reserved	WRCYCLETIME	Reserved	RDCYCLETIME
R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-70. GPMC_CONFIG5_i Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-24	PAGEBURSTACCESSTIME		Delay between successive words in a multiple access
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		Fh	15 GPMC_FCLK cycles
23-21	Reserved	0	Reserved
20-16	RDACCESSTIME		Delay between start cycle time and first data valid
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		1Fh	31 GPMC_FCLK cycles
15-13	Reserved	0	Reserved
12-8	WRCYCLETIME		Total write cycle time
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		1Fh	31 GPMC_FCLK cycles
7-5	Reserved	0	Reserved
4-0	RDCYCLETIME		Total read cycle time
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		1Fh	31 GPMC_FCLK cycles

9.5.16 GPMC_CONFIG6_i

WrAccessTime, WrDataOnADmuxBus, Cycle2Cycle, and BusTurnAround parameters configuration

Figure 9-66. GPMC_CONFIG6_i

31 29	28 24	23 20	19 16
Reserved	WRACCESSTIME	Reserved	WRDATAONADMUXBUS
R-0	R/W-0	R-0	R/W-0

15 12	11 8
Reserved	CYCLE2CYCLEDELAY
R-0	R/W-0

7	6	5 4	3 0
CYCLE2CYCLE SAMECSEN	CYCLE2CYCLE DIFFCSEN	Reserved	BUSTURNAROUND
R/W-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-71. GPMC_CONFIG6_i Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved
28-24	WRACCESSTIME		Delay from StartAccessTime to the GPMC.FCLK rising edge corresponding the the GPMC.CLK rising edge used by the attached memory for the first data capture
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		1Fh	31 GPMC_FCLK cycles
23-20	Reserved	0	Reserved
19-16	WRDATAONADMUXBUS	0-Fh	Specifies on which GPMC.FCLK rising edge the first data of the synchronous burst write is driven in the add/data multiplexed bus
15-12	Reserved	0	Reserved
11-8	CYCLE2CYCLEDELAY		Chip select high pulse delay between two successive accesses
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		Fh	15 GPMC_FCLK cycles
7	CYCLE2CYCLESAMECSEN		Add Cycle2CycleDelay between two successive accesses to the same chip-select (any access type)
		0	No delay between the two accesses
		1	Add CYCLE2CYCLEDELAY
6	CYCLE2CYCLEDIFFCSEN		Add Cycle2CycleDelay between two successive accesses to a different chip-select (any access type)
		0	No delay between the two accesses
		1	Add CYCLE2CYCLEDELAY
5-4	Reserved	0	Reserved
3-0	BUSTURNAROUND		Bus turn around latency between two successive accesses to the same chip-select (read to write) or to a different chip-select (read to read and read to write)
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		Fh	15 GPMC_FCLK cycles

9.5.17 GPMC_CONFIG7_i

Chip-select address mapping configuration.

Figure 9-67. GPMC_CONFIG7_i

31	16
Reserved	
R-0	

15	12	11	8	7	6	5	0
Reserved		MASKADDRESS		Rsvd	CSVALID	BASEADDRESS	
R-0		R/W-0		R-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-72. GPMC_CONFIG7_i Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11-8	MASKADDRESS		Chip-select mask address. Values not listed must be avoided as they create holes in the chip-select address space.
		0	Chip-select size of 256 Mbytes
		8h	Chip-select size of 128 Mbytes
		Ch	Chip-select size of 64 Mbytes
		Eh	Chip-select size of 32 Mbytes
		Fh	Chip-select size of 16 Mbytes
7	Reserved	0	Reserved
6	CSVALID		Chip-select enable (reset value is 1 for $\overline{CS}$[0] and 0 for $\overline{CS}$[1-5]).
		0	$\overline{CS}$ disabled
		1	$\overline{CS}$ enabled
5-0	BASEADDRESS	0-3Fh	Chip-select base address.
			CSi base address where i = 0 to 3 (16 Mbytes minimum granularity). Bits [5-0] corresponds to A29, A28, A27, A26, A25, and A24.

9.5.18 GPMC_NAND_COMMAND_i

This register is not a true register, just an address location.

Figure 9-68. GPMC_NAND_COMMAND_i

31	0
GPMC_NAND_COMMAND_i	
W-0	

LEGEND: W = Write only; -n = value after reset

Table 9-73. GPMC_NAND_COMMAND_i Field Descriptions

Bit	Field	Value	Description
31-0	GPMC_NAND_COMMAND_i	0-FFFF FFFFh	Writing data at the GPMC_NAND_COMMAND_i location places the data as the NAND command value on the bus, using a regular asynchronous write access.

9.5.19 GPMC_NAND_ADDRESS_i

This register is not a true register, just an address location.

Figure 9-69. GPMC_NAND_ADDRESS_i

31	0
GPMC_NAND_ADDRESS_i	
W-0	

LEGEND: W = Write only; -n = value after reset

Table 9-74. GPMC_NAND_ADDRESS_i Field Descriptions

Bit	Field	Value	Description
31-0	GPMC_NAND_ADDRESS_i	0-FFFF FFFFh	Writing data at the GPMC_NAND_ADDRESS_i location places the data as the NAND partial address value on the bus, using a regular asynchronous write access.

9.5.20 GPMC_NAND_DATA_i

This register is not a true register, just an address location.

Figure 9-70. GPMC_NAND_DATA_i

31	0
GPMC_NAND_DATA_i	
R/W	

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -n = value after reset

Table 9-75. GPMC_NAND_DATA_i Field Descriptions

Bit	Field	Value	Description
31-0	GPMC_NAND_DATA_i	0-FFFF FFFFh	Reading data from the GPMC_NAND_DATA_i location or from any location in the associated chip-select memory region activates an asynchronous read access.

9.5.21 GPMC_PREFETCH_CONFIG1

Figure 9-71. GPMC_PREFETCH_CONFIG1

31	30 28	27	26 24
Reserved	CYCLEOPTIMIZATION	ENABLEOPTIMIZED ACCESS	ENGINECSSELECTOR
R-0	R/W-0	R/W-0	R/W-0

23	22 20	19 16
PFPWENROUNDROBIN	Reserved	PFPWWEIGHTEDPRIO
R/W-0	R-0	R/W-0

15	14 8
Reserved	FIFOTHRESHOLD
R-0	R/W-0

7	6	5 4	3	2	1	0
ENABLEENGINE	Reserved	WAITPINSELECTOR	SYNCHROMODE	DMAMODE	Reserved	ACCESSMODE
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0

LEGEND: R = Read only; -*n* = value after reset

Table 9-76. GPMC_PREFETCH_CONFIG1 Field Descriptions

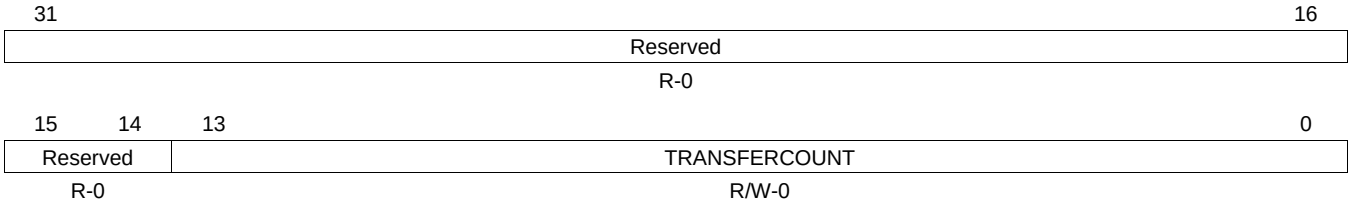
Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	CYCLEOPTIMIZATION		Define the number of GPMC.FCLK cycles to be substracted from RdCycleTime, WrCycleTime, RdAccessTime, CSRdOffTime, CSWrOffTime, ADVRdOffTime, ADVWrOffTime, OEOffTime, WEOffTime
		0	0 GPMC_FCLK cycle
		1h	1 GPMC_FCLK cycle
		⋮	⋮
		7h	7 GPMC_FCLK cycles
27	ENABLEOPTIMIZEDACCESS		Enables access cycle optimization
		0	Access cycle optimization is disabled
		1	Access cycle optimization is enabled
26-24	ENGINECSSELECTOR		Selects the $\overline{CS}$ where Prefetch Postwrite engine is active
		0	$\overline{CS0}$
		1h	$\overline{CS1}$
		2h	$\overline{CS2}$
		3h	$\overline{CS3}$
		4h	$\overline{CS4}$
		5h	$\overline{CS5}$
		6h	$\overline{CS6}$
		7h	$\overline{CS7}$
23	PFPWENROUNDROBIN		Enables the PFPW RoundRobin arbitration
		0	Prefetch Postwrite engine round robin arbitration is disabled
		1	Prefetch Postwrite engine round robin arbitration is enabled
22-20	Reserved	0	Reserved
19-16	PFPWWEIGHTEDPRIO		When an arbitration occurs between a direct memory access and a PFPW engine access, the direct memory access is always serviced. If the PFPWEnRoundRobin is enabled,
		0	The next access is granted to the PFPW engine
		1h	The two next accesses are granted to the PFPW engine
		⋮	⋮
		Fh	The 16 next accesses are granted to the PFPW engine
15	Reserved	0	Reserved

Table 9-76. GPMC_PREFETCH_CONFIG1 Field Descriptions (continued)

Bit	Field	Value	Description
14-8	FIFOTHRESHOLD		Selects the maximum number of bytes read from the FIFO or written to the FIFO by the host on a DMA or interrupt request
		0	0 byte
		1h	1 byte
		⋮	⋮
		40h	64 bytes
7	ENABLEENGINE		Enables the Prefetch Postwite engine
		0	Prefetch Postwrite engine is disabled
		1	Prefetch Postwrite engine is enabled
6	Reserved	0	Reserved
5-4	WAITPINSELECTOR		Select which WAIT pin edge detector should start the engine in synchronized mode
		0	Selects Wait0EdgeDetection
		1h	Selects Wait1EdgeDetection
		2h	Reserved
		3h	Reserved
3	SYNCHROMODE		Selects when the engine starts the access to CS
		0	Engine starts the access to CS as soon as STARTENGINE is set
		1	Engine starts the access to CS as soon as STARTENGINE is set AND wait to non wait edge detection on the selected wait pin
2	DMAMODE		Selects interrupt synchronization or DMA request synchronization
		0	Interrupt synchronization is enabled. Only interrupt line will be activated on FIFO threshold crossing.
		1	DMA request synchronization is enabled. A DMA request protocol is used.
1	Reserved	0	Reserved
0	ACCESSMODE		Selects pre-fetch read or write posting accesses
		0	Prefetch read mode
		1	Write-posting mode

9.5.22 GPMC_PREFETCH_CONFIG2

Figure 9-72. GPMC_PREFETCH_CONFIG2



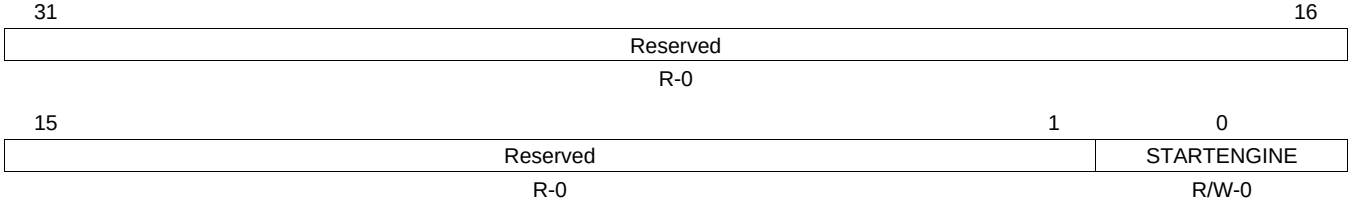
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-77. GPMC_PREFETCH_CONFIG2 Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13-0	TRANSFERCOUNT		Selects the number of bytes to be read or written by the engine to the selected $\overline{CS}$
		0	0 byte
		1h	1 byte
		⋮	⋮
		2000h	8 Kbytes

9.5.23 GPMC_PREFETCH_CONTROL

Figure 9-73. GPMC_PREFETCH_CONTROL



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-78. GPMC_PREFETCH_CONTROL Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	STARTENGINE		Resets the FIFO pointer and starts the engine
		R0	Engine is stopped
		W0	Stops the engine
		R1	Engine is running
		W1	Resets the FIFO pointer to 0 in prefetch mode and 40h in postwrite mode and starts the engine

9.5.24 GPMC_PREFETCH_STATUS

Figure 9-74. GPMC_PREFETCH_STATUS

31	30	24	23	17	16
Rsvd	FIFOPOINTER		Reserved		FIFOTHRESHOLDSTATUS
R-0	R-0		R-0		R-0

15	14	13	0
Reserved		COUNTVALUE	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-79. GPMC_PREFETCH_STATUS Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-24	FIFOPOINTER	0	0 byte available to be read or 0 free empty place to be written
		⋮	⋮
		40h	64 bytes available to be read or 64 empty places to be written
23-17	Reserved	0	Reserved
16	FIFOTHRESHOLDSTATUS		Set when FIFOPointer exceeds FIFOThreshold value
		0	FIFOPointer smaller or equal to FIFOThreshold. Writing to this bit has no effect
		1	FIFOPointer greater than FIFOThreshold. Writing to this bit has no effect
15-14	Reserved	0	Reserved
13-0	COUNTVALUE		Number of remaining bytes to be read or to be written by the engine according to the TransferCount value
		0	0 byte remaining to be read or to be written
		1h	1 byte remaining to be read or to be writte
		⋮	⋮
		2000h	8 Kbytes remaining to be read or to be written

9.5.25 GPMC_ECC_CONFIG

Figure 9-75. GPMC_ECC_CONFIG

31	17	16
Reserved		ECCALGORITHM
R-0		R/W-0

15	14	13	12	11	8
Reserved		ECCBCHTSEL		ECCWRAPMODE	
R-0		R/W-0		R/W-0	

7	6	4	3	1	0
ECC16B	ECCTOPSECTOR		ECCCS		ECCENABLE
R/W-0	R/W-0		R/W-0		R/W-0

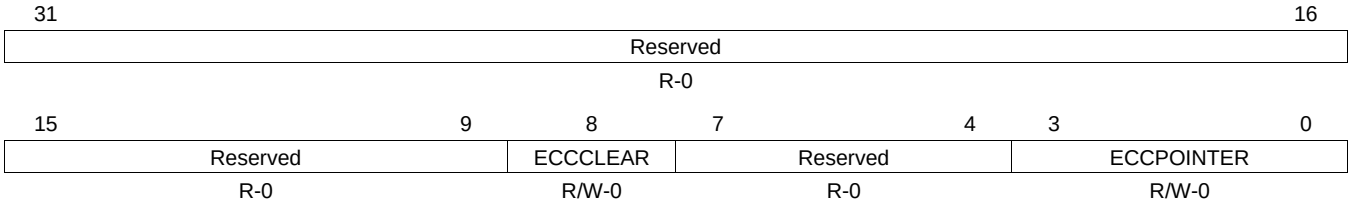
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-80. GPMC_ECC_CONFIG Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	ECCALGORITHM		ECC algorithm used
		0	Hamming code
		1	BCH code
15-14	Reserved	0	Reserved
13-12	ECCBCHTSEL		Error correction capability used for BCH
		0	Up to 4 bits error correction (t = 4)
		1h	Up to 8 bits error correction (t = 8)
		2h	Up to 16 bits error correction (t = 16)
		3h	Reserved
11-8	ECCWRAPMODE	0-Fh	Spare area organization definition for the BCH algorithm. See the BCH syndrome/parity calculator module functional specification for more details
7	ECC16B		Selects an ECC calculated on 16 columns
		0	ECC calculated on 8 columns
		1	ECC calculated on 16 columns
6-4	ECCTOPSECTOR		Number of sectors to process with the BCH algorithm
		0	1 sector (512 Bytes page)
		1h	2 sectors (2 × 512 Bytes = 1kB page)
		2h	3 sectors (3 × 512 Bytes)
		3h	4 sectors (4 × 512 Bytes = 2kB page)
		⋮	⋮
		7h	8 sectors (8 × 512 Bytes = 4kB page)
3-1	ECCCS		Selects the Chip-select where ECC is computed
		0	Chip-select 0
		1h	Chip-select 1
		2h	Chip-select 2
		3h	Chip-select 3
		4h	Chip-select 4
		5h	Chip-select 5
		6h-7h	Reserved
0	ECCENABLE		Enables the ECC feature
		0	ECC disabled
		1	ECC enabled

9.5.26 GPMC_ECC_CONTROL

Figure 9-76. GPMC_ECC_CONTROL



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-81. GPMC_ECC_CONTROL Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	ECCCLEAR		Clear all ECC result registers
		0	All reads return 0 Write of 0 is ignored
		1	Write of 1 clears all ECC result registers
7-4	Reserved	0	Reserved
3-0	ECCPOINTER		Selects ECC result register (Reads to this field give the dynamic position of the ECC pointer - Writes to this field select the ECC result register where the first ECC computation will be stored). Writing values not listed disables the ECC engine (ECCEnable bit of GPMC_ECC_CONFIG cleared to 0).
		0	Writing 0 disables the ECC engine (ECCENABLE bit of GPMC_ECC_CONFIG cleared to 0)
		1h	ECC result register 1 selected
		2h	ECC result register 2 selected
		3h	ECC result register 3 selected
		4h	ECC result register 4 selected
		5h	ECC result register 5 selected
		6h	ECC result register 6 selected
		7h	ECC result register 7 selected
		8h	ECC result register 8 selected
		9h	ECC result register 9 selected

9.5.27 GPMC_ECC_SIZE_CONFIG

Figure 9-77. GPMC_ECC_SIZE_CONFIG

31	30	29	22	21	20	19	16
Reserved		ECCSIZE1		Reserved		ECCSIZE0	
R-0		R/W-0		R-0		R/W-0	

15	12	11	9	8
ECCSIZE0		Reserved		ECC9RESULTSIZE
R/W-0		R-0		R/W-0

7	6	5	4	3	2	1	0
ECC8RESULTSIZE	ECC7RESULTSIZE	ECC6RESULTSIZE	ECC5RESULTSIZE	ECC4RESULTSIZE	ECC3RESULTSIZE	ECC2RESULTSIZE	ECC1RESULTSIZE
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-82. GPMC_ECC_SIZE_CONFIG Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved
29-22	ECCSIZE1		Defines ECC size 1
		0	2 Bytes
		1h	4 Bytes
		2h	6 Bytes
		3h	8 Bytes
		⋮	⋮
		FFh	512 Bytes
21-20	Reserved	0	Reserved
19-12	ECCSIZE0		Defines ECC size 0
		0	2 Bytes
		1h	4 Bytes
		2h	6 Bytes
		3h	8 Bytes
		⋮	⋮
		FFh	512 Bytes
11-9	Reserved	0	Reserved
8	ECC9RESULTSIZE		Selects ECC size for ECC 9 result register
		0	ECCSIZE0 selected
		1	ECCSIZE1 selected
7	ECC8RESULTSIZE		Selects ECC size for ECC 8 result register
		0	ECCSIZE0 selected
		1	ECCSIZE1 selected
6	ECC7RESULTSIZE		Selects ECC size for ECC 7 result register
		0	ECCSIZE0 selected
		1	ECCSIZE1 selected
5	ECC6RESULTSIZE		Selects ECC size for ECC 6 result register
		0	ECCSIZE0 selected
		1	ECCSIZE1 selected
4	ECC5RESULTSIZE		Selects ECC size for ECC 5 result register
		0	ECCSIZE0 selected
		1	ECCSIZE1 selected

Table 9-82. GPMC_ECC_SIZE_CONFIG Field Descriptions (continued)

Bit	Field	Value	Description
3	ECC4RESULTSIZE		Selects ECC size for ECC 4 result register
		0	ECCSIZE0 selected
		1	ECCSIZE1 selected
2	ECC3RESULTSIZE		Selects ECC size for ECC 3 result register
		0	ECCSIZE0 selected
		1	ECCSIZE1 selected
1	ECC2RESULTSIZE		Selects ECC size for ECC 2 result register
		0	ECCSIZE0 selected
		1	ECCSIZE1 selected
0	ECC1RESULTSIZE		Selects ECC size for ECC 1 result register
		0	ECCSIZE0 selected
		1	ECCSIZE1 selected

9.5.28 GPMC_ECCj_RESULT

Figure 9-78. GPMC_ECCj_RESULT

31	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		P2048O	P1024O	P512O	P256O	P128O	P64O	P32O	P16O	P8O	P4O	P2O	P1O
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

15	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		P2048E	P1024E	P512E	P256E	P128E	P64E	P32E	P16E	P8E	P4E	P2E	P1E
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

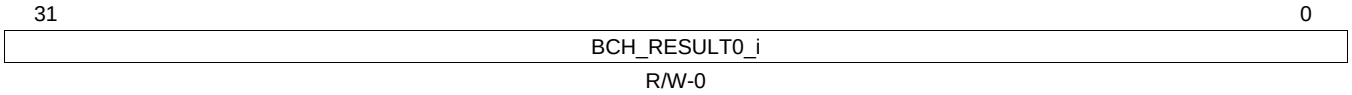
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-83. GPMC_ECCj_RESULT Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27	P2048O	0-1	Odd Row Parity bit 2048, only used for ECC computed on 512 Bytes
26	P1024O	0-1	Odd Row Parity bit 1024
25	P512O	0-1	Odd Row Parity bit 512
24	P256O	0-1	Odd Row Parity bit 256
23	P128O	0-1	Odd Row Parity bit 128
22	P64O	0-1	Odd Row Parity bit 64
21	P32O	0-1	Odd Row Parity bit 32
20	P16O	0-1	Odd Row Parity bit 16
19	P8O	0-1	Odd Row Parity bit 8
18	P4O	0-1	Odd Column Parity bit 4
17	P2O	0-1	Odd Column Parity bit 2
16	P1O	0-1	Odd Column Parity bit 1
15-12	Reserved	0	Reserved
11	P2048E	0-1	Even Row Parity bit 2048, only used for ECC computed on 512 Bytes
10	P1024E	0-1	Even Row Parity bit 1024
9	P512E	0-1	Even Row Parity bit 512
8	P256E	0-1	Even Row Parity bit 256
7	P128E	0-1	Even Row Parity bit 128
6	P64E	0-1	Even Row Parity bit 64
5	P32E	0-1	Even Row Parity bit 32
4	P16E	0-1	Even Row Parity bit 16
3	P8E	0-1	Even Row Parity bit 8
2	P4E	0-1	Even Column Parity bit 4
1	P2E	0-1	Even Column Parity bit 2
0	P1E	0-1	Even Column Parity bit 1

9.5.29 GPMC_BCH_RESULT0_i

Figure 9-79. GPMC_BCH_RESULT0_i



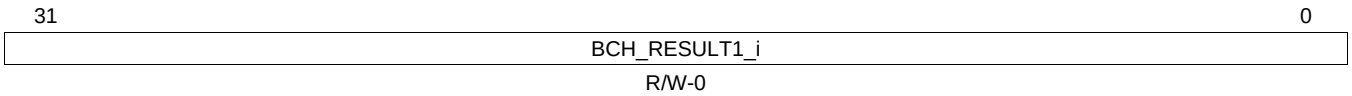
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-84. GPMC_BCH_RESULT0_i Field Descriptions

Bit	Field	Value	Description
31-0	BCH_RESULT0_i	0-FFFF FFFFh	BCH ECC result, bits 0 to 31

9.5.30 GPMC_BCH_RESULT1_i

Figure 9-80. GPMC_BCH_RESULT1_i



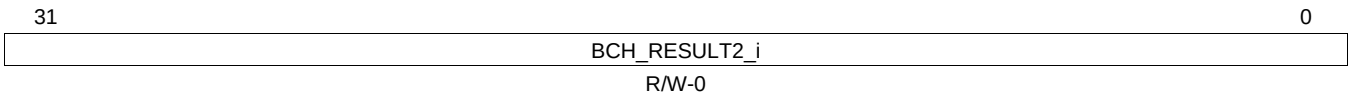
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-85. GPMC_BCH_RESULT1_i Field Descriptions

Bit	Field	Value	Description
31-0	BCH_RESULT1_i	0-FFFF FFFFh	BCH ECC result, bits 32 to 63

9.5.31 GPMC_BCH_RESULT2_i

Figure 9-81. GPMC_BCH_RESULT2_i



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-86. GPMC_BCH_RESULT2_i Field Descriptions

Bit	Field	Value	Description
31-0	BCH_RESULT2_i	0-FFFF FFFFh	BCH ECC result, bits 64 to 95

9.5.32 GPMC_BCH_RESULT3_i

Figure 9-82. GPMC_BCH_RESULT3_i

31	0
BCH_RESULT3_i	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-87. GPMC_BCH_RESULT3_i Field Descriptions

Bit	Field	Value	Description
31-0	BCH_RESULT3_i	0-FFFF FFFFh	BCH ECC result, bits 96 to 127

9.5.33 GPMC_BCH_SWDATA

This register is used to directly pass data to the BCH ECC calculator without accessing the actual NAND flash interface.

Figure 9-83. GPMC_BCH_SWDATA

31	16
Reserved	
R-0	

15	0
BCH_DATA	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-88. GPMC_BCH_SWDATA Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	BCH_DATA	0-FFFFh	Data to be included in the BCH calculation. Only bits 0 to 7 are taken into account, if the calculator is configured to use 8 bits data (GPMC_ECC_CONFIG[7] ECC16B = 0).

9.5.34 GPMC_BCH_RESULT4_i

Figure 9-84. GPMC_BCH_RESULT4_i

31	0
BCH_RESULT4_i	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-89. GPMC_BCH_RESULT4_i Field Descriptions

Bit	Field	Value	Description
31-0	BCH_RESULT4_i	0-FFFF FFFFh	BCH ECC result, bits 128 to 159

9.5.35 GPMC_BCH_RESULT5_i

Figure 9-85. GPMC_BCH_RESULT5_i

31	0
BCH_RESULT5_i	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-90. GPMC_BCH_RESULT5_i Field Descriptions

Bit	Field	Value	Description
31-0	BCH_RESULT5_i	0-FFFF FFFFh	BCH ECC result, bits 160 to 191

9.5.36 GPMC_BCH_RESULT6_i

Figure 9-86. GPMC_BCH_RESULT6_i

31	0
BCH_RESULT6_i	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 9-91. GPMC_BCH_RESULT6_i Field Descriptions

Bit	Field	Value	Description
31-0	BCH_RESULT6_i	0-FFFF FFFFh	BCH ECC result, bits 192 to 207

High-Definition Multimedia Interface (HDMI)

This chapter describes the high-definition multimedia interface (HDMI) module.

Topic **Page**

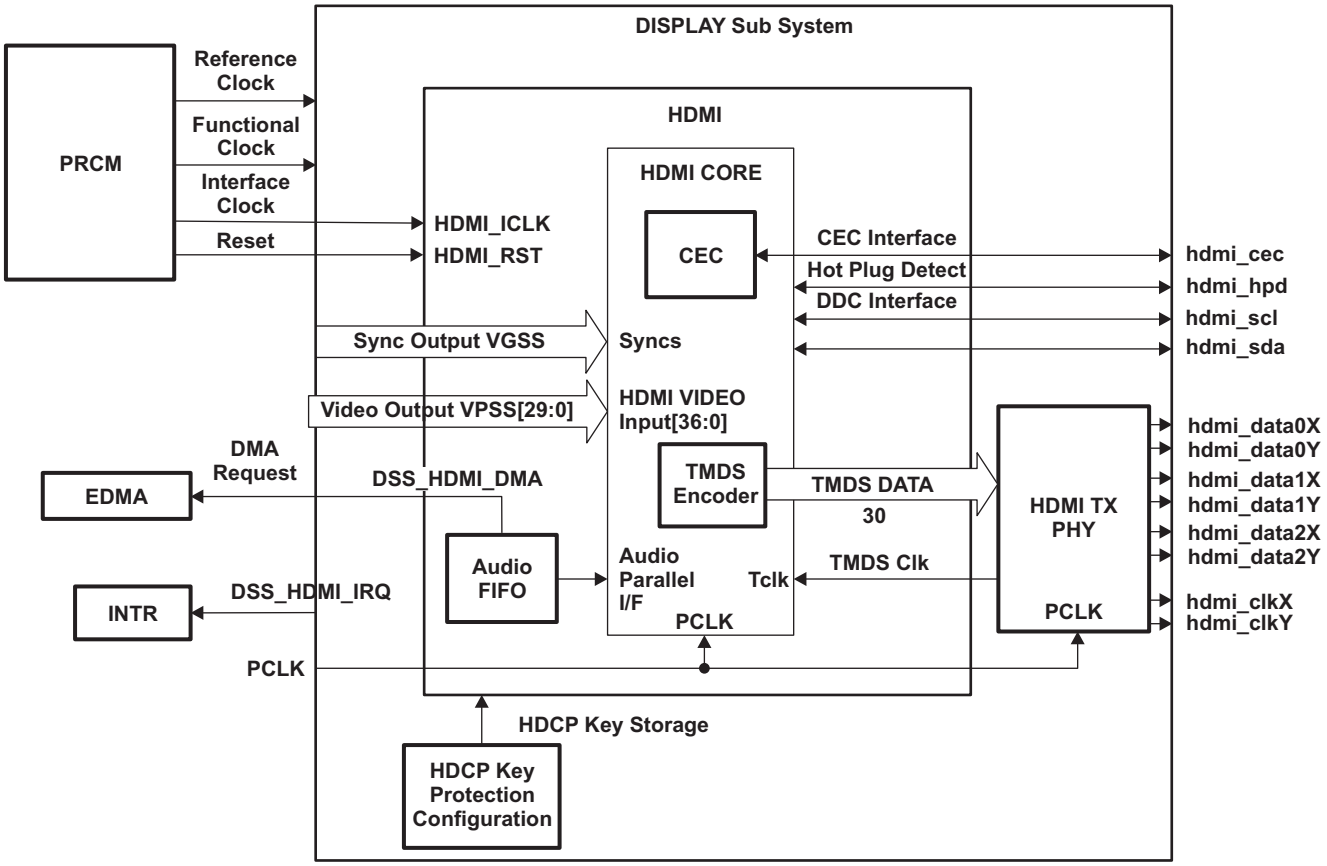
10.1 Introduction

10.1.1 Overview

The high-definition multimedia interface wrapper (HDMI_WP) module is used to encapsulate the HDMI, providing compatibility and interface to the HDMI TXPHY module. The HDMI_WP module contains the support logic for the HDMI IP, including the HDMI core (HDMI_CORE) and CEC core. The major additional parts comprise a slave interface port for configuration and buffering for the audio data streams.

Figure 10-1 is an overview of the HDMI module.

Figure 10-1. HDMI Overview



10.1.2 Main Features

The HDMI module provides the following main features:

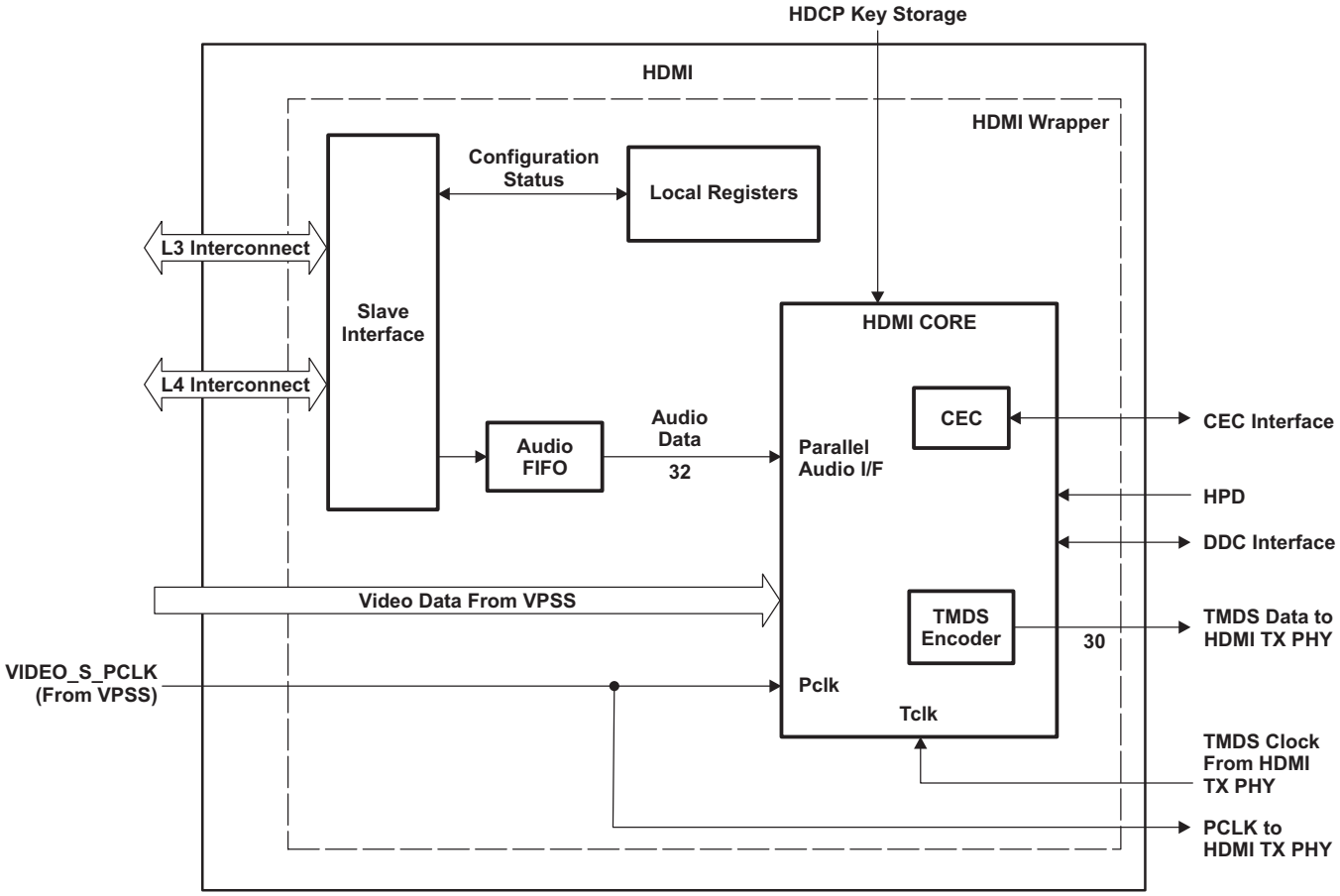
- HDMI 1.3, HDCP 1.2, and DVI 1.0 compliant
- EIA/CEA-861-D video format support
- VESA Display Monitor Timing (DMT) video format support
- Support for deep-color mode:
  - 10-bit/component color depth up to 1080p at 60 Hz
  - 12-bit/component color depth up to 720p/1080i at 60 Hz
- Supports up to 165-MHz pixel clock (1920 × 1080p at 60 Hz)
- Video formats: 24-bit RGB
- Uncompressed multichannel (up to eight channels) audio (L-PCM) support
- Master inter-intergrated circuit (I2C) interface for display data channel (DDC) connection

- Consumer Electronic Control (CEC) interface
- Integrated High-bandwidth Digital Content Protection (HDCP) encryption engine for transmitting protected audio and video content (authentication performed by software)
- Integrated Transition Minimized Differential Signaling (TMDS) and TER4 encoders for data island support
- Integrated TMDS physical layer (PHY) (three TMDS differential data lanes + TMDS differential clock lane)
  - Up to 1.85625 Gbps per lane at (1080p at 60 Hz at 10 bits/component, lower resolutions at 12 bits/component)
  - 928.125 Mbps per lane at (720p/1080i at 60 Hz 10 bits/component, lower resolutions at 12 bits/component)
  - 337.5 Mbps at (576p at 50 Hz/480p at 60 Hz 10 bits/component)

10.1.3 Functional Block Diagram

Figure 10-2 is a functional block diagram of the HDMI module.

Figure 10-2. HDMI Block Diagram



10.1.4 Video Formats and Timings

Table 10-1 lists the video timings supported by the HDMI module, according to the CEA-861-D standard.

Table 10-2 lists the video timings supported by the HDMI module, according to the VESA DMT standard.

Table 10-1. HDMI Video Timings (CEA-861-D)

Refresh Rate	Resolution
24 Hz (low-field rate)	1920 x 1080p
50 Hz	1920 x 1080p
	1920 x 1080i
	1280 x 720p
	2880 x 576p
	1440 x 576p
	1440 x 576i
	720 x 576p
59.94 Hz	1920 x 1080p
	2880 x 480p
	1440 x 480p
	1440 x 480i
	720 x 480p
	640 x 480p
60 Hz	1920 x 1080p
	1920 x 1080i
	2880 x 480p
	1280 x 720p
	1440 x 480p
	1440 x 480i
	720 x 480p
	640 x 480p

Table 10-2. HDMI Video Timings (VESA DMT)

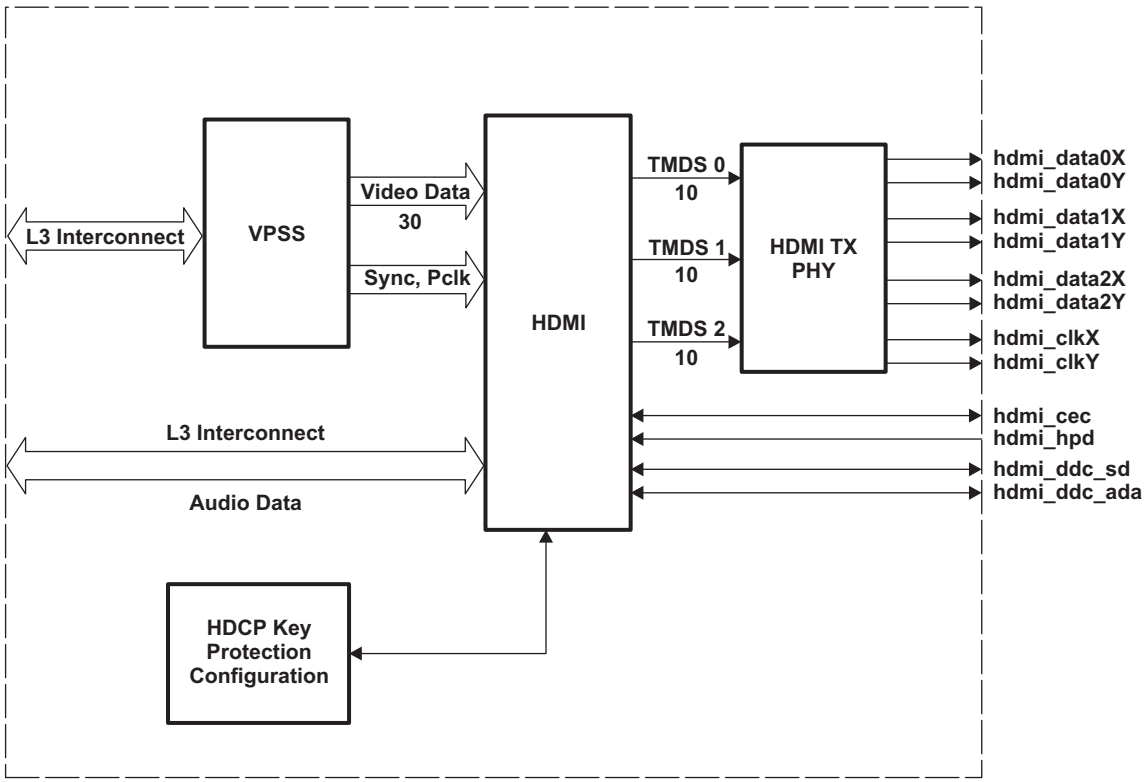
Refresh Rate	Resolution
60 Hz	640 x 480p
	800 x 600p
	848 x 480p
	1024 x 768p
	1280 x 768p
	1280 x 800p
	1280 x 960p
	1280 x 1024p
	1360 x 768p
	1366 x 768p
	1400 x 1050p
	1440 x 900p
	1680 x 1050p
	1920 x 1080p
60 Hz (reduced blanking)	1280 x 768p
	1280 x 800p
	1400 x 1050p
	1440 x 900p
	1680 x 1050p

10.1.5 Environment

This section describes the HDMI application fields from an environment point of view (external connections). It describes HDMI connectivity options, lists all possible interfaces, and describes the protocol and data format used in each case.

Figure 10-3 shows the external connections and simplified data path for the HDMI module.

Figure 10-3. HDMI Environment



The following paths are used to transmit video and audio data through the HDMI link:

- Video data path:
  - Video Processing Sub System (VPSS)
  - HDMI module
  - HDMI complex input/output (I/O) (HDMI_TXPHY)

The VPSS module provides synchronization signals to the HDMI and synchronously sends 30 bits of pixel data to the HDMI with these signals. The digital output of the VPSS is always a 30-bit RGB value .For more information about the connection between the VPSS and HDMI, see Section 1.4.7.1, HDMI Video Interface. For more information about the VPSS settings and features, see the Video Processing Sub System chapter in the device TRM. The video data received on the HDMI video slave port is always a 30-bit RGB value. The HDMI module performs pixel processing that allows manipulation of the pixels received on the video interface. The processed data is sent to the HDCP for encryption, or directly to the HDMI_TXPHY module after TDMS encoding, if there is no HDCP encryption enabled.

- Audio data path:
  - Level 3 (L3) interconnect
  - HDMI module
  - HDMI complex I/O (HDMI_TXPHY)

The audio data is received on the buffered HDMI audio port through the L3 Interconnect using a direct memory access (DMA) request to the device system DMA (EDMA) module, or interrupt request (IRQ) to the device microprocessor unit (MPU) interrupt controllers (INTCs). In the HDMI, the data is packed, formatted, and sent to the HDMI_TXPHY module.

Additionally, the HDMI module provides universal remote control capability through the CEC protocol on the hdmi_cec pin, which allows common communication between different pieces of consumer electronics when the device is connected to them by HDMI cable.

The HDMI module can drive the display data channel (DDC) bus, which supports a variety of I2C commands, on the hdmi_ddc_scl and hdmi_ddc_sda pins. This allows the TV-set/monitor to provide information to the host device, such as supported resolutions.

Hot plug detection (HPD) capability is supported by the HDMI module on the hdmi_hpd pin. It allows detection of the presence of an attached TV display.

10.2 Architecture

This section describes the HDMI configuration.

10.2.1 Clock Configuration

10.2.1.1 Clock Domains

There are three main clock domains within the HDMI module:

- L3 clock domain: Runs at the frequency of the DSS_L3_ICLK clock, generated by the device PRCM module, and is asynchronous with the other clocks
- TCLK clock domain: Runs at the frequency of the TMDS clock (TMDS_CLK) generated by the HDMI_TXPHY module
- PCLK clock domain: VPSS to HDMI_WP input runs at PCLK generated by VPSS Venc.

10.2.1.2 CEC Interface Clock Configuration

The clock for the CEC interface module (CEC_DDC_CLK) is generated inside the HDMI module from the DSS_HDMI clock (48 MHz) by division. The divider value is set through the HDMI_WP_CLK[5:0] CEC_DIV bit field, as defined in Table 10-3. The clock frequency required by the CEC protocol is 2 MHz, and is achieved by setting the CEC_DIV bit field to 18h.

Table 10-3. CEC Clock Generation

Reference Clock	HDMI_WP_CLK[5:0] CEC_DIV Value	CEC Clock
DSS_HDMI	0	Gated
-	1	Free running
-	...	...
-	18h	2-MHz clock
-	...	...

10.2.1.3 Time-Out Capability

The HDMI module provides time-out capability to the DSS_HDMI clock. The OCP_TIME_OUT_INTR interrupt is generated, if the HDMI_WP_CLK[16] OCP_TIME_OUT_DIS bit is cleared to 0 and the DSS_HDMI clock is not provided to the HDMI module. For more information about the OCP_TIME_OUT_INTR interrupt, see Section 10.2.6. The time-out capability can be disabled by writing 1 to the HDMI_WP_CLK[16] OCP_TIME_OUT_DIS bit. By default, this option is enabled.

10.2.2 Signals

Table 10-4 describes the HDMI module signals.

Table 10-4. HDMI I/O Signal Description

Signal	I/O[1]	Description	Function
hdmi_data0x hdmi_data0y	O	Transmit data lane 0	TMDS serial data output
hdmi_data1x hdmi_data1y	O	Transmit data lane 1	TMDS serial data output
hdmi_data2x hdmi_data2y	O	Transmit data lane 2	TMDS serial data output
hdmi_clockx hdmi_clocky	O	Transmit clock lane	TMDS clock output
hdmi_cec	I/O	CEC interface line	CEC interface input/output
hdmi_hpd	I	Monitor charge input	Used to connect to the HDMI hot plug pin to detect the presence of an attached monitor
hdmi_ddc_scl	I/O	DDC I2C clock line	DDC clock input/output
hdmi_ddc_sda	I/O	DDC I2C clock line	DDC clock input/output

(1) I = Input; O = Output

10.2.3 Integration

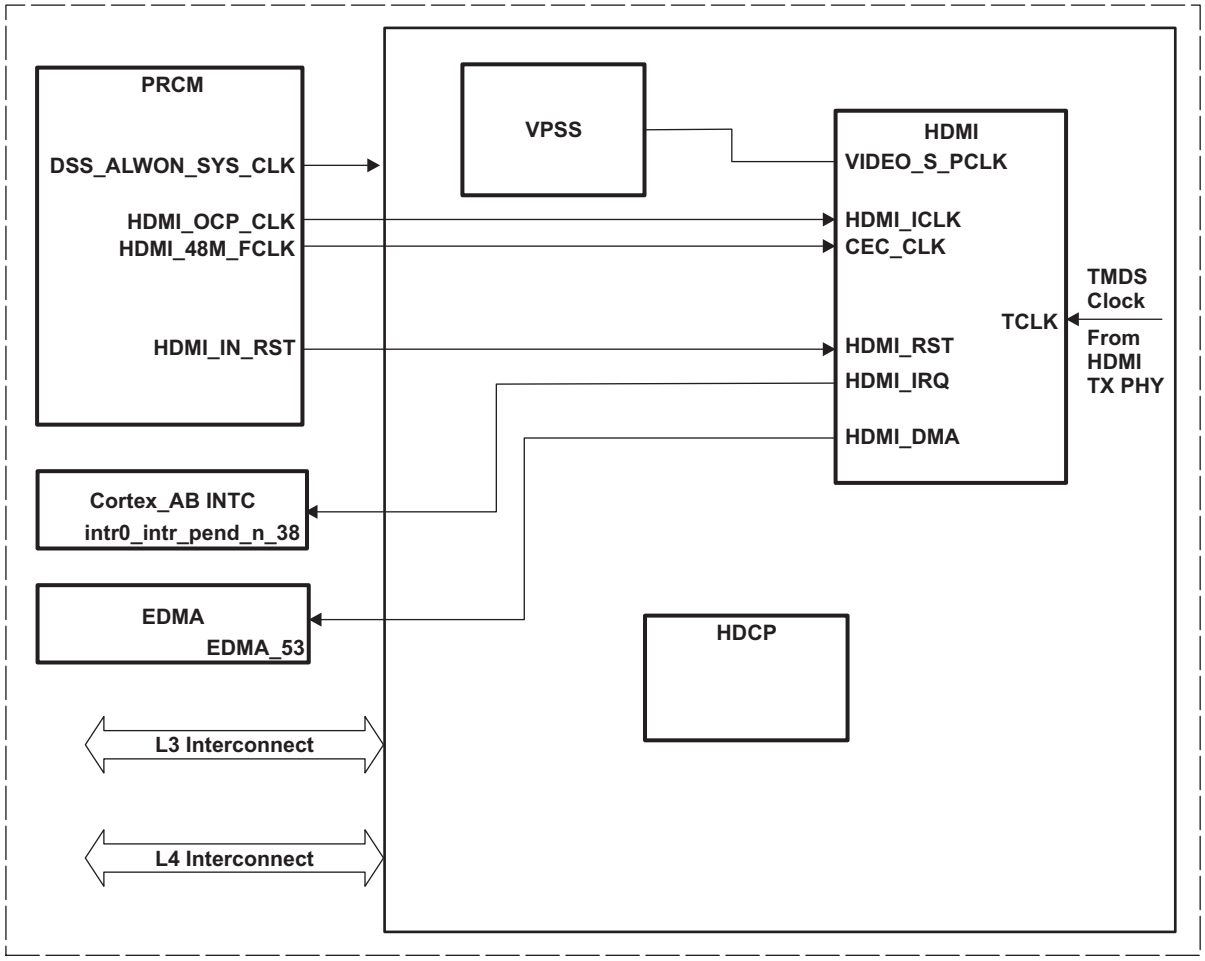
This section describes the module integration in the device (see Figure 10-4), including information about clocks, resets, and hardware requests.

Table 10-5, Table 10-6, and Table 10-7 summarize the integration of the module in the device.

10.2.4 Software Reset

Global reset of the module is done at the PRCM module level. The HDMI module can be reset by software. This has the same effect as a hardware reset. The HDMI module can be reset by setting the HDMI_WP_SYSCONFIG[0] SOFTRESET bit to 1. Software can monitor this bit to wait for the reset to complete.

Figure 10-4. HDMI Integration



NOTE: For more information about the IDLE hardware handshake and the wake-up request, see Sleep (Idle) and Wake-Up Management, in Power, Reset, and Clock Management.

Table 10-5. Integration Attributes

Module Instance	Attributes
HDMI	Power Domain — Interconnect Active Domain —L3 for Data; L4 for Configuration

Table 10-6. Clocks and Resets

Module Instance	Destination Signal Name	Source Signal Name	Source	Destination
		Clocks		
HDMI	DSS_HDMI	HDMI_PHY_48_S_FCLK	PRCM	The CEC/DDC clock is derived from this clock. Also reference clock for HDMI_TXPHY module.
	HDMI_ICLK	DSS_L3_ICLK	PRCM	Interface clock
	TCLK	TMDS_CLK	HDMI_TXPHY	The TMDS encoded data is sent to the HDMI_TXPHY module on this clock.
VPSS	VIDEO_S_PCLK	HDMI_CLK	VPSS	HDMI
		Resets		
HDMI	HDMI_RST	HDMI_IN_RST	PRCM	Nonretention reset

Table 10-7. Hardware Requests

Module Instance	Desination Signal Name	Source Signal Name	Destination	Description
HDMI	DSS_HDMI_IRQ	intr0_intr_pend_n_38	CortexA8	HDMI Interrupt Request
HDMI	DSS_HDMI_DMA	EDMA_53	EDMA	HDMI audio DMA request

10.2.5 Power Management

Table 10-8 describes the power-management features available to the HDMI module.

NOTE: For information about source clock gating and a description of the sleep/wake-up transitions, see the Sleep (Idle) and Wake-Up Management section in the Power, Reset, and Clock Management..

For descriptions of the EnaWakeUp, IdleMode, ClockActivity, and StandbyMode features, see the AutoIDLE Clock Control section in the Power, Reset, and Clock Management.

Table 10-8. Local Power-Management Features

Feature	Registers	Description
Slave idle modes	HDMI_WP_SYSCONFIG[3:2] IDLEMODE bit field	Force-idle, no-idle, smart-idle, and smart-idle wakeup-capable modes are available.

10.2.6 Interrupt Requests

One external interrupt line is generated by the HDMI module: DSS_HDMI_IRQ (for more information, see Table 10-7). Table 10-9 lists the event flags, and their masks, that can cause internal module interrupts that are mapped to DSS_HDMI_IRQ.

Table 10-9. HDMI Interrupt Events

Event Flag	Event Mask	Map To	Description
HDMI_WP_IRQSTATUS[10] AUDIO_FIFO_SAMPLE_REQ_INTR	HDMI_WP_IRQENABLE_SET[10] ENABLE_SET_AUDIO_FIFO_ SAMPLE_REQ_INTR	AUDIO_FIFO_SAMPLE_ REQ_INTR	Audio FIFO requests some data in IRQ mode
HDMI_WP_IRQSTATUS[9] AUDIO_FIFO_OVERFLOW_INTR	HDMI_WP_IRQENABLE_SET[9] ENABLE_SET_AUDIO_FIFO_ OVERFLOW_INTR	AUDIO_FIFO_ OVERFLOW_INTR	Audio FIFO overflow
HDMI_WP_IRQSTATUS[8] AUDIO_FIFO_UNDERFLOW_INTR	HDMI_WP_IRQENABLE_SET[8] ENABLE_SET_AUDIO_FIFO_ UNDERFLOW_INTR	AUDIO_FIFO_ UNDERFLOW_INTR	Audio FIFO underflow
HDMI_WP_IRQSTATUS[4] OCP_TIME_OUT_INTR	HDMI_WP_IRQENABLE_SET[4] OCP_TIME_OUT_INTR	OCP_TIME_OUT_INTR	Time-out interrupt in case the DSS_HDMI clock is not provided
HDMI_WP_IRQSTATUS[0] CORE_INTR	HDMI_WP_IRQENABLE_ SET[0] ENABLE_SET_CORE_INTR	CORE_INTR	HDMI core has generated internally an interrupt. Software must read the interrupt status register in the HDMI core to identify the interrupt event(s).

NOTE: The HDMI_WP_IRQENABLE_SET register is used to enable the interrupt event(s), by writing 1 in the desired bit field(s). To disable the interrupt event(s), the HDMI_WP_IRQENABLE_CLR register must used, by writing 1 in the respective bit field(s).

10.2.7 DMA Requests

The HDMI module generates one DMA request for audio data: DSS_HDMI_DMA (for more information, see Section 10.2.8.2.2.1).

10.2.8 Wrapper Functions

10.2.8.1 Wrapper Video Interface

10.2.8.1.1 Video Interface Features

- Video slave port with 30 bit RGB data format (deep color mode)
- Slave operational mode:
  - Synchronization signals, pixel clock, data-enable and data pixels are provided to the HDMI module
- CEA 861-D video timings support at 50 Hz, 59.94 Hz, and 60 Hz (low field rate 24 Hz included)
- VESA DMT video timings support at 60 Hz (reduced blanking included)
  - 10 bits per color component for 1080p/60fps and 1080p/50fps – 12 bits per color component otherwise
- Support for TCLK up to 185.625 MHz (TCLK derived from PCLK)

10.2.8.1.2 Video Interface Description

NOTE: The signals described in this section are internal and not bounded outside the device.

The HDMI module receives synchronization signals, pixel clock, and 30 bits of pixel data on its video slave port from VPSS module (Table 10-10). The digital output of the VPSS is always a 30-bit RGB value. This is extended to 36 bits by adding LSB zeroes, as listed in Table 10-11.

Table 10-10. Video Port Signals

Signal Name	Width	Direction	Description
VIDEO_S_PCLK	---	IN	Input pixel clock (PCLK)
VIDEO_S_DATA	35:0	IN	Video data input bus
VIDEO_S_DE	---	IN	Data Enable
VIDEO_S_HS	---	IN	Horizontal sync
VIDEO_S_VS	---	IN	Vertical sync
VIDEO_S_FID	---	IN	Field identifier

Table 10-11. HDMI Video Port Mapping

Color Component	VPSS HD_VENC Output Bus	HDMI Video Data Input Bus	Color Component
R9	VPSS_DATA[29]	HDMI_VIDEO_DATA[35]	R11
R8	VPSS_DATA[28]	HDMI_VIDEO_DATA[34]	R10
R7	VPSS_DATA[27]	HDMI_VIDEO_DATA[33]	R9
R6	VPSS_DATA[26]	HDMI_VIDEO_DATA[32]	R8
R5	VPSS_DATA[25]	HDMI_VIDEO_DATA[31]	R7
R4	VPSS_DATA[24]	HDMI_VIDEO_DATA[30]	R6
R3	VPSS_DATA[23]	HDMI_VIDEO_DATA[29]	R5
R2	VPSS_DATA[22]	HDMI_VIDEO_DATA[28]	R4
R1	VPSS_DATA[21]	HDMI_VIDEO_DATA[27]	R3
R0	VPSS_DATA[20]	HDMI_VIDEO_DATA[26]	R2
GND	GND	HDMI_VIDEO_DATA[25]	R1
GND	GND	HDMI_VIDEO_DATA[24]	R0
G9	VPSS_DATA[19]	HDMI_VIDEO_DATA[23]	G11
G8	VPSS_DATA[18]	HDMI_VIDEO_DATA[22]	R10
G7	VPSS_DATA[17]	HDMI_VIDEO_DATA[21]	G9
G6	VPSS_DATA[16]	HDMI_VIDEO_DATA[20]	G8
G5	VPSS_DATA[15]	HDMI_VIDEO_DATA[19]	G7
G4	VPSS_DATA[14]	HDMI_VIDEO_DATA[18]	G6
G3	VPSS_DATA[13]	HDMI_VIDEO_DATA[17]	G5
G2	VPSS_DATA[12]	HDMI_VIDEO_DATA[16]	G4
G1	VPSS_DATA[11]	HDMI_VIDEO_DATA[15]	G3
G0	VPSS_DATA[10]	HDMI_VIDEO_DATA[14]	G2
GND	GND	HDMI_VIDEO_DATA[13]	G1
GND	GND	HDMI_VIDEO_DATA[12]	G0
B9	VPSS_DATA[9]	HDMI_VIDEO_DATA[11]	B11
B8	VPSS_DATA[8]	HDMI_VIDEO_DATA[10]	B10
B7	VPSS_DATA[7]	HDMI_VIDEO_DATA[9]	B9
B6	VPSS_DATA[6]	HDMI_VIDEO_DATA[8]	B8
B5	VPSS_DATA[5]	HDMI_VIDEO_DATA[7]	B7

Table 10-11. HDMI Video Port Mapping (continued)

Color Component	VPSS HD_VENC Output Bus	HDMI Video Data Input Bus	Color Component
B4	VPSS_DATA[4]	HDMI_VIDEO_DATA[6]	B6
B3	VPSS_DATA[3]	HDMI_VIDEO_DATA[5]	B5
B2	VPSS_DATA[2]	HDMI_VIDEO_DATA[4]	B4
B1	VPSS_DATA[1]	HDMI_VIDEO_DATA[3]	B3
B0	VPSS_DATA[0]	HDMI_VIDEO_DATA[2]	B2
GND	GND	HDMI_VIDEO_DATA[1]	B1
GND	GND	HDMI_VIDEO_DATA[0]	B0

10.2.8.1.3 Video Interface Configuration

The slave port is connected to the HDMI module. Always clear the HDMI_WP_VIDEO_CFG[1:0] Mode bit field to 0.

According to the video data bus connection and pixel format, packing mode must be selected. The format supported by the HDMI module is 36-bit RGB. The packing can be selected through the HDMI_WP_VIDEO_CFG[10:8] PACKING_MODE bit field:

- HDMI_WP_VIDEO_CFG[10:8] PACKING_MODE = 0: If 10-bit deep-color mode is selected and

RGB format is used on the video port data bus:

- HDMI_WP_VIDEO_CFG[10:8] PACKING_MODE = 1: If 8-bit deep-color mode is not selected and

RGB format is used on the video port data bus:

- HDMI_WP_VIDEO_CFG[10:8] PACKING_MODE = 7h: If 12-bit deep-color mode is selected and

RGB format is used on the video port data bus. No change on input video data lines. The VIDEO_S_DATA[35:0] data is provided without any change in it.

Video Timing is controlled by VENC module inside VPSS which drives HDMI module. Please refer VPSS PRG for information about timing for different resolutions.

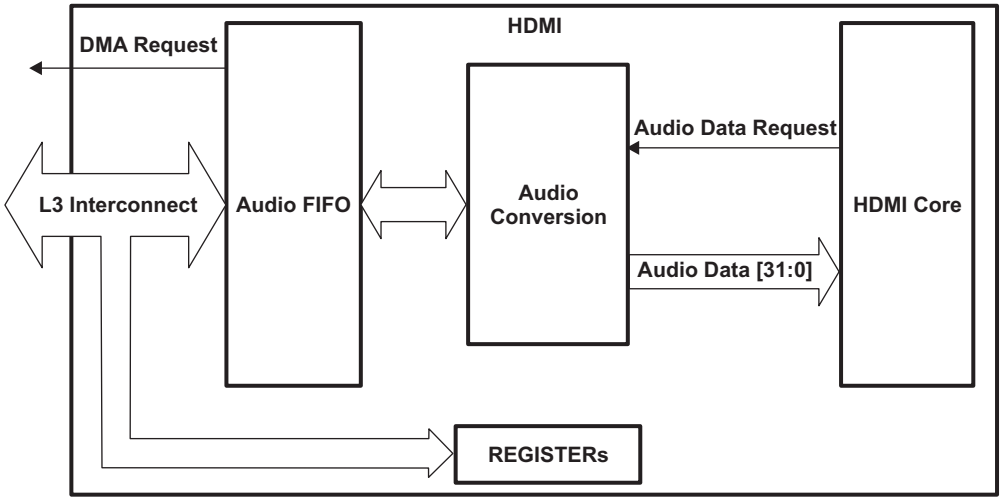
10.2.8.2 Wrapper Audio Interface

10.2.8.2.1 Audio Interface Description

The HDMI module supports parallel interface that is based on a filling of the audio FIFO through the L3 interconnect. The audio data and register configuration is received through the L3 interconnect on a request to the device EDMA module. Each 32-bit access is split into four consecutive accesses to the HDMI Core 8-bit registers of the 32-bit access, starting with the least-significant bit (LSB) to the MSB.

Figure 10-5 is an overview of the audio interface to the HDMI core.

Figure 10-5. HDMI Audio Interface Overview



10.2.8.2.2 Audio FIFO Overview

The audio FIFO provides buffering to ensure that audio samples can always be delivered to the HDMI core exactly when required, despite the L3 interconnect latency.

10.2.8.2.2.1 Audio Data Request

By default, the audio FIFO is filled by the device EDMA using a standard DMA request signal mapped to DSS_HDMI_DMA on the display subsystem level. For additional mapping information, see Table 10-7.

The FIFO can also use an IRQ signal (AUDIO_FIFO_SAMPLE_REQ_INTR) to request audio data, instead of a DMA request, using the same threshold. The IRQ status can be read in the HDMI_WP_IRQSTATUS[10] AUDIO_FIFO_SAMPLE_REQ_INTR bit. For more information about the audio IRQ, see Table 10-9.

The selection between DMA or IRQ request can be done through the HDMI_WP_AUDIO_CTRL[9] DMA_OR_IRQ bit.

The audio data is written in the HDMI_WP_AUDIO_DATA[31:0] FIFO_DAT bit field.

The request is made based on comparison of fullness with register programmable thresholds. The DMA/IRQ is generated when the number of samples (24- or 16-bit) in the FIFO is less than or equal to the threshold value.

- The size of samples is selected through the HDMI_WP_AUDIO_CFG[0] SAMPLE_SIZE bit.
- The valid number of samples can be read from the HDMI_WP_AUDIO_CTRL[25:16].

NUMBER_OF_SAMPLE bit field:

- The threshold value can be set in the HDMI_WP_AUDIO_CTRL[8:0] TRESHOLD_VALUE bit field.

The DSS_HDMI_DMA signal is released (deactivated) when the last transfer of the DMA burst is performed in the FIFO. An internal counter is used to do this. The reset value of this counter is set to 16 and the maximum value is set to 64. In this case, the maximum length of the DMA transfer is 64 (32-bit interface access). The counter can be configured through the HDMI_WP_AUDIO_CFG2[15:8] DMA_TRANSFER bit field.

The threshold value and the DMA burst size must be set to ensure the filling of the FIFO to four samples. The minimum level (otherwise, underflow occurs) is four audio samples, which can be 2 × 32-bit data (in case of two samples per word) or 4 × 32-bit data (IEC or one sample per word). The interconnect latency must be considered. The number of samples is controlled through the HDMI_WP_AUDIO_CFG[1] SAMPLE_NBR bit.

10.2.8.2.2.2 Audio FIFO Error (Underflow and Overflow)

The underflow and overflow status are required and highlighted by sending interrupt requests IRQs).

The FIFO underflow condition occurs when the HDMI core is requesting data from the FIFO when the FIFO is empty. The status can be read through the HDMI_WP_IRQSTATUS[8] AUDIO_FIFO_UNDERFLOW_INTR bit.

The underflow IRQ may happen during a sequence only if one of the following occurs:

- The audio data path is not set correctly. In this case, the device EDMA cannot fill the FIFO fast enough.
- It is the end of the sequence and the number of samples is not a multiple of two times the value in the HDMI_WP_AUDIO_CFG2[7:0] BLOCK_SIZE bit field (384 samples by default).

The FIFO filling must be ensured by the system before the HDMI core is enabled; this means that in steady state the filling of the FIFO is always four words of 32 bits (except at the end).

The FIFO overflow condition occurs when an audio data is written (by the device EDMA or MPU modules) while the FIFO is full. The status can be read through the HDMI_WP_IRQSTATUS[9] AUDIO_FIFO_OVERFLOW_INTR bit.

The overflow IRQ may happen only if:

- The device EDMA module does not respect the DMA transfer length.
- The threshold is not set to the correct value in the HDMI_WP_AUDIO_CTRL[8:0] TRESHOLD_VALUE bit field.
- The device MPU module does not check the filling of the FIFO when the audio data is requested through an IRQ.

10.2.8.2.2.3 Audio FIFO Reset

The audio sample in the FIFO can be flushed by writing 0 in the HDMI_WP_AUDIO_CTRL[31] WRAPPER_ENABLE bit. As soon as the enable signal is deactivated (set to 0),the following occur: 1. The audio FIFO is reset. 2. The read and write pointers are reset. 3. The DSS_HDMI_DMA signal is deasserted. 4. Only the threshold value in the HDMI_WP_AUDIO_CTRL[8:0] TRESHOLD_VALUE bit field is kept.

10.2.8.2.3 Audio FIFO Data Formats

The format of the audio data in the audio FIFO can be configured through the HDMI_WP_AUDIO_CFG[4] IEC bit field.

- IEC = 0: The audio format is L-PCM.
- IEC = 1: The audio format is IEC 60958/IEC 61937.

10.2.8.2.3.1 L-PCM Format in Audio FIFO

Table 10-12 describes the 16-bit data format in the audio FIFO.

Table 10-12. PCM, 16-Bit Format

Order	Bits [31:16]	Bits [15:0]
---	16 bits, 2 channels (stereo)	---
1	Right - Channel 2	Left - Channel 1
---	16 bits, 4 channels	---
1	Right - Channel 2	Left - Channel 1
2	Right - Channel 4	Left - Channel 3
3	Right - Channel 2	Left - Channel 1
4	Right - Channel 4	Left - Channel 3
---	16 bits, 8 channels	---
1	Right - Channel 2	Left - Channel 1

Table 10-12. PCM, 16-Bit Format (continued)

Order	Bits [31:16]	Bits [15:0]
2	Right - Channel 4	Left - Channel 3
3	Right - Channel 6	Left - Channel 5
4	Right - Channel 8	Left - Channel 7

Table 10-13 describes the 24-bit data format in the audio FIFO. For the following formats, if less than 24 bits are valid, the data can be MSB- or LSB-justified. This is controlled through the HDMI_WP_AUDIO_CFG[3] JUSTIFY bit field.

Table 10-13. PCM, 24-Bit Format

Order	Bits [31:24]	Bits [23:0]
---	24 bits, 2 channels (stereo right - justified)	---
1	Unused	Left - Channel 1
2	Unused	Right - Channel 2
---	24 bits, 2 channels stereo left - justified	---
1	Left - Channel 1	Unused
2	Right - Channel 2	Unused
---	24 bits, 3 channels stereo right - justified	---
1	Unused	Left - Channel 1
2	Unused	Right - Channel 2
3	Unused	Center - Channel 3
4	Unused	Left - Channel 1
5	Unused	Right - Channel 2
6	Unused	Center - Channel 3
---	24 bits, 8 channels (right-justified)	---
1	Unused	Left - Channel 1
2	Unused	Right - Channel 2
3	Unused	Left - Channel 3
4	Unused	Right - Channel 4
5	Unused	Left - Channel 5
6	Unused	Right - Channel 6
7	Unused	Left - Channel 7
8	Unused	Right - Channel 8

NOTE: Left justification for three- and eight-channel data formats is also possible.

10.2.8.2.3.2 Speaker Mapping and Channel Ordering in the Audio FIFO

The speaker mapping in the eight channels is defined by the CEA-861-D specification and is described in Table 10-14. The CEA code is used to program the Packet Byte 4 of the Audio Info FRAME register inside the HDMI core (for more information, see the HDMI standard v1.3).

Table 10-14. Speaker Mapping Versus Channel

CEA Code	8	7	6	5	4	3	2	1
0h	-	-	-	-	-	-	FR	FL
1h	-	-	-	-	-	LFE	FR	FL
2h	-	-	-	-	FC	-	FR	FL

Table 10-14. Speaker Mapping Versus Channel (continued)

CEA Code	8	7	6	5	4	3	2	1
3h	-	-	-	-	FC	LFE	FR	FL
4h	-	-	-	RC	-	-	FR	FL
5h	-	-	-	RC	-	LFE	FR	FL
6h	-	-	-	RC	FC	-	FR	FL
7h	-	-	-	RC	FC	LFE	FR	FL
8h	-	-	RR	RL	-	-	FR	FL
9h	-	-	RR	RL	-	LFE	FR	FL
Ah	-	-	RR	RL	FC	-	FR	FL
Bh	-	-	RR	RL	FC	LFE	FR	FL
Ch	-	RC	RR	RL	-	-	FR	FL
Dh	-	RC	RR	RL	-	LFE	FR	FL
Eh	-	RC	RR	RL	FC	-	FR	FL
Fh	-	RC	RR	RL	FC	LFE	FR	FL
10h	RRC	RLC	RR	RL	-	-	FR	FL
11h	RRC	RLC	RR	RL	-	LFE	FR	FL
12h	RRC	RLC	RR	RL	FC	-	FR	FL
13h	RRC	RLC	RR	RL	FC	LFE	FR	FL
14h	FRC	FLC	-	-	-	-	FR	FL
15h	FRC	FLC	-	-	-	LFE	FR	FL
16h	FRC	FLC	-	-	FC	-	FR	FL
17h	FRC	FLC	-	-	FC	LFE	FR	FL
18h	FRC	FLC	-	RC	-	-	FR	FL
19h	FRC	FLC	-	RC	-	LFE	FR	FL
1Ah	FRC	FLC	-	RC	FC	-	FR	FL
1Bh	FRC	FLC	-	RC	FC	LFE	FR	FL
1Ch	FRC	FLC	RR	RL	-	-	FR	FL
1Dh	FRC	FLC	RR	RL	-	LFE	FR	FL
1Eh	FRC	FLC	RR	RL	FC	-	FR	FL
1Fh	FRC	FLC	RR	RL	FC	LFE	FR	FL
20h	-	FCH	RR	RL	FC	-	FR	FL
21h	-	FCH	RR	RL	FC	LFE	FR	FL
22h	TC	-	RR	RL	FC	-	FR	FL
23h	TC	-	RR	RL	FC	LFE	FR	FL
24h	FRH	FLH	RR	RL	-	-	FR	FL
25h	FRH	FLH	RR	RL	-	LFE	FR	FL
26h	FRW	FLW	RR	RL	-	-	FR	FL
27h	FRW	FLW	RR	RL	-	LFE	FR	FL
28h	TC	RC	RR	RL	FC	-	FR	FL
29h	TC	RC	RR	RL	FC	LFE	FR	FL
2Ah	FCH	RC	RR	RL	FC	-	FR	FL
2Bh	FCH	RC	RR	RL	FC	LFE	FR	FL
2Ch	TC	FCH	RR	RL	FC	-	FR	FL
2Dh	TC	FCH	RR	RL	FC	LFE	FR	FL
2Eh	FRH	FLH	RR	RL	FC	-	FR	FL
2Fh	FRH	FLH	RR	RL	FC	LFE	FR	FL
30h	FRW	FLW	RR	RL	FC	-	FR	FL
31h	FRW	FLW	RR	RL	FC	LFE	FR	FL

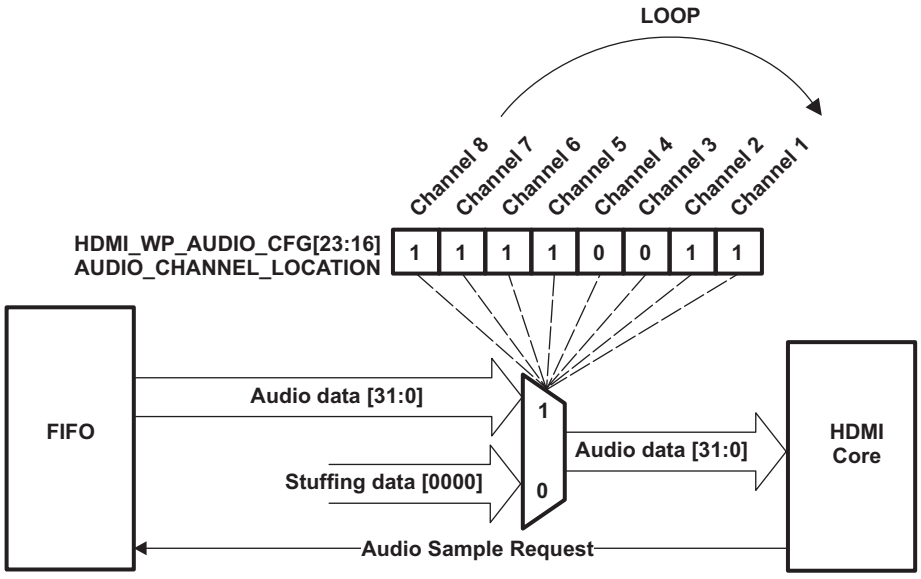
In Table 10-14, the following acronym definitions are used:

- FL: Front left
- FC: Front center
- FR: Front right
- FLC: Front left center
- FRC: Front right center
- RL: Rear left
- RC: Rear center
- RR: Rear right
- RLC: Rear left center
- RRC: Rear right center
- LFE: Low-frequency effect
- FLW: Front left wide
- FRW: Front right wide
- FLH: Front left high
- FCH: Front center high
- FRH: Front right high
- TC: Top center

The audio FIFO can be filled from two to eight channels, but the HDMI core supports only an even number of channels (the missing sample must be filled with 0, depending on the audio format).

The HDMI_WP_AUDIO_CFG[23:16] AUDIO_CHANNEL_LOCATION bit field indicates the location and the active channels. The AUDIO_CHANNEL_LOCATION field contains 8-bit fields, each corresponding to one channel. The HDMI_WP_AUDIO_CFG[26:24] STEREO_CHANNEL_ENABLE bit field indicates the number of stereo channels used. In Figure 10-6, the registers are filled accordingly with the CEA channel/speaker allocation code 10h received from the HDMI core. Upon a data request from the HDMI core, the data coming from the audio FIFO or the stuffing (filling zeros) data are sent.

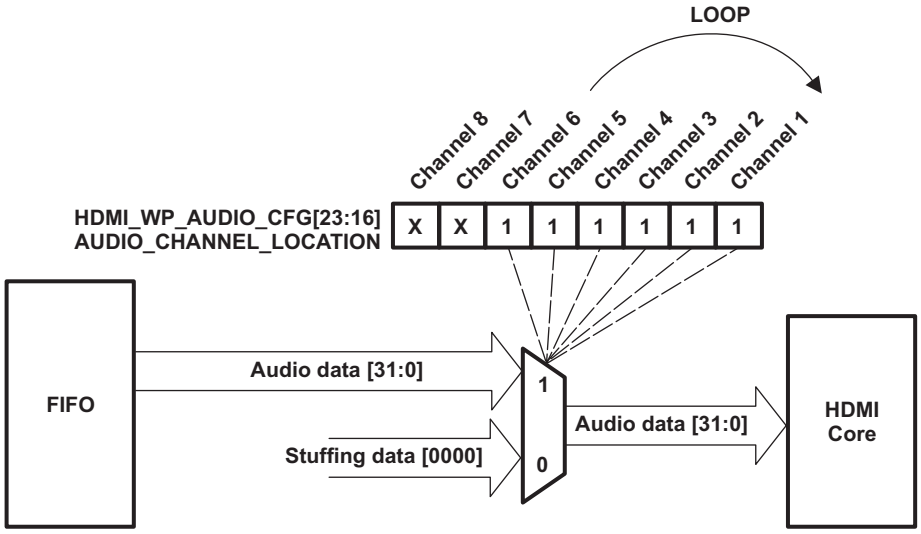
Figure 10-6. Audio Data Stuffing Behavior



If only three stereo channels are set in the HDMI wrapper, the loop can be optimized and set to 6. If three stereo channels are enabled in the HDMI core module, the function must be set to 3h in the HDMI_WP_AUDIO_CFG[26:24] STEREO_CHANNEL_ENABLE bit field.

Figure 10-7 shows an example with the CEA channel/speaker allocation code 7h and only three stereo channels enabled in the HDMI core. This format is also supported with the four channels enabled, but channel 7/8 must be cleared to 0.

Figure 10-7. Audio Data Stuffing Behavior with Only Three Stereo Channels Active



10.2.8.2.3.3 IEC 61958 Format

In the IEC 61958 format (Table 10-15), the channel status and user data bits are supplied with the audio data.

The valid bits are supplied with the audio data but may be overridden based on the audio FIFO operation(forced to invalid, if FIFO data is invalid). They may also be forced to valid or invalid.

This format supports only two channels (stereo) or two monophonic channels.

The audio FIFO can be filled directly with data aligned with the 60958 format. In this case, the VUCP and the preamble data are available.

Table 10-15. IEC 60958 Sample Format

Order	Bits [31:28]	Bits [27:4]	Bits [3:0]
1	VUCP	Left - Channel 1	Preamble
2	VUCP	Right - Channel 2	Preamble

10.2.8.2.3.4 IEC 61937 Format

The IEC 61937 format (Table 10-16) is similar to the IEC 61958 format, with multichannel supported.

Table 10-16. IEC 60937 Format

Order	Bits [31:28]	Bits [27:4]	Bits [3:0]
1	VUCP	Left - Channel 1	Preamble
2	VUCP	Right - Channel 2	Preamble
3	VUCP	Left - Channel 3	Preamble
4	VUCP	Right - Channel 4	Preamble
5	VUCP	Left - Channel 5	Preamble
6	VUCP	Right - Channel 6	Preamble

Table 10-16. IEC 60937 Format (continued)

Order	Bits [31:28]	Bits [27:4]	Bits [3:0]
7	VUCP	Left - Channel 7	Preamble
8	VUCP	Right - Channel 8	Preamble

NOTE: The compressed formats are supported only if software can stuff the missing data with zeros to reach the bandwidth defined in the IEC 60958 format.

10.2.8.2.4 AUDIO FIFO Data Adaptation

The HDMI core supports only the IEC 60958 format, or similar formats. Between the audio FIFO and the HDMI core, a format adaptation is done to optimize the audio data transfer.

There are three different formats in the audio FIFO:

- Two L-PCM samples on 16 bits in a 32-bit container
- One L-PCM sample on 24 bits in a 32-bit container
- IEC 61937 or IEC 60958 sample format compliant

The audio formats supported include 16- and 24-bit modes for stereo (two channels) and 8-channel modes.

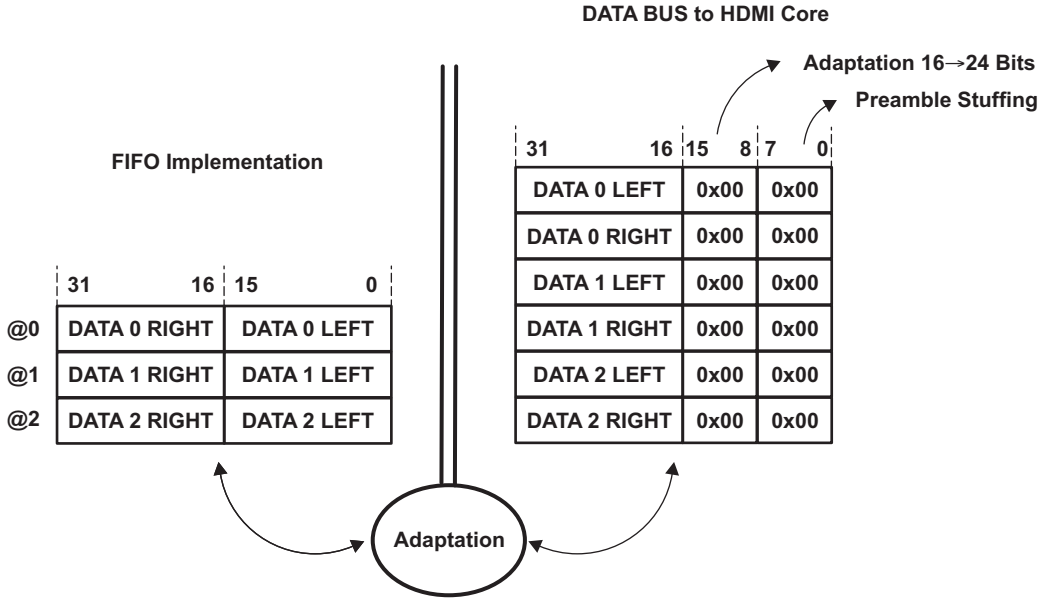
10.2.8.2.4.1 IEC 61937 and IEC 60958 Format Adaptation

The IEC 61937 and IEC 60958 formats are fully compatible with the HDMI core. In this case, data in the FIFO is sent to the HDMI core without adaptation. This is true for 16- and 24-bit modes and it is channel-number independent.

10.2.8.2.4.2 L-PCM 16-Bit Format Adaptation

When the audio FIFO is filled with 16-bit samples on the 32-bit container, the data is adapted to be similar to the IEC 61937/IEC 60958 format, as shown in Figure 10-8.

Figure 10-8. L-PCM 16-Bit Format Adaptation



10.2.8.2.4.3 L-PCM 24-bit Format Adaptation

When the audio FIFO is filled with a 24-bit sample on the 32-bit container, it is already in a format similar to IEC 61937/IEC 60958, left- or right-justified, depending on the FIFO filling (the HDMI_WP_AUDIO_CFG[3] JUSTIFY bit). In this case, the data in the audio FIFO is sent directly to the HDMI core without adaptation. However, the 8 dummy bits (MSB or LSB, depending on the justification) are filled with 0 data.

10.2.9 TXPHY Functions

10.2.9.1 TXPHY Overview

NOTE: The HDMI_TXPHY module must be configured before any transfer on the HDMI link. The HDMI module is fully functional only when the TMDS clock is provided. It has an internal PLL which will take in PCLK as an input and provide the required TCLK based on register configuration.

The HDMI_TXPHY receives 30 bits of TMDS parallel encoded data from the HDMI module and then prepares the data for transmission by serializing it. HDMI_TXPHY is a complex I/O with four unidirectional lane modules. This includes three data lane modules and one clock lane module. Each lane module has two data pads (DX and DY), which form a differential pair. These correspond to the HDMI serial output transmit signals, as described in Table 10-4. The data pads are connected with a complementary lane module on the external HDMI receiver device using a point-to-point interconnect.

The maximum data rate supported is 1.85625 Gbps per data lane. The lane module function and position are configurable. That is, any lane module can be chosen as a clock or a data lane module, and the DX/DY data pad for each lane module can be configured as a positive polarity (DP) or negative polarity (DN) pin.

10.2.9.2 TXPHY Clock Domains

The HDMI_TXPHY module has the following clock domains:

- TMDS clock domain: This is the data interface clock to the HDMI module. The data received on the HDMI_TXPHY inputs are sampled on this clock.
- HFBITCLK clock domain: This is the high frequency clock on which the transmit of the serialized data occurs.
- REFCLK clock domain: This is the 48-MHz free-running clock (DSS_HDMI) that is used as a reference for the switched capacitor circuit.

10.2.9.3 TXPHY RX Connection Detect

The HDMI_TXPHY can detect a pullup to 3.3 V on any clock and data lines. This situation occurs when an external HDMI receiver is connected to the device.

The information can be read by reading TMDS_CNTL2[0], RSEN.

10.2.9.4 TXPHY Data Interface

10.2.9.4.1 Input Interface

The HDMI_TXPHY module receives the 30-bit parallel data on its D0 to D2 inputs. The data is sampled on the rising edge of the TMDS clock.

10.2.9.4.2 Output Interface

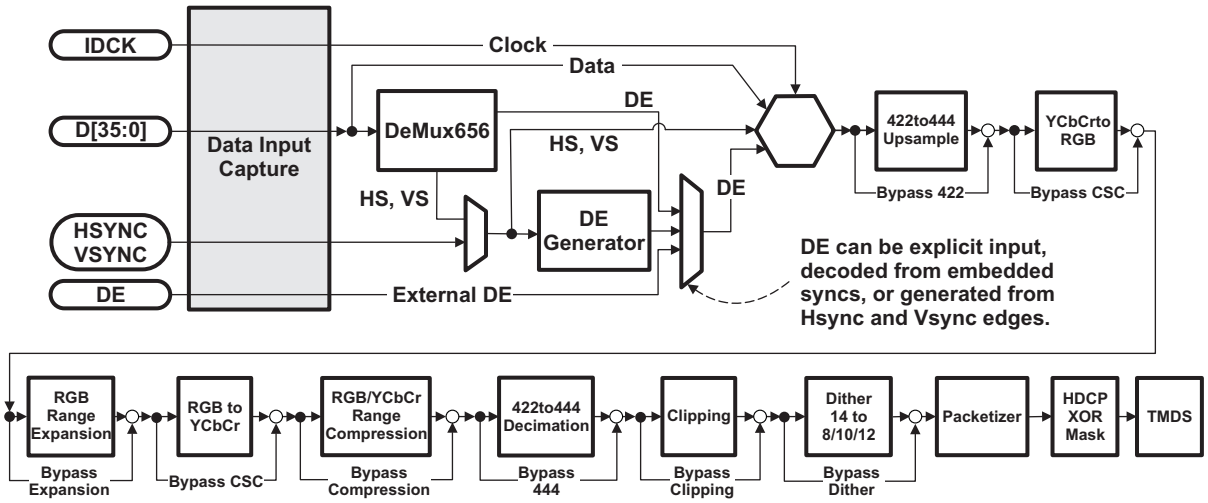
The serialized data on differential output lines DX/DY has the order LSB first and MSB last. The TMDS clock on the clock lane has a falling edge at the start of the 10-bit serial data word on the line.

10.2.10 Programming Video Input Mode and Video Output Mode

Specific registers in the HDMI Transmitter must be programmed according to the selection of input video bus mode and output video format. See Figure 10-9.

- The DE parameters need to be programmed only when the DE Generator is enabled. It is not recommended to enable the DE Generator.
- Set RANGE to 1 whenever converting YCbCr data and outputting full-range (0-255) RGB (PC mode) data across HDMI. When outputting limited-range (16-235) RGB (CE mode) data across the link, clear RANGE to 0.
- Set WIDE_BUS to the number of bits per input video channel.
- Set DITHER_MODE to the number of bits per output video channel supported by the Sink. By default, the HDMI Transmitter dithers the input (if DITHER is enabled) or truncates the input (if DITHER is disabled) to 8 bits.
- PB_CTRL1 and PB_CTRL2 should be programmed once the core is powered up only.
- Always clear (set to 0) the DE_ADJ field in IADJUST register when DE is not enabled (Recommended usage).
- DIV_ENC_BYP field in TEST_TXCTRL register should always be programmed to zero.
- ten_bit_bypass in TMDS_CNTL9 register and enc_byp in BIST_CNTL register should always be set to 1.
- Program the HDMI registers and then enable the VENC on VPSS side. All timings are provided by the VENC module in VPSS.

Figure 10-9. Transmitter Video Data Processing Path



10.3 HDMI Registers

Table 10-17 shows the base address offset for the HDMI registers. For the base address of the wrapper and core registers, see Table 1-11. For the base address of the PHY registers, see Table 1-12.

Table 10-17. HDMI Registers

Base Address Offset	Module	Section
0000h	HDMI Wrapper Registers	Section 10.3.1
0400h	HDMI Core System Registers	Section 10.3.2
0800h	HDMI IP Core Gamut Registers	Section 10.3.3
0900h	HDMI IP Core Audio Video Registers	Section 10.3.4
0D00h	HDMI IP Core CEC Registers	Section 10.3.5
2000h	HDMI PHY Registers	Section 10.3.6

10.3.1 HDMI Wrapper Registers

Table 10-18 lists the HDMI wrapper registers.

Table 10-18. HDMI Wrapper Registers

Address Offset	Acronym	Register Name
00h	HDMI_WP_REVISION	IP Revision Identifier
10h	HDMI_WP_SYSCONFIG	Clock management configuration
24h	HDMI_WP_IRQSTATUS_RAW	Raw Interrupt Status
28h	HDMI_WP_IRQSTATUS	Interrupt Status
2Ch	HDMI_WP_IRQENABLE_SET	Interrupt Enable
30h	HDMI_WP_IRQENABLE_CLR	Interrupt Disable
44h	HDMI_WP_DEBOUNCE	Glitch filter
50h	HDMI_WP_VIDEO_CFG	Configuration of HDMI Wrapper video
70h	HDMI_WP_CLK	Configuration of clocks
80h	HDMI_WP_AUDIO_CFG	Audio Configuration in FIFO
84h	HDMI_WP_AUDIO_CFG2	Audio configuration of DMA
88h	HDMI_WP_AUDIO_CTRL	Audio FIFO control
8Ch	HDMI_WP_AUDIO_DATA	TX Data of FIFO

10.3.1.1 IP Revision Identifier Register (HDMI_WP_REVISION)

The IP revision identifier register is shown in Figure 10-10 and described in Table 10-19.

Figure 10-10. IP Revision Identifier Register (HDMI_WP_REVISION)

31	0
Revision	
R-5003 0200h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

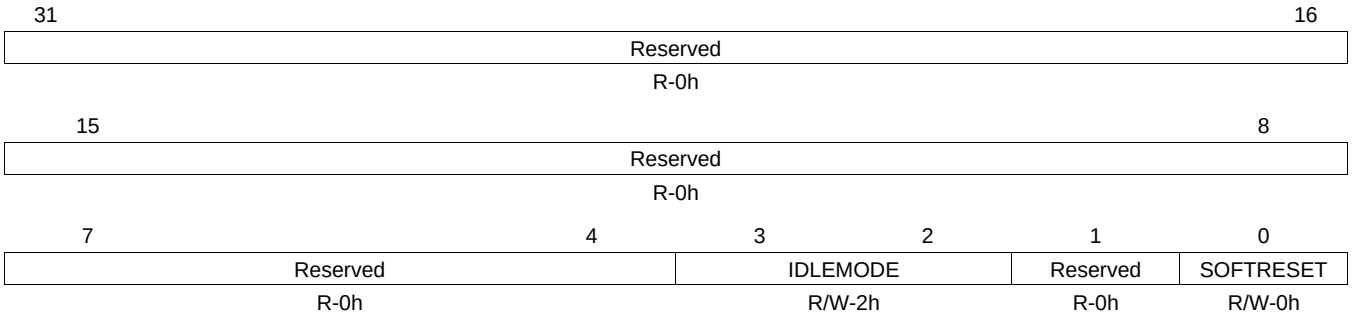
Table 10-19. IP Revision Identifier Register (HDMI_WP_REVISION) Field Descriptions

Bit	Field	Description
31-0	Revision	IP Revision

10.3.1.2 Clock Management Configuration Register (HDMI_WP_SYSCONFIG)

The Clock management configuration register is shown in Figure 24-23 and described in Table 10-20.

Figure 10-11. Clock Management Configuration Register (HDMI_WP_SYSCONFIG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-20. Clock Management Configuration Register (HDMI_WP_SYSCONFIG) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3-2	IDLEMODE		Configuration of the local target state management mode. By definition, target can handle read/write transaction as long as it is out of IDLE state. Default = 2h.
		0	Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally, that is regardless of the IP module's internal requirements. Backup mode, for debug only.
		1h	No-idle mode: local target never enters idle state. Used in case Audio is transferred (to avoid DSS_L3_ICLK clock to be shut down)
		2h	Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module should not generate (IRQ- or DMA-request-related) wakeup events. To be used, if only Video is transmitted (since the DSS_L3_ICLK clock is not required)
		3h	Smart-idle wakeup-capable mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module may generate (IRQ- or DMA-request-related) wakeup events when in idle state.
1	Reserved	0	Reserved
0	SOFTRESET		Software reset. (Optional)
		R0	Software reset done, no pending action
		W0	No action
		R1	Software reset ongoing
		W1	Initiate software reset

10.3.1.3 Raw Interrupt Status Register (HDMI_WP_IRQSTATUS_RAW)

The raw interrupt status register is shown in Figure 10-12 and described in Table 10-21.

Figure 10-12. Raw Interrupt Status Register (HDMI_WP_IRQSTATUS_RAW)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved					AUDIO_FIFO_SAMPLE_REQ_INTR	AUDIO_FIFO_OVERFLOW_INTR	AUDIO_FIFO_UNDERFLOW_INTR
R-0h					R/W1S-0h	R/W1S-0h	R/W1S-0h

7	6	5	4	3	2	1	0
Reserved			OCP_TIME_OUT_INTR	Reserved			CORE_INTR
R-0h			R/W1S-0h	R-0h			R/W1S-0h

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set, write 0 has no effect; -*n* = value after reset

Table 10-21. Raw Interrupt Status Register (HDMI_WP_IRQSTATUS_RAW) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	AUDIO_FIFO_SAMPLE_REQ_INTR		Settable raw status for audio events
		R0	No event pending
		W0	No action
		R1	IRQ event pending
		W1	Set event
9	AUDIO_FIFO_OVERFLOW_INTR		Settable raw status for audio events
		R0	No event pending
		W0	No action
		R1	IRQ event pending
		W1	Set event
8	AUDIO_FIFO_UNDERFLOW_INTR		Settable raw status for audio events
		R0	No event pending
		W0	No action
		R1	IRQ event pending
		W1	Set event
7-5	Reserved	0	Reserved
4	OCP_TIME_OUT_INTR		Settable raw status for OCP timeout interrupt
		R0	No event pending
		W0	No action
		R1	IRQ event pending
		W1	Set event
3-1	Reserved	0	Reserved
0	CORE_INTR		Settable raw status for HDMI Core interrupt
		R0	Software reset done, no pending action
		W0	No action
		R1	Software reset ongoing
		W1	Set event

10.3.1.4 Interrupt Status Register (HDMI_WP_IRQSTATUS)

The interrupt status register is shown in Figure 10-13 and described in Table 10-22.

Figure 10-13. Interrupt Status Register (HDMI_WP_IRQSTATUS)

31	16
Reserved	
R-0h	

15	11	10	9	8
Reserved		AUDIO_FIFO_SAMPLE_REQ_ INTR	AUDIO_FIFO_OVERFLOW_INTR	AUDIO_FIFO_UNDERFLOW_INTR
R-0h		R/W1C-0h	R/W1C-0h	R/W1C-0h

7	5	4	3	1	0
Reserved		OCP_TIME_OUT_INTR	Reserved		CORE_INTR
R-0h		R/W1C-0h	R-0h		R/W1C-0h

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear, write 0 has no effect; *-n* = value after reset

Table 10-22. Interrupt Status Register (HDMI_WP_IRQSTATUS) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	AUDIO_FIFO_SAMPLE_REQ_INTR	R0	No event pending
		W0	No action
		R1	IRQ event pending
		W1	Set event
9	AUDIO_FIFO_OVERFLOW_INTR	R0	No event pending
		W0	No action
		R1	Clear pending event, if any
		W1	Set event
8	AUDIO_FIFO_UNDERFLOW_INTR	R0	No event pending
		W0	No action
		R1	Clear pending event, if any
		W1	Set event
7-5	Reserved	0	Reserved
4	OCP_TIME_OUT_INTR	R0	No event pending
		W0	No action
		R1	Clear pending event, if any
		W1	Set event
3-1	Reserved	0	Reserved
0	CORE_INTR	R0	Software reset done, no pending action
		W0	No action
		R1	Clear pending event, if any
		W1	Set event

10.3.1.5 Interrupt Enable Register (HDMI_WP_IRQENABLE_SET)

The interrupt enable register is shown in Figure 10-14 and described in Table 10-23.

Figure 10-14. Interrupt Enable Register (HDMI_WP_IRQENABLE_SET)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved					ENABLE_SET_AUDIO_FIFO_SAMPLE_REQ_INTR	ENABLE_SET_AUDIO_FIFO_OVERFLOW_INTR	ENABLE_SET_AUDIO_FIFO_UNDERFLOW_INTR
R-0h					R/W1S-0h	R/W1S-0h	R/W1S-0h

7	6	5	4	3	2	1	0
Reserved			ENABLE_SET_OCP_TIME_OUT_INTR	Reserved			ENABLE_SET_CORE_INTR
R-0h			R/W1S-0h	R-0h			R/W1S-0h

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set, write 0 has no effect; -*n* = value after reset

Table 10-23. Interrupt Enable Register (HDMI_WP_IRQENABLE_SET) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	ENABLE_SET_AUDIO_FIFO_SAMPLE_REQ_INTR		Enable for audio interrupt events for sample request. Generated only in IRQ mode (instead of DMA mode default value)
		R0	Interrupt disabled
		W0	No action
		R1	Interrupt enabled
		W1	Enable interrupt
9	ENABLE_SET_AUDIO_FIFO_OVERFLOW_INTR		Enable for audio interrupt events for FIFO overflow
		R0	Interrupt disabled
		W0	No action
		R1	Interrupt enabled
		W1	Enable interrupt
8	ENABLE_SET_AUDIO_FIFO_UNDERFLOW_INTR		Enable for audio interrupt events for FIFO underflow
		R0	Interrupt disabled
		W0	No action
		R1	Interrupt enabled
		W1	Enable interrupt
7-5	Reserved	0	Reserved
4	ENABLE_SET_OCP_TIME_OUT_INTR		Enable for interrupt events for OCP timeout interrupt
		R0	Interrupt disabled
		W0	No action
		R1	Interrupt enabled
		W1	Enable interrupt
3-1	Reserved	0	Reserved
0	ENABLE_SET_CORE_INTR		Enable for audio interrupt events for core interrupt
		R0	Interrupt disabled
		W0	No action
		R1	Interrupt enabled
		W1	Enable interrupt

10.3.1.6 Interrupt Disable Register (HDMI_WP_IRQENABLE_CLEAR)

The interrupt disable register is shown in Figure 10-15 and described in Table 10-24.

Figure 10-15. Interrupt Disable Register (HDMI_WP_IRQENABLE_CLEAR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved					ENABLE_CLEAR_AUDIO_FIFO_SAMPLE_REQ_INTR	ENABLE_CLEAR_AUDIO_FIFO_OVERFLOW_INTR	ENABLE_CLEAR_AUDIO_FIFO_UNDERFLOW_INTR
R-0h					R/W1C-0h	R/W1C-0h	R/W1C-0h

7	6	5	4	3	2	1	0
Reserved			ENABLE_CLEAR_OCP_TIME_OUT_INTR	Reserved			ENABLE_CLEAR_CORE_INTR
R-0h			R/W1C-0h	R-0h			R/W1C-0h

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear, write 0 has no effect; *-n* = value after reset

Table 10-24. Interrupt Disable Register (HDMI_WP_IRQENABLE_CLEAR) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	ENABLE_CLEAR_AUDIO_FIFO_SAMPLE_REQ_INTR	R0	Interrupt disabled
		W0	No action
		R1	Interrupt enabled
		W1	Enable interrupt
9	ENABLE_CLEAR_AUDIO_FIFO_OVERFLOW_INTR	R0	Interrupt disabled
		W0	No action
		R1	Interrupt enabled
		W1	Enable interrupt
8	ENABLE_CLEAR_AUDIO_FIFO_UNDERFLOW_INTR	R0	Interrupt disabled
		W0	No action
		R1	Interrupt enabled
		W1	Enable interrupt
7-5	Reserved	0	Reserved
4	ENABLE_CLEAR_OCP_TIME_OUT_INTR	R0	Interrupt disabled
		W0	No action
		R1	Interrupt enabled
		W1	Enable interrupt
3-1	Reserved	0	Reserved
0	ENABLE_CLEAR_CORE_INTR	R0	Interrupt disabled
		W0	No action
		R1	Interrupt enabled
		W1	Enable interrupt

10.3.1.7 Glitch Filter Register (HDMI_WP_DEBOUNCE)

The glitch filter register is shown in Figure 10-16 and described in Table 10-25.

Figure 10-16. Glitch Filter Register (HDMI_WP_DEBOUNCE)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		RXDET						Reserved		LINE5VSHORT					
R-0h		R/W-14h						R-0h		R/W-14h					

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-25. Glitch Filter Register (HDMI_WP_DEBOUNCE) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13-8	RXDET		Glitch Filter for RXDET Input.
		0	Disabled
		1-3Fh	From 1–63, size of the glitch filtered based on OCP clock frequency
7-6	Reserved	0	Reserved
5-0	LINE5VSHORT		Glitch filter for line 5v short input.
		0	Disabled
		1-3Fh	From 1–63, size of the glitch filtered based on OCP clock frequency

10.3.1.8 Configuration of HDMI Wrapper Video Register (HDMI_WP_VIDEO_CFG)

The configuration of HDMI wrapper video register is shown in Figure 10-17 and described in Table 10-26.

Figure 10-17. Configuration of HDMI Wrapper Video Register (HDMI_WP_VIDEO_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved					PACKING_MODE		
R-20h					R/W-0h		

7	6	5	4	3	2	1	0
Reserved		CORE_VSYNC_INV	CORE_HSYNC_INV	Reserved		MODE	
R-0h		R/W-0h	R/W-0h	R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

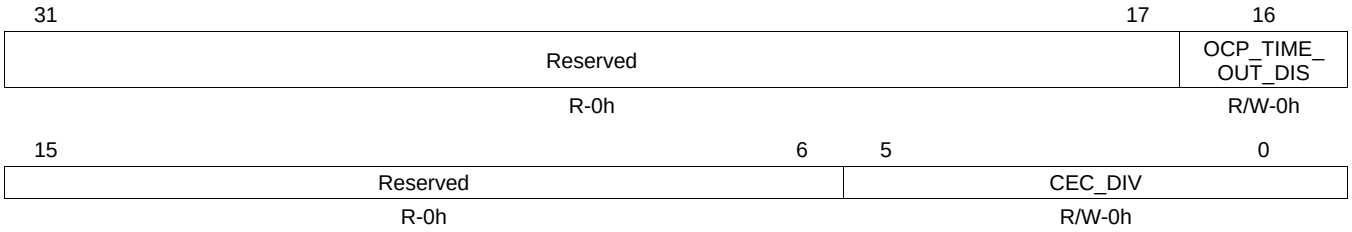
Table 10-26. Configuration of HDMI Wrapper Video Register (HDMI_WP_VIDEO_CFG) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	20h	Reserved
10-8	PACKING_MODE	0	Used to pack 10-bits RGB or YUV444 input data in case the video input is not already packed. Will pack video_data[35:26] - video_data[23:14] - video_data[11:2]
		1h	Used to pack 24-bits RGB, 24-bit YUV444, 16-bit YUV422 or 24-bit YUV422 input data in case the video input is not already packed. Will pack video_data[35:28] - video_data[23:16] - video_data[11:4]
		2h	Used to pack 20-bits YUV422 input data in case the video input is not already packed. Will pack video_data[35:28] - video_data[23:16] - video_data[11:10] - video_data[7:6]
		3h-6h	Reserved
		7h	No change on input video_data lines. The video_data[35:0] is provided to the core without any change on it.
7-6	Reserved	0	Reserved
5	CORE_VSYNC_INV		Enables to invert or not the VSYNC signal provided to the HDMI core
		0	VSYNC is unchanged
		1	VSYNC is inverted
4	CORE_HSYNC_INV		Enables to invert or not the HSYNC signal provided to the HDMI core
		0	HSYNC is unchanged
		1	HSYNC is inverted
3-2	Reserved	0	Reserved
1-0	MODE	0	Always program to 0

10.3.1.9 Configuration of Clocks Register (HDMI_WP_CLK)

The configuration of clocks register is shown in Figure 10-18 and described in Table 10-27.

Figure 10-18. Configuration of Clocks Register (HDMI_WP_CLK)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-27. Configuration of Clocks Register (HDMI_WP_CLK) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	OCP_TIME_OUT_DIS		Timeout in case CEC_DDC_CLK not provided.
		0	Timeout after 4095 OCP clock cycles after inactivity. HDMI Core register interface (due to CEC_DDC_CLK not provided to HDMI). An interrupt is generated. No error response is provided on the OCP interface.
		1	No timeout capability
15-6	Reserved	0	Reserved
5-0	CEC_DIV		Defines the divisor value to be used for the generation of the CEC clock (1 MHz) from the input CEC_DDC clock (48 MHz). If 48 MHz is provided, the division by 24 is required (18h) to get the expected CEC clock speed (2 MHz) The valid values are from 0 to 63.
		0	Gated
		1	Free-running

10.3.1.10 Audio Configuration in FIFO Register (HDMI_WP_AUDIO_CFG)

The audio configuration in FIFO register is shown in Figure 10-19 and described in Table 10-28.

Figure 10-19. Audio Configuration in FIFO Register (HDMI_WP_AUDIO_CFG)

31	27	26	24
Reserved		STEREO_CHANNEL_ENABLE	
R-0h		R/W-1h	

23	16
AUDIO_CHANNEL_LOCATION	
R/W-3h	

15	8
Reserved	
R-0h	

7	6	5	4	3	2	1	0
Reserved		BLOCK_START_END_DISABLE	IEC	JUSTIFY	LEFT_BEFORE	SAMPLE_NBR	SAMPLE_SIZE
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-1h	R/W-0h	R/W-1h

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-28. Audio Configuration in FIFO Register (HDMI_WP_AUDIO_CFG) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-24	STEREO_CHANNEL_ENABLE		This field contains the number of stereo channels enabled in the HDMI_CORE module. Must be set by the software and aligned with the register set in the HDMI_core.
		0	No stereo channel enabled in the HDMI_core module
		1h	1 stereo channel enabled in the HDMI_core module
		2h	2 stereo channels enabled in the HDMI_core module
		3h	3 stereo channels enabled in the HDMI_core module
		4h	4 stereo channels enabled in the HDMI_core module
		5h-7h	Reserved
23-16	AUDIO_CHANNEL_LOCATION	3h	This field contains which channels are active. It is used also to define which registers are stuffed with zero.
15-6	Reserved	0	Reserved
5	BLOCK_START_END_DISABLE		This field disables or not the block end start generation. This field does not affect the pcm format.
		0	Block signals are generated.
		1	Block signals are not generated; Block start is set to high and block end to low
4	IEC		Indicate if the format of the FIFO is compliant with the IEC format.
		0	Audio format is L-PCM; Format not aligned with IEC
		1	Audio format is IEC 60 958/61937; Format is aligned with IEC format
3	JUSTIFY		This field shows the justification (not applicable if IEC format)
		0	Justify left
		1	Justify right
2	LEFT_BEFORE		DO NOT USE—Software always uses LEFT_BEFORE = 1
		0	The first sample is the right
		1	The first sample is the left
1	SAMPLE_NBR		This field indicates the number of sample per word (32 bits) (not applicable if IEC format)
		0	1 sample per word, 32 bit
		1	2 samples per word, 32 bits (only compatible with sample on 16 bits)

Table 10-28. Audio Configuration in FIFO Register (HDMI_WP_AUDIO_CFG) Field Descriptions (continued)

Bit	Field	Value	Description
0	SAMPLE_SIZE		Audio sample size, 16 bits or 24 bits (not applicable if IEC format)
		0	Sample is on 16 bits
		1	Sample is on 24 bits

10.3.1.11 Audio Configuration of DMA Register (HDMI_WP_AUDIO_CFG2)

The audio configuration of DMA register is shown in Figure 10-20 and described in Table 10-29.

Figure 10-20. Audio Configuration of DMA Register (HDMI_WP_AUDIO_CFG2)

31	16
Reserved	
R-0h	

15	8	7	0
DMA_TRANSFER		BLOCK_SIZE	
R/W-10h		R/W-C0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-29. Audio Configuration of DMA Register (HDMI_WP_AUDIO_CFG2) Field Descriptions

Bit	Field	Description
31-16	Reserved	Reserved
15-8	DMA_TRANSFER	Use to control the dma request. When the number of OCP dma access is performed, the DMA request signal is de-asserted (set to low) All dma access are performed on 32 bit in HDMI_WP_AUDIO_DATA register Encoded value (from 0 to 255). The value 0 is invalid.
7-0	BLOCK_SIZE	Define the block size if audio sample are compressed default value is 192 to match the IEC 60958 standard.

10.3.1.12 Audio FIFO Control Register (HDMI_WP_AUDIO_CTRL)

The audio FIFO control register is shown in Figure 10-21 and described in Table 10-30.

Figure 10-21. Audio FIFO Control Register (HDMI_WP_AUDIO_CTRL)

31	30	29 26	25 24
WRAPPER_ENABLE	CORE_REQ_ENABLE	Reserved	NUMBER_OF_SAMPLE
R/W-0h	R/W-0h	R-0h	R-0h

23 16
NUMBER_OF_SAMPLE
R-0h

15 10	9	8
Reserved	DMA_OR_IRQ	THRESHOLD_VALUE
R-0h	R/W-0h	R/W-0h

7 0
THRESHOLD_VALUE
R/W-10h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-30. Audio FIFO Control Register (HDMI_WP_AUDIO_CTRL) Field Descriptions

Bit	Field	Value	Description
31	WRAPPER_ENABLE		This filed enable the audio wrapper. When disable the audio path is under reset Register setting are not affected by the enable.
		0	Wrapper is disabled
		1	Wrapper is enabled
30	CORE_REQ_ENABLE	0	Enables the Audio data request generated by the core. This is used to start the audio data transmission from the wrapper to the core. This must be enabled only after the 5th VSYNC is generated to ensure that the Audio and Video are in Sync.
29-26	Reserved	0	Reserved
25-16	NUMBER_OF_SAMPLE	0	Shows the number of valid sample (16 or 24 bits) in the FIFO (depends of the FIFO format setting)
15-10	Reserved	0	Reserved
9	DMA_OR_IRQ		Indicated if the threshold generates a DMA or an IRQ
		0	When the threshold is reached a DMA req will be generated
		1	When the threshold is reached an IRQ will be generated
8-0	THRESHOLD_VALUE	10h	The DMA/IRQ is generated if the number of sample (24 or 16 bit) in the FIFO is less or equal to the threshold value. The maximum threshold is 511 and the minimum is 0.

10.3.1.13 TX Data of FIFO Register (HDMI_WP_AUDIO_DATA)

The TX data of FIFO register is shown in Figure 10-22 and described in Table 10-31.

Figure 10-22. TX Data of FIFO Register (HDMI_WP_AUDIO_DATA)

31	0
FIFO_DATA	
W-0h	

LEGEND: W = Write only; *-n* = value after reset

Table 10-31. TX Data of FIFO Register (HDMI_WP_AUDIO_DATA) Field Descriptions

Bit	Field	Description
31-0	FIFO_DATA	Audio TX DATA. Can be accessed in 32-bit mode only. Read returns 0.

10.3.2 HDMI Core System Registers

Table 10-32 lists the HDMI core system registers.

Table 10-32. HDMI Core System Registers

Address Offset	Acronym	Register Name
00h	VND_IDL	Vendor ID Register
04h	VND_IDH	Vendor ID Register
08h	DEV_IDL	Device ID Register
0Ch	DEV_IDH	Device ID Register
10h	DEV_REV	Device Revision Register
14h	SRST	Software Reset Register
20h	SYS_CTRL1	System Control Register 1
24h	SYS_STAT	System Status Register
28h	SYS_CTRL3	Legacy Registers
34h	DCTL	Data Control Register
3Ch	HDCP_CTRL	HDCP Control Register
40h-50h	BKSV__0 - BKSV__4	HDCP BKSV Register
54h-70h	AN__0 - AN__7	HDCP AN Register
74h-84h	AKSV__0 - AKSV__4	HDCP AKSV Register
88h	RI1	HDCP Ri Register
8Ch	RI2	HDCP Ri Register
90h	RI_128_COMP	HDCP Ri 128 Compare Register
94h	I_CNT	HDCP I Counter Register
98h	RI_STAT	Ri Status Register
9Ch	RI_CMD	Ri Command Register
A0h	RI_START	Ri Line Start Register
A4h	RI_RX_L	Ri From RX Registers
A8h	RI_RX_H	Ri From RX Registers
ACh	RI_DEBUG	Ri Debug Registers
C8h	DE_DLY	VIDEO DE Delay Register
C8h	DE_DLY	VIDEO DE Delay Register
CCh	DE_CTRL	VIDEO DE Control Register
D0h	DE_TOP	VIDEO DE Top Register
D8h	DE_CNTL	VIDEO DE Count Register
DCh	DE_CNTH	VIDEO DE Count Register
E0h	DE_LINL	VIDEO DE Line Register
E4h	DE_LINH_1	VIDEO DE Line Register
E8h	HRES_L	Video H Resolution Register
ECh	HRES_H	Video H Resolution Register
F0h	VRES_L	Video V Resolution Register
F4h	VRES_H	Video V Resolution Register
F8h	IADJUST	Video Interlace Adjustment Register
FCh	POL_DETECT	Video SYNC Polarity Detection Register
100h	HBIT_2HSYNC1	Video Hbit to HSYNC Register
104h	HBIT_2HSYNC2	Video Hbit to HSYNC Register
108h	FLD2_HS_OFSTL	Video Field2 HSYNC Offset Register
10Ch	FLD2_HS_OFSTH	Video Field2 HSYNC Offset Register
110h	HWIDTH1	Video HSYNC Length Register
114h	HWIDTH2	Video HSYNC Length Register
118h	VBIT_TO_VSYNC	Video Vbit to VSYNC Register

Table 10-32. HDMI Core System Registers (continued)

Address Offset	Acronym	Register Name
11Ch	VWIDTH	Video VSYNC Length Register
120h	VID_CTRL	Video Control Register
124h	VID_ACEN	Video Action Enable Register
128h	VID_MODE	Video Mode1 Register
12Ch	VID_BLANK1	Video Blanking Registers
130h	VID_BLANK2	Video Blanking Registers
134h	VID_BLANK3	Video Blanking Registers
138h	DC_HEADER	Deep Color Header Register
13Ch	VID_DITHER	Video Mode2 Register
140h	RGB2XVYCC_CT	RGB_2_xvYCC control Register
144h	R2Y_COEFF_LOW	RGB_2_xvYCC Conversion R_2_Y Register
148h	R2Y_COEFF_UP	RGB_2_xvYCC Conversion R_2_Y Register
14Ch	G2Y_COEFF_LOW	RGB_2_xvYCC Conversion G_2_Y Register
150h	G2Y_COEFF_UP	RGB_2_xvYCC Conversion G_2_Y Register
154h	B2Y_COEFF_LOW	RGB_2_xvYCC Conversion B_2_Y Register
158h	B2Y_COEFF_UP	RGB_2_xvYCC Conversion B_2_Y Register
15Ch	R2CB_COEFF_LOW	RGB_2_xvYCC Conversion R_2_Cb Register
160h	R2CB_COEFF_UP	RGB_2_xvYCC Conversion R_2_Cb Register
164h	G2CB_COEFF_LOW	RGB_2_xvYCC Conversion G_2_Cb Register
168h	G2CB_COEFF_UP	RGB_2_xvYCC Conversion G_2_Cb Register
16Ch	B2CB_COEFF_LOW	RGB_2_xvYCC Conversion B_2_Cb Register
170h	B2CB_COEFF_UP	RGB_2_xvYCC Conversion B_2_Cb Register
174h	R2CR_COEFF_LOW	RGB_2_xvYCC Conversion R_2_Cr Register
178h	R2CR_COEFF_UP	RGB_2_xvYCC Conversion R_2_Cr Register
17Ch	G2CR_COEFF_LOW	RGB_2_xvYCC Conversion G_2_Cr Register
180h	G2CR_COEFF_UP	RGB_2_xvYCC Conversion G_2_Cr Register
184h	B2CR_COEFF_LOW	RGB_2_xvYCC Conversion B_2_Cr Register
188h	B2CR_COEFF_UP	RGB_2_xvYCC Conversion B_2_Cr Register
18Ch	RGB_OFFSET_LOW	RGB_2_xvYCC RGB Input Offset Register
190h	RGB_OFFSET_UP	RGB_2_xvYCC RGB Input Offset Register
194h	Y_OFFSET_LOW	RGB_2_xvYCC Conversion Y Output Offset Register
198h	Y_OFFSET_UP	RGB_2_xvYCC Conversion Y Output Offset Register
19Ch	CBCR_OFFSET_LOW	RGB_2_xvYCC Conversion CbCr Output Offset Register
1A0h	CBCR_OFFSET_UP	RGB_2_xvYCC Conversion CbCr Output Offset Register
1C0h	INTR_STATE	Interrupt State Register
1C4h	INTR1	Interrupt Source Register
1C8h	INTR2	Interrupt Source Register
1CCh	INTR3	Interrupt Source Register
1D0h	INTR4	Interrupt Source Register
1D4h	INT_UNMASK1	Interrupt Unmask Register
1D8h	INT_UNMASK2	Interrupt Unmask Register
1DCh	INT_UNMASK3	Interrupt Unmask Register
1E0h	INT_UNMASK4	Interrupt Unmask Register
1E4h	INT_CTRL	Interrupt Control Register
240h	XVYCC2RGB_CTL	xvYCC_2_RGB Control Register
244h	Y2R_COEFF_LOW	xvYCC_2_RGB Conversion Y_2_R Register
248h	Y2R_COEFF_UP	xvYCC_2_RGB Conversion Y_2_R Register

Table 10-32. HDMI Core System Registers (continued)

Address Offset	Acronym	Register Name
24Ch	CR2R_COEFF_LOW	xvYCC_2_RGB Conversion Cr_2_R Register
250h	CR2R_COEFF_UP	xvYCC_2_RGB Conversion Cr_2_R Register
254h	CB2B_COEFF_LOW	xvYCC_2_RGB Conversion Cb_2_B Register
258h	CB2B_COEFF_UP	xvYCC_2_RGB Conversion Cb_2_B Register
25Ch	CR2G_COEFF_LOW	xvYCC_2_RGB Conversion Cr_2_G Register
260h	CR2G_COEFF_UP	xvYCC_2_RGB Conversion Cr_2_G Register
264h	CB2G_COEFF_LOW	xvYCC_2_RGB Conversion Cb_2_G Register
268h	CB2G_COEFF_UP	xvYCC_2_RGB Conversion Cb_2_G Register
26Ch	YOFFSET1_LOW	xvYCC_2_RGB Conversion Y Offset Register
270h	YOFFSET1_UP	xvYCC_2_RGB Conversion Y Offset Register
274h	OFFSET1_LOW	xvYCC_2_RGB Conversion Offset1 Register
278h	OFFSET1_MID	xvYCC_2_RGB Conversion Offset1 Register
27Ch	OFFSET1_UP	xvYCC_2_RGB Conversion Offset1 Register
280h	OFFSET2_LOW	xvYCC_2_RGB Conversion Offset2 Register
284h	OFFSET2_UP	xvYCC_2_RGB Conversion Offset2 Register
288h	DCLEVEL_LOW	xvYCC_2_RGB Conversion DC Level Register
28Ch	DCLEVEL_UP	xvYCC_2_RGB Conversion DC Level Register
3B0h	DDC_MAN	DDC I2C Manual Register
3B4h	DDC_ADDR	DDC I2C Target Slave Address Register
3B8h	DDC_SEGM	DDC I2C Target Segment Address Register
3BCh	DDC_OFFSET	DDC I2C Target Offset Address Register
3C0h	DDC_COUNT1	DDC I2C Data Count Register
3C4h	DDC_COUNT2	DDC I2C Data Count Register
3C8h	DDC_STATUS	DDC I2C Status Register
3CCh	DDC_CMD	DDC I2C Command Register
3D0h	DDC_DATA	DDC I2C Data Register
3D4h	DDC_FIFOCNT	DDC I2C FIFO Count Register
3E4h	EPST	ROM Status Register
3E8h	EPCM	ROM Command Register

10.3.2.1 Vendor IDL Register (VND_IDL)

The vendor IDL register is shown in Figure 10-23 and described in Table 10-33.

Figure 10-23. Vendor ID Register (VND_IDL)

31	8	7	0
Reserved		VND_IDL	
R-0h		R-1h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-33. Vendor ID Register (VND_IDL) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	VND_IDL	Vendor ID low byte. Provides unique vendor identification through I2C.

10.3.2.2 Vendor IDH Register (VND_IDH)

The vendor IDH register is shown in Figure 10-24 and described in Table 10-34.

Figure 10-24. Vendor ID Register (VND_IDH)

31	8	7	0
Reserved		VND_IDH	
R-0h		R-0h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-34. Vendor ID Register (VND_IDH) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	VND_IDH	Vendor ID high byte. Provides unique vendor identification through I2C.

10.3.2.3 Device IDL Register (DEV_IDL)

The device IDL register is shown in Figure 10-25 and described in Table 10-35.

Figure 10-25. Device IDL Register (DEV_IDL)

31	8	7	0
Reserved		DEV_IDL	
R-0h		R-36h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-35. Device IDL Register (DEV_IDL) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DEV_ID	Device ID low byte. Provides unique vendor identification through I2C.

10.3.2.4 Device IDH Register (DEV_IDH)

The device IDH register is shown in Figure 10-26 and described in Table 10-36.

Figure 10-26. Device IDH Register (DEV_IDH)

31	8	7	0
Reserved		DEV_IDH	
R-0h		R-97h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-36. Device IDH Register (DEV_IDH) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DEV_ID	Device ID high byte. Provides unique vendor identification through I2C.

10.3.2.5 Device Revision Register (DEV_REV)

The device revision register is shown in Figure 10-27 and desribed in Table 10-37.

Figure 10-27. Device Revision Register (DEV_REV)

31	8	7	0
Reserved		DEV_REV	
R-0h		R-0h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-37. Device Revision Register (DEV_REV) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DEV_REV	Allows distinction between revisions of same device.

10.3.2.6 Software Reset Register (SRST)

The software reset register (SRST) is shown in Figure 10-28 and described in Table 10-38.

Figure 10-28. Software Reset Register (SRST)

31	16
Reserved	
R-0h	

15	2	1	0
Reserved		FIFORST	SWRST
R-0h		R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-38. Software Reset Register (SRST) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	FIFORST		Audio FIFO reset
		0	Normal operation
		1	Reset (flush) audio FIFO
0	SWRST		Software reset
		0	Normal operation
		1	Reset all sections, including audio FIFO but not writable registers or HDCP

10.3.2.7 System Control Register 1 (SYS_CTRL1)

The system control register 1 is shown in Figure 10-29 and described in Table 10-39.

Figure 10-29. System Control Register 1 (SYS_CTRL1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	VSYNC	VEN	HEN	Reserved	BSEL	EDGE	PD
R-0h	R-x	R/W-1h	R/W-1h	R-0h	R/W-1h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; x = value is indeterminate

Table 10-39. System Control Register 1 (SYS_CTRL1) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6	VSYNC	x	The current status of the VSYNC input pin. Refer to the INTR2 register for an interrupt tied to VSYNC active edge.
5	VEN		VSYNC enable
		0	Fixed LOW
		1	Follow VSYNC input
4	HEN		HSYNC enable
		0	Fixed LOW
		1	Follow HSYNC input
3	Reserved	0	Reserved
2	BSEL		Input bus select
		0	12-bit Data Bus
		1	24-bit Data Bus
1	EDGE		Edge select
		0	Latch Input on Falling Edge
		1	Latch Input on Rising Edge
0	PD		Power down mode. Most other register values are not affected by assertion of the PD bit.
		0	Interrupts are in power-down mode
		1	Normal operation

10.3.2.8 System Status Register (SYS_STAT)

The system status register is shown in Figure 10-30 and described in Table 10-40.

Figure 10-30. System Status Register (SYS_STAT)

31	16
Reserved	
R-0h	

15	8
Reserved	
R-0h	

7	6	3	2	1	0
VLOW	Reserved		RSEN	HPD	P_STABLE
R-1h	R-0h		R-x	R-X	R-x

LEGEND: R = Read only; -*n* = value after reset; x = value is indeterminate

Table 10-40. System Status Register (SYS_STAT) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	VLOW	1	VREF mode. Always HIGH.
6-4	Reserved	0	Reserved
2	RSEN		Receiver sense (works in DC-coupled systems only) RSEN is active when the TMDS link is terminated, usually into a powered-on TMDS receiver chip. An active RSEN implies an active Hot Plug Detect; as the link must also be physically connected.
		0	No receiver is connected
		1	Receiver is connected and powered on
1	HPD	x	Hot plug detect. The state of the hot plug detect pin can be read here.
0	P_STABLE	x	IDCK (io_pclkpin) to TMDS clock (v_ck2x) is stable and the Transmitter can send reliable data on the TMDS link. A change to the IDCK sets this bit LOW. After a subsequent LOW to HIGH transition, indicating a stable input clock, a software reset is recommended.

10.3.2.9 System Control Register 3 (SYS_CTRL3)

The system control register 3 is shown in Figure 10-31 and shown in Table 10-41.

Figure 10-31. System Control Register 3 (SYS_CTRL3)

31	16
Reserved	
R-0h	

15	3	2	1	0
Reserved		CTL		Reserved
R-0h		R/W-0h		R-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-41. System Control Register 3 (SYS_CTRL3) Field Descriptions

Bit	Field	Description
31-3	Reserved	Reserved
2-1	CTL	The states of these control bits are transmitted across the TMDS link during blanking times for DVI 1.0 mode only.
0	Reserved	Reserved

10.3.2.10 Data Control Register (DCTL)

The data control register is shown in Figure 10-32 and described in Table 10-42.

Figure 10-32. Data Control Register (DCTL)

31	16
Reserved	
R-0h	

15	3	2	1	0
Reserved		VID_BLANK	AUD_MUTE	HDCP_SEL
R-0h		R/W-0h	R/W-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-42. Data Control Register (DCTL) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	VID_BLANK	0	Normal operation (video output is not blanked)
		1	Video output is blanked and the colors sent are those specified in registers VID_BLANK1, VID_BLANK2, and VID_BLANK3.
1	AUD_MUTE	0	Do not send zeroes in audio pocket
		1	Send zeroes in audio pocket
0	HDCP_SEL	0	This register gets the value of the pin io_hdcp_sel. When connected to 1 b0; enables firmware to take the decision whether to send unencrypted data or not. By connecting to 1 b1; HDMI Tx will be able to send ONLY encrypted data.

10.3.2.11 HDCP Control Register (HDCP_CTRL)

The HDCP control register is shown in Figure 10-33 and described in Table 10-43.

Figure 10-33. HDCP Control Register (HDCP_CTRL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	ENC_ON	BKSV_ERR	RX_RPTR	TX_ANSTOP	CP_RESTN	RI_RDY	ENC_EN
R-0h	R-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-43. HDCP Control Register (HDCP_CTRL) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6	ENC_ON		Encryption status
		0	Encryption disabled or not in progress
		1	Encryption enabled and in progress
5	BKSV_ERR		BKSV error. To clear BKSV_ERR; the firmware must first set the TX_ANSTOP bit; then perform an authentication twice with a valid BKSV value.
		0	No error in BKSV format
		1	Error in BKSV format
4	RX_RPTR		Repeater. If the HDMI Receiver is in a Repeater, write bit 4 (RX_RPTR) to 1 before beginning the authentication process. Note: This bit is written when the Source device detects an attached HDCP Repeater device. This is a necessary step in the computation of shared values in the HDCP protocol. For more information, sefer section 2.2 in the HDCP 1.0 Specification.
		0	Single HDMI Receiver
		1	HDMI Receiver is a Repeater
3	TX_ANSTOP	0	AN control. When cleared; the cipher engine is allowed to free run and the AN registers cycle through pseudo-random values. When set; the cipher engines stops and the AN register may be read and initialized in the HDCP-capable Receiver. To clear this bit; toggle the RX_RPTR bit. This bit is also cleared when the hardware is reset or BKSV_ERR is set.
2	CP_RESTN		Content protection reset
		0	Reset
		1	Normal operation
1	RI_RDY		Ri Ready
		0	Ri first value is not ready
		1	Ri first value is ready in the HDMI Transmitter. Cleared by a hardware reset. This bit is also cleared when the first byte of BKSV is written into the HDMI Transmitter (performed at the beginning of the next authentication process).
0	ENC_EN		Encryption enable
		0	Encryption is disabled
		1	Encryption is enabled. See description of HDCP_SEL bit in register DCTL (see Section 10.3.2.10).

10.3.2.12 HDCP BKSV Register (BKSV__0-BKSV__4)

The HDCP BKSV register is shown in Figure 10-34 and described in Table 10-44.

Figure 10-34. HDCP BKSV Register (BKSV__0-BKSV__4)

31	8	7	0
Reserved		BKSV	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-44. HDCP BKSV Register (BKSV__0-BKSV__4) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	BKSV	Write : Written with the HDCP receiver key selection vector register value. Writing 5th BKSV byte triggers the authentication logic in the HDMI Transmitter. Write this byte last.

10.3.2.13 HDCP AN Register (AN__0-AN__7)

The HDCP AN register is shown in Figure 10-35 and described in Table 10-45.

Figure 10-35. HDCP AN Register (AN__0-AN__7)

31	8	7	0
Reserved		AN	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-45. HDCP AN Register (AN__0-AN__7) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AN	AN value is an HDCP 64-Bit pseudo-random value.

10.3.2.14 HDCP AKSV Register (AKSV__0-AKSV__4)

The HDCP AKSV registe is shown in Figure 10-36 and described in Table 10-46.

Figure 10-36. HDCP AKSV Register (AKSV__0-AKSV__4)

31	8	7	0
Reserved		AKSV	
R-0h		R-x	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; x = value is indeterminate

Table 10-46. HDCP AKSV Register (AKSV__0-AKSV__4) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AKSV	HDCP-capable transmitter s key selection vector. Fifth AKSV byte triggers the authentication logic in the receiver. Write this byte last.

10.3.2.15 HDCP Ri1 Register (RI1)

The HDCP Ri1 register is shown in Figure 10-37 and described in Table 10-47.

Figure 10-37. HDCP Ri1 Register (RI1)

31	8	7	0
Reserved		RI	
R-0h		R-0h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-47. HDCP Ri Register (RI1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	RI1	R1i register. The value of this register should be read and compared with the HDMI Receiver s Ri value.

10.3.2.16 HDCP Ri2 Register (RI2)

The HDCP Ri2 register is shown in Figure 10-38 and described in Table 10-48.

Figure 10-38. HDCP Ri2 Register (RI2)

31	8	7	0
Reserved		RI2	
R-0h		R-0h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-48. HDCP Ri2 Register (RI2) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	RI2	Ri2 register. The value of this register should be read and compared with the HDMI Receiver s Ri value.

10.3.2.17 HDCP Ri 128 Compare Register (RI_128_COMP)

The HDCP Ri 128 compare register is shown in Figure 10-39 and described in Table 10-49.

Figure 10-39. HDCP Ri 128 Compare Register (RI_128_COMP)

31	7	6	0
Reserved		RI_128_COMP	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-49. HDCP Ri 128 Compare Register (RI_128_COMP) Field Descriptions

Bit	Field	Description
31-7	Reserved	Reserved
6-0	RI_128_COMP	Limit counter for Ri comparison. When the frame counter (I_CNT) reaches the index set in this register an RI_128 interrupt is generated.

10.3.2.18 HDCP I Counter Register (I_CNT)

The HDCP I counter register is shown in Figure 10-40 and described in Table 10-50.

Figure 10-40. HDCP I Counter Register (I_CNT)

31	7	6	0
Reserved		I_CNT	
R-0h		R-0h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-50. HDCP I Counter Register (I_CNT) Field Descriptions

Bit	Field	Description
31-7	Reserved	Reserved
6-0	I_CNT	Current value of I counter. This register counts frames from 0 to 127, then rolls over to zero.

10.3.2.19 Ri Status Register (RI_STAT)

The Ri status register is shown in Figure 10-41 and described in Table 10-51.

Figure 10-41. Ri Status Register (RI_STAT)

31	1	0
Reserved		RI_STARTED
R-0h		R-0h

LEGEND: R = Read only; -*n* = value after reset

Table 10-51. Ri Status Register (RI_STAT) Field Descriptions

Bit	Field	Description
31-1	Reserved	Reserved
0	RI_STARTED	Ri check started status. This signal is used for handshaking between the firmware and the hardware. After the Ri check is enabled, the hardware waits for the DDC master to finish the current transaction before taking control. After this bit is set, the firmware loses the ability to use the DDC Master, unless it disables Ri Check and resets to 0.

10.3.2.20 Ri Command Register (RI_CMD)

The Ri command register is shown in Figure 10-42 and described in Table 10-52.

Figure 10-42. Ri Command Register (RI_CMD)

31	16
Reserved	
R-0h	

15	2	1	0
Reserved		BCAP_EN	RI_EN
R-0h		R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-52. Ri Command Register (RI_CMD) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	BCAP_EN		Enable polling of the BCAP_DONE bit (in the register INTR2). Note: To poll the BCAP_DONE bit; the ENC_EN (in HDPC_CTRL register) bit and the Ri_EN (in RI_CMD register) bit must be enabled on the HDMI Transmitter.
		0	Disable
		1	Enable
0	RI_EN		Enable Ri check. Check bit RI_STARTED of the Ri_STAT register for firmware and hardware DDC control handshaking.
		0	Disable
		1	Enable

10.3.2.21 Ri Line Start Register (RI_START)

The Ri line start register is shown in Figure 10-43 and described in Table 10-53.

Figure 10-43. Ri Line Start Register (RI_START)

31	8	7	0
Reserved		RI_LINE_START	
R-0h		R/W-4h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-53. Ri Line Start Register (RI_START) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	RI_LINE_START	Indicates at what line within frame 127 or 0 to start the Ri Check. Note that the value for this register bit represents the power of 2; the 2 LSBs are 0.

10.3.2.22 Ri From RX Registers (Low) (RI_RX_L)

The Ri from RX registers (low) are shown in Figure 10-44 and described in Table 10-54.

Figure 10-44. Ri From RX Registers (Low) (RI_RX_L)

31	8	7	0
Reserved		RI_RX	
R-0h		R-0h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-54. Ri From RX Register (Low) (RI_RX_L) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	RI_RX	Ri_RX[15:8] and Ri_RX[7:0]. This value represents the HDMI Receiver s Ri value if any of the Ri Check errors occurred.

10.3.2.23 Ri From RX Registers (High) (RI_RX_H)

The Ri from RX registers (high) are shown in Figure 10-45 and described in Table 10-55.

Figure 10-45. Ri From RX Registers (High) (RI_RX_H)

31	8	7	0
Reserved		RI_RX	
R-0h		R-0h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-55. Ri From RX Registers (High) (RI_RX_H) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	RI_RX	Ri_RX[15:8] and Ri_RX[7:0]. This value represents the HDMI Receiver s Ri value if any of the Ri Check errors occurred.

10.3.2.24 Ri Debug Registers (RI_DEBUG)

The Ri debug registers are shown in Figure 10-46 and described in Table 10-56.

Figure 10-46. Ri Debug Registers (RI_DEBUG)

31	16
Reserved	
R-0h	

15	8
Reserved	
R-0h	

7	6	5	0
RI_DBG_TRASH	RI_DBG_HOLD	Reserved	
R/W-0h	R/W-0h	R-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-56. Ri Debug Registers (RI_DEBUG) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RI_DBG_TRASH		Ri value trash
		0	Continue with regular updates to Ri
		1	Force a corruption of the Ri values
6	RI_DBG_HOLD		Ri value hold
		0	Continue with regular updates to Ri
		1	Hold the Ri value steady; stop updating
5-0	Reserved	0	Reserved

10.3.2.25 VIDEO DE Delay Register (DE_DLY)

The VIDEO DE delay register is shown in Figure 10-47 and described in Table 10-57.

Figure 10-47. VIDEO DE Delay Register (DE_DLY)

31	8	7	0
Reserved		DE_DLY[7:0]	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-57. VIDEO DE Delay Register (DE_DLY) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DE_DLY	Width of the area to the left of the active display. The unit of measure is pixels. This register should be set to the sum of (HSYNC width) + (horizontal back porch) + (horizontal left border), and is used only for DE generation. Note that this 12-bit value includes four bits from register DE_CTRL (see Section 10.3.2.26). The valid range is 1-4095. 0 is invalid.

10.3.2.26 VIDEO DE Control Register (DE_CTRL)

The VIDEO DE control register is shown in Figure 10-48 and described in Table 10-58.

Figure 10-48. VIDEO DE Control Register (DE_CTRL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	DE_GEN	VS_POL	HS_POL	DE_DLY[11:8]			
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-58. VIDEO DE Control Register (DE_CTRL) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6	DE_GEN		Generate DE signal
		0	Disabled
		1	Enabled
5	VS_POL		VSYNC polarity
		0	Positive polarity (leading edge rises)
		1	Negative polarity (leading edge falls)
			Set this bit to the input VSYNC polarity for the source that provides VSYNC. For embedded syncs, set this bit to the desired VSYNC polarity that is generated from the embedded sync codes.
4	HS_POL		HSYNC polarity
		0	Positive polarity (leading edge rises)
		1	Negative polarity (leading edge falls)
			Set this bit to the input HSYNC polarity for the source that provides HSYNC. For embedded syncs, set this bit to the desired HSYNC polarity that is generated from the embedded sync codes.
3-0	DE_DLY	0	Width of the area to the left of the active display. The unit of measure is pixels. This register should be set to the sum of (HSYNC width) + (horizontal back porch) + (horizontal left border), and is used only for DE generation.
			Note that this 12-bit value includes eight bits from register DE_DLY (see Section 10.3.2.25).

10.3.2.27 VIDEO DE Top Register (DE_TOP)

The VIDEO DE top register is shown in Figure 10-49 and described in Table 10-59.

Figure 10-49. VIDEO DE Top Register (DE_TOP)

31	8	7	0
Reserved		DE_TOP	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-59. VIDEO DE Top Register (DE_TOP) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DE_TOP	Defines the height of the area above the active display. The unit of measure is lines (HSYNC pulses). This register should be set to the sum of (VSYNC width) + (vertical back porch) + (vertical top border). The valid range is 1-127. 0 is invalid.

10.3.2.28 VIDEO DE Count Register (DE_CNTL)

Figure 10-50. VIDEO DE Count Register (DE_CNTL)

31	8	7	0
Reserved		DE_CNT[7:0]	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

The VIDEO DE count register is shown in Figure 10-50 and described in Table 10-60.

Table 10-60. VIDEO DE Count Register (DE_CNTL) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DE_CNT	Defines the width of the active display. The unit of measure is pixels. This register should be set to the desired horizontal resolution. Note that this 12-bit value includes four bits from register DE_CNTH (see Section 10.3.2.29). The valid range is 1-4095. 0 is invalid.

10.3.2.29 VIDEO DE Count Register (DE_CNTH)

The VIDEO DE count register is shown in Figure 10-51 and described in Table 10-61.

Figure 10-51. VIDEO DE Count Register (DE_CNTH)

31	4	3	0
Reserved		DE_CNT[11:8]	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-61. VIDEO DE Count Register (DE_CNTH) Field Descriptions

Bit	Field	Description
31-4	Reserved	Reserved
3-0	DE_CNT	Defines the width of the active display. The unit of measure is pixels. This register should be set to the desired horizontal resolution. Note that this 12-bit value includes eight bits from register DE_CNTL (see Section 10.3.2.28). The valid range is 1-4095. 0 is invalid.

10.3.2.30 VIDEO DE Line Register (DE_LINL)

The VIDEO DE line register is shown in Figure 10-52 and described in Table 10-62.

Figure 10-52. VIDEO DE Line Register (DE_LINL)

31	8	7	0
Reserved		DE_LIN[7:0]	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-62. VIDEO DE Line Register (DE_LINL) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DE_LIN	Defines the height of the active display. The unit of measure is lines (HSYNC pulses). Set this register to the desired vertical resolution. For interlaced modes; set this register to the number of lines per field; which is half the overall vertical resolution. Note that this 11-bit value includes three bits from register DE_LINH_1 (see Section 10.3.2.31). The valid range is 1-2047. 0 is invalid.

10.3.2.31 VIDEO DE Line Register (DE_LINH_1)

The VIDEO DE line register is shown in Figure 10-53 and described in Table 10-63.

Figure 10-53. VIDEO DE Line Register (DE_LINH_1)

31	3	2	0
Reserved		DE_LIN[10:8]	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-63. VIDEO DE Line Register (DE_LINH_1) Field Descriptions

Bit	Field	Description
31-3	Reserved	Reserved
2-0	DE_LIN	Defines the height of the active display. The unit of measure is lines (HSYNC pulses). Set this register to the desired vertical resolution. For interlaced modes; set this register to the number of lines per field; which is half the overall vertical resolution. Note that this 11-bit value includes eight bits from register DE_LINL (see Section 10.3.2.30). The valid range is 1-2047. 0 is invalid.

10.3.2.32 Video H Resolution Register (HRES_L)

The video H resolution register is shown in Figure 10-54 and described in Table 10-64.

Figure 10-54. Video H Resolution Register (HRES_L)

31	8	7	0
Reserved		H_RES[7:0]	
R-0h		R-0h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-64. Video H Resolution Register (HRES_L) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	H_RES	Measures the time between two HSYNC active edges. The unit of measure is pixels. Note that this 13-bit value includes five bits from register HRES_H (see Section 10.3.2.33).

10.3.2.33 Video H Resolution Register (HRES_H)

The video H resolution register is shown in Figure 10-55 and described in Table 10-65.

Figure 10-55. Video H Resolution Register (HRES_H)

31	5	4	0
Reserved		H_RES[12:8]	
R-0h		R-0h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-65. Video H Resolution Register (HRES_H) Field Descriptions

Bit	Field	Description
31-5	Reserved	Reserved
4-0	H_RES	Measures the time between two HSYNC active edges. The unit of measure is pixels. Note that this 13-bit value includes eight bits from register HRES_L (see Section 10.3.2.32).

10.3.2.34 Video V Resolution Register (VRES_L)

The video V resolution register is shown in Figure 10-56 and described in Table 10-66.

Figure 10-56. Video V Resolution Register (VRES_L)

31	8	7	0
Reserved		V_RES[7:0]	
R-0h		R-0h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-66. Video V Resolution Low Register (VRES_L) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	V_RES	Measures the time between two VSYNC active edges. The unit of measure is lines. Note that this 11-bit value includes three bits from register VRES_H (see Section 10.3.2.35).

10.3.2.35 Video V Resolution Register (VRES_H)

The video V resolution register is shown in Figure 10-57 and described in Table 10-67.

Figure 10-57. Video V Resolution Register (VRES_H)

31	3	2	0
Reserved		V_RES[10:8]	
R-0h		R-0h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-67. Video V Resolution Register (VRES_H) Field Descriptions

Bit	Field	Description
31-3	Reserved	Reserved
2-0	V_RES	Measures the time between two VSYNC active edges. The unit of measure is lines. Note that this 11-bit value includes eight bits from register VRES_L (see Section 10.3.2.34).

10.3.2.36 Video Interlace Adjustment Register (IADJUST)

The video interface adjustment register is shown in Figure 10-58 and described in Table 10-68.

Figure 10-58. Video Interlace Adjustment Register (IADJUST)

31	16
Reserved	
R-0h	

15	3	2	1	0
Reserved		DE_ADJ	F2VADJ	F2VOFST
R-0h		R/W-1h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-68. Video Interlace Adjustment Register (IADJUST) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	DE_ADJ		Setting this bit HIGH disables VSYNC adjustments and sets the DE generator to be more compatible with the existing transmitters. Clearing this bit enables detection circuits to locate the position of VSYNC relative to HSYNC and only include HSYNC edges that are greater than 3/4 lines from VSYNC in the line count for DE_TOP. Enable VSYNC detection when not using DE generator.
		0	Enable VSYNC
		1	Disable VSYNC.
1	F2VADJ		VBIT_TO_VSYNC value (in register VBIT_TO_VSYNC) adjust enable
		0	The VBIT_TO_VSYNC value is not adjusted.
		1	The VBIT_TO_VSYNC value is adjusted during field 2 of an interlace frame according to the setting of the F2VOFST bit.
0	F2VOFST		VBIT_TO_VSYNC value (in register VBIT_TO_VSYNC) offset
		0	If F2VADJ is set, VBIT_TO_VSYNC is decremented by one during field 2 of an interlace frame.
		1	If F2VADJ is set, VBIT_TO_VSYNC is incremented by one during field 2 of an interlace frame.

10.3.2.37 Video SYNC Polarity Detection Register (POL_DETECT)

The video SYNCH polarity detection register is shown in Figure 10-59 and described in Table 10-69.

Figure 10-59. Video SYNC Polarity Detection Register (POL_DETECT)

31	16
Reserved	
R-0h	

15	3	2	1	0
Reserved		I_DET	VPOL_DET	HPOL_DET
R-0h		R-0h	R-0h	R-0h

LEGEND: R = Read only; -*n* = value after reset

Table 10-69. Video SYNC Polarity Detection Register (POL_DETECT) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	I_DET		Interlace detect
		0	Non-interlaced video
		1	Interlaced video. This bit is set by checking for a varying VSYNC timing characteristic of interlaced modes.
1	VPOL_DET		Detected input VSYNC polarity using internal circuit
		0	Active high (leading edge rises)
		1	Active low (leading edge falls)
0	HPOL_DET		Detected input HSYNC polarity using internal circuit
		0	Active high (leading edge rises)
		1	Active low (leading edge falls)

10.3.2.38 Video Hbit to HSYNC Register (HBIT_2HSYNC1)

The video Hbit to HSYNC register is shown in Figure 10-60 and described in Table 10-70.

Figure 10-60. Video Hbit to HSYNC Register (HBIT_2HSYNC1)

31	8	7	0
Reserved		HBIT_TO_HSYNC[7:0]	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-70. Video Hbit to HSYNC Register (HBIT_2HSYNC1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	HBIT_TO_HSYNC	Creates HSYNC pulses. Set this register to the delay from the detection of an EAV sequence (H bit change from 1 to 0) to the active edge of HSYNC. The unit of measure is pixels. Note that this 10-bit value includes two bits from register HBIT_2HSYNC2 (see Section 10.3.2.39). The valid range is 1-1023. 0 is invalid.

10.3.2.39 Video Hbit to HSYNC Register (HBIT_2HSYNC2)

The video Hbit to HSYNC register is shown in Figure 10-61 and described in Table 10-71.

Figure 10-61. Video Hbit to HSYNC Register (HBIT_2HSYNC2)

31	2	1	0
Reserved		HBIT_TO_HSYNC[9:8]	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-71. Video Hbit to HSYNC Register (HBIT_2HSYNC2) Field Descriptions

Bit	Field	Description
31-2	Reserved	Reserved
1-0	HBIT_TO_HSYNC	Creates HSYNC pulses. Set this register to the delay from the detection of an EAV sequence (H bit change from 1 to 0) to the active edge of HSYNC. The unit of measure is pixels. Note that this 10-bit value includes eight bits from register HBIT_2HSYNC1 (see Section 10.3.2.38). The valid range is 1-1023. 0 is invalid.

10.3.2.40 Video Field2 HSYNC Offset Register (FLD2_HS_OFSTL)

The video field2 HSYNC offset register is shown in Figure 10-62 and described in Table 10-72.

Figure 10-62. Video Field2 HSYNC Offset Register (FLD2_HS_OFSTL)

31	8	7	0
Reserved		FIELD2_OFST[7:0]	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-72. Video Field2 HSYNC Offset Register (FLD2_HS_OFSTL) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	FIELD2_OFST	Determines VSYNC pixel offset for the odd field of an interlaced source. Set this to half the number of pixels/line. Note that this 12-bit value includes four bits from register FLD2_HS_OFSTH (see Section 10.3.2.41). The valid range is 1-4095. 0 is invalid.

10.3.2.41 Video Field2 HSYNC Offset Register (FLD2_HS_OFSTH)

Figure 10-63. Video Field2 HSYNC Offset Register (FLD2_HS_OFSTH)

31	4	3	0
Reserved		FIELD2_OFST[11:8]	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

The video field2 HSYNC offset register is shown in Figure 10-63 and described in Table 10-73.

Table 10-73. Video Field2 HSYNC Offset Register (FLD2_HS_OFSTH) Field Descriptions

Bit	Field	Description
31-4	Reserved	Reserved
3-0	FIELD2_OFST	Determines VSYNC pixel offset for the odd field of an interlaced source. Set this to half the number of pixels/line. Note that this 12-bit value includes eight bits from register FLD2_HS_OFSTL (see Section 10.3.2.40). The valid range is 1-4095. 0 is invalid.

10.3.2.42 Video HSYNC Length Register (HWIDTH1)

The video HSYNC length register is shown in Figure 10-64 and described in Table 10-74.

Figure 10-64. Video HSYNC Length Register (HWIDTH1)

31	8	7	0
Reserved		HWIDTH[7:0]	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-74. Video HSYNC Length Register (HWIDTH1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	HWIDTH	Sets the width of the HSYNC pulses. Set this register to the desired HSYNC pulse width. The unit of measure is pixels. Note that this 10-bit value includes two bits from register HWIDTH2 (see Section 10.3.2.43). The valid range is 1-1023. 0 is invalid.

10.3.2.43 Video HSYNC Length Register (HWIDTH2)

The video HSYNC length register is shown in Figure 10-65 and described in Table 10-75.

Figure 10-65. Video HSYNC Length Register (HWIDTH2)

31	2	1	0
Reserved		HWIDTH[9:8]	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-75. Video HSYNC Length Register (HWIDTH2) Field Descriptions

Bit	Field	Description
31-2	Reserved	Reserved
1-0	HWIDTH	Sets the width of the HSYNC pulses. Set this register to the desired HSYNC pulse width. The unit of measure is pixels. Note that this 10-bit value includes eight bits from register HWIDTH1 (see Section 10.3.2.42). The valid range is 1-1023. 0 is invalid.

10.3.2.44 Video Vbit to VSYNC Register (VBIT_TO_VSYNC)

The video Vbit to VSYNC register is shown in Figure 10-66 and desribed in Table 10-76.

Figure 10-66. Video Vbit to VSYNC Register (VBIT_TO_VSYNC)

31	6	5	0
Reserved		VBIT_TO_VSYNC	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-76. Video Vbit to VSYNC Register (VBIT_TO_VSYNC) Field Descriptions

Bit	Field	Description
31-6	Reserved	Reserved
5-0	VBIT_TO_VSYNC	Sets the delay from the detection of V bit changing from 1 to 0 in an EAV sequence; to the asserting edge of VSYNC. The unit of measure is lines. The valid range is 1-63. 0 is invalid.

10.3.2.45 Video VSYNC Length Register (VWIDTH)

The video VSYNC length register is shown in Figure 10-67 and described in Table 10-77.

Figure 10-67. Video VSYNC Length Register (VWIDTH)

31	6	5	0
Reserved		VWIDTH	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-77. Video VSYNC Length Register (VWIDTH) Field Descriptions

Bit	Field	Description
31-6	Reserved	Reserved
5-0	VWIDTH	Sets the width of VSYNC pulse. The unit of measure is lines. The valid range is 1-63. 0 is invalid.

10.3.2.46 Video Control Register (VID_CTRL)

The video control register is shown in Figure 10-68 and described in Table 10-78.

Figure 10-68. Video Control Register (VID_CTRL)

31	16
Reserved	
R-0h	

15	8	7	6	5	4	3	2	1	0
Reserved		IFPOL	Reserved	EXTN	CSCSEL	Reserved		ICLK	
R-0h		R/W-0h	R-0h	R/W-0h	R/W-0h	R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-78. Video Control Register (VID_CTRL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	IFPOL		Invert field polarity. This bit is used when the 656 Flag bit is opposite the standard polarity for Field1 and Field2. Inverting the field polarity causes the sync extraction to format HSYNC and VSYNC properly based on the F bit. In embedded sync mode, the HDMI Transmitter does not detect even from odd field, except based on the setting of the F bit. With explicit syncs, the HDMI transmitter encodes HSYNC and VSYNC across the HDMI/TMDS link regardless of field sequence.
		0	Do not invert field bit
		1	Invert field bit
6	Reserved	0	Reserved
5	EXTN		Extended bit mode
		0	All 8-bit input modes
		1	All 12-bit 4:2:2 input modes. For 4:2:2 inputs wider than 8 bits but less than 12 bits; the unused bits should be cleared to 0.
4	CSCSEL		Color space conversion standard select
		0	BT.601 conversion
		1	BT.709 conversion
3-2	Reserved	0	Reserved
1-0	ICLK		Clock mode If the DEMUX bit in the VID_MODE register is cleared to 0, set the ICLK bit and the pixel replication field of the AVI v2 data byte 5 to the same value. If the DEMUX bit in the VID_MODE register is set to 1, set the pixel replication field of the AVI v2 data byte 5 to the next higher pixel replication rate. For example, if DEMUX = 1 and ICLK = 1, set the pixel replication field of AVI v2 data byte 5 to 3h.
		0	Pixel data is not replicated
		1h	Pixels are replicated once (each sent twice)
		2h	Reserved
		3h	Pixels are replicated 4 times (each sent four times)

10.3.2.47 Video Action Enable Register (VID_ACEN)

The video action enable register is shown in Figure 10-69 and described in Table 10-79.

Figure 10-69. Video Action Enable Register (VID_ACEN)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
WIDE_BUS		Reserved	CLIP_CS_ID	RANGE_CLIP	RGB_2_YCBCR	RANGE_CMPS	DOWN_SMPL
R/W-0h		R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-79. Video Action Enable Register (VID_ACEN) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	WIDE_BUS		Identifies the number of bits per input video channel
		0	8 bits per channel or 24-bit bus mode
		1h	10 bits per channel or 30-bit bus mode
		2h	12 bits per channel or 36-bit bus mode
		3h	Reserved
5	Reserved	0	Reserved
4	CLIP_CS_ID		Identifies the output color space on the link - used by the Clipper block to determine which way to clip
		0	Output color space is RGB 1
		1	Output color space is YCbCr
3	RANGE_CLIP		Enable range clip from 16 to 235 (RGB and Y)/ 240 (CbCr)
		0	Disable
		1	Enable
2	RGB_2_YCBCR		Enable RGB to YCbCr color-space converter
		0	Disable
		1	Enable
1	RANGE_CMPS		Enable range compress 0-255 to 16-234
		0	Disable
		1	Enable
0	DOWN_SMPL		Enable down sampler 4:4:4 to 4:2:2
		0	Disable
		1	Enable

10.3.2.48 Video Mode1 Register (VID_MODE)

The video mode1 register is shown in Figure 10-70 and described in Table 10-80.

Figure 10-70. Video Mode1 Register (VID_MODE)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
DITHER_MODE		DITHER	RANGE	CSC	UPSMP	DEMUX	SYNCEX
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-80. Video Mode1 Register (VID_MODE) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	DITHER_MODE		Identifies the number of bits per output video channel
		0	Dither to 8 bits
		1h	Dither to 10 bits
		2h	Dither to 12 bits
		3h	Reserved
5	DITHER		Dither enable
		0	Dither disabled; the video output is truncated to the output width specified in the DITHER_MODE bit.
		1	Dither enabled; the video output is dithered to the output width specified in the DITHER_MODE bit.
4	RANGE		Data range 16–235 to 0–255 expansion
			When this bit is set, the HDMI transmitter expands the range of pixel data values from 16-235 into the full 8-bit range of 0-255. This is suitable for translating input YCbCr data into output RGB data in PC modes that use the complete range. The HDMI Specification allows one non-CEA-861D mode in the first and only 18-byte descriptor of the Sink's EDID 1.3. This is the native resolution of the Sink, which may be RGB. It may be a standard PC resolution (XGA, SXGA, WXGA, and so on), or a specific native resolution. In these cases (or for a Sink with the Type B HDMI connector, which allows multiple PC modes), when the HDMI transmitter receives YCbCr data, the data must be expanded to full range for outputting RGB full-range modes.
		0	Disable
		1	Enable
3	CSC		YcbCr to RGB color space conversion
		0	Disable
		1	Enable
2	UPSMP		Upsampling 4:2:2 to 4:4:4
		0	Disable
		1	Enable
1	DEMUX		One- to two-data-channel demux
		0	Disable
		1	Enable
0	SYNCEX		Embedded sync extraction
		0	Disable
		1	Enable

10.3.2.49 Video Blanking Register (VID_BLANK1)

The video blanking register is shown in Figure 10-71 and described in Table 10-81.

Figure 10-71. Video Blanking Register (VID_BLANK1)

31	8	7	0
Reserved		VID_BLANK1	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-81. Video Blanking Registers (VID_BLANK1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	VID_BLANK1	Defines the video blanking value for Channel 1 (Blue).

10.3.2.50 Video Blanking Register (VID_BLANK2)

The video blanking register is shown in Figure 10-72 and described in Table 10-82.

Figure 10-72. Video Blanking Register (VID_BLANK2)

31	8	7	0
Reserved		VID_BLANK2	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-82. Video Blanking Register (VID_BLANK2) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	VID_BLANK2	Defines the video blanking value for Channel 2 (Green).

10.3.2.51 Video Blanking Register (VID_BLANK3)

The video blanking register is shown in Figure 10-73 and described in Table 10-83.

Figure 10-73. Video Blanking Register (VID_BLANK3)

31	8	7	0
Reserved		VID_BLANK3	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-83. Video Blanking Register (VID_BLANK3) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	VID_BLANK3	Defines the video blanking value for Channel 3 (Red).

10.3.2.52 Deep Color Header Register (DC_HEADER)

The deep color header register is shown in Figure 10-74 and described in Table 10-84.

Figure 10-74. Deep Color Header Register (DC_HEADER)

31	8	7	0
Reserved		DC_HEADER	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-84. Deep Color Header Register (DC_HEADER) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DC_HEADER	The least-significant byte of the deep color header that sends the TMDS dynamic phase once per frame.

10.3.2.53 Video Mode2 Register (VID_DITHER)

The video mode2 register is shown in Figure 10-75 and described in Table 10-85.

Figure 10-75. Video Mode2 Register (VID_DITHER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	M_D2	UP2	422_EN	D_BC_EN	D_GC_EN	D_RC_EN	DRD
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-85. Video Mode2 Register (VID_DITHER) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6	M_D2		Dither + round option
		0	Disable the function
		1	Enable the function
5	UP2		Dither + 2'b10 option
		0	Disable the function
		1	Enable the function
4	422_EN		Enable Mode 4:2:2 for dithering and clipping
		0	Disable the function
		1	Enable the function
3	D_BC_EN		Enable adding random number on Blue channel data.
		0	Disable the function
		1	Enable the function
2	D_GC_EN		Enable adding random number on Green channel data.
		0	Disable the function
		1	Enable the function
1	D_RC_EN		Enable adding random number on Red channel data.
		0	Disable the function
		1	Enable the function
0	DRD		Dither round: Add random number + 2b10 then truncate the LSB 2-bit.
		0	Disable the function
		1	Enable the function

10.3.2.54 RGB_2_xvYCC Control Register (RGB2XVYCC_CT)

The RGB_2_xvYCC control register is shown in Figure 10-76 and described in Table 10-86.

Figure 10-76. RGB_2_xvYCC Control Register (RGB2XVYCC_CT)

31	16
Reserved	
R-0h	

15	3	2	1	0
Reserved		XV_CO_OV	XV_FUS	XV_EN
R-0h		R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-86. RGB_2_xvYCC control Register (RGB2XVYCC_CT) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	XV_CO_OV		Override internal CSC coefficients with register 51 to XX values
		0	Disable the function
		1	Enable the function
1	XV_FUS		xvYCC fullscale mode
		0	Disable the function
		1	Enable the function
0	XV_EN		xvYCC enable
		0	Disable the function
		1	Enable the function

10.3.2.55 RGB_2_xvYCC Conversion R_2_Y Register (R2Y_COEFF_LOW)

The RGB_2_xvYCC conversion R_2_Y register is shown in Figure 10-77 and described in Table 10-87.

Figure 10-77. RGB_2_xvYCC Conversion R_2_Y Register (R2Y_COEFF_LOW)

31	8	7	0
Reserved		R2YCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-87. RGB_2_xvYCC Conversion R_2_Y Register (R2Y_COEFF_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	R2YCOEFF_L	RGB to xvYCC conversion. R to Y coefficient lower byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.56 RGB_2_xvYCC Conversion R_2_Y Register (R2Y_COEFF_UP)

The RGB_2_xvYCC conversion R_2_Y register is shown in Figure 10-78 and described in Table 10-88.

Figure 10-78. RGB_2_xvYCC Conversion R_2_Y Register (R2Y_COEFF_UP)

31	8	7	0
Reserved		R2YCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-88. RGB_2_xvYCC Conversion R_2_Y Register (R2Y_COEFF_UP) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	R2YCOEFF_H	RGB to xvYCC conversion. R to Y coefficient upper byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.57 RGB_2_xvYCC Conversion G_2_Y Register (G2Y_COEFF_LOW)

The RGB_2_xvYCC conversion G_2_Y register is shown in Figure 10-79 and described in Table 10-89.

Figure 10-79. RGB_2_xvYCC Conversion G_2_Y Register (G2Y_COEFF_LOW)

31	8	7	0
Reserved		G2YCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-89. RGB_2_xvYCC Conversion G_2_Y Register (G2Y_COEFF_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	G2YCOEFF_L	RGB to xvYCC conversion. G to Y coefficient lower byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.58 RGB_2_xvYCC Conversion G_2_Y Register (G2Y_COEFF_UP)

The RGB_2_xvYCC conversion G_2_Y register is shown in Figure 10-80 and described in Table 10-90.

Figure 10-80. RGB_2_xvYCC Conversion G_2_Y Register (G2Y_COEFF_UP)

31	8	7	0
Reserved		G2YCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-90. RGB_2_xvYCC Conversion G_2_Y Register (G2Y_COEFF_UP) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	G2YCOEFF_H	RGB to xvYCC conversion. G to Y coefficient upper byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.59 RGB_2_xvYCC Conversion B_2_Y Register (B2Y_COEFF_LOW)

The RGB_2_xvYCC conversion B_2_Y register is shown in Figure 10-81 and described in Table 10-91.

Figure 10-81. RGB_2_xvYCC Conversion B_2_Y Register (B2Y_COEFF_LOW)

31	8	7	0
Reserved		B2YCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-91. RGB_2_xvYCC Conversion B_2_Y Register (B2Y_COEFF_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	B2YCOEFF_L	RGB to xvYCC conversion. B to Y coefficient lower byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.60 RGB_2_xvYCC Conversion B_2_Y Register (B2Y_COEFF_UP)

The RGB_2_xvYCC conversion B_2_Y register is shown in Figure 10-82 and described in Table 10-92.

Figure 10-82. RGB_2_xvYCC Conversion B_2_Y Register (B2Y_COEFF_UP)

31	8	7	0
Reserved		B2YCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-92. RGB_2_xvYCC Conversion B_2_Y Register (B2Y_COEFF_UP) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	B2YCOEFF_H	RGB to xvYCC conversion. B to Y coefficient upper byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.61 RGB_2_xvYCC Conversion R_2_Cb Register (R2CB_COEFF_LOW)

The RGB_2_xvYCC conversion R_2_Cb register is shown in Figure 10-83 and described in Table 10-93.

Figure 10-83. RGB_2_xvYCC Conversion R_2_Cb Register (R2CB_COEFF_LOW)

31	8	7	0
Reserved		R2CBCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-93. RGB_2_xvYCC Conversion R_2_Cb Register (R2CB_COEFF_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	R2CBCOEFF_L	RGB to xvYCC conversion. R to Cb coefficient lower byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.62 RGB_2_xvYCC Conversion R_2_Cb Register (R2CB_COEFF_UP)

The RGB_2_xvYCC conversion R_2_Cb register is shown in Figure 10-84 and described in Table 10-94.

Figure 10-84. RGB_2_xvYCC Conversion R_2_Cb Register (R2CB_COEFF_UP)

31	8	7	0
Reserved		R2CBCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-94. RGB_2_xvYCC Conversion R_2_Cb Register (R2CB_COEFF_UP) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	R2CBCOEFF_H	RGB to xvYCC conversion. R to Cb coefficient upper byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.63 RGB_2_xvYCC Conversion G_2_Cb Register (G2CB_COEFF_LOW)

The RGB_2_xvYCC conversion G_2_Cb register is shown in Figure 10-85 and described in Table 10-95.

Figure 10-85. RGB_2_xvYCC Conversion G_2_Cb Register (G2CB_COEFF_LOW)

31	8	7	0
Reserved		G2CBCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-95. RGB_2_xvYCC Conversion G_2_Cb Register (G2CB_COEFF_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	G2CBCOEFF_L	RGB to xvYCC conversion. G to Cb coefficient lower byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.64 RGB_2_xvYCC Conversion G_2_Cb Register (G2CB_COEFF_UP)

The RGB_2_xvYCC conversion G_2_Cb register is shown in Figure 10-86 and described in Table 10-96.

Figure 10-86. RGB_2_xvYCC Conversion G_2_Cb Register (G2CB_COEFF_UP)

31	8	7	0
Reserved		G2CBCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-96. RGB_2_xvYCC Conversion G_2_Cb Register (G2CB_COEFF_UP) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	G2CBCOEFF_H	RGB to xvYCC conversion. G to Cb coefficient upper byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.65 RGB_2_xvYCC Conversion B_2_Cb Register (B2CB_COEFF_LOW)

The RGB_2_xvYCC conversion G_2_Cb register is shown in Figure 10-87 and described in Table 10-97.

Figure 10-87. RGB_2_xvYCC Conversion B_2_Cb Register (B2CB_COEFF_LOW)

31	8	7	0
Reserved		B2CBCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-97. RGB_2_xvYCC Conversion B_2_Cb Register (B2CB_COEFF_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	B2CBCOEFF_L	RGB to xvYCC conversion. B to Cb coefficient lower byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.66 RGB_2_xvYCC Conversion B_2_Cb Register (B2CB_COEFF_UP)

The RGB_2_xvYCC conversion G_2_Cb register is shown in Figure 10-88 and described in Table 10-98.

Figure 10-88. RGB_2_xvYCC Conversion B_2_Cb Register (B2CB_COEFF_UP)

31	8	7	0
Reserved		B2CBCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-98. RGB_2_xvYCC Conversion B_2_Cb Register (B2CB_COEFF_UP) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	B2CBCOEFF_H	RGB to xvYCC conversion. B to Cb coefficient upper byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.67 RGB_2_xvYCC Conversion R_2_Cr Register (R2CR_COEFF_LOW)

The RGB_2_xvYCC conversion R_2_Cr register is shown in Figure 10-89 and described in Table 10-99.

Figure 10-89. RGB_2_xvYCC Conversion R_2_Cr Register (R2CR_COEFF_LOW)

31	8	7	0
Reserved		R2CRCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-99. RGB_2_xvYCC Conversion R_2_Cr Register (R2CR_COEFF_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	R2CRCOEFF_L	RGB to xvYCC conversion. R to Cr coefficient lower byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.68 RGB_2_xvYCC Conversion R_2_Cr Register (R2CR_COEFF_UP)

The RGB_2_xvYCC conversion R_2_Cr register is shown in Figure 10-90 and described in Table 10-100.

Figure 10-90. RGB_2_xvYCC Conversion R_2_Cr Register (R2CR_COEFF_UP)

31	8	7	0
Reserved		R2CRCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-100. RGB_2_xvYCC Conversion R_2_Cr Register (R2CR_COEFF_UP) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	R2CRCOEFF_H	RGB to xvYCC conversion. R to Cr coefficient upper byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.69 RGB_2_xvYCC Conversion G_2_Cr Register (G2CR_COEFF_LOW)

The RGB_2_xvYCC conversion G_2_Cr register is shown in Figure 10-91 and described in Table 10-101.

Figure 10-91. RGB_2_xvYCC Conversion G_2_Cr Register (G2CR_COEFF_LOW)

31	8	7	0
Reserved		G2CRCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-101. RGB_2_xvYCC Conversion G_2_Cr Register (G2CR_COEFF_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	G2CRCOEFF_L	RGB to xvYCC conversion. G to Cr coefficient lower byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.70 RGB_2_xvYCC Conversion G_2_Cr Register (G2CR_COEFF_UP)

The RGB_2_xvYCC conversion G_2_Cr register is shown in Figure 10-92 and described in Table 10-102.

Figure 10-92. RGB_2_xvYCC Conversion G_2_Cr Register (G2CR_COEFF_UP)

31	8	7	0
Reserved		G2CRCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-102. RGB_2_xvYCC Conversion G_2_Cr Register (G2CR_COEFF_UP) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	G2CRCOEFF_H	RGB to xvYCC conversion. G to Cr coefficient upper byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.71 RGB_2_xvYCC Conversion B_2_Cr Register (B2CR_COEFF_LOW)

The RGB_2_xvYCC conversion B_2_Cr register is shown in Figure 10-93 and described in Table 10-103.

Figure 10-93. RGB_2_xvYCC Conversion B_2_Cr Register (B2CR_COEFF_LOW)

31	8	7	0
Reserved		B2CRCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-103. RGB_2_xvYCC Conversion B_2_Cr Register (B2CR_COEFF_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	B2CRCOEFF_L	RGB to xvYCC conversion. B to Cr coefficient lower byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.72 RGB_2_xvYCC Conversion B_2_Cr Register (B2CR_COEFF_UP)

The RGB_2_xvYCC conversion B_2_Cr register is shown in Figure 10-94 and described in Table 10-104.

Figure 10-94. RGB_2_xvYCC Conversion B_2_Cr Register (B2CR_COEFF_UP)

31	8	7	0
Reserved		B2CRCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-104. RGB_2_xvYCC Conversion B_2_Cr Register (B2CR_COEFF_UP) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	B2CRCOEFF_H	RGB to xvYCC conversion. B to Cr coefficient upper byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.73 RGB_2_xvYCC RGB Input Offset Register (RGB_OFFSET_LOW)

The RGB_2_xvYCC RGB input offset register is shown in Figure 10-95 and described in Table 10-105.

Figure 10-95. RGB_2_xvYCC RGB Input Offset Register (RGB_OFFSET_LOW)

31	8	7	0
Reserved		RGB_OFFS_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-105. RGB_2_xvYCC RGB Input Offset Register (RGB_OFFSET_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	RGB_OFFS_L	Input RGB offset value lower byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.74 RGB_2_xvYCC RGB Input Offset Register (RGB_OFFSET_UP)

The RGB_2_xvYCC RGB input offset register is shown in Figure 10-96 and described in Table 10-106.

Figure 10-96. RGB_2_xvYCC RGB Input Offset Register (RGB_OFFSET_UP)

31	8	7	0
Reserved		RGB_OFFS_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-106. RGB_2_xvYCC RGB Input Offset Register (RGB_OFFSET_UP) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	RGB_OFFS_H	Input RGB offset value upper byte (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.75 RGB_2_xvYCC Conversion Y Output Offset Register (Y_OFFSET_LOW)

The RGB_2_xvYCC conversion Y output offset register is shown in Figure 10-97 and described in Table 10-107.

Figure 10-97. RGB_2_xvYCC Conversion Y Output Offset Register (Y_OFFSET_LOW)

31	7	6	0
Reserved		Y_OFFS_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-107. RGB_2_xvYCC Conversion Y Output Offset Register (Y_OFFSET_LOW) Field Descriptions

Bit	Field	Description
31-7	Reserved	Reserved
6-0	Y_OFFS_L	Output Y offset value lower 7 bits (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.76 RGB_2_xvYCC Conversion Y Output Offset Register (Y_OFFSET_UP)

The RGB_2_xvYCC conversion Y output offset register is shown in Figure 10-98 and described in Table 10-108.

Figure 10-98. RGB_2_xvYCC Conversion Y Output Offset Register (Y_OFFSET_UP)

31	7	6	0
Reserved		Y_OFFS_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-108. RGB_2_xvYCC Conversion Y Output Offset Register (Y_OFFSET_UP) Field Descriptions

Bit	Field	Description
31-7	Reserved	Reserved
6-0	Y_OFFS_H	Output Y offset value upper 7 bits (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.77 RGB_2_xvYCC Conversion CbCr Output Offset Register (CBCR_OFFSET_LOW)

The RGB_2_xvYCC conversion CbCr output offset register is shown in Figure 10-99 and described in Table 10-109.

Figure 10-99. RGB_2_xvYCC Conversion CbCr Output Offset Register (CBCR_OFFSET_LOW)

31	7	6	0
Reserved		CBCR_OFFS_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-109. RGB_2_xvYCC Conversion CbCr Output Offset Register (CBCR_OFFSET_LOW) Field Descriptions

Bit	Field	Description
31-7	Reserved	Reserved
6-0	CBCR_OFFS_L	Output CbCr offset value lower 7 bits (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.78 RGB_2_xvYCC Conversion CbCr Output Offset Register (CBCR_OFFSET_UP)

The RGB_2_xvYCC conversion CbCr output offset register is shown in Figure 10-100 and described in Table 10-110.

Figure 10-100. RGB_2_xvYCC Conversion CbCr Output Offset Register (CBCR_OFFSET_UP)

31	7	6	0
Reserved		CBCR_OFFS_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-110. RGB_2_xvYCC Conversion CbCr Output Offset Register (CBCR_OFFSET_UP) Field Descriptions

Bit	Field	Description
31-7	Reserved	Reserved
6-0	CBCR_OFFS_H	Output CbCr offset value upper 7 bits (override internal CSC value when XV_CO_OV = 1 in register RGB_2_XVYCC_CT)

10.3.2.79 Interrupt State Register (INTR_STATE)

The interrupt state register is shown in Figure 10-101 and described in Table 10-111.

Figure 10-101. Interrupt State Register (INTR_STATE)

31	1	0
Reserved		INTR
R-0h		R-0h

LEGEND: R = Read only; -*n* = value after reset

Table 10-111. Interrupt State Register (INTR_STATE) Field Descriptions

Bit	Field	Description
31-1	Reserved	Reserved
0	INTR	Interrupt State. When an interrupt is asserted; this bit is set to 1. The polarity of the INT output signal is set using this bit and the POLARITY bit in the INT_CTRL register. Only INTR1, INTR2, INTR3, and INTR4 bits with matching set bits in INT_UNMASK can contribute to setting the INTR bit.

10.3.2.80 Interrupt Source Register (INTR1)

The interrupt source register is shown in Figure 10-102 and described in Table 10-112.

Figure 10-102. Interrupt Source Register (INTR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
SOFT	HPD	RSEN	DROP_SAMPLE	BIP_HASE_ERR	RI_128	OVER_RUN	UNDER_RUN
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R = Read only; -*n* = value after reset

Table 10-112. Interrupt Source Register (INTR1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7	SOFT	Software Induced Interrupt. Allows the firmware to generate an interrupt directly.
6	HPD	Monitor detect interrupt. Asserted if hot plug detect has changed state. The HDMI transmitter signals a change in the connectivity to a Sink, either unplug or plug. HDMI specifies that hot plug be active only when the Sink s EDID is ready to be read and that hot plug be toggled any time there is a change in connectivity downstream of an attached repeater.
5	RSEN	Receiver sense interrupt asserted if RSEN has changed. This interrupt is set whenever V_{CC} is applied to or removed from the attached HDMI receiver chip. A receiver with multiple input ports can also disconnect the TMDS termination to the unused port, which triggers this RSEN interrupt.
4	DROP_SAMPLE	New preamble forced to drop sample (S/PDIF input only). If the HDMI transmitter detects an 8-bit preamble in the S/PDIF input stream before the subframe has been captured, this interrupt is set. A S/PDIF input that stops signaling or a flat line condition can create such a premature preamble.
3	BIP_HASE_ERR	Input S/PDIF stream has bi-phase error. This can occur when there is noise or an Fs rate change on the S/PDIF input.
2	RI_128	Input counted past frame count threshold set in RI_128_COMP register. This interrupt occurs when the count written to register RI_128_COMP is matched by the VSYNC (frame) counter in the HDMI transmitter. It should trigger the firmware to perform a link integrity check. Such a match occurs every 128 frames.
1	OVER_RUN	Audio FIFO overflow. This interrupt occurs if the audio FIFO overflows when more samples are written into it than are drawn out across the HDMI link. Such a condition can occur from a transient change in the Fs or pixel clock rate.
0	UNDER_RUN	Audio FIFO underflow. Similar to OVER_RUN, this interrupt occurs when the audio FIFO empties.

10.3.2.81 Interrupt Source Register (INTR2)

The interrupt source register is shown in Figure 10-103 and described in Table 10-113.

Figure 10-103. Interrupt Source Register (INTR2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
BCAP_DONE	SPDIF_PAR	ENC_DIS	PREAM_ERR	CTS_CHG	ACR_OVR	TCLK_STBL	VSYNC_REC
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-x	R-0h

LEGEND: R = Read only; -*n* = value after reset; x = value is indeterminate

Table 10-113. Interrupt Source Register (INTR2) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7	BCAP_DONE	If set, this interrupt detected that the FIFORDY bit is set to 1.
6	SPDIF_PAR	S/PDIF parity error. The S/PDIF stream includes a parity (P) bit at the end of each sub-frame. An interrupt occurs if the calculated parity does not match the state of this bit.
5	ENC_DIS	The ENC_EN bit (in register HDCP_CTRL) changed from 1 to 0. This interrupt occurs if encryption is turned off.
4	PREAM_ERR	This condition is the opposite of the condition that causes DROP_SAMPLE (in register INTR1). This interrupt occurs if a preamble is expected but not found when decoding the S/PDIF stream.
3	CTS_CHG	Change in ACR CTS value. This interrupt occurs when the change is of an unexpected magnitude. Such an interrupt should be expected when changing Fs or pixel clock frequency.
2	ACR_OVR	ACR packet overwrite. This interrupt occurs if the HDMI transmitter puts a NCTS packet into the queue before the previous NCTS packet has been sent. This can happen if very long active data times do not allow for sufficient NCTS packet bandwidth. For all CEA- 861D modes, no ACR_OVR interrupt should occur.
1	TCLK_STBL	Whenever IDCK changes, there is a temporary instability in the internal clocking. This interrupt is set when the internal clocking has stabilized.
0	VSYNC_REC	Asserted when VSYNC active edge is recognized. It is useful for triggering firmware actions that occur during vertical blanking.

10.3.2.82 Interrupt Source Register (INTR3)

The interrupt source register is shown in Figure 10-104 and described in Table 10-114.

Figure 10-104. Interrupt Source Register (INTR3)

31	30	29	28	27	26	25	16
Reserved							
R-0h							

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
RI_ERR_3	RI_ERR_2	RI_ERR_1	RI_ERR_0	DDC_CMD_DONE	DDC_FIFO_HALF	DDC_FIFO_FULL	DDC_FIFO_EMPTY
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R = Read only; -*n* = value after reset

Table 10-114. Interrupt Source Register (INTR3) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7	RI_ERR_3	Ri and Ri do not match during frame 127 (ICNT .1).
6	RI_ERR_2	Ri and Ri do not match during frame 0 (ICNT).
5	RI_ERR_1	Ri did not change between frame 127 and 0.
4	RI_ERR_0	Ri not read within one frame.
3	DDC_CMD_DONE	DDC command is complete.
2	DDC_FIFO_HALF	DDC FIFO is half full.
1	DDC_FIFO_FULL	DDC FIFO is full.
0	DDC_FIFO_EMPTY	DDC FIFO is empty. Reset value is 0, but can be set after reset since the FIFO is empty.

10.3.2.83 Interrupt Source Register (INTR4)

The interrupt source register is shown in Figure 10-105 and described in Table 10-115.

Figure 10-105. Interrupt Source Register (INTR4)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved				REG_INTR4_STAT3	REG_INTR4_STAT2	REG_INTR4_STAT1	DSD_INVALID
R-0h				R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear, write 0 has no effect; -*n* = value after reset

Table 10-115. Interrupt Source Register (INTR4) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	REG_INTR4_STAT3		CEC interrupt.
		0	No interrupt occurred
		1	Interrupt asserted
2	REG_INTR4_STAT2		For each interrupt bit:
		0	No interrupt occurred
		1	Interrupt asserted
1	REG_INTR4_STAT1		For each interrupt bit:
		0	No interrupt occurred
		1	Interrupt asserted
0	DSD_INVALID	0	DSD stream got invalid sequence: more then 24 bits of the same value. Asserted if set to 1.

10.3.2.84 Interrupt Unmask Register (INT_UNMASK1)

The interrupt unmask register is shown in Figure 10-106 and described in Table 10-116.

Figure 10-106. Interrupt Unmask Register (INT_UNMASK1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
SOFT	HPD	RSEN	DROP_SAMPLE	BIP_HASE_ERR	RI_128	OVER_RUN	UNDER_RUN
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-116. Interrupt Unmask Register (INT_UNMASK1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7	SOFT	Software Induced Interrupt. Allows the firmware to generate an interrupt directly.
6	HPD	Monitor detect interrupt. Asserted if hot plug detect has changed state. The HDMI transmitter signals a change in the connectivity to a Sink, either unplug or plug. HDMI specifies that hot plug be active only when the Sink s EDID is ready to be read and that hot plug be toggled any time there is a change in connectivity downstream of an attached repeater.
5	RSEN	Receiver sense interrupt asserted if RSEN has changed. This interrupt is set whenever V_{CC} is applied to or removed from the attached HDMI receiver chip. A receiver with multiple input ports can also disconnect the TMDS termination to the unused port, which triggers this RSEN interrupt.
4	DROP_SAMPLE	New preamble forced to drop sample (S/PDIF input only). If the HDMI transmitter detects an 8-bit preamble in the S/PDIF input stream before the subframe has been captured, this interrupt is set. A S/PDIF input that stops signaling or a flat line condition can create such a premature preamble.
3	BIP_HASE_ERR	Input S/PDIF stream has bi-phase error. This can occur when there is noise or an Fs rate change on the S/PDIF input.
2	RI_128	Input counted past frame count threshold set in RI_128_COMP register. This interrupt occurs when the count written to register RI_128_COMP is matched by the VSYNC (frame) counter in the HDMI transmitter. It should trigger the firmware to perform a link integrity check; such a match occurs every 128 frames.
1	OVER_RUN	Audio FIFO overflow. This interrupt occurs if the audio FIFO overflows when more samples are written into it than are drawn out across the HDMI link. Such a condition can occur from a transient change in the Fs or pixel clock rate.
0	UNDER_RUN	Audio FIFO underflow. Similar to OVER_RUN. This interrupt occurs when the audio FIFO empties.

10.3.2.85 Interrupt Unmask Register (INT_UNMASK2)

The interrupt unmask register is shown in Figure 10-107 and described in Table 10-117.

Figure 10-107. Interrupt Unmask Register (INT_UNMASK2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
BCAP_DONE	SPDIF_PAR	ENC_DIS	PREAM_ERR	CTS_CHG	ACR_OVR	TCLK_STBL	VSYNC_REC
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

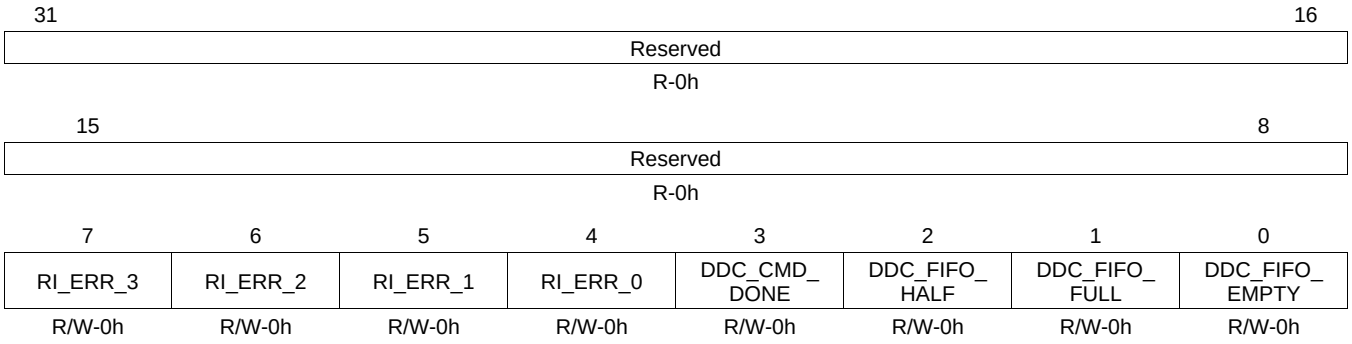
Table 10-117. Interrupt Unmask Register (INT_UNMASK2) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7	BCAP_DONE	If set, this interrupt detected that the FIFORDY bit is set to 1.
6	SPDIF_PAR	S/PDIF parity error. The S/PDIF stream includes a parity (P) bit at the end of each sub-frame. An interrupt occurs if the calculated parity does not match the state of this bit.
5	ENC_DIS	The ENC_EN bit (in register HDCP_CTRL) changed from 1 to 0. This interrupt occurs if encryption is turned off.
4	PREAM_ERR	This condition is the opposite of the condition that causes DROP_SAMPLE (in register INTR1). This interrupt occurs if a preamble is expected but not found when decoding the S/PDIF stream.
3	CTS_CHG	Change in ACR CTS value. This interrupt occurs when the change is of an unexpected magnitude. Such an interrupt should be expected when changing Fs or pixel clock frequency.
2	ACR_OVR	ACR packet overwrite. This interrupt occurs if the HDMI transmitter puts a NCTS packet into the queue before the previous NCTS packet has been sent. This can happen if very long active data times do not allow for sufficient NCTS packet bandwidth. For all CEA- 861D modes, no ACR_OVR interrupt should occur.
1	TCLK_STBL	TCLK_STABLE. Whenever IDCK changes, there is a temporary instability in the internal clocking. This interrupt is set when the internal clocking has stabilized.
0	VSYNC_REC	Asserted when VSYNC active edge is recognized, it is useful for triggering firmware actions that occur during vertical blanking.

10.3.2.86 Interrupt Unmask Register (INT_UNMASK3)

The interrupt unmask register is shown in Figure 10-108 and described in Table 10-118.

Figure 10-108. Interrupt Unmask Register (INT_UNMASK3)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

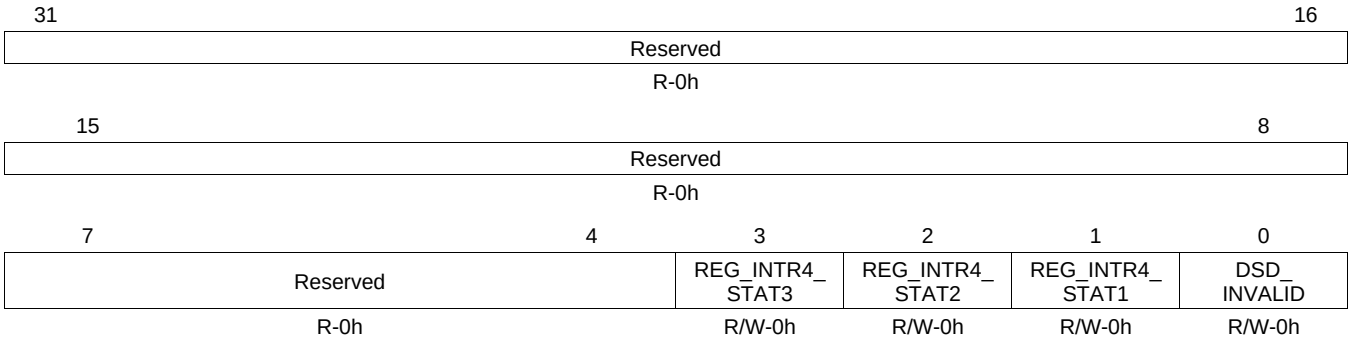
Table 10-118. Interrupt Unmask Register (INT_UNMASK3) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7	RI_ERR_3	Ri and Ri do not match during frame 127 (ICNT .1).
6	RI_ERR_2	Ri and Ri do not match during frame 0 (ICNT).
5	RI_ERR_1	Ri did not change between frame 127 and 0.
4	RI_ERR_0	Ri not read within one frame.
3	DDC_CMD_DONE	DDC command is complete.
2	DDC_FIFO_HALF	DDC FIFO is half full.
1	DDC_FIFO_FULL	DDC FIFO is full.
0	DDC_FIFO_EMPTY	DDC FIFO is empty. Reset value is 0, but can be set after reset since the FIFO Is empty.

10.3.2.87 Interrupt Unmask Register (INT_UNMASK4)

The interrupt unmask register is shown in Figure 10-109 and described in Table 10-119.

Figure 10-109. Interrupt Unmask Register (INT_UNMASK4)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-119. Interrupt Unmask Register (INT_UNMASK4) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	REG_INTR4_STAT3		CEC interrupt
		0	No interrupt occurred
		1	Interrupt asserted
2	REG_INTR4_STAT2		For each interrupt bit:
		0	No interrupt occurred
		1	Interrupt asserted
1	REG_INTR4_STAT1		For each interrupt bit:
		0	No interrupt occurred
		1	Interrupt asserted
0	DSD_INVALID	0	DSD stream got invalid sequence: more then 24 bits of the same value. Asserted if set to 1.

10.3.2.88 Interrupt Control Register (INT_CTRL)

The interrupt control register is shown in Figure 10-110 and described in Table 10-120.

Figure 10-110. Interrupt Control Register (INT_CTRL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved				SOFT_INTR	OPEN_DRAIN	POLARITY	Reserved
R-0h				R/W-0h	R/W-0h	R/W-1h	R-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-120. Interrupt Control Register (INT_CTRL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	SOFT_INTR		Set software interrupt
		0	Clear interrupt
		1	Set interrupt
2	OPEN_DRAIN		INT pin output type
		0	Push/Pull
		1	Open Drain pin
1	POLARITY		INT pin assertion level
		0	Assert HIGH
		1	Assert LOW
0	Reserved	0	Reserved

10.3.2.89 xvYCC_2_RGB Control Register (XVYCC2RGB_CTL)

The xvYCC_2_RGB control register is shown in Figure 10-111 and described in Table 10-121.

Figure 10-111. xvYCC_2_RGB Control Register (XVYCC2RGB_CTL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved			EXP_ONLY	BYP_ALL	SW_OVR	FULLRANGE	XVYCCSEL
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-121. xvYCC_2_RGB Control Register (XVYCC2RGB_CTL) Field Descriptions

Bit	Field	Description
31-5	Reserved	Reserved
4	EXP_ONLY	When set to 1, indicates the internal CSC will be bypassed and only range expansion is on.
3	BYP_ALL	When set to 1, indicates all the functions will be bypassed.
2	SW_OVR	Software over ride. When turned on, all coefficients and offsets are coming from registers not internal hardware decision.
1	FULLRANGE	xvYCC full-range expansion enable
0	XVYCCSEL	Indicates the source. When set to 1, is xvYCC; when cleared to 0, is YcbCr. It can be configured by firmware. Under auto video configure (AVC) mode, this control comes from HDMI packet decoding.

10.3.2.90 xvYCC_2_RGB Conversion Y_2_R Register (Y2R_COEFF_LOW)

The xvYCC_2_RGB conversion Y_2_R register is shown in Figure 10-112 and described in Table 10-122.

Figure 10-112. xvYCC_2_RGB Conversion Y_2_R Register (Y2R_COEFF_LOW)

31	8	7	0
Reserved		Y2RCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-122. xvYCC_2_RGB Conversion Y_2_R Register (Y2R_COEFF_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	Y2RCOEFF_L	xvYCC to RGB conversion. Y to R coefficient lower byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.91 xvYCC_2_RGB Conversion Y_2_R Register (Y2R_COEFF_UP)

The xvYCC_2_RGB conversion Y_2_R register is shown in Figure 10-113 and described in Table 10-123.

Figure 10-113. xvYCC_2_RGB Conversion Y_2_R Register (Y2R_COEFF_UP)

31	5	4	0
Reserved		Y2RCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-123. xvYCC_2_RGB Conversion Y_2_R Register (Y2R_COEFF_UP) Field Descriptions

Bit	Field	Description
31-5	Reserved	Reserved
4-0	Y2RCOEFF_H	xvYCC to RGB conversion. Y to R coefficient upper byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.92 xvYCC_2_RGB Conversion Cr_2_R Register (CR2R_COEFF_LOW)

The xvYCC_2_RGB conversion Cr_2_R register is shown in Figure 10-114 and described in Table 10-124.

Figure 10-114. xvYCC_2_RGB Conversion Cr_2_R Register (CR2R_COEFF_LOW)

31	8	7	0
Reserved		CR2RCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-124. xvYCC_2_RGB Conversion Cr_2_R Register (CR2R_COEFF_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CR2RCOEFF_L	xvYCC to RGB conversion. Cr to R coefficient lower byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.93 xvYCC_2_RGB Conversion Cr_2_R Register (C2R2R_COEFF_UP)

The xvYCC_2_RGB conversion Cr_2_R register is shown in Figure 10-115 and described in Table 10-125.

Figure 10-115. xvYCC_2_RGB Conversion Cr_2_R Register (C2R2R_COEFF_UP)

31	5	4	0
Reserved		CR2RCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-125. xvYCC_2_RGB Conversion Cr_2_R Register (C2R2R_COEFF_UP) Field Descriptions

Bit	Field	Description
31-5	Reserved	Reserved
4-0	CR2RCOEFF_H	xvYCC to RGB conversion. Cr to R coefficient upper byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.94 xvYCC_2_RGB Conversion Cb_2_B Register (CB2B_COEFF_LOW)

The xvYCC_2_RGB conversion Cb_2_B register is shown in Figure 10-116 and described in Table 10-126.

Figure 10-116. xvYCC_2_RGB Conversion Cb_2_B Register (CB2B_COEFF_LOW)

31	8	7	0
Reserved		CB2BCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-126. xvYCC_2_RGB Conversion Cb_2_B Register (CB2B_COEFF_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CB2BCOEFF_L	xvYCC to RGB conversion. Cb to B coefficient lower byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.95 xvYCC_2_RGB Conversion Cb_2_B Register (CB2B_COEFF_UP)

The xvYCC_2_RGB Conversion Cb_2_B Register is shown in Figure 10-117 and described in Table 10-127.

Figure 10-117. xvYCC_2_RGB Conversion Cb_2_B Register (CB2B_COEFF_UP)

31	5	4	0
Reserved		CB2BCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-127. xvYCC_2_RGB Conversion Cb_2_B Register (CB2B_COEFF_UP) Field Descriptions

Bit	Field	Description
31-5	Reserved	Reserved
4-0	CB2BCOEFF_H	xvYCC to RGB conversion. Cb to B coefficient upper byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.96 xvYCC_2_RGB Conversion Cr_2_G Register (CR2G_COEFF_LOW)

The xvYCC_2_RGB conversion Cr_2_G register is shown in Figure 10-118 and described in Table 10-128.

Figure 10-118. xvYCC_2_RGB Conversion Cr_2_G Register (CR2G_COEFF_LOW)

31	8	7	0
Reserved		CR2GCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-128. CR2G_COEFF_LOW Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CR2GCOEFF_L	xvYCC to RGB conversion. Cr to G coefficient lower byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.97 xvYCC_2_RGB Conversion Cr_2_G Register (CR2G_COEFF_UP)

The xvYCC_2_RGB conversion Cr_2_G register is shown in Figure 10-119 and described in Table 10-129.

Figure 10-119. xvYCC_2_RGB Conversion Cr_2_G Register (CR2G_COEFF_UP)

31	5	4	0
Reserved		CR2GCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-129. xvYCC_2_RGB Conversion Cr_2_G Register (CR2G_COEFF_UP) Field Descriptions

Bit	Field	Description
31-5	Reserved	Reserved
4-0	CR2GCOEFF_H	xvYCC to RGB conversion. Cr to G coefficient upper byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.98 xvYCC_2_RGB Conversion Cb_2_G Register (CB2G_COEFF_LOW)

The xvYCC_2_RGB conversion Cb_2_G register is shown in Figure 10-120 and described in Table 10-130.

Figure 10-120. xvYCC_2_RGB Conversion Cb_2_G Register (CB2G_COEFF_LOW)

31	8	7	0
Reserved		CB2GCOEFF_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-130. xvYCC_2_RGB Conversion Cb_2_G Register (CB2G_COEFF_LOW)

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CB2GCOEFF_L	xvYCC to RGB conversion. Cb to G coefficient lower byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.99 xvYCC_2_RGB Conversion Cb_2_G Register (CB2G_COEFF_UP)

The xvYCC_2_RGB conversion Cb_2_G register is shown in Figure 10-121 and described in Table 10-131.

Figure 10-121. xvYCC_2_RGB Conversion Cb_2_G Register (CB2G_COEFF_UP)

31	5	4	0
Reserved		CB2GCOEFF_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-131. xvYCC_2_RGB Conversion Cb_2_G Register (CB2G_COEFF_UP) Field Descriptions

Bit	Field	Description
31-5	Reserved	Reserved
4-0	CB2GCOEFF_H	xvYCC to RGB conversion. Cb to G coefficient upper byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.100 xvYCC_2_RGB Conversion Y Offset Register (YOFFSET1_LOW)

The xvYCC_2_RGB conversion Y offset register is shown in Figure 10-122 and described in Table 10-132.

Figure 10-122. xvYCC_2_RGB Conversion Y Offset Register (YOFFSET1_LOW)

31	8	7	0
Reserved		YOFFS1_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-132. xvYCC_2_RGB Conversion Y Offset Register (YOFFSET1_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	YOFFS1_L	xvYCC2RGB Y offset coefficient lower byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.101 xvYCC_2_RGB Conversion Y Offset Register (YOFFSET1_UP)

The xvYCC_2_RGB conversion Y offset register is shown in Figure 10-123 and described in Table 10-133.

Figure 10-123. xvYCC_2_RGB Conversion Y Offset Register (YOFFSET1_UP)

31	4	3	0
Reserved		YOFFS1_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-133. xvYCC_2_RGB Conversion Y Offset Register (YOFFSET1_UP) Field Descriptions

Bit	Field	Description
31-4	Reserved	Reserved
3-0	YOFFS1_H	xvYCC2RGB Y offset coefficient upper byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.102 xvYCC_2_RGB Conversion Offset1 Register (OFFSET1_LOW)

The xvYCC_2_RGB conversion offset1 register is shown in Figure 10-124 and described in Table 10-134.

Figure 10-124. xvYCC_2_RGB Conversion Offset1 Register (OFFSET1_LOW)

31	8	7	0
Reserved		OFFS1_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-134. xvYCC_2_RGB Conversion Offset1 Register (OFFSET1_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	OFFS1_L	xvYCC2RGB offset1 coefficient lower byte. Offset for RGB channel before right shifting, which is subtractive if software override is on (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.103 xvYCC_2_RGB Conversion Offset1 Register (OFFSET1_MID)

The xvYCC_2_RGB conversion offset1 register is shown in Figure 10-125 and described in Table 10-135.

Figure 10-125. xvYCC_2_RGB Conversion Offset1 Register (OFFSET1_MID)

31	8	7	0
Reserved		OFFS1_M	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-135. xvYCC_2_RGB Conversion Offset1 Register (OFFSET1_MID) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	OFFS1_M	xvYCC2RGB offset1 coefficient mid byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.104 xvYCC_2_RGB Conversion Offset1 Register (OFFSET1_UP)

The xvYCC_2_RGB conversion offset1 register is shown in Figure 10-126 and described in Table 10-136.

Figure 10-126. xvYCC_2_RGB Conversion Offset1 Register (OFFSET1_UP)

31	8	7	0
Reserved		OFFS1_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-136. xvYCC_2_RGB Conversion Offset1 Register (OFFSET1_UP) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	OFFS1_H	xvYCC2RGB offset1 coefficient upper byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.105 xvYCC_2_RGB Conversion Offset2 Register (OFFSET2_LOW)

The xvYCC_2_RGB conversion offset2 register is shown in Figure 10-127 and described in Table 10-137.

Figure 10-127. xvYCC_2_RGB Conversion Offset2 Register (OFFSET2_LOW)

31	8	7	0
Reserved		OFFS2_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-137. xvYCC_2_RGB Conversion Offset2 Register (OFFSET2_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	OFFS2_L	xvYCC2RGB offset2 coefficient lower byte. Offset for RGB channel before right shifting, which is subtractive if software override is on (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.106 xvYCC_2_RGB Conversion Offset2 Register (OFFSET2_UP)

The xvYCC_2_RGB conversion offset2 register is shown in Figure 10-128 and described in Table 10-138.

Figure 10-128. xvYCC_2_RGB Conversion Offset2 Register (OFFSET2_UP)

31	4	3	0
Reserved		OFFS2_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-138. xvYCC_2_RGB Conversion Offset2 Register (OFFSET2_UP) Field Descriptions

Bit	Field	Description
31-4	Reserved	Reserved
3-0	OFFS2_H	xvYCC2RGB offset2 coefficient upper byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.107 xvYCC_2_RGB Conversion DC Level Register (DCLEVEL_LOW)

The xvYCC_2_RGB conversion DC level register is shown in Figure 10-129 and described in Table 10-139.

Figure 10-129. xvYCC_2_RGB Conversion DC Level Register (DCLEVEL_LOW)

31	8	7	0
Reserved		DC_LEV_L	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-139. xvYCC_2_RGB Conversion DC Level Register (DCLEVEL_LOW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DC_LEV_L	xvYCC2RGB DC level coefficient lower byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.108 xvYCC_2_RGB Conversion DC Level Register (DCLEVEL_UP)

The xvYCC_2_RGB conversion DC level register is shown in Figure 10-130 and described in Table 10-140.

Figure 10-130. xvYCC_2_RGB Conversion DC Level Register (DCLEVEL_UP)

31	6	5	0
Reserved		DC_LEV_H	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-140. xvYCC_2_RGB Conversion DC Level Register (DCLEVEL_UP) Field Descriptions

Bit	Field	Description
31-6	Reserved	Reserved
5-0	DC_LEV_H	xvYCC2RGB DC level coefficient upper byte (override internal CSC value when SW_OVR = 1 in register XVYCC2RGB_CTL)

10.3.2.109 DDC I2C Manual Register (DDC_MAN)

The DDC I2C manual register is shown in Figure 10-131 and described in Table 10-141.

Figure 10-131. DDC I2C Manual Register (DDC_MAN)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
MAN_OVR	Reserved	MAN_SDA	MAN_SCL	Reserved		IO_SCL	IO_SDA
R/W-0	R-0h	R/W-0h	R/W-0h	R-0h		R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-141. DDC I2C Manual Register (DDC_MAN) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	MAN_OVR		Manual override of SCL and SDA output
		0	Normal operation
		1	Override port with MAN_SCL and MAN_SDA states
6	Reserved	0	Reserved
5	MAN_SDA	0	Manual SDA output
4	MAN_SCL	0	Manual SCL output
3-2	Reserved	0	Reserved
1	IO_SCL	0	DDC SCL input state
0	IO_SDA	0	DDC SDA input state

10.3.2.110 DDC I2C Target Slave Address Register (DDC_ADDR)

The DDC I2C target slave address register (DDC_ADDR) is shown in Figure 10-132 and described in Table 10-142.

Figure 10-132. DDC I2C Target Slave Address Register (DDC_ADDR)

31	8	7	1	0
Reserved		DDC_ADDR		Reserved
R-0h		R/W-0h		R-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-142. DDC I2C Target Slave Address Register (DDC_ADDR) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-1	DDC_ADDR	DDC device address
0	Reserved	Reserved

10.3.2.111 DDC I2C Target Segment Address Register (DDC_SEGM)

The DDC I2C target segment address register is shown in Figure 10-133 and described in Table 10-143.

Figure 10-133. DDC I2C Target Segment Address Register (DDC_SEGM)

31	8	7	0
Reserved		DDC_SEGM	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-143. DDC I2C Target Segment Address Register (DDC_SEGM) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DDC_SEGM	DDC segment address

10.3.2.112 DDC I2C Target Offset Address Register (DDC_OFFSET)

The DDC I2C target offset address register is shown in Figure 10-134 and described in Table 10-144.

Figure 10-134. DDC I2C Target Offset Address Register (DDC_OFFSET)

31	8	7	0
Reserved		DDC_OFFSET	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-144. DDC I2C Target Offset Address Register (DDC_OFFSET) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DDC_OFFSET	DDC offset address

10.3.2.113 DDC I2C Data Count Register (DDC_COUNT1)

The DDC I2C data count register is shown in Figure 10-135 and described in Table 10-145.

Figure 10-135. DDC I2C Data Count Register (DDC_COUNT1)

31	8	7	0
Reserved		DDC_COUNT1	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-145. DDC I2C Data Count Register (DDC_COUNT1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DDC_COUNT	The total number of bytes to be read from the slave or written to the slave before a stop bit is sent on the DDC bus. For example, if the HDCP KSV FIFO length is 635 bytes (127 devices × 5 bytes/KSV), the DDC_COUNT must be 27Bh.

10.3.2.114 DDC I2C Data Count Register (DDC_COUNT2)

The DDC I2C data count register is shown in Figure 10-136 and described in Table 10-146.

Figure 10-136. DDC I2C Data Count Register (DDC_COUNT2)

31	8	7	0
Reserved		DDC_COUNT2	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-146. DDC I2C Data Count Register (DDC_COUNT2) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DDC_COUNT	The total number of bytes to be read from the slave or written to the slave before a Stop bit is sent on the DDC bus. For example, if the HDCP KSV FIFO length is 635 bytes (127 devices × 5 bytes/KSV), the DDC_COUNT must be 27Bh.

10.3.2.115 DDC I2C Status Register (DDC_STATUS)

The DDC I2C Status Register (DDC_STATUS) is shown in Figure 10-137 and described in Table 10-147.

Figure 10-137. DDC I2C Status Register (DDC_STATUS)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	BUS_LOW	NO_ACK	IN_PROG	FIFO_FULL	FIFO_EMP	FRD_USE	FWT_USE
R-0h	R-0h	R-0h	R-0h	R-0h	R-1h	R-0h	R-0h

LEGEND: R = Read only; -*n* = value after reset

Table 10-147. DDC I2C Status Register (DDC_STATUS) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6	BUS_LOW		I2C bus is LOW
		0	I2C bus is not pulled low by an external device
		1	I2C transaction did not start because I2C bus is pulled LOW by an external device
5	NO_ACK		HDMI transmitter did not receive an ACK
		0	HDMI transmitter did receive an ACK
		1	HDMI transmitter did not receive an ACK from slave device during address or data write
4	IN_PROG		DDC operation in progress
		0	DDC operation is not in progress
		1	DDC operation in progress
3	FIFO_FULL		DDC FIFO full
		0	DDC FIFO is not full
		1	DDC FIFO is full
2	FIFO_EMP		DDC FIFO empty
		0	DDC FIFO is not empty
		1	DDC FIFO is empty
1	FRD_USE		DDC FIFO read in use
		0	DDC FIFO read is not in use
		1	DDC FIFO read is in use
0	FWT_USE		DDC FIFO write in use
		0	DDC FIFO write is not in use
		1	DDC FIFO write is in use

10.3.2.116 DDC I2C Command Register (DDC_CMD)

The DDC I2C command register is shown in Figure 10-138 and described in Table 10-148.

Figure 10-138. DDC I2C Command Register (DDC_CMD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		DDC_FLT_EN	SDA_DEL_EN	DDC_CMD			
R-0h		R/W-0h	R/W-0h	R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-148. DDC I2C Command Register (DDC_CMD) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5	DDC_FLT_EN		Enable the DCC delay. A DDC delay is inserted into the SDA line to create a 300 ns delay for the falling edge of the DDC SDA signal to avoid an erroneous I2C START condition. The real start condition must have a setup time of 600 ns so that this delay of 300 ns does not remove the real START condition. Filtering is done using a ring oscillator.
		0	Enable
		1	Disable
4	SDA_DEL_EN		Enable 3 ns glitch filtering on the DDC clock and data line. Filtering is done using a ring oscillator.
		0	Enable
		1	Disable
3-0	DDC_CMD		DDC command Writing to this register immediately initiates the I2C transaction on the DDC bus. Note: The clear FIFO command resets the FIFO read and write pointers to zero. Data formerly loaded into the FIFO cannot be re-read after a clear FIFO, as the FIFO will be empty. Other command codes are reserved and may cause the DDC bus to hang if used. The clock SCL command resets any I2C devices on the DDC lines. This should be initiated once before initiating the DDC commands.
		0	Current address read with no ACK on last byte
		2h	Sequential read with no ACK on last byte
		4h	Enhanced DDC read with no ACK on last byte
		6h	Sequential write ignoring ACK on last byte
		7h	Sequential write requiring ACK on last byte
		9h	Clear FIFO
		Ah	Clock SCL
		Fh	Abort transaction
			All other values are reserved.

10.3.2.117 DDC I2C Data Register (DDC_DATA)

The DDC I2C data register is shown in Figure 10-139 and described in Table 10-149.

Figure 10-139. DDC I2C Data Register (DDC_DATA)

31	8	7	0
Reserved		DDC_DATA	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-149. DDC I2C Data Register (DDC_DATA) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	DDC_DATA	DDC data input

10.3.2.118 DDC I2C FIFO Count Register (DDC_FIFOCNT)

The DDC I2C FIFO count register is shown in Figure 10-140 and described in Table 10-150.

Figure 10-140. DDC I2C FIFO Count Register (DDC_FIFOCNT)

31	5	4	0
Reserved		DDC_FIFOCNT	
R-0h		R-0h	

LEGEND: R = Read only; -*n* = value after reset

Table 10-150. DDC I2C FIFO Count Register (DDC_FIFOCNT) Field Descriptions

Bit	Field	Description
31-5	Reserved	Reserved
4-0	DDC_FIFOCNT	FIFO data byte count (the number of bytes in the FIFO). The DDC FIFO size is 16. The maximum value for DDC_FIFOCNT is 10h.

10.3.2.119 ROM Status Register (EPST)

The ROM status register is shown in Figure 10-141 and described in Table 10-151.

Figure 10-141. ROM Status Register (EPST)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	BIST2_ERR	BIST1_ERR	Reserved			CRC_ERR	CMDD
R-0h	R/W-0h	R/W-0h	R-0h			R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-151. ROM Status Register (EPST) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6	BIST2_ERR		BIST self-authentication test 2 error
		0	No error
		1	BIST self-authentication test 2 error
5	BIST1_ERR		BIST self-authentication test 1 error
		0	No error
		1	BIST self-authentication test 1 error
4-2	Reserved	0	Reserved
1	CRC_ERR		CRC error
		0	No error
		1	CRC error
0	CMDD		Command done
		0	After reset. Set to 1 once the KSV keys are read (when osclk is running).
		1	Command done (last operation completed successfully).

10.3.2.120 ROM Command Register (EPCM)

The ROM command register is shown in Figure 10-142 and described in Table 10-152.

Figure 10-142. ROM Command Register (EPCM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		LD_KSV	EPCM				
R-0h		R/W-0h	R/W-0h				

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-152. ROM Command Register (EPCM) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5	LD_KSV		Load KSV enable
		0	Write 0 before enabling again
		1	Enable loading of KSV from OTP
4-0	EPCM		Command. Before writing a new value into this register, verify that the previous command is complete by checking the CMDD bit in the EPST register.
		3h	Run all BIST tests
		4h	Run only CRC test
		8h	Run only BIST self authentication test 1 (16-bit CRC to verify OTP contents)
		10h	Run only BIST self authentication test 2 (2 pass authentication to verify the HDCP cipher engine)
			All other values are reserved.

10.3.3 HDMI IP Core Gamut Registers

Table 10-153 lists the HDMI IP core gamut registers.

Table 10-153. HDMI IP Core Gamut Registers

Address Offset	Acronym	Register Name
00h	GAMUT_HEADER1	Gamut Metadata Register
04h	GAMUT_HEADER2	Gamut Metadata Register
08h	GAMUT_HEADER3	Gamut Metadata Register
0Ch-78h	GAMUT_DBYTE__0 - GAMUT_DBYTE__27	Gamut Metadata Registers

10.3.3.1 Gamut Metadata Register (GAMUT_HEADER1)

The gamut metadata register is shown in Figure 10-143 and described in Table 10-154.

Figure 10-143. Gamut Metadata Register (GAMUT_HEADER1)

31	8	7	0
Reserved		HEADER1	
R-0h		R/W-Ah	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

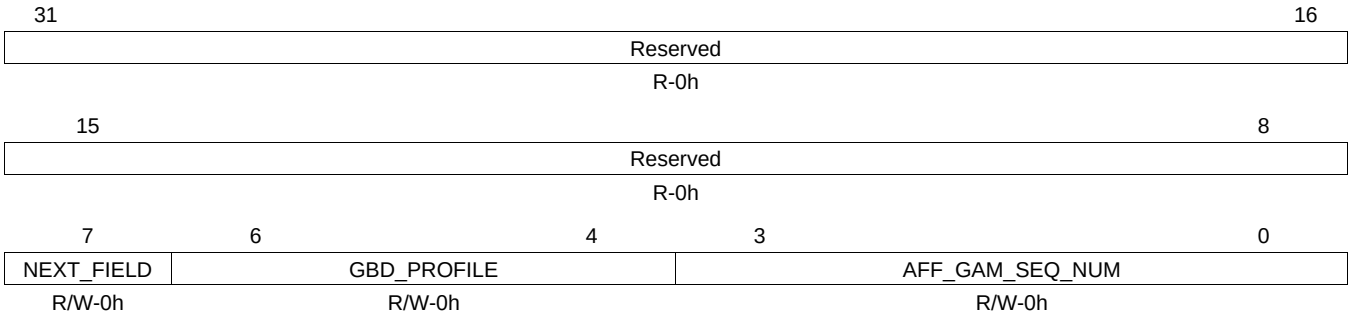
Table 10-154. Gamut Metadata Register (GAMUT_HEADER1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	HEADER1	Gamut metadata header information

10.3.3.2 Gamut Metadata Register (GAMUT_HEADER2)

The gamut metadata register is shown in Figure 10-144 and described in Table 10-155.

Figure 10-144. Gamut Metadata Register (GAMUT_HEADER2)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

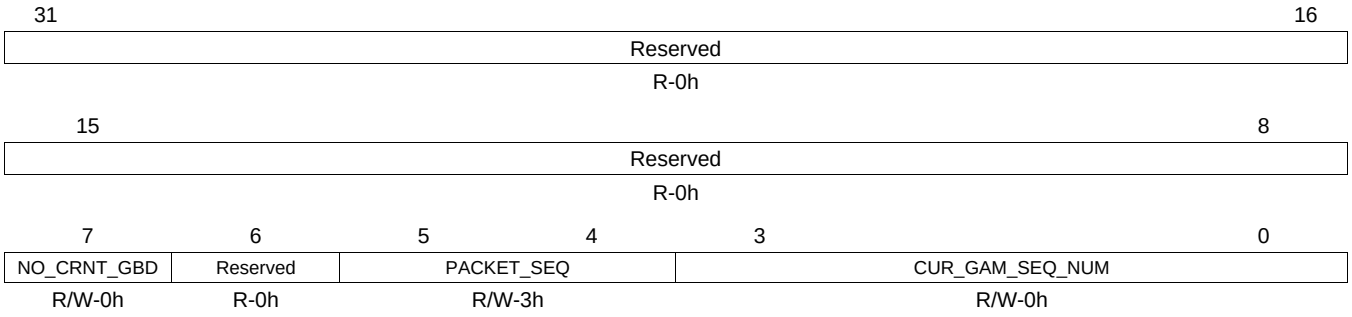
Table 10-155. Gamut Metadata Register (GAMUT_HEADER2) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	NEXT_FIELD	0	Indicates that the GBD will be effective on the next video field. NEXT_FIELD should be set even if the GBD is already effective (for example, Current = Affected)
6-4	GBD_PROFILE		Transmission profile number. Values from 4h-7h are reserved.
		0	P0
		1h	P1
		2h	P2
		3h	P3
3-0	AFF_GAM_SEQ_NUM	0	Indicates which video fields are relevant for this metadata.

10.3.3.3 Gamut Metadata Register (GAMUT_HEADER3)

The gamut metadata register is shown in Figure 10-145 and described in Table 10-156.

Figure 10-145. Gamut Metadata Register (GAMUT_HEADER3)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

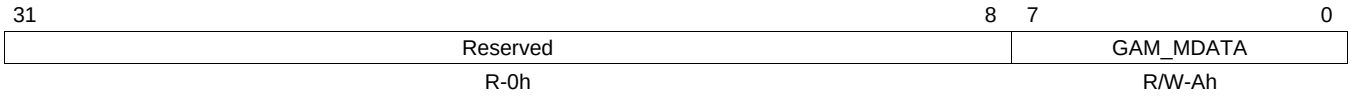
Table 10-156. Gamut Metadata Register (GAMUT_HEADER3) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	NO_CRNT_GBD	0	Indicates no gamut metadata available for currently transmitted video. When set to 1, CUR_GAM_SEQ_NUM shall be ignored by the sink.
6	Reserved	0	Reserved
5-4	PACKET_SEQ		Indicates the position of current packet.
		0	Intermediate packet
		1h	First packet
		2h	Last packet
		3h	Only packet in sequence
3-0	CUR_GAM_SEQ_NUM	0	Indicates the gamut number of the currently transmitted video stream.

10.3.3.4 Gamut Metadata Registers (GAMUT_DBYTE__0-GAMUT_DBYTE__27)

The gamut metadata registers are shown in Figure 10-146 and described in Table 10-157.

Figure 10-146. Gamut Metadata Registers (GAMUT_DBYTE__0-GAMUT_DBYTE__27)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-157. Gamut Metadata Registers (GAMUT_DBYTE__0-GAMUT_DBYTE__27) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	GAM_MDATA	Gamut metadata data bytes

10.3.4 HDMI IP Core Audio Video Registers

Table 10-158 lists the HDMI IP core audio video registers.

Table 10-158. HDMI IP Core Audio Video Registers

Address Offset	Acronym	Register Name
04h	ACR_CTRL	ACR Control Register
08h	FREQ_SVAL	ACR Audio Frequency Register
0Ch	N_SVAL1	ACR N Software Value Register
10h	N_SVAL2	ACR N Software Value Register
14h	N_SVAL3	ACR N Software Value Register
18h	CTS_SVAL1	ACR CTS Software Value Register
1Ch	CTS_SVAL2	ACR CTS Software Value Register
20h	CTS_SVAL3	ACR CTS Software Value Register
24h	CTS_HVAL1	ACR CTS Hardware Value Register
28h	CTS_HVAL2	ACR CTS Hardware Value Register
2Ch	CTS_HVAL3	ACR CTS Hardware Value Register
50h	AUD_MODE	Audio In Mode Register
54h	SPDIF_CTRL	Audio In S/PDIF Control Register
60h	HW_SPDIF_FS	Audio In S/PDIF Extracted Fs and Length Register
64h	SWAP_I2S	Audio In I2S Channel Swap Register
6Ch	SPDIF_ERTH	Audio Error Threshold Register
70h	I2S_IN_MAP	Audio In I2S Data In Map Register
74h	I2S_IN_CTRL	Audio In I2S Control Register
78h	I2S_CHST0	Audio In I2S Channel Status Registers
7Ch	I2S_CHST1	Audio In I2S Channel Status Registers
80h	I2S_CHST2	Audio In I2S Channel Status Registers
84h	I2S_CHST4	Audio In I2S Channel Status Registers
88h	I2S_CHST5	Audio In I2S Channel Status Registers
8Ch	ASRC	Audio Sample Rate Conversion Register
90h	I2S_IN_LEN	Audio I2S Input Length Register
BCh	HDMI_CTRL	HDMI Control Register
C0h	AUDO_TXSTAT	Audio Path Status Register
CCh	AUD_PAR_BUSCLK_1	Audio Input Data Rate Adjustment Register
D0h	AUD_PAR_BUSCLK_2	Audio Input Data Rate Adjustment Register
D4h	AUD_PAR_BUSCLK_3	Audio Input Data Rate Adjustment Register
F0h	TEST_TXCTRL	Test Control Register
F4h	DPD	Diagnostic Power Down Register
F8h	PB_CTRL1	Packet Buffer Control 1 Register
FCh	PB_CTRL2	Packet Buffer Control 2 Register
100h	AVI_TYPE	Packet Registers
104h	AVI_VERS	Packet Registers
108h	AVI_LEN	Packet Registers
10Ch	AVI_CHSUM	Packet Registers
110h-148h	AVI_DBYTE_0 - AVI_DBYTE_14	Packet Registers
180h	SPD_TYPE	SPD InfoFrame Registers
184h	SPD_VERS	SPD InfoFrame Registers
188h	SPD_LEN	SPD InfoFrame Registers
18Ch	SPD_CHSUM	SPD InfoFrame Registers
190h-1F8h	SPD_DBYTE_0 - SPD_DBYTE_26	SPD InfoFrame Registers
200h	AUDIO_TYPE	Audio InfoFrame Registers

Table 10-158. HDMI IP Core Audio Video Registers (continued)

Address Offset	Acronym	Register Name
204h	AUDIO_VERS	Audio InfoFrame Registers
208h	AUDIO_LEN	Audio InfoFrame Registers
20Ch	AUDIO_CHSUM	Audio InfoFrame Registers
210h-234h	AUDIO_DBYTE_0 - AUDIO_DBYTE_9	Audio InfoFrame Registers
280h	MPEG_TYPE	MPEG InfoFrame Registers
284h	MPEG_VERS	MPEG InfoFrame Registers
288h	MPEG_LEN	MPEG InfoFrame Registers
28Ch	MPEG_CHSUM	MPEG InfoFrame Registers
290h-2F8h	MPEG_DBYTE_0 - MPEG_DBYTE_26	MPEG InfoFrame Registers
300h-378h	GEN_DBYTE_0 - GEN_DBYTE_30	Generic Packet Registers
37Ch	CP_BYTE1	General Control Packet Register
380h-3F8h	GEN2_DBYTE_0 - GEN2_DBYTE_30	Generic Packet 2 Registers
3FCh	CEC_ADDR_ID	CEC Slave ID Register

10.3.4.1 ACR Control Register (ACR_CTRL)

The ACR control register is shown in Figure 10-147 and described in Table 10-159.

Figure 10-147. ACR Control Register (ACR_CTRL)

31	16
Reserved	
R-0h	

15	2	1	0
Reserved		NCTSPKT_EN	CTS_SEL
R-0h		R/W-1h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-159. ACR Control Register (ACR_CTRL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	NCTSPKT_EN		CTS request enable
		0	N/CTS packet is disabled
		1	N/CTS packet is enabled
0	CTS_SEL		CTS source select
		0	Send hardware-updated CTS value in N/CTS packet (recommended)
		1	Send software-updated CTS value in N/CTS packet (for diagnostic use)

10.3.4.2 ACR Audio Frequency Register (FREQ_SVAL)

The ACR audio frequency register is shown in Figure 10-148 and described in Table 10-160.

Figure 10-148. ACR Audio Frequency Register (FREQ_SVAL)

31	16
Reserved	
R-0h	

15	3	2	0
Reserved		MCLK_CONF	
R-0h		R/W-1h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-160. ACR Audio Frequency Register (FREQ_SVAL) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	MCLK_CONF		MCLK input mode. The HDMI transmitter uses these bits to divide the MCLK input to produce CTS values according to the 128 × Fs formula. The MCLK to Fs ratio is for the input Fs, not the down-sampled output Fs (see Section 10.3.4.25).
		0	MCLK is 128 × Fs
		1h	MCLK is 256 × Fs
		2h	MCLK is 384 × Fs
		3h	MCLK is 512 × Fs
		4h	MCLK is 768 × Fs
		5h	MCLK is 1024 × Fs
		6h	MCLK is 1152 × Fs
		7h	MCLK is 192 × Fs

10.3.4.3 ACR N Software Value Register 1 (N_SVAL1)

The ACR N software value register is shown in Figure 10-149 and described in Table 10-161.

Figure 10-149. ACR N Software Value Register 1 (N_SVAL1)

31	8	7	0
Reserved		N_SVAL	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-161. ACR N Software Value Register 1 (N_SVAL1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	N_SVAL	N value for audio clock regeneration method, a 20-bit value. This must be written to the registers to create the correct divisor for audio clock regeneration. Only values greater than 0 are valid. This register must be written after a hardware reset.

10.3.4.4 ACR N Software Value Register 2 (N_SVAL2)

The ACR N software value register is shown in Figure 10-150 and described in Table 10-162.

Figure 10-150. ACR N Software Value Register 2 (N_SVAL2)

31	8	7	0
Reserved		N_SVAL	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-162. ACR N Software Value Register 2 (N_SVAL2) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	N_SVAL	N value for audio clock regeneration method, a 20-bit value. This must be written to the registers to create the correct divisor for audio clock regeneration. Only values greater than 0 are valid. This register must be written after a hardware reset.

10.3.4.5 ACR N Software Value Register 3 (N_SVAL3)

Figure 10-151. ACR N Software Value Register 3 (N_SVAL3)

31	4	3	0
Reserved		N_SVAL	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-163. ACR N Software Value Register 3 (N_SVAL3) Field Descriptions

Bit	Field	Description
31-4	Reserved	Reserved
3-0	N_SVAL	N Value for audio clock regeneration method; a 20-bit value. This must be written to the registers to create the correct divisor for audio clock regeneration. Only values greater than 0 are valid. This register must be written after a hardware reset.

10.3.4.6 ACR CTS Software Value Register 1 (CTS_SVAL1)

Figure 10-152. ACR CTS Software Value Register 1 (CTS_SVAL1)

31	8	7	0
Reserved		CTS_SVAL	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-164. ACR CTS Software Value Register 1 (CTS_SVAL1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CTS_SVAL	CTS Value for audio clock regeneration method; a 20-bit value. Diagnostic use and applied only when the CTS_SEL bit (in register ACR_CTRL) is set to 1.

10.3.4.7 ACR CTS Software Value Register 2 (CTS_SVAL2)

Figure 10-153. ACR CTS Software Value Register 2 (CTS_SVAL2)

31	8	7	0
Reserved		CTS_SVAL	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-165. ACR CTS Software Value Register 2 (CTS_SVAL2) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CTS_SVAL	CTS Value for audio clock regeneration method; a 20-bit value. Diagnostic use and applied only when the CTS_SEL bit (in register ACR_CTRL) is set to 1.

10.3.4.8 ACR CTS Software Value Register 3 (CTS_SVAL3)

Figure 10-154. ACR CTS Software Value Register 3 (CTS_SVAL3)

31	4	3	0
Reserved		CTS_SVAL	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-166. ACR CTS Software Value Register 3 (CTS_SVAL3) Field Descriptions

Bit	Field	Description
31-4	Reserved	Reserved
3-0	CTS_SVAL	CTS Value for audio clock regeneration method; a 20-bit value. Diagnostic use and applied only when the CTS_SEL bit (in register ACR_CTRL) is set to 1.

10.3.4.9 ACR CTS Hardware Value Register 1 (CTS_HVAL1)

Figure 10-155. ACR CTS Hardware Value Register 1 (CTS_HVAL1)

31	8	7	0
Reserved		CTS_HVAL	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-167. ACR CTS Hardware Value Register 1 (CTS_HVAL1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CTS_HVAL	CTS Value for audio clock regeneration method; a 20-bit value. This value is measured and stored here by the hardware when MCLK is active and N is valid.

10.3.4.10 ACR CTS Hardware Value Register 2 (CTS_HVAL2)

Figure 10-156. ACR CTS Hardware Value Register 2 (CTS_HVAL2)

31	8	7	0
Reserved		CTS_HVAL	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-168. ACR CTS Hardware Value Register 2 (CTS_HVAL2) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CTS_HVAL	CTS Value for audio clock regeneration method; a 20-bit value. This value is measured and stored here by the hardware when MCLK is active and N is valid.

10.3.4.11 ACR CTS Hardware Value Register 3 (CTS_HVAL3)

Figure 10-157. ACR CTS Hardware Value Register 3 (CTS_HVAL3)

31	4	3	0
Reserved		CTS_HVAL	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-169. ACR CTS Hardware Value Register 3 (CTS_HVAL3) Field Descriptions

Bit	Field	Description
31-4	Reserved	Reserved
3-0	CTS_HVAL	CTS Value for audio clock regeneration method; a 20-bit value. This value is measured and stored here by the hardware when MCLK is active and N is valid.

10.3.4.12 Audio In Mode Register (AUD_MODE)

Figure 10-158. Audio In Mode Register (AUD_MODE)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
SD3_EN	SD2_EN	SD1_EN	SD0_EN	DSD_EN	AUD_PAR_EN	SPDIF_EN	AUD_EN
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-170. Audio In Mode Register (AUD_MODE) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	SD3_EN		I2S input channel 3
		0	Disable
		1	Enable
6	SD2_EN		I2S input channel 2
		0	Disable
		1	Enable
5	SD1_EN		I2S input channel 1
		0	Disable
		1	Enable
4	SD0_EN		I2S input channel 0
		0	Disable
		1	Enable
3	DSD_EN		Direct Stream Digital Audio enable
		0	Disable
		1	Enable
2	AUD_PAR_EN		Parallel audio enable
		0	Disable
		1	Enable
1	SPDIF_EN		S/PDIF input stream enable
		0	Disable
		1	Enable
0	AUD_EN		Audio input stream enable
		0	Disable
		1	Enable

10.3.4.13 Audio In S/PDIF Control Register (SPDIF_CTRL)

Figure 10-159. Audio In S/PDIF Control Register (SPDIF_CTRL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved				NOAUDIO	Reserved	FS_OVERRIDE	Reserved
R-0h				R-0h	R-0h	R/W-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-171. Audio In S/PDIF Control Register (SPDIF_CTRL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	NOAUDIO		No S/PDIF audio
		0	Detected change on the S/PDIF input
		1	No change detected on the S/PDIF input
2	Reserved	0	Reserved
1	FS_OVERRIDE		S/PDIF input stream override
		0	Use input S/PDIF stream s detected FS
		1	Use software FS in I2S_CHST4 register
0	Reserved	0	Reserved

10.3.4.14 Audio In S/PDIF Extracted Fs and Length Register (HW_SPDIF_FS)

Figure 10-160. Audio In S/PDIF Extracted Fs and Length Register (HW_SPDIF_FS)

31				16
Reserved				
R-0h				
15				8
Reserved				
R-0h				
7	5	4	3	0
HW_SPDIF_LEN		HW_MAXLEN		HW_SPDIF_FS
R-0h		R-0h		R-0h

LEGEND: R = Read only; -*n* = value after reset

Table 10-172. Audio In S/PDIF Extracted Fs and Length Register (HW_SPDIF_FS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-5	HW_SPDIF_LEN		Channel status bits 33 to 35 (bit 33 = LSB). Combines with HW_MAXLEN bit to indicate sample size.
			When HW_MAXLEN = 0 (20 bits):
		0	Reserved
		1h	16 bits
		2h	18 bits
		3h	Reserved
		4h	19 bits
		5h	20 bits
		6h	17 bits
		7h	Reserved
			When HW_MAXLEN = 1 (24 bits):
		0	Reserved
		1h	20 bits
		2h	22 bits
		3h	Reserved
		4h	23 bits
		5h	24 bits
		6h	21 bits
		7h	Reserved
4	HW_MAXLEN		Maximum sample length (channel status bit 32)
		0	20 bits
		1	24 bits
3-0	HW_SPDIF_FS	0-Fh	Set to the FS extracted from the S/PDIF input channel status bits 24-27.

10.3.4.15 Audio In I2S Channel Swap Register (SWAP_I2S)

Figure 10-161. Audio In I2S Channel Swap Register (SWAP_I2S)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
SWCH3	SWCH2	SWCH1	SWCH0	RSVDRW			
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-9h			

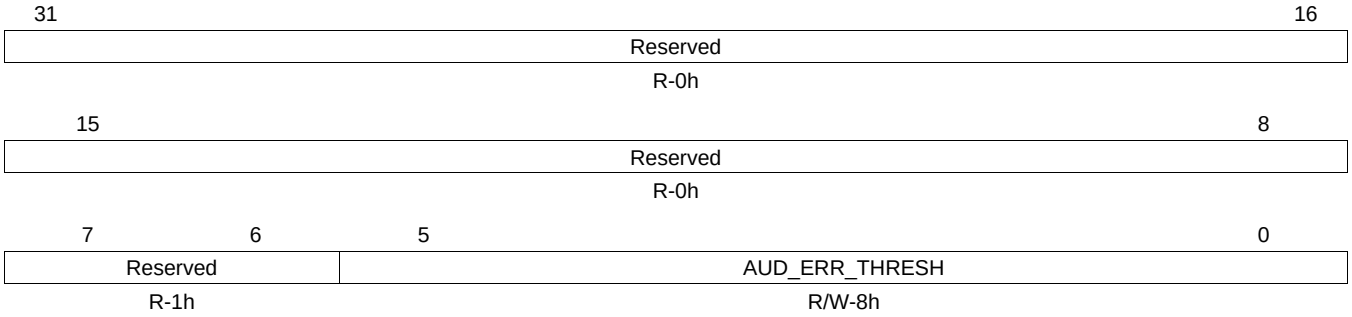
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-173. Audio In I2S Channel Swap Register (SWAP_I2S) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	SWCH3		Swap left-right channels for I2S Channel 3
		0	Do not swap left and right
		1	Swap left and right
6	SWCH2		Swap left-right channels for I2S Channel 2
		0	Do not swap left and right
		1	Swap left and right
5	SWCH1		Swap left-right channels for I2S Channel 1
		0	Do not swap left and right
		1	Swap left and right
4	SWCH0		Swap left-right channels for I2S Channel 0
		0	Do not swap left and right
		1	Swap left and right
3-0	RSVDRW	9h	Reserved. Do not modify.

10.3.4.16 Audio Error Threshold Register (SPDIF_ERTH)

Figure 10-162. Audio Error Threshold Register (SPDIF_ERTH)



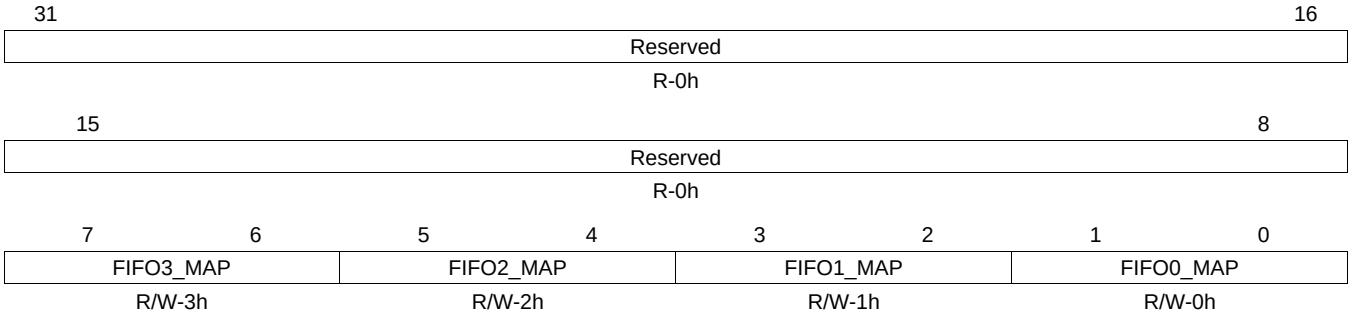
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-174. Audio Error Threshold Register (SPDIF_ERTH) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	Reserved	1	Reserved
5-0	AUD_ERR_THRESH	8h	Specifies the error threshold level. The frame is marked as invalid if the number of bi-phase mark encoding errors in the audio stream exceeds this threshold level during frame decoding.

10.3.4.17 Audio In I2S Data In Map Register (I2S_IN_MAP)

Figure 10-163. Audio In I2S Data In Map Register (I2S_IN_MAP)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-175. Audio In I2S Data In Map Register (I2S_IN_MAP) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	FIFO3_MAP		Channel map to FIFO3 (for HDMI Layout 1).
		0	Map SD0 to FIFO3
		1h	Map SD1 to FIFO3
		2h	Map SD2 to FIFO3
		3h	Map SD3 to FIFO3
5-4	FIFO2_MAP		Channel map to FIFO2 (for HDMI Layout 1).
		0	Map SD0 to FIFO2
		1h	Map SD1 to FIFO2
		2h	Map SD2 to FIFO2
		3h	Map SD3 to FIFO2
3-2	FIFO1_MAP		Channel map to FIFO1 (for HDMI Layout 1).
		0	Map SD0 to FIFO1
		1h	Map SD1 to FIFO1
		2h	Map SD2 to FIFO1
		3h	Map SD3 to FIFO1
1-0	FIFO0_MAP		Channel map to FIFO0 (for HDMI Layout 1).
		0	Map SD0 to FIFO0
		1h	Map SD1 to FIFO0
		2h	Map SD2 to FIFO0
		3h	Map SD3 to FIFO0

10.3.4.18 Audio In I2S Control Register (I2S_IN_CTRL)

Figure 10-164. Audio In I2S Control Register (I2S_IN_CTRL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
HBRA_ON	SCK_EDGE	CBIT_ORDER	VBIT	I2S_WS	I2S_JUST	I2S_DIR	I2S_SHIFT
R/W-0h	R/W-1h	R/W-0h	R/W-0h	R/W-0h	R/W-1h	R/W-0h	R/W-1h

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-176. Audio In I2S Control Register (I2S_IN_CTRL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	HBRA_ON		High Bit Rate Audio On. All of the I2S control bits will apply to the control of the High Bit Rate Audio.
		0	Input stream is not high bit rate
		1	Input stream is high bit rate
6	SCK_EDGE		SCK sample edge
		0	Sample edge is falling; SD3-SD0 and WS source should change state on the rising edge of SCK
		1	Sample clock is rising; SD3-SD0 and WS source should change state on the falling edge of SCK
5	CBIT_ORDER	0	This bit should be set to 1 for High Bit Rate Audio.
4	VBIT		V bit value
		0	PCM
		1	Compressed
3	I2S_WS		WS polarity
		0	Left polarity when WS is LOW
		1	Left polarity when WS is HIGH
2	I2S_JUST		SD justify
		0	Data is left-justified
		1	Data is right-justified
1	I2S_DIR		SD direction
		0	MSB shifted first
		1	LSB shifted first
0	I2S_SHIFT		WS to SD first bit shift
		0	First bit shift (refer to the Philips Specification)
		1	No shift

10.3.4.19 Audio In I2S Channel Status Register 0 (I2S_CHST0)

Figure 10-165. Audio In I2S Channel Status Register 0 (I2S_CHST0)

31	8	7	0
Reserved		I2S_CHST0	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-177. I2S_CHST0 Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	I2S_CHST0	Channel Status Byte 0

10.3.4.20 Audio In I2S Channel Status Register 1 (I2S_CHST1)

Figure 10-166. Audio In I2S Channel Status Register 0 (I2S_CHST1)

31	8	7	0
Reserved		I2S_CHST1	
R-0h		R/W-0h	

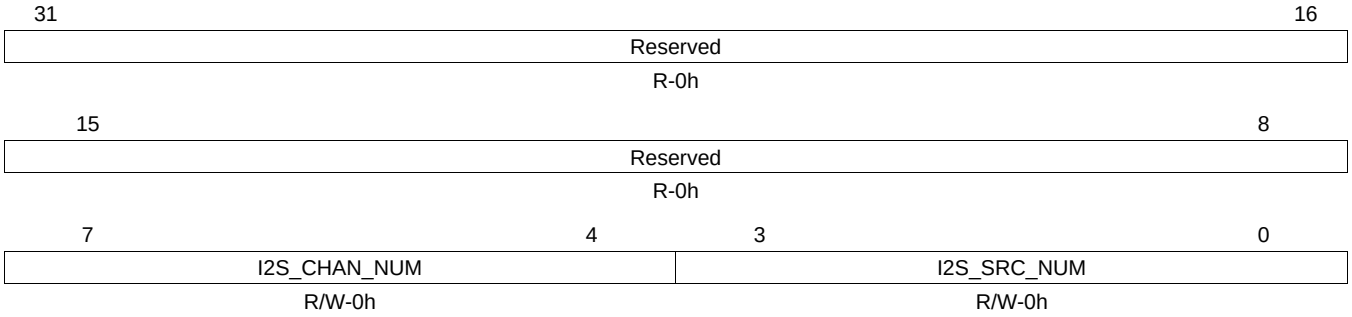
LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-178. I2S_CHST1 Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	I2S_CHST1	Channel Status Byte 1: Category Code

10.3.4.21 Audio In I2S Channel Status Register 2 (I2S_CHST2)

Figure 10-167. Audio In I2S Channel Status Register 2 (I2S_CHST2)



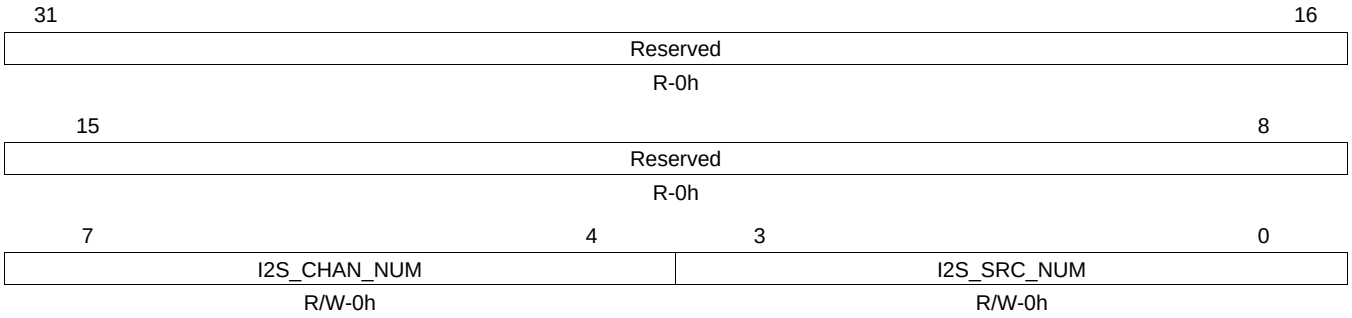
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-179. Audio In I2S Channel Status Register 2 (I2S_CHST2) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-4	I2S_CHAN_NUM	Channel Status Byte 2: Source Number
3-0	I2S_SRC_NUM	Channel Status Byte 2: Source Number

10.3.4.22 Audio In I2S Channel Status Register 3 (I2S_CHST3)

Figure 10-168. Audio In I2S Channel Status Register 3 (I2S_CHST3)



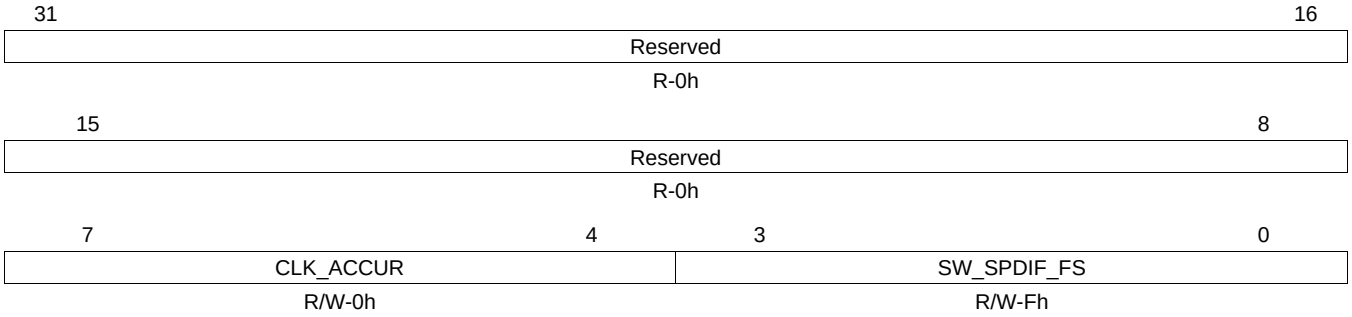
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-180. Audio In I2S Channel Status Register 3 (I2S_CHST3) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-4	I2S_CHAN_NUM	Channel Status Byte 2: Source Number
3-0	I2S_SRC_NUM	Channel Status Byte 2: Source Number

10.3.4.23 Audio In I2S Channel Status Register 4 (I2S_CHST4)

Figure 10-169. Audio In I2S Channel Status Register 4 (I2S_CHST4)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-181. Audio In I2S Channel Status Register 4 (I2S_CHST4) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-4	CLK_ACCUR	Clock Accuracy
3-0	SW_SPDIF_FS	Sampling frequency as set by software

10.3.4.24 Audio In I2S Channel Status Register 5 (I2S_CHST5)

Figure 10-170. Audio In I2S Channel Status Register 5 (I2S_CHST5)

31	16
Reserved	
R-0h	

15	8
Reserved	
R-0h	

7	4	3	1	0
FS_ORIG		I2S_LEN		I2S_MAXLEN
R/W-0h		R/W-0h		R/W-1h

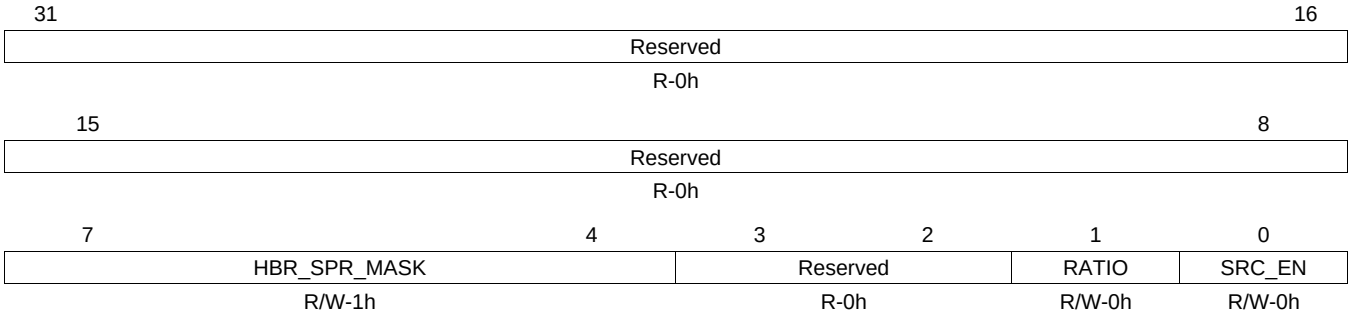
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-182. Audio In I2S Channel Status Register 5 (I2S_CHST5) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-4	FS_ORIG	0	Original Fs
3-1	I2S_LEN		Audio sample word length. Defined in bits with I2S_MAXLEN.
			When I2S_MAXLEN = 0 (20 bits):
		0	Reserved
		1h	16 bits
		2h	18 bits
		3h	Reserved
		4h	19 bits
		5h	20 bits
		6h	17 bits
		7h	Reserved
			When I2S_MAXLEN = 1 (24 bits):
		0	Reserved
		1h	20 bits
		2h	22 bits
		3h	Reserved
		4h	23 bits
		5h	24 bits
		6h	21 bits
		7h	Reserved
0	I2S_MAXLEN		Maximum audio sample word length.
		0	20 bits
		1	24 bits

10.3.4.25 Audio Sample Rate Conversion Register (ASRC)

Figure 10-171. Audio Sample Rate Conversion Register (ASRC)



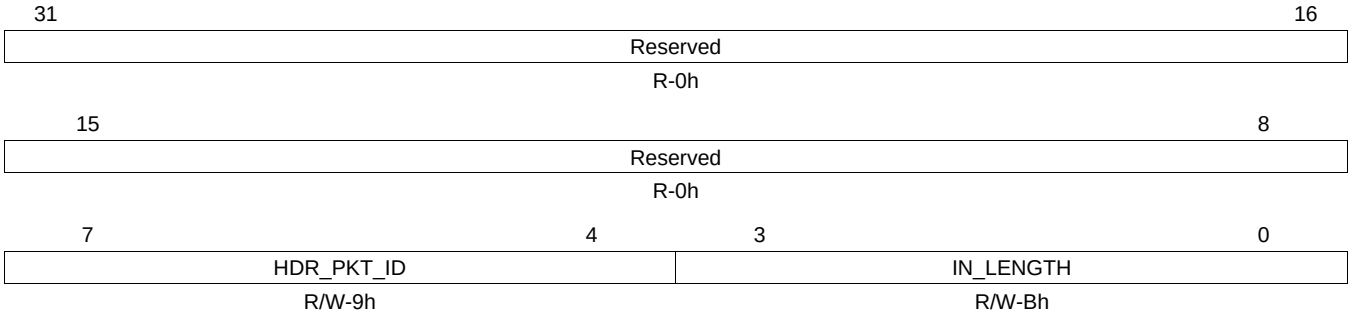
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-183. Audio Sample Rate Conversion Register (ASRC) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-4	HBR_SPR_MASK		Mask for the sample present and flat bit of the High Bit Rate Audio header. Each bit masks out one of the subpacket sample present bits.
		0	Mask out
		1	Unmask bits 7-4 must be programmed to 0 when HBRA mode is selected
3-2	Reserved	0	Reserved
1	RATIO		Sample rate down-conversion ratio when the SRC_EN bit is set to 1.
		0	Down-sample is 2-to-1
		1	Down-sample is 4-to-1
0	SRC_EN		Audio sample rate conversion
		0	Disable
		1	Enable

10.3.4.26 Audio I2S Input Length Register (I2S_IN_LEN)

Figure 10-172. Audio I2S Input Length Register (I2S_IN_LEN)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-184. Audio I2S Input Length Register (I2S_IN_LEN) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-4	HDR_PKT_ID	9h	The ID of the High Bit Rate Audio packet header
3-0	IN_LENGTH		Number of valid bits in the input I2S stream. Used for the extraction of the I2S data from the input stream.
		0-1h	Reserved
		2h	16 bits
		3h	Reserved
		4h	18 bits
		5h	22 bits
		6h-7h	Reserved
		8h	19 bits
		9h	23 bits
		Ah	20 bits
		Bh	24 bits
		Ch	17 bits
		Dh	21 bits
		Eh-Fh	Reserved

10.3.4.27 HDMI Control Register (HDMI_CTRL)

Figure 10-173. HDMI Control Register (HDMI_CTRL)

31	16
Reserved	
R-0h	

15	8
Reserved	
R-0h	

7	6	5	3	2	1	0
Reserved	DC_EN	PACKET_MODE		LAYOUT		HDMI_MODE
R-0h	R/W-0h	R/W-0h		R/W-0h		R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-185. HDMI Control Register (HDMI_CTRL) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6	DC_EN		Deep-color packet enable.
			The following data is sent in data byte 1 of the packet: 7 = Reserved, 6 = PP2, 5 = PP1, 4 = PP0, 3 = PP_valid, 2 = CD2, 1 = CD1, 0 = CD0, The CD bits indicate deep-color mode (defined in the same register bits 5:3). The PP bits indicate the fragment s phase related information that comes from the hardware state machine.
		0	Do not send deep-color related information in the packet to the HDMI Receiver.
		1	Send deep-color related information in the packet to the HDMI Receiver.
5-3	PACKET_MODE		Specifies the number of bits per pixel sent to the paketizer. The firmware must program 24 bits per pixel (8 bits per pixel; no packing) for initialization.
		0-3h	Reserved
		4h	24 bits per pixel (8 bits per pixel; no packing)
		5h	30 bits per pixel (10 bits per pixel pack to 8 bits)
		6h	36 bits per pixel (12 bits per pixel pack to 8 bits)
		7h	48 bits per pixel (16 bits per pixel; no packing)
2-1	LAYOUT		Audio packet header layout indicator.
		0	Layout 0 (2-channel)
		1h	Layout 1 (up to 8 channels)
		2h-3h	Reserved
0	HDMI_MODE		HDMI Mode
		0	Disable
		1	Enable

10.3.4.28 Audio Path Status Register (AUDO_TXSTAT)

NOTE: MUTE is not set immediately when the SETAVM bit in register CP_BYTE1 is written. After writing SETAVM to 1, MUTE is set after the Control Packet is transmitted in HDMI mode. In DVI mode, MUTE is set at the start of VSYNC. MUTE is cleared when a Control Packet with CLRAVM = 1 is sent, or (in DVI mode) at the start of VSYNC after CLRAVM has been written to 1. Note that the combinations {SETAVM, CLRAVM} = {0,0} and {1,1} are not supported by HDMI. Such a packet is not compliant.

Figure 10-174. Audio Path Status Register (AUDO_TXSTAT)

31	16
Reserved	
R-0h	

15	8
Reserved	
R-0h	

7	3	2	1	0
Reserved		MUTE	NPACKET_EN_VS_HIGH	NPACKET_EN
R-0h		R-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-186. Audio Path Status Register (AUDO_TXSTAT) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	MUTE		General Control Packet mute status
		0	No packet with SETAVM = 1 has been sent
		1	A packet with SETAVM = 1 has been sent
1	NPACKET_EN_VS_HIGH	0	Enables null packet flooding only when VSync is high.
0	NPACKET_EN	0	Enables null packet flooding all the time.

10.3.4.29 Audio Input Data Rate Adjustment Register 1 (AUD_PAR_BUSCLK_1)

Figure 10-175. Audio Input Data Rate Adjustment Register 1 (AUD_PAR_BUSCLK_1)

31	8	7	0
Reserved		AUD_PAR_BUSCLK_1	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-187. Audio Input Data Rate Adjustment Register 1 (AUD_PAR_BUSCLK_1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AUD_PAR_BUSCLK_1	Decimal part of adjustment parameter

10.3.4.30 Audio Input Data Rate Adjustment Register 2 (AUD_PAR_BUSCLK_2)

Figure 10-176. Audio Input Data Rate Adjustment Register 2 (AUD_PAR_BUSCLK_2)

31	8	7	0
Reserved		AUD_PAR_BUSCLK_2	
R-0h		R/W-8h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-188. AUD_PAR_BUSCLK_2 Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AUD_PAR_BUSCLK_2	Lower byte of integer part of parameter

10.3.4.31 Audio Input Data Rate Adjustment Register 3 (AUD_PAR_BUSCLK_3)

Figure 10-177. Audio Input Data Rate Adjustment Register 3 (AUD_PAR_BUSCLK_3)

31	8	7	0
Reserved		AUD_PAR_BUSCLK_3	
R-0h		R/W-0h	

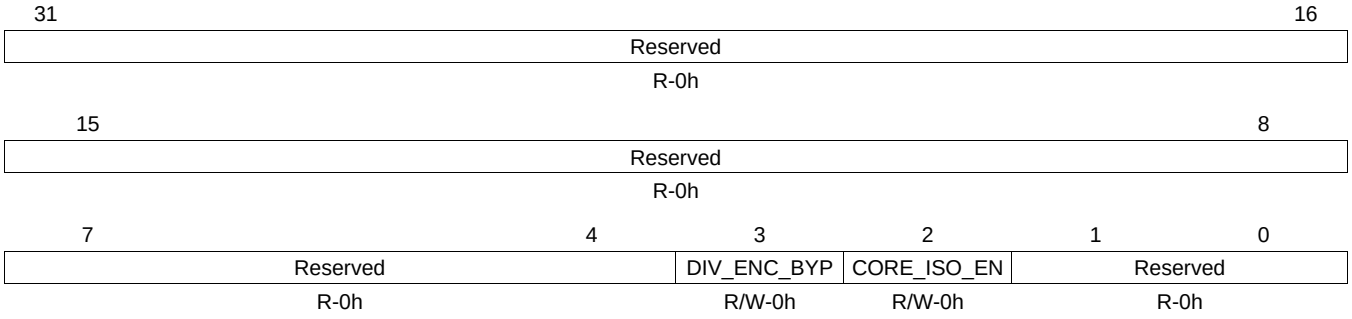
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-189. AUD_PAR_BUSCLK_3 Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AUD_PAR_BUSCLK_3	Upper byte of integer part of parameter

10.3.4.32 Test Control Register (TEST_TXCTRL)

Figure 10-178. Test Control Register (TEST_TXCTRL)



LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-190. Test Control Register (TEST_TXCTRL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	DIV_ENC_BYP		DVI encoder bypass
		0	Normal operation
		1	Bypass DVI encoder logic. Never set this bit. Always use the DVI encoder.
2	CORE_ISO_EN		TMDS Core Isolation Enable
		0	Normal operation
		1	Input pins mixed to TMDS Tx core; to emulate discrete Tx.
1-0	Reserved	0	Reserved

10.3.4.33 Diagnostic Power Down Register (DPD)

Figure 10-179. Diagnostic Power Down Register (DPD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
VID_BYP_EN	Reserved			TCLKPHZ	PDIDCK	PDOSC	PDTOT
R/W-0h	R-0h			R/W-1h	R/W-1h	R/W-1h	R/W-1h

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-191. Diagnostic Power Down Register (DPD) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	VID_BYP_EN		Enable bypath of the video path. The CORE_ISO_EN bit (in register TEST_TXCTRL) must be set and the HDCP cypher must be disabled (HDCP_CTRL.EWNC_EN = 0).
		0	Disable
		1	Enable
6-4	Reserved	0	Reserved
3	TCLKPHZ		Selects the TCLK phase
		0	Default phase; the same as TMDS core 1
		1	Invert TCLK; change the phase 180 degrees
2	PDIDCK		Power down IDCK input
		0	Power down, gate off IDCK signal to disable all IDCK-based logic
		1	Normal operation
1	PDOSC		Power down internal oscillator. This disables the I2C port to the internal ROM (disabling loading), halts all interrupt updates, and disables the Master DDC block.
		0	Power down
		1	Normal operation
0	PDTOT		Power down total
		0	Power down everything; INT source is RSEN
		1	Normal operation

10.3.4.34 Packet Buffer Control 1 Register (PB_CTRL1)

Figure 10-180. Packet Buffer Control 1 Register (PB_CTRL1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
MPEG_EN	MPEG_RPT	AUD_EN	AUD_RPT	SPD_EN	SPD_RPT	AVI_EN	AVI_RPT
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-192. Packet Buffer Control 1 Register (PB_CTRL1) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	MPEG_EN		Enable MPEG InfoFrame transmission
		0	Disable
		1	Enable
6	MPEG_RPT		Repeat MPEG InfoFrame transmission
		0	Disable (send once after enable bit is set)
		1	Enable (send in every VBLANK period)
5	AUD_EN		Enable Audio InfoFrame transmission
		0	Disable
		1	Enable
4	AUD_RPT		Repeat Audio InfoFrame transmission
		0	Disable (send once after enable bit is set)
		1	Enable (send in every VBLANK period)
3	SPD_EN		Enable General Control Packet transmission
		0	Disable
		1	Enable
2	SPD_RPT		Repeat SPD InfoFrame transmission
		0	Disable (send once after enable bit is set)
		1	Enable (send in every VBLANK period)
1	AVI_EN		Enable AVI InfoFrame transmission
		0	Disable
		1	Enable
0	AVI_RPT		Repeat AVI InfoFrame transmission
		0	Disable (send once after enable bit is set)
		1	Enable (send in every VBLANK period)

10.3.4.35 Packet Buffer Control 2 Register (PB_CTRL2)

Figure 10-181. Packet Buffer Control 2 Register (PB_CTRL2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
GAM_EN	GAM_RPT	GEN2_EN	GEN2_RPT	CP_EN	CP_RPT	GEN_EN	GEN_RPT
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-193. Packet Buffer Control 2 Register (PB_CTRL2) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	GAM_EN		Enable Gamut Metadata InfoFrame transmission on HDMI.
		0	Disable
		1	Enable
6	GAM_RPT		Repeat Gamut Metadata InfoFrame Packet data each frame.
		0	Disable (send once after enable bit is set)
		1	Enable (send in every VBLANK period)
5	GEN2_EN		Enable Generic 2 Packet transmission
		0	Disable
		1	Enable
4	GEN2_RPT		Repeat Generic 2 Packet transmission
		0	Disable (send once after enable bit is set)
		1	Enable (send in every VBLANK period)
3	CP_EN		Enable General Control Packet transmission
		0	Disable
		1	Enable
2	CP_RPT		Repeat General Control Packet transmission
		0	Disable (send once after enable bit is set)
		1	Enable (send in every VBLANK period)
1	GEN_EN		Enable Generic Packet transmission
		0	Disable
		1	Enable
0	GEN_RPT		Repeat Generic Packet transmission
		0	Disable (send once after enable bit is set)
		1	Enable (send in every VBLANK period)

10.3.4.36 AVI InfoFrame Register (AVI_TYPE)

Figure 10-182. AVI InfoFrame Register (AVI_TYPE)

31	8	7	0
Reserved		AVI_TYPE	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-194. AVI InfoFrame Register (AVI_TYPE) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AVI_TYPE	AVI InfoFrame Type Code. AVI_HDR[7:0]

10.3.4.37 AVI InfoFrame Register (AVI_VERS)

Figure 10-183. AVI InfoFrame Register (AVI_VERS)

31	8	7	0
Reserved		AVI_VERS	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-195. AVI InfoFrame Register (AVI_VERS) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AVI_VERS	AVI InfoFrame Version Code. AVI_HDR[15:8]

10.3.4.38 AVI InfoFrame Register (AVI_LEN)

Figure 10-184. AVI InfoFrame Register (AVI_LEN)

31	8	7	0
Reserved		AVI_LEN	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-196. AVI InfoFrame Register (AVI_LEN) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AVI_LEN	AVI InfoFrame Length. AVI_HDR[23:16]

10.3.4.39 AVI InfoFrame Register (AVI_CHSUM)

Figure 10-185. AVI InfoFrame Register (AVI_CHSUM)

31	8	7	0
Reserved		AVI_CHSUM	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-197. AVI InfoFrame Register (AVI_CHSUM) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AVI_CHSUM	AVI InfoFrame Checksum. AVI_HDR[31:24]

10.3.4.40 AVI InfoFrame Registers (AVI_DBYTE_0-AVI_DBYTE_14)

Figure 10-186. AVI InfoFrame Registers (AVI_DBYTE_0-AVI_DBYTE_14)

31	8	7	0
Reserved		AVI_DATA	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-198. AVI InfoFrame Registers (AVI_DBYTE_0-AVI_DBYTE_14) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AVI_DATA	AVI InfoFrame Data Bytes.

10.3.4.41 SPD InfoFrame Register (SPD_TYPE)

Figure 10-187. SPD InfoFrame Register (SPD_TYPE)

31	8	7	0
Reserved		SPD_TYPE	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-199. SPD InfoFrame Register (SPD_TYPE) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	SPD_TYPE	SPD InfoFrame Type Code. SPD_HDR[7:0]

10.3.4.42 SPD InfoFrame Register (SPD_VERS)

Figure 10-188. SPD InfoFrame Register (SPD_VERS)

31	8	7	0
Reserved		SPD_VERS	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-200. SPD InfoFrame Register (SPD_VERS) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	SPD_VERS	SPD InfoFrame Version Code. SPD_HDR[15:8]

10.3.4.43 SPD InfoFrame Register (SPD_LEN)

Figure 10-189. SPD InfoFrame Register (SPD_LEN)

31	8	7	0
Reserved		SPD_LEN	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-201. SPD InfoFrame Register (SPD_LEN) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	SPD_LEN	SPD InfoFrame Length. SPD_HDR[23:16]

10.3.4.44 SPD InfoFrame Register (SPD_CHSUM)

Figure 10-190. SPD InfoFrame Register (SPD_CHSUM)

31	8	7	0
Reserved		SPD_CHSUM	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-202. SPD InfoFrame Register (SPD_CHSUM) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	SPD_CHSUM	SPD InfoFrame Checksum. SPD_HDR[31:24]

10.3.4.45 SPD InfoFrame Registers (SPD_DBYTE_0-SPD_DBYTE_26)

Figure 10-191. SPD InfoFrame Registers (SPD_DBYTE_0-SPD_DBYTE_26)

31	8	7	0
Reserved		SPD_DATA	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-203. SPD InfoFrame Registers (SPD_DBYTE_0-SPD_DBYTE_26) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	SPD_DATA	SPD InfoFrame Data Bytes.

10.3.4.46 Audio InfoFrame Register (AUDIO_TYPE)

Figure 10-192. Audio InfoFrame Register (AUDIO_TYPE)

31	8	7	0
Reserved		AUDIO_TYPE	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-204. Audio InfoFrame Register (AUDIO_TYPE) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AUDIO_TYPE	AUDIO InfoFrame Type Code. AUDIO_HDR[7:0]

10.3.4.47 Audio InfoFrame Register (AUDIO_VERS)

Figure 10-193. Audio InfoFrame Register (AUDIO_VERS)

31	8	7	0
Reserved		AUDIO_VERS	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-205. Audio InfoFrame Register (AUDIO_VERS) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AUDIO_VERS	AUDIO InfoFrame Version Code. AUDIO_HDR[15:8]

10.3.4.48 Audio InfoFrame Register (AUDIO_LEN)

Figure 10-194. Audio InfoFrame Register (AUDIO_LEN)

31	8	7	0
Reserved		AUDIO_LEN	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-206. Audio InfoFrame Register (AUDIO_LEN) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AUDIO_LEN	AUDIO InfoFrame Length. AUDIO_HDR[23:16]

10.3.4.49 Audio InfoFrame Register (AUDIO_CHSUM)

Figure 10-195. Audio InfoFrame Register (AUDIO_CHSUM)

31	8	7	0
Reserved		AUDIO_CHSUM	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-207. Audio InfoFrame Register (AUDIO_CHSUM) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AUDIO_CHSUM	AUDIO InfoFrame Checksum. AUDIO_HDR[31:24]

10.3.4.50 Audio InfoFrame Registers (AUDIO_DBYTE_0-AUDIO_DBYTE_9)

Figure 10-196. Audio InfoFrame Registers (AUDIO_DBYTE_0-AUDIO_DBYTE_9)

31	8	7	0
Reserved		AUDIO_DATA	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-208. Audio InfoFrame Registers (AUDIO_DBYTE_0-AUDIO_DBYTE_9) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	AUDIO_DATA	AUDIO InfoFrame Data Bytes

10.3.4.51 MPEG InfoFrame Register (MPEG_TYPE)

Figure 10-197. MPEG InfoFrame Register (MPEG_TYPE)

31	8	7	0
Reserved		MPEG_TYPE	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-209. MPEG InfoFrame Register (MPEG_TYPE) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	MPEG_TYPE	MPEG InfoFrame Type Code. MPEG_HDR[7:0]

10.3.4.52 MPEG InfoFrame Register (MPEG_VERS)

Figure 10-198. MPEG InfoFrame Register (MPEG_VERS)

31	8	7	0
Reserved		MPEG_VERS	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-210. MPEG InfoFrame Register (MPEG_VERS) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	MPEG_VERSN	MPEG InfoFrame Version Code. MPEG_HDR[15:8]

10.3.4.53 MPEG InfoFrame Register (MPEG_LEN)

Figure 10-199. MPEG InfoFrame Register (MPEG_LEN)

31	8	7	0
Reserved		MPEG_LEN	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-211. MPEG InfoFrame Register (MPEG_LEN) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	MPEG_LEN	MPEG InfoFrame Length. MPEG_HDR[23:16]

10.3.4.54 MPEG InfoFrame Register (MPEG_CHSUM)

Figure 10-200. MPEG InfoFrame Register (MPEG_CHSUM)

31	8	7	0
Reserved		MPEG_CHSUM	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-212. MPEG InfoFrame Register (MPEG_CHSUM) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	MPEG_CHSUM	MPEG InfoFrame Checksum. MPEG_HDR[31:24]

10.3.4.55 MPEG InfoFrame Registers (MPEG_DBYTE_0-MPEG_DBYTE_26)

Figure 10-201. MPEG InfoFrame Registers (MPEG_DBYTE_0-MPEG_DBYTE_26)

31	8	7	0
Reserved		MPEG_DATA	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-213. MPEG InfoFrame Registers (MPEG_DBYTE_0-MPEG_DBYTE_26) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	MPEG_DATA	MPEG InfoFrame Data Bytes

10.3.4.56 Generic Packet Registers (GEN_DBYTE_0-GEN_DBYTE_30)

Figure 10-202. Generic Packet Registers (GEN_DBYTE_0-GEN_DBYTE_30)

31	8	7	0
Reserved		GEN_DATA	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-214. Generic Packet Registers (GEN_DBYTE_0-GEN_DBYTE_30) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	GEN_DATA	Generic Packet Data Bytes

10.3.4.57 General Control Packet Register (CP_BYTE1)

Figure 10-203. General Control Packet Register (CP_BYTE1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved			CLRAVM	Reserved			SETAVM
R-0h			R/W-0h	R-0h			R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-215. General Control Packet Register (CP_BYTE1) Field Descriptions

Bit	Field	Description
31-5	Reserved	Reserved
4	CLRAVM	Clear AV Mute flag
3-1	Reserved	Reserved
0	SETAVM	Set AV Mute flag. When the AVMUTE flag is set, the HDMI Transmitter sends a General Control Packet on the TMDS link to inform the Sink that the data may be incorrect. The HDMI Transmitter sends blanklevel data for all video packets and 00 for all audio packet data. When the AVMUTE flag is set, the Sink assumes that no valid data is being received. Optionally, the Sink can apply a mute function to the audio data and/or a blank function to the video data.

10.3.4.58 Generic Packet 2 Registers (GEN2_DBYTE_0-GEN2_DBYTE_30)

Figure 10-204. Generic Packet 2 Registers (GEN2_DBYTE_0-GEN2_DBYTE_30)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
GEN2_DATA							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-216. Generic Packet 2 Registers (GEN2_DBYTE_0-GEN2_DBYTE_30) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	GEN2_DATA	Generic Packet 2 Data Bytes

10.3.4.59 CEC Slave ID Register (CEC_ADDR_ID)

Figure 10-205. CEC Slave ID Register (CEC_ADDR_ID)

31	8	7	0
Reserved		CEC_ID	
R-0h		R/W-C0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-217. CEC Slave ID Register (CEC_ADDR_ID) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CEC_ID	CEC I2C slave address ID

10.3.5 HDMI IP Core CEC Registers

Table 10-218 lists the HDMI IP core CEC registers.

Table 10-218. HDMI IP Core CEC Registers

Address Offset	Acronym	Register Name
00h	CEC_DEV_ID	CEC Device ID Register
04h	CEC_SPEC	CEC Specification Register
08h	CEC_SUFF	CEC Specification Suffix Register
0Ch	CEC_FW	CEC Firmware Revision Register
10h	CEC_DBG_0	CEC Debug Register 0
14h	CEC_DBG_1	CEC Debug Register 1
18h	CEC_DBG_2	CEC Debug Register 2
1Ch	CEC_DBG_3	CEC Debug Register 3
20h	CEC_TX_INIT	CEC Tx Initialization Register
24h	CEC_TX_DEST	CEC Tx Destination Register
38h	CEC_SETUP	CEC Setup Register
3Ch	CEC_TX_COMMAND	CEC Tx Command Register
40h-78h	CEC_TX_OPERAND_0 - CEC_TX_OPERAND_14	CEC Tx Operand Registers
7Ch	CEC_TRANSMIT_DATA	CEC Transmit Data Register
88h	CEC_CA_7_0	CEC Capture ID0 Register
8Ch	CEC_CA_15_8	CEC Capture ID0 Register
90h	CEC_INT_ENABLE_0	CEC Interrupt Enable Register 0
94h	CEC_INT_ENABLE_1	CEC Interrupt Enable Register 1
98h	CEC_INT_STATUS_0	CEC Interrupt Status Register 0
9Ch	CEC_INT_STATUS_1	CEC Interrupt Status Register 1
B0h	CEC_RX_CONTROL	CEC RX Control Register
B4h	CEC_RX_COUNT	CEC Rx Count Register
B8h	CEC_RX_CMD_HEADER	CEC Rx Command Header Register
BCh	CEC_RX_COMMAND	CEC Rx Command Register
C0h-F8h	CEC_RX_OPERAND_0 - CEC_RX_OPERAND_14	CEC Rx Operand Registers

10.3.5.1 CEC Device ID Register (CEC_DEV_ID)

Figure 10-206. CEC Device ID Register (CEC_DEV_ID)

31	8	7	0
Reserved		CEC_DEV_ID	
R-0h		R-CCh	

LEGEND: R = Read only; -*n* = value after reset

Table 10-219. CEC Device ID Register (CEC_DEV_ID) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CEC_DEV_ID	ID of CEC device

10.3.5.2 CEC Specification Register (CEC_SPEC)

Figure 10-207. CEC Specification Register (CEC_SPEC)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
CEC_REL				CEC_REV			
R-1h				R-3h			

LEGEND: R = Read only; -*n* = value after reset

Table 10-220. CEC Specification Register (CEC_SPEC) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-4	CEC_REL	CEC Specification major release
3-0	CEC_REV	CEC Specification minor release

10.3.5.3 CEC Specification Suffix Register (CEC_SUFF)

Figure 10-208. EC Specification Suffix Register (CEC_SUFF)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
SUB_SYS	Reserved			CEC_SUFF			
R-1h	R-0h			R-0h			

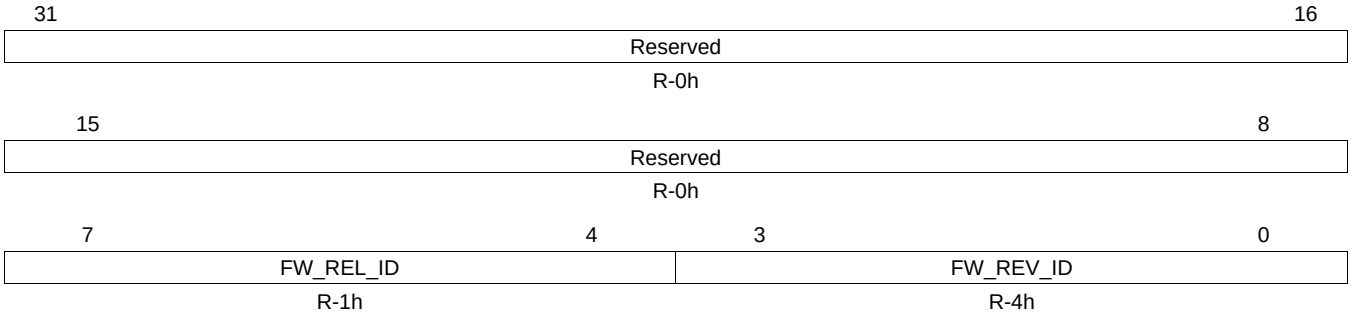
LEGEND: R = Read only; -*n* = value after reset

Table 10-221. EC Specification Suffix Register (CEC_SUFF) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	SUB_SYS		Subsystem.
		0	Firmware
		1	Hardware
6-4	Reserved	0	Reserved
3-0	CEC_SUFF	0	CEC Specification Suffix (0 = a for rev 1.2a)

10.3.5.4 CEC Firmware Revision Register (CEC_FW)

Figure 10-209. CEC Firmware Revision Register (CEC_FW)



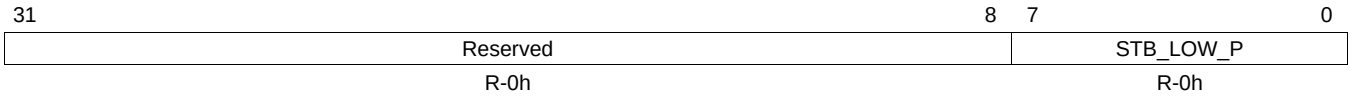
LEGEND: R = Read only; -*n* = value after reset

Table 10-222. CEC Firmware Revision Register (CEC_FW) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-4	FW_REL_ID	Firmware Release ID
3-0	FW_REV_ID	Firmware Revision ID

10.3.5.5 CEC Debug Register 0 (CEC_DBG_0)

Figure 10-210. CEC Debug Register 0 (CEC_DBG_0)



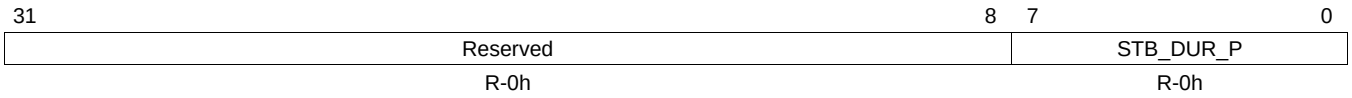
LEGEND: R = Read only; -*n* = value after reset

Table 10-223. CEC Debug Register 0 (CEC_DBG_0) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	STB_LOW_P	Start Bit Low Period. Measured in units of 250 µs. Expected range is 3.5 ms (Eh) to 3.9 ms (0Fh).

10.3.5.6 CEC Debug Register 1 (CEC_DBG_1)

Figure 10-211. CEC Debug Register 1 (CEC_DBG_1)



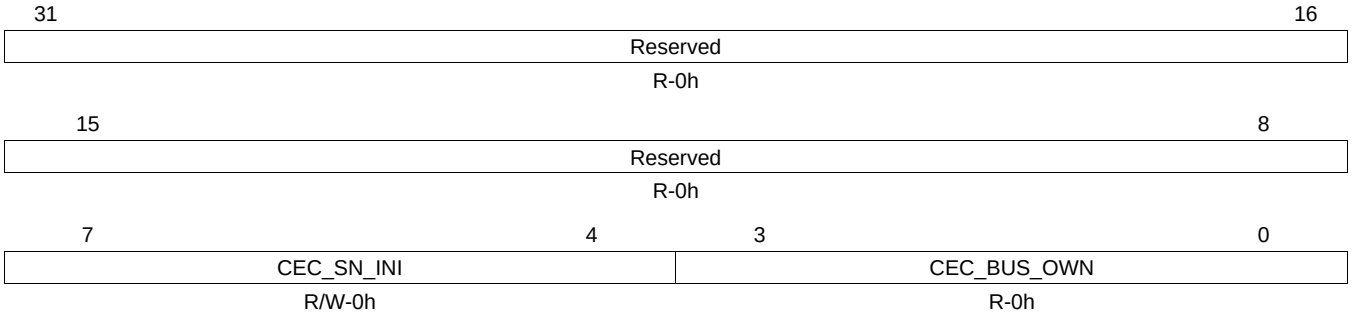
LEGEND: R = Read only; -*n* = value after reset

Table 10-224. CEC Debug Register 1 (CEC_DBG_1) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	STB_DUR_P	Start Bit Duration Period. Measured in units of 250 µs. Expected range is 4.3 ms (11h) to 4.7 ms (12h).

10.3.5.7 CEC Debug Register 2 (CEC_DBG_2)

Figure 10-212. CEC Debug Register 2 (CEC_DBG_2)



LEGEND: R = Read only; -*n* = value after reset

Table 10-225. CEC Debug Register 2 (CEC_DBG_2) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-4	CEC_SN_INI	CEC Snoop Initiator. Values 0 to Fh.
3-0	CEC_BUS_OWN	Current CEC bus owner. Values 0 to Fh.

10.3.5.8 CEC Debug Register 3 (CEC_DBG_3)

Figure 10-213. CEC Debug Register 3 (CEC_DBG_3)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
FL_FIF	FR_RT_CNT			Reserved	INV_ACK	ACKN_HEAD	CEC_SN
R/W-0h	R-0h			R-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-226. CEC Debug Register 3 (CEC_DBG_3) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	FL_FIF		Flush Tx FIFO
		0	No
		1	Yes, self-resetting bit
6-4	FR_RT_CNT	0	Frame Retransmit Count. Values 0 to 5.
3	Reserved	0	Reserved
2	INV_ACK		Invert ACK to Broadcast Commands
		0	No
		1	Yes
1	ACKN_HEAD		ACK/NACK Header Block
		0	ACK
		1	NACK
0	CEC_SN		CEC snoop
		0	Disable
		1	Enable

10.3.5.9 CEC Tx Initialization Register (CEC_TX_INIT)

Figure 10-214. CEC Tx Initialization Register (CEC_TX_INIT)

31	4	3	0
Reserved		CEC_INIT_ID	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-227. CEC Tx Initialization Register (CEC_TX_INIT) Field Descriptions

Bit	Field	Description
31-4	Reserved	Reserved
3-0	CEC_INIT_ID	CEC Initiator ID

10.3.5.10 CEC Tx Destination Register (CEC_TX_DEST)

Figure 10-215. CEC Tx Destination Register (CEC_TX_DEST)

31	16
Reserved	
R-0h	

15	8
Reserved	
R-0h	

7	6	4	3	0
CEC_SD_POLL	Reserved		CEC_DEST_ID	
R/W-0h	R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-228. CEC Tx Destination Register (CEC_TX_DEST) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	CEC_SD_POLL		Generate a polling message
		0	No
		1	Yes, self-resetting bit
6-4	Reserved	0	Reserved
3-0	CEC_DEST_ID	0	CEC Destination ID. Identifies the target device for the command. Must be written before writing the corresponding CEC command.

10.3.5.11 CEC Setup Register (CEC_SETUP)

Figure 10-216. CEC Setup Register (CEC_SETUP)

31	16
Reserved	
R-0h	

15	8
Reserved	
R-0h	

7	5	4	3	2	1	0
Reserved		CEC_PTRH	Reserved	CEC_FORCE_NON_CALIB	CEC_CAL_EN	CEC_CAL_SW
R-0h		R/W-1h	R-0h	R/W-1h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-229. CEC Setup Register (CEC_SETUP) Field Descriptions

Bit	Field	Description
31-5	Reserved	Reserved
4	CEC_PTRH	CEC pass-thru register
3	Reserved	Reserved
2	CEC_FORCE_NON_CALIB	Must be set to 1 if calibration is not needed (If a 2 MHz crystal clock is available)
1	CEC_CAL_EN	CEC calibration enable register (self clearing)
0	CEC_CAL_SW	Must be set to 1 for 10 ms in case of calibration

10.3.5.12 CEC Tx Command Register (CEC_TX_COMMAND)

Figure 10-217. CEC Tx Command Register (CEC_TX_COMMAND)

31	8	7	0
Reserved		CEC_TX_COM	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-230. CEC Tx Command Register (CEC_TX_COMMAND) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CEC_TX_COM	CEC Tx Command

10.3.5.13 CEC Tx Operand Registers (CEC_TX_OPERAND_0-CEC_TX_OPERAND_14)

Figure 10-218. CEC Tx Operand Registers (CEC_TX_OPERAND_0-CEC_TX_OPERAND_14)

31	8	7	0
Reserved		CEC_TX_COM	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-231. CEC Tx Operand Registers (CEC_TX_OPERAND_0-CEC_TX_OPERAND_14) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CEC_TX_COM	CEC Tx Operand

10.3.5.14 CEC Transmit Data Register (CEC_TRANSMIT_DATA)

Figure 10-219. CEC Transmit Data Register (CEC_TRANSMIT_DATA)

31	16
Reserved	
R-0h	

15	8
Reserved	
R-0h	

7	6	5	4	3	0
Reserved	TX_BFR_ACC	TX_AUT_CALC	TRA_CEC_CMD	TX_CNT	
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-232. CEC Transmit Data Register (CEC_TRANSMIT_DATA) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6	TX_BFR_ACC		Read back internal buffer contents from 8Fh–9Eh.
		0	No
		1	Yes
5	TX_AUT_CALC		Auto-Calculate TX_CNT and send.
		0	No
		1	Yes
4	TRA_CEC_CMD		Send CEC Command and TX_CNT Operands.
		0	No
		1	Yes
3-0	TX_CNT		Transmit Byte Count. Selects the number of CEC_TX_OPERAND bytes to send with the command.
		0	No operands
		1h-Fh	1 operand to 15 operands

10.3.5.15 CEC Capture ID0 Register (CEC_CA_7_0)

Figure 10-220. CEC Capture ID0 Register (CEC_CA_7_0)

31	8	7	0
Reserved		CEC_CAP_ID	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-233. CEC Capture ID0 Register (CEC_CA_7_0) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CEC_CAP_ID	The CEC Capture ID register is separate from the CEC Initiator ID. It selects the received commands that will be captured in the receive FIFO and acknowledged. If the command destination matches one of the bits set in this register or is a Broadcast cycle; it will be captured.

10.3.5.16 CEC Capture ID0 Register (CEC_CA_15_8)

Figure 10-221. CEC Capture ID0 Register (CEC_CA_15_8)

31	8	7	0
Reserved		CEC_CAP_ID	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-234. CEC Capture ID0 Register (CEC_CA_15_8) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CEC_CAP_ID	The CEC Capture ID register is separate from the CEC Initiator ID. It selects the received commands that will be captured in the receive FIFO and acknowledged. If the command destination matches one of the bits set in this register or is a Broadcast cycle; it will be captured.

10.3.5.17 CEC Interrupt Enable Register 0 (CEC_INIT_ENABLE_0)

Figure 10-222. CEC Interrupt Enable Register 0 (CEC_INIT_ENABLE_0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		CEC_INTR1_MASK5	Reserved		CEC_INTR1_MASK2	CEC_INTR1_MASK1	CEC_INTR1_MASK0
R-0h		R/W-0h	R-0h		R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-235. CEC Interrupt Enable Register 0 (CEC_INIT_ENABLE_0) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5	CEC_INTR1_MASK5		Tx: Transmit Buffer Full/Empty Change event
		0	Disable
		1	Enable
4-3	Reserved	0	Reserved
2	CEC_INTR1_MASK2		Transmitter FIFO Empty Event
		0	Disable
		1	Enable
1	CEC_INTR1_MASK1		Receiver FIFO Not Empty Event
		0	Disable
		1	Enable
0	CEC_INTR1_MASK0		Command Being Received Event
		0	Disable
		1	Enable

10.3.5.18 CEC Interrupt Enable Register 1 (CEC_INIT_ENABLE_1)

Figure 10-223. CEC Interrupt Enable Register 1 (CEC_INIT_ENABLE_1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved				CEC_INTR2_MASK3	CEC_INTR2_MASK2	CEC_INTR2_MASK1	CEC_INTR2_MASK0
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-236. CEC Interrupt Enable Register 1 (CEC_INIT_ENABLE_1) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	CEC_INTR2_MASK3		Rx FIFO Overrun Error Event
		0	Disable
		1	Enable
2	CEC_INTR2_MASK2		Short Pulse Detected Event
		0	Disable
		1	Enable
1	CEC_INTR2_MASK1		Frame Retransmit Count Exceeded Event
		0	Disable
		1	Enable
0	CEC_INTR2_MASK0		Start Bit Irregularity Event
		0	Disable
		1	Enable

10.3.5.19 CEC Interrupt Status Register 0 (CEC_INIT_STATUS_0)

Figure 10-224. CEC Interrupt Status Register 0 (CEC_INIT_STATUS_0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
CEC_LSTAT	TFIF_BFULL	CEC_INTR1_STAT4	Reserved		CEC_INTR1_STAT2	CEC_INTR1_STAT1	CEC_INTR1_STAT0
R-0h	R-0h	R/W-0h	R-0h		R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-237. CEC Interrupt Status Register 0 (CEC_INIT_STATUS_0) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	CEC_LSTAT		CEC line current state
		0	Low
		1	High
6	TFIF_BFULL		Tx FIFO Transmit Buffer Full
		0	No
		1	Yes
5	CEC_INTR1_STAT4		Tx: Transmit Buffer Full/Empty Change Event Pending
		0	No
		1	Yes
4-3	Reserved	0	Reserved
2	CEC_INTR1_STAT2		Transmitter FIFO Empty Event Pending
		0	No
		1	Yes
1	CEC_INTR1_STAT1		Receiver FIFO Not Empty Event Pending
		0	Disable
		1	Enable
0	CEC_INTR1_STAT0		Command Being Received Event Pending
		0	Disable
		1	Enable

10.3.5.20 CEC Interrupt Status Register 1 (CEC_INIT_STATUS_1)

Figure 10-225. CEC Interrupt Status Register 1 (CEC_INIT_STATUS_1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved				CEC_INTR2_STAT3	CEC_INTR2_STAT2	CEC_INTR2_STAT1	CEC_INTR2_STAT0
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-238. CEC_INIT_STATUS1 Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	CEC_INTR2_STAT3		Rx FIFO Overrun Error Event
		0	Disable
		1	Enable
2	CEC_INTR2_STAT2		Short Pulse Detected Event
		0	Disable
		1	Enable
1	CEC_INTR2_STAT1		Frame Retransmit Count Exceeded Event
		0	Disable
		1	Enable
0	CEC_INTR2_STAT0		Start Bit Irregularity Event
		0	Disable
		1	Enable

10.3.5.21 CEC RX Control Register (CEC_RX_CONTROL)

Figure 10-226. CEC RX Control Register (CEC_RX_CONTROL)

31	16
Reserved	
R-0h	

15	8
Reserved	
R-0h	

7	2	1	0
Reserved		CLR_RX_FIF_ALL	CLR_RX_FIF_CUR
R-0h		R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-239. CEC RX Control Register (CEC_RX_CONTROL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	CLR_RX_FIF_ALL		Clear all frames from Rx FIFO
		0	No
		1	Yes, self-resetting bit
0	CLR_RX_FIF_CUR		Clear Current Frame from Rx FIFO
		0	No
		1	Yes, self-resetting bit

10.3.5.22 CEC Rx Count Register (CEC_RX_COUNT)

Figure 10-227. CEC Rx Count Register (CEC_RX_COUNT)

31	16
Reserved	
R-0h	

15	8
Reserved	
R-0h	

7	6	4	3	0
RX_ERROR	CEC_RX_CMD_CNT		CEC_RX_BYTE_CNT	
R-0h	R-0h		R-0h	

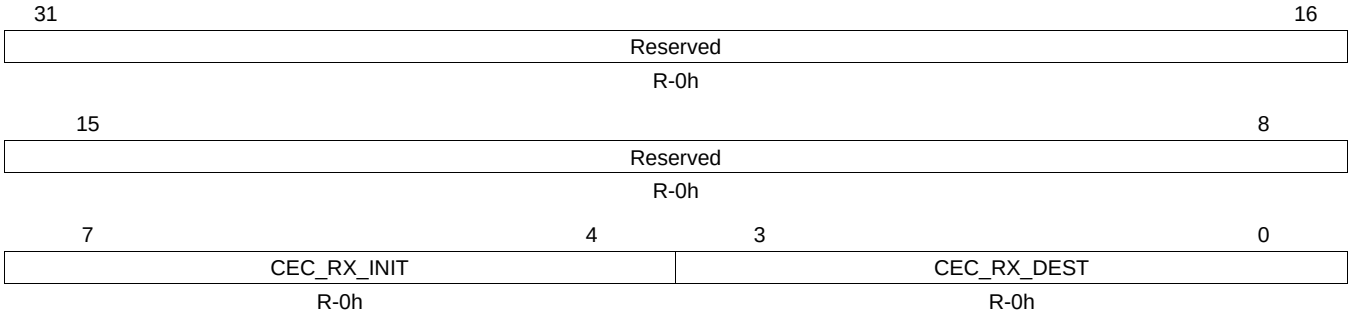
LEGEND: R = Read only; -*n* = value after reset

Table 10-240. CEC Rx Count Register (CEC_RX_COUNT) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RX_ERROR		Error associated with this message
		0	No
		1	Yes
6-4	CEC_RX_CMD_CNT	0	CEC Receive FIFO Frame Count. Returns the number of frames awaiting reading in the FIFO; 0-3.
3-0	CEC_RX_BYTE_CNT	0	CEC Receive Byte Count. Returns the number of operands in the current frame.

10.3.5.23 CEC Rx Command Header Register (CEC_RX_CMD_HEADER)

Figure 10-228. CEC Rx Command Header Register (CEC_RX_CMD_HEADER)



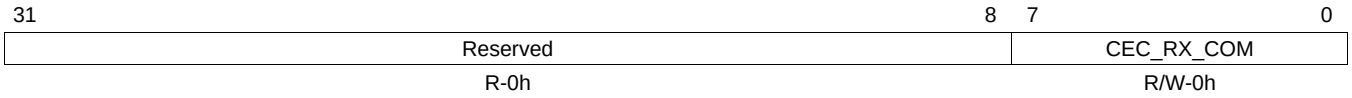
LEGEND: R = Read only; -*n* = value after reset

Table 10-241. CEC Rx Command Header Register (CEC_RX_CMD_HEADER) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-4	CEC_RX_INIT	CEC Initiator ID - identifies the initiator of the current frame.
3-0	CEC_RX_DEST	CEC Destination ID - identifies the intended target of the current frame.

10.3.5.24 CEC Rx Command Register (CEC_RX_COMMAND)

Figure 10-229. CEC Rx Command Register (CEC_RX_COMMAND)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-242. CEC Rx Command Register (CEC_RX_COMMAND) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CEC_RX_COM	CEC Rx Command

10.3.5.25 CEC Rx Operand Registers (CEC_RX_OPERAND_0-CEC_RX_OPERAND_14)

Figure 10-230. CEC Rx Operand Registers (CEC_RX_OPERAND_0-CEC_RX_OPERAND_14)

31	8	7	0
Reserved		CEC_RX_OP	
R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-243. CEC Rx Operand Registers (CEC_RX_OPERAND_0-CEC_RX_OPERAND_14) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7-0	CEC_RX_OP	CEC Rx Operand

10.3.6 HDMI PHY Registers

Table 10-244lists the HDMI PHY registers.

Table 10-244. HDMI PHY Registers

Address Offset	Acronym	Register Name
04h	TMDS_CNTL2	TMDS Control Register
08h	TMDS_CNTL3	TMDS Control Register
0Ch	BIST_CNTL	BIST Control Register
20h	TMDS_CNTL9	TMDS Control Register

10.3.6.1 TMDS Control Register 2 (TMDS_CNTL2)

Figure 10-231. TMDS Control Register 2 (TMDS_CNTL2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		OE	TERM_EN	Reserved		Clkdetect	Rsen
R-0h		R/W-0h	R/W-0h	R-2h		R-x	R-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; x = value is indeterminate

Table 10-245. TMDS Control Register 2 (TMDS_CNTL2) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5	OE		Output enable control
		0	Output drivers of TMDS outputs are switched off
		1	Output drivers are switched on
4	TERM_EN		Source termination enable control
		0	Bus termination is switched off
		1	Bus termination is active; driver must also be enabled
3-2	Reserved	2h	Reserved
1	Clkdetect		Clock detector output
		0	If clock < 2.5 MHz
		1	If clock > 2.5 MHz
0	Rsen		Receiver sense output
		0	When receiver is disconnected
		1	When receiver is connected

10.3.6.2 TMDS Control Register 3 (TMDS_CNTL3)

Figure 10-232. TMDS Control Register 3 (TMDS_CNTL3)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved			clkmult_ctl		dpcolor_ctl		pdb
R-5h			R/W-1h		R/W-0h		R/W-1h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-246. TMDS Control Register 3 (TMDS_CNTL3) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	5h	Reserved
4-3	clkmult_ctl		Clock multiplication factor control. Use this setting when pixel repetition is needed. Output clock “m_clkout_dig” depends on the dpcolor_ctl and clkmult_ctl bits. For example, if the input clock frequency p_clkin is 30 MHz: dpcolor_ctl = 2h and clkmult_ctl = 3h, m_clkout_dig = 30 × 1.5 × 4 = 180 MHz; dpcolor_ctl = 1h and clkmult_ctl = 3h, m_clkout_dig = 30 × 1.25 × 4 = 150 MHz; dpcolor_ctl = 0 and clkmult_ctl = 3h, m_clkout_dig = 30 × 1.0 × 4 = 120 MHz.
		0	0.5×
		1h	1×
		2h	2×
		3h	4×
2-1	dpcolor_ctl		Deep color mode control
		0	8 bit/channel
		1h	10 bit/channel
		2h	12 bit/channel
		3h	Reserved
0	pdb		Powerdown control
		0	PHY will be power down
		1	No power down

10.3.6.3 BIST Control Register (BIST_CNTL)

Figure 10-233. BIST Control Register (BIST_CNTL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved					enc_byp	Reserved	
R-8h					R/W-0h	R-0h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-247. BIST Control Register (BIST_CNTL) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	8h	Reserved
2	enc_byp		DVI encoder bypass. Always bypass the DVI encoder (set this bit to 1).
		0	Data goes through the DVI encoder
		1	Bypass the DVI encoder
1-0	Reserved	0	Reserved

10.3.6.4 TMDS Control Register 9 (TMDS_CNTL9)

Figure 10-234. TMDS Control Register 9 (TMDS_CNTL9)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0h															

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
ten_bit_bypass	Reserved						
R/W-0h	R-0h						

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-248. TMDS Control Register 9 (TMDS_CNTL9) Field Descriptions

Bit	Field	Description
31-8	Reserved	Reserved
7	ten_bit_bypass	Always set this bit to 1.
6-0	Reserved	Reserved

Inter-Integrated Circuit (I2C) Controller Module

This chapter describes the multi-master inter-integrated circuit (I2C) controller module which provides an interface between a CPU and any I2C-bus-compatible device that connects via the I2C serial bus. External components attached to the I2C bus can serially transmit/receive up to 8-bit data to/from the CPU device through the two-wire I2C interface.

Topic **Page**

11.1 Introduction

11.1.1 Overview

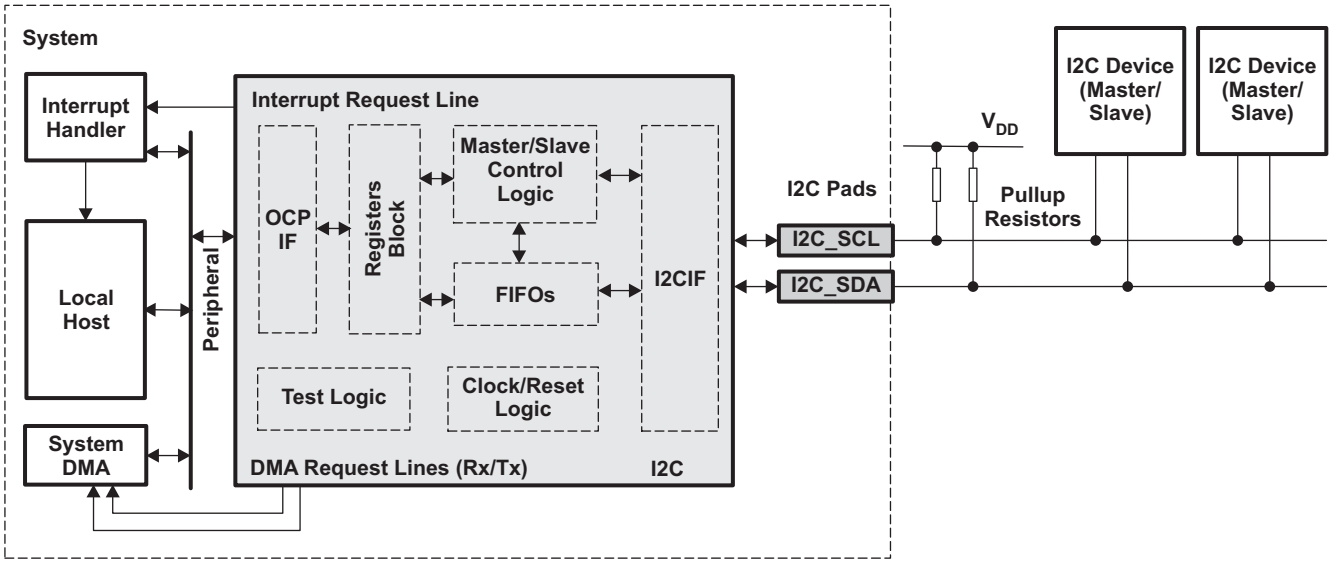
The multi-master I2C peripheral provides an interface between a CPU and any I2C-bus-compatible device that connects via the I2C serial bus. External components attached to the I2C bus can serially transmit/receive up to 8-bit data to/from the CPU device through the two-wire I2C interface.

The I2C bus is a multi-master bus. The I2C controller does support the multi-master mode that allows more than one device capable of controlling the bus to be connected to it. Each I2C device is recognized by a unique address and can operate as either transmitter or receiver, according to the function of the device. In addition to being a transmitter or receiver, a device connected to the I2C bus can also be considered as master or slave when performing data transfers. Note that a master device is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. During this transfer, any device addressed by this master is considered a slave.

11.1.2 Functional Block Diagram

Figure 11-1 shows an example of a system with multiple I2C compatible devices in which the I2C serial ports are all connected together for a two-way transfer from one device to other devices.

Figure 11-1. I2C Functional Block Diagram



11.1.3 Features

The multimaster I2C controller has the following features:

- Compliance with Philips I2C specification version 2.1
- Support for standard mode (up to 100K bits/s) and fast mode (up to 400K bits/s)
- 7-bit and 10-bit device addressing modes
- General call
- Start/restart/stop
- Multimaster transmitter/slave receiver mode
- Multimaster receiver/slave transmitter mode
- Combined master transmit/receive and receive/transmit mode
- Built-in FIFO size of 32 bytes for buffered read or write
- Module enable/disable capability

- Programmable clock generation
- 8-bit-wide data access
- Designed for low-power consumption design
- Two DMA channels
- Wide interrupt capability

11.2 Architecture

The I2C peripheral consists of the following primary blocks:

- A serial interface: one data pin (I2C_SDA) and one clock pin (I2C_SCL).
- Data registers to temporarily hold receive data and transmit data traveling between the I2C_SDA pin and the CPU or the DMA controller.
- Control and status registers
- A peripheral data bus interface to enable the CPU and the DMA controller to access the I2C peripheral registers.
- A clock synchronizer to synchronize the I2C input clock (from the processor clock generator) and the clock on the I2C_SCL pin, and to synchronize data transfers with masters of different clock speeds.
- A prescaler to divide down the input clock that is driven to the I2C peripheral
- A noise filter on each of the two pins, I2C_SDA and I2C_SCL
- An arbitrator to handle arbitration between the I2C peripheral (when it is a master) and another master
- Interrupt generation logic, so that an interrupt can be sent to the CPU
- DMA event generation logic to send an interrupt to the CPU upon reception or transmission of data.

11.2.1 I2C Master/Slave Contoller Signals

Data is communicated to devices interfacing with the I2C via the serial data line (SDA) and the serial clock line (SCL). These two wires can carry information between a device and others connected to the I2C bus. Both SDA and SCL are bi-directional pins. They must be connected to a positive supply voltage via a pull-up resistor. When the bus is free, both pins are high. The driver of these two pins has an open drain to perform the required wired-AND function.

An example of multiple I2C modules that are connected for a two-way transfer from one device to other devices is shown in Figure 11-2.

Figure 11-2. Multiple I2C Modules Connected

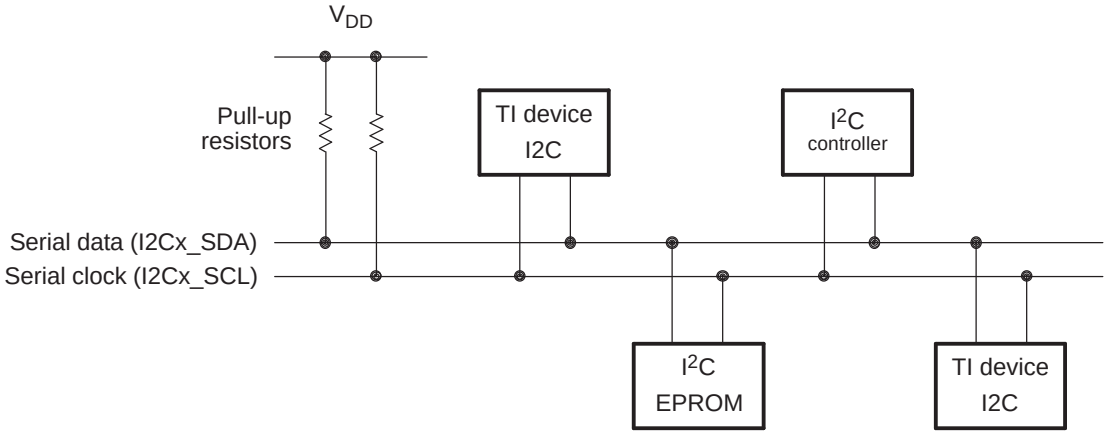


Table 11-1. Signal Pads

Name	I2C Mode	
	Default Operating Mode	Description
I2C_SCL	In/ Out	I2C serial CLK line Open-drain output buffer. Requires external pull-up resistor (Rp).
I2C_SDA	In/ Out	I2C serial data line Open-drain output buffer. Requires external pull-up resistor (Rp).

11.2.2 I2C Reset

The I2C module can be reset in the following three ways:

- A device reset causes all registers to be reset to their default values.
- A software reset by setting the SRST bit in the I2C_SYSC register. This bit has exactly the same action on the module logic as the device reset. All registers are reset to power up reset values.
- The I2C_EN bit in the I2C_CON register can be used to hold the I2C module in reset. When the device reset is removed, I2C_EN = 0 keeps the functional part of I2C module in reset state and all configuration registers can be accessed. I2C_EN = 0 does not reset the registers to power up reset values.

Table 11-2. Reset State of I2C Signals

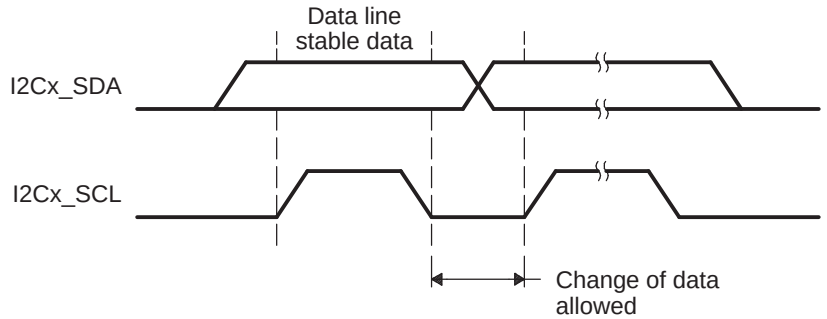
Pin	I/O/Z [1]	System Reset	I2C Reset (I2C_EN = 0)
SDA	I/O/Z	High impedance	High impedance
SCL	I/O/Z	High impedance	High impedance

(1) I = Input, O = Outpu, Z = High impedance

11.2.3 Data Validity

The data on the SDA line must be stable during the high period of the clock. The high and low states of the data line can only change when the clock signal on the SCL line is LOW.

Figure 11-3. Bit Transfer on the I2C Bus

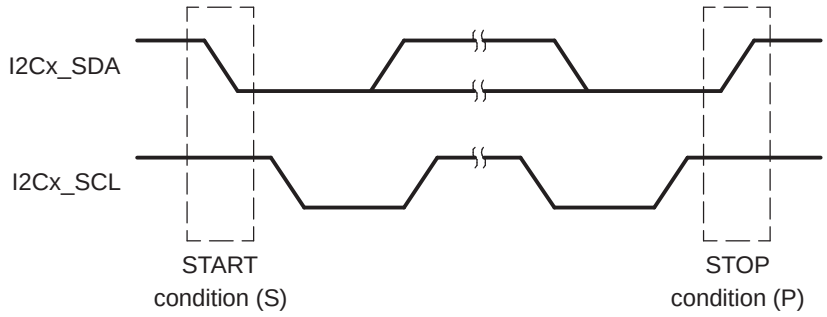


11.2.4 START & STOP Conditions

The I2C module generates START and STOP conditions when it is configured as a master.

- START condition is a high-to-low transition on the SDA line while SCL is high.
- STOP condition is a low-to-high transition on the SDA line while SCL is high.
- The bus is considered to be busy after the START condition (BB = 1) and free after the STOP condition (BB = 0).

Figure 11-4. Start and Stop Condition Events

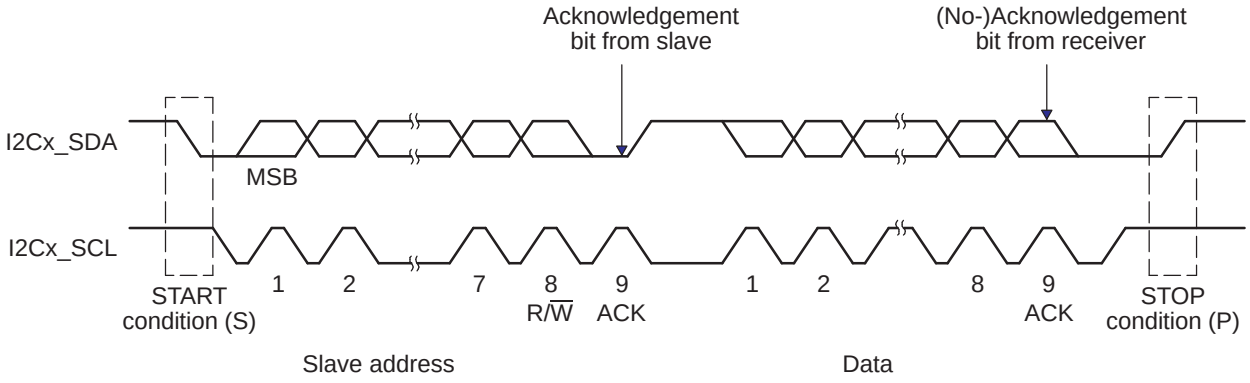


11.2.5 I2C Operation

11.2.5.1 Serial Data Formats

The I2C controller operates in 8-bit word data format (byte write access supported for the last access). Each byte put on the SDA line is 8 bits long. The number of bytes that can be transmitted or received is restricted by the value programmed in the DCOUNT register. The data is transferred with the most significant bit (MSB) first. Each byte is followed by an acknowledge bit from the I2C module if it is in receiver mode.

Figure 11-5. I2C Data Transfer



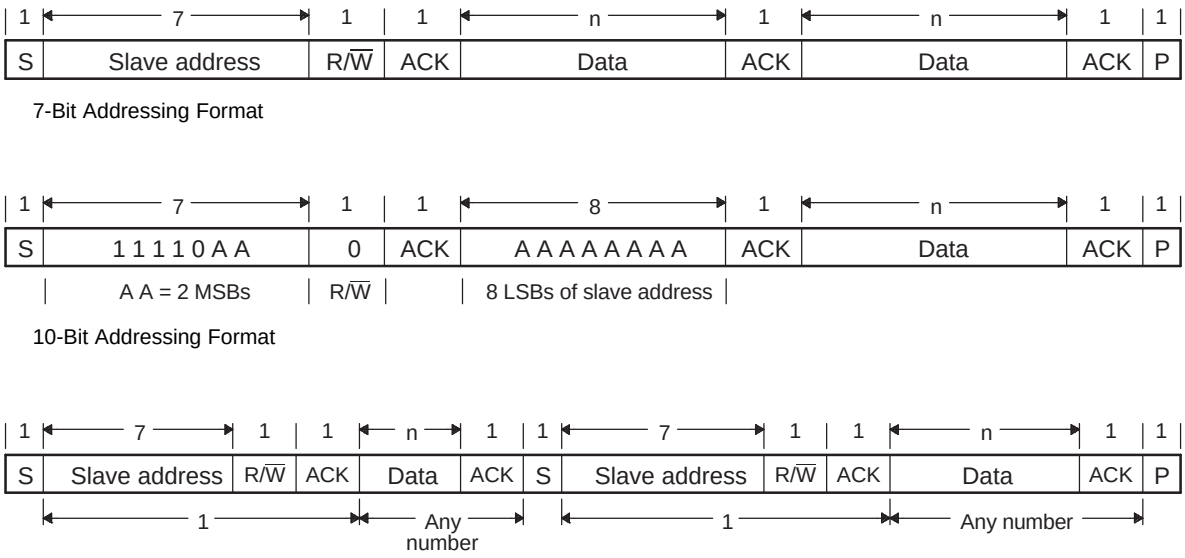
The I2C module supports two data formats, as shown in Figure 11-6:

- 7-bit/10-bit addressing format
- 7-bit/10-bit addressing format with repeated start condition

The first byte after a start condition (S) always consists of 8 bits. In the acknowledge mode, an extra bit dedicated for acknowledgment is inserted after each byte. In the addressing formats with 7-bit addresses, the first byte is composed of 7 MSB slave address bits and 1 LSB R/$\overline{W}$ bit. In the addressing formats with 10-bit addresses, the first byte is composed of 7 MSB slave address bits, such as 11110XX, where XX is the two MSB of the 10-bit addresses, and 1 LSB R/$\overline{W}$ bit, which is 0 in this case.

The least significant R/$\overline{W}$ of the address byte indicates the direction of transmission of the following data bytes. If R/$\overline{W}$ is 0, the master writes data into the selected slave; if it is 1, the master reads data out of the slave.

Figure 11-6. I2C Data Transfer Formats



11.2.5.2 Master Transmitter

In this mode, data assembled in one of the previously described data formats is shifted out on the serial data line SDA in synch with the self-generated clock pulses on the serial clock line SCL. The clock pulses are inhibited and SCL held low when the intervention of the processor is required (XUDF) after a byte has been transmitted.

11.2.5.3 Master Receiver

This mode can only be entered from the master transmitter mode. With either of the address formats (Figure 6 (a), (b), and (c)), the master receiver is entered after the slave address byte and bit R/$\overline{W}$ has been transmitted, if R/$\overline{W}$ is high. Serial data bits received on bus line SDA are shifted in synch with the self-generated clock pulses on SCL. The clock pulses are inhibited and SCL held low when the intervention of the processor is required (ROVR) after a byte has been received. At the end of a transfer, it generates the stop condition.

11.2.5.4 Slave Transmitter

This mode can only be entered from the slave receiver mode. With either of the address formats (Figure 6 (a), (b), and (c)), the slave transmitter is entered if the slave address byte is the same as its own address and bit R/$\overline{W}$ has been transmitted, if R/$\overline{W}$ is high. The slave transmitter shifts the serial data out on the data line SDA in synch with the clock pulses that are generated by the master device. It does not generate the clock but it can hold clock line SCL low while intervention of the CPU is required (XUDF).

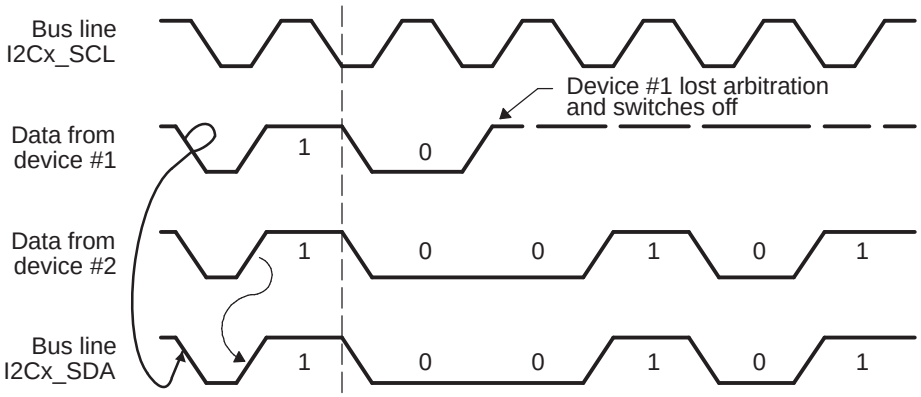
11.2.5.5 Slave Receiver

In this mode, serial data bits received on the bus line SDA are shifted-in in synch with the clock pulses on SCL that are generated by the master device. It does not generate the clock but it can hold clock line SCL low while intervention of the CPU is required (ROVR) following the reception of a byte.

11.2.6 Arbitration

If two or more master transmitters start a transmission on the same bus almost simultaneously, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial bus by the competing transmitters. When a transmitter senses that a high signal it has presented on the bus has been overruled by a low signal, it switches to the slave receiver mode, sets the arbitration lost (AL) flag, and generates the arbitration lost interrupt. Figure 11-7 shows the arbitration procedure between two devices. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. Should two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

Figure 11-7. Arbitration Procedure Between Two Master Transmitters

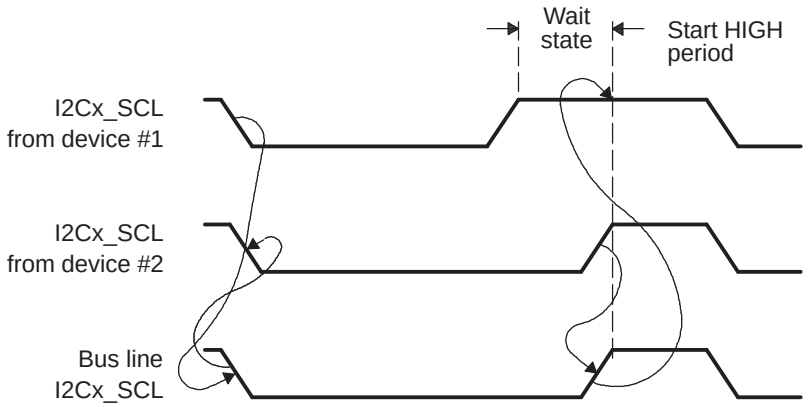


11.2.7 I2C Clock Generation and I2C Clock Synchronization

Under normal conditions, only one master device generates the clock signal, SCL. During the arbitration procedure, however, there are two or more master devices and the clock must be synchronized so that the data output can be compared. The wired-AND property of the clock line means that a device that first generates a low period of the clock line overrules the other devices. At this high/low transition, the clock generators of the other devices are forced to start generation of their own low period. The clock line is then held low by the device with the longest low period, while the other devices that finish their low periods must wait for the clock line to be released before starting their high periods. A synchronized signal on the clock line is thus obtained, where the slowest device determines the length of the low period and the fastest the length of the high period.

If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the WAIT-state. In this way a slave can slow down a fast master and the slow device can create enough time to store a received byte or to prepare a byte to be transmitted. Figure 11-8 illustrates the clock synchronization.

Figure 11-8. Synchronization of Two I2C Clock Generators



11.2.8 Prescaler (SCLK/ICLK)

The I2C module is operated with a functional clock (SCLK) frequency that can be in a range of 12-100 MHz, according to I2C mode that must be used (an internal ~24 MHz clock (ICLK) is recommended in case of F/S operation mode). Note that the frequency of the functional clock influences directly the I2C bus performance and timings.

The internal clock used for I2C logic - ICLK - is generated via the I2C prescaler block. The prescaler consists of a 4-bit register - I2C _PSC, and is used to divide the system clock (SCLK) to obtain the internal required clock for the I2C module.

11.2.9 Noise Filter

The noise filter is used to suppress any noise that is 50 ns or less, in the case of F/S mode of operation. It is designed to suppress noise with one ICLK. The noise filter is always one ICLK cycle, regardless of the bus speed. For FS mode (prescaler = 4, ICLK = 24 MHz), the maximum width of the suppressed spikes is 41.6 ns. To ensure a correct filtering, the prescaler must be programmed accordingly.

11.2.10 I2C Interrupts

The I2C module generates 12 types of interrupt: addressed as slave, bus free (stop condition detected), access error, start condition, arbitration-lost, no acknowledge, general call, registers-ready-for-access, receive and transmit data, receive and transmit draining. These 12 interrupts are accompanied with 12 interrupt masks and flags defined in the I2C_IRQENABLE_SET and respectively I2C_IRQSTATUS_RAW registers. Note that all these 12 interrupt events are sharing the same hardware interrupt line.

- Addressed As Slave interrupt (AAS) is generated to inform the Local Host that an external master addressed the module as a slave. When this interrupt occurs, the CPU can check the I2C_ACTOA status register to check which of the 4 own addresses was used by the external master to access the module.
- Bus Free interrupt (BF) is generated to inform the Local Host that the I2C bus became free (when a Stop Condition is detected on the bus) and the module can initiate his own I2C transaction.
- Start Condition interrupt (STC) is generated after the module being in idle mode have detected (synchronously or asynchronously) a possible Start Condition on the bus (signalized with WakeUp).
- Access Error interrupt (AERR) is generated if a Data read access is performed while RX FIFO is empty or a Data write access is performed while TX FIFO is full.
- Arbitration lost interrupt (AL) is generated when the I2C arbitration procedure is lost.
- No-acknowledge interrupt (NACK) is generated when the master I2C does not receive acknowledge from the receiver.
- General call interrupt (GC) is generated when the device detects the address of all zeros (8 bits).
- Registers-ready-for-access interrupt (ARDY) is generated by the I2C when the previously programmed address, data, and command have been performed and the status bits have been updated. This interrupt is used to let the CPU know that the I2C registers are ready for access.
- Receive interrupt/status (RRDY) is generated when there is received data ready to be read by the CPU from the I2C_DATA register (see the FIFO Management subsection (Section 11.2.13) for a complete description of required conditions for interrupt generation). The CPU can alternatively poll this bit to read the received data from the I2C_DATA register.
- Transmit interrupt/status (XRDY) is generated when the CPU needs to put more data in the I2C_DATA register after the transmitted data has been shifted out on the SDA pin (see the FIFO Management subsection (Section 11.2.13)for a complete description of required conditions for interrupt generation). The CPU can alternatively poll this bit to write the next transmitted data into the I2C_DATA register.
- Receive draining interrupt (RDR) is generated when the transfer length is not a multiple of threshold value, to inform the CPU that it can read the amount of data left to be transferred and to enable the draining mechanism. (see the Draining Feature subsection (Section 11.2.13.4) for additional details).
- Transmit draining interrupt (XDR) is generated when the transfer length is not a multiple of threshold value, to inform the CPU that it can write the amount of data left to be written and to enable the draining mechanism. (see the Draining Feature subsection (Section 11.2.13.4) for additional details).

When the interrupt signal is activated, the Local Host must read the I2C_IRQSTATUS_RAW register to define the type of the interrupt, process the request, and then write into these registers the correct value to clear the interrupt flag.

11.2.11 DMA Events

The I2C module can generate two DMA requests events, read (I2C_DMA_RX) and write (I2C_DMA_TX) that can be used by the DMA controller to synchronously read received data from the I2C_DATA or write transmitted data to the I2C_DATA register. The DMA read and write requests are generated in a similar manner as RRDY and XRDY, respectively.

The I2C DMA request signals (I2C_DMA_TX and I2C_DMA_RX) are activated according to the FIFO Management subsection (Section 11.2.13).

11.2.12 Interrupt and DMA Events

I2C has two DMA channels (Tx and Rx). For event numbers, see the *Device Interrupts and EDMA Events* chapter.

I2C has one interrupt line for all the interrupt requests. For the interrupt number, see the *Device Interrupts and EDMA Events* chapter.

11.2.13 FIFO Management

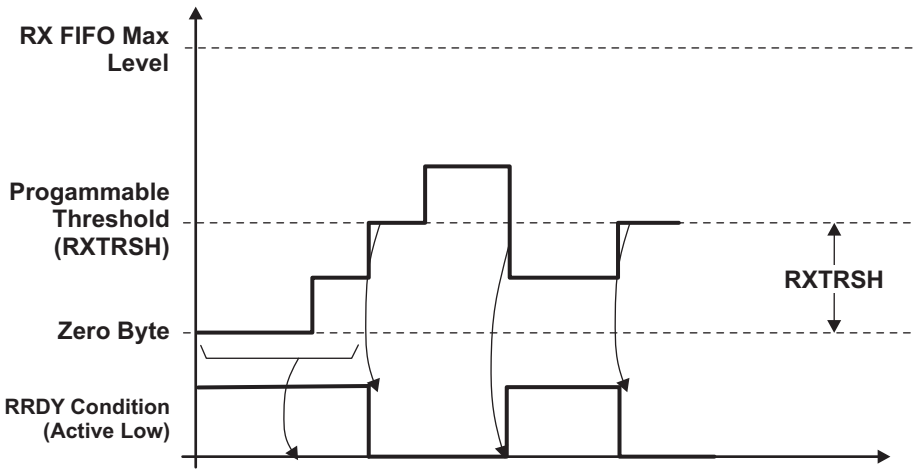
The I2C module implements two internal 32-bytes FIFOs with dual clock for RX and TX modes. The depth of the FIFOs is reflected in I2C_BUFSTAT.FIFODEPTH register.

11.2.13.1 FIFO Interrupt Mode Operation

In FIFO interrupt mode (relevant interrupts enabled via I2C_IRQENABLE_SET register), the processor is informed of the status of the receiver and transmitter by an interrupt signal. These interrupts are raised when receive/transmit FIFO threshold (defined by I2C_BUF.TXTRSH or I2C_BUF.RXTRSH) is reached; the interrupt signals instruct the Local Host to transfer data to the destination (from the I2C module in receive mode and/or from any source to the I2C FIFO in transmit mode).

Figure 11-9 and Figure 11-10, respectively, illustrate receive and transmit operations from FIFO management point of view.

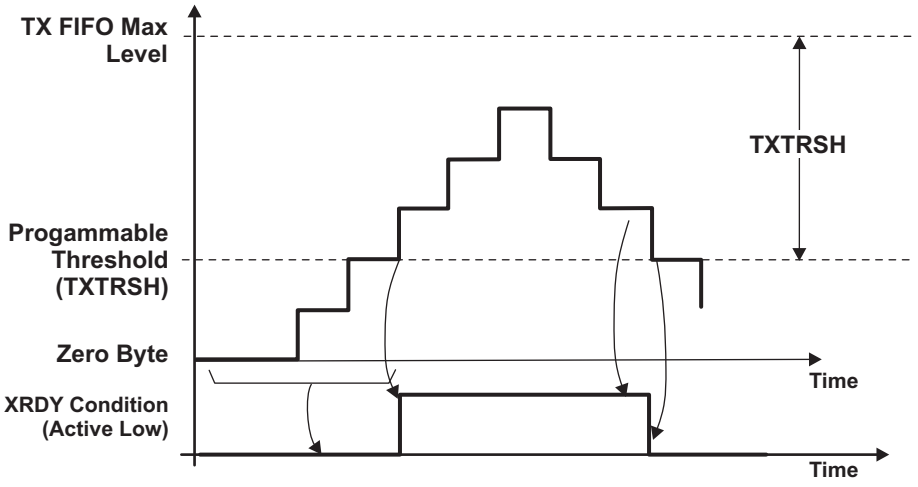
Figure 11-9. Receive FIFO Interrupt Request Generation



Note that in Figure 11-9, the RRDY Condition illustrates that the condition for generating a RRDY interrupt is achieved. The interrupt request is generated when this signal is active, and it can be cleared only by the CPU by writing a 1 in the corresponding interrupt flag. If the condition is still present after clearing the previous interrupt, another interrupt request will be generated.

In receive mode, RRDY interrupt is not generated until the FIFO reaches its receive threshold. Once low, the interrupt can only be de-asserted when the Local Host has handled enough bytes to make the FIFO level below threshold. For each interrupt, the Local Host can be configured to read an amount of bytes equal with the value of the RX FIFO threshold + 1.

Figure 11-10. Transmit FIFO Interrupt Request Generation



Note that in the figure above, the XRDY Condition illustrates that the condition for generating a XRDY interrupt is achieved. The interrupt request is generated when this condition is achieved (when TX FIFO is empty, or the TX FIFO threshold is not reached and there are still data bytes to be transferred in the TX FIFO), and it can be cleared only by the CPU by writing a 1 in the corresponding interrupt flag after transmitting the configured number of bytes. If the condition is still present after clearing the previous interrupt, another interrupt request will be generated.

Note that in interrupt mode, the module offers two options for the CPU application to handle the interrupts:

- When detecting an interrupt request (XRDY or RRDY type), the CPU can write/read one data byte to/from the FIFO and then clear the interrupt. The module will not reassert the interrupt until the interrupt condition is not met.
- When detecting an interrupt request (XRDY or RRDY type), the CPU can be programmed to write/read the amount of data bytes specified by the corresponding FIFO threshold (I2C_BUF.TXTRSH + 1 or I2C_BUF.RXTRSH + 1). In this case, the interrupt condition will be cleared and the next interrupt will be asserted again when the XRDY or RRDY condition is met again.

If the second interrupt serving approach is used, an additional mechanism (draining feature) is implemented for the case when the transfer length is not a multiple of FIFO threshold (see the Draining Feature subsection (Section 11.2.13.4)).

In slave TX mode, the draining feature cannot be used, since the transfer length is not known at the configuration time, and the external master can end the transfer at any point by not acknowledging one data byte.

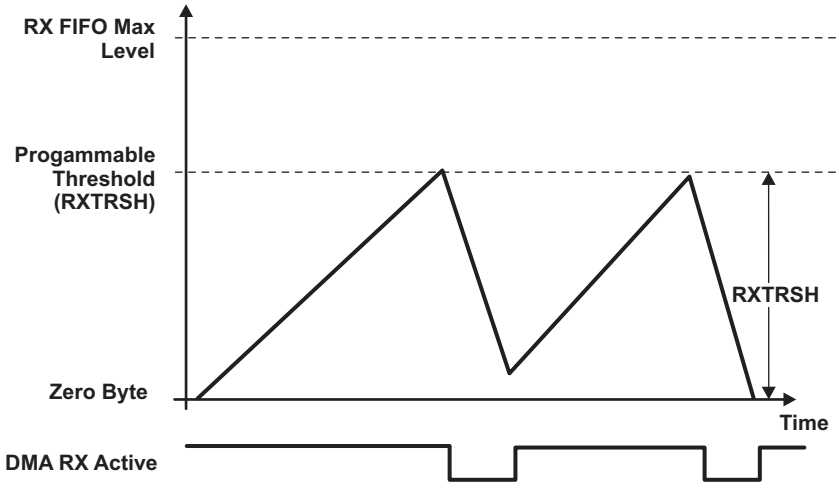
11.2.13.2 FIFO Polling Mode Operation

In FIFO polled mode (I2C_IRQENABLE_SET.XRDY_IE and I2C_IRQENABLE_SET.RRDY_IE disabled and DMA disabled), the status of the module (receiver or transmitter) can be checked by polling the XRDY and RRDY status registers (I2C_IRQSTATUS_RAW) (RDR and XDR can also be polled if draining feature is used). The XRDY and RRDY flags are accurately reflecting the interrupt conditions mentioned in Interrupt Mode. This mode is an alternative to the FIFO interrupt mode of operation, where the status of the receiver and transmitter is automatically known by means of interrupts sent to the CPU.

11.2.13.3 FIFO DMA Mode Operation

In receive mode, a DMA request is generated as soon as the receive FIFO exceeds its threshold level defined in the threshold level register (I2C_BUF.RXTRSH +1). This request should be de-asserted when the number of bytes defined by the threshold level has been read by the DMA, by setting the I2C_DMARXENABLE_CLR.DMARX_ENABLE_CLEAR field.

Figure 11-11. Receive FIFO DMA Request Generation



In transmit mode, a DMA request is automatically asserted when the transmit FIFO is empty. This request should be de-asserted when the number of bytes defined by the number in the threshold register (I2C_BUF.TXTHRS+1) has been written in the FIFO by the DMA, by setting the I2C_DMATXENABLE_CLR. DMATX_ENABLE_CLEAR field. If an insufficient number of characters are written, then the DMA request will remain active. Figure 11-12 and Figure 11-13 illustrate DMA TX transfers with different values for TXTRSH.

Figure 11-12. Transmit FIFO DMA Request Generation (High Threshold)

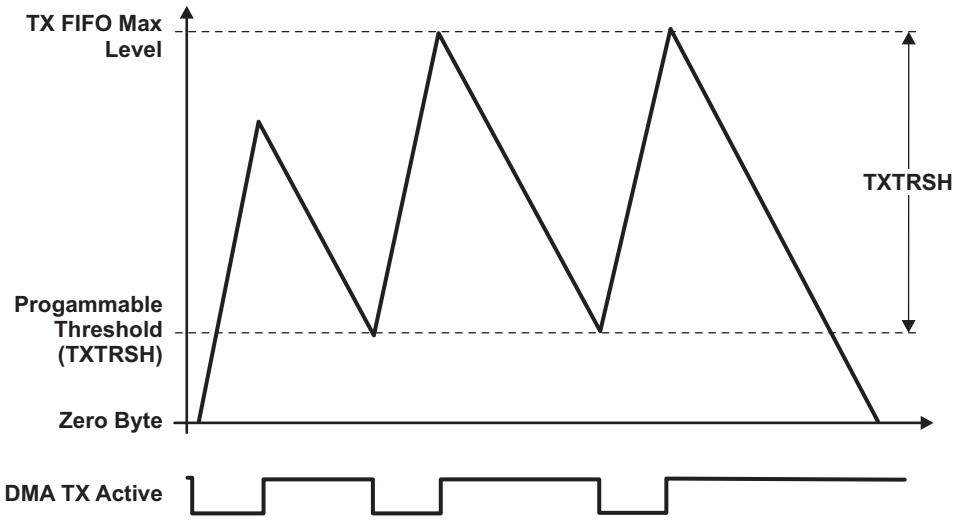
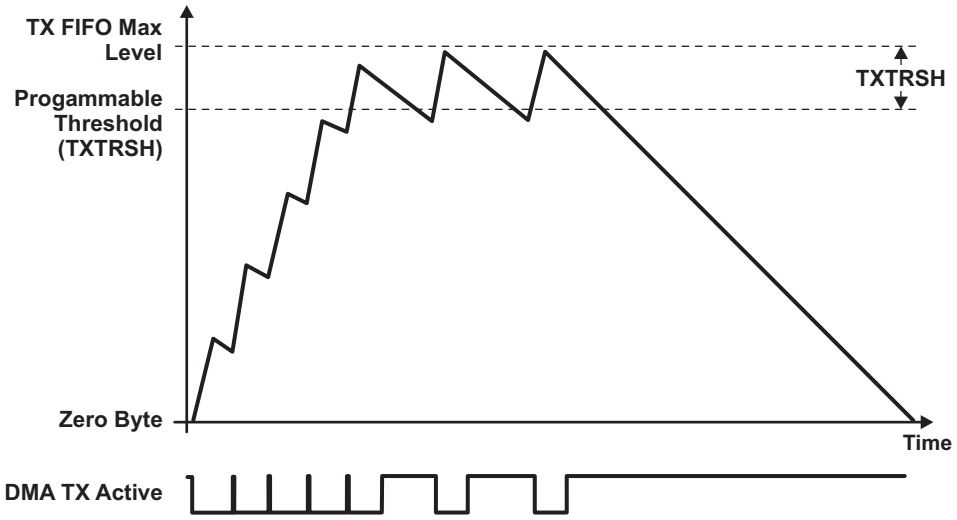


Figure 11-13. Transmit FIFO DMA Request Generation (Low Threshold)



Note that also in DMA mode it is possible to have a transfer whose length is not multiple of the configured FIFO threshold. In this case, the DMA draining feature is also used for transferring the additional bytes of the transfer (see the Draining Feature subsection (Section 11.2.13.4) for additional details).

In I2C Slave RX Mode, the Local Host can program the RX threshold with the desired value, and use the FIFO draining feature at the end of the I2C transfer to extract from the FIFO the remaining bytes if the threshold is not reached (see the Draining Feature subsection (Section 11.2.13.4) for additional details).

Note that in I2C Slave TX Mode, the TX FIFO threshold should be set to 1 (I2C_BUF.TXTRSH=0, default value), since the length of the transfer may not be known at configuration time. In this way, the interrupt (or accordingly, DMA) requests will be generated for each byte requested by the remote I2C master to be transferred over the I2C bus. This configuration will prevent the I2C core to request additional data from the CPU or from the DMA controller (using IRQ or DMA), data that will eventually not be extracted from the FIFO by the external master (which can use not acknowledge at any time to end the transfer). If the TX threshold is not set to 1, the module will generate an interrupt or assert a DMA only when the external master requests a byte and the FIFO is empty. However, in this case the TX FIFO will require to be cleared at the end of the transfer.

The I2C module offers the possibility to the user to clear the RX or TX FIFO. This is achieved through I2C_BUF.RXFIFO_CLR and I2C_BUF.TXFIFO_CLR registers, which act like software reset for the FIFOs. In DMA mode, these bits will also reset the DMA state machines.

The FIFO clearing feature can be used when the module is configured as a transmitter, the external receiver responds with a NACK in the middle of the transfer, and there is still data in TX FIFO waiting to be transferred.

On the Functional (I2C) domain, the thresholds can always be considered equal to 1. This means that the I2C Core can start transferring data on the I2C bus whenever it has data in the FIFOs (FIFO is not empty).

11.2.13.4 Draining Feature

The Draining Feature is implemented by the I2C core for handling the end of the transfers whose length is not a multiple of FIFO threshold value, and offers the possibility to transfer the remaining amount of bytes (since the threshold is not reached).

Note that this feature prevents the CPU or the DMA controller to attempt more FIFO accesses than necessary (for example, to generate at the end of a transfer a DMA RX request having in the FIFO less bytes than the configured DMA transfer length). Otherwise, an Access Error interrupt will be generated (see I2C_IRQSTATUS_RAW.AERR interrupt).

The Draining mechanism will generate an interrupt (I2C_IRQSTATUS_RAW.RDR or I2C_IRQSTATUS_RAW.XDR) at the end of the transfer informing the CPU that it needs to check the amount of data left to be transferred (I2C_BUFSTAT.TXSTAT or RXSTAT) and to enable the Draining Feature of the DMA controller if DMA mode is enabled (by re-configuring the DMA transfer length according to this value), or perform only the required number of data accesses, if DMA mode is disabled.

In receiving mode (master or slave), if the RX FIFO threshold is not reached but the transfer was ended on the I2C bus and there is still data left in the FIFO (less than the threshold), the receive draining interrupt (I2C_IRQSTATUS_RAW.RDR) will be asserted to inform the local host that it can read the amount of data in the FIFO (I2C_BUFSTAT.RXSTAT). The CPU will perform a number of data read accesses equal with RXSTAT value (if interrupt or polling mode) or re-configure the DMA controller with the required value in order to drain the FIFO.

In master transmit mode, if the TX FIFO threshold is not reached but the amount of data remained to be written in the FIFO is less than TXTRSH, the transmit draining interrupt (I2C_IRQSTATUS_RAW.XDR) will be asserted to inform the local host that it can write the amount of data remained to be written in the TX FIFO (I2C_BUFSTAT.TXSTAT). The CPU will need to write the required number of data bytes (specified by TXSTAT value) or re-configure the DMA controller with the required value in order to transfer the last bytes to the FIFO.

Note that in master mode, the CPU can alternatively skip the checking of TXSTAT and RXSTAT values since it can obtain internally this information (by computing DCOUNT modulo TX/RXTRSH).

The draining feature is default disabled, and it can be enabled using I2C_IRQENABLE_SET.XDR_IE or I2C_IRQENABLE_SET.RDR_IE registers (default disabled) only for the transfers with length not equal with the threshold value.

11.2.14 How to Program I2C

11.2.14.1 Module Configuration Before Enabling the Module

1. Program the prescaler to obtain an approximately 12-MHz I2C module clock (I2C_PSC = x; this value is to be calculated and is dependent on the System clock frequency).
2. Program the I2C clock to obtain 100 Kbps or 400 Kbps (SCLL = x and SCLH = x; these values are to be calculated and are dependent on the System clock frequency).
3. Configure its own address (I2C_OA = x) - only in case of I2C operating mode (F/S mode).
4. Take the I2C module out of reset (I2C_CON:I2C_EN = 1).

11.2.14.2 Initialization Procedure

1. Configure the I2C mode register (I2C_CON) bits.
2. Enable interrupt masks (I2C_IRQENABLE_SET), if using interrupt for transmit/receive data.
3. Enable the DMA (I2C_BUF and I2C_DMA/RX/TX/ENABLE_SET) and program the DMA controller - only in case of I2C operating mode (F/S mode), if using DMA for transmit/receive data.

11.2.14.3 Configure Slave Address and DATA Counter Registers

In master mode, configure the slave address (I2C_SA = x) and the number of bytes associated with the transfer (I2C_CNT = x).

11.2.14.4 Initiate a Transfer

Poll the bus busy (BB) bit in the I2C status register (I2C_IRQSTATUS_RAW). If it is cleared to 0 (bus not busy), configure START/STOP (I2C_CON: STT / I2C_CON: STP condition to initiate a transfer) - only in case of I2C operating mode (F/S mode).

11.2.14.5 Receive Data

Poll the receive data ready interrupt flag bit (RRDY) in the I2C status register (I2C_IRQSTATUS_RAW), use the RRDY interrupt (I2C_IRQENABLE_SET.RRDY_IE set) or use the DMA RX (I2C_BUF.RDMA_EN set together with I2C_DMARXENABLE_SET) to read the receive data in the data receive register (I2C_DATA). Use draining feature (I2C_IRQSTATUS_RAW.RDR enabled by I2C_IRQENABLE_SET.RDR_IE)) if the transfer length is not equal with FIFO threshold.

11.2.14.6 Transmit Data

Poll the transmit data ready interrupt flag bit (XRDY) in the I2C status register (I2C_IRQSTATUS_RAW), use the XRDY interrupt (I2C_IRQENABLE_SET.XRDY_IE set) or use the DMA TX (I2C_BUF.XDMA_EN set together with I2C_DMATXENABLE_SET) to write data into the data transmit register (I2C_DATA). Use draining feature (I2C_IRQSTATUS_RAW.XDR enabled by I2C_IRQENABLE_SET.XDR_IE)) if the transfer length is not equal with FIFO threshold.

11.3 I2C Registers

Table 11-3 lists the registers for the I2C module. For the base address of these registers, see Table 1-12.

NOTE: All bits defined as reserved must be written by software with 0s, for preserving future compatibility. When read, any reserved bit returns 0. Also, note that it is good software practice to use complete mask patterns for setting or testing individually bit fields within a register.

Table 11-3. I2C Registers

Address Offset	Acronym	Register Name	Section
00h	I2C_REVNB_LO	Module Revision Register (low bytes)	Section 11.3.1
04h	I2C_REVNB_HI	Module Revision Register (high bytes)	Section 11.3.2
10h	I2C_SYSC	System Configuration Register	Section 11.3.3
20h	I2C_EOI	I2C End of Interrupt Register	Section 11.3.4
24h	I2C_IRQSTATUS_RAW	I2C Status Raw Register	Section 11.3.5
28h	I2C_IRQSTATUS	I2C Status Register	Section 11.3.6
2Ch	I2C_IRQENABLE_SET	I2C Interrupt Enable Set Register	Section 11.3.7
30h	I2C_IRQENABLE_CLR	I2C Interrupt Enable Clear Register	Section 11.3.8
34h	I2C_WE	I2C Wakeup Enable Register	Section 11.3.9
38h	I2C_DMARXENABLE_SET	Receive DMA Enable Set Register	Section 11.3.10
3Ch	I2C_DMATXENABLE_SET	Transmit DMA Enable Set Register	Section 11.3.11
40h	I2C_DMARXENABLE_CLR	Receive DMA Enable Clear Register	Section 11.3.12
44h	I2C_DMATXENABLE_CLR	Transmit DMA Enable Clear Register	Section 11.3.13
48h	I2C_DMARXWAKE_EN	Receive DMA Wakeup Register	Section 11.3.14
4Ch	I2C_DMATXWAKE_EN	Transmit DMA Wakeup Register	Section 11.3.15
90h	2C_SYSS	System Status Register	Section 11.3.16
94h	I2C_BUF	Buffer Configuration Register	Section 11.3.17
98h	I2C_CNT	Data Counter Register	Section 11.3.18
9Ch	I2C_DATA	Data Access Register	Section 11.3.19
A4h	I2C_CON	I2C Configuration Register	Section 11.3.20
A8h	I2C_OA	I2C Own Address Register	Section 11.3.21
ACh	I2C_SA	I2C Slave Address Register	Section 11.3.22
B0h	I2C_PSC	I2C Clock Prescaler Register	Section 11.3.23
B4h	I2C_SCLL	I2C SCL Low Time Register	Section 11.3.24
B8h	I2C_SCLH	I2C SCL High Time Register	Section 11.3.25
BCh	I2C_SYSTEST	System Test Register	Section 11.3.26
C0h	I2C_BUFSTAT	I2C Buffer Status Register	Section 11.3.27
C4h	I2C_OA1	I2C Own Address 1 Register	Section 11.3.28
C8h	I2C_OA2	I2C Own Address 2 Register	Section 11.3.29
CCh	I2C_OA3	I2C Own Address 3 Register	Section 11.3.30
D0h	I2C_ACTOA	Active Own Address Register	Section 11.3.31
D4h	I2C_SBLOCK	I2C Clock Blocking Enable Register	Section 11.3.32

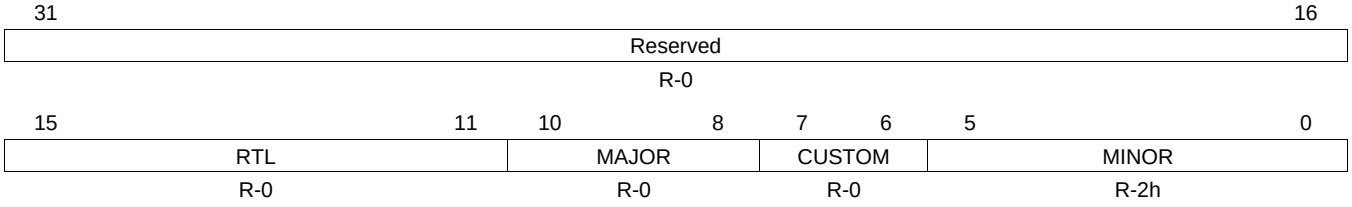
11.3.1 I2C_REVNB_LO Register (Module Revision LOW BYTES)

This read-only register contains the hard-coded revision number of the module. A write to this register has no effect.

NOTE:

- I2C controller with interrupt using interrupt vector register (I2C_IV) is revision 1.x.
- I2C controller with interrupt using status register bits (I2C_IRQSTATUS_RAW) is revision 2.x.

Figure 11-14. I2C_REVNB_LO Register (Module Revision) (LOW BYTES)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-4. I2C_REVNB_LO Register (Module Revision) (LOW BYTES) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-11	RTL	0-1Fh	RTL version.
10-8	MAJOR	0-7h	Major Revision. This field changes when there is a major feature change. This field does not change due to bug fix, or minor feature change.
7-6	CUSTOM	0-3h	Indicates a special version for a particular device. Consequence of use may avoid use of standard Chip Support Library (CSL) / Drivers. 0 if non-custom.
5-0	MINOR	0-3Fh	Minor Revision This field changes when features are scaled up or down. This field does not change due to bug fix, or major feature change.

11.3.2 I2C_REVNB_HI Register (HIGH BYTES) (Module Revision)

A reset has no effect on the value returned.

Figure 11-15. I2C_REVNB_HI Register (HIGH BYTES) (Module Revision)

31	16
Reserved	
R-0	

15	14	13	12	11	0
SCHEME		Reserved		FUNC	
R-1		R-1		R-40h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-5. I2C_REVNB_HI Register (HIGH BYTES) (Module Revision) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-14	SCHEME	0-3h	Used to distinguish between old Scheme and current. Spare bit to encode future schemes.
13-12	Reserved	0-3h	Reads return 1h
11-0	FUNC	0-FFFh	Function: Indicates a software compatible module family

11.3.3 I2C_SYSC Register (System Configuration)

This register allows controlling various parameters of the peripheral interface.

Figure 11-16. I2C_SYSC Register (System Configuration)

31	16
Reserved	
R-0	

15 10	9 8	7 5	4 3	2	1	0
Reserved	CLKACTIVITY	Reserved	IDLEMODE	ENAWAKEUP	SRST	AUTOIDLE
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-6. I2C_SYSC Register (System Configuration) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-8	CLKACTIVITY		Clock Activity selection bits. Those bits (one bit for each clock signal present on the boundary of the module) are set to 1 to disable external clock gating mechanism in Idle Mode.
		0h	Both clocks can be cut off
		1h	Only Interface/OCP clock must be kept active; system clock can be cut off
		2h	Only system clock must be kept active; Interface/OCP clock can be cut off
		3h	Both clocks must be kept active
			Values after reset are low (for both 2 bits).
			Note: If the System (functional) Clock is cut-off, the module will assert a WakeUp event when it will asynchronously detect a Start Condition on the I2C Bus. Note that in this case the first transfer will not be taken into account by the module (NACK will be detected by the external master).
7-5	Reserved	0	Reads return 0h.
4-3	IDLEMODE		Idle Mode selection bits. These two bits are used to select one of the idle mode operation mechanisms.
		0h	Force Idle mode
		1h	No Idle mode
		2h	Smart Idle mode
		3h	smartidle_wakeup
			Value after reset is 00 (Force Idle).
2	ENAWAKEUP		Enable Wakeup control bit. When this bit is set to 1, the module enables its own wakeup mechanism.
		0	Wakeup mechanism is disabled
		1	Wakeup mechanism is enabled
			Value after reset is low.
1	SRST		SoftReset bit. When this bit is set to 1, entire module is reset as for the hardware reset. This bit is automatically cleared to 0 by the core and it is only reset by the hardware reset. During reads, it always returns 0.
		0	Normal mode
		1	The module is reset
			Value after reset is low.
0	AUTOIDLE		Autoidle bit. When this bit is set to 1, the module activates its own idle mode mechanism. By evaluating its internal state, the module can decide to gate part of his internal clock tree in order to improve the overall power consumption.
		0	Auto Idle mechanism is disabled
		1	Auto Idle mechanism is enabled
			Value after reset is high.

11.3.4 I2C_EOI Register (I2C End of Interrupt)

Software End-Of-Interrupt: Allows the generation of further pulses on the interrupt line, if an new interrupt event is pending, when using the pulsed output. Unused when using the level interrupt line (depending on module integration).

Figure 11-17. I2C_EOI Register (I2C End of Interrupt)

31	1	0
Reserved		LINE_NUMBER
R-0		W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-7. I2C_EOI Register (I2C End of Interrupt) Field Descriptions

Bit	Field	Description
31-1	Reserved	Reserved
0	LINE_NUMBER	Software End Of Interrupt (EOI) control. Write number of interrupt output.

11.3.5 I2C_IRQSTATUS_RAW Register (I2C Status Raw)

This register provides core status information for interrupt handling, showing all active events (enabled and not enabled). The fields are read-write. Writing a 1 to a bit will set it to 1, that is, trigger the IRQ (mostly for debug). Writing a 0 will have no effect, that is, the register value will not be modified. Only enabled, active events will trigger an actual interrupt request on the IRQ output line.

Figure 11-18. I2C_IRQSTATUS_RAW Register (I2C Status Raw)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	XDR	RDR	BB	ROVR	XUDF	AAS	BF
R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
AERR	STC	GC	XRDY	RRDY	ARDY	NACK	AL
R/W-0	R/W-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-8. I2C_IRQSTATUS_RAW Register (I2C Status Raw) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Write 0s for future compatibility. Read returns 0.
14	XDR		Transmit draining IRQ status. I2C Master Transmit mode only. This read/clear only bit is set to 1 when the module is configured as a master transmitter, the TX FIFO level is below the configured threshold (TXTRSH) and the amount of data still to be transferred is less than TXTRSH. When this bit is set to 1 by the core, CPU must read the I2C_BUFSTAT.TXSTAT register in order to check the amount of data that need to be written in the TX FIFO. Then, according to the mode set (DMA or interrupt), the CPU can enable the DMA draining feature of the DMA controller with the number of data bytes to be transferred (I2C_BUFSTAT.TXSTAT), or generate write data accesses according to this value (IRQ mode). The interrupt needs to be cleared after the DMA controller was reconfigured (if DMA mode enabled), or before generating data accesses to the FIFO (if IRQ mode enabled). If the corresponding interrupt was enabled, an interrupt is signaled to the local host. The CPU can also poll this bit. For more details about TDR generation, refer to the FIFO Management subsection. The CPU can only clear this bit by writing a 1 into this register. A write 0 has no effect.
		0	Transmit draining inactive
		1	Transmit draining enabled
			Value after reset is low.
13	RDR		Receive draining IRQ status. I2C Receive mode only. This read/clear only bit is set to 1 when the module is configured as a receiver, a stop condition was received on the bus and the RX FIFO level is below the configured threshold (RXTRSH). When this bit is set to 1 by the core, CPU must read the I2C_BUFSTAT.RXSTAT register in order to check the amount of data left to be transferred from the FIFO. Then, according to the mode set (DMA or interrupt), the CPU needs to enable the draining feature of the DMA controller with the number of data bytes to be transferred (I2C_BUFSTAT.RXSTAT), or generate read data accesses according to this value (IRQ mode). The interrupt needs to be cleared after the DMA controller was reconfigured (if DMA mode enabled), or before generating data accesses to the FIFO (if IRQ mode enabled). If the corresponding interrupt was enabled, an interrupt is signaled to the local host. The CPU can also poll this bit. For more details about RDR generation, refer to the FIFO Management subsection. The CPU can only clear this bit by writing a 1 into this register. A write 0 has no effect.
		0	Receive draining inactive
		1	Receive draining enabled
			Value after reset is low.

Table 11-8. I2C_IRQSTATUS_RAW Register (I2C Status Raw) Field Descriptions (continued)

Bit	Field	Value	Description
12	BB		This read-only bit indicates the state of the serial bus. In slave mode, on reception of a start condition, the device sets BB to 1. BB is cleared to 0 after reception of a stop condition. In master mode, the software controls BB. To start a transmission with a start condition, MST, TRX, and STT must be set to 1 in the I2C_CON register. To end a transmission with a stop condition, STP must be set to 1 in the I2C_CON register. When BB = 1 and STT = 1, a restart condition is generated.
		0	Bus is free
		1	Bus is occupied
			Value after reset is low.
11	ROVR		Receive overrun status. Writing into this bit has no effect. I2C receive mode only. This read-only bit indicates whether the receiver has experienced overrun. Overrun occurs when the shift register is full and the receive FIFO is full. An overrun condition does not result in a data loss; the peripheral is just holding the bus (low on SCL) and prevents other bytes from being received. ROVR is set to 1 when the I2C has recognized an overrun. ROVR is clear when reading I2C_DATA register, or when resetting the I2C (I2C_CON:I2C_EN = 0).
		0	Normal operation
		1	Receiver overrun
			Value after reset is low.
10	XUDF		Transmit underflow status. Writing into this bit has no effect. I2C transmit mode only. This read-only bit indicates whether the transmitter has experienced underflow. In master transmit mode, underflow occurs when the shift register is empty, the transmit FIFO is empty, and there are still some bytes to transmit (DCOUNT M 0). In slave transmit mode, underflow occurs when the shift register is empty, the transmit FIFO is empty, and there are still some bytes to transmit (read request from external I2C master). XUDF is set to 1 when the I2C has recognized an underflow. The core holds the line till the underflow cause has disappeared. XUDF is clear when writing I2C_DATA register or resetting the I2C (I2C_CON:I2C_EN = 0).
		0	Normal operation
		1	Transmit underflow
			Value after reset is low.
9	AAS		Address recognized as slave IRQ status. I2C mode only. This read only bit is set to 1 by the device when it has recognized its own slave address (or one of the alternative own addresses), or an address of all zeros (8 bits). When this bit is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled. This bit can be cleared in 2 ways: • If the interrupt was enabled, it will be cleared by writing 1 into this register (writing 0 has no effect) • If the interrupt was not enabled, the AAS bit is reset to 0 by restart or stop
		0	No action
		1	Address recognized
			Value after reset is low.
8	BF		I2C mode only. This read only bit is set to 1 by the device when the I2C bus became free (after a transfer is ended on the bus – stop condition detected). This interrupt informs the Local Host that it can initiate its own I2C transfer on the bus. When this bit is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled. The CPU can only clear this bit by writing a 1 into this register. Writing 0 has no effect.
		0	No action
		1	Bus Free
			Value after reset is low.

Table 11-8. I2C_IRQSTATUS_RAW Register (I2C Status Raw) Field Descriptions (continued)

Bit	Field	Value	Description
7	AERR		Access Error IRQ status. I2C mode only. This read/clear only bit is set to 1 by the device if an Interface/OCP write access is performed to I2C_DATA while the TX FIFO is full or if an Interface/OCP read access is performed to the I2C_DATA while the RX FIFO is empty. Note that, when the RX FIFO is empty, a read access will return to the previous read data value. When the TX FIFO is full, a write access is ignored. In both events, the FIFO pointers will not be updated. When this bit is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled. The CPU can only clear this bit by writing a 1 into this register. Writing 0 has no effect.
		0	No action
		1	Access Error
			Value after reset is low.
6	STC		Start Condition IRQ status. I2C mode only. This read/clear only bit is set to 1 by the device if previously the module was in idle mode and a start condition was asynchronously detected on the I2C Bus and signalized with an Wakeup (if the I2C_SYSC.ClockActivity allows the system clock to be cut-off). When the Active Mode will be restored and the interrupt generated, this bit will indicate the reason of the wakeup. **Note 1:** The corresponding interrupt for this bit should be enabled only if the module was configured to allow the possibility of cutting-off the system clock while in Idle State (I2C_SYSC.ClockActivity = 00 or 01). **Note 2:** The first transfer (corresponding to the detected start condition) will be lost (not taken into account by the module) and it will be used only for generating the WakeUp enable for restoring the Active Mode of the module. On the I2C line, the external master which generated the transfer will detect this behavior as a not acknowledge to the address phase and will possibly restart the transfer. The CPU can only clear this bit by writing a 1 into this register. Writing 0 has no effect.
		0	No action
		1	Start Condition detected
			Value after reset is low.
5	GC		General call IRQ status. Set to '1' by core when General call address detected and interrupt signaled to MPUSS. Write '1' to clear. I2C mode only. This read/clear only bit is set to 1 by the device if it detects the address of all zeros (8 bits) (general call). When this bit is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled. The CPU can only clear this bit by writing a 1 into this register. Writing 0 has no effect. **Note:** When this bit is set to 1, AAS also reads as 1.
		0	No general call detected
		1	General call address detected
			Value after reset is low.
4	XRDY		Transmit data ready IRQ status. Set to '1' by core when transmitter and when new data is requested. When set to '1' by core, an interrupt is signaled to MPUSS. Write '1' to clear. Transmit mode only (I2C mode). This read/clear only bit (XRDY) is set to 1 when the I2C peripheral is a master or slave transmitter, the CPU needs to send data through the I2C bus, and the module (transmitter) requires new data to be served. Note that a master transmitter requests new data if the FIFO TX level is below the threshold (TXTRSH) and the required amount of data remained to be transmitted (I2C_BUFSTAT.TXSTAT) is greater than the threshold. A slave transmitter requests new data when the FIFO TX level is below the threshold (if TXTRSH > 1), or anytime there is a read request from external master (for each acknowledge received from the master), if TXTRSH = 1. When this bit is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled. The CPU can also poll this bit (refer to the FIFO Management subsection for details about XRDY generation). The CPU can only clear this bit by writing a 1 into this register. Writing 0 has no effect. **Note:** If the DMA transmit mode is enabled (I2C_BUF.XDMA_EN is set, together with I2C_DMATXENABLE_SET), this bit is forced to 0 and no interrupt will be generated; instead, a DMA TX request to the main DMA controller of the system is generated.
		0	Transmission ongoing
		1	Transmit data ready
			Value after reset is low.

Table 11-8. I2C_IRQSTATUS_RAW Register (I2C Status Raw) Field Descriptions (continued)

Bit	Field	Value	Description
3	RRDY		Receive mode only (I2C mode). This read/clear only RRDY is set to 1 when the RX FIFO level is above the configured threshold (RXTRSH). When this bit is set to 1 by the core, CPU is able to read new data from the I2C_DATA register. If the corresponding interrupt was enabled, an interrupt is signaled to the local host. The CPU to read the received data in I2C_DATA register can also poll this bit (refer to the FIFO Management subsection for details about RRDY generation). The CPU can only clear this bit by writing a 1 into this register. A write 0 has no effect. If the DMA receive mode is enabled (I2C_BUF.RDMA_EN is set, together with I2C_DMARXENABLE_SET), this bit is forced to 0 and no interrupt will be generated; instead a DMA RX request to the main DMA controller of the system is generated.
		0	Receive FIFO threshold not reached
		1	Receive data ready for read (RX FIFO threshold reached)
			Value after reset is low.
2	ARDY		I2C mode only. This read/clear only bit, when set to 1, indicates that the previously programmed data and command (receive or transmit, master or slave) has been performed and status bit has been updated. The CPU uses this flag to let it know that the I2C registers are ready to be accessed again. The CPU can only clear this bit by writing a 1 into this register. A write 0 has no effect.
			Mode / **Others** / **ARDY Set Condition**: I2C Master transmit / STP = 1 / DCOUNT = 0 I2C Master receive / STP = 1 / DCOUNT = 0 and receiver FIFO empty I2C Master transmit / STP = 0 / DCOUNT passed 0 I2C Master receive / STP = 0 / DCOUNT passed 0 and receiver FIFO empty I2C Master transmit / - / Stop or restart condition received from master I2C Slave receive / - / Stop or restart condition and receiver FIFO empty
		0	No action
		1	Access ready
			Value after reset is low.
1	NACK		No acknowledgement IRQ status. Bit is set when No Acknowledge has been received, an interrupt is signaled to MPUSS. Write '1' to clear this bit. I2C mode only. The read/clear only No Acknowledge flag bit is set when the hardware detects No Acknowledge has been received. When a NACK event occurs on the bus, this bit is set to 1, the core automatically ends the transfer and clears the MST/STP bits in the I2C_CON register and the I2C becomes a slave. Clearing the FIFOs from remaining data might be required. The CPU can only clear this bit by writing a 1 into this register. Writing 0 has no effect.
		0	Normal operation
		1	Not Acknowledge detected
			Value after reset is low.
0	AL		Arbitration lost IRQ status. This bit is automatically set by the hardware when it loses the Arbitration in master transmit mode, an interrupt is signaled to MPUSS. During reads, it always returns 0. I2C mode only. The read/clear only Arbitration Lost flag bit is set to 1 when the device (configured in master mode) detects it has lost an arbitration (in Address Phase). This happens when two or more masters initiate a transfer on the I2C bus almost simultaneously or when the I2C attempts to start a transfer while BB (bus busy) is 1. When this is set to 1 due to arbitration lost, the core automatically clears the MST/STP bits in the I2C_CON register and the I2C becomes a slave receiver. The CPU can only clear this bit by writing a 1 to this register. Writing 0 has no effect.
		0	Normal operation
		1	Arbitration lost detected
			Value after reset is low.

11.3.6 I2C_IRQSTATUS Register (I2C Status)

This register provides core status information for interrupt handling, showing all active and enabled events and masking the others. The fields are read-write. Writing a 1 to a bit will clear it to 0, that is, clear the IRQ. Writing a 0 will have no effect, that is, the register value will not be modified. Only enabled, active events will trigger an actual interrupt request on the IRQ output line.

For all the internal fields of the I2C_IRQSTATUS register, the descriptions given in the I2C_IRQSTATUS_RAW subsection are valid.

Figure 11-19. I2C_IRQSTATUS Register (I2C Status)

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	XDR	RDR	BB	ROVR	XUDF	AAS	BF
R-0	R/W1C-0	R/W1C-0	R/W-0	R/W-0	R/W1C-0	R/W1C-0	R/W1C-0
7	6	5	4	3	2	1	0
AERR	STC	GC	XRDY	RRDY	ARDY	NACK	AL
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-9. I2C_IRQSTATUS Register (I2C Status) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Write 0s for future compatibility. Read returns 0.
14	XDR		Transmit draining IRQ enabled status.
		0	Transmit draining inactive
		1	Transmit draining enabled
13	RDR		Receive draining IRQ enabled status.
		0	Receive draining inactive
		1	Receive draining enabled
12	BB		Bus busy enabled status.Writing into this bit has no effect.
		0	Bus is free
		1	Bus is occupied
11	ROVR		Receive overrun enabled status.Writing into this bit has no effect.
		0	Normal operation
		1	Receiver overrun
10	XUDF		Transmit underflow enabled status.Writing into this bit has no effect.
		0	Normal operation
		1	Transmit underflow
9	AAS		Address recognized as slave IRQ enabled status.
		0	No action
		1	Address recognized
8	BF		Bus Free IRQ enabled status.
		0	No action
		1	Bus free
7	AERR		Access Error IRQ enabled status.
		0	No action
		1	Access error
6	STC		Start Condition IRQ enabled status.
		0	No action
		1	Start condition detected

Table 11-9. I2C_IRQSTATUS Register (I2C Status) Field Descriptions (continued)

Bit	Field	Value	Description
5	GC		General call IRQ enabled status. Set to '1' by core when General call address detected and interrupt signaled to MPUSS. Write '1' to clear.
		0	No general call detected
		1	General call address detected
4	XRDY		Transmit data ready IRQ enabled status. Set to '1' by core when transmitter and when new data is requested. When set to '1' by core, an interrupt is signaled to MPUSS. Write '1' to clear.
		0	Transmision ongoing
		1	Transmit data ready
3	RRDY		Receive data ready IRQ enabled status. Set to '1' by core when receiver mode, a new data is able to be read. When set to '1' by core, an interrupt is signaled to MPUSS. Write '1' to clear.
		0	No data available
		1	Receive data available
2	ARDY		Register access ready IRQ enabled status. When set to '1' it indicates that previous access has been performed and registers are ready to be accessed again. An interrupt is signaled to MPUSS. Write '1' to clear.
		0	Module busy
		1	Access ready
1	NACK		No acknowledgement IRQ enabled status. Bit is set when No Acknowledge has been received, an interrupt is signaled to MPUSS. Write '1' to clear this bit.
		0	Normal operation
		1	Not Acknowledge detected
0	AL		Arbitration lost IRQ enabled status. This bit is automatically set by the hardware when it loses the Arbitration in master transmit mode, an interrupt is signaled to MPUSS. During reads, it always returns 0.
		0	Normal operation
		1	Arbitration lost detected

11.3.7 I2C_IRQENABLE_SET Register (I2C Interrupt Enable Set)

All 1-bit fields enable a specific interrupt event to trigger an interrupt request. Writing a 1 to a bit will enable the field. Writing a 0 will have no effect, that is, the register value will not be modified.

For all the internal fields of the I2C_IRQENABLE_SET register, the descriptions given in the I2C_IRQSTATUS_RAW subsection are valid.

Figure 11-20. I2C_IRQENABLE_SET Register (I2C Interrupt Enable Set)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	XDR_IE	RDR_IE	Reserved	ROVR	XUDF	AAS_IE	BF_IE
R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
AERR_IE	STC_IE	GC_IE	XRDY_IE	RRDY_IE	ARDY_IE	NACK_IE	AL_IE
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-10. I2C_IRQENABLE_SET Register (I2C Interrupt Enable Set) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Write 0s for future compatibility. Read returns 0.
14	XDR_IE		Transmit draining interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[XDR].
		0	Transmit draining interrupt disabled
		1	Transmit draining interrupt enabled
13	RDR_IE		Receive draining interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[RDR].
		0	Receive draining interrupt disabled
		1	Receive draining interrupt enabled
12	Reserved	0	Reserved
11	ROVR		Receive overrun enable set.
		0	Receive overrun interrupt disabled
		1	Receive draining interrupt enabled
10	XUDF		Transmit underflow enable set.
		0	Transmit underflow interrupt disabled
		1	Transmit underflow interrupt enabled
9	AAS_IE		Addressed as slave interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[AAS].
		0	Addressed as slave interrupt disabled
		1	Addressed as slave interrupt enabled
8	BF_IE		Bus free interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[BF].
		0	Bus free interrupt disabled
		1	Bus free interrupt enabled
7	AERR_IE		Access error interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[AERR].
		0	Access error interrupt disabled
		1	Access error interrupt enabled
6	STC_IE		Start condition interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[STC].
		0	Start condition interrupt disabled
		1	Start condition interrupt enabled

Table 11-10. I2C_IRQENABLE_SET Register (I2C Interrupt Enable Set) Field Descriptions (continued)

Bit	Field	Value	Description
5	GC_IE		General call interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[GC].
		0	General call interrupt disabled
		1	General call interrupt enabled
4	XRDY_IE		Transmit data ready interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[XRDY].
		0	Transmit data ready interrupt disabled
		1	Transmit data ready interrupt enabled
3	RRDY_IE		Receive data ready interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[RRDY]
		0	Receive data ready interrupt disabled
		1	Receive data ready interrupt enabled
2	ARDY_IE		Register access ready interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[ARDY].
		0	Register access ready interrupt disabled
		1	Register access ready interrupt enabled
1	NACK_IE		No acknowledgement interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[NACK].
		0	Not Acknowledge interrupt disabled
		1	Not Acknowledge interrupt enabled
0	AL_IE		Arbitration lost interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[AL].
		0	Arbitration lost interrupt disabled
		1	Arbitration lost interrupt enabled

11.3.8 I2C_IRQENABLE_CLR Register (I2C Interrupt Enable Clear)

All 1-bit fields clear a specific interrupt event. Writing a 1 to a bit will disable the interrupt field. Writing a 0 will have no effect, that is, the register value will not be modified.

For all the internal fields of the I2C_IRQENABLE_CLR register, the descriptions given in the I2C_IRQSTATUS_RAW subsection are valid.

Figure 11-21. I2C_IRQENABLE_CLR Register (I2C Interrupt Enable Clear)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	XDR_IE	RDR_IE	Reserved	ROVR	XUDF	AAS_IE	BF_IE
R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
AERR_IE	STC_IE	GC_IE	XRDY_IE	RRDY_IE	ARDY_IE	NACK_IE	AL_IE
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-11. I2C_IRQENABLE_CLR Register (I2C Interrupt Enable Clear) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Write 0s for future compatibility. Read returns 0.
14	XDR_IE		Transmit draining interrupt enable clear. Mask or unmask the interrupt signaled by bit in I2C_STAT[XDR].
		0	Transmit draining interrupt disabled
		1	Transmit draining interrupt enabled
13	RDR_IE		Receive draining interrupt enable clear. Mask or unmask the interrupt signaled by bit in I2C_STAT[RDR].
		0	Receive draining interrupt disabled
		1	Receive draining interrupt enabled
12	Reserved	0	Reserved
11	ROVR		Receive overrun enable clear.
		0	Receive overrun interrupt disabled
		1	Receive draining interrupt enabled
10	XUDF		Transmit underflow enable clear.
		0	Transmit underflow interrupt disabled
		1	Transmit underflow interrupt enabled
9	AAS_IE		Addressed as slave interrupt enable clear. Mask or unmask the interrupt signaled by bit in I2C_STAT[AAS].
		0	Addressed as slave interrupt disabled
		1	Addressed as slave interrupt enabled
8	BF_IE		Bus Free interrupt enable clear. Mask or unmask the interrupt signaled by bit in I2C_STAT[BF].
		0	Bus free interrupt disabled
		1	Bus free interrupt enabled
7	AERR_IE		Access error interrupt enable clear. Mask or unmask the interrupt signaled by bit in I2C_STAT[AERR].
		0	Access error interrupt disabled
		1	Access error interrupt enabled
6	STC_IE		Start condition interrupt enable clear. Mask or unmask the interrupt signaled by bit in I2C_STAT[STC].
		0	Start condition interrupt disabled
		1	Start condition interrupt enabled

Table 11-11. I2C_IRQENABLE_CLR Register (I2C Interrupt Enable Clear) Field Descriptions (continued)

Bit	Field	Value	Description
5	GC_IE		General call interrupt enable clear. Mask or unmask the interrupt signaled by bit in I2C_STAT[GC].
		0	General call interrupt disabled
		1	General call interrupt enabled
4	XRDY_IE		Transmit data ready interrupt enable clear. Mask or unmask the interrupt signaled by bit in I2C_STAT[XRDY].
		0	Transmit data ready interrupt disabled
		1	Transmit data ready interrupt enabled
3	RRDY_IE		Receive data ready interrupt enable set. Mask or unmask the interrupt signaled by bit in I2C_STAT[RRDY]
		0	Receive data ready interrupt disabled
		1	Receive data ready interrupt enabled
2	ARDY_IE		Register access ready interrupt enable clear. Mask or unmask the interrupt signaled by bit in I2C_STAT[ARDY].
		0	Register access ready interrupt disabled
		1	Register access ready interrupt enabled
1	NACK_IE		No acknowledgement interrupt enable clear. Mask or unmask the interrupt signaled by bit in I2C_STAT[NACK].
		0	Not Acknowledge interrupt disabled
		1	Not Acknowledge interrupt enabled
0	AL_IE		Arbitration lost interrupt enable clear. Mask or unmask the interrupt signaled by bit in I2C_STAT[AL].
		0	Arbitration lost interrupt disabled
		1	Arbitration lost interrupt enabled

11.3.9 I2C_WE Register (I2C Wakeup Enable)

Every 1-bit field in the I2C_WE register enables a specific (synchronous) IRQ request source to generate an asynchronous wakeup (on the appropriate swakeup line). When a bit location is set to 1 by the local host, a wakeup is signaled to the local host if the corresponding event is captured by the core of the I2C controller. Value after reset is low (all bits).

NOTE:

- There is no need for an Access Error WakeUp event, since this event occurs only when the module is in Active Mode (for Interface/OCP accesses to FIFO) and is signaled by an interrupt.
- With the exception of Start Condition WakeUp, which is asynchronously detected when the Functional clock is turned-off, all the other WakeUp events require the Functional (System) clock to be enabled.

Figure 11-22. I2C_WE Register (I2C Wakeup Enable)

31	30	29	28	27	26	25	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved	XDR_WE	RDR_WE	Reserved	ROVR_WE	XUDF_WE	AAS_WE	BF_WE
R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved	STC_WE	GC_WE	Reserved	DRDY_WE	ARDY_WE	NACK_WE	AL_WE
R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-12. I2C_WE Register (I2C Wakeup Enable) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14	XDR_WE		Transmit draining wakeup enable. This read/write bit is used to enable or disable wakeup signal generation when I2C module is in idle mode, the TX FIFO level is below the threshold and the amount of data left to be transferred is less than TXTRSH value. This allows for the module to inform the CPU that it can check the amount of data to be written to the FIFO.
		0	Transmit draining wakeup disabled
		1	Transmit draining wakeup enabled
13	RDR_WE		Receive draining wakeup enable. This read/write bit is used to enable or disable wakeup signal generation when I2C is in idle mode, configured as a receiver, and it has detected a stop condition on the bus but the RX FIFO threshold is not reached (but the FIFO is not empty). This allows for the module to inform the CPU that it can check the amount of data to be transferred from the FIFO.
		0	Receive draining wakeup disabled
		1	Receive draining wakeup enabled
12	Reserved	0	Reserved
11	ROVR_WE		Receive overrun wakeup enable
		0	Receive overrun wakeup disabled
		1	Receive overrun wakeup enabled
10	XUDF_WE		Transmit underflow wakeup enable
		0	Transmit underflow wakeup disabled
		1	Transmit underflow wakeup enabled

Table 11-12. I2C_WE Register (I2C Wakeup Enable) Field Descriptions (continued)

Bit	Field	Value	Description
9	AAS_WE		Address as slave IRQ wakeup enable. This read/write bit is used to enable or disable wakeup signal generation when I2C module is in idle mode, and external master addresses the I2C module as a slave. This allows for the module to inform the CPU that it can check which of the own addresses was used by the external master to access the I2C core.
		0	Addressed as slave wakeup disabled
		1	Addressed as slave wakeup enabled
8	BF_WE		Bus free IRQ wakeup enable. This read/write bit is used to enable or disable wakeup signal generation when I2C module is in idle mode and the I2C bus became free. This allows for the module to inform the CPU that it can initiate its own transfer on the I2C line.
		0	Bus Free wakeup disabled
		1	Bus Free wakeup enabled
7	Reserved	0	Reserved
6	STC_WE		Start condition IRQ wakeup set. This read/write bit is used to enable or disable wakeup signal generation when I2C module is in idle mode (with the functional clock inactive) and a possible start condition is detected on the I2C line. The STC WakeUp is generated only if the I2C_SYSC.ClockActivity field indicates that the functional clock can be disabled. Note that if the functional clock is not active, the start condition is asynchronously detected (no filtering and synchronization is used). For this reason, it is possible that the signalized start condition to be a glitch. If the functional clock cannot be disabled (I2C_SYSC.ClockActivity = 10 or 11), the programmer should not enable this wakeup, since the module has other synchronously detected WakeUp event that might be used to exit from idle mode, only if the detected transfer is accessing the I2C module.
		0	Start condition wakeup disabled
		1	Start condition wakeup enabled
5	GC_WE		General call IRQ wakeup enable. This read/write bit is used to enable or disable wakeup signal generation when I2C module is in idle mode and a general call is received on I2C line.
		0	General call wakeup disabled
		1	General call wakeup enabled
4	Reserved	0	Reserved
3	DRDY_WE		Receive/Transmit data ready IRQ wakeup enable. This read/write bit is used to enable or disable wakeup signal generation when I2C module is involved into a long transfer and no more registers accesses are performed on the interface (for example module are set in F/S I2C master transmitter mode and FIFO is full). If in the middle of such a transaction, the FIFO buffer needs more data to be transferred, CPU must be informed to write (in case of transmitter mode) or read (if receiver mode) in/from the FIFO.
		0	Transmit/receive data ready wakeup disabled
		1	Transmit/receive data ready wakeup enabled
2	ARDY_WE		Register access ready IRQ wakeup enable. This read/write bit is used to enable or disable wakeup signal generation when I2C module is involved into a long transfer and no more registers accesses are performed on the interface (for example the module is set in F/S I2C master transmitter mode and FIFO is full). If the current transaction is finished, the module needs to inform CPU about transmission completion.
		0	Register access ready wakeup disabled
		1	Register access ready wakeup enabled
1	NACK_WE		No acknowledgment IRQ wakeup enable. This read/write bit is used to enable or disable wakeup signal generation when I2C module is involved into a long transfer and no more registers accesses are performed on the interface (for example the module is set in F/S I2C master transmitter mode and FIFO is full). If in the middle of such of a transaction a Not Acknowledgment event is raised, the module needs to inform CPU about transmission error.
		0	Not Acknowledge wakeup disabled
		1	Not Acknowledge wakeup enabled

Table 11-12. I2C_WE Register (I2C Wakeup Enable) Field Descriptions (continued)

Bit	Field	Value	Description
0	AL_WE		Arbitration lost IRQ wakeup enable. This read/write bit is used to enable or disable wakeup signal generation when I2C module is configured as a master and it loses the arbitration. This wake up is very useful when the module is configured as a master transmitter, all the necessary data is provided in the FIFO Tx, STT is enabled and the module enters in Idle Mode. If the module loses the arbitration, an Arbitration Lost event is raised and the module needs to inform CPU about transmission error. **Note:** The AL wakeup must be enabled only for multimaster communication. If the AL_WE is not enabled and the scenario described above occurs, the module will not be able to inform the CPU about the state of the transfer and it will be blocked in an undetermined state.
		0	Arbitration lost wakeup disabled
		1	Arbitration lost wakeup enabled

11.3.10 I2C_DMARXENABLE_SET Register (Receive DMA Enable Set)

The 1-bit field enables a receive DMA request. Writing a 1 to this field will set it to 1. Writing a 0 will have no effect, that is, the register value is not modified. Note that the I2C_BUF.RDMA_EN field is the global (slave) DMA enabler, and that it is disabled by default. The I2C_BUF.RDMA_EN field should also be set to 1 to enable a receive DMA request.

Figure 11-23. I2C_DMARXENABLE_SET Register (Receive DMA Enable Set)

31	1	0
Reserved		DMARX_ENABLE_SET
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-13. I2C_DMARXENABLE_SET Register (Receive DMA Enable Set) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	DMARX_ENABLE_SET	0-1	Receive DMA channel enable set.

11.3.11 I2C_DMATXENABLE_SET Register (Transmit DMA Enable Set)

The 1-bit field enables a transmit DMA request. Writing a 1 to this field will set it to 1. Writing a 0 will have no effect, that is, the register value is not modified. Note that the I2C_BUF.XDMA_EN field is the global (slave) DMA enabler, and that it is disabled by default. The I2C_BUF.XDMA_EN field should also be set to 1 to enable a transmit DMA request.

Figure 11-24. I2C_DMATXENABLE_SET Register (Transmit DMA Enable Set)

31	1	0
Reserved		DMATX_ENABLE_SET
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-14. I2C_DMATXENABLE_SET Register (Transmit DMA Enable Set) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	DMATX_ENABLE_SET	0-1	Transmit DMA channel enable set.

11.3.12 I2C_DMARXENABLE_CLR Register (Receive DMA Enable Clear)

The 1-bit field disables a receive DMA request. Writing a 1 to a bit will clear it to 0. Another result of setting to '1' the DMARX_ENABLE_CLEAR field, is the reset of the DMA RX request and wakeup lines. Writing a 0 will have no effect, that is, the register value is not modified.

Figure 11-25. I2C_DMARXENABLE_CLR Register (Receive DMA Enable Clear)

31	1	0
Reserved		DMARX_ENABLE_CLEAR
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-15. I2C_DMARXENABLE_CLR Register (Receive DMA Enable Clear) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	DMARX_ENABLE_CLEAR	0-1	Receive DMA channel enable clear.

11.3.13 I2C_DMATXENABLE_CLR Register (Transmit DMA Enable Clear)

The 1-bit field disables a transmit DMA request. Writing a 1 to a bit will clear it to 0. Another result of setting to '1' the DMATX_ENABLE_CLEAR field, is the reset of the DMA TX request and wakeup lines. Writing a 0 will have no effect, that is, the register value is not modified.

Figure 11-26. I2C_DMATXENABLE_CLR Register (Transmit DMA Enable Clear)

31	1	0
Reserved		DMATX_ENABLE_CLEAR
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-16. I2C_DMATXENABLE_CLR Register (Transmit DMA Enable Clear) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	DMATX_ENABLE_CLEAR	0-1	Transmit DMA channel enable clear.

11.3.14 I2C_DMARXWAKE_EN Register (Receive DMA Wakeup)

All 1-bit fields enable a specific (synchronous) DMA request source to generate an asynchronous wakeup (on the appropriate swakeup line). Note that the I2C_SYSC.ENAWAKEUP field is the global (slave) wakeup enabler, and that it is disabled by default.

Figure 11-27. I2C_DMARXWAKE_EN Register (Receive DMA Wakeup)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	XDR	RDR	Reserved	ROVR	XUDF	AAS	BF
R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
Reserved	STC	GC	Reserved	DRDY	ARDY	NACK	AL
R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 11-17. I2C_DMARXWAKE_EN Register (Receive DMA Wakeup) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14	XDR		Transmit draining wakeup set.
		0	Transmit draining interrupt disabled
		1	Transmit draining interrupt enabled
13	RDR		Receive draining wakeup set.
		0	Receive draining interrupt disabled
		1	Receive draining interrupt enabled
12	Reserved	0	Reserved
11	ROVR		Receive overrun wakeup set.
		0	Receive overrun interrupt disabled
		1	Receive draining interrupt enabled
10	XUDF		Transmit underflow wakeup set.
		0	Transmit underflow interrupt disabled
		1	Transmit underflow interrupt enabled
9	AAS		Address as slave IRQ wakeup set.
		0	Addressed as slave interrupt disabled
		1	Addressed as slave interrupt enabled
8	BF		Bus free IRQ wakeup set.
		0	Bus free wakeup disabled
		1	Bus free wakeup enabled
7	Reserved	0	Reserved
6	STC		Start condition IRQ wakeup set.
		0	Start condition wakeup disabled
		1	Start condition wakeup enabled
5	GC		General call IRQ wakeup set.
		0	General call wakeup disabled
		1	General call wakeup enabled
4	Reserved	0	Reserved
3	DRDY		Receive/transmit data ready IRQ wakeup set.
		0	Transmit/receive data ready wakeup disabled
		1	Transmit/receive data ready wakeup enabled

Table 11-17. I2C_DMARXWAKE_EN Register (Receive DMA Wakeup) Field Descriptions (continued)

Bit	Field	Value	Description
2	ARDY		Register access ready IRQ wakeup set.
		0	Register access ready wakeup disabled
		1	Register access ready wakeup enabled
1	NACK		No acknowledgment IRQ wakeup set.
		0	Not Acknowledge wakeup disabled
		1	Not Acknowledge wakeup enabled
0	AL		Arbitration lost IRQ wakeup set.
		0	Arbitration lost wakeup disabled
		1	Arbitration lost wakeup enabled

11.3.15 I2C_DMATXWAKE_EN Register (Transmit DMA Wakeup)

All 1-bit fields enable a specific (synchronous) DMA request source to generate an asynchronous wakeup (on the appropriate swakeup line). Note that the I2C_SYSC.ENAWAKEUP field is the global (slave) wakeup enabler, and that it is disabled by default.

Figure 11-28. I2C_DMATXWAKE_EN Register (Transmit DMA Wakeup)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	XDR	RDR	Reserved	ROVR	XUDF	AAS	BF
R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
Reserved	STC	GC	Reserved	DRDY	ARDY	NACK	AL
R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 11-18. I2C_DMATXWAKE_EN Register (Transmit DMA Wakeup) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14	XDR		Transmit draining wakeup set.
		0	Transmit draining interrupt disabled
		1	Transmit draining interrupt enabled
13	RDR		Receive draining wakeup set.
		0	Receive draining interrupt disabled
		1	Receive draining interrupt enabled
12	Reserved	0	Reserved
11	ROVR		Receive overrun wakeup set.
		0	Receive overrun interrupt disabled
		1	Receive draining interrupt enabled
10	XUDF		Transmit underflow wakeup set.
		0	Transmit underflow interrupt disabled
		1	Transmit underflow interrupt enabled
9	AAS		Address as slave IRQ wakeup set.
		0	Addressed as slave interrupt disabled
		1	Addressed as slave interrupt enabled
8	BF		Bus free IRQ wakeup set.
		0	Bus free wakeup disabled
		1	Bus free wakeup enabled
7	Reserved	0	Reserved
6	STC		Start condition IRQ wakeup set.
		0	Start condition wakeup disabled
		1	Start condition wakeup enabled
5	GC		General call IRQ wakeup set.
		0	General call wakeup disabled
		1	General call wakeup enabled
4	Reserved	0	Reserved
3	DRDY		Receive/transmit data ready IRQ wakeup set.
		0	Transmit/receive data ready wakeup disabled
		1	Transmit/receive data ready wakeup enabled

Table 11-18. I2C_DMATXWAKE_EN Register (Transmit DMA Wakeup) Field Descriptions (continued)

Bit	Field	Value	Description
2	ARDY		Register access ready IRQ wakeup set.
		0	Register access ready wakeup disabled
		1	Register access ready wakeup enabled
1	NACK		No acknowledgment IRQ wakeup set.
		0	Not Acknowledge wakeup disabled
		1	Not Acknowledge wakeup enabled
0	AL		Arbitration lost IRQ wakeup set.
		0	Arbitration lost wakeup disabled
		1	Arbitration lost wakeup enabled

11.3.16 I2C_SYSS Register (System Status)

Figure 11-29. I2C_SYSS Register (System Status)

31	1	0
Reserved		RDONE
R-0		R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-19. I2C_SYSS Register (System Status) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	RDONE		Reset done bit. This read-only bit indicates the state of the reset in case of hardware reset, global software reset (I2C_SYSC.SRST) or partial software reset (I2C_CON.I2C_EN). The module must receive all its clocks before it can grant a reset-completed status.
		0	Internal module reset in ongoing
		1	Reset completed
			Value after reset is low.

11.3.17 I2C_BUF Register (Buffer Configuration)

This read/write register enables DMA transfers and allows the configuration of FIFO thresholds for the FIFO management (see the FIFO Management subsection).

Figure 11-30. I2C_BUF Register (Buffer Configuration)

31						16
Reserved						
R-0						
15	14	13	8	7	6	5	0
RDMA_EN	RXFIFO_CLR	RXTRSH		XDMA_EN	TXFIFO_CLR	TXTRSH	
R/W-0	R/W-0	R/W-0		R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-20. I2C_BUF Register (Buffer Configuration) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RDMA_EN		Receive DMA channel enable. When this bit is set to 1, the receive DMA channel is enabled and the receive data ready status bit (I2C_IRQSTATUS_RAW: RRDY) is forced to 0 by the core.
		0	Receive DMA channel disabled
		1	Receive DMA channel enabled
			Value after reset is low.
14	RXFIFO_CLR		Receive FIFO clear. When set, receive FIFO is cleared (hardware reset for RX FIFO generated). This bit is automatically reset by the hardware. During reads, it always returns 0.
		0	Normal mode
		1	Rx FIFO is reset
			Value after reset is low.
13-8	RXTRSH	0-3Fh	Threshold value for FIFO buffer in RX mode. The receive threshold value is used to specify the trigger level for data receive transfers. The value is specified from the Interface/OCP point of view.
		0	Receive Threshold value = 1
		1	Receive Threshold value = 2
		-	
		-	
		3Fh	Receive Threshold value = 64
			Value after reset is 00h.
			For the FIFO management description, see the FIFO Management subsection.
			Note1: programmed threshold cannot exceed the actual depth of the FIFO.
			Note2: the threshold must not be changed while a transfer is in progress (after STT was configured or after the module was addressed as a slave).
7	XDMA_EN		Transmit DMA channel enable. When this bit is set to 1, the transmit DMA channel is enabled and the transmit data ready status (I2C_IRQSTATUS_RAW: XRDY) bit is forced to 0 by the core.
		0	Transmit DMA channel disabled
		1	Transmit DMA channel enabled
			Value after reset is low.
6	TXFIFO_CLR		Transmit FIFO clear. When set, transmit FIFO is cleared (hardware reset for TX FIFO). This bit is automatically reset by the hardware. During reads, it always returns 0.
		0	Normal mode
		1	Tx FIFO is reset
			Value after reset is low.

Table 11-20. I2C_BUF Register (Buffer Configuration) Field Descriptions (continued)

Bit	Field	Value	Description
5-0	TXTRSH	0-3Fh	Threshold value for FIFO buffer in TX mode. The Transmit Threshold value is used to specify the trigger level for data transfers. The value is specified from the OCP point of view.
		0	Transmit Threshold value = 1
		1	Transmit Threshold value = 2
		-	
		-	
		3Fh	Transmit Threshold value = 64
			Value after reset is 00h
			Note1: programmed threshold cannot exceed the actual depth of the FIFO.
			Note2: the threshold must not be changed while a transfer is in progress (after STT was configured or after the module was addressed as a slave).

11.3.18 I2C_CNT Register (Data Counter)

CAUTION

During an active transfer phase (between STT having been set to 1 and reception of ARDY), no modification must be done in this register. Changing it may result in an unpredictable behavior.

This read/write register is used to control the numbers of bytes in the I2C data payload.

Figure 11-31. I2C_CNT Register (Data Counter)

31	16	15	0
Reserved		DCOUNT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-21. I2C_CNT Register (Data Counter) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	DCOUNT	0-FFFFh	Data count. I2C Master Mode only (receive or transmit; F/S). This 16-bit countdown counter decrements by 1 for every byte received or sent through the I2C interface. A write initializes DCOUNT to a saved initial value. A read returns the number of bytes that are yet to be received or sent. A read into DCOUNT returns the initial value only before a start condition and after a stop condition. When DCOUNT reaches 0, the core generates a stop condition if a stop condition was specified (I2C_CON.STP = 1) and the ARDY status flag is set to 1 in the I2C_IRQSTATUS_RAW register. Note that DCOUNT must not be reconfigured after I2C_CON.STT was enabled and before ARDY is received. **Note1:** In case of I2C mode of operation, if I2C_CON.STP = 0, then the I2C asserts SCL = 0 when DCOUNT reaches 0. The CPU can then reprogram DCOUNT to a new value and resume sending or receiving data with a new start condition (restart). This process repeats until the CPU sets to 1 the I2C_CON.STP bit. The ARDY flag is set each time DCOUNT reaches 0 and DCOUNT is reloaded to its initial value.
		0	Data counter = 65536 bytes (2^16)
		1	Data counter = 1 bytes
		-	
		-	
		FFFFh	Data counter = 65535 bytes (2^16 - 1) Values after reset are low (all 16 bits). **Note2:** Since for DCOUNT = 0, the transfer length is 65536, the module does not allow the possibility to initiate zero data bytes transfers.

11.3.19 I2C_DATA Register (Data Access)

This register is the entry point for the local host to read data from or write data to the FIFO buffer.

Figure 11-32. I2C_DATA Register (Data Access)

31	8	7	0
Reserved		DATA	
R-0		R/W-x	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 11-22. I2C_DATA Register (Data Access) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	DATA	0-FFh	Transmit/Receive data FIFO endpoint. When read, this register contains the received I2C data. When written, this register contains the byte value to transmit over the I2C data. In SYSTEST loop back mode (I2C_SYSTEST: TMODE = 11), this register is also the entry/receive point for the data. Values after reset are unknown (all 8-bits). **Note:** A read access, when the buffer is empty, returns the previous read data value. A write access, when the buffer is full, is ignored. In both events, the FIFO pointers are not updated and an Access Error (AERR) Interrupt is generated.

11.3.20 I2C_CON Register (I2C Configuration)

CAUTION

During an active transfer phase (between STT having been set to 1 and reception of ARDY), no modification must be done in this register (except STP enable). Changing it may result in an unpredictable behavior.

Figure 11-33. I2C_CON Register (I2C Configuration)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
I2C_EN	Reserved	OPMODE		STB	MST	TRX	XSA
R/W-0	R-0	R/W-0		R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
XOA0	XOA01	XOA02	XOA3	Reserved		STP	STT
R/W-0	R/W-0	R/W-0	R/W-0	R-0		R/W-0	R/W-0

LEGEND: RR/W-0/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-23. I2C_CON Register (I2C Configuration) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	I2C_EN		I2C module enable. When this bit is cleared to 0, the I2C controller is not enabled and reset. When 0, receive and transmit FIFOs are cleared and all status bits are set to their default values. All configuration registers (I2C_IRQENABLE_SET, I2C_IRQWAKE_SET, I2C_BUF, I2C_CNT, I2C_CON, I2C_OA, I2C_SA, I2C_PSC, I2C_SCLL and I2C_SCLH) are not reset, they keep their initial values and can be accessed. The CPU must set this bit to 1 for normal operation.
		0	Controller in reset. FIFO are cleared and status bits are set to their default value.
		1	Module enabled
			Value after reset is low.
14	Reserved	0	Reserved
13-12	OPMODE		Operation mode selection. These two bits select module operation mode.
		0h	I2C Fast/Standard mode
		1h	Reserved
		2h	Reserved
		3h	Reserved
			Value after reset is 00.
11	STB		Start byte mode (I2C master mode only). The start byte mode bit is set to 1 by the CPU to configure the I2C in start byte mode (I2C_SA = 0000 0001). See the Philips I2C spec for more details [1].
		0	Normal mode
		1	Start byte mode
			Value after reset is low.
10	MST		Master/slave mode (I2C mode only). When this bit is cleared, the I2C controller is in the slave mode and the serial clock (SCL) is received from the master device. When this bit is set, the I2C controller is in the master mode and generates the serial clock.
			Note: This bit is automatically cleared at the end of the transfer on a detected stop condition, in case of arbitration lost or when the module is configured as a master but addressed as a slave by an external master.
		0	Slave mode
		1	Master mode
			Value after reset is low.

Table 11-23. I2C_CON Register (I2C Configuration) Field Descriptions (continued)

Bit	Field	Value	Description
9	TRX		Transmitter/receiver mode (i2C master mode only). When this bit is cleared, the I2C controller is in the receiver mode and data on data line SDA is shifted into the receiver FIFO and can be read from I2C_DATA register. When this bit is set, the I2C controller is in the transmitter mode and the data written in the transmitter FIFO via I2C_DATA is shifted out on data line SDA.
		0	Receiver mode
		1	Transmitter mode
			Value after reset is low.
			The operating modes are defined as follows:
			MST / **TRX** / **Operating Modes**: 0 / x / Slave receiver; 0 / x / Slave transmitter; 1 / 0 / Master receiver; 1 / 1 / Master transmitter
8	XSA		Expand slave address. (I2C mode only). When set, this bit expands the slave address to 10-bit.
		0	7-bit address mode
		1	10-bit address mode
			Value after reset is low.
7	XOA0		Expand own address 0. (I2C mode only). When set, this bit expands the base own address (OA0) to 10-bit.
		0	7-bit address mode
		1	10-bit address mode
			Value after reset is low.
6	XOA1		Expand own address 1. (I2C mode only). When set, this bit expands the first alternative own address (OA1) to 10-bit.
		0	7-bit address mode
		1	10-bit address mode
			Value after reset is low.
5	XOA2		Expand own address 2. (I2C mode only). When set, this bit expands the second alternative own address (OA2) to 10-bit.
		0	7-bit address mode.
		1	10-bit address mode
			Value after reset is low.
4	XOA3		Expand own address 3. When set, this bit expands the third alternative own address (OA3) to 10-bit.
		0	7-bit address mode
		1	10-bit address mode
			Value after reset is low.
3-2	Reserved	0	Reserved
1	STP		Stop condition (I2C master mode only). This bit can be set to a 1 by the CPU to generate a stop condition. It is reset to 0 by the hardware after the stop condition has been generated. The stop condition is generated when DCOUNT passes 0.
			When this bit is not set to 1 before the end of the transfer (DCOUNT = 0), the stop condition is not generated and the SCL line is hold to 0 by the master, which can re-start a new transfer by setting the STT bit to 1.
		0	No action or stop condition detected
		1	Stop condition queried
			Value after reset is low

Table 11-23. I2C_CON Register (I2C Configuration) Field Descriptions (continued)

Bit	Field	Value	Description
0	STT		Start condition (I2C master mode only). This bit can be set to a 1 by the CPU to generate a start condition. It is reset to 0 by the hardware after the start condition has been generated. The start/stop bits can be configured to generate different transfer formats.
		0	No action or start condition detected
		1	Start condition queried
			Value after reset is low.

STT	STP	Conditions	Bus Activities
1	0	Start	S-A-D
0	1	Stop	P
1	1	Start-Stop (DCOUNT= n)	S-A-D..(n)..D-P
1	0	Start (DCOUNT= n)	S-A-D..(n)..D

Note: DCOUNT is data count value in I2C_CNT register.

11.3.21 I2C_OA Register (I2C Own Address)

CAUTION

During an active transfer phase (between STT having been set to 1 and reception of ARDY), no modification must be done in this register. Changing it may result in an unpredictable behavior.

This register is used to specify the module's base I2C 7-bit or 10-bit address (base own address).

Figure 11-34. I2C_OA Register (I2C Own Address)

31	10	9	0
Reserved		OA	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 11-24. I2C_OA Register (I2C Own Address) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-0	OA	0-3FFh	Own address. This field specifies either: • A 10-bit address coded on OA [9:0] when XOA (Expand Own Address, I2C_CON[7]) is set to 1. • A 7-bit address coded on OA [6:0] when XOA (Expand Own Address, I2C_CON[7]) is cleared to 0. In this case, OA [9:7] bits must be cleared to 000 by application software. Value after reset is low (all 10 bits).

11.3.22 I2C_SA Register (I2C Slave Address)

CAUTION

During an active transfer phase (between STT having been set to 1 and reception of ARDY), no modification must be done in this register. Changing it may result in an unpredictable behavior.

This register is used to specify the addressed I2C module 7-bit or 10-bit address (slave address).

Figure 11-35. I2C_SA Register (I2C Own Address)

31	10	9	0
Reserved		SA	
R-0		R/W-3FFh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-25. I2C_SA Register (I2C Slave Address) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-0	SA	0-3FFh	Slave address. This field specifies either: • A 10-bit address coded on SA [9:0] when XSA (Expand Slave Address, I2C_CON[8]) is set to 1. • A 7-bit address coded on SA [6:0] when XSA (Expand Slave Address, I2C_CON[8]) is cleared to 0. In this case, SA [9:7] bits must be cleared to 000 by application software. Value after reset is low (all 10 bits).

11.3.23 I2C_PSC Register (I2C Clock Prescaler)

CAUTION

During an active mode (I2C_EN bit in I2C_CON register is set to 1), no modification must be done in this register. Changing it may result in an unpredictable behavior.

This register is used to specify the internal clocking of the I2C peripheral core.

Figure 11-36. I2C_PSC Register (I2C Own Address)

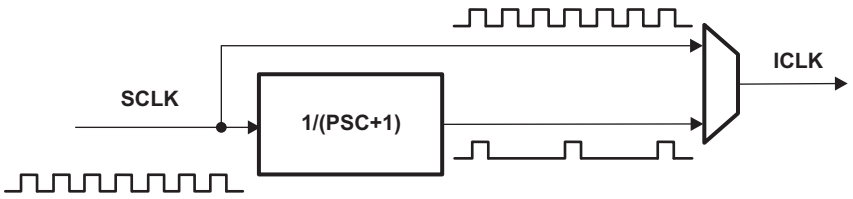
31	8	7	0
Reserved		PSC	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-26. I2C_PSC Register (I2C Clock Prescaler) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	PSC	0-FFh	Fast/Standard mode prescale sampling clock divider value. The core uses this 8-bit value to divide the system clock (SCLK) and generates its own internal sampling clock (ICLK) for Fast and Standard operation modes. The core logic is sampled at the clock rate of the system clock for the module divided by (PSC + 1). 0h Divide by 1 1h Divide by 2 - - FFh Divide by 256 Value after reset is low (all 8 bits).

Figure 11-37. Clock Divider



11.3.24 I2C_SCLL Register (I2C SCL Low Time)

CAUTION

During an active mode (I2C_EN bit in I2C_CON register is set to 1), no modification must be done in this register. Changing it may result in an unpredictable behavior.

This register is used to determine the SCL low time value when master.

Figure 11-38. I2C_SCLL Register (I2C SCL Low Time)

31	8	7	0
Reserved		SCLL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-27. I2C_SCLL Register (I2C SCL Low Time) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	SCLL	0-FFh	Fast/Standard mode SCL low time. I2C master mode only (FS). This 8-bit value is used to generate the SCL low time value (tLOW) when the peripheral is operated in master mode. t_{LOW} = (SCLL + 7) * I_{CLK} time period, Value after reset is low (all 8 bits).

11.3.25 I2C_SCLH Register (I2C SCL High Time)

CAUTION

During an active mode (I2C_EN bit in I2C_CON register is set to 1), no modification must be done in this register. Changing it may result in an unpredictable behavior.

This register is used to determine the SCL high time value when master.

Figure 11-39. I2C_SCLH Register (I2C SCL High Time)

31	8	7	0
Reserved		SCLH	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-28. I2C_SCLH Register (I2C SCL High Time) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	SCLH	0-FFh	Fast/Standard mode SCL low time. I2C master mode only (FS). This 8-bit value is used to generate the SCL high time value (tHIGH) when the peripheral is operated in master mode. t_{HIGH} = (SCLH + 5 + floor[t_rise/I_{CLK} Period]) * I_{CLK} time period. Value after reset is low (all 8 bits).

11.3.26 I2C_SYSTEST Register (System Test)

CAUTION

Never enable this register for normal I2C operation

This register is used to facilitate system-level tests by overriding some of the standard functional features of the peripheral. It allows testing of SCL counters, controlling the signals that connect to I/O pins, or creating digital loop-back for self-test when the module is configured in system test (SYSTEST) mode. It also provides stop/non-stop functionality in the debug mode.

Figure 11-40. I2C_SYSTEST Register (System Test)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
ST_EN	FREE	TMODE		SSB	Reserved		SCL_I_FUNC
R/W-0	R/W-0	R/W-0		R/W-0	R-0		R-1

7	6	5	4	3	2	1	0
SCL_O_FUNC	SDA_I_FUNC	SDA_O_FUNC	Reserved	SCL_I	SCL_O	SDA_I	SDA_O
R-1	R-1	R-1	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-29. I2C_SYSTEST Register (System Test) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	ST_EN		System test enable. This bit must be set to 1 to permit other system test registers bits to be set.
		0	Normal mode. All others bits in register are read only.
		1	System test enabled. Permit other system test registers bits to be set.
			Value after reset is low.
14	FREE		Free running mode (on breakpoint). This bit is used to determine the state of the I2C controller when a breakpoint is encountered in the HLL debugger.
			Note: This bit can be set independently of ST_EN value. FREE = 0: the I2C controller stops immediately after completion of the on-going bit transfer. Stopping the transfer is achieved by forcing the SCL line low. Note that in this case there will be no status register updates.
			FREE = 1: the I2C interface runs free. When Suspend indication will be asserted, there will be no accesses on the OCP Interface (the CPU is in debug mode) and consequently the FIFOs will reach full/empty state (according to RX or TX modes) and the I2C SDA line will be kept low. Note that the status registers will be updated, but no DMA, IRQ or WakeUp will be generated.
			The status registers likely to be updated in this mode are: I2C_IRQSTATUS_RAW.XRDY, I2C_IRQSTATUS_RAW.RRDY, I2C_IRQSTATUS_RAW.XUDF, I2C_IRQSTATUS_RAW.ROVR, I2C_IRQSTATUS_RAW.ARDY and I2C_IRQSTATUS_RAW.NACK.
		0	Stop mode (on breakpoint condition). If Master mode, it stops after completion of the on-going bit transfer. In slave mode, it stops during the phase transfer when 1 byte is completely transmitted/received.
		1	Free running mode
			Value after reset is low.

Table 11-29. I2C_SYSTEST Register (System Test) Field Descriptions (continued)

Bit	Field	Value	Description
13-12	TMODE		Test mode select. In normal functional mode (ST_EN = 0), these bits are don't care. They are always read as 00 and a write is ignored. In system test mode (ST_EN = 1), these bits can be set according to the following table to permit various system tests.
		0	Functional mode (default)
		1h	Reserved
		2h	Test of SCL counters (SCLL, SCLH, PSC). SCL provides a permanent clock with master mode.
		3h	Loop back mode select + SDA/SCL IO mode select
			Values after reset are low (2 bits). SCL counter test mode: in this mode, the SCL pin is driven with a permanent clock as if mastered with the parameters set in the I2C_PSC, I2C_SCLL, and I2C_SCLH registers. Loop back mode: in the master transmit mode only, data transmitted out of the I2C_DATA register (write action) is received in the same I2C_DATA register via an internal path through the FIFO buffer. The DMA and interrupt requests are normally generated if enabled. SDA/SCL IO mode: in this mode, the SCL IO and SDA IO are controlled via the I2C_SYSTEST [5:0] register bits.
11	SSB		Set status bits. Writing 1 into this bit also sets the 6 read/clear-only status bits contained in I2C_IRQSTATUS_RAW register (bits 5:0) to 1. Writing 0 into this bit doesn't clear status bits that are already set; only writing 1 into a set status bit can clear it (see I2C_IRQSTATUS_RAW operation). This bit must be cleared prior attempting to clear a status bit.
		0	No action
		1	Set all interrupt status bits to 1
			Value after reset is low.
10-9	Reserved	0	Reserved
8	SCL_I_FUNC		SCL line input value (functional mode). This read-only bit returns the logical state taken by the SCL line (either 1 or 0). It is active both in functional and test mode
		Read:0	Read 0 from SCL line
		Read:1	Read 1 from SCL line
			Value after reset is low.
7	SCL_O_FUNC		SCL line output value (functional mode). This read-only bit returns the value driven by the module on the SCL line (either 1 or 0). It is active both in functional and test mode.
		Read:0	Driven 0 on SCL line
		Read:1	Driven 1 on SCL line
			Value after reset is low.
6	SDA_I_FUNC		SDA line input value (functional mode). This read-only bit returns the logical state taken by the SDA line (either 1 or 0). It is active both in functional and test mode.
		Read:0	Read 0 from SDA line
		Read:1	Read 1 from SDA line
			Value after reset is low.
5	SDA_O_FUNC		SDA line output value (functional mode). This read-only bit returns the value driven by the module on the SDA line (either 1 or 0). It is active both in functional and test mode.
		Read:0	Driven 0 to SDA line
		Read:1	Driven 1 to SDA line
			Value after reset is low.
4	Reserved	0	Reserved
3	SCL_I		SCL line sense input value. In normal functional mode (ST_EN = 0), this read-only bit always reads 0. In system test mode (ST_EN = 1 & TMODE = 11), this read-only bit returns the logical state taken by the SCL line (either 1 or 0).
		Read:0	Read 0 from SCL line
		Read:1	Read 1 from SCL line
			Value after reset is low.

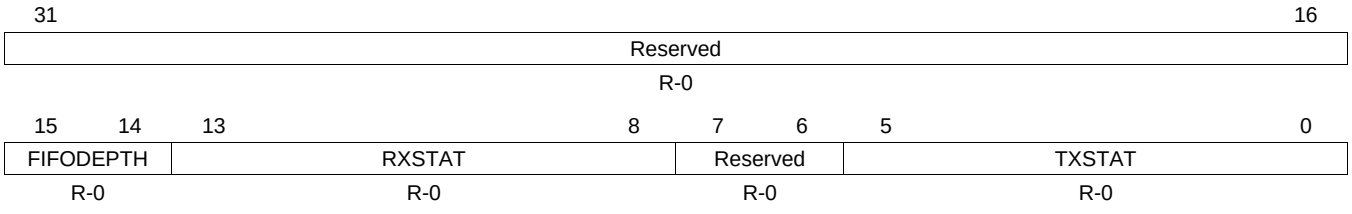
Table 11-29. I2C_SYSTEST Register (System Test) Field Descriptions (continued)

Bit	Field	Value	Description
2	SCL_O		SCL line drive output value. In normal functional mode (ST_EN = 0), this bit is don't care. It always reads 0 and a write is ignored. In system test mode (ST_EN = 1 & TMODE = 11), a 0 forces a low level on the SCL line and a 1 puts the I2C output driver to a high-impedance state.
		Read:0	Forces 0 on the SCL data line
		Read:1	SCL output driver in high-impedance state
			Value after reset is low.
1	SDA_I		SDA line sense input value. In normal functional mode (ST_EN = 0), this read-only bit always reads 0. In system test mode (ST_EN = 1 & TMODE = 11), this read-only bit returns the logical state taken by the SDA line (either 1 or 0).
		Read:0	Read 0 from SDA line
		Read:1	Read 1 from SDA line
			Value after reset is low.
0	SDA_O		SDA line drive output value. In normal functional mode (ST_EN = 0), this bit is don't care. It reads as 0 and a write is ignored. In system test mode (ST_EN = 1 & TMODE = 11), a 0 forces a low level on the SDA line and a 1 puts the I2C output driver to a high-impedance state.
		0	Write 0 to SDA line
		1	Write 1 to SDA line
			Value after reset is low.

11.3.27 I2C_BUFSTAT Register (I2C Buffer Status)

This read-only register reflects the status of the internal buffers for the FIFO management (see the FIFO Management subsection).

Figure 11-41. I2C_BUFSTAT Register (I2C Buffer Status)



LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 11-30. I2C_BUFSTAT Register (I2C Buffer Status) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-14	FIFODEPTH		Internal FIFO buffers depth. This read-only bit indicates the internal FIFO buffer depth.
		0	8-bytes FIFO
		1h	16-bytes FIFO
		2h	32-bytes FIFO
		3h	64-bytes FIFO
			Value after reset is given by the boundary module generic parameter.
13-8	RXSTAT	0-3Fh	RX buffer status. This read-only field indicates the number of bytes to be transferred from the FIFO at the end of the I2C transfer (when RDR is asserted). It corresponds to the level indication of the RX FIFO (number of written locations).
			Value after reset is 0.
7-6	Reserved	0	Reserved
5-0	TXSTAT	0-3Fh	TX buffer status. This read-only field indicates the number of data bytes still left to be written in the TX FIFO (it's equal with the initial value of I2C_CNT.DCOUNT minus the number of data bytes already written in the TX FIFO through the OCP Interface).
			Value after reset is equal with 0.

11.3.28 I2C_OA1 Register (OA1) (Own Address 1)

CAUTION

During an active transfer phase (between STT has been set to 1 and receiving of ARDY), no modification must be done in this register. Changing it may result in an unpredictable behavior.

This register is used to specify the first alternative I2C 7-bit or 10-bit address (own address 1 - OA1).

Figure 11-42. I2C_OA1 Register (OA1) (Own Address 1)

31	10	9	0
Reserved		0A1	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-31. I2C_OA1 Register (OA1) (Own Address 1) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-0	OA1	0-3FFh	Own address 1. This field specifies either: • A 10-bit address coded on OA1 [9:0] when XOA1 (Expand Own Address 1 - XOA1, I2C_CON[6]) is set to 1. • A 7-bit address coded on OA1 [6:0] when XOA1 (Expand Own Address 1 – XOA1, I2C_CON[6]) is cleared to 0. In this case, OA1 [9:7] bits must be cleared to 000 by application software. Value after reset is low (all 10 bits).

11.3.29 I2C_OA2 Register (I2C Own Address 2)

CAUTION

During an active transfer phase (between STT has been set to 1 and receiving of ARDY), no modification must be done in this register. Changing it may result in an unpredictable behavior.

This register is used to specify the first alternative I2C 7-bit or 10-bit address (own address 2 - OA2).

Figure 11-43. I2C_OA2 Register (I2C Own Address 2)

31	10	9	0
Reserved		0A2	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-32. I2C_OA2 Register (I2C Own Address 2) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-0	OA2	0-3FFh	Own address 2. This field specifies either: • A 10-bit address coded on OA2 [9:0] when XOA1 (Expand Own Address 2 - XOA2, I2C_CON[5]) is set to 1. • A 7-bit address coded on OA2 [6:0] when XOA2 (Expand Own Address 2 – XOA2, I2C_CON[5]) is cleared to 0. In this case, OA2 [9:7] bits must be cleared to 000 by application software. Value after reset is low (all 10 bits).

11.3.30 I2C_OA3 Register (I2C Own Address 3)

CAUTION

During an active transfer phase (between STT has been set to 1 and receiving of ARDY), no modification must be done in this register. Changing it may result in an unpredictable behavior.

This register is used to specify the first alternative I2C 7-bit or 10-bit address (own address 3 - OA3).

Figure 11-44. I2C_OA3 Register (I2C Own Address 3)

31	10	9	0
Reserved		0A3	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-33. I2C_OA3 Register (I2C Own Address 3) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-0	OA3	0-3FFh	Own address 2. This field specifies either: • A 10-bit address coded on OA3 [9:0] when XOA3 (Expand Own Address 3 - XOA3, I2C_CON[4]) is set to 1. • A 7-bit address coded on OA3 [6:0] when XOA1 (Expand Own Address 3 – XOA3, I2C_CON[4]) is cleared to 0. In this case, OA3 [9:7] bits must be cleared to 000 by application software. Value after reset is low (all 10 bits).

11.3.31 I2C_ACTOA Register (Active Own Address)

This read-only register is used to indicate which one of the module's four own addresses the external master used when addressing the module. The CPU can read this register when the AAS indication was activated. The indication is cleared at the end of the transfer.

Figure 11-45. I2C_ACTOA Register (Active Own Address)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		OA3_ACT	OA2_ACT	OA1_ACT	OA0_ACT
R-0		R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-34. I2C_ACTOA Register (Active Own Address) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	OA3_ACT		Own address 3 active. When a bit location is set to 1 by the core, it signalizes to the Local Host that an external master using the corresponding own address addressed the module.
		0	Own address inactive
		1	Own address active
			Value after reset is low.
2	OA2_ACT		Own address 2 active. When a bit location is set to 1 by the core, it signalizes to the Local Host that an external master using the corresponding own address addressed the module.
		0	Own address inactive
		1	Own address active
			Value after reset is low.
1	OA1_ACT		Own address 1 active. When a bit location is set to 1 by the core, it signalizes to the Local Host that an external master using the corresponding own address addressed the module.
		0	Own address inactive
		1	Own address active
			Value after reset is low.
0	OA0_ACT		Own address 0 active. When a bit location is set to 1 by the core, it signalizes to the Local Host that an external master using the corresponding own address addressed the module.
		0	Own address inactive
		1	Own address active
			Value after reset is low.

11.3.32 I2C_SBLOCK Register (I2C Clock Blocking Enable)

This read/write register controls the automatic blocking of I2C clock feature in slave mode. It is used for the Local Host to configure for which of the 4 own addresses, the core must block the I2C clock (keep SCL line low) right after the Address Phase, when it is addressed as a slave.

Figure 11-46. I2C_SBLOCK Register (I2C Clock Blocking Enable)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		OA3_EN	OA2_EN	OA1_EN	OA0_EN
R-0		R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-35. I2C_SBLOCK Register (I2C Clock Blocking Enable) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	OA3_EN		Enable I2C clock blocking for own address 3. When the CPU sets a bit location to 1, if an external master using the corresponding own address addresses the core, the core will block the I2C clock right after the address phase. For releasing the I2C clock the CPU must write '0' in the corresponding field.
		0	I2C clock released
		1	I2C clock blocked
			Value after reset is low.
2	OA2_EN		Enable I2C clock blocking for own address 2. When the CPU sets a bit location to 1, if an external master using the corresponding own address addresses the core, the core will block the I2C clock right after the address phase. For releasing the I2C clock the CPU must write '0' in the corresponding field.
		0	I2C clock released
		1	I2C clock blocked
			Value after reset is low.
1	OA1_EN		Enable I2C clock blocking for own address 1. When the CPU sets a bit location to 1, if an external master using the corresponding own address addresses the core, the core will block the I2C clock right after the address phase. For releasing the I2C clock the CPU must write '0' in the corresponding field.
		0	I2C clock released
		1	I2C clock blocked
			Value after reset is low.
0	OA0_EN		Enable I2C clock blocking for own address 0. When the CPU sets a bit location to 1, if an external master using the corresponding own address addresses the core, the core will block the I2C clock right after the address phase. For releasing the I2C clock the CPU must write '0' in the corresponding field.
		0	I2C clock released
		1	I2C clock blocked
			Value after reset is low.

ARM Interrupt Controller (AINTC)

This chapter describes the ARM interrupt controller (AINTC)

Topic **Page**

12.1 Introduction

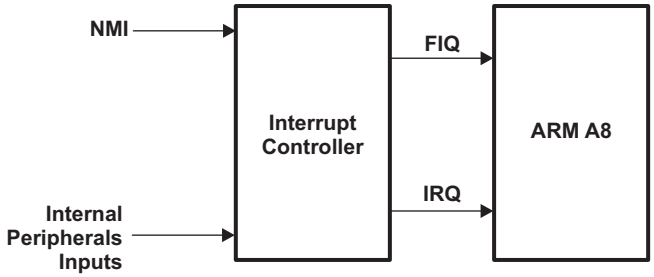
12.1.1 Overview

The Host ARM interrupt controller (AINTC) is responsible for prioritizing all service requests from the system peripherals and generating either nIRQ or nFIQ to the host. The type of the interrupt (nIRQ or nFIQ) and the priority of the interrupt inputs are programmable. It has the capability to handle up to 128 requests that can be prioritized as A8 nFIQ or nIRQ interrupt requests. The general features of the AINTC are:

- Up to 128 level sensitive interrupts inputs
- Individual priority for each interrupt input
- Each interrupt can be steered to nFIQ or nIRQ
- Independent priority sorting for nFIQ and nIRQ

Figure 12-1 shows the internal interrupt scheme.

Figure 12-1. Interrupt Controller



12.1.2 Functional Block Diagram

The interrupt controller processes incoming interrupts by masking and priority sorting to produce the interrupt signals for the processor to which it is attached. Figure 12-2 shows the top-level view of interrupt processing.

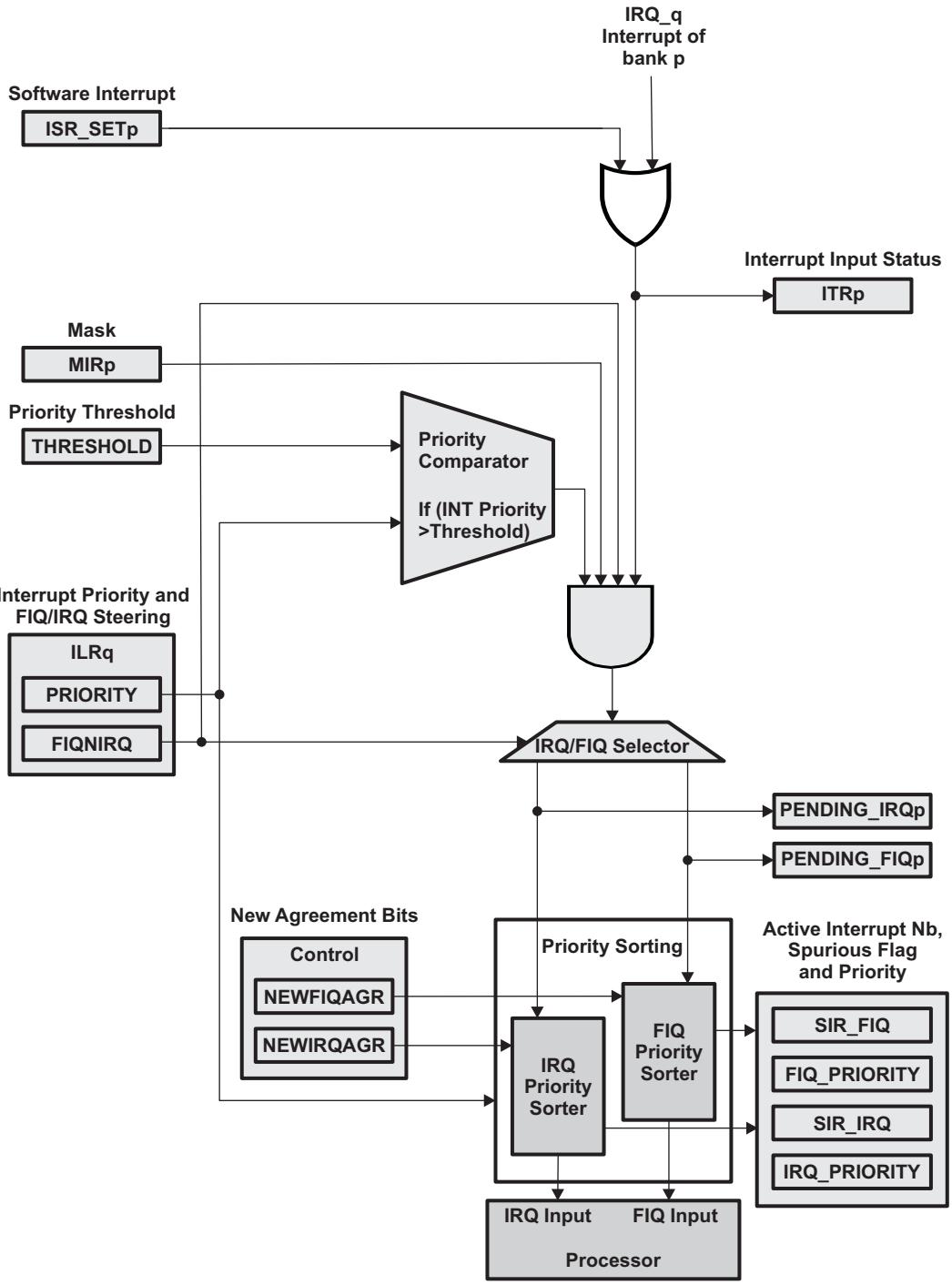
12.1.3 Environment

The sources of interrupt for AINTC are:

- External interrupts through NMI pin. Note that interrupts from external peripherals of SOC need to be routed through GPIO.
- Interrupt from peripherals inside the SOC and ARM A8.

All the unmasked interrupts can generate a wake up to the interrupt controller.

Figure 12-2. Interrupt Controller Block Diagram

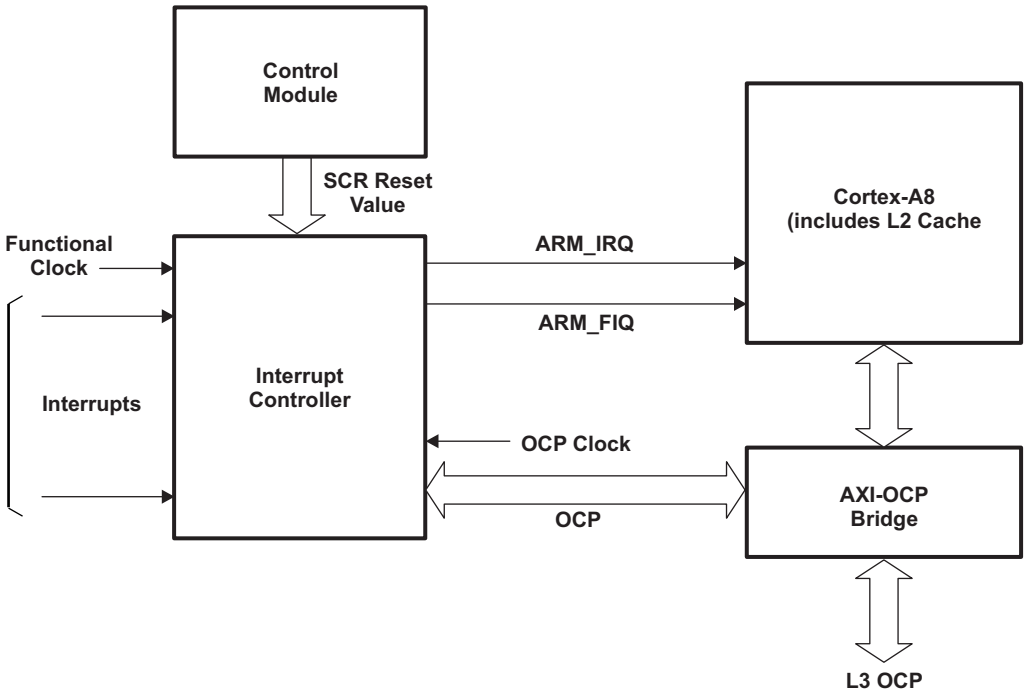


12.1.4 AINTC Integration

The ARM interrupt controller (AINTC) is the interface between the many incoming interrupts and the two interrupt inputs of the ARM A8. The two interrupts inputs to ARM are FIQ and IRQ. Figure 12-3 shows the integration of the interrupt controller in the ARM A8 subsystem.

The ARM A8 subsystem interrupt controller is directly connected to the ARM A8 by an ARM peripheral port. Consequently, the ARM A8 subsystem interrupt controller is accessible and visible only by the A8.

Figure 12-3. AINTC Integration



12.2 Architecture

This section discusses the architecture of the interrupt controller.

12.2.1 Clocking and Reset

12.2.1.1 ARM A8 Interrupt Controller Clocks

The ARM interrupt controller runs at half the rate of the ARM A8 functional clock. The interface clock, used for register access, runs at the rate of the interconnect bus clock (equal to the rate of the ARM A8 interface clock).

12.2.1.2 Hardware and Software Reset

Table 12-1 lists the ARM A8 subsystem interrupt controller resets.

Table 12-1. AINTC Resets

Type	Name	Source	Activation
Hardware	ARM A8 top-level Reset	PRCM	Active low
Software	SOFTRESET	INTCPS_SYSCONFIG[1]SOFTRESET bit, write 1 to clear. Read returns 0.	Active high

12.2.2 Interrupt Request Lines

Table 12-2 lists the incoming and outgoing interrupt lines of the AINTC.

For mapping of interrupt from various modules to the AINTC, refer to your device-specific data manual. Check availability of modules and features in a particular device. Ensure that interrupts of unavailable modules and features are masked in ARM A8 subsystem.

Table 12-2. AINTC Interrupt Inputs and Outputs

Type	Number	Name	Mapping	Comment
Interrupt request inputs	Up to 128	Refer to data manual	Refer to data manual	Inputs to AINTC module, source from various modules.
Interrupt request outputs	2	INTC_FIQ INTC_IRQ	INTC_FIQ INTC_IRQ	FIQ is fast interrupt to ARM IRQ is normal input to ARM A8

12.2.3 Interrupt Processing

12.2.3.1 Input Selection

The AINTC supports only level-sensitive incoming interrupt detection. A peripheral asserting an interrupt maintains it until software has handled the interrupt and instructed the peripheral to deassert the interrupt. A software interrupt is generated if the corresponding bit in the INTCPS_ISR_SETn register is set (register bank number: n = [0,1,2,3] for the MPU subsystem interrupt controller, 128 incoming interrupt lines are supported). The software interrupt clears when the corresponding bit in the INTCPS_ISR_CLEARn register is written. Typical use of this feature is software debugging.

12.2.3.2 Masking

12.2.3.2.1 Individual Masking

Detection of interrupts on each incoming interrupt line can be enabled or disabled independently by the INTCPS_MIRn interrupt mask register. In response to an unmasked incoming interrupt, the AINTC can generate one of two types of interrupt requests to the processor:

- IRQ: low-priority interrupt request
- FIQ: fast interrupt request

The type of interrupt request is determined by the INTCPS_ILRm[0] FIQNIRQ bit (m= [0,127]). The current incoming interrupt status before masking is readable from the INTCPS_ITRn register. After masking and IRQ/FIQ selection, and before priority sorting is done, the interrupt status is readable from the INTCPS_PENDING_IRQn and INTCPS_PENDING_FIQn registers.

12.2.3.2.2 Priority Masking

To enable faster processing of high-priority interrupts, a programmable priority masking threshold is provided (the INTCPS_THRESHOLD[7:0] PRIORITYTHRESHOLD field). This priority threshold allows preemption by higher priority interrupts; all interrupts of lower or equal priority than the threshold are masked. However, priority 0 can never be masked by this threshold; a priority threshold of 0 is treated the same way as priority 1. PRIORITY and PRIORITYTHRESHOLD fields values can be set between 0 and 7Fh; 0 is the highest priority and 7Fh is the lowest priority. When priority masking is not necessary, a priority threshold value of FFh disables the priority threshold mechanism. This value is also the reset default for backward compatibility with previous versions of the INTC.

12.2.3.3 Priority Sorting

A priority level (0 being the highest) is assigned to each incoming interrupt line. Both the priority level and the interrupt request type are configured by the INTCPS_ILRm register. If more than one incoming interrupt with the same priority level and interrupt request type occur simultaneously, the highest-numbered interrupt is serviced first. When one or more unmasked incoming interrupts are detected, the AINTC separates between IRQ and FIQ using the corresponding INTCPS_ILRm[0] FIQNIRQ bit. The result is placed in INTCPS_PENDING_IRQn or INTCPS_PENDING_FIQn. If no other interrupts are currently being processed, AINTC asserts IRQ/FIQ and starts the priority computation. Priority sorting for IRQ and FIQ can execute in parallel. Each IRQ/FIQ priority sorter determines the highest priority interrupt number. Each priority number is placed in the corresponding INTCPS_SIR_IRQ[6:0] ACTIVEIRQ field or INTCPS_SIR_FIQ[6:0] ACTIVEFIQ field. The value is preserved until the corresponding INTCPS_CONTROL NEWIRQAGR or NEWFIQAGR bit is set. Once the interrupting peripheral device has been serviced and the incoming interrupt deasserted, you must write to the appropriate NEWIRQAGR or NEWFIQAGR bit to indicate to the AINTC the interrupt has been handled. If there are any pending unmasked incoming interrupts for this interrupt request type, the AINTC restarts the appropriate priority sorter; otherwise, the IRQ or FIQ interrupt line is deasserted.

12.2.4 Module Power Saving

The AINTC provides an auto-idle function in its three clock domains:

- Interface clock
- Functional clock
- Synchronizer clock

The interface clock auto-idle power-saving mode is enabled if the INTCPS_SYSCONFIG[0] AUTOIDLE bit is set to 1. When this mode is enabled and there is no activity on the bus interface, the interface clock is disabled internally to the module, thus reducing power consumption. When there is new activity on the bus interface, the interface clock restarts without any latency penalty. After reset, this mode is disabled, by default. The functional clock auto-idle power-saving mode is enabled if the INTCPS_IDLE[0] FUNCIDLE bit is cleared to 0. When this mode is enabled and there is no active interrupt (IRQ or FIQ interrupt being processed or generated) or no pending incoming interrupt, the functional clock is disabled internally to the module, thus reducing power consumption. When a new unmasked incoming interrupt is detected, the functional clock restarts and the AINTC processes the interrupt. If this mode is disabled, the interrupt latency is reduced by one cycle. After reset, this mode is enabled, by default. The synchronizer clock allows external asynchronous interrupts to be resynchronized before they are masked. The synchronizer input clock has an auto-idle power-saving mode enabled if the INTCPS_IDLE[1] TURBO bit is set to 1. If the auto-idle mode is enabled, the standby power is reduced, but the IRQ or FIQ interrupt latency increases from four to six functional clock cycles. This feature can be enabled dynamically according to the requirements of the device. After reset, this mode is disabled, by default.

12.2.5 Error Handling

The following accesses will cause error:

- Privilege violation (attempt to access PROTECTION register in user mode or any register in user mode if Protection bit is set).
- Unsupported commands.

The following will NOT cause any error response:

- Access to a non-decoded address.
- Write to a read-only register.

12.2.6 Interrupt Latency

The IRQ/FIQ interrupt generation takes four AINTC functional clock cycles (plus or minus one cycle) if the INTCPS_IDLE[1] TURBO bit is cleared to 0. If the TURBO bit is set to 1, the interrupt generation takes six cycles, but power consumption is reduced while waiting for an interrupt. These latencies can be reduced by one cycle by disabling functional clock auto-idle (INTCPS_IDLE[0] FUNCIDLE bit set to 1), but power consumption is increased, so the benefit is minimal. To minimize interrupt latency when an unmasked interrupt occurs, the IRQ or FIQ interrupt is generated before priority sorting completion. The priority sorting takes 10 functional clock cycles, which is less than the minimum number of cycles required for the MPU to switch to the interrupt context after reception of the IRQ or FIQ event.

Any read of the INTCPS_SIR_IRQ or INTCPS_SIR_FIQ register during the priority sorting process stalls until priority sorting is complete and the relevant register is updated. However, the delay between the interrupt request being generated and the interrupt service routine being executed is such that priority sorting always completes before the INTCPS_SIR_IRQ or INTCPS_SIR_FIQ register is read.

12.3 Basic Programming Model

12.3.1 Initialization Sequence

1. Program the INTCPS_SYSCONFIG register: If necessary, enable the interface clock autogating by setting the AUTOIDLE bit.
2. Program the INTCPS_IDLE register: If necessary, disable functional clock autogating or enable synchronizer autogating by setting the FUNCIDLE bit or TURBO bit accordingly.
3. Program the INTCPS_ILRm register for each interrupt line: Assign a priority level and set the FIQNIRQ bit for an FIQ interrupt (by default, interrupts are mapped to IRQ and priority is 0 [highest]).
4. Program the INTCPS_MIRn register: Enable interrupts (by default, all interrupt lines are masked).

   To program the INTCPS_MIRn register, the INTCPS_MIR_SETn and INTCPS_MIR_CLEARn registers are provided to facilitate the masking, even if it is possible for backward-compatibility to write directly to the INTCPS_MIRn register.

12.3.2 AINTC Processing Sequence

After the INTCPS_MIRn and INTCPS_ILRm registers are configured to enable and assign priorities to incoming interrupts, the interrupt is processed as explained in the following subsections. IRQ and FIQ processing sequences are quite similar, the differences for the FIQ sequence are shown after a '/' character in the code below.

1. One or more unmasked incoming interrupts (M_IRQ_n signals) are received and IRQ or FIQ outputs (IRQ/FIQ) are not currently asserted.
2. If the INTCPS_ILRm[0] FIQNIRQ bit is cleared to 0, the MPU_INTC_IRQ output signal is generated. If the FIQNIRQ bit is set to 1, the MPU_INTC_FIQ output signal is generated.
3. The AINTC performs the priority sorting and updates the INTCPS_SIR_IRQ[6:0] ACTIVEIRQ /INTCPS_SIR_FIQ[6:0] ACTIVEFIQ field with the current interrupt number.
4. During priority sorting, if the IRQ/FIQ is enabled at the host processor side, the host processor automatically saves the current context and executes the ISR as follows.

   The ARM host processor automatically performs the following actions in pseudo code:

```
LR = PC + 4                    /* return link */
SPSR = CPSR                    /* Save CPSR before execution */
CPSR[5] = 0                    /* Execute in ARM state */
CPSR[7] = 1                    /* Disable IRQ */
CPSR[8] = 1                    /* Disable Imprecise Data Aborts */
CPSR[9] = CP15_reg1_EEbit      /* Endianness on exception entry */
if interrupt == IRQ then
CPSR[4:0] = 0b10010            /* Enter IRQ mode */
if high vectors configured then
PC = 0xFFFF0018
else
PC = 0x00000018                /* execute interrupt vector */
else if interrupt == FIQ then
CPSR[4:0] = 0b10001            /* Enter FIQ mode */
CPSR[6] = 1                    /* Disable FIQ */
if high vectors configured then
PC = 0xFFFF001C
else
PC = 0x0000001C                /* execute interrupt vector */
endif
```

5. The ISR saves the remaining context, identifies the interrupt source by reading the ACTIVEIRQ/ACTIVEFIQ field, and jumps to the relevant subroutine handler as follows:

> **CAUTION**
>
> The code in steps 5 and 7 is an assembly code compatible with ARM architecture V6 and V7. This code is developed for the Texas Instruments Code Composer Studio tool set. It is a draft version, only tested on an emulated environment.

```
 ;INTCPS_SIR_IRQ/INTCPS_SIR_FIQ register address
INTCPS_SIR_IRQ_ADDR/INTCPS_SIR_FIQ_ADDR .word 0x48200040/0x48200044
 ; ACTIVEIRQ bit field mask to get only the bit field
ACTIVEIRQ_MASK .equ 0x7F
_IRQ_ISR/_FIQ_ISR:
 ; Save the critical context
STMFD SP!, {R0-R12, LR} ; Save working registers and the Link register
MRS R11, SPSR ; Save the SPSR into R11
 ; Get the number of the highest priority active IRQ/FIQ
LDR R10, INTCPS_SIR_IRQ_ADDR/INTCPS_SIR_FIQ_ADDR
LDR R10, [R10] ; Get the INTCPS_SIR_IRQ/INTCPS_SIR_FIQ register
AND R10, R10, #ACTIVEIRQ_MASK ; Apply the mask to get the active IRQ number
 ; Jump to relevant subroutine handler
LDR PC, [PC, R10, lsl #2] ; PC base address points this instruction + 8
NOP ; To index the table by the PC
 ; Table of handler start addresses
.word IRQ0handler ;For IRQ0 of BANK0
.word IRQ1handler
.word IRQ2handler
```

6. The subroutine handler executes code specific to the peripheral generating the interrupt by handling the event and deasserting the interrupt condition at the peripheral side.

```
 ; IRQ0 subroutine
IRQ0handler:
 ; Save working registers
STMFD SP!, {R0-R1}
 ; Now read-modify-write the peripheral module status register
 ; to de-assert the M_IRQ_0 interrupt signal
 ; De-Assert the peripheral interrupt
MOV R0, #0x7 ; Mask for 3 flags
LDR R1, MODULE0_STATUS_REG_ADDR ; Get the address of the module Status Register
STR R0, [R1] ; Clear the 3 flags
 ; Restore working registers LDMFD SP!, {R0-R1}
 ; Jump to the end part of the ISR
B IRQ_ISR_end/FIQ_ISR_end
```

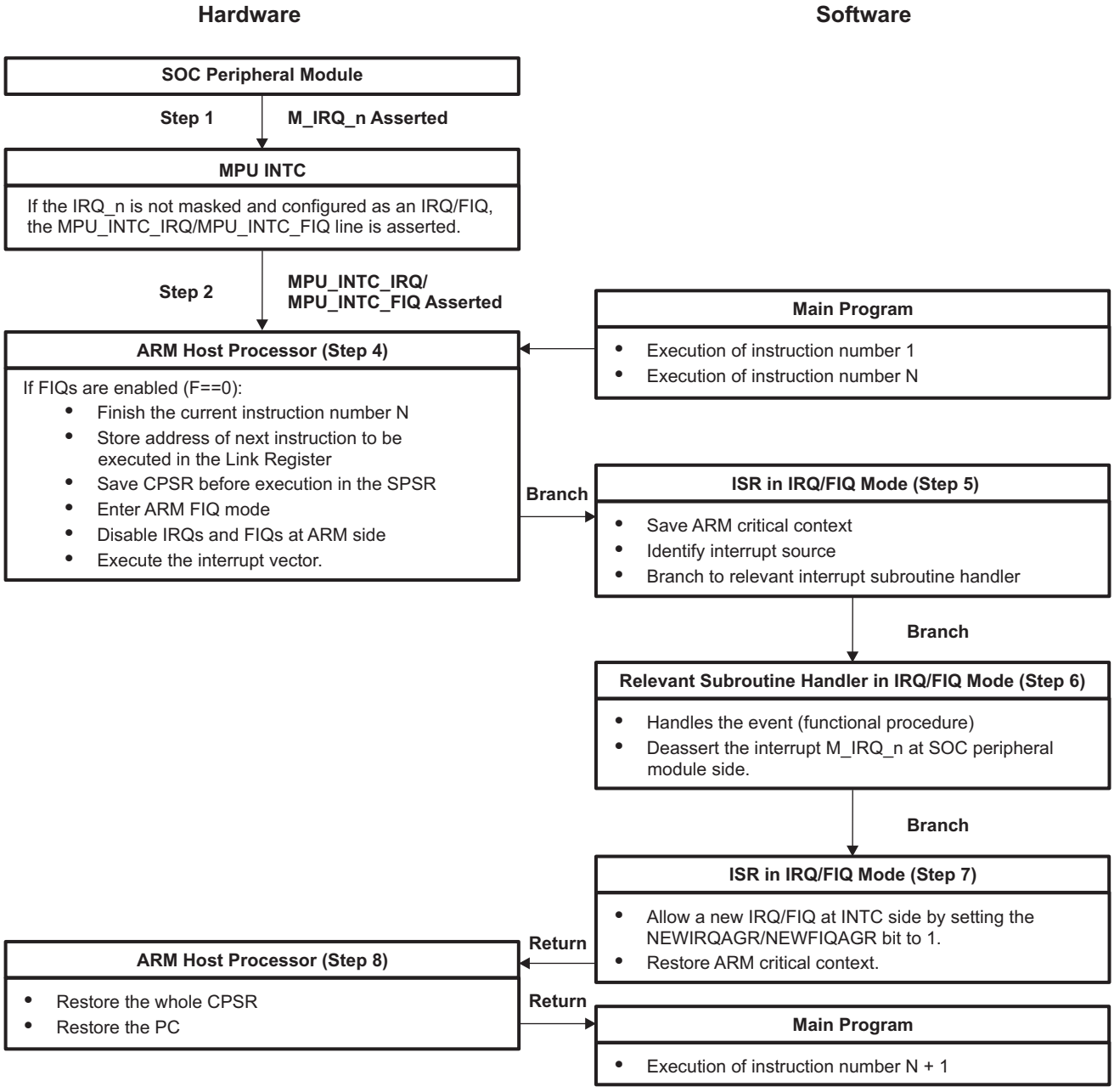
7. After the return of the subroutine, the ISR sets the NEWIRQAGR/NEWFIQAGR bit to enable the processing of subsequent pending IRQs/FIQs and to restore ARM context in the following code. Because the writes are posted on an Interconnect bus, to be sure that the preceding writes are done before enabling IRQs/FIQs, a Data Synchronization Barrier is used. This operation ensure that the IRQ/FIQ line is de-asserted before IRQ/FIQ enabling. After that, the AINTC processes any other pending interrupts or deasserts the IRQ/FIQ signal if there is no interrupt.

```
 ; INTCPS_CONTROL register address
INTCPS_CONTROL_ADDR .word 0x48200048;
NEWIRQAGR/NEWFIQAGR bit mask to set only the NEWIRQAGR/NEWFIQAGR bit
NEWIRQAGR_MASK/NEWFIQAGR_MASK .equ 0x01/0x02
IRQ_ISR_end/FIQ_ISR_end:
 ; Allow new IRQs/FIQs at INTC side
 ; The INTCPS_CONTROL register is a write only register so no need to write back others bits
MOV R0, #NEWIRQAGR_MASK/NEWFIQAGR_MASK ; Get the NEWIRQAGR/NEWFIQAGR bit position
LDR R1, INTCPS_CONTROL_ADDR
STR R0, [R1] ; Write the NEWIRQAGR/NEWFIQAGR bit to allow new IRQs/FIQ
 ; Data Synchronization Barrier MOV R0, #0
MCR P15, #0, R0, C7, C10, #4
 ; restore critical context
MSR SPSR, R11 ; Restore the SPSR from R11
LDMFD SP!, {R0-R12, LR} ; Restore working registers and  Link register
 ; Return after handling the interrupt
SUBS PC, LR, #4
8. After the ISR return, the ARM automatically restores its context as follows:
CPSR = SPSR
PC = LR
```

Figure 12-4 shows the IRQ/FIQ processing sequence from the originating device peripheral module to the main program interruption.

The priority sorting mechanism is frozen during an interrupt processing sequence. If an interrupt condition occurs during this time, the interrupt is not lost. It is sorted when the NEWIRQAGR/NEWFIQAGR bit is set (priority sorting is reactivated).

Figure 12-4. IRQ/FIQ Processing Sequence



12.3.3 AINTC Preemptive Processing Sequence

Preemptive interrupts, also called nested interrupts, can reduce the latencies for higher priority interrupts. A preemptive ISR can be suspended by a higher priority interrupt. Thus, the higher priority interrupt can be served immediately. Nested interrupts must be used carefully to avoid using corrupted data. Programmers must save corruptible registers and enable IRQ or FIQ at ARM side. IRQ and FIQ processing sequences are quite similar, the differences for the FIQ sequence are shown after a '/' character in the code below.

To enable IRQ/FIQ preemption by higher priority IRQs/FIQs, programers can follow this procedure to write the ISR.

At the beginning of an IRQ/FIQ ISR:

1. Save the ARM critical context registers.
2. Save the INTCPS_THRESHOLD PRIORITYTHRESHOLD field before modifying it.
3. Read the active interrupt priority in the INTCPS_IRQ_PRIORITY IRQPRIORITY/INTCPS_FIQ_PRIORITY FIQPRIORITY field and write it to the PRIORITYTHRESHOLD(1) field.
4. Read the active interrupt number in the INTCPS_SIR_IRQ[6:0] ACTIVEIRQ/INTCPS_SIR_FIQ[6:0] ACTIVEFIQ field to identify the interrupt source.
5. Write 1 to the appropriate INTCPS_CONTROL NEWIRQAGR and (2) NEWFIQAGR bit while an interrupt is still processing to allow only higher priority interrupts to preempt.
6. Because the writes are posted on an Interconnect bus, to be sure that the preceding writes are done before enabling IRQs/FIQs, a Data Synchronization Barrier is used. This operation ensure that the IRQ line is de-asserted before IRQ/FIQ enabling.
7. Enable IRQ/FIQ at ARM side.
8. Jump to the relevant subroutine handler.

The following sample code shows the previous steps:

CAUTION

The following code is an assembly code compatible with ARM architecture V6 and V7. This code is developed for the Texas Instruments Code Composer Studio tool set. It is a draft version, only tested on an emulated environment.

```
 ; bit field mask to get only the bit field
ACTIVEPRIO_MASK .equ 0x7F
_IRQ_ISR:
 ; Step 1 : Save the critical context
STMFD SP!, {R0-R12, LR} ; Save working registers
MRS R11, SPSR ; Save the SPSR into R11
 ; Step 2 : Save the INTCPS_THRESHOLD register into R12
LDR R0, INTCPS_THRESHOLD_ADDR
LDR R12, [R0]
(1) The priority-
threshold mechanism is enabled automatically when writing a priority in the range of 0x00 to
0x7F. Writing a value of 0xFF (reset default) disables the priority-
threshold mechanism. Values between 0x3F and 0xFF must not be used. When the hardware-
priority threshold is in use, the priorities of interrupts selected as FIQ or IRQ become linked
otherwise, they are independent. When they are linked, all FIQ priorities must be set higher than
all IRQ priorities to maintain the relative priority of FIQ over IRQ.
(2) When handling FIQs using the priority-
threshold mechanism, both NEWFIQAGR and NEWIRQAGR bits must be written at the same time to ensure
that the new priority threshold is applied while an IRQ sort is in progress. This IRQ will not
have been seen by the ARM, as it will have been masked on entry to the FIQ ISR. However, the
source of the IRQ remains active and it is finally processed when the priority threshold falls to
a priority sufficiently low to allow it to be processed. The precaution of writing to New FIQ
Agreement is not required during an IRQ ISR, as FIQ sorting is not affected (provided all FIQ
priorities are higher than all IRQ priorities).
 ; Step 3 : Get the priority of the highest priority active IRQ
LDR R1, INTCPS_IRQ_PRIORITY_ADDR/INTCPS_FIQ_PRIORITY_ADDR
LDR R1, [R1] ; Get the INTCPS_IRQ_PRIORITY/INTCPS_FIQ_PRIORITY register
AND R1, R1, #ACTIVEPRIO_MASK ; Apply the mask to get the priority of the IRQ
STR R1, [R0] ; Write it to the INTCPS_THRESHOLD register
 ; Step 4 : Get the number of the highest priority active IRQ
LDR R10, INTCPS_SIR_IRQ_ADDR/INTCPS_SIR_FIQ_ADDR
LDR R10, [R10] ; Get the INTCPS_SIR_IRQ/INTCPS_SIR_FIQ register
AND R10, R10, #ACTIVEIRQ_MASK ; Apply the mask to get the active IRQ number
 ; Step 5 : Allow new IRQs and FIQs at INTC side
MOV R0, #0x1/0x3 ; Get the NEWIRQAGR and NEWFIQAGR bit position
LDR R1, INTCPS_CONTROL_ADDR
STR R0, [R1] ; Write the NEWIRQAGR and NEWFIQAGR bit
 ; Step 6 : Data Synchronization Barrier
MOV R0, #0 MCR P15, #0, R0, C7, C10, #4
 ; Step 7 : Read-modify-write the CPSR to enable IRQs/FIQs at ARM side
MRS R0, CPSR ; Read the status register
BIC R0, R0, #0x80/0x40 ; Clear the I/F bit
MSR CPSR, R0 ; Write it back to enable IRQs
 ; Step 8 : Jump to relevant subroutine handler
LDR PC, [PC, R10, lsl #2] ; PC base address points this instruction + 8
NOP ; To index the table by the PC
 ; Table of handler start addresses
.word IRQ0handler ;IRQ0 BANK0
.word IRQ1handler
.word IRQ2handler
```

After the return of the relevant IRQ/FIQ subroutine handle:

1. Disable IRQs/FIQs at ARM side.
2. Restore the INTCPS_THRESHOLD PRIORITYTHRESHOLD field.
3. Restore the ARM critical context registers.

The following sample code shows the three previous steps:

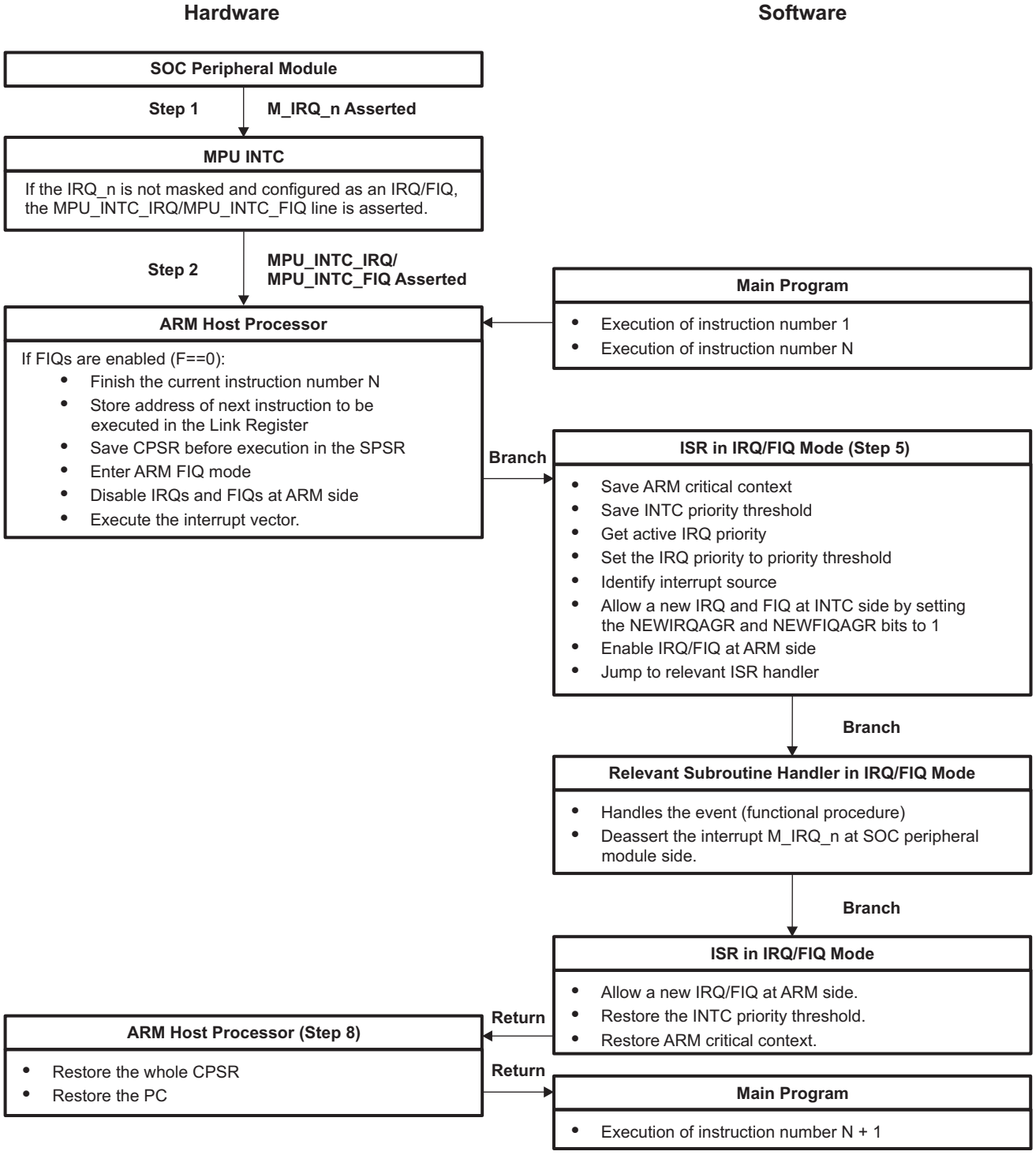
CAUTION

The following code is an assembly code compatible with ARM architecture V6 and V7. This code is developed for the Texas Instruments Code Composer Studio tool set. It is a draft version, only tested on an emulated environment.

```
IRQ_ISR_end:
 ; Step 1 : Read-modify-write the CPSR to disable IRQs/FIQs at ARM side
MRS R0, CPSR ; Read the CPSR
ORR R0, R0, #0x80/0x40 ; Set the I/F bit
MSR CPSR, R0 ; Write it back to disable IRQs
 ; Step 2 : Restore the INTCPS_THRESHOLD register from R12
LDR R0, INTCPS_THRESHOLD_ADDR
STR R12, [R0]
 ; Step 3 : Restore critical context
MSR SPSR, R11 ; Restore the SPSR from R11
LDMFD SP!, {R0-R12, LR} ; Restore working registers and Link register
 ; Return after handling the interrupt
SUBS PC, LR, #4
```

Figure 12-5 shows the nested IRQ/FIQ processing sequence from the originating device peripheral module to the main program interruption.

Figure 12-5. Nested IRQ/FIQ Processing Sequence



12.3.4 Interrupt Preemption

If wanting to enable pre-emption by higher priority interrupts, the ISR should read the active interrupt priority and write it to the priority threshold register. Writing a '1' to the appropriate NEW_IRQ_AGR or NEW_FIQ_AGR bits of the CONTROL register while still processing the interrupt will now allow only higher priority interrupts to pre-empt.

For each level of pre-emption, the programmer must save the threshold value before modifying it and restore it at the end of that ISR level.

The priority threshold mechanism is enabled automatically when writing a priority in the range of 0 to 7Fh as will be read from the IRQ_PRIORITY and FIQ_PRIORITY registers. Writing a value of FFh (reset default) disables the priority threshold mechanism.

When the hardware priority threshold is in use the priorities of interrupts selected as FIQ or IRQ become linked, otherwise they are independent. When linked, it is required that all FIQ priorities be set higher than all IRQ priorities to maintain the relative priority of FIQ over IRQ.

When handling FIQs using the priority threshold mechanism, it is required to write both New FIQ Agreement and New IRQ Agreement bits at the same time to cover the case that the new priority threshold is applied while an IRQ sorting is in progress. This IRQ will not have been seen by the ARM as it will have been masked on entry to the FIQ ISR. However, the source of the IRQ will remain active and it will be finally processed when the priority threshold falls to a low enough priority. The precaution of writing to New FIQ Agreement (as well as New IRQ Agreement) is not required during an IRQ ISR as FIQ sorting will not be affected (provided all FIQ priorities are higher than all IRQ priorities).

12.3.5 AINTC Spurious Interrupt Handling

The spurious flag indicates whether the result of the sorting (a window of 10 AINTC functional clock cycles after the interrupt assertion) is invalid. The sorting is invalid if:

- The interrupt that triggered the sorting is no longer active during the sorting.
- A change in the mask has affected the result during the sorting time.

As a result, the values in the INTCPS_MIRn, INTCPS_ILRm, or INTCPS_MIR_SETn registers must not be changed while the corresponding interrupt is asserted. Only the active interrupt input that triggered the sort can be masked before it turn on the sort. If these registers are changed within the 10-cycle window after the interrupt assertion. The resulting values of the following registers become invalid:

- INTCPS_SIR_IRQ
- INTCPS_SIR_FIQ
- INTCPS_IRQ_PRIORITY
- INTCPS_FIQ_PRIORITY

This condition is detected for both IRQ and FIQ, and the invalid status is flagged across the SPURIOUSIRQFLAG (see NOTE 1) and SPURIOUSFIQFLAG (see NOTE 2) bit fields in the SIR and PRIORITY registers. A 0 indicates valid and a 1 indicates invalid interrupt number and priority. The invalid indication can be tested in software as a false register value.

NOTE:

1. The INTCPS_SIR_IRQ[31:7] SPURIOUSIRQFLAG bit field is a copy of the INTCPS_IRQ_PRIORITY[31:7] SPURIOUSIRQFLAG bit field.
2. The INTCPS_SIR_FIQ[31:7] SPURIOUSFIQFLAG bit field is a copy of the INTCPS_FIQ_PRIORITY[31:7] SPURIOUSFIQFLAG bit field.

12.4 AINTC Registers

If the INTCPS_PROTECTION[0] PROTECTION bit is set, access to the AINTC registers is restricted to the supervisor mode. Access to the INTCPS_PROTECTION register is always restricted to privileged mode.

Each register from INTCPS_ITRn to INTCPS_PENDING_FIQn contains 32 bits, 1 bit for each interrupt (in ascending order: bit 0 of the INTCPS_ITR0 register applies to interrupt line 0; bit 0 of the INTCPS_ITR1 register applies to interrupt line 32).

CAUTION

The AINTC registers are limited to 32-bit and 16-bit data accesses; 8-bit data access is not allowed and can corrupt the register content.

Table 12-3 lists the memory-mapped registers for the AINTC. For the base address of these registers, see Table 1-15.

Table 12-3. ARM Interrupt Controller (AINTC) Registers

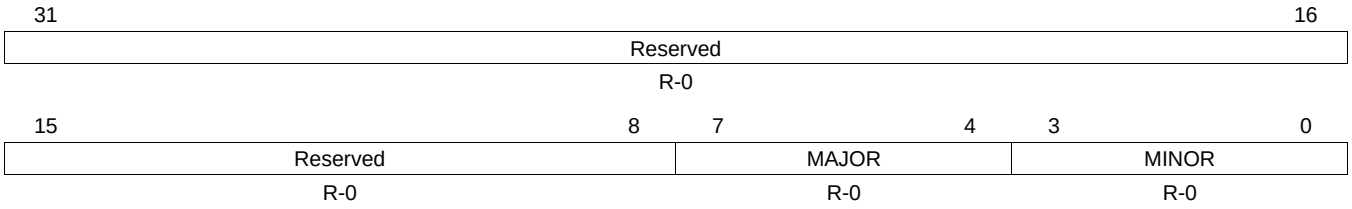
Address Offset	Acronym	Register Description	Section
0h	INTCPS_REVISION	Revision Identification Register	Section 12.4.1
10h	INTCPS_SYSCONFIG	System Configuration Register	Section 12.4.2
14h	INTCPS_SYSSTATUS	System Status Register	Section 12.4.3
40h	INTCPS_SIR_IRQ	Spurious IRQ Flag Register	Section 12.4.4
44h	INTCPS_SIR_FIQ	Spurious FIQ Flag Register	Section 12.4.5
48h	INTCPS_CONTROL	Control Register	Section 12.4.6
4Ch	INTCPS_PROTECTION	Protection Mode Register	Section 12.4.7
50h	INTCPS_IDLE	Idle Mode Register	Section 12.4.8
60h	INTCPS_IRQ_PRIORITY	IRQ Priority Register	Section 12.4.9
64h	INTCPS_FIQ_PRIORITY	FIQ Priority Register	Section 12.4.10
68h	INTCPS_THRESHOLD	Priority Threshold Register	Section 12.4.11
80h + (20h × n)(1)	INTCPS_ITR0-3	Interrupt Status Register	Section 12.4.12
84h + (20h × n)(1)	INTCPS_MIR0-3	Interrupt Mask Register	Section 12.4.13
88h + (20h × n)(1)	INTCPS_MIR_CLEAR0-3	Interrupt Mask Clear Register	Section 12.4.14
8Ch + (20h × n)(1)	INTCPS_MIR_SET0-3	Interrupt Mask Set Register	Section 12.4.15
90h + (20h × n)(1)	INTCPS_ISR_SET0-3	Software Interrupt Set Register	Section 12.4.16
94h + (20h × n)(1)	INTCPS_ISR_CLEAR0-3	Software Interrupt Clear Register	Section 12.4.17
98h + (20h × n)(1)	INTCPS_PENDING_IRQ0-3	IRQ Status Register	Section 12.4.18
9Ch + (20h × n)(1)	INTCPS_PENDING_FIQ0-3	FIQ Status Register	Section 12.4.19
100h + (4h × m)(1)	INTCPS_ILR0-127	Interrupt Priority Register	Section 12.4.20

(1) n = 0 to 3, m = 0 to 127

12.4.1 INTCPS_REVISION Register

This register contains the IP revision code.

Figure 12-6. INTCPS_REVISION Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

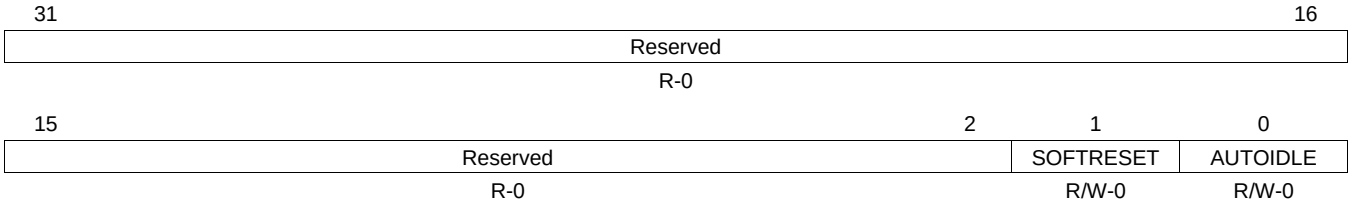
Table 12-4. INTCPS_REVISION Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0
7-4	MAJOR	0	Major Revision.
3-0	MINOR	0	Minor Revision.

12.4.2 INTCPS_SYSCONFIG Register

This register controls various parameters of the module interface.

Figure 12-7. INTCPS_SYSCONFIG Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-5. INTCPS_SYSCONFIG Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Write 0s for future compatibility. Read returns reset value.
1	SOFTRESET		Software reset. Set this bit to trigger a module reset. The bit is automatically reset by the hardware. Read returns 0.
		W0	No effect.
		W1	The module is reset.
0	AUTOIDLE		Internal interface clock gating strategy.
		0	Interface clock is free-running.
		1	Automatic interface clock gating strategy is applied, based on the interface bus activity.

12.4.3 INTCPS_SYSSTATUS Register

This register provides status information about the module.

Figure 12-8. INTCPS_SYSSTATUS Register

31	16
Reserved	
R-0	

15	1	0
Reserved		RESETDONE
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-6. INTCPS_SYSSTATUS Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Read returns reset value.
0	RESETDONE		Internal reset monitoring.
		0	Internal module reset is ongoing.
		1	Reset complete.

12.4.4 INTCPS_SIR_IRQ Register

This register supplies the currently active IRQ interrupt number.

Figure 12-9. INTCPS_SIR_IRQ Register

31	16
SPURIOUSIRQFLAG	
R-1FF FFFFh	

15	7	6	0
SPURIOUSIRQFLAG		ACTIVEIRQ	
R-1FF FFFFh		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-7. INTCPS_SIR_IRQ Register Field Descriptions

Bit	Field	Value	Description
31-7	SPURIOUSIRQFLAG	0-1FF FFFFh	Spurious IRQ flag
6-0	ACTIVEIRQ	0-7Fh	Active IRQ number

12.4.5 INTCPS_SIR_FIQ Register

This register supplies the currently active FIQ interrupt number.

Figure 12-10. INTCPS_SIR_FIQ Register

31	16
SPURIOUSFIQFLAG	
R-1FF FFFFh	

15	7	6	0
SPURIOUSFIQFLAG		ACTIVEFIQ	
R-1FF FFFFh		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-8. INTCPS_SIR_FIQ Register Field Descriptions

Bit	Field	Value	Description
31-7	SPURIOUSFIQFLAG	0-1FF FFFFh	Spurious FIQ flag
6-0	ACTIVEFIQ	0-7Fh	Active FIQ number

12.4.6 INTCPS_CONTROL Register

This register contains the new interrupt agreement bits.

Figure 12-11. INTCPS_CONTROL Register

31	16
Reserved	
R-0	

15	2	1	0
Reserved		NEWFIQAGR	NEWIRQAGR
R-0		W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-9. INTCPS_CONTROL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Write 0s for future compatibility. Read returns reset value.
1	NEWFIQAGR		Reset FIQ output and enable new FIQ generation.
		W0	No effect
		W1	Reset FIQ output and enable new FIQ generation.
0	NEWIRQAGR		New IRQ generation
		W0	No effect
		W1	Reset IRQ output and enable new IRQ generation.

12.4.7 INTCPS_PROTECTION Register

This register controls protection of the other registers. It can be accessed only in supervisor mode, regardless of the current value of the protection bit.

Figure 12-12. INTCPS_PROTECTION Register

31	16
Reserved	
R-0	

15	1	0
Reserved		PROTECTION
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-10. INTCPS_PROTECTION Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Write 0s for future compatibility. Read returns reset value.
0	PROTECTION		Protection mode
		0	Protection mode is disabled.
		1	Protection mode is enabled. When enabled, all the AINTC registers are accessible only in privileged mode.

12.4.8 INTCPS_IDLE Register

This register controls the functional clock auto-idle and the synchronizer clock auto-gating.

Figure 12-13. INTCPS_IDLE Register

31	16
Reserved	
R-0	

15	2	1	0
Reserved		TURBO	FUNCIDLE
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-11. INTCPS_IDLE Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Write 0s for future compatibility. Read returns reset value.
1	TURBO		Input synchronizer clock auto-gating
		0	Input synchronizer clock is free-running.
		1	Input synchronizer clock is auto-gated based on interrupt input activity.
0	FUNCIDLE		Functional clock idle mode
		0	Functional clock gating strategy is applied.
		1	Functional clock is free-running.

12.4.9 INTCPS_IRQ_PRIORITY Register

This register supplies the currently active IRQ priority level.

Figure 12-14. INTCPS_IRQ_PRIORITY Register

31	16
SPURIOUSIRQFLAG	
R-1FF FFFFh	

15	7	6	0
SPURIOUSIRQFLAG		IRQPRIORITY	
R-1FF FFFFh		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-12. INTCPS_IRQ_PRIORITY Register Field Descriptions

Bit	Field	Value	Description
31-7	SPURIOUSIRQFLAG	0-1FF FFFFh	Spurious IRQ flag
6-0	IRQPRIORITY	0-7Fh	Current IRQ priority

12.4.10 INTCPS_FIQ_PRIORITY Register

This register supplies the currently active FIQ priority level.

Figure 12-15. INTCPS_FIQ_PRIORITY Register

31	16
SPURIOUSFIQFLAG	
R-1FF FFFFh	

15	7	6	0
SPURIOUSFIQFLAG		FIQPRIORITY	
R-1FF FFFFh		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

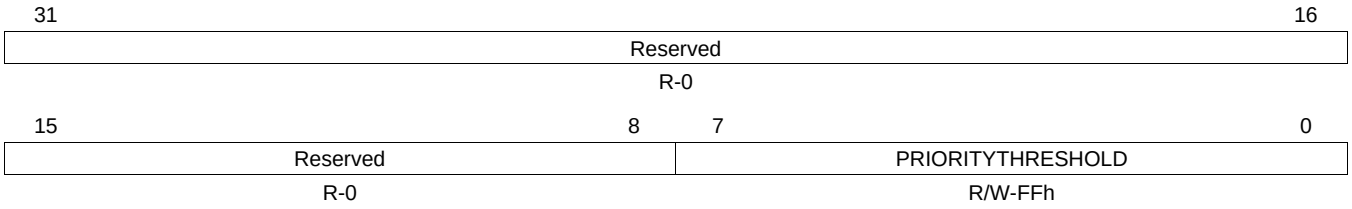
Table 12-13. INTCPS_FIQ_PRIORITY Register Field Descriptions

Bit	Field	Value	Description
31-7	SPURIOUSFIQFLAG	0-1FF FFFFh	Spurious FIQ flag
6-0	FIQPRIORITY	0-7Fh	Current FIQ priority

12.4.11 INTCPS_THRESHOLD Register

This register sets the priority threshold.

Figure 12-16. INTCPS_THRESHOLD Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

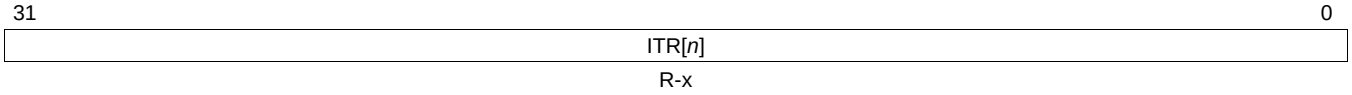
Table 12-14. INTCPS_THRESHOLD Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Write 0s for future compatibility. Read returns reset value.
7-0	PRIORITYTHRESHOLD	0-FFh	Priority threshold
		0-7Fh	Priority threshold
		FFh	Priority threshold is disabled.

12.4.12 INTCPS_ITR0-3 Registers

This register shows the raw interrupt input status before masking.

Figure 12-17. INTCPS_ITRn Register



LEGEND: R = Read only; -*n* = value after reset

Table 12-15. INTCPS_ITRn Register Field Descriptions

Bit	Field	Value	Description
31-0	ITR[*n*]	0-FFFF FFFFh	Interrupt status before masking

12.4.13 INTCPS_MIR0-3 Registers

This register contains the interrupt mask.

Figure 12-18. INTCPS_MIRn Register

31	0
MR[*n*]	
R/W-FFFF FFFFh	

LEGEND: R = Read only; -*n* = value after reset

Table 12-16. INTCPS_MIRn Register Field Descriptions

Bit	Field	Value	Description
31-0	MR[*n*]		Interrupt mask
		0	Interrupt is unmasked.
		1	Interrupt is masked.

12.4.14 INTCPS_MIR_CLEAR0-3 Registers

This register is used to clear the interrupt mask bits.

Figure 12-19. INTCPS_MIR_CLEARn Register

31	0
MIRCLEAR[*n*]	
W-0	

LEGEND: W = Write only; -*n* = value after reset

Table 12-17. INTCPS_MIR_CLEARn Register Field Descriptions

Bit	Field	Value	Description
31-0	MIRCLEAR[*n*]		Clear the interrupt mask [*n*] bits. Read returns 0
		W0	No effect.
		W1	Clears the MIR mask bit [*n*] to 0.

12.4.15 INTCPS_MIR_SET0-3 Registers

This register is used to set the interrupt mask bits.

Figure 12-20. INTCPS_MIR_SETn Register

31	0
MIRSET[*n*]	
W-0	

LEGEND: W = Write only; -*n* = value after reset

Table 12-18. INTCPS_MIR_SETn Register Field Descriptions

Bit	Field	Value	Description
31-0	MIRSET[*n*]		Mask the interrupt [*n*] bits. Read returns 0.
		W0	No effect.
		W1	Set the MIR mask [*n*] bit to 1

12.4.16 INTCPS_ISR_SET0-3 Registers

This register is used to set the software interrupt bits. It is also used to read the currently active software interrupts.

Figure 12-21. INTCPS_ISR_SETn Register

31	0
ISRSET[*n*]	
R/W-0	

LEGEND: R = Read only; -*n* = value after reset

Table 12-19. INTCPS_ISR_SETn Register Field Descriptions

Bit	Field	Value	Description
31-0	ISRSET[*n*]		Set the software interrupt [*n*] bits. Read returns the currently active software interrupts.
		W0	No effect.
		W1	Sets the software interrupt [*n*] bit to 1.

12.4.17 INTCPS_ISR_CLEAR0-3 Registers

This register is used to clear the software interrupt bits.

Figure 12-22. INTCPS_ISR_CLEARn Register

31	0
ISRCLEAR[*n*]	
R/W-FFFF FFFFh	

LEGEND: R = Read only; -*n* = value after reset

Table 12-20. INTCPS_ISR_CLEARn Register Field Descriptions

Bit	Field	Value	Description
31-0	ISRCLEAR[*n*]		Clear the software interrupt [*n*] bits. Read returns 0.
		W0	No effect.
		W1	Clears the software interrupt [*n*] bit to 0.

12.4.18 INTCPS_PENDING_IRQ0-3 Registers

This register contains the IRQ status after masking.

Figure 12-23. INTCPS_PENDING_IRQn Register

31	0
PENDINGIRQ	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 12-21. INTCPS_PENDING_IRQn Register Field Descriptions

Bit	Field	Value	Description
31-0	PENDINGIRQ	0-FFFF FFFFh	IRQ status after masking.

12.4.19 INTCPS_PENDING_FIQ0-3 Registers

This register contains the FIQ status after masking.

Figure 12-24. INTCPS_PENDING_FIQn Register

31	0
PENDINGFIQ	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 12-22. INTCPS_PENDING_FIQn Register Field Descriptions

Bit	Field	Value	Description
31-0	PENDINGFIQ	0-FFFF FFFFh	FIQ status after masking.

12.4.20 INTCPS_ILR0-127 Registers

These registers contain the priority for the interrupts and the FIQ/IRQ steering.

Figure 12-25. INTCPS_ILRm Register

31	16
Reserved	
R-0	

15	9	8	2	1	0
Reserved		PRIORITY		Reserved	FIQNIRQ
R-0		R/W-0		R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-23. INTCPS_ILRm Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Write 0s for future compatibility
8-2	PRIORITY	0-7Fh	Interrupt priority
1	Reserved	0	Write 0 for future compatibility. Read returns reset value.
0	FIQNIRQ		Interrupt IRQ/FIQ mapping. Read returns reset value.
		W0	Interrupt is routed to IRQ.
		W1	Interrupt is routed to FIQ.

Secure Digital (SD)/ Secure Digital I/O (SDIO) Card Interface

This chapter describes the secure digital/secure digital I/O (SD/SDIO) card interface.

Topic **Page**

13.1 Introduction

13.1.1 Overview

This device contains a secure data/secure digital I/O (SD/SDIO) host controller that provides an interface between a local host (LH) such as a microprocessor unit (MPU) or digital signal processor (DSP) and either SD memory cards, or SDIO cards and handles SD/SDIO transactions with minimal LH intervention.

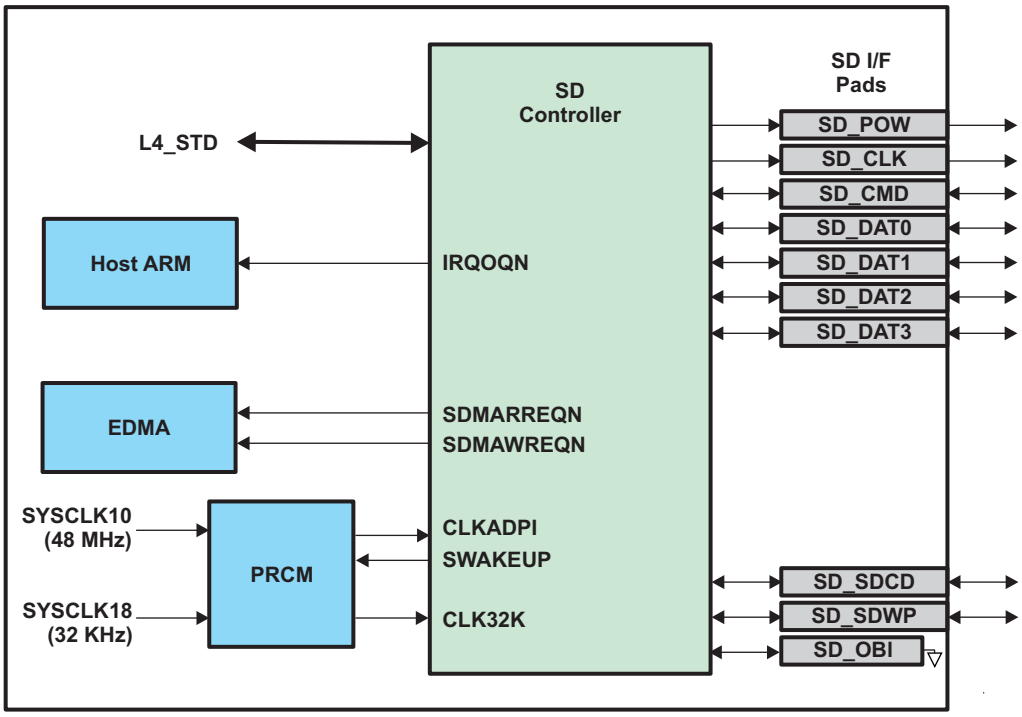
The application interface manages transaction semantics. The SD/SDIO host controller deals with SD/SDIO protocol at transmission level, data packing, adding cyclic redundancy checks (CRC), start/end bit, and checking for syntactical correctness.

The application interface can send every SD/SDIO command and either poll for the status of the adapter or wait for an interrupt request, which is sent back in case of exceptions or to warn of end of operation.

The application interface can read card responses or flag registers. It can also mask individual interrupt sources. All these operations can be performed by reading and writing control registers. The SD/SDIO host controller also supports two DMA channels.

Figure 13-1 gives an overview of the SD/SDIO1 controller instance.

Figure 13-1. SD/SDIO1 Overview



13.1.2 Features

The main features of the SD/SDIO host controller are:

- Built-in 1024-byte buffer for read or write
- Full compliance with SD command/response sets as defined in the *SD Physical Layer Specification*, v2.00.
- Full compliance with SDIO command/response sets and interrupt/read-wait mode as defined in the *SDIO Card Specification, Part E1*, v2.00
- Full compliance with SD Host Controller Standard Specification sets as defined in the *SD Card Specification, Part A2, SD Host Controller Standard Specification*, v2.00
- Full compliance with CE-ATA command/response sets as defined in the *CE-ATA Standard Specification*
- Full compliance with ATA for MMCA specification
- Support command completion signal (CCS) and command completion signal disable (CCSD) management as specified in the *CE-ATA Standard Specification*
- Flexible architecture allowing support for new command structure
- Support:
  - 1-bit or 4-bit transfer mode specifications for SD and SDIO cards
  - 3.3v cards
  - Built-in 1024-byte buffer for read or write
  - 32-bit-wide access bus to maximize bus throughput
  - Single interrupt line for multiple interrupt source events
  - Two slave DMA channels (1 for TX, 1 for RX)
  - Programmable clock generation
  - SDIO Read Wait and Suspend/Resume functions
  - Stop at block gap
  - SDA 2.0 Part A2 programming model
- Supported Data Rates
  - 96MHz functional clock source input
  - Up to 192Mbit/sec (24MByte/sec) in High-Speed SD mode 4-bit data transfer
  - Up to 24Mbit/sec (3MByte/sec) in Default SD mode 1-bit data transfer

13.2 Architecture

One SD/SDIO host controller can support one SD card, or one SDIO card.

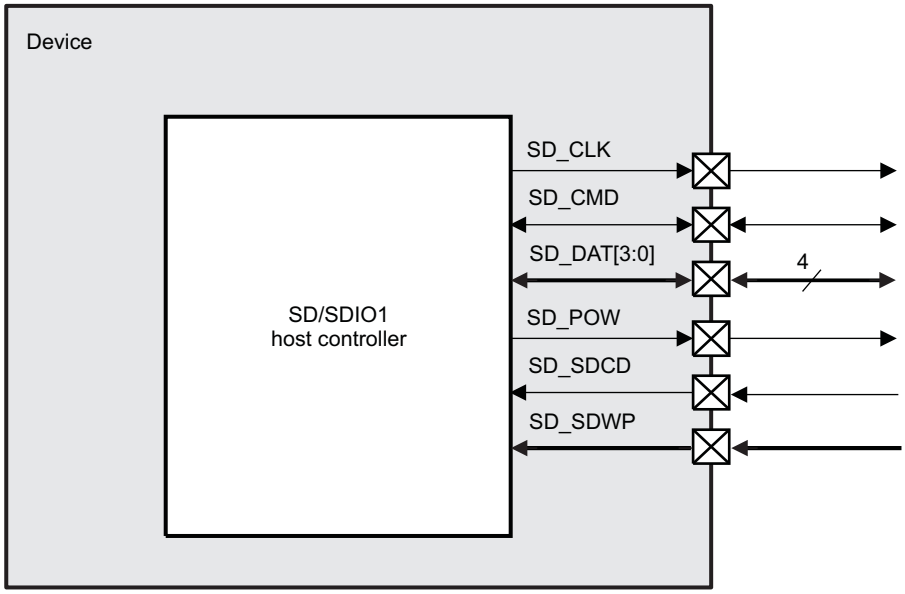
Other combinations (for example, two SD cards) are not supported through a single controller.

13.2.1 SD/SDIO Functional Modes

13.2.1.1 SD/SDIO Connected to an SD Card, or an SDIO Card

Figure 13-2 shows the SD/SDIO1 host controller connected to an SD or SDIO card and its related external connections.

Figure 13-2. SD1 Connectivity to an SD Card



The following SD/SDIO controller pins are used

- **SD_CLK** This pin provides the clock to the memory card from the SD controller.
- **SD_CMD** This pin is used for two-way communication between the connected card and the SD/SDIO controller. The SD/SDIO controller transmits commands to the card and the memory card drives responses to the commands on this pin.
- **SD_DAT3-0** Depending on the type of card you are using, you may need to connect 1 or 4 data lines. The number of DAT pins (the data bus width) is set by the Data Transfer Width (DTW) bit in the control register (SD_HCTL).
- **SD_POW** Used for SD card's cards on/off power supply control. When high, denotes power-on condition.
- **SD_SDCD** This input pin serves as the SD/SDIO carrier detect. This signal is received from a mechanical switch on the slot.
- **SD_SDWP** This input pin is used for the SD/SDIO card's write protect. This signal is received from a mechanical protect switch on the slot (system dependant). Applicable only for SD and SDIO cards that have a mechanical sliding tablet on the side of the card.

  Note: The SD_CLK pin functions as an output but must be configured as an I/O to internally loopback the clock to time the inputs.

Table 13-1 provides a summary of these pins.

Table 13-1. SD/SDIO Controller Pins and Descriptions

Pin	Type	1-Bit Mode	4-Bit Mode
SD_CLK[1]	O	Clock Line	Clock Line
SD_CMD	I/O	Command Line	Command Line
SD_DAT0	I/O	Data Line 0	Data Line 0
SD_DAT1	I/O	(not used)	Data Line 1
SD_DAT2	I/O	(not used)	Data Line 2
SD_DAT3	I/O	(not used)	Data Line 3
SD_POW	O	SD Card Power Output Enable	SD Card Power Output Enable
SD_SDCD	I	SD Card Detect	SD Card Detect
SD_SDWP	I	SD Card Write Protect	SD Card Write Protect

(1) The SD_CLK pin functions as an output but must be configured as an I/O to internally loopback the clock to time the inputs.

13.2.1.2 Protocol and Data Format

The bus protocol between the SD/SDIO host controller and the card is message-based. Each message is represented by one of the following parts:

Command: A command starts an operation. The command is transferred serially from the SD/SDIO host controller to the card on the SD_CMD line.

Response: A response is an answer to a command. The response is sent from the card to the SD/SDIO host controller. It is transferred serially on the SD_CMD line.

Data: Data are transferred from the SD/SDIO host controller to the card or from a card to the SD/SDIO host controller using the DATA lines.

Busy: The SD_DAT0 signal is maintained low by the card as far as it is programming the data received.

CRC status: CRC result is sent by the card through the SD_DAT0 line when executing a write transfer. In the case of transmission error, occurring on any of the active data lines, the card sends a negative CRC status on SD_DAT0. In the case of successful transmission, over all active data lines, the card sends a positive CRC status on SD_DAT0 and starts the data programming procedure.

13.2.1.2.1 Protocol

The SD/SDIO interface only supports block-oriented operation.

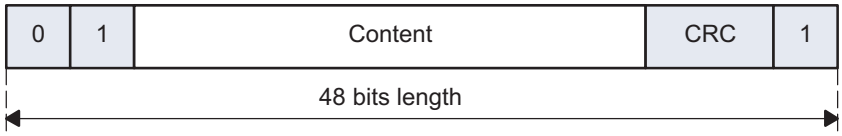
There are specific commands for block-oriented operation. See the *SD Memory Card Specifications* and the *SDIO Card Specification, Part E1* for details about commands and programming sequences supported by the SD and SDIO cards.

13.2.1.2.2 Data Format

Coding Scheme for Command Token

Command packets always start with 0 and end with 1. The second bit is a transmitter bit1 for a host command. The content is the command index (coded by 6 bits) and an argument (for example, an address), coded by 32 bits. The content is protected by 7-bit CRC checksum (see Figure 13-3).

Figure 13-3. Command Token Format



Coding Scheme for Response Token

Response packets always start with 0 and end with a 1. The second bit is a transmitter bit0 for a card response. The content is different for each type of response (R1, R2, R3, R4, R5, and R6) and the content is protected by 7-bit CRC checksum. Depending on the type of commands sent to the card, the SD_CMD register must be configured differently to avoid false CRC or index errors to be flagged on command response (see Table 13-2). For more details about response types, see the *SD Memory Card Specification*, or the *SDIO Card Specification*.

Figure 13-4 and Figure 13-5 depict the 48-bit and 136-bit response packets, respectively.

Table 13-2. Response Type Summary(1)

Response Type SD_CMD[17:16] RSP_TYPE	Index Check Enable SD_CMD[20] CICE	CRC Check Enable SD_CMD[19] CCCE	Name of Response Type
00	0	0	No Response
01	0	1	R2
10	0	0	R3 (R4 for SD cards)
10	1	1	R1, R6, R5 (R7 for SD cards)
11	1	1	R1b, R5b

(1) The SD/SDIO host controller assumes that both clocks may be switched off, whatever the value set in the SD_SYSCONFIG[9:8] CLOCKACTIVITY bit.

Figure 13-4. 48-Bit Response Packet (R1, R3, R4, R5, R6)

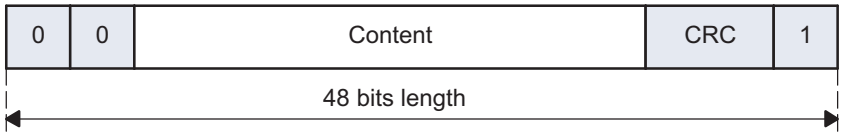
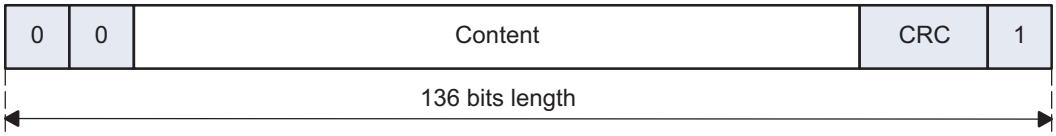


Figure 13-5. 136-Bit Response Packet (R2)



Coding Scheme for Data Token

Data tokens always start with 0 and end with 1 (see Figure 13-6, Figure 13-7, Figure 13-8).

Figure 13-6. Data Packet for Sequential Transfer (1-Bit)

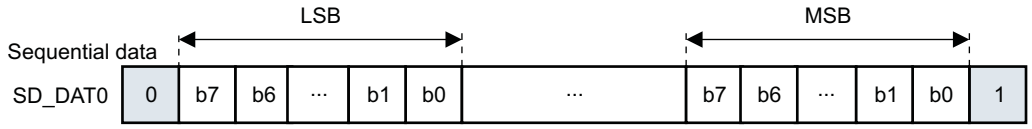


Figure 13-7. Data Packet for Block Transfer (1-Bit)

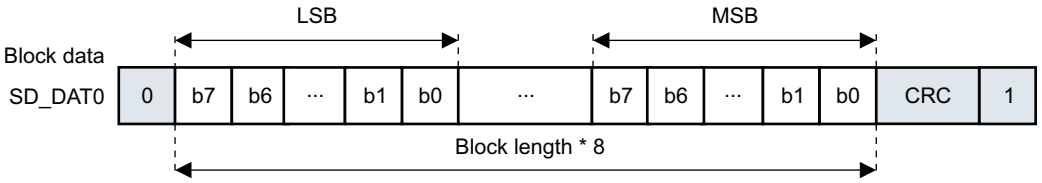
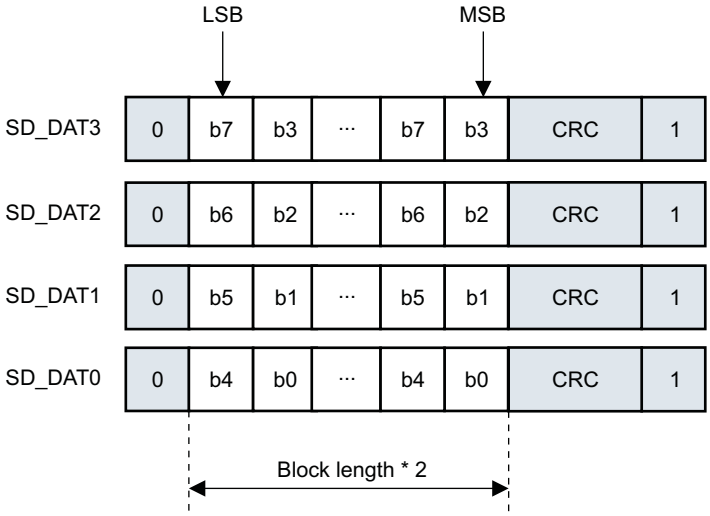


Figure 13-8. Data Packet for Block Transfer (4-Bit)



13.2.2 Resets

13.2.2.1 Hardware Reset

The module is reinitialized by the hardware.

The SD_SYSSTATUS[0] RESETDONE bit can be monitored by the software to check if the module is ready-to-use after a hardware reset.

This hardware reset signal has a global reset action on the module. All configuration registers and all state machines are reset in all clock domains.

This hardware reset signal has a global reset action on the module. All configuration registers and all state-machines are reset in all clock domains.

13.2.2.2 Software Reset

The module is reinitialized by software through the SD_SYSCONFIG[1] SOFTRESET bit. This bit has the same action on the module logic as the hardware signal except for:

- Debounce logic
- SD_PSTATE, SD_CAPA, and SD_CUR_CAPA registers (see corresponding register descriptions)

The SOFTRESET bit is active high. The bit is automatically reinitialized to 0 by the hardware. The SD_SYSCTL[24] SRA bit has the same action as the SOFTRESET bit on the design.

The SD_SYSSTATUS[0] RESETDONE bit can be monitored by the software to check if the module is ready-to-use after a software reset.

Moreover, two partial software reset bits are provided:

- SD_SYSCTL[26] SRD bit
- SD_SYSCTL[25] SRC bit

These two reset bits are useful to reinitialize data or command processes respectively in case of line conflict. When set to 1, a reset process is automatically released when the reset completes:

- The SD_SYSCTL[26] SRD bit resets all finite state-machines and status management that handle data transfers on both the interface and functional side.
- The SD_SYSCTL[25] SRC bit resets all finite state-machines and status management that handle command transfers on both the interface and functional side.

NOTE: If **any** of the clock inputs are not present for the SD/SDIO peripheral, the software reset will not complete.

13.2.3 Power Management

The SD/SDIO host controller can enter into different modes and save power:

- Normal mode
- Idle mode

The two modes are mutually exclusive (the module can be in normal mode or in idle mode). The SD/SDIO host controller is compliant with the PRCM module handshake protocol. When the SD/SDIO power domain is off, the only way to wake up the power domain and different SD/SDIO clocks is to monitor the SD_DAT1 input pin state via a different GPIO line for each SD/SDIO interface.

13.2.3.1 Normal Mode

The autogating of interface and functional clocks occurs when the following conditions are met:

- The SD_SYSCONFIG[0] AUTOIDLE bit is set to 1.

The autogating of interface and functional clocks stops when the following conditions are met:

- A register access occurs through the L3 (or L4) interconnect.
- A wake-up event occurs (an interrupt from a SDIO card).
- A transaction on the SD/SDIO interface starts.

Then the SD/SDIO host controller enters in low-power state even if SD_SYSCONFIG[0] AUTOIDLE is cleared to 0. The functional clock is internally switched off and only interconnect read and write accesses are allowed.

13.2.3.2 Idle Mode

The clocks provided to SD/SDIO are switched off upon a PRCM module request. They are switched back upon module request. The SD/SDIO host controller complies with the PRCM module handshaking protocol:

- Idle request from the system power manager

- Idle acknowledgment from the SD/SDIO host controller
- Wake-up request from the SD/SDIO host controller

The idle acknowledgment varies according to the SD_SYSCONFIG[4:3] SIDLEMODE bit field:

- 0: Force-idle mode. The SD/SDIO host controller acknowledges the system power manager request unconditionally.
- 1h: No-idle mode. The SD/SDIO host controller ignores the system power manager request and behaves normally as if the request was not asserted.
- 2h: Smart-idle mode. The SD/SDIO host controller acknowledges the system power manager request according to its internal state.
- 3h: Smart-idle wake-up-capable mode. The SD/SDIO host controller acknowledges the system power manager request according to its internal state. However, the module may generate wake-up events when in idle state (related to IRQ or DMA requests)

During the smart-idle mode period, the SD/SDIO host controller acknowledges that the OCP and Functional clocks may be switched off whatever the value set in the SD_SYSCONFIG[9:8] CLOCKACTIVITY field.

13.2.3.3 Transition from Normal Mode to Smart-Idle Mode

Smart-idle mode is enabled when the SD_SYSCONFIG[4:3] SIDLEMODE bit field is set to 2h or 3h. The SD/SDIO host controller goes into idle mode when the PRCM issues an idle request, according to its internal activity. The SD/SDIO host controller acknowledges the idle request from the PRCM after ensuring the following:

- The current multi/single-block transfer is completed.
- Any interrupt or DMA request is asserted.
- There is no card interrupt on the SD_dat1 signal.

As long as the SD/SDIO controller does not acknowledge the idle request, if an event occurs, the SD/SDIO host controller can still generate an interrupt or a DMA request. In this case, the module ignores the idle request from the PRCM.

As soon as the SD/SDIO controller acknowledges the idle request from the PRCM:

- If Smart-Idle mode, the module does not assert any new interrupt or DMA request
- If Smart-Idle wake-up-capable mode, the module may generate wake-up events related to interrupt or DMA request.

13.2.3.4 Transition from Smart-Idle Mode to Normal Mode

The SD/SDIO host controller detects the end of the idle period when the PRCM deasserts the idle request. For the wake-up event, there is a corresponding interrupt status in the SD_STAT register. The SD/SDIO host controller operates the conversion between wake-up and interrupt (or DMA request) upon exit from smart-idle mode if the associated enable bit is set in the SD_ISE register.

Interrupts and wake-up events have independent enable/disable controls, accessible through the SD_HCTL and SD_ISE registers. The overall consistency must be ensured by software.

The interrupt status register SD_STAT is updated with the event that caused the wake-up in the CIRQ bit when the SD_IE[8] CIRQ_ENABLE associated bit is enabled. Then, the wake-up event at the origin of the transition from smart-idle mode to normal mode is converted into its corresponding interrupt or DMA request. (The SD_STAT register is updated and the status of the interrupt signal changes.)

When the idle request from the PRCM is deasserted, the module switches back to normal mode. The module is fully operational.

13.2.3.5 Force-Idle Mode

Force-idle mode is enabled when the SD_SYSCONFIG[4:3] SIDLEMODE bit field is cleared to 0. Force-idle mode is an idle mode where the SD/SDIO host controller responds unconditionally to the idle request from the PRCM. Moreover, in this mode, the SD/SDIO host controller unconditionally deasserts interrupts and DMA request lines are asserted.

The transition from normal mode to force-idle mode does not affect the bits of the SD_STAT register. In force-idle mode, the interrupt and DMA request lines are deasserted. Interface Clock (OCP) and functional clock (CLKADPI) can be switched off.

CAUTION

In Force-ilde mode, an idle request from the PRCM during a command or a data transfer can lead to an unexpected and unpredictable result. When the module is idle, any access to the module generates an error as long as the OCP clock is alive.

The module exits the force-idle mode when the PRCM deassertes the idle request. Then the module switches back to normal mode. The module is fully operational. Interrupt and DMA request lines are optionally asserted one clock cycle later.

13.2.3.6 Local Power Management

Table 13-3 describes power-management features available for the SD/SDIO modules.

Table 13-3. Local Power Management Features

Feature	Registers	Description
Clock Auto Gating	SD_SYSCONFIG AUTOIDLE bit	This bit allows a local power optimization inside module, by gating the OCP clock upon the interface activity or gating the CLKADPI clock upon the internal activity.
Slave Idle Modes	SD_SYSCONFIG SIDLEMODE bit	Force-idle, No-idle, and Smart-idle modes
Clock Activity	SD_SYSCONFIG CLOCKACTIVITY bit	See Table 13-4 for configuration details.
Master Standby Modes	SD_SYSCONFIG STANDBYMODE bit	Force-idle, No-idle, and Smart-idle modes
Global Wake-Up Enable	SD_SYSCONFIG ENAWAKEUP bit	This bit enables the wake-up feature at module level.
Wake-Up Sources Enable	SD_HCTL register	This register holds one active high enable bit per event source able to generate wake-up signal.

CAUTION

The PRCM module has no hardware means of reading CLOCKACTIVITY settings. Thus, software must ensure consistent programming between the CLOCKACTIVITY and clock PRCM control bits.

Table 13-4. Clock Activity Settings

CLOCKACTIVITY Values	Clock State When Module is in IDLE State		Features Available when Module is in IDLE State	Wake-Up Events
	OCP Clock	CLKADPI		
00	OFF	OFF	None	Card Interrupt
10	OFF	ON	None	
01	ON	OFF	None	
11	ON	ON	All	

13.2.4 Interrupt Requests

Several internal module events can generate an interrupt. Each interrupt has a status bit, an interrupt enable bit, and a signal status enable:

- The status of each type of interrupt is automatically updated in the SD_STAT register; it indicates which service is required.
- The interrupt status enable bits of the SD_IE register enable/disable the automatic update of the SD_STAT register on an event-by-event basis.
- The interrupt signal enable bits of the SD_ISE register enable/disable the transmission of an interrupt request on the interrupt line IRQ (from the SD/SDIOi host controller to the MPU subsystem interrupt controller) on an event-by-event basis.

If an interrupt status is disabled in the SD_IE register, then the corresponding interrupt request is not transmitted, and the value of the corresponding interrupt signal enable in the SD_ISE register is ignored.

When an interrupt event occurs, the corresponding status bit is automatically set to 1 (the SD/SDIO host controller updates the status bit) in the SD_STAT register. If later a mask is applied on the interrupt in the SD_ISE register, the interrupt request is deactivated.

When the interrupt source has not been serviced, if the interrupt status is cleared in the SD_STAT register and the corresponding mask is removed from the SD_ISE register, the interrupt status is not asserted again in the SD_STAT register and the SD/SDIOi host controller does not transmit an interrupt request.

CAUTION

If the buffer write ready interrupt (BWR) or the buffer read ready only interrupt (BRR) are not serviced and are cleared in the SD_STAT register, and the corresponding mask is removed, then the SD/SDIOi host controller will wait for the service of the interrupt without updating the status SD_STAT or transmitting an interrupt request.

Table 13-5 lists the event flags, and their mask, that can cause module interrupts.

Table 13-5. Events

Event Flag	Event Mask	Map To	Description
SD_STAT[29] BADA	SD_IE[29] BADA_ENABLE	IRQ	Bad Access to Data space. This bit is set automatically to indicate a bad access to buffer when not allowed. This bit is set during a read access to the data register (SD_DATA) while buffer reads are not allowed (SD_PSTATE[11] BRE = 0). This bit is set during a write access to the data register (SD_DATA) while buffer writes are not allowed (SD_STATE[10] BWE = 0)
SD_STAT[28] CERR	SD_IE[28] CERR_ENABLE	IRQ	Card Error. This bit is set automatically when there is at least one error in a response of type R1, R1b, R6, R5 or R5b. Only bits referenced as type E(error) in status field in the response can set a card status error. An error bit in the response is flagged only if corresponding bit in card status response errros SD_CSRE is set. There is not card detection for auto CMD12 command.
SD_STAT[25] ADMAE	SD_IE[25] ADMAE_ENABLE	IRQ	AMDA error. This bit is set when the host controller detects errors during ADMA based data transfer. The stat of the ADMA at an error occurrence is saved in the ADMA Error Status Register. In addition, the host controller generates this interrupt when it detects invalid descriptor data (Valid = 0) at teh ST_FDS state.
SD_STAT[24] ACE	SD_IE[24] ACE_ENABLE	IRQ	Auto CMD12 error. This bit is set automatically when one of the bits in Auto CMD12 Error status register has changed from 0 to 1
SD_STAT[22] DEB	SD_IE[22] DEB_ENABLE	IRQ	Data End Bit error. This bit is set automatically when detecting a 0 at the end bit position of read data on DAT line or at the end position of the CRC status in write mode.
SD_STAT[21] DCRC	SD_IE[21] DCRC_ENABLE	IRQ	Data CRC error. This bit is set automatically when there is a CRC16 error in the data phase response following a block read command or if there is a 3-bit CRC status different of a position "010" token during a block write command.

Table 13-5. Events (continued)

Event Flag	Event Mask	Map To	Description
SD_STAT[20] DTO	SD_IE[20] DTO_ENABLE	IRQ	Data Timeout error. This bit is set automatically according to the following conditions: A) busy timeout for R1b, R5b response. B) busy timeout after write CRC status. C) write CRC status timeout, or D) read data timeout.
SD_STAT[19] CIE	SD_IE[19] CIE_ENABLE	IRQ	Command Index Error. This bit is set automatically when response index differs from corresponding command index previously emitted. The check is enabled through SD_CMD[20] CICE bit.
SD_STAT[18] CEB	SD_IE[18] CEB_ENABLE	IRQ	Command End Bit error. This bit is set automatically when detecting a 0 at the end bit position of a command response.
SD_STAT[17] CCRC	SD_IE[17] CCRC_ENABLE	IRQ	Command CRC error. This bit is set automatically when there is a CRC7 error in the command response. CRC check is enabled through the SD_CMD[19] CCCE bit.
SD_STAT[16] CTO	SD_IE[16] CTO_ENABLE	IRQ	Command Timeout error. This bit is set automatically when no response is received within 64 clock cycles from the end bit of the command. For commands the reply within 5 clock cycles, the timeout is still detected at 64 clock cycles.
SD_STAT[15] ERRI	SD_IE[15] ERRI_ENABLE	IRQ	Error Interrupt. If any of the bits in the Error Interrupt Status register (SD_STAT[24:15]) are set, the this bit is set to 1.
SD_STAT[10] BSR	SD_IE[10] BSR_ENABLE	IRQ	Boot Status Received interrupt. This bit is set automatically when SD_CON[18] BOOT_CF0 is set to 1 or 2h and boot status is received on the dat0 line.
SD_STAT[8] CIRQ	SD_IE[8] CIRQ_ENABLE	IRQ	Card Interrupt. This bit is only used for SD, SDIO, and CE-ATA cards. In 1-bit mode, interrupt source is asynchronous (can be a source of asynchronous wake-up). In 4-bit mode, interrupt source is sampled during the interrupt cycle. In CE-ATA mode, interrupt source is detected when the card drive CMD line to zero during one cycle after data transmission end.
SD_STAT[5] BRR	SD_IE[5] BRR_ENABLE	IRQ	Buffer Read ready. This bit is set automatically during a read operation to the card when one block specified by SD_BLK[10:0] BLEN is completely written in the buffer. It indicates that the memory card has filled out the buffger and the local host needs to empty the buffer by reading it.
SD_STAT[4] BWR	SD_IE[4] BWR_ENABLE	IRQ	Buffer Write ready. This bit is automatically set during a write operation to the card when the host can write a complete block as specified by SD_BLK[10:0] BLEN. It indicates that the memory card has emptied one block from the bugger and the local host is able to write one block of data into the buffer.
SD_STAT[3] DMA	SD_IE[3] DMA_ENABLE	IRQ	DMA interrupt. This status is set when an interrupt is required in the AMDA instruction and after the data transfer is complete.
SD_STAT[2] BGE	SD_IE[2] BGE_ENABLE	IRQ	Block Gap event. When a stop at block gap is requested (SD_HCTL[16] SBGR), this bit is automatically set when transaction is stopped at the block gap during a read or write operation.
SD_STAT[1] TC	SD_IE[1] TC_ENABLE	IRQ	Transfer completed. This bit is always set when a read/write transfer is completed or between two blocks when the transfer is stopped due to a stop at block gap requested (SD_HCTL[16 SBGR). In read mode, this bit is automatically set on completion of a read transfer (SD_PSTATE[9] RTA). In write mode, this bit is automatically set on completion of the DAT line use (SD_PSTATE[2] DLA).
SD_STAT[0] CC	SD_IE[0] CC_ENABLE	IRQ	Command complete. This bit is set when a 1-to-0 transition occurs in the register command inhibit (SD_PSTATE[0] CMDI). If the command is a type for which no response is expected, then the command complete interrupt is generated at the end of the command. A command timeout error (SD_STAT[16] CTO) has higher priority than command complete (SD_STAT[0] CC). If a response is expected but none is received, the a Command Timeout error is detected and signaled instead of the Command Complete interrupt.

13.2.4.1 Interrupt-Driven Operation

An interrupt enable bit must be set in the SD_IE register to enable the module internal source of interrupt.

When an interrupt event occurs, the single interrupt line is asserted and the LH must:

- Read the SD_STAT register to identify which event occurred.
- Write 1 into the corresponding bit of the SD_STAT register to clear the interrupt status and release the interrupt line (if a read is done after this write, this would return 0).

NOTE: In the SD_STAT register, Card Interrupt (CIRQ) and Error Interrupt (ERRI) bits cannot be cleared.

The SD_STAT[8] CIRQ status bit must be masked by disabling the SD_IE[8] CIRQ_ENABLE bit (cleared to 0), then the interrupt routine must clear SDIO interrupt source in SDIO card common control register (CCCR).

The SD_STAT[15] ERRI bit is automatically cleared when all status bits in SD_STAT[31:16] are cleared.

13.2.4.2 Polling

When the interrupt capability of an event is disabled in the SD_ISE register, the interrupt line is not asserted:

- Software can poll the status bit in the SD_STAT register to detect when the corresponding event occurs.
- Writing 1 into the corresponding bit of the SD_STAT register clears the interrupt status and does not affect the interrupt line state.

NOTE: See the note in Section 13.2.4.1 concerning CIRQ and ERRI bits clearing.

13.2.5 DMA Modes

The device supports DMA slave mode only. In this case, the controller is slave on DMA transaction managed by two separated requests (SDMAWREQN and SDMARREQN)

13.2.5.1 DMA Slave Mode Operations

The SD/SDIO controller can be interfaced with a DMA controller. At system level, the advantage is to discharge the local host (LH) of the data transfers. The module does not support wide DMA access (above 1024 bytes) for SD cards as specified in the *SD Card Specification*and *SD Host Controller Standard Specification*.

The DMA request is issued if the following conditions are met:

- The SD_CMD[0] DE bit is set to 1 to trigger the initial DMA request (the write must be done when running the data transfer command).
- A command was emitted on the SD_cmd line.
- There is enough space in the buffer of the SD/SDIO controller to write an entire block (BLEN writes).

13.2.5.1.1 DMA Receive Mode

In a DMA block read operation (single or multiple), the request signal SDMARREQN is asserted to its active level when a complete block is written in the buffer. The block size transfer is specified in the SD_BLK[10:0] BLEN field.

The SDMARREQN signal is deasserted to its inactive level when the sDMA has read one single word from the buffer. Only one request is sent per block; the DMA controller can make a 1-shot read access or several DMA bursts, in which case the DMA controller must manage the number of burst accesses, according to block size BLEN field.

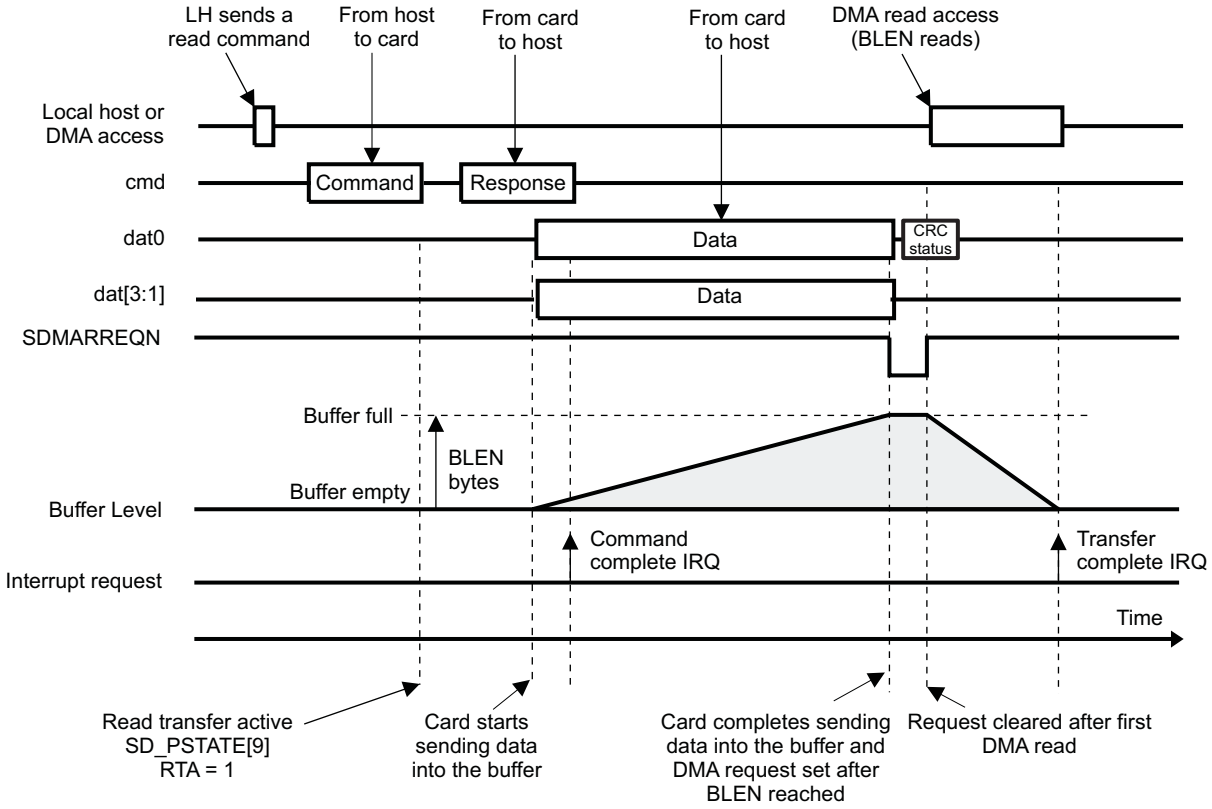
New DMA requests are internally masked if the sDMA has not read exactly BLEN bytes and a new complete block is not ready. As DMA accesses are in 32-bit, then the number of sDMA read is Integer(BLEN/4)+1.

The receive buffer never overflows. In multiple block transfers for block size above 512 bytes, when the buffer gets full, the CLK clock signal (provided to the card) is momentarily stopped until the sDMA or the MPU performs a read access, which reads a complete block in the buffer.

Figure 13-9 provides a summary:

- DMA transfer size = BLEN buffer size in one shot or by burst
- One DMA request per block

Figure 13-9. DMA Receive Mode



13.2.5.1.2 DMA Transmit Mode

In a DMA block write operation (single or multiple), the request signal SDMAWREQN is asserted to its active level when a complete block is to be written to the buffer. The block size transfer is specified in the SD_BLK[10:0] BLEN field.

The SDMAWREQN signal is deasserted to its inactive level when the sDMA has written one single word to the buffer.

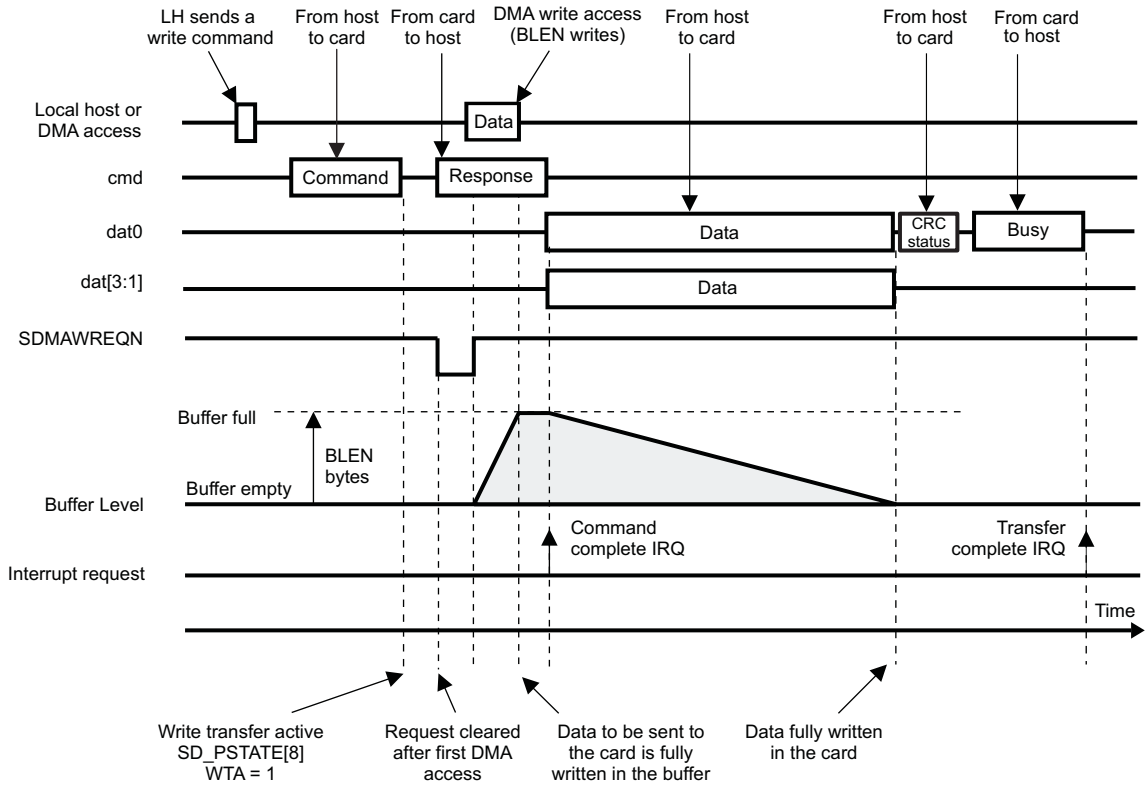
Only one request is sent per block; the DMA controller can make a 1-shot write access or multiple write DMA bursts, in which case the DMA controller must manage the number of burst accesses, according to block size BLEN field.

New DMA requests are internally masked if the sDMA has not written exactly BLEN bytes (as DMA accesses are in 32-bit, then the number of sDMA read is Integer(BLEN/4)+1) and if there is not enough memory space to write a complete block in the buffer.

Figure 13-10 provides a summary:

- DMA transfer size = BLEN buffer size in one shot or by burst
- One DMA request per block

Figure 13-10. DMA Transmit Mode



13.2.6 Buffer Management

13.2.6.1 Data Buffer

The SD/SDIO host controller uses a data buffer. This buffer transfers data from one data bus (Interconnect) to another data bus (SD or SDIO card bus) and vice versa.

The buffer is the heart of the interface and ensures the transfer between the two interfaces (L4 and the card). To enhance performance, the data buffer is completed by a prefetch register and a post-write buffer that are not accessible by the host controller.

The read access time of the prefetch register is faster than the one of the data buffer. The prefetch register allows data to be read from the data buffer at an increased speed by preloading data into the prefetch register.

The entry point of the data buffer, the prefetch buffer, and the post-write buffer is the 32-bit register SD_DATA. A write access to the SD_DATA register followed by a read access from the SD_DATA register corresponds to a write access to the post-write buffer followed by a read access to the prefetch buffer. As a consequence, it is normal that the data of the write access to the SD_DATA register and the data of the read access to the SD_DATA register are different.

The number of 32-bit accesses to the SD_DATA register that are needed to read (or write) a data block with a size of SD_BLK[10:0] BLEN, and equals the rounded up result of BLEN divided by 4. The maximum block size supported by the host controller is hard-coded in the register SD_CAPA[17:16] MBL field and cannot be changed.

A read access to the SD_DATA register is allowed only when the buffer read enable status is set to 1 (SD_PSTATE[11] BRE); otherwise, a bad access (SD_STAT[29] BADA) is signaled.

A write access to the SD_DATA register is allowed only when the buffer write enable status is set to 1 (SD_PSTATE[10] BWE); otherwise, a bad access (SD_STAT[29] BADA) is signaled and the data is not written.

The data buffer has two modes of operation to store and read of the first and second portions of the data buffer:

- When the size of the data block to transfer is less than or equal to MEM_SIZE/2 (in double buffering), two data transfers can occur from one data bus to the other data bus and vice versa at the same time. The SD/SDIO controller uses the two portions of the data buffer in a ping-pong manner so that storing and reading of the first and second portions of the data buffer are automatically interchanged from time to time so that data may be read from one portion (for instance, through a DMA read access on the interconnect bus) while data (for instance, from the card) is being stored into the other portion and vice versa. When BLEN is less than or equal to 200h (that is, less or equal to 512Bytes), each of the two portions of the buffer that can be used have a size of BLEN (that is, 32-bits x BLEN div by 4). Not more than this total size of 2 times 32-bits × BLEN div by 4 can be used.
- When the size of the data block to transfer is larger than MEM_SIZE/2, only one data transfer can occur from one data bus to the other data bus at a time. The SD/SDIO host controller uses the entire data buffer as a single portion. In this mode, a bad access (SD_STAT[29] BADA) is signaled when two data transfers occur from one data bus to the other data bus and vice versa at the same time.

CAUTION

The SD_CMD[4] DDIR bit must be configured before a transfer to indicate the direction of the transfer.

Figure 13-11 shows the buffer management for writing and Figure 13-12 shows the buffer management for reading.

Figure 13-11. Buffer Management for a Write

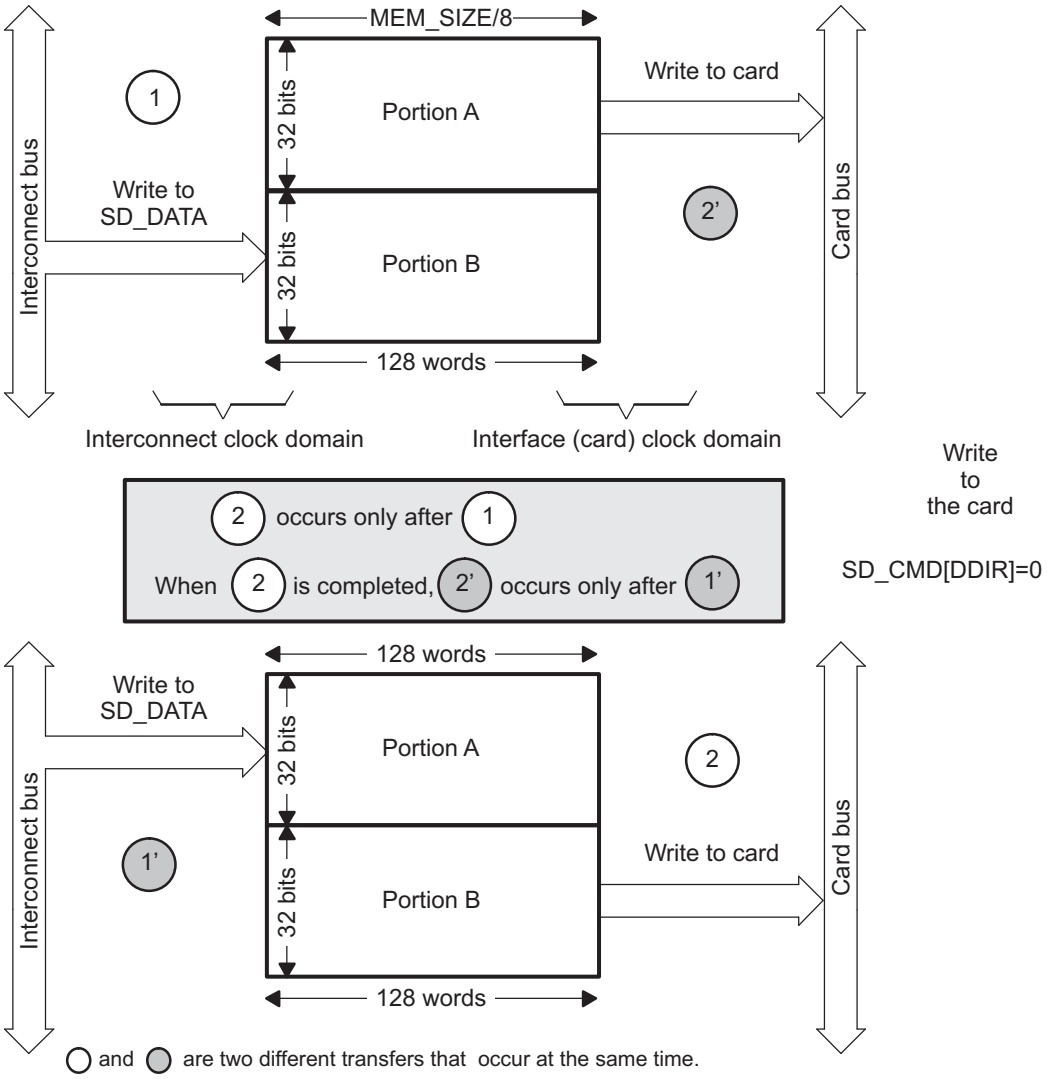
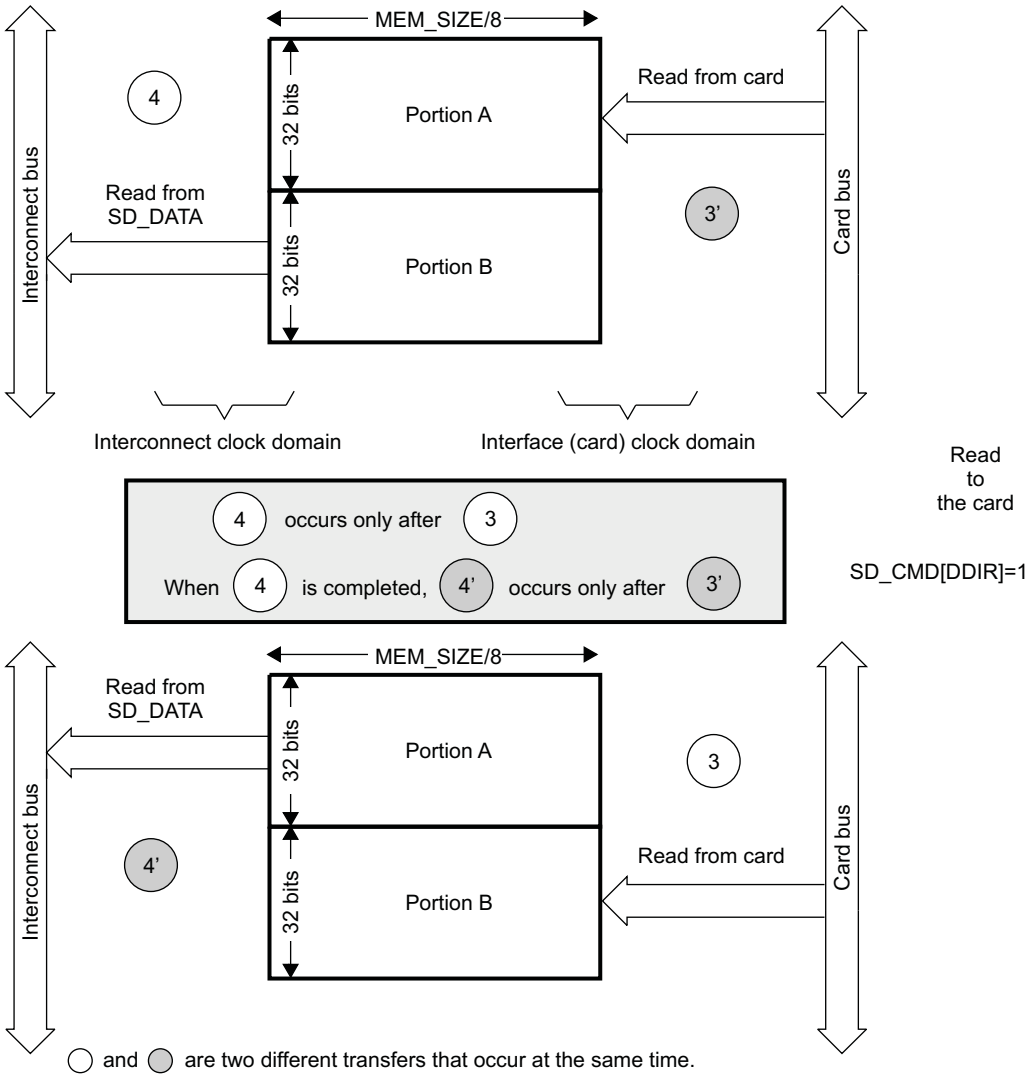


Figure 13-12. Buffer Management for a Read



13.2.6.1.1 Memory Size, Block Length, and Buffer Management Relationship

The maximum block length and buffer management that can be targeted by a system depends on memory depth setting (see Table 13-6).

Table 13-6. Memory Size, BLEN, and Buffer Relationship

Memory Size([5:2] MEMSIZE in bytes)	512	1024	2048	4096
Maximum block length supported	512	1024	2048	2048
Double-buffering for maximum block length	N/A	BLEN <= 512	BLEN <= 1024	BLEN <= 2048
Single-buffering for block length	BLEN <= 512	512 < BLEN <= 1024	1024 < BLEN <= 2048	N/A

13.2.6.1.2 Data Buffer Status

The data buffer status is defined in the following interrupt status register and status register:

- Interrupt status registers (see Figure 13-50):
  - SD_STAT[29] BADABad access to data space
  - SD_STAT[5] BRRBuffer read ready
  - SD_STAT[4] BWRBuffer write ready
- Status registers (see Figure 13-47):
  - SD_PSTATE[11] BREBuffer read enable
  - SD_PSTATE[10] BWEBuffer write enable

13.2.7 Transfer Process

The process of a transfer is dependent on the type of command. It can be with or without a response, with or without data.

13.2.7.1 Different Types of Commands

Different types of commands are specific to SD or SDIO cards. See the *SD Memory Card Specifications*, the *SDIO Card Specification, Part E1*, or the *SD Card Specification, Part A2, SD Host Controller Standard Specification* for more details.

13.2.7.2 Different Types of Responses

Different types of responses are specific to SD, or SDIO cards. See the *SD Memory Card Specifications*, the *SDIO Card Specification, Part E1*, or the *SD Card Specification, Part A2, SD Host Controller Standard Specification* for more details.

Table 13-7 shows how the SD and SDIO responses are stored in the SD_RSPxx registers.

When the host controller modifies part of the SD_RSPxx registers, it preserves the unmodified bits.

The host controller stores the Auto CMD12 response in the SD_RSP76[31:0] register because the Host Controller may have a multiple block data DAT line transfer executing concurrently with a command. This allows the host controller to avoid overwriting the Auto CMD12 response with the command response stored in SD_RSP10 register and vice versa.

Table 13-7. SD and SDIO Responses in the SD_RSPxx Registers

Kind of Response	Response Field[1]	Response Register
R1, R1b (normal response), R3, R4, R5, R5b, R6, R7	RESP[39:8]	SD_RSP10[31:0]
R1b (Auto CMD12 response)	RESP[39:8]	SD_RSP76[31:0]
R2	RESP[127:0]	SD_RSP76[31:0]
		SD_RSP54[31:0]
		SD_RSP32[31:0]
		SD_RSP10[31:0]

(1) RESP refers to the command response format described in the specifications mentioned above.

13.2.8 Transfer or Command Status and Error Reporting

Flags in the SD/SDIO host controller show status of communication with the card:

- A timeout (of a command, a data, or a response)
- A CRC

Error conditions generate interrupts. See Table 13-8 and register description for more details.

Table 13-8. CC and TC Values Upon Error Detected

Error hold in the SD_STAT Register		CC	TC	Comments
29	BADA			No dependency with CC or TC. BADA is related to the register accesses. Its assertion is not dependent of the ongoing transfer.
28	CERR	1		CC is set upon CERR.
22	DEB		1	TC is set upon DEB.
21	DCRC		1	TC is set upon DCRC.
20	DTO			DTO and TC are mutually exclusive. DCRC and DEB cannot occur with DTO.
19	CIE	1		CC is set upon CIE.
18	CEB	1		CC is set upon CEB.
17	CCRC	1		CC can be set upon CCRC. See CTO comment.
16	CTO			CTO and CC are mutually exclusive. CIE, CEB and CERR cannot occur with CTO. CTO can occur at the same time as CCRC bit indicates a command abort due to a contention on CMD line. In this case no CC appears.

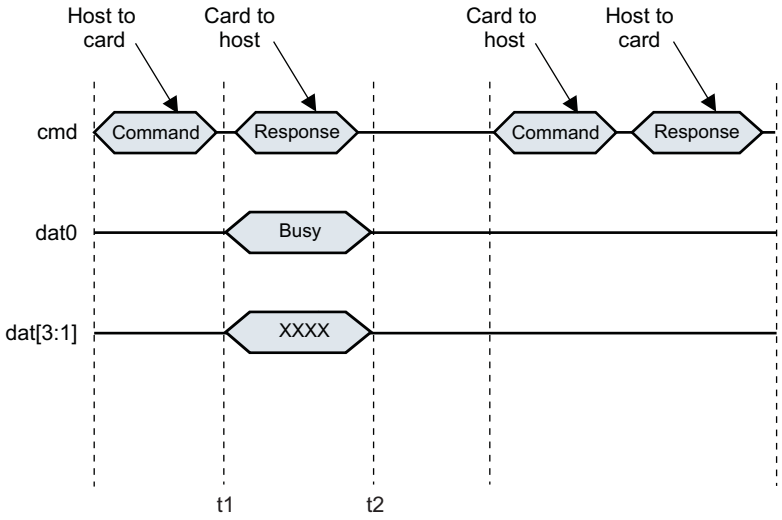
SD_STAT[21] DCRC event can be asserted in the following conditions:

- Busy timeout for R1b, R5b response type
- Busy timeout after write CRC status
- Write CRC status timeout
- Read data timeout
- Boot acknowledge timeout

13.2.8.1 Busy Timeout for R1b, R5b Response Type

Figure 13-13 shows DCRD event condition asserted when there is a busy timeout for R1b or R5b responses.

Figure 13-13. Busy Timeout for R1b, R5b Responses



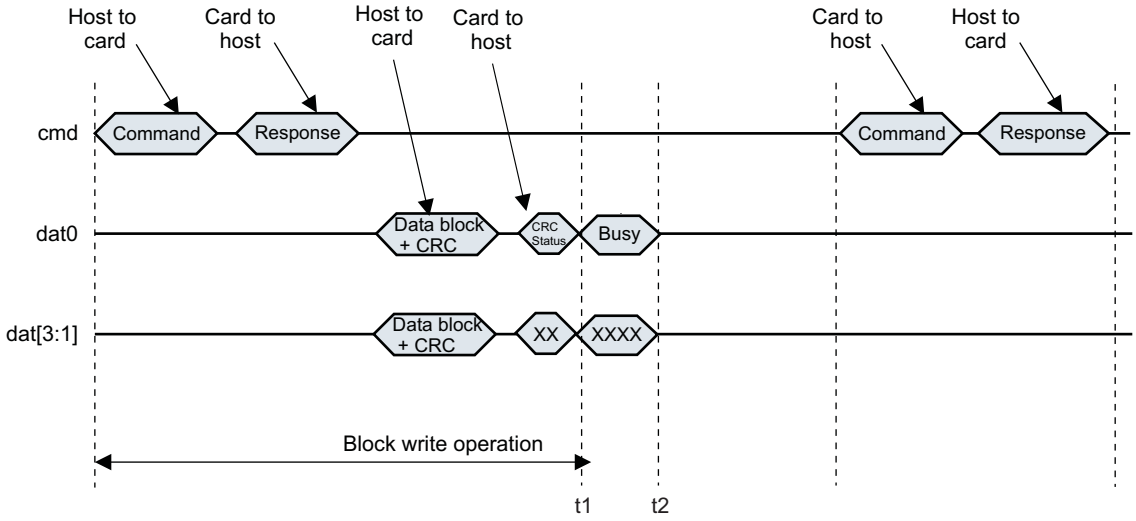
t1 - Data timeout counter is loaded and starts after R1b, R5b response type.

t2 - Data timeout counter stops and if it is 0, SD_STAT[21] DCRC is generated.

13.2.8.2 Busy Timeout After Write CRC Status

Figure 13-14 shows DCRC event condition asserted when there is busy timeout after write CRC status.

Figure 13-14. Busy Timeout After Write CRC Status



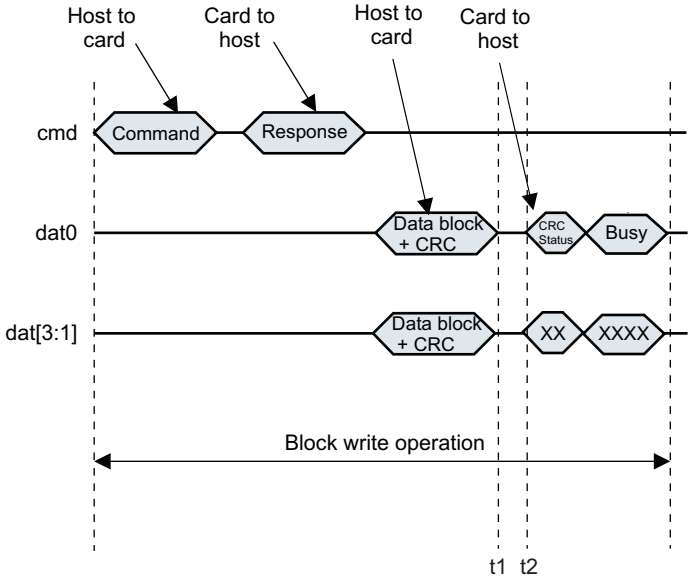
t1 - Data timeout counter is loaded and starts after CRC status.

t2 - Data timeout counter stops and if it is 0, SD_STAT[21] DCRC is generated.

13.2.8.3 Write CRC Status Timeout

Figure 13-15 shows DCRC event condition asserted when there is write CRC status timeout.

Figure 13-15. Write CRC Status Timeout



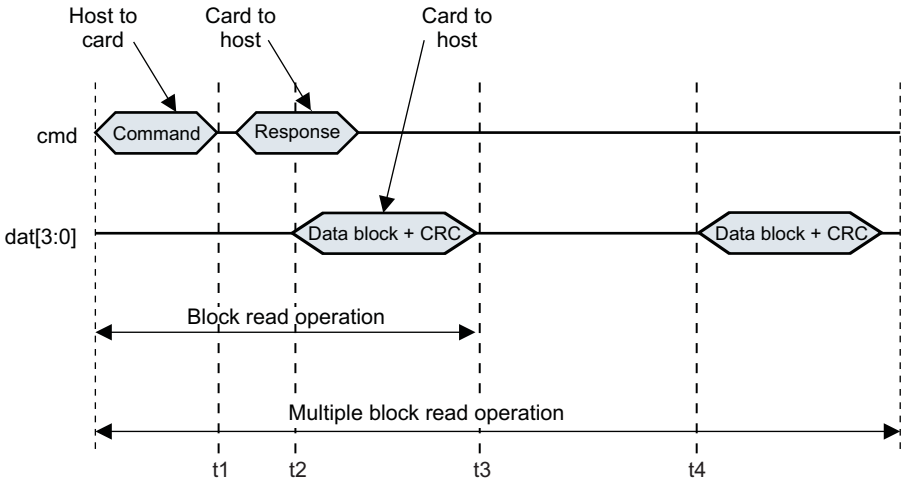
t1 - Data timeout counter is loaded and starts after Data block + CRC.

t2 - Data timeout counter stops and if it is 0, SD_STAT[21] DCRC is generated.

13.2.8.4 Read Data Timeout

Figure 13-16 shows DCRC event condition asserted when there is read data timeout.

Figure 13-16. Read Data Timeout



t1 - Data timeout counter is loaded and starts after Command transmission.

t2 - Data timeout counter stops and if it is 0, SD_STAT[21] DCRC is generated.

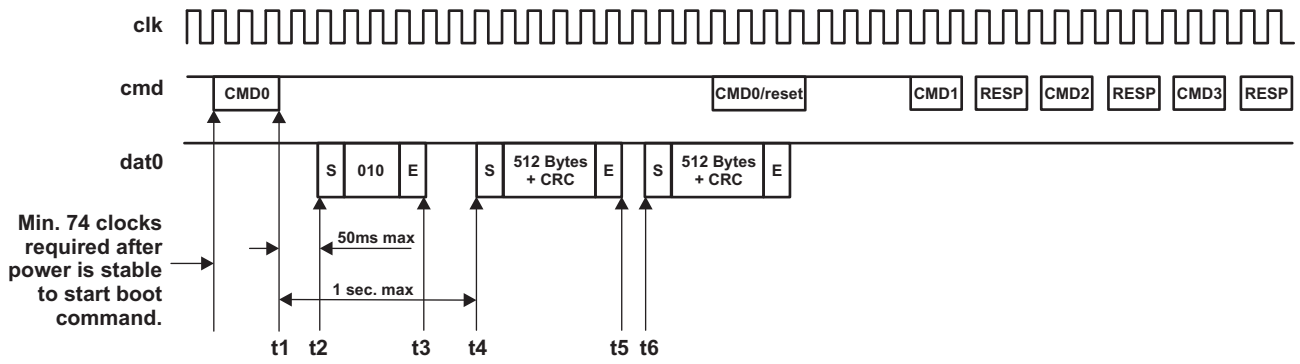
t3 - Data timeout counter is loaded and starts after Data block + CRC transmission.

t4 - Data timeout counter stops and if it is 0, SD_STAT[21] DCRC is generated.

13.2.8.5 Boot Acknowledge Timeout

Figure 13-17 shows DCRC event condition asserted when there is boot acknowledge timeout and CMD0 is used.

Figure 13-17. Boot Acknowledge Timeout When Using CMD0



t1 - Data timeout counter is loaded and starts after CMD0.

t2 - Data timeout counter stops and if it is 0, SD_STAT[21] DCRC is generated.

t3 - Data timeout counter is loaded and starts.

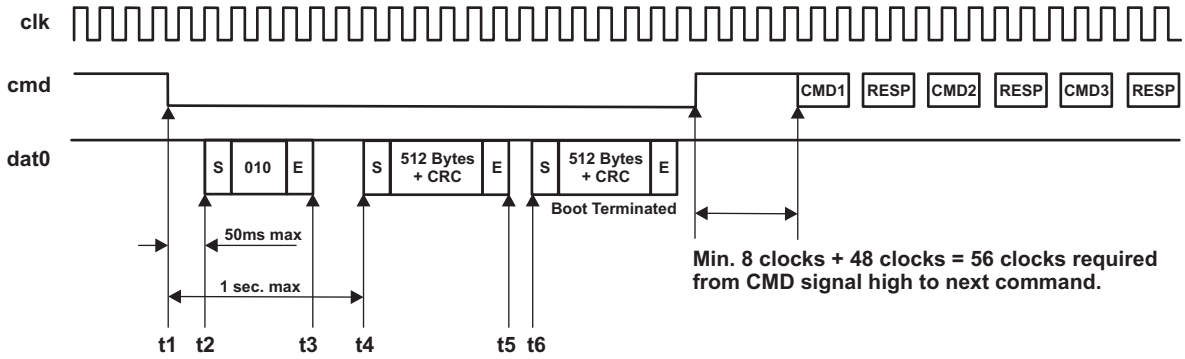
t4 - Data timeout counter stops and if it is 0, SD_STAT[21] DCRC is generated.

t5 - Data timeout counter is loaded and starts after Data + CRC transmission.

t6 - Data timeout counter stops and if it is 0, SD_STAT[21] DCRC is generated.

Figure 13-18 shows DCRC event condition asserted when there is boot acknolwedge timeout and CMD line is held low.

Figure 13-18. Boot Acknowledge Timeout When CMD Held Low



t1 - Data timeout counter is loaded and starts after cmd line is tied to 0.

t2 - Data timeout counter stops and if it is 0, SD_STAT[21] DCRC is generated.

t3 - Data timeout counter is loaded and starts.

t4 - Data timeout counter stops and if it is 0, SD_STAT[21] DCRC is generated.

t5 - Data timeout counter is loaded and starts after Data + CRC transmission.

t6 - Data timeout counter stops and if it is 0, SD_STAT[21] DCRC is generated.

13.2.9 Auto Command 12 Timings

With the UHS definition of SD cards with higher frequency for clocks up to 208, SD standard imposes a specific timing for Auto CMD12 "end bit" arrival.

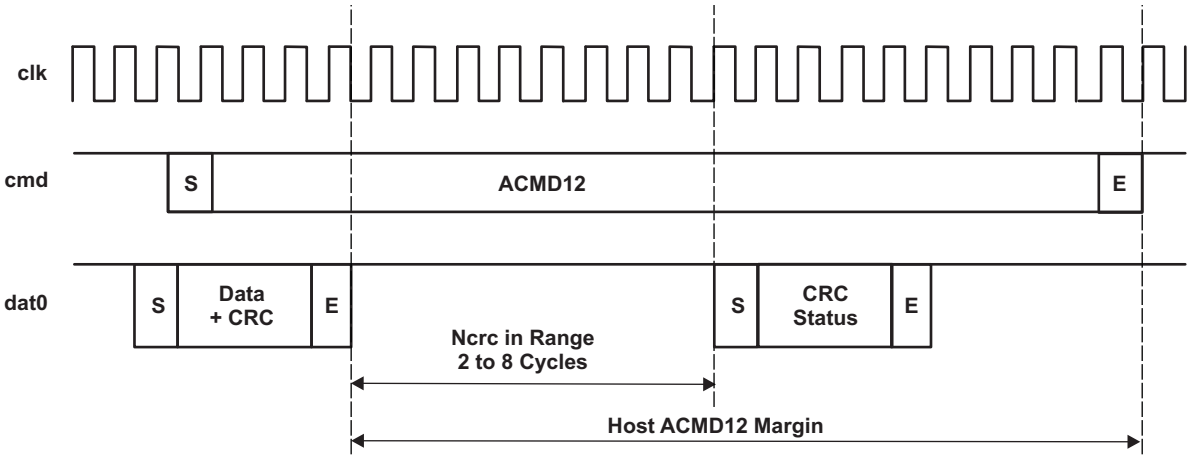
13.2.9.1 Auto Command 12 Timings During Write Transfer

A margin named Ncrc in range of 2 to 8 cycles has been defined for SDR50 and SDR104 card components for write data transfers, as auto command 12 'end bit' shall arrive after the CRC status "end bit".

Figure 13-19 shows auto CMD12 timings during write transfer.

The Host controller has a margin of 18 clock cycles to make sure that auto CMD12 'end bit' arrives after the CRC status. This margin does not depend on bus configuration, DDR or standard transfer, 1,4, or 8 bus width transfer.

Figure 13-19. Auto CMD12 Timing During Write Transfer



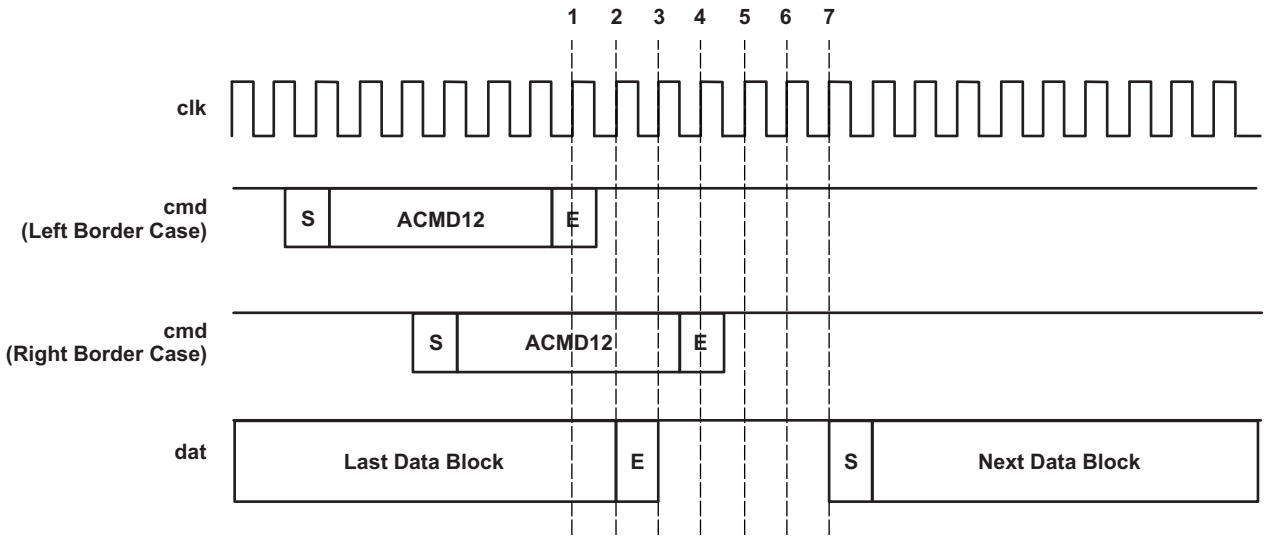
13.2.9.2 Auto Command 12 Timings During Read Transfer

With UHS very high speed cards gap timing between 2 successive cards has been extended to 4 cycles instead of 2. By the way, it gives more flexibility for Host Auto CMD12 arrival in order to receive the last complete and reliable block. SD controller only follows the 'Left Border Case' defined by SD UHS specification.

Figure 13-20 shows auto CMD12 timings during read transfer.

The auto CMD12 arrival sent by the Host controller is not sensitive to the bus configuration whether it is DDR or standard transfer, 1,4, or 8 bit bus width transfer.

Figure 13-20. Auto CMD12 Timings During Read Transfer



13.2.10 Transfer Stop

Whenever a transfer is initiated, the transmission may be willed to stop whereas it is still not finished. Several cases can be faced depending on the transfer type:

- Multiple blocks oriented transfers (for which transfer length is known)
- Continuous stream transfers (which have an infinite length)

NOTE: Since the SD/SDIO controller manages transfers based on a block granularity, the buffer will accept a block only if there is enough space to completely store it. Consequently, if a block is pending in the buffer, no command will be sent to the card because the card clock will be shut off by the controller.

The SD/SDIO controller includes two features which make a transfer stop more convenient and easier to manage:

- Auto CMD12 (for SD only).

  This feature is enabled by setting the SD_CMD[2] ACEN bit to 1 (this setting is relevant for a SD transfer with a known number of blocks to transfer). When the Auto CMD12 feature is enabled, the SD/SDIO controller will automatically issue a CMD12 command when the expected number of blocks has been exchanged.

- Stop at block gap

  This feature is enabled by setting the SD_HCTL[16] SBGR bit to 1. When enabled, this capability holds the transfer on until the end of a block boundary. If a stop transmission is needed, software can use this pause to send a CMD12 to the card.

Table 13-9 shows the common ways to stop a transfer, indicating command to send and features to enable.

Table 13-9. SD/SDIO Controller Transfer Stop Command Summary

		WRITE Transfer		READ Transfer	
		SD	SDIO	SD	SDIO
Single block		Transfer ends automatically Wait TC	Transfer ends automatically Wait TC	Transfer ends automatically Wait TC	Transfer ends automatically Wait TC
Multi blocks (finite or infinite)	Before the programmed block boundary	Send CMD12 Wait TC	Send CMD52 Wait TC	Send CMD12 Wait TC	Send CMD52 Wait TC
	Stop at the end of the transfer (finite transfer only)	Auto CMD12 active Transfer ends automatically Wait TC	Set SD_HCTL[16] SBGR bit to 1. Send CMD52 Wait TC	Auto CMD12 active Transfer ends automatically Wait TC	**If READ_WAIT supported** Stop at block gap Wait TC
					If READ_WAIT not supported Send CMD52 Wait TC

NOTE: The SD/SDIO controller will send the stop command to the card on a block boundary, regardless the moment the command was written to the controller registers.

13.2.11 Output Signals Generation

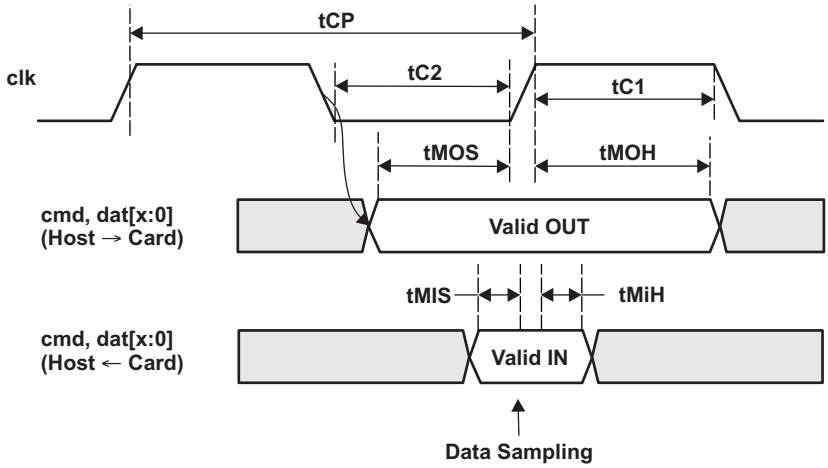
The SD/SDIO output signals can be driven on either falling edge or rising edge depending on the SD_HCTL[2] HSPE bit. This feature allows to reach better timing performance, and thus to increase data transfer frequency.

13.2.11.1 Generation on Falling Edge of Clock

The controller is by default in this mode to maximize hold timings. In this case, SD_HCTL[2] HSPE bit is cleared to 0.

Figure 13-21 shows the output signals of the module when generating from the falling edge of the clock.

Figure 13-21. Output Driven on Falling Edge



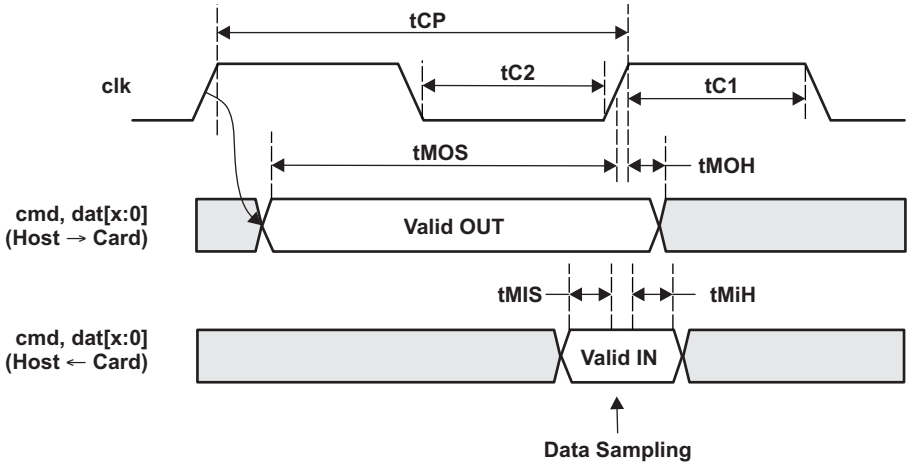
13.2.11.2 Generation on Rising Edge of Clock

This mode increases setup timings and allows reaching higher bus frequency. This feature is activated by setting SD_HCTL[2] HSPE bit to 1. The controller shall be set in this mode to support SDR transfers.

NOTE: Do not use this feature in Dual Data Rate mode (when SD_CON[19] DDR is set to 1).

Figure 13-22 shows the output signals of the module when generating from the rising edge of the clock.

Figure 13-22. Output Driven on Rising Edge



13.2.12 Card Boot Mode Management

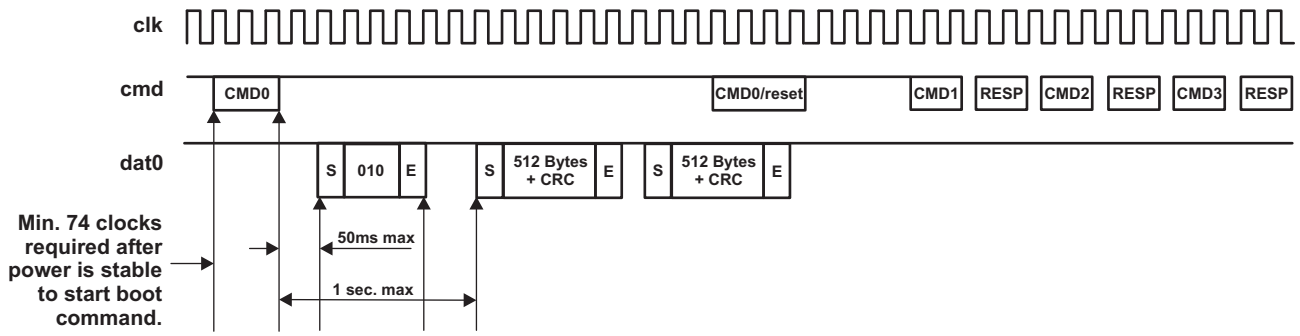
Boot Operation Mode allows the SD/SDIO host controller to read boot data from the connected slave by keeping CMD line low after power-on (or sending CMD0 with specific argument) before issuing CMD1. The data can be read from either boot area or user area, depending on register setting. Power-on boot defines a way for the boot-code to be accessed by the SD/SDIO host controller without an upper-level software driver, speeding the time it takes for a controller to access the boot code.

The two possible ways to issue a boot command (either issuing a CMD0 or driving the CMD line to 0 during the whole boot phase) are described in the following sections.

13.2.12.1 Boot Mode Using CMD0

Figure 13-23 shows the timing diagram of a boot sequence using CMD0.

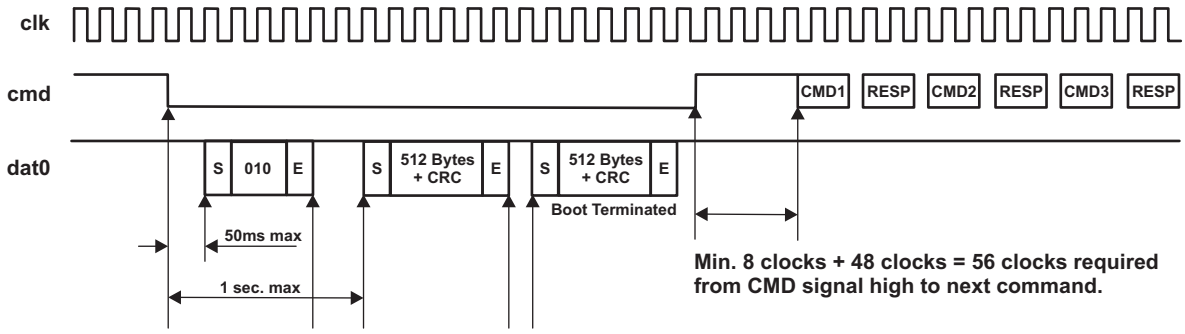
Figure 13-23. Boot Mode With CMD0



13.2.12.2 Boot Mode With CMD Held Low

Figure 13-24 shows the timing diagram of a boot sequence with CMD line tied to 0.

Figure 13-24. Boot Mode With CMD Line Tied to 0



13.2.13 CE-ATA Command Completion Disable Management

The SD/SDIO controller supports CE-ATA features, in particular the detection of command completion token. When a command that requires a command completion signal (SD_CON[12] CEATA and SD_CMD[2] ACEN set to 1) is launched, the host system is no longer allowed to emit a new command in parallel of data transfer unless it is a command completion disable token.

The settings to emit a command completion disable token follow:

- SD_CON[12] CEATA is set to 1.
- SD_CON[2] HR set to 1.
- Clear the SD_ARG register.
- Write into SD_CMD register with value 0000 0000h.

When a command completion disable token was emitted (that is, SD_STAT[0] CC received), the host system is again allowed to emit another type of command (for example a transfer abort command CMD12 to abort transfer).

A critical case can be met when command completion signal disable (CCSD) is emitted during the last data block transfer, the sequence on command line could be sent very close to command completion signal (CCS) token sent by the card.

Three cases can be met:

- CCS is receive just before CCSD is emitted:

  An interrupt CIRQ is generated with CCS detection, CCSD is transmitted to card then an interrupt CC is generated when CCSD ends. In this case, card consider the CCSD sequence.
- CCS is not generated or generated during the CCSD transfer:

  The CCS bit cannot be detected (conflict is not possible as they drive the same level on command line, then no CIRQ interrupt is generated; besides CC interrupt is generated when CCSD ends).
- CCS is generated without CCSD token required:

  Only the interrupt CIRQ is generated when CCS is detected.

13.2.14 Test Registers

Test registers are available to be compliant with SD Host controller specification. This feature is useful to generate interrupts manually for driver debugging. The Force Event register (SD_FE) is used to control the Error Interrupt Status and Auto CMD12 Error Status. The System Test register (SD_SYSTEST) is used to control the signals that connect to I/O pins when the module is configured in system test (SD_CON[4] MODE = 1) mode for boundary connectivity verification.

13.2.15 SD/SDIO Hardware Status Features

Table 13-10 summarizes the SD/SDIO hardware status features.

Table 13-10. SD/SDIO Hardware Status Features

Feature	Type	Register/Bit Field/ Observability Control	Description
Interrupt flags		See Section 13.2.4.	
CMD line signal level	Status	[24] CLEV	Indicates the level of the cmd line
DAT lines signal level	Status	[23:20] DLEV	Indicates the level of the data lines
Buffer read enable	Status	[11] BRE	Readable data exists in the buffer.
Buffer write enable	Status	[10] BWE	Indicates whether there is enough space in the buffer to write BLEN bytes of data
Read transfer active	Status	[9] RTA	This status is used for detecting completion of a read transfer.
Write transfer active	Status	[8] WTA	This status indicates a write transfer active.
Data line active	Status	[2] DLA	Indicates whether the data lines are active
Command Inhibit (data lines)	Status	[1] DATI	Indicates whether issuing of command using data lines is allowed
Command inhibit (CMD line)	Status	[0] CMDI	Indicates whether issuing of command using CMD line is allowed

13.3 Low-Level Programming Models

13.3.1 Surrounding Modules Global Initialization

This section identifies the requirements of initializing the surrounding modules when the module has to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration and environment of the SD/SDIO modules.

Table 13-11. Global Initialization for Surrounding Modules

Surrounding Modules	Comments
PRCM	Module interface and functional clocks must be enabled. For more information, see *Power, Reset, and Clock Management (PRCM)* chapter.
Control module	Module-specific pad muxing and configuration must be set in the control module. See Control Module section in *Chip Level Resources* chapter.
(optional) MPU INTC (or DSP INTC)	MPU INTC configuration must be done to enable the interrupts from the SD module. See *Interrupt Controller* chapter.
(optional) sDMA (or dDMA)	DMA configuration must be done to enable the module DMA channel requests. See SDMA chapter.
(optional) Interconnect	For more information about the interconnect configuration, see Bus Interconnect section in *Chip Level Resources* chapter.

NOTE: The MPU/DSP interrupt controller and the sDMA/dDMA configurations are necessary, if the interrupt and DMA based communication modes are used.

13.3.2 SD/SDIO Controller Initialization Flow

The next sections outline the four steps to initialize the SD/SDIO controller:

- Initialize Clocks
- Software reset of the controller
- Set module's hardware capabilities
- Set module's Idle and Wake-Up modes

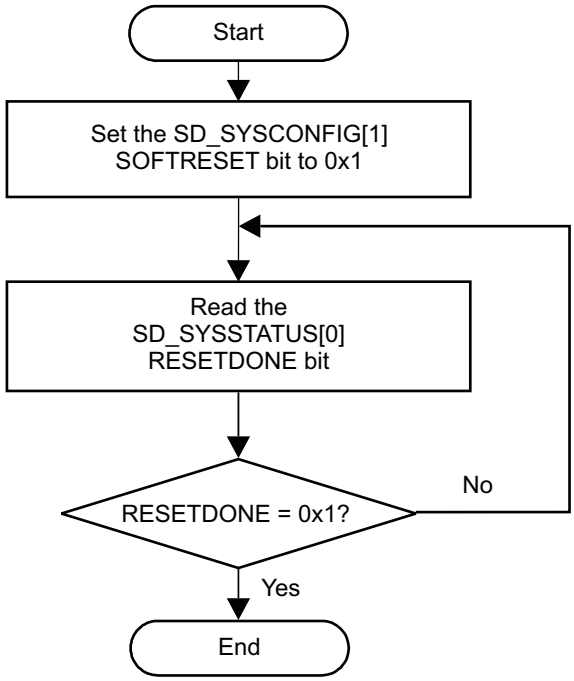
13.3.2.1 Enable OCP and CLKADPI Clocks

Prior to any SD register access one must enable the SD OCP clock and CLKADPI clock in PRCM module registers. Refer to the *Power, Reset, and Clock Management (PRCM)* chapter.

13.3.2.2 SD Soft Reset Flow

Figure 13-25 shows the soft reset process of SD/SDIO controller.

Figure 13-25. SD/SDIO Controller Software Reset Flow



13.3.2.3 Set SD Default Capabilities

Software must read capabilities (in boot ROM for instance) and is allowed to set (write) SD_CAPA[26:24] and SD_CUR_CAPA[23:0] registers before the SD/SDIO host driver is started.

13.3.2.4 Wake-Up Configuration

Table 13-12 details SD controller wake-up configuration.

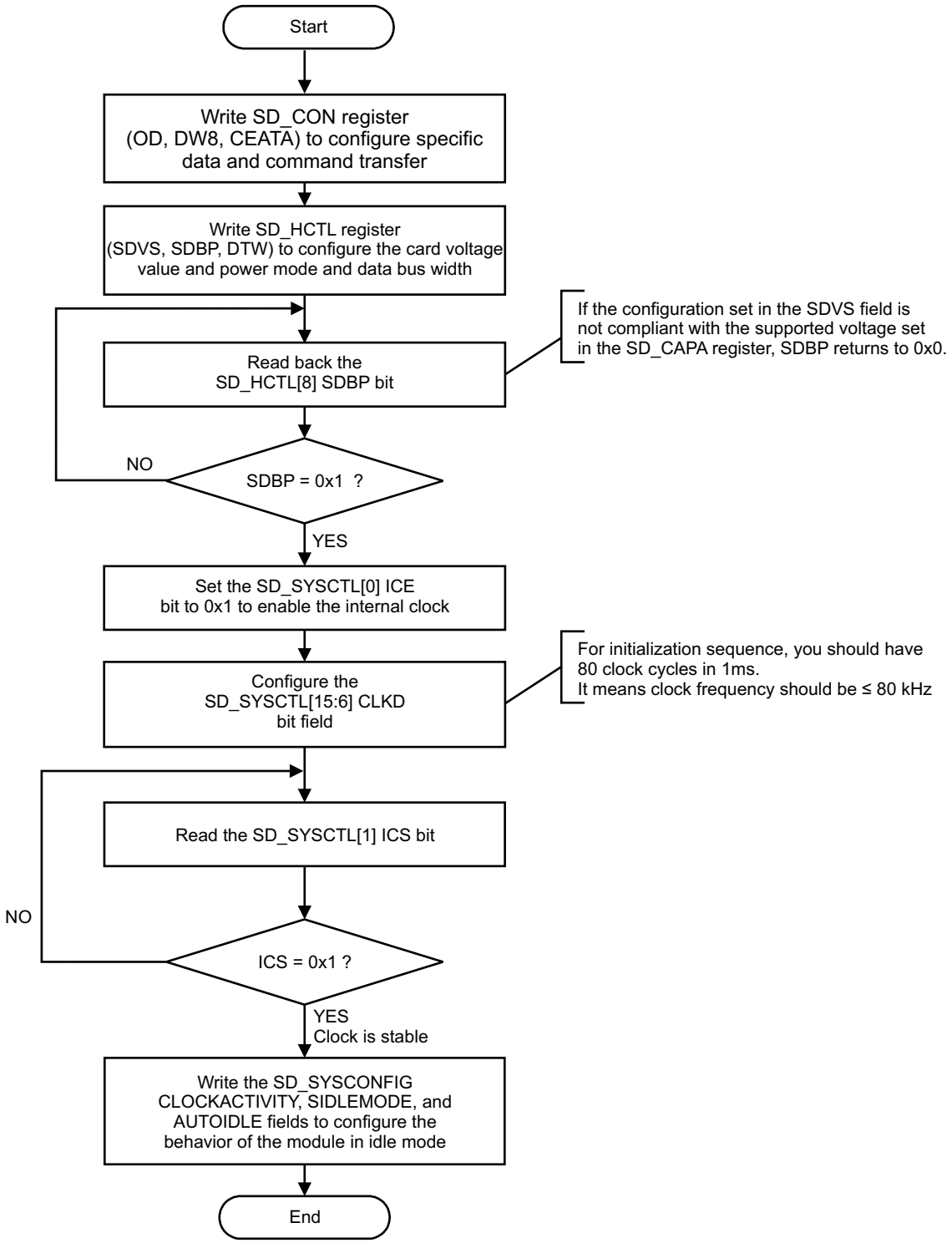
Table 13-12. SD/SDIO Controller Wake-Up Configuration

Step	Access Type	Register/Bit Field/Programming Model
Configure wake-up bit (if necessary).	W	SD_SYSCONFIG[2] ENAWAKEUP
Enable wake-up events on SD card interrupt (if necessary).	W	SD_HCTL[24] IWE
SDIO Card onlyEnable card interrupt (if necessary).	W	SD_IE[8] CIRQENABLE

13.3.2.5 Host and Bus Configuration

Figure 13-26 details the bus configuration process.

Figure 13-26. SD/SDIO Controller Bus Configuration Flow



13.3.3 Operational Modes Configuration

13.3.3.1 Basic Operations for SD/SDIO Host Controller

The SD/SDIO controller performs data transfers: data to card (referred to as write transfers) and data from card (referred to as read transfers).

The host controller requires transfers to run on a block-by-block basis, rather than on a DMA burst size basis. A single DMA request (or block request interrupt) is signaled for each block. Pipelining is supported as long as the block size is less than one half of the memory buffer size.

13.3.3.2 Card Detection, Identification, and Selection

Figure 13-27 and Figure 13-28 show the card identification and selection process.

Figure 13-27. SD/SDIO Controller Card Identification and Selection - Part 1

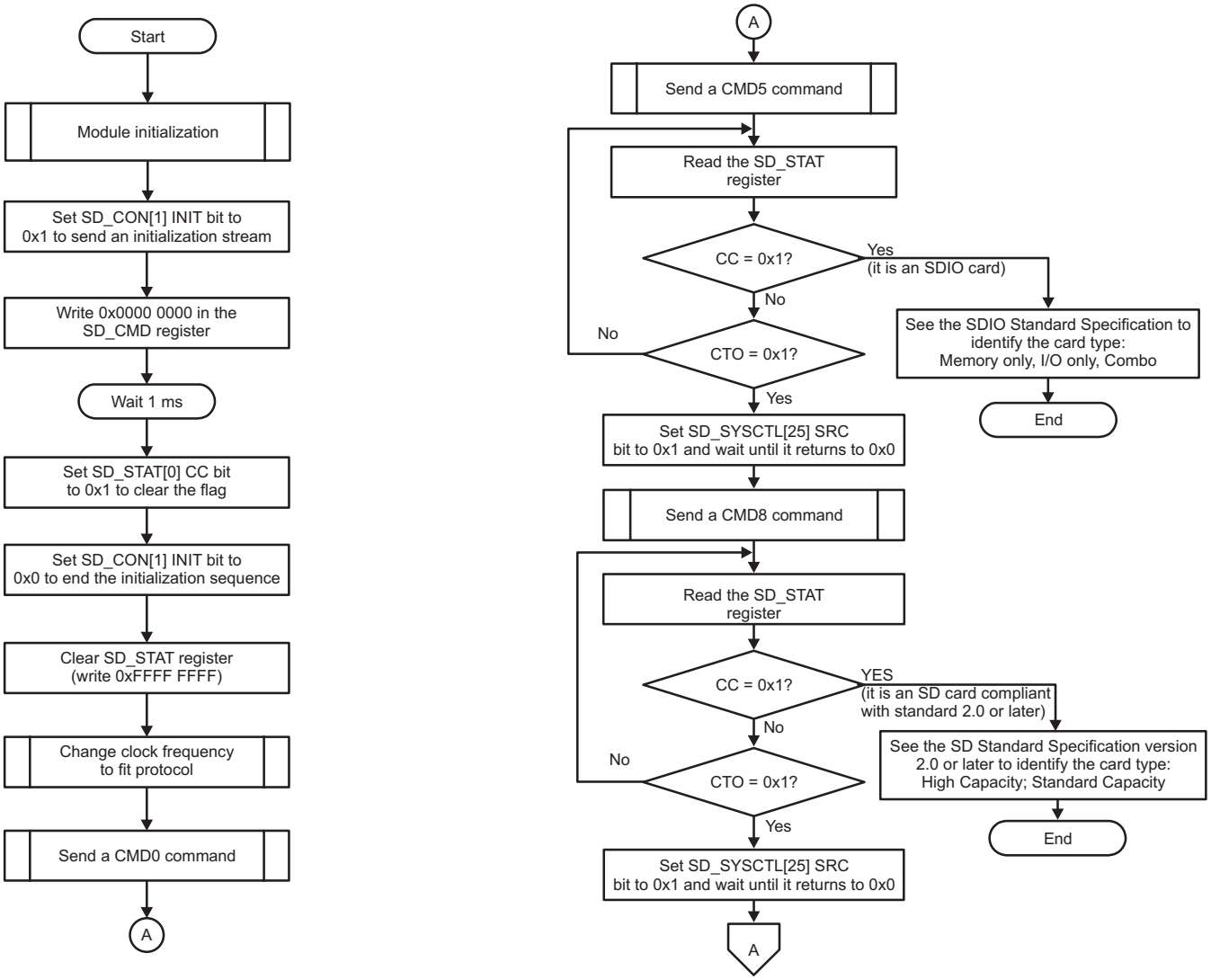
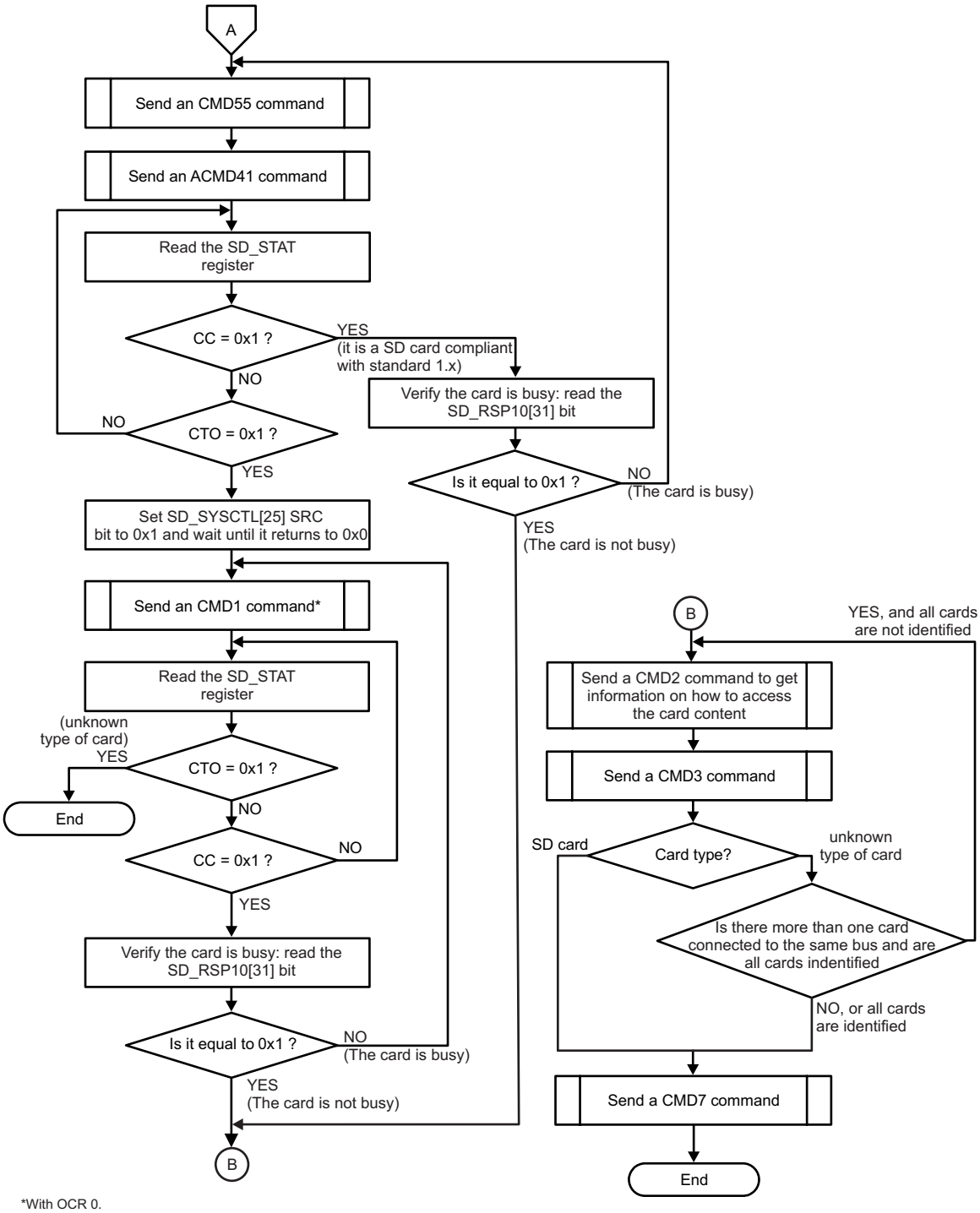


Figure 13-28. SD/SDIO Controller Card Identification and Selection - Part 2



*With OCR 0.

13.4 SD/SDIO Registers

Table 13-13 lists the SD/SDIO registers. For the base address of these registers, see Table 1-12.

CAUTION

The SD/SDIO registers are limited to 32-bit data accesses. 16-bit and 8-bit are not allowed and can corrupt register content.

Table 13-13. SD/SDIO Registers

Address Offset	Acronym	Register Name	Section
0	SD_HL_REV	IP Revision Identifier	Section 13.4.1
4h	SD_HL_HWINFO	Hardware Configuration	Section 13.4.2
10h	SD_HL_SYSCONFIG	Clock Management Configuration	Section 13.4.3
110h	SD_SYSCONFIG	System Configuration	Section 13.4.4
114h	SD_SYSSTATUS	System Status	Section 13.4.5
124h	SD_CSRE	Card status response error	Section 13.4.6
128h	SD_SYSTEST	System Test	Section 13.4.7
12Ch	SD_CON	Configuration	Section 13.4.8
130h	SD_PWCNT	Power counter	Section 13.4.9
200h	SD_SDMASA	SDMA System address:	Section 13.4.10
204h	SD_BLK	Transfer Length Configuration	Section 13.4.11
208h	SD_ARG	Command argument	Section 13.4.12
20Ch	SD_CMD	Command and transfer mode	Section 13.4.13
210h	SD_RSP10	Command Response 0 and 1	Section 13.4.14
214h	SD_RSP32	Command Response 2 and 3	Section 13.4.15
218h	SD_RSP54	Command Response 4 and 5	Section 13.4.16
21Ch	SD_RSP76	Command Response 6 and 7	Section 13.4.17
220h	SD_DATA	Data	Section 13.4.18
224h	SD_PSTATE	Present state	Section 13.4.19
228h	SD_HCTL	Host Control	Section 13.4.20
22Ch	SD_SYSCTL	SD system control	Section 13.4.21
230h	SD_STAT	SD interrupt status	Section 13.4.22
234h	SD_IE	SD interrupt enable	Section 13.4.23
238h	SD_ISE	SD interrupt enable set	Section 13.4.24
23Ch	SD_AC12	Auto CMD12 Error Status	Section 13.4.25
240h	SD_CAPA	Capabilities	Section 13.4.26
248h	SD_CUR_CAPA	Maximum current capabilities	Section 13.4.27
250h	SD_FE	Force Event	Section 13.4.28
254h	SD_ADMAES	ADMA Error Status	Section 13.4.29
258h	SD_ADMASAL	ADMA System address Low bits	Section 13.4.30
25Ch	SD_ADMASAH	ADMA System address High bits	Section 13.4.31
2FCh	SD_REV	Versions	Section 13.4.32

13.4.1 IP Revision Identifier Register (SD_HL_REV)

This register holds the IP Revision Identifier (X.Y.R) information.

Figure 13-29. SD_HL_REV Register

31	30	29	28	27	16
SCHEME		Reserved		FUNC	
R-0		R-0		R-20h	

15	11	10	8	7	6	5	0
R_RTL		X_MAJOR		CUSTOM		Y_MINOR	
R-0		R-x		R-0		R-x	

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

13.4.2 IP Module Hardware Configuration Register (SD_HL_HWINFO)

This register provides information about the IP module's hardware configuration, typically the module HDL generics (if any).

Figure 13-30. SD_HL_HWINFO Register

31	16
Reserved	
R-0	

15	2	1	0
Reserved		MEM_SIZE	MADMA_EN
R-0		R-0	R-x

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

13.4.3 Clock Management Configuration (SD_HL_SYSCONFIG)

This register provides information about the clock management configuration.

Figure 13-31. SD_HL_SYSCONFIG Register

31	16
Reserved	
R-0	

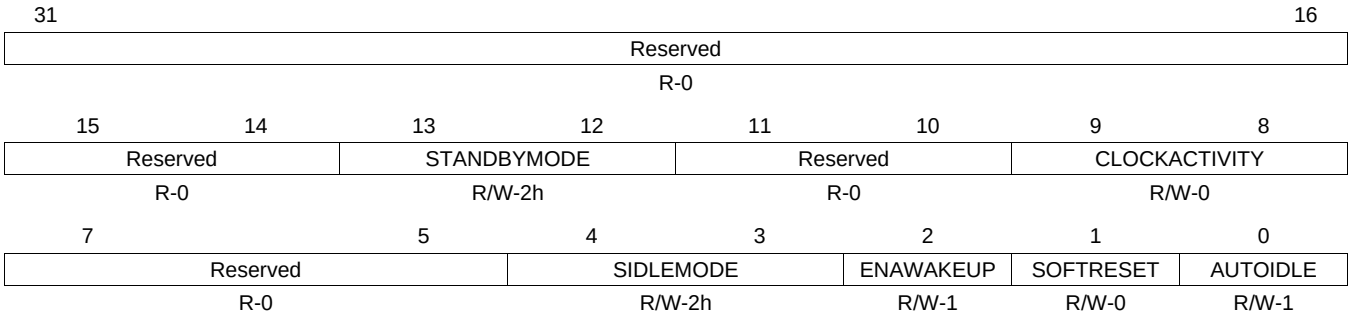
15	6	5	4	3	2	1	0
Reserved		STANDBYMODE		IDLEMODE		FREEEMU	SOFTRESET
R-0		R/W-2h		R/W-2h		R/W-0	R/W-0

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

13.4.4 System Configuration Register (SD_SYSCONFIG)

This register allows controlling various parameters of the OCP interface. The system configuration register (SD_SYSCONFIG) is shown in Figure 13-32 and described in Table 13-14.

Figure 13-32. System Configuration Register (SD_SYSCONFIG)



LEGENDR/W = Read/Write; R = Read only; *-n* = value after reset

Table 13-14. System Configuration Register (SD_SYSCONFIG) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	These bits are initialized to zero, and writes to them are ignored. Reads return 0.
13-12	STANDBYMODE		Master interface power Management, standby/wait control. The bit field is only useful when generic parameter MADMA_EN (Master ADMA enable) is set as active, otherwise it is a read only register read a 0.
		0	Force-standby. Mstandby is forced unconditionally.
		1h	No-standby. Mstandby is never asserted.
		2h	Smart-standby modelocal initiator standby status depends on local conditions, the module functional requirement from the initiator. IP module shall not generate (initiator-related) wake-up events.
		3h	Smart-Standby wake-up-capable modelocal initiator standby status depends on local conditions, the module functional requirement from the initiator. IP module can generate (master-related) wake-up events when in standby state. Mode is only relevant if the appropriate IP module "mwake-up" output is implemented. Functional clock is maintained. Interface clock may be switched off.
11-10	Reserved	0	These bits are initialized to zero, and writes to them are ignored. Reads return 0.
9-8	CLOCKACTIVITY		Clocks activity during wake up mode period.
		Bit 8	Interface clock
		Bit 9	Functional clock
		0	Interface and Functional clock may be switched off.
		1h	Interface clock is maintained. Functional clock may be switched-off.
		2h	Functional clock is maintained. Interface clock may be switched-off.
		3h	Interface and Functional clocks are maintained.
7-5	Reserved	0	These bits are initialized to zero, and writes to them are ignored. Reads return 0.
4-3	SIDLEMODE		Power management
		0	If an idle request is detected, the SD/SDIO host controller acknowledges it unconditionally and goes in Inactive mode. Interrupt and DMA requests are unconditionally deasserted.
		1h	If an idle request is detected, the request is ignored and the module keeps on behaving normally.
		2h	If an idle request is detected, the module will switch to wake up mode based on its internal activity, and the wake up capability can be used if the wake up capability is enabled (bit SD_SYSCONFIG[2] ENAWAKEUP bit is set to 1).
		3h	Smart-idle Wake-up capable modelocal target's idle state eventually acknowledges the system's idle requests, depending on the IP modules internal requirements. IP module may generate (IRQ- or DMA-request-related) wake-up events when in idle state. Mode is only relevant if the appropriate IP module "swake-up" output(s) is (are) implemented.

Table 13-14. System Configuration Register (SD_SYSCONFIG) Field Descriptions (continued)

Bit	Field	Value	Description
2	ENAWAKEUP		Wake-up feature control
		0	Wake-up capability is disabled.
		1	Wake-up capability is enabled.
1	SOFTRESET		Software reset. The bit is automatically reset by the hardware. During reset, it always returns 0.
		Read 0	Normal mode
		Write 0	No effect
		Read 1	The module is reset.
		Write 1	Trigger a module reset.
0	AUTOIDLE		Internal Clock gating strategy
		Read 0	Clocks are free-running.
		Write 1	Automatic clock gating strategy is applied, based on the interconnect and interface activity.

13.4.5 System Status Register (SD_SYSSTATUS)

This register provides status information about the module excluding the interrupt status information. The system status register (SD_SYSSTATUS) is shown in Figure 13-33 and described in Table 13-15.

Figure 13-33. System Status Register (SD_SYSSTATUS)

31	1	0
Reserved		RESETDONE
R-0		R-0

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-15. System Status Register (SD_SYSSTATUS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved bit field. Do not write any value.
0	RESETDONE		Internal Reset Monitoring. Notethe debounce clock , the interface clock and the functional clock shall be provided to the SD/SDIO host controller to allow the internal reset monitoring.
		Read 0	Internal module reset is on-going
		Read 1	Reset completed

13.4.6 Card Status Response Error (SD_CSRE)

This register enables the host controller to detect card status errors of response type R1, R1b for all cards and of R5, R5b and R6 response for cards types SD or SDIO. When a bit SD_CSRE[i] is set to 1, if the corresponding bit at the same position in the response SD_RSP10[I] is set to 1, the host controller indicates a card error (SD_STAT[28] CERR bit) interrupt status to avoid the host driver reading the response register (SD_RSP10).

NOTE: No automatic card error detection for autoCMD12 is implemented; the host system has to check autoCMD12 response register (SD_RSP76) for possible card errors.

The card status response error (SD_CSRE) is shown in Figure 13-34 and described in Table 13-16.

Figure 13-34. Card Status Response Error (SD_CSRE)

31	0
CSRE	
R/W-0	

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-16. Card Status Response Error (SD_CSRE) Field Descriptions

Bit	Field	Value	Description
31-0	CSRE		Card status response error

13.4.7 System Test Register (SD_SYSTEST)

This register is used to control the signals that connect to I/O pins when the module is configured in system test (SYSTEST) mode for boundary connectivity verification.

NOTE: In SYSTEST mode, a write into SD_CMD register will not start a transfer. The buffer behaves as a stack accessible only by the local host (push and pop operations). In this mode, the Transfer Block Size (SD_BLK[10:0] BLEN bits) and the Blocks count for current transfer (SD_BLK[31:16] NBLK bits) are needed to generate a Buffer write ready interrupt (SD_STAT[4] BWR bit) or a Buffer read ready interrupt (SD_STAT[5] BRR bit) and DMA requests if enabled.

The system test register (SD_SYSTEST) is shown in Figure 13-35 and described in Table 13-17.

Figure 13-35. System Test Register (SD_SYSTEST)

31						17	16
Reserved							OBI
R-0							R/W-0
15	14	13	12	11	10	9	8
SDCD	SDWP	WAKD	SSB	D7D	D6D	D5D	D4D
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
D3D	D2D	D1D	D0D	DDIR	CDAT	CDIR	MCKD
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-17. System Test Register (SD_SYSTEST) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved bit field. Do not write any value. Reads return 0.
16	OBI		Out-of-band interrupt (OBI) data value.
		0	The out-of-band interrupt pin is driven low.
		1	The out-of-band interrupt pin is driven high.
15	SDCD		Card detect input signal (SDCD) data value
		0	The card detect pin is driven low.
		1	The card detect pin is driven high.
14	SDWP		Write protect input signal (SDWP) data value
		0	The write protect pin SDWP is driven low.
		1	he write protect pin SDWP is driven high.
13	WAKD		Wake request output signal data value
		Read 0	No action. Returns 0.
		Write 0	The pin SWAKEUP is driven low.
		Read 1	No action. Returns 1.
		Write 1	The pin SWAKEUP is driven high.
12	SSB		Set status bit. This bit must be cleared prior attempting to clear a status bit of the interrupt status register (SD_STAT).
		Read 0	No action. Returns 0.
		Write 0	Clear this SSB bit field. Writing 0 does not clear already set status bits.
		Read 1	No action. Returns 1.
		Write 1	Force to 1 all status bits of the interrupt status register (SD_STAT) only if the corresponding bit field in the Interrupt signal enable register (SD_ISE) is set.

Table 13-17. System Test Register (SD_SYSTEST) Field Descriptions (continued)

Bit	Field	Value	Description
11	D7D		DAT7 input/output signal data value
		Read 0	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT7 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.
		Write 0	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT7 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
		Read 1	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT7 line (high) If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.
		Write 1	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT7 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
10	D6D		DAT6 input/output signal data value
		Read 0	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT6 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.
		Write 0	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT6 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
		Read 1	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT6 line (high) If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.
		Write 1	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT6 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
9	D5D		DAT5 input/output signal data value
		Read 0	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT5 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.
		Write 0	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT5 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
		Read 1	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT5 line (high) If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.
		Write 1	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT5 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
8	D4D		DAT4 input/output signal data value
		Read 0	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT4 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.
		Write 0	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT4 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
		Read 1	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT4 line (high) If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.
		Write 1	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT4 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
7	D3D		DAT3 input/output signal data value
		Read 0	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT3 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.
		Write 0	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT3 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
		Read 1	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT3 line (high) If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.
		Write 1	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT3 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
6	D2D		DAT2 input/output signal data value
		Read 0	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT2 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.
		Write 0	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT2 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
		Read 1	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT2 line (high) If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.
		Write 1	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT2 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.

Table 13-17. System Test Register (SD_SYSTEST) Field Descriptions (continued)

Bit	Field	Value	Description
5	D1D		DAT1 input/output signal data value
		Read 0	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT1 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.
		Write 0	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT1 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
		Read 1	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT1 line (high) If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.
		Write 1	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT1 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
4	D0D		DAT0 input/output signal data value
		Read 0	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT0 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.
		Write 0	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT0 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
		Read 1	If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT0 line (high) If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.
		Write 1	If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT0 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.
3	DDIR		Control of the DAT[7:0] pins direction
		Read 0	No action. Returns 0.
		Write 0	The DAT lines are outputs (host to card).
		Read 1	No action. Returns 1.
		Write 1	The DAT lines are inputs (card to host).
2	CDAT		CMD input/output signal data value
		Read 0	If SD_SYSTEST[1] CDIR bit = 1 (input mode direction), returns the value on the CMD line (low). If SD_SYSTEST[1] CDIR bit = 0 (output mode direction), returns 0 .
		Write 0	If SD_SYSTEST[1] CDIR bit = 0 (output mode direction), the CMD line is driven low. If SD_SYSTEST[1] CDIR bit = 1 (input mode direction), no effect.
		Read 1	If SD_SYSTEST[1] CDIR bit = 1 (input mode direction), returns the value on the CMD line (high) If SD_SYSTEST[1] CDIR bit = 0 (output mode direction), returns 1 .
		Write 1	If SD_SYSTEST[1] CDIR bit = 0 (output mode direction), the CMD line is driven high. If SD_SYSTEST[1] CDIR bit = 1 (input mode direction), no effect.
1	CDIR		Control of the CMD pin direction
		Read 0	No action. Returns 0.
		Write 0	The CMD line is an output (host to card).
		Read 1	No action. Returns 1.
		Write 1	The CMD line is an input (card to host) .
0	MCKD		clock output signal data value
		Read 0	No action. Returns 0.
		Write 0	The output clock is driven low.
		Read 1	No action. Returns 1.
		Write 1	The output clock is driven high.

13.4.8 Configuration Register (SD_CON)

This register is used:

- To select the functional mode for any card
- To send an initialization sequence to any card
- To send an initialization sequence to any card
- To enable the detection on the SD_DAT1 signal of a card interrupt for SDIO cards only

It also configures:

- The parameters related to the card detect and write protect input signals

The configuration register (SD_CON) is shown in Figure 13-36 and described in Table 13-18.

Figure 13-36. Configuration Register (SD_CON)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved		SDMA_LnE	DMA_MnS	DDR	BOOT_CF0	BOOT_ACK	CLKEXTFREE
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8
Reserved				CTPL	DVAL		WPP
R-0				R/W-0	R/W-3h		R/W-0

7	6	5	4	3	2	1	0
CDP	Reserved		MODE	Reserved		INIT	Reserved
R/W-0	R-0		R/W-0	R-0		R/W-0	R-0

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-18. Configuration Register (SD_CON) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved	0	Reserved bit field. Do not write any value.
21	SDMA_LnE		Slave DMA Level/Edge Request. The waveform of the DMA request can be configured either edge sensitive with early de-assertion on first access to SD_DATA register or late de-assertion, request remains active until last allowed data written into SD_DATA.
		0	Slave DMA edge sensitive.
		1	Slave DMA level sensitive.
20	DMA_MnS		DMA Master or Slave selection. When this bit is set and the controller is configured to use the DMA, Ocp master interface is used to get datas from system using ADMA2 procedure (direct access to the memory). This option is only available if generic parameter MADMA_EN is asserted to 1.
		0	The controller is slave on data transfers with system.
		1	Not available on this device.
19	DDR		Dual Data Rate mode. When this register is set, the controller uses both clock edge to emit or receive data. Odd bytes are transmitted on falling edges and even bytes are transmitted on rise edges. It only applies on Data bytes and CRC, Start, end bits and CRC status are kept full cycle. This bit field is only meaningful and active for even clock divider ratio of SD_SYSCTL[CLKD], it is insensitive to SD_HCTL[HSPE] setting
		0	Standard modeData are transmitted on a single edge.
		1	Data Bytes and CRC are transmitted on both edges.

Table 13-18. Configuration Register (SD_CON) Field Descriptions (continued)

Bit	Field	Value	Description
18	BOOT_CF0		Boot Status Supported. This register is set when the CMD line needs to be forced to 0 for a boot sequence. CMD line is driven to 0 after writing in SD_CMD. The line is released when this bit field is de-asserted and aborts data transfer in case of a pending transaction.
		Read 0	CMD line not forced.
		Read 1	CMD line is released when it was previously forced to 0 by a boot sequence.
		Write 0	CMD line forced to 0 is enabled.
		Write 1	CMD line forced to 0 is enabled and will be active after writing into SD_CMD register.
17	BOOT_ACK		Book acknowledge received. When this bit is set the controller should receive a boot status on DAT0 line after next command issued. If no status is received a data timeout will be generated.
		0	No acknowledge to be received.
		1	A boot status will be received on DAT0 line after issuing a command.
16	CLKEXTFREE		External clock free running. This register is used to maintain card clock out of transfer transaction to enable slave module (for example to generate a synchronous interrupt on SD_DAT1). The Clock will be maintain only if SD_SYSCTL[2] CEN bit is set.
		0	External card clock is cut off outside active transaction period.
		1	External card clock is maintain even out of active transaction period only if SD_SYSCTL[2] CEN bit is set.
15-12	Reserved	0	Reserved bit field. Do not write any value.
11	CTPL		Control Power for SD_DAT1 line (SD cards). By default, this bit is cleared to 0 and the host controller automatically disables all the input buffers outside of a transaction to minimize the leakage current.
			SDIO cards. When this bit is set to 1, the host controller automatically disables all the input buffers except the buffer of SD_DAT1 outside of a transaction in order to detect asynchronous card interrupt on SD_DAT1 line and minimize the leakage current of the buffers.
		0	Disable all the input buffers outside of a transaction.
		1	Disable all the input buffers except the buffer of SD_DAT1 outside of a transaction.
10-9	DVAL		Debounce filter value (all cards). This register is used to define a debounce period to filter the card detect input signal (SDCD). The usage of the card detect input signal (SDCD) is optional and depends on the system integration and the type of the connector housing that accommodates the card.
		0	33 us debounce period
		1h	231 us debounce period
		2h	1 ms debounce period
		3h	8.4 ms debounce period
8	WPP		Write protect polarity (SD and SDIO cards only). This bit selects the active level of the write protect input signal (SDWP). The usage of the write protect input signal (SDWP) is optional and depends on the system integration and the type of the connector housing that accommodates the card.
		0	Active high level
		1	Active low level
7	CDP		Card detect polarity (all cards). This bit selects the active level of the write protect input signal (SDWP). The usage of the write protect input signal (SDWP) is optional and depends on the system integration and the type of the connector housing that accommodates the card.
		0	Active high level
		1	Active low level
6-5	Reserved	0	Reserved bit field. Do not write any value.
4	MODE		Mode select (all cards). This bit selects the functional mode.
		0	Functional mode. Transfers to the SD/SDIO cards follow the card protocol. The clock is enabled. SD transfers are operated under the control of the SD_CMD register.
		1	SYSTEST mode. SYSTEST mode. The signal pins are configured as general-purpose input/output and the 1024-byte buffer is configured as a stack memory accessible only by the local host or system DMA. The pins retain their default type (input, output or in-out). SYSTEST mode is operated under the control of the SYSTEST register.
3-2	Reserved	0	Reserved bit field. Do not write any value.

Table 13-18. Configuration Register (SD_CON) Field Descriptions (continued)

Bit	Field	Value	Description
1	INIT		Send initialization stream (all cards). When this bit is set to 1, and the card is idle, an initialization sequence is sent to the card. An initialization sequence consists of setting the SD_CMD line to 1 during 80 clock cycles. The initialization sequence is mandatory - but it is not required to do it through this bit - this bit makes it easier. Clock divider (SD_SYSCTL[15:6] CLKD bits) should be set to ensure that 80 clock periods are greater than 1ms. Notein this mode, there is no command sent to the card and no response is expected. A command complete interrupt will be generated once the initialization sequence is completed. SD_STAT[0] CC bit can be polled.
		0	The host does not send an initialization sequence
		1	The host sends an initialization sequence
0	Reserved	0	Reserved bit field. Do not write any value.

13.4.9 Power Counter Register (SD_PWCNT)

This register is used to program a counter to delay command transfers after activating the PAD power, this value depends on PAD characteristics and voltage. The power counter register (SD_PWCNT) is shown in Figure 13-37 and described in Table 13-19.

Figure 13-37. Power Counter Register (SD_PWCNT)

31	16	15	0
Reserved		PWRCNT	
R-0		R/W-0	

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-19. Power Counter Register (SD_PWCNT) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	These bits are initialized to zero, and writes to them are ignored. Reads return 0.
15-0	PWRCNT		Power counter register. This register is used to introduce a delay between the PAD ACTIVE pin assertion and the command issued.
		0	No additional delay added
		1h	TCF delay (card clock period)
		2h	TCF x 2 delay (card clock period)
		FFFEh	TCF x 65534 delay (card clock period)
		FFFFh	TCF x 65535 delay (card clock period)

13.4.10 SDMA System Address (SD_SDMASA)

This register is used to program a counter to delay command transfers after activating the PAD power. This value depends on PAD characteristics and voltage. The SDMA System address (SD_SDMASA) is shown in Figure 13-34 and described in Table 13-16.

Figure 13-38. Card Status Response Error (SD_SDMASA)

31	0
SDMA_SYSADDR	
R-0	

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-20. Card Status Response Error (SD_SDMASA) Field Descriptions

Bit	Field	Value	Description
31-0	SDMA_SYSADDR		This register contains the system memory address for a SDMA transfer. When the Host Controller stops a SDMA transfer, this register shall point to the system address of the next contiguous data position. It can be accessed only if no transaction is executing (after a transaction has stopped). Read operations during transfers may return an invalid value. The Host Driver shall initialize this register before starting a SDMA transaction. After SDMA has stopped, the next system address of the next contiguous data position can be read from this register. The SDMA transfer waits at the every boundary specified by the Host SDMA Buffer Boundary in the Block Size register. The Host Controller generates DMA Interrupt to request the Host Driver to update this register. The Host Driver sets the next system address of the next data position to this register. When the most upper byte of this register (003h) is written, the Host Controller restarts the SDMA transfer. When restarting SDMA by the Resume command or by setting Continue Request in the Block Gap Control register, the Host Controller shall start at the next contiguous address stored here in the SDMA System Address register. ADMA does not use this register.

13.4.11 Transfer Length Configuration Register (SD_BLK)

This register shall be used for any card. SD_BLK[BLEN] is the block size register. SD_BLK[NBLK] is the block count register.

The transfer length configuration register (SD_BLK) is shown in Figure 13-39 and described in Table 13-21.

Figure 13-39. Transfer Length Configuration Register (SD_BLK)

31 16	15 12	11 0
NBLK	Reserved	BLEN
R/W-0	R-0	R/W-0

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-21. Transfer Length Configuration Register (SD_BLK) Field Descriptions

Bit	Field	Value	Description
31-16	NBLK		Blocks count for current transfer. This register is enabled when Block count Enable (SD_CMD[1] BCE bit) is set to 1 and is valid only for multiple block transfers. Setting the block count to 0 results no data blocks being transferred. NoteThe host controller decrements the block count after each block transfer and stops when the count reaches zero. This register can be accessed only if no transaction is executing (after a transaction has stopped). Read operations during transfers may return an invalid value and write operation will be ignored. In suspend context, the number of blocks yet to be transferred can be determined by reading this register. When restoring transfer context prior to issuing a Resume command, The local host shall restore the previously saved block count.
		0	Stop count
		1h	1 block
		2h	2 blocks
		FFFFh	65535 blocks
15-12	Reserved	0	Reserved bit field. Do not write any value.
11-0	BLEN		Transfer block size. This register specifies the block size for block data transfers. Read operations during transfers may return an invalid value, and write operations are ignored. When a CMD12 command is issued to stop the transfer, a read of the BLEN field after transfer completion (SD_STAT[1] TC bit set to 1) will not return the true byte number of data length while the stop occurs but the value written in this register before transfer is launched.
		0	No data transfer
		1h	1 byte block length
		2h	2 bytes block length
		3h	3 bytes block length
		1FFh	511 bytes block length
		200h	512 bytes block length
		7FFh	2047 bytes block length
		800h	2048 bytes block length

13.4.12 Command Argument Register (SD_ARG)

This register contains command argument specified as bit 39-8 of Command-Format. These registers must be initialized prior to sending the command itself to the card (write action into the register SD_CMD register). Only exception is for a command index specifying stuff bits in arguments, making a write unnecessary.

The command argument register (SD_ARG) is shown in Figure 13-40 and described in Table 13-22.

Figure 13-40. Command Argument Register (SD_ARG)

31	0
ARG	
R/W-0	

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-22. Command Argument Register (SD_ARG) Field Descriptions

Bit	Field	Value	Description
31-0	ARG		Command argument bits [31:0] .

13.4.13 Command and Transfer Mode Register (SD_CMD)

- SD_CMD[31:16] = the command register
- SD_CMD[15:0] = the transfer mode

This register configures the data and command transfers. A write into the most significant byte send the command. A write into SD_CMD[15:0] registers during data transfer has no effect.

This register can be used for any card.

NOTE: In SYSTEST mode, a write into SD_CMD register will not start a transfer.

The command and transfer mode register (SD_CMD) is shown in Figure 13-41 and described in Table 13-23.

Figure 13-41. Command and Transfer Mode Register (SD_CMD)

31	30	29					24
Reserved		INDX					
R-0		R/W-0					

23	22	21	20	19	18	17	16
CMD_TYPE		DP	CICE	CCCE	Reserved	RSP_TYPE	
R/W-0		R/W-0	R/W-0	R/W-0	R-0	R/W-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved		MSBS	DDIR	Reserved	ACEN	BCE	DE
R-0		R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-23. Command and Transfer Mode Register (SD_CMD) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved bit field. Do not write any value.
29-24	INDX		Command index binary encoded value from 0 to 63 specifying the command number send to card.
		0	CMD0 or ACMD0
		1h	CMD1 or ACMD1
		2h	CMD2 or ACMD2
		3h	CMD3 or ACMD3
		4h	CMD4 or ACMD4
		5h	CMD5 or ACMD5
		6h	CMD6 or ACMD6
		7h	CMD7 or ACMD7
		8h	CMD8 or ACMD8
		9h	CMD9 or ACMD9
		Ah	CMD10 or ACMD10
		Bh	CMD11 or ACMD11
		Ch	CMD12 or ACMD12
		Dh	CMD13 or ACMD13
		Eh	CMD14 or ACMD14
		Fh	CMD15 or ACMD15
		10h	CMD16 or ACMD16
		11h	CMD17 or ACMD17
		12h	CMD18 or ACMD18
		13h	CMD19 or ACMD19
		14h	CMD20 or ACMD20
		15h	CMD21 or ACMD21
		16h	CMD22 or ACMD22
		17h	CMD23 or ACMD23
		18h	CMD24 or ACMD24
		19h	CMD25 or ACMD25
		1Ah	CMD26 or ACMD26
		1Bh	CMD27 or ACMD27
		1Ch	CMD28 or ACMD28
		1Dh	CMD29 or ACMD29
		1Eh	CMD30 or ACMD30
		1Fh	CMD31 or ACMD31
		20h	CMD32 or ACMD32
		21h	CMD33 or ACMD33
		22h	CMD34 or ACMD34
		23h	CMD35 or ACMD35
		24h	CMD36 or ACMD36
		25h	CMD37 or ACMD37
		26h	CMD38 or ACMD38
		27h	CMD39 or ACMD39
		28h	CMD40 or ACMD40
		29h	CMD41 or ACMD41
		2Ah	CMD42 or ACMD42
		2Bh	CMD43 or ACMD43
		2Ch	CMD44 or ACMD44
		2Dh	CMD45 or ACMD45

Table 13-23. Command and Transfer Mode Register (SD_CMD) Field Descriptions (continued)

Bit	Field	Value	Description
		2Eh	CMD46 or ACMD46
		2Fh	CMD47 or ACMD47
		30h	CMD48 or ACMD48
		31h	CMD49 or ACMD49
		32h	CMD50 or ACMD5
		33h	CMD51 or ACMD5
		34h	CMD52 or ACMD5
		35h	CMD53 or ACMD5
		36h	CMD54 or ACMD5
		37h	CMD55 or ACMD5
		38h	CMD56 or ACMD5
		39h	CMD57 or ACMD5
		3Ah	CMD58 or ACMD5
		3Bh	CMD59 or ACMD5
		3Ch	CMD60 or ACMD60
		3Dh	CMD61 or ACMD61
		3Eh	CMD62 or ACMD62
		3Fh	CMD63 or ACMD63
23-22	CMD_TYPE		Command type. This register specifies three types of special commandSuspend, Resume and Abort. These bits shall be cleared to 0b00 for all other commands.
		0	Others commands
		1h	Upon CMD52 "Bus Suspend" operation
		2h	Upon CMD52 "Function Select" operation
		3h	Upon CMD12 or CMD52 "I/O Abort" command
21	DP		Data present select. This register indicates that data is present and SD_DAT line shall be used. It must be cleared to 0 in the following conditions: • Command using only SD_CMD line • Command with no data transfer but using busy signal on SD_DAT0 • Resume command
		0	Command with no data transfer
		1	Command with data transfer
20	CICE		Command Index check enable. This bit must be set to 1 to enable index check on command response to compare the index field in the response against the index of the command. If the index is not the same in the response as in the command, it is reported as a command index error (SD_STAT[19] CIE bit set to1) NoteThe CICE bit cannot be configured for an Auto CMD12, then index check is automatically checked when this command is issued.
		0	Index check disable
		1	Index check enable
19	CCCE		Command CRC check enable. This bit must be set to 1 to enable CRC7 check on command response to protect the response against transmission errors on the bus. If an error is detected, it is reported as a command CRC error (SD_STAT[17] CCRC bit set to 1). NoteThe CCCE bit cannot be configured for an Auto CMD12, and then CRC check is automatically checked when this command is issued.
		0	CRC7 check disable
		1	CRC7 check enable
18	Reserved	0	Reserved bit field. Do not write any value.
17-16	RSP_TYPE		Response type. This bits defines the response type of the command.
		0	No response
		1h	Response Length 136 bits
		2h	Response Length 48 bits
		3h	Response Length 48 bits with busy after response

Table 13-23. Command and Transfer Mode Register (SD_CMD) Field Descriptions (continued)

Bit	Field	Value	Description
15-6	Reserved	0	Reserved bit field. Do not write any value.
5	MSBS		Multi/Single block select. This bit must be set to 1 for data transfer in case of multi block command. For any others command this bit shall be cleared to 0.
		0	Single block. If this bit is 0, it is not necessary to set the register SD_BLK[31:16] NBLK bits.
		1	Multi block. When Block Count is disabled (SD_CMD[1] BCE bit is cleared to 0) in Multiple block transfers (SD_CMD[5] MSBS bit is set to 1), the module can perform infinite transfer.
4	DDIR		Data transfer Direction. Select This bit defines either data transfer will be a read or a write.
		0	Data Write (host to card)
		1	Data Read (card to host)
3	Reserved	0	Reserved bit field. Do not write any value.
2	ACEN		Auto CMD12 Enable (SD cards only). When this bit is set to 1, the host controller issues a CMD12 automatically after the transfer completion of the last block. The Host Driver shall not set this bit to issue commands that do not require CMD12 to stop data transfer. In particular, secure commands do not require CMD12. For CE-ATA commands (SD_CON[12] CEATA bit set to 1), auto CMD12 is useless; therefore when this bit is set the mechanism to detect command completion signal, named CCS, interrupt is activated.
		0	Auto CMD12 disable
		1	Auto CMD12 enable or CCS detection enabled.
1	BCE		Block Count Enable (Multiple block transfers only). This bit is used to enable the block count register (SD_BLK[31:16] NBLK bits). When Block Count is disabled (SD_CMD[1] BCE bit is cleared to 0) in Multiple block transfers (SD_CMD[5] MSBS bits is set to 1), the module can perform infinite transfer.
		0	Block count disabled for infinite transfer.
		1	Block count enabled for multiple block transfer with known number of blocks
0	DE		DMA Enable. This bit is used to enable DMA mode for host data access.
		0	DMA mode disable
		1	DMA mode enable

13.4.14 Command Response[31:0] Register (SD_RSP10)

This 32-bit register holds bits positions [31:0] of command response type R1, R1b, R2, R3, R4, R5, R5b, or R6. The command response[31:0] register (SD_RSP10) is shown in Figure 13-42 and described in Table 13-24.

Figure 13-42. Command Response[31:0] Register (SD_RSP10)

31	16	15	0
RSP1		RSP0	
R-0		R-0	

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-24. Command Response[31:0] Register (SD_RSP10) Field Descriptions

Bit	Field	Value	Description
31-16	RSP1		Command Response [31:16]
15-0	RSP0		Command Response [15:0]

13.4.15 Command Response[63:32] Register (SD_RSP32)

This 32-bit register holds bits positions [63:32] of command response type R2. The command response[63:32] register (SD_RSP32) is shown in Figure 13-43 and described in Figure 13-43.

Figure 13-43. Command Response[63:32] Register (SD_RSP32)

31	16	15	0
RSP3		RSP2	
R-0		R-0	

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-25. Command Response[63:32] Register (SD_RSP32) Field Descriptions

Bit	Field	Value	Description
31-16	RSP3		Command Response [63:48]
15-0	RSP2		Command Response [47:32]

13.4.16 Command Response[95:64] Register (SD_RSP54)

This 32-bit register holds bits positions [95:64] of command response type R2. The command response[95:64] register (SD_RSP54) is shown in Figure 13-44 and described in Table 13-26.

Figure 13-44. Command Response[95:64] Register (SD_RSP54)

31	16	15	0
RSP5		RSP4	
R-0		R-0	

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-26. Command Response[95:64] Register (SD_RSP54) Field Descriptions

Bit	Field	Value	Description
31-16	RSP5		Command Response [95:80]
15-0	RSP4		Command Response [79:64]

13.4.17 Command Response[127:96] Register (SD_RSP76)

This 32-bit register holds bits positions [127:96] of command response type R2. The command response[127:96] register (SD_RSP76) is shown in Figure 13-45 and described in Table 13-27.

Figure 13-45. Command Response[127:96] Register (SD_RSP76)

31	16	15	0
RSP7		RSP6	
R-0		R-0	

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-27. Command Response[127:96] Register (SD_RSP76) Field Descriptions

Bit	Field	Value	Description
31-16	RSP7		Command Response [127:112]
15-0	RSP6		Command Response [111:96]

13.4.18 Data Register (SD_DATA)

This register is the 32-bit entry point of the buffer for read or write data transfers. The buffer size is 32bitsx256(1024 bytes). Bytes within a word are stored and read in little endian format. This buffer can be used as two 512 byte buffers to transfer data efficiently without reducing the throughput.

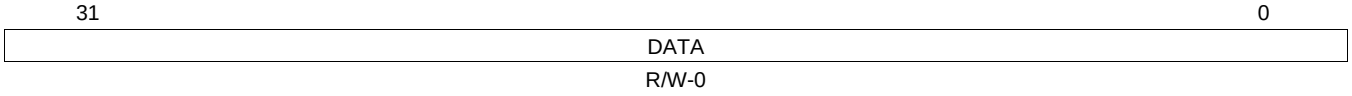
Sequential and contiguous access is necessary to increment the pointer correctly. Random or skipped access is not allowed. In little endian, if the local host accesses this register byte-wise or 16bit-wise, the least significant byte (bits [7:0]) must always be written/read first. The update of the buffer address is done on the most significant byte write for full 32-bit DATA register or on the most significant byte of the last word of block transfer.

Example 1Byte or 16-bit access

- Mbyteen[3:0]=0001 (1-byte) => Mbyteen[3:0]=0010 (1-byte) => Mbyteen[3:0]=1100 (2-bytes) OK
- Mbyteen[3:0]=0001 (1-byte) => Mbyteen[3:0]=0010 (1-byte) => Mbyteen[3:0]=0100 (1-byte) OK
- Mbyteen[3:0]=0001 (1-byte) => Mbyteen[3:0]=0010 (1-byte) => Mbyteen[3:0]=1000 (1-byte) Bad

The data register (SD_DATA) is shown in Figure 13-46 and described in Table 13-28.

Figure 13-46. Data Register (SD_DATA)



LEGENDR/W = Read/Write; R = Read only; *-n* = value after reset

Table 13-28. Data Register (SD_DATA) Field Descriptions

Bit	Field	Value	Description
31-0	DATA		Data register [31:0]. In functional mode (SD_CON[4] MODE bit set to the default value 0): A read access to this register is allowed only when the buffer read enable status is set to 1 (SD_PSTATE[11] BRE bit), otherwise a bad access (SD_STAT[29] BADA bit) is signaled. A write access to this register is allowed only when the buffer write enable status is set to 1 (SD_PSTATE[10] BWE bit), otherwise a bad access (SD_STAT[29] BADA bit) is signaled and the data is not written.

13.4.19 Present State Register (SD_PSTATE)

The Host can get the status of the Host controller from this 32-bit read only register. The present state register (SD_PSTATE) is shown in Figure 13-47 and described in Table 13-29.

Figure 13-47. Present State Register (SD_PSTATE)

31	25	24	23	20	19	18	17	16
Reserved		CLEV	DLEV		WP	CDPL	CSS	CINS
R-0		R-x	R-x		R-0	R-0	R-0	R-0

15	12	11	10	9	8	7	3	2	1	0
Reserved		BRE	BWE	RTA	WTA	Reserved		DLA	DATI	CMDI
R-0		R-0	R-0	R-0	R-0	R-0		R-0	R-0	R-0

LEGENDR/W = Read/Write; R = Read only; *-n* = value after reset

Table 13-29. Present State Register (SD_PSTATE) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved bit field. Do not write any value.
24	CLEV		SD_CMD line signal level. This status is used to check the SD_CMD line level to recover from errors, and for debugging. The value of this register after reset depends on the SD_CMD line level at that time.
		Read 0	The SD_CMD line level is 0.
		Read 1	The SD_CMD line level is 1.
23-20	DLEV		SD_DAT[3:0] line signal level • SD_DAT3 => bit 23 • SD_DAT2 => bit 22 • SD_DAT1 => bit 21 • SD_DAT0 => bit 20 This status is used to check SD_DAT line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from SD_DAT0. The value of these registers after reset depends on the SD_DAT lines level at that time.
19	WP		Write Protect. SD/SDIO1 only. SDIO cards only. This bit reflects the write protect input pin (SDWP) level. The value of this register after reset depends one the protect input pin (SDWP) level at that time.
		Read 0	If SD_CON[8] WPP is cleared to 0 (default), the card is write protected, otherwise the card is not write protected.
		Read 1	If SD_CON[8] WPP is cleared to 0 (default), the card is not write protected, otherwise the card is write protected.
18			Card Detect Pin Level. SD/SDIO1 only. SDIO cards only. This bit reflects the inverse value of the card detect input pin (SDCD). Debouncing is not performed on this bit and is valid only when Card State is stable. (SD_PSTATE[17] is set to 1). This bit must be debounced by software. The value of this register after reset depends on the card detect input pin (SDCD) level at that time.
		Read 0	The value of the card detect input pin (SDCD) is 1.
		Read 1	The value of the card detect input pin (SDCD) is 0.
17			Card State Stable. This bit is used for testing. It is set to 1 only when Card Detect Pin Level is stable (SD_PSTATE[18] CPDL). Debouncing is performed on the card detect input pin (SDCD) to detect card stability. This bit is not affected by software reset.
		Read 0	Reset or Debouncing.
		Read 1	Reset or Debouncing.

Table 13-29. Present State Register (SD_PSTATE) Field Descriptions (continued)

Bit	Field	Value	Description
16			Card inserted. This bit is the debounced value of the card detect input pin (SDCD). An inactive to active transition of the card detect input pin (SDCD) will generate a card insertion interrupt (SD_STAT[CINS]). A active to inactive transition of the card detect input pin (SDCD) will generate a card removal interrupt (SD_STAT[REM]). This bit is not affected by a software reset.
		Read 0	If SD_CON[CDP] is cleared to 0 (default), no card is detected. The card may have been removed from the card slot. If SD_CON[CDP] is set to 1, the card has been inserted.
		Read 1	If SD_CON[CDP] is cleared to 0 (default), the card has been inserted from the card slot. If SD_CON[CDP] is set to 1, no card is detected. The card may have been removed from the card slot.
15-12	Reserved	0	Reserved bit field. Do not write any value.
11	BRE		Buffer read enable. This bit is used for non-DMA read transfers. It indicates that a complete block specified by SD_BLK[10:0] BLEN bits has been written in the buffer and is ready to be read. It is cleared to 0 when the entire block is read from the buffer. It is set to 1 when a block data is ready in the buffer and generates the Buffer read ready status of interrupt (SD_STAT[5] BRR bit).
		Read 0	Read BLEN bytes disable
		Read 1	Read BLEN bytes enable. Readable data exists in the buffer.
10	BWE		Buffer Write enable. This status is used for non-DMA write transfers. It indicates if space is available for write data.
		Read 0	There is no room left in the buffer to write BLEN bytes of data.
		Read 1	There is enough space in the buffer to write BLEN bytes of data.
9	RTA		Read transfer active. This status is used for detecting completion of a read transfer. It is set to 1 after the end bit of read command or by activating a continue request (SD_HCTL[17] CR bit) following a stop at block gap request. This bit is cleared to 0 when all data have been read by the local host after last block or after a stop at block gap request.
		Read 0	No valid data on the SD_DAT lines.
		Read 1	Read data transfer on going.
8	WTA		Write transfer active. This status indicates a write transfer active. It is set to 1 after the end bit of write command or by activating a continue request (SD_HCTL[17] CR bit) following a stop at block gap request. This bit is cleared to 0 when CRC status has been received after last block or after a stop at block gap request.
		Read 0	No valid data on the SD_DAT lines.
		Read 1	Write data transfer on going.
7-3	Reserved	0	Reserved bit field. Do not write any value.
2	DLA		SD_DAT line active. This status bit indicates whether one of the SD_DAT lines is in use. In the case of read transactions (card to host)This bit is set to 1 after the end bit of read command or by activating continue request SD_HCTL[17] CR bit. This bit is cleared to 0 when the host controller received the end bit of the last data block or at the beginning of the read wait mode. In the case of write transactions (host to card)This bit is set to 1 after the end bit of write command or by activating continue request SD_HCTL[17] CR bit. This bit is cleared to 0 on the end of busy event for the last block; host controller must wait 8 clock cycles with line not busy to really consider not "busy state" or after the busy block as a result of a stop at gap request.
		Read 0	SD_DAT line inactive
		Read 1	SD_DAT line active
1	DATI		Command inhibit (SD_DAT). This status bit is generated if either SD_DAT line is active (SD_PSTATE[2] DLA bit) or Read transfer is active (SD_PSTATE[9] RTA bit) or when a command with busy is issued. This bit prevents the local host to issue a command. A change of this bit from 1 to 0 generates a transfer complete interrupt (SD_STAT[1] TC bit).
		Read 0	Issuing of command using the SD_DAT lines is allowed
		Read 1	Issuing of command using SD_DAT lines is not allowed

Table 13-29. Present State Register (SD_PSTATE) Field Descriptions (continued)

Bit	Field	Value	Description
0	CMDI		Command inhibit(SD_CMD). This status bit indicates that the SD_CMD line is in use. This bit is cleared to 0 when the most significant byte is written into the command register. This bit is not set when Auto CMD12 is transmitted. This bit is cleared to 0 in either the following cases: • After the end bit of the command response, excepted if there is a command conflict error (SD_STAT[17] CCRC bit or SD_STAT[18] CEB bit set to 1) or a Auto CMD12 is not executed (SD_AC12[0] ACNE bit). • After the end bit of the command without response (SD_CMD[17:16] RSP_TYPE bits set to "00"). In case of a command data error is detected (SD_STAT[19] CTO bit set to 1), this register is not automatically cleared.
		Read 0	Issuing of command using SD_CMD line is allowed
		Read 1	Issuing of command using SD_CMD line is not allowed

13.4.20 Control Register (SD_HCTL)

This register defines the host controls to set power, wake-up, and transfer parameters.

- SD_HCTL[31:24] = Wake-up control
- SD_HCTL[23:16] = Block gap control
- SD_HCTL[15:8] = Power control
- SD_HCTL[7:0] = Host control

The control register (SD_HCTL) is shown in Figure 13-48 and described in Table 13-30.

Figure 13-48. Control Register (SD_HCTL)

31	28	27	26	25	24	23	20	19	18	17	16
Reserved		OBWE	REM	INS	IWE	Reserved		IBG	RWC	CR	SBGR
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R-0		R/W-0	R/W-0	R/W-0	R/W-0

15	12	11	9	8	7	6	5	4	3	2	1	0
Reserved		SDVS		SDBP	CDSS	CDTL	Rsvd	DMAS		HSPE	DTW	Rsvd
R-0		R/W-0		R/W-0	R/W-0	R/W-0	R-0	R/W-2h		R/W-0	R/W-0	R-0

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-30. Control Register (SD_HCTL) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved bit field. Do not write any value.
27	OBWE		Wake-up event enable for 'out-of-band' Interrupt. This bit enables wake-up events for 'out-of-band' assertion. Wake-up is generated if the wake-up feature is enabled (SD_SYSCONFIG[2] ENAWAKEUP bit). The write to this register is ignored when SD_CON[14] OBIE bit is not set.
		0	Disable wake-up on 'out-of-band' Interrupt
		1	Enable wake-up on 'out-of-band' Interrupt
26	REM		Wake-up event enable on SD card removal. This bit enables wake-up events for card removal assertion. Wake-up is generated if the wake-up feature is enabled (SD_SYSCONFIG[2] ENAWAKEUP bit).
		0	Disable wake-up on card removal
		1	Enable wake-up on card removal
25	INS		Wake-up event enable on SD card insertion This bit enables wake-up events for card insertion assertion. Wake-up is generated if the wake-up feature is enabled (SD_SYSCONFIG[2] ENAWAKEUP bit).
		0	Disable wake-up on card insertion
		1	Enable wake-up on card insertion
24	IWE		Wake-up event enable on SD card interrupt. This bit enables wake-up events for card interrupt assertion. Wake-up is generated if the wake-up feature is enabled (SD_SYSCONFIG[2] ENAWAKEUP bit) and enable status bit is set (SD_IE[8] CIRQ_ENABLE bit).
		0	Disable wake-up on card interrupt
		1	Enable wake-up on card interrupt
23-20	Reserved	0	Reserved bit field. Do not write any value.
19	IBG		Interrupt block at gap. This bit is valid only in 4-bit mode of SDIO card to enable interrupt detection in the interrupt cycle at block gap for a multiple block transfer. For SD card this bit should be cleared to 0.
		0	Disable interrupt detection at the block gap in 4-bit mode
		1	Enable interrupt detection at the block gap in 4-bit mode
18	RWC		Read wait control. The read wait function is optional only for SDIO cards. If the card supports read wait, this bit must be enabled, then requesting a stop at block gap (SD_HCTL[16] SBGR bit) generates a read wait period after the current end of block. Be careful, if read wait is not supported it may cause a conflict on SD_DAT line.
		0	Disable read wait control. Suspend/resume cannot be supported.
		1	Enable read wait control

Table 13-30. Control Register (SD_HCTL) Field Descriptions (continued)

Bit	Field	Value	Description
17	CR		Continue request. This bit is used to restart a transaction that was stopped by requesting a stop at block gap (SD_HCTL[16] SBGR bit). Set this bit to 1 restarts the transfer. The bit is automatically cleared to 0 by the host controller when transfer has restarted, that is, SD_DAT line is active (SD_PSTATE[2] DLA bit) or transferring data (SD_PSTATE[8] WTA bit). The Stop at block gap request must be disabled (SD_HCTL[16] SBGR bit =0) before setting this bit.
		0	No affect
		1	Transfer restart
16	SBGR		Stop at block gap request. This bit is used to stop executing a transaction at the next block gap. The transfer can restart with a continue request (SD_HCTL[17] CR bit) or during a suspend/resume sequence. In case of read transfer, the card must support read wait control. In case of write transfer, the host driver shall set this bit after all block data written. Until the transfer completion (SD_STAT[1] TC bit set to 1), the host driver shall leave this bit set to 1.If this bit is set, the local host shall not write to the data register (SD_DATA).
		0	Transfer mode
		1	Stop at block gap
15-12	Reserved	0	Reserved bit field. Do not write any value.
11-9	SDVS		SD bus voltage select (All cards). The host driver should set these bits to select the voltage level for the card according to the voltage supported by the system (SD_CAPA[26] VS18 bit, SD_CAPA[25] VS30 bit, SD_CAPA[24] VS33 bit) before starting a transfer.
		0-6h	Reserved
		7h	3.3 V (Typical)
8	SDBP		SD bus power. Before setting this bit, the host driver shall select the SD bus voltage (SD_HCTL[11:9] SDVS bits). If the host controller detects the No card state, this bit is automatically cleared to 0. If the module is power off, a write in the command register (SD_CMD) will not start the transfer. A write to this bit has no effect if the selected SD bus voltage is not supported according to capability register (SD_CAPA[VS*]).
		0	Power off
		1	Power on
7	CDSS		Card Detect Signal Selection. This bit selects source for the card detection. When the source for the card detection is switched, the interrupt should be disabled during the switching period by clearing the Interrupt Status/Signal Enable register in order to mask unexpected interrupt being caused by the glitch. The Interrupt Status/Signal Enable should be disabled during over the period of debouncing.
		0	SDCD# is selected (for normal use).
		1	The Card Detect Test Level is selected (for test purposes).
6	CDTL		Card Detect Test Level. This bit is enabled while the Card Detect Signal Selection is set to 1 and it indicates card inserted or not.
		0	No card
		1	Card inserted.
5	Reserved	0	Reserved bit field. Do not write any value.
4-3	DMAS		DMA Select. One of the supported DMA modes can be selected. The host driver shall check support of DMA modes by referencing the Capabilities register. Use of selected DMA is determined by DMA Enable of the Transfer Mode register. This register is only meaningful when MADMA_EN is set to 1. When MADMA_EN is cleared to 0 the bit field is read only and returned value is 0.
		0-1h	Reserved
		2h	32-bit Address ADMA2 is selected.
		3h	Reserved
2	HSPE		High Speed Enable. Before setting this bit, the Host Driver shall check the High Speed Support in the Capabilities register. If this bit is cleared to 0 (default), the Host Controller outputs CMD line and DAT lines at the falling edge of the SD Clock. If this bit is set to 1, the Host Controller outputs CMD line and DAT lines at the rising edge of the SD Clock. This bit shall not be set when dual data rate mode is activated in SD_CON[DDR].
		0	Normal speed mode
		1	High speed mode

Table 13-30. Control Register (SD_HCTL) Field Descriptions (continued)

Bit	Field	Value	Description
1	DTW		Data transfer width. This bit must be set following a valid SET_BUS_WIDTH command (ACMD6) with the value written in bit 1 of the argument. Prior to this command, the SD card configuration register (SCR) must be verified for the supported bus width by the SD card.
		0	1-bit Data width (SD_DAT0 used)
		1	4-bit Data width (SD_DAT[3:0] used)
0	Reserved	0	Reserved bit field. Do not write any value.

13.4.21 SD System Control Register (SD_SYSCTL)

This register defines the system controls to set software resets, clock frequency management and data timeout.

- SD_SYSCTL[31:24] = Software resets
- SD_SYSCTL[23:16] = Timeout control
- SD_SYSCTL[15:0] = Clock control

The SD system control register (SD_SYSCTL) is shown in Figure 13-49 and described in Table 13-31.

Figure 13-49. SD System Control Register (SD_SYSCTL)

31	27	26	25	24	23	20	19	16
Reserved		SRD	SRC	SRA	Reserved		DTO	
R-0		R/W-0	R/W-0	R/W-0	R-0		R/W-0	

15	6	5	3	2	1	0
CLKD		Reserved		CEN	ICS	ICE
R/W-0		R-0		R/W-0	R-0	R/W-0

LEGENDR/W = Read/Write; R = Read only; *-n* = value after reset

Table 13-31. SD System Control Register (SD_SYSCTL) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved bit field. Do not write any value.
26	SRD		Software reset for SD_DAT line. This bit is set to 1 for reset and released to 0 when completed SD_DAT finite state machine in both clock domain are also reset. These registers are cleared by the SD_SYSCTL[26] SRD bit: • SD_DATA • SD_PSTATEBRE, BWE, RTA, WTA, DLA and DATI • SD_HCTLSBGR and CR • SD_STATBRR, BWR, BGE and TC Interconnect is reinitialized. **Note**If a soft reset is issued when an interrupt is asserted, data may be lost.
		0	Reset completed
		1	Software reset for SD_DAT line
25	SRC		Software reset for SD_CMD line. This bit is set to 1 for reset and released to 0 when completed. SD_CMD finite state machine in both clock domain are also reset. These registers are cleared by the SD_SYSCTL[25] SRC bit: • SD_PSTATECMDI • SD_STATCC Interconnect is reinitialized. **Note**If a soft reset is issued when an interrupt is asserted, data may be lost.
		0	Reset completed
		1	Software reset for SD_CMD line
24	SRA		Software reset for all. This bit is set to 1 for reset , and released to 0 when completed. This reset affects the entire host controller except for the card detection circuit and capabilities registers.
		0	Reset completed
		1	Software reset for all the design
23-20	Reserved	0	Reserved bit field. Do not write any value.

Table 13-31. SD System Control Register (SD_SYSCTL) Field Descriptions (continued)

Bit	Field	Value	Description
19-16	DTO		Data timeout counter value and busy timeout. This value determines the interval by which SD_DAT lines timeouts are detected. The host driver needs to set this bit field based on: • The maximum read access time (NAC) (Refer to the SD Specification Part1 Physical Layer) • The data read access time values (TAAC and NSAC) in the card specific data register (CSD) of the card • The timeout clock base frequency (SD_CAPA[5:0] TCF bits) If the card does not respond within the specified number of cycles, a data timeout error occurs (SD_STAT[20] DTO bit). The SD_SYSCTL[19,16] DTO bit field is also used to check busy duration, to generate busy timeout for commands with busy response or for busy programming during a write command. Timeout on CRC status is generated if no CRC token is present after a block write.
		0	TCF x 2^13
		1h	TCF x 2^14
		Eh	TCF x 2^27
		Fh	Reserved
15-6	CLKD		Clock frequency select. These bits define the ratio between a reference clock frequency (system dependant) and the output clock frequency on the SD_CLK pin of either the memory card (SD or SDIO).
		0	Clock Ref bypass
		1h	Clock Ref bypass
		2h	Clock Ref / 2
		3h	Clock Ref / 3
		3FFh	Clock Ref / 1023
5-3	Reserved	0	Reserved bit field. Do not write any value.
2	CEN		Clock enable. This bit controls if the clock is provided to the card or not.
		0	The clock is not provided to the card . Clock frequency can be changed .
		1	The clock is provided to the card and can be automatically gated when SD_SYSCONFIG[0] AUTOIDLE bit is set to 1 (default value) . The host driver shall wait to set this bit to 1 until the Internal clock is stable (SD_SYSCTL[1] ICS bit).
1	ICS		Internal clock stable (status)This bit indicates either the internal clock is stable or not.
		0	The internal clock is not stable.
		1	The internal clock is stable after enabling the clock (SD_SYSCTL[0] ICE bit) or after changing the clock ratio (SD_SYSCTL[15:6] CLKD bits).
0	ICE		Internal clock enable. This register controls the internal clock activity. In very low power state, the internal clock is stopped. NoteThe activity of the debounce clock (used for wake-up events) and the interface clock (used for reads and writes to the module register map) are not affected by this register.
		0	The internal clock is stopped (very low power state).
		1	The internal clock oscillates and can be automatically gated when SD_SYSCONFIG[0] AUTOIDLE bit is set to 1 (default value).

13.4.22 Interrupt Status Register (SD_STAT)

The interrupt status regroups all the status of the module internal events that can generate an interrupt.

- SD_STAT[31:16] = Error Interrupt Status
- SD_STAT[15:0] = Normal Interrupt Status

The error bits are located in the upper 16 bits of the SD_STAT register. All bits are cleared by writing a 1 to them. Additionally, bits 15 and 8 serve as special error bits. These cannot be cleared by writing a 1 to them. Bit 15 (ERRI) is automatically cleared when the error causing to ERRI to be set is handled. (that is, when bits 31:16 are cleared, bit 15 will be automatically cleared). Bit 8 (CIRQ) is cleared by writing a 0 to SD_IE[8] (masking the interrupt) and servicing the interrupt.

The interrupt status register (SD_STAT) is shown in Figure 13-50 and described in Table 13-32.

Figure 13-50. Interrupt Status Register (SD_STAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		BADA	CERR	Reserved		ADMAE	ACE	Rsvd	DEB	DCRC	DTO	CIE	CEB	CCRC	CTO
R-0		R/W-0	R/W-0	R-0		R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	10	9	8	7	6	5	4	3	2	1	0
ERRI	Reserved	BSR	OBI	CIRQ	CREM	CINS	BRR	BWR	DMA	BGE	TC	CC
R-0	R-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-32. Interrupt Status Register (SD_STAT) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved bit field. Do not write any value
29	BADA		Bad access to data space. This bit is set automatically to indicate a bad access to buffer when not allowed: • During a read access to the data register (SD_DATA) while buffer reads are not allowed (SD_PSTATE[11] BRE bit =0) • During a write access to the data register (SD_DATA) while buffer writes are not allowed (SD_PSTATE[10] BWE bit=0)
		Read 0	No interrupt
		Write 0	Status bit unchanged
		Read 1	Bad access
		Write 1	Status is cleared.
28	CERR		Card error. This bit is set automatically when there is at least one error in a response of type R1, R1b, R6, R5 or R5b. Only bits referenced as type E (error) in status field in the response can set a card status error. An error bit in the response is flagged only if corresponding bit in card status response error SD_CSRE in set. There is no card error detection for autoCMD12 command. The host driver shall read SD_RSP76 register to detect error bits in the command response.
		Read 0	No error
		Write 0	Status bit unchanged
		Read 1	Card error
		Write 1	Status is cleared.
27-26	Reserved	0	Reserved bit field. Do not write any value

Table 13-32. Interrupt Status Register (SD_STAT) Field Descriptions (continued)

Bit	Field	Value	Description
25	ADMAE		ADMA Error. This bit is set when the Host Controller detects errors during ADMA based data transfer. The state of the ADMA at an error occurrence is saved in the ADMA Error Status Register. In addition, the Host Controller generates this interrupt when it detects invalid descriptor data (Valid=0) at the ST_FDS state. ADMA Error State in the ADMA Error Status indicates that an error occurs in ST_FDS state. The Host Driver may find that Valid bit is not set at the error descriptor.
		Read 0	No interrupt
		Write 0	Status bit unchanged
		Read 1	ADMA error
		Write 1	Status is cleared.
24	ACE		Auto CMD12 error. This bit is set automatically when one of the bits in Auto CMD12 Error status register has changed from 0 to 1.
		Read 0	No error
		Write 0	Status bit unchanged
		Read 1	AutoCMD12 error
		Write 1	Status is cleared.
23	Reserved	0	Reserved bit field. Do not write any value
22	DEB		Data End Bit error. This bit is set automatically when detecting a 0 at the end bit position of read data on SD_DAT line or at the end position of the CRC status in write mode.
		Read 0	No error
		Write 0	Status bit unchanged
		Read 1	Data end bit error
		Write 1	Status is cleared.
21	DCRC		Data CRC Error. This bit is set automatically when there is a CRC16 error in the data phase response following a block read command or if there is a 3-bit CRC status different of a position "010" token during a block write command.
		Read 0	No error
		Write 0	Status bit unchanged
		Read 1	Data CRC error
		Write 1	Status is cleared.
20	DTO		Data timeout error. This bit is set automatically according to the following conditions: • Busy timeout for R1b, R5b response type • Busy timeout after write CRC status • Write CRC status timeout • Read data timeout
		Read 0	No error
		Write 0	Status bit unchanged
		Read 1	Time out
		Write 1	Status is cleared.
19	CIE		Command index error. This bit is set automatically when response index differs from corresponding command index previously emitted. It depends on the enable bit (SD_CMD[20] CICE).
		Read 0	No error
		Write 0	Status bit unchanged
		Read 1	Command index error
		Write 1	Status is cleared.

Table 13-32. Interrupt Status Register (SD_STAT) Field Descriptions (continued)

Bit	Field	Value	Description
18	CEB		Command end bit error. This bit is set automatically when detecting a 0 at the end bit position of a command response.
		Read 0	No error
		Write 0	Status bit unchanged
		Read 1	Command end bit error
		Write 1	Status is cleared.
17	CCRC		Command CRC error. This bit is set automatically when there is a CRC7 error in the command response depending on the enable bit (SD_CMD[19] CCCE).
		Read 0	No error
		Write 0	Status bit unchanged
		Read 1	Command CRC error
		Write 1	Status is cleared.
16	CTO		Command timeout error. This bit is set automatically when no response is received within 64 clock cycles from the end bit of the command. For commands that reply within 5 clock cycles - the timeout is still detected at 64 clock cycles.
		Read 0	No error
		Write 0	Status bit unchanged
		Read 1	Time Out
		Write 1	Status is cleared.
15	ERRI		Error interrupt. If any of the bits in the Error Interrupt Status register (SD_STAT[31:16]) are set, then this bit is set to 1. Therefore the host driver can efficiently test for an error by checking this bit first. Writes to this bit are ignored.
		Read 0	No interrupt
		Read 1	Error interrupt event(s) occurred
14-11	Reserved	0	Reserved bit field. Do not write any value
10	BSR		Boot Status Received Interrupt. This bit is set automatically when SD_CON[BOOT] is set 1 or 2 and a boot status is received on DAT[0] line.
		Read 0	No interrupt
		Write 0	Status bit unchanged
		Read 1	Boot Status Received Interrupt occurred.
		Write 1	Status is cleared.
9	OBI		Out-of-band interrupt. This bit is set automatically when SD_CON[14] OBIE bit is set and an out-of-band interrupt occurs on OBI pin. The interrupt detection depends on polarity controlled by SD_CON[13] OBIP bit. The out-of-band interrupt signal is a system specific feature for future use, this signal is not required for existing specification implementation.
		Read 0	No out-of-band interrupt
		Write 0	Status bit unchanged
		Read 1	Interrupt out-of-band occurs
		Write 1	Status is cleared.

Table 13-32. Interrupt Status Register (SD_STAT) Field Descriptions (continued)

Bit	Field	Value	Description
8	CIRQ		Card interrupt. In 1-bit mode, interrupt source is asynchronous (can be a source of asynchronous wake-up). In 4-bit mode, interrupt source is sampled during the interrupt cycle. In CE-ATA mode, interrupt source is detected when the card drives SD_CMD line to zero during one cycle after data transmission end. All modes above are fully exclusive. The controller interrupt must be clear by setting SD_IE[8] CIRQ_ENABLE to 0, then the host driver must start the interrupt service with card (clearing card interrupt status) to remove card interrupt source. Otherwise the Controller interrupt will be reasserted as soon as SD_IE[8] CIRQ_ENABLE is set to 1. Writes to this bit are ignored.
		Read 0	No card interrupt
		Read 1	Generate card interrupt
7	CREM		Card Removal. This bit is set automatically when SD_PSTATE[CINS] changes from 1 to 0. A clear of this bit doesn't affect Card inserted present state (SD_PSTATE[CINS]).
		Read 0	Card State stable or debouncing
		Write 0	Status bit unchanged
		Read 1	Card Removed
		Write 1	Status is cleared
6	CINS		Card Insertion. This bit is set automatically when SD_PSTATE[CINS] changes from 0 to 1. A clear of this bit doesn't affect Card inserted present state (SD_PSTATE[CINS]).
		Read 0	Card State stable or debouncing
		Write 0	Status bit unchanged
		Read 1	Card inserted
		Write 1	Status is cleared.
5	BRR		Buffer read ready. This bit is set automatically during a read operation to the card (see class 2 - block oriented read commands) when one block specified by the SD_BLK[10:0] BLEN bit field is completely written in the buffer. It indicates that the memory card has filled out the buffer and that the local host needs to empty the buffer by reading it. NoteIf the DMA receive-mode is enabled, this bit is never set; instead a DMA receive request to the main DMA controller of the system is generated.
		Read 0	Not ready to read buffer
		Write 0	Status bit unchanged
		Read 1	Ready to read buffer
		Write 1	Status is cleared.
4	BWR		Buffer write ready. This bit is set automatically during a write operation to the card (see class 4 - block oriented write command) when the host can write a complete block as specified by SD_BLK[10:0] BLEN. It indicates that the memory card has emptied one block from the buffer and that the local host is able to write one block of data into the buffer. NoteIf the DMA transmit mode is enabled, this bit is never set; instead, a DMA transmit request to the main DMA controller of the system is generated.
		Read 0	Not ready to write buffer
		Write 0	Status bit unchanged
		Read 1	Ready to write buffer
		Write 1	Status is cleared.
3	DMA		DMA Interrupt. This status is set when an interrupt is required in the ADMA instruction and after the data transfer completion.
		Read 0	DMA Interrupt detected
		Write 0	Status bit unchanged
		Read 1	No DMA Interrupt
		Write 1	Status is cleared.

Table 13-32. Interrupt Status Register (SD_STAT) Field Descriptions (continued)

Bit	Field	Value	Description
2	BGE		Block gap event. When a stop at block gap is requested (SD_HCTL[16] SBGR bit), this bit is automatically set when transaction is stopped at the block gap during a read or write operation. This event does not occur when the stop at block gap is requested on the last block. In read mode, a 1-to-0 transition of the SD_DAT line active status (SD_PSTATE[2] DLA bit) between data blocks generates a Block gap event interrupt.
		Read 0	No block gap event
		Write 0	Status bit unchanged
		Read 1	Transaction stopped at block gap
		Write 1	Status is cleared
1	TC		Transfer completed. This bit is always set when a read/write transfer is completed or between two blocks when the transfer is stopped due to a stop at block gap request (SD_HCTL[16] SBGR bit). This bit is also set when exiting a command in a busy state (if the command has a busy notification capability). In Read modeThis bit is automatically set on completion of a read transfer (SD_PSTATE[9] RTA bit). In write modeThis bit is set automatically on completion of the SD_DAT line use (SD_PSTATE[2] DLA bit).
		Read 0	No transfer complete
		Write 0	Status bit unchanged
		Read 1	Data transfer complete
		Write 1	Status is cleared
0	CC		Command complete. This bit is set when a 1-to-0 transition occurs in the register command inhibit (SD_PSTATE[0] CMDI bit) If the command is a type for which no response is expected, then the command complete interrupt is generated at the end of the command. A command timeout error (SD_STAT[16] CTO bit) has higher priority than command complete (SD_STAT[0] CC bit). If a response is expected but none is received, then a command timeout error is detected and signaled instead of the command complete interrupt.
		Read 0	No command complete
		Write 0	Status bit unchanged
		Read 1	Command complete
		Write 1	Status is cleared

13.4.23 Interrupt SD Enable Register (SD_IE)

This register allows to enable/disable the module to set status bits, on an event-by-event basis.

- SD_IE[31:16] = Error Interrupt Status Enable
- SD_IE[15:0] = Normal Interrupt Status Enable

The interrupt SD enable register (SD_IE) is shown in Figure 13-51 and described in Table 13-33.

Figure 13-51. Interrupt SD Enable Register (SD_IE)

31	30	29	28	27	26	25	24
Reserved		BADA_ENABLE	CERR_ENABLE	Reserved		ADMA_ENABLE	ACE_ENABLE
R-0		R/W-0	R/W-0	R-0		R/W-0	R/W-0
23	22	21	20	19	18	17	16
Reserved	DEB_ENABLE	DCRC_ENABLE	DTO_ENABLE	CIE_ENABLE	CEB_ENABLE	CCRC_ENABLE	CTO_ENABLE
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14			11	10	9	8
NULL	Reserved				BSR_ENABLE	OBI_ENABLE	CIRQ_ENABLE
R-0	R-0				R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
CREM_ENABLE	CINS_ENABLE	BRR_ENABLE	BWR_ENABLE	DMA_ENABLE	BGE_ENABLE	TC_ENABLE	CC_ENABLE
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGENDR/W = Read/Write; R = Read only; *-n* = value after reset

Table 13-33. Interrupt SD Enable Register (SD_IE) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved bit field. Do not write any value.
29	BADA_ENABLE		Bad access to data space interrupt enable
		0	Masked
		1	Enabled
28	CERR_ENABLE		Card error interrupt enable
		0	Masked
		1	Enabled
27-26	Reserved	0	Reserved bit field. Do not write any value.
25	ADMA_ENABLE		ADMA error Interrupt Enable
		0	Masked
		1	Enabled
24	ACE_ENABLE		Auto CMD12 error interrupt enable
		0	Masked
		1	Enabled
23	Reserved	0	Reserved bit field. Do not write any value.
22	DEB_ENABLE		Data end bit error interrupt enable
		0	Masked
		1	Enabled
21	DCRC_ENABLE		Data CRC error interrupt enable
		0	Masked
		1	Enabled
20	DTO_ENABLE		Data timeout error interrupt enable
		0	Masked
		1	Enabled

Table 13-33. Interrupt SD Enable Register (SD_IE) Field Descriptions (continued)

Bit	Field	Value	Description
19	CIE_ENABLE		Command index error interrupt enable
		0	Masked
		1	Enabled
18	CEB_ENABLE		Command end bit error interrupt enable
		0	Masked
		1	Enabled
17	CCRC_ENABLE		Command CRC error interrupt enable
		0	Masked
		1	Enabled
16	CTO_ENABLE		Command timeout error interrupt enable
		0	Masked
		1	Enabled
15	NULL	0	Fixed to 0. The host driver shall control error interrupts using the Error Interrupt Signal Enable register. Writes to this bit are ignored.
14-11	Reserved	0	Reserved bit field. Do not write any value.
10	BSR_ENABLE		Boot Status Interrupt Enable A write to this register when SD_CON[BOOT] is cleared to 0 is ignored.
		0	Masked
		1	Enabled
9	OBI_ENABLE		Out-of-band interrupt enable A write to this register when SD_CON[14] OBIE is cleared to 0 is ignored.
		0	Masked
		1	Enabled
8	CIRQ_ENABLE		Card interrupt enable. A clear of this bit also clears the corresponding status bit. During 1-bit mode, if the interrupt routine does not remove the source of a card interrupt in the SDIO card, the status bit is reasserted when this bit is set to 1. This bit must be set to 1 when entering in smart idle mode to enable system to identity wake-up event and to allow controller to clear internal wake-up source.
		0	Masked
		1	Enabled
7	CREM_ENABLE		Card Removal interrupt Enable This bit must be set to 1 when entering in smart idle mode to enable system to identity wake-up event and to allow controller to clear internal wake-up source.
		0	Masked
		1	Enabled
6	CINS_ENABLE		Card Insertion interrupt Enable This bit must be set to 1 when entering in smart idle mode to enable system to identity wake-up event and to allow controller to clear internal wake-up source.
		0	Masked
		1	Enabled
5	BRR_ENABLE		Buffer read ready interrupt enable
		0	Masked
		1	Enabled
4	BWR_ENABLE		Buffer write ready interrupt enable
		0	Masked
		1	Enabled
3	DMA_ENABLE		DMA interrupt enable
		0	Masked
		1	Enable
2	BGE_ENABLE		Block gap event interrupt enable
		0	Masked
		1	Enabled

Table 13-33. Interrupt SD Enable Register (SD_IE) Field Descriptions (continued)

Bit	Field	Value	Description
1	TC_ENABLE		Transfer completed interrupt enable
		0	Masked
		1	Enabled
0	CC_ENABLE		Command completed interrupt enable
		0	Masked
		1	Enabled

13.4.24 Interrupt Signal Enable Register (SD_ISE)

This register allows you to enable/disable the module to set status bits, on an event-by-event basis.

- SD_ISE[31:16] = Error Interrupt Signal Enable
- SD_ISE[15:0] = Normal Interrupt Signal Enable

The Interrupt Signal Enable Register (SD_ISE) is shown in Figure 13-52 and described in Table 13-34.

Figure 13-52. Interrupt Signal Enable Register (SD_ISE)

31	30	29	28	27	26	25	24
Reserved		BADA_SIGEN	CERR_SIGEN	Reserved		ADMA_SIGEN	ACE_SIGEN
R-0		R/W-0	R/W-0	R-0		R/W-0	R/W-0
23	22	21	20	19	18	17	16
Reserved	DEB_SIGEN	DCRC_SIGEN	DTO_SIGEN	CIE_SIGEN	CEB_SIGEN	CCRC_SIGEN	CTO_SIGEN
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14			11	10	9	8
NULL	Reserved				BSR_SIGEN	OBI_SIGEN	CIRQ_SIGEN
R-0	R-0				R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
CREM_SIGEN	CINS_SIGEN	BRR_SIGEN	BWR_SIGEN	DMA_SIGEN	BGE_SIGEN	TC_SIGEN	CC_SIGEN
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGENDR/W = Read/Write; R = Read only; *-n* = value after reset

Table 13-34. Interrupt Signal Enable Register (SD_ISE) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved bit field. Do not write any value.
29	BADA_SIGEN		Bad access to data space interrupt enable
		0	Masked
		1	Enabled
28	CERR_SIGEN		Card error interrupt signal status enable
		0	Masked
		1	Enabled
27-26	Reserved	0	Reserved bit field. Do not write any value.
25	ADMA_SIGEN		ADMA error signal status enable
		0	Masked
		1	Enabled
24	ACE_SIGEN		Auto CMD12 error signal status enable
		0	Masked
		1	Enabled
23	Reserved	0	Reserved bit field. Do not write any value.
22	DEB_SIGEN		Data end bit error signal status enable
		0	Masked
		1	Enabled
21	DCRC_SIGEN		Data CRC error signal status enable
		0	Masked
		1	Enabled
20	DTO_SIGEN		Data timeout error signal status enable
		0	Masked The host controller provides the clock to the card until the card sends the data or the transfer is aborted.
		1	Enabled

Table 13-34. Interrupt Signal Enable Register (SD_ISE) Field Descriptions (continued)

Bit	Field	Value	Description
19	CIE_SIGEN		Command index error signal status enable
		0	Masked
		1	Enabled
18	CEB_SIGEN		Command end bit error signal status enable
		0	Masked
		1	Enabled
17	CCRC_SIGEN		Command CRC error signal status enable
		0	Masked
		1	Enabled
16	CTO_SIGEN		Command timeout error signal status enable
		0	Masked
		1	Enabled
15	NULL		Fixed to 0. The host driver shall control error interrupts using the error interrupt signal enable register. Writes to this bit are ignored.
14-11	Reserved	0	Reserved bit field. Do not write any value.
10	BSR_SIGEN		Boot Status signal status enable. A write to this register when SD_CON[BOOT] is cleared to 0 is ignored
		0	Masked
		1	Enabled
9	OBI_SIGEN		Out-of-band interrupt signal status enable. A write to this register when SD_CON[14] OBIE is cleared to 0 is ignored.
		0	Masked
		1	Enabled
8	CIRQ_SIGEN		Card interrupt signal status enable. A clear of this bit also clears the corresponding status bit. During 1-bit mode, if the interrupt routine does not remove the source of a card interrupt in the SDIO card, the status bit is reasserted when this bit is set to 1. This bit must be set to 1 when entering in smart idle mode to enable system to identity wake-up event and to allow controller to clear internal wake-up source.
		0	Masked
		1	Enabled
7	CREM_SIGEN		Card Removal signal status enable This bit must be set to 1 when entering in smart idle mode to enable system to identity wake-up event and to allow controller to clear internal wake-up source.
		0	Masked
		1	Enabled
6	CINS_SIGEN		Card Insertion signal status enable. This bit must be set to 1 when entering in smart idle mode to enable system to identity wake-up event and to allow controller to clear internal wake-up source.
		0	Masked
		1	Enabled
5	BRR_SIGEN		Buffer read ready signal status enable
		0	Masked
		1	Enabled
4	BWR_SIGEN		Buffer write ready signal status enable
		0	Masked
		1	Enabled
3	DMA_SIGEN		DMA signal status enable
		0	Masked
		1	Enabled
2	BGE_SIGEN		Block gap event signal status enable
		0	Masked
		1	Enabled

Table 13-34. Interrupt Signal Enable Register (SD_ISE) Field Descriptions (continued)

Bit	Field	Value	Description
1	TC_SIGEN		Transfer completed signal status enable
		0	Masked
		1	Enabled
0	CC_SIGEN		Command completed signal status enable
		0	Masked
		1	Enabled

13.4.25 Auto CMD12 Error Status Register (SD_AC12)

The host driver may determine which of the errors cases related to Auto CMD12 has occurred by checking this SD_AC12 register when an auto CMD12 error interrupt occurs.

This register is valid only when auto CMD12 is enabled (SD_CMD[2] ACEN bit) and auto CMD12Error (SD_STAT[24] ACE bit) is set to 1.

NOTE: These bits are automatically reset when starting a new adtc command with data.

The auto CMD12 error status register (SD_AC12) is shown in Figure 13-53 and described in Table 13-35.

Figure 13-53. Auto CMD12 Error Status Register (SD_AC12)

31	8	7	6	5	4	3	2	1	0
Reserved		CNI	Reserved		ACIE	ACEB	ACCE	ACTO	ACNE
R-0		R-0	R-0		R-0	R-0	R-0	R-0	R-0

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-35. Auto CMD12 Error Status Register (SD_AC12) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved bit field. Do not write any value.
7	CNI		Command not issue by auto CMD12 error. If this bit is set to 1, it means that pending command is not executed due to auto CMD12 error ACEB, ACCE, ACTO, or ACNE.
		Read 0	Not error
		Read 1	Command not issued
6-5	Reserved	0	Reserved bit field. Do not write any value.
4	ACIE		Auto CMD12 index error. This bit is a set to 1 when response index differs from corresponding command auto CMD12 index previously emitted.
			This bit depends on the command index check enable (SD_CMD[20] CICE bit).
		Read 0	No error
		Read 1	Auto CMD12 index error
3	ACEB		Auto CMD12 end bit error. This bit is set to 1 when detecting a 0 at the end bit position of auto CMD12 command response.
		Read 0	No error
		Read 1	AutoCMD12 end bit error
2	ACCE		Auto CMD12 CRC error. This bit is automatically set to 1 when a CRC7 error is detected in the auto CMD12 command response depending on the enable in the SD_CMD[19] CCCE bit.
		Read 0	No error
		Read 1	Auto CMD12 CRC error
1	ACTO		Auto CMD12 timeout error. This bit is set to 1 if no response is received within 64 clock cycles from the end bit of the auto CMD12 command.
		Read 0	No error
		Read 1	Auto CMD12 time out
0	ACNE		Auto CMD12 not executed. This bit is set to 1 if multiple block data transfer command has started and if an error occurs in command before auto CMD12 starts.
		Read 0	Auto CMD12 executed
		Read 1	Auto CMD12 not executed

13.4.26 Capabilities Register (SD_CAPA)

This register lists the capabilities of the SD/SDIO host controller. The capabilities register (SD_CAPA) is shown in Figure 13-54 and described in Table 13-36.

Figure 13-54. Capabilities Register (SD_CAPA)

31	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		64BIT	Rsvd	VS18	VS30	VS33	SRS	DS	HSS	Rsvd	AD2S	Rsvd	MBL	
R-0		R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-1	R-1	R-1	R-0	R-1	R-0	R-1h	

15	14	13	8	7	6	5	0
Reserved		BCF		TCU	Rsvd	TCF	
R-0		R-0		R-1	R-0	R-0	

LEGENDR/W = Read/Write; R = Read only; *-n* = value after reset

Table 13-36. Capabilities Register (SD_CAPA) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved bit field. Do not write any value.
28	64BIT		64 Bit System Bus Support. Setting 1 to this bit indicates that the Host Controller supports 64-bit address descriptor mode and is connected to 64-bit address system bus.
		Read 0	32 bit System bus address
		Read 1	64 bit System bus address
27	Reserved	0	Reserved bit. Any write to this bit results in 0.
26	VS18		Voltage support 1.8 V. Initialization of this register (via a write access to this register) depends on the system capabilities. The host driver shall not modify this register after the initialization. This register is only reinitialized by a hard reset (via RESET signal).
		Read 0	1.8 V not supported
		Write 0	1.8 V not supported
		Read 1	1.8 V supported
		Write 1	1.8 V supported
25	VS30		Voltage support 3.0V. Initialization of this register (via a write access to this register) depends on the system capabilities. The host driver shall not modify this register after the initialization. This register is only reinitialized by a hard reset (via RESET signal)
		Read 0	3.0 V not supported
		Write 0	3.0 V not supported
		Read 1	3.0 V supported
		Write 1	3.0 V supported
24	VS33		Voltage support 3.3V. Initialization of this register (via a write access to this register) depends on the system capabilities. The host driver shall not modify this register after the initialization. This register is only reinitialized by a hard reset (via RESET signal)
		Read 0	3.3 V not supported
		Write 0	3.3 V not supported
		Read 1	3.3 V supported
		Write 1	3.3 V supported
23	SRS		Suspend/resume support (SDIO cards only). This bit indicates whether the host controller supports Suspend/Resume functionality.
		Read 0	The Host controller does not suspend/resume functionality.
		Read 1	The Host controller supports suspend/resume functionality.
22	DS		DMA support. This bit indicates that the Host controller is able to use DMA to transfer data between system memory and the Host controller directly.
		Read 0	DMA not supported
		Read 1	DMA supported

Table 13-36. Capabilities Register (SD_CAPA) Field Descriptions (continued)

Bit	Field	Value	Description
21	HSS		High-speed support. This bit indicates that the host controller supports high speed operations and can supply an up-to-52 MHz clock to the card.
		Read 0	DMA not supported
		Read 1	DMA supported
20	Reserved	0	Reserved bit field. Do not write any value.
19	AD2S		This bit indicates whether the Host Controller is capable of using ADMA2. It depends on setting of generic parameter MADMA_EN.
		Read 0	ADMA2 supported
		Read 1	ADMA2 not supported
18	Reserved	0	Reserved bit field. Do not write any value.
17-16	MBL		Maximum block length. This value indicates the maximum block size that the host driver can read and write to the buffer in the host controller.
			The host controller supports 512 bytes and 1024 bytes block transfers.
		Read 0	512 bytes
		Read 1	1024 bytes
		Read 2	2048 bytes
15-14	Reserved	0	Reserved bit field. Do not write any value.
13-8	BCF		Base clock frequency for clock provided to the card.
		Read 0	The value indicating the base (maximum) frequency for the output clock provided to the card is system dependent and is not available in this register. Get the information via another method. See the *Power, Reset, and Clock Management* chapter for more information on the value of FUNC_96M_CLK clock signal.
7	TCU		Timeout clock unit. This bit shows the unit of base clock frequency used to detect Data Timeout Error (SD_STAT[20] DTO bit).
		Read 0	kHz
		Read 1	MHz
6	Reserved	0	Reserved. This bit is initialized to zero, and writes to it are ignored.
5-0	TCF		Timeout clock frequency. The timeout clock frequency is used to detect Data Timeout Error (SD_STAT[20] DTO bit).
		Read 0	The timeout clock frequency depends on the frequency of the clock provided to the card. The value of the timeout clock frequency is not available in this register.

13.4.27 Maximum Current Capabilities Register (SD_CUR_CAPA)

This register indicates the maximum current capability for each voltage. The value is meaningful if the voltage support is set in the capabilities register (SD_CAPA).

Initialization of this register (via a write access to this register) depends on the system capabilities. The host driver shall not modify this register after the initialization.

This register is only reinitialized by a hard reset (via RESET signal). The maximum current capabilities register (SD_CUR_CAPA) is shown in Figure 13-55 and described in Table 13-37.

Figure 13-55. Maximum Current Capabilities Register (SD_CUR_CAPA)

31 24	23 16	15 8	7 0
Reserved	CUR_1V8	CUR_3V0	CUR_3V3
R-0	R/W-0	R/W-0	R/W-0

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-37. Maximum Current Capabilities Register (SD_CUR_CAPA) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved. This bit is initialized to zero, and writes to it are ignored.
23-16	CUR_1V8		Maximum current for 1.8 V
		Read 0	The maximum current capability for this voltage is not available. Feature not implemented.
15-8	CUR_3V0		Maximum current for 3.0 V
		Read 0	The maximum current capability for this voltage is not available. Feature not implemented.
7-0	CUR_3V3		Maximum current for 3.3 V
		Read 0	The maximum current capability for this voltage is not available. Feature not implemented.

13.4.28 Force Event Register for Error Interrupt Status (SD_FE)

The Force Event register is not a physically implemented register. Rather, it is an address at which the Error Interrupt Status register can be written. The effect of a write to this address will be reflected in the Error Interrupt Status Register, if corresponding bit of the Error Interrupt Status Enable Register is set.

The force event register (SD_FE) is shown in Figure 13-56 and described in Table 13-38.

Figure 13-56. Interrupt Signal Enable Register (SD_ISE)

31	30	29	28	27	26	25	24
Reserved		FE_BADA	FE_CERR	Reserved		FE_ADMAE	FE_ACE
R-0		W-0	W-0	R-0		W-0	W-0

23	22	21	20	19	18	17	16
Reserved	FE_DEB	FE_DCRC	FE_DTO	FE_CIE	FE_CEB	FE_CCRC	FE_CTO
R-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
FE_CNI	Reserved		FE_ACIE	FE_ACEB	FE_ACCE	FE_ACTO	FE_ACNE
W-0	R-0		W-0	W-0	W-0	W-0	W-0

LEGENDR/W = Read/Write; R = Read only; *-n* = value after reset

Table 13-38. Force Event Register (SD_FE) Field Descriptions

Bits	Field	Value	Description
31-30	Reserved	0	Reserved bit field. Any writes to these bits result in 0.
29	FE_BADA		Force Event Bad access to data space.
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
28	FE_CERR		Force Event Card error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
27-26	Reserved	0	Reserved bit field. Any writes to these bits result in 0.
25	FE_ADMAE		Force Event ADMA error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
24	FE_ACE		Force Event Auto CMD12 error.
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
23	Reserved	0	Reserved bit field. Any writes to this bit result in 0.
22	FE_DEB		Force Event Data End Bit error.
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
21	FE_DCRC		Force Event Data CRC error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
20	FE_DTO		Force Event Data timeout error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.

Table 13-38. Force Event Register (SD_FE) Field Descriptions (continued)

Bits	Field	Value	Description
19	FE_CIE		Force Event Command index error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
18	FE_CEB		Force Event Command end bit error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
17	FE_CCRC		Force Event Comemand CRC error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
16	FE_CTO		Force Event Command Timeout error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
15-8	Reserved	0	Reserved bit field. Any writes to these bits result in 0.
7	FE_CNI		Force Event Command not issue by Auto CMD12 error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
6-5	Reserved	0	Reserved bit field. Any writes to these bits result in 0.
4	FE_ACIE		Force Event Auto CMD12 index error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
3	FE_ACEB		Force Event Auto CMD12 end bit error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
2	FE_ACCE		Force Event Auto CMD12 CRC error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
1	FE_ACTO		Force Event Auto CMD12 timeout error
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.
0	FE_ACNE		Force Event Auto CMD12 not executed.
		Write 0	No effect; no interrupt
		Write 1	Interrupt forced.

13.4.29 ADMA Error Status Register (SD_ADMAES)

When an ADMA Error Interrupt has occurred, the ADMA Error States field in this register holds the ADMA state and the ADMA System Address Register holds the address around the error descriptor. For recovering the error, the Host Driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address.
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address.
- ST_CADR: This sate is never set because do not generate ADMA error in this state.
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address.

In the case of a write operation, the Host Driver should use ACMD22 to get the number of written block rather than using this information, since unwritten data may exist in the Host Controller. The Host Controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid = 0) at the ST_FDS state. In this case, ADMA Error State indicates that an error occurs at ST_FDS state. The Host Driver may find that the Valid bit is not set in the error descriptor.

The ADMA error status register (SD_BLK) is shown in Figure 13-57 and described in Table 13-39.

Figure 13-57. ADMA Error Status Register (SD_ADMAES)

31	3	2	1	0
Reserved		LME	AES	
R-0		W-0	R/W-0	

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-39. ADMA Error Status Register (SD_ADMAES) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved bit field. Do not write any value.
2	LME		ADMA Length Mismatch Error: • While Block Count Enable is being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length. • Total data length cannot be divided by the block length.
		0	No error
		1	Error
1-0	AES		ADMA Error State. This field indicates the state of ADMA when an error occurred during an ADMA data transfer. This field never indicates "10" because ADMA never stops in this state.
		Read 0	ST_STOP (Stop DMA). Contents of the SYS_SDR register
		Write 1h	ST_STOP (Stop DMA). Points to the error descriptor.
		Read 2h	Never set this state. (Not used)
		Write 3h	ST_TFR (Transfer Data). Points to the 'next' of the error descriptor.

13.4.30 ADMA System Address Low Bits Register (SD_ADMASAL)

This register holds the byte address of the executing command of the Descriptor table. The 32-bit Address Descriptor uses the lower 32 bits of this register. At the start of ADMA, the Host Driver shall set the start address of the Descriptor table.

The ADMA System address low bits register (SD_ADMASAL) is shown in Figure 13-58 and described in Table 13-40.

Figure 13-58. ADMA System Address Low Bits (SD_ADMASAL)

31	0
ADMA_A32B	
R/W-0	

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-40. ADMA System Address Low Bits (SD_ADMASAL) Field Descriptions

Bit	Field	Value	Description
31-0	ADMA_A32B		The ADMA increments this register address, which points to the next line, whenever fetching a Descriptor line. When the ADMA Error Interrupt is generated, this register holds the valid Descriptor address depending on the ADMA state. The Host Driver shall program the Descriptor Table on a 32-bit boundary and set the 32-bit boundary address to this register. ADMA2 ignores the lower 2 bits of this register and assumes it to be 00b.

13.4.31 ADMA System Address High Bits Register (SD_ADMASAH)

The ADMA System address high bits register (SD_ADMASAH) is shown in Figure 13-58 and described in Table 13-40.

Figure 13-59. ADMA System Address High Bits Register (SD_ADMASAH)

31	0
ADMA_A32B	
R/W-0	

LEGENDR/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-41. ADMA System Address High Bits Register (SD_ADMASAH) Field Descriptions

Bit	Field	Value	Description
31-0	ADMA_A32B		

13.4.32 Versions Register (SD_REV)

This register contains the hard coded RTL vendor revision number, the version number of SD specification compliancy and a slot status bit.

- SD_REV[31:16] = Host Controller Version
- SD_REV[15:0] = Slot Interrupt Status

The versions register (SD_REV) is shown in Figure 13-60 and described in Table 13-42.

Figure 13-60. Versions Register (SD_REV)

31 24	23 16	15 1	0
VREV	SREV	Reserved	SIS
R-x	R-0	R-0	R-0

LEGENDR/W = Read/Write; R = Read only; *-n* = value after reset

Table 13-42. Versions Register (SD_REV) Field Descriptions

Bit	Field	Value	Description
31-24	VREV	0	Vendor Version Number. Bits [7:4] is the major revision, bits [3:0] is the minor revision. Examples: 10h for 1.0 21h for 2.1
23-16	SREV	Read 0	Specification Version Number. This status indicates the Standard SD Host Controller Specification Version. The upper and lower 4-bits indicate the version. SD Host Specification Version 1.0
15-1	Reserved	0	Reserved bit field. Do not write any value.
0	SIS		Slot Interrupt Status. This status bit indicates the inverted state of interrupt signal for the module. By a power on reset or by setting a software reset for all (SD_SYSCTL[24] SRA), the interrupt signal shall be deasserted and this status shall read 0.

Multichannel Audio Serial Port (McASP)

This chapter is a description of the IP of the multichannel audio serial port (McASP).

Topic **Page**

14.1 Introduction

14.1.1 Purpose of the Peripheral

The multichannel audio serial port (McASP) functions as a general-purpose audio serial port optimized for the needs of multichannel audio applications. The McASP is useful for time-division multiplexed (TDM) stream, Inter-Integrated Sound (I2S) protocols, and intercomponent digital audio interface transmission (DIT). The MCASP has the flexibility to gluelessly connect to a Sony / Philips digital interface (S/PDIF) transmit physical layer component. The McASP consists of transmit and receive sections that may operate synchronized, or completely independently with separate master clocks, bit clocks, and frame syncs, and using different transmit modes with different bit-stream formats. The McASP module also includes serializers that can be individually enabled to either transmit or receive.

Although intercomponent digital audio interface reception (DIR) mode (that is, S/PDIF stream receiving) is NOT natively supported by the MCASP module, a specific TDM mode implementation for the MCASP receivers allows an easy connection to external DIR components (for example, S/PDIF to I2S format converters).

14.1.2 Features

Features of the McASP include:

- Two independent clock generator modules for transmit and receive.
  - Clocking flexibility allows the McASP to receive and transmit at different rates. For example, the McASP can receive data at 48 kHz but output up-sampled data at 96 kHz or 192 kHz.
- Independent transmit and receive modules, each includes:
  - Programmable clock and frame sync generator.
  - TDM streams from 2 to 32, and 384 time slots.
  - Support for time slot sizes of 8, 12, 16, 20, 24, 28, and 32 bits.
  - Data formatter for bit manipulation.
- Individually assignable serial data pins.
- Glueless connection to audio analog-to-digital converters (ADC), digital-to-analog converters (DAC), codec, digital audio interface receiver (DIR), and S/PDIF transmit physical layer components.
- Wide variety of I2S and similar bit-stream format.
- Integrated digital audio interface transmitter (DIT) supports (up to 10 transmit pins):
  - S/PDIF, IEC60958-1, AES-3 formats.
  - Enhanced channel status/user data RAM.
- 384-slot TDM with external digital audio interface receiver (DIR) device.
  - For DIR reception, an external DIR receiver integrated circuit should be used with I2S output format and connected to the McASP receive section.
- Extensive error checking and recovery.
  - Transmit underruns and receiver overruns due to the system not meeting real-time requirements.
  - Early or late frame sync in TDM mode.
  - Out-of-range high-frequency master clock for both transmit and receive.
  - External error signal coming into the AMUTEIN input.
  - DMA error due to incorrect programming.
- up to 16 serial data pins depending of the McASP instance in use (see your device-specific data manual) - each serializer having associated one transmit (Tx) and one receive (Rx) channel, which allows support of:
  - 16 data channels, each configurable to either transmit or receive at a time
  - A single 32-bit buffer per serializer for transmit and receive operations
  - One transmit interrupt request (AXINT) and one receive interrupt request (ARINT) linked with each of the 16 serializers

NOTE: Because a serializer receive and transmit channels data is shared on the same MCASP data pin, you can choose to have either Tx or Rx function from a serializer, NOT both at the same time.

14.1.3 Protocols Supported

The McASP supports a wide variety of protocols.

- Transmit section supports:
  - Wide variety of I2S and similar bit-stream formats.
  - TDM streams from 2 to 32 time slots.
  - S/PDIF, IEC60958-1, AES-3 formats.
- Receive section supports:
  - Wide variety of I2S and similar bit-stream formats.
  - TDM streams from 2 to 32 time slots.
  - TDM stream of 384 time slots specifically designed for easy interface to external digital interface receiver (DIR) device transmitting DIR frames to McASP using the I2S protocol (one time slot for each DIR subframe).

The transmit and receive sections may each be individually programmed to support the following options on the basic serial protocol:

- Programmable clock and frame sync polarity (rising or falling edge): ACLKR/X, AHCLKR/X, and AFSR/X.
- Slot length (number of bits per time slot): 8, 12, 16, 20, 24, 28, 32 bits supported.
- Word length (bits per word): 8, 12, 16, 20, 24, 28, 32 bits; always less than or equal to the time slot length.
- First-bit data delay: 0, 1, 2 bit clocks.
- Left/right alignment of word inside slot.
- Bit order: MSB first or LSB first.
- Bit mask/pad/rotate function.
  - Automatically aligns data internally in either Q31 or integer formats.
  - Automatically masks non-significant bits (sets to 0, 1, or extends value of another bit).

In DIT mode for McASP, additional features of the transmitter are:

- Transmit-only mode 384 time slots (subframe) per frame.
- Bi-phase encoded 3.3 V output.
- Support for consumer and professional applications.
- Channel status RAM (384 bits).
- User data RAM (384 bits).
- Separate valid bit (V) for subframe A, B.

In I2S mode, the transmit and receive sections can support simultaneous transfers on up to all serial data pins operating as 192 kHz stereo channels.

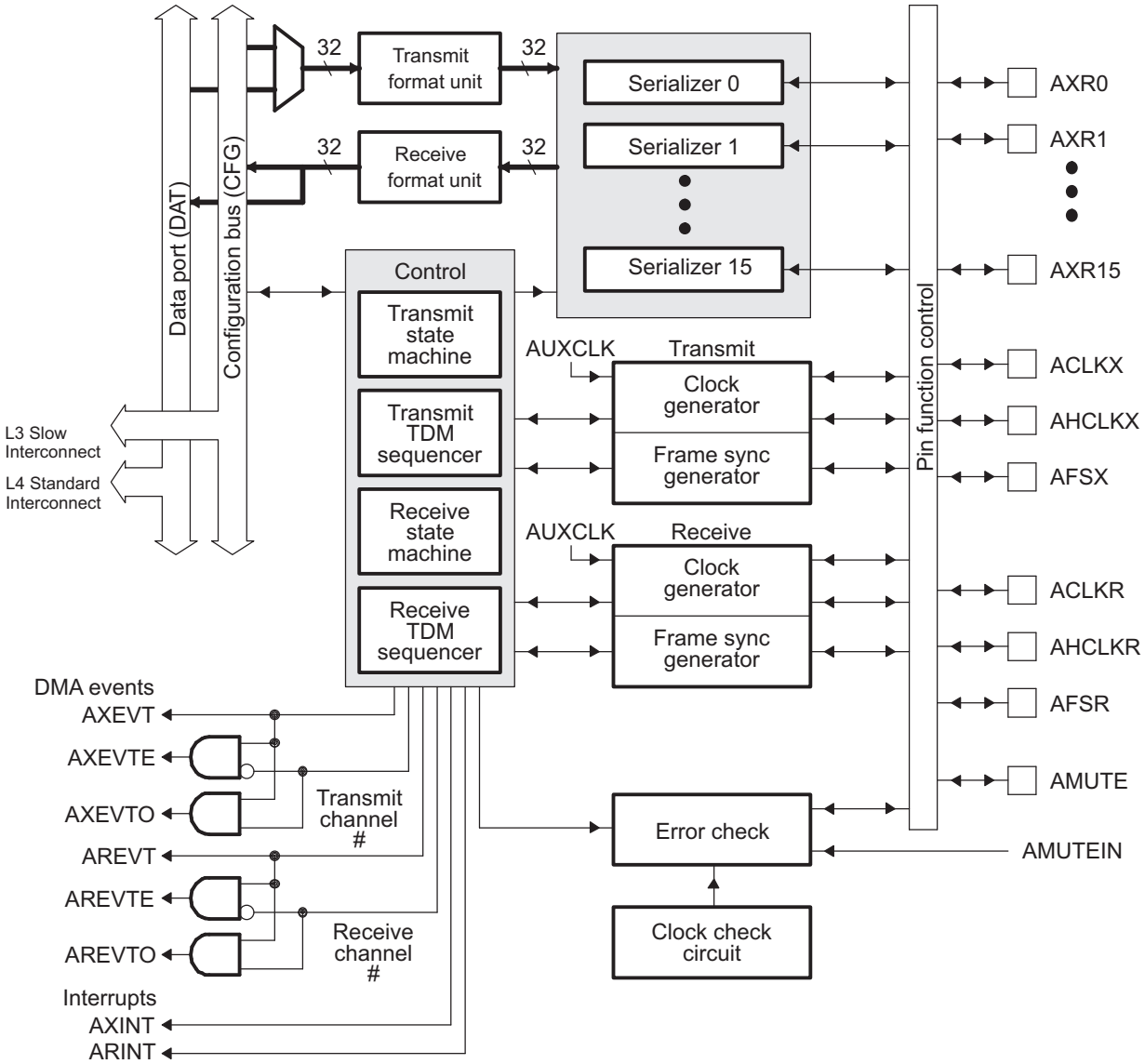
In DIT mode, the transmitter can support a 192 kHz frame rate (stereo) on up to all serial data pins simultaneously (note that the internal bit clock for DIT runs two times faster than the equivalent bit clock for I2S mode, due to the need to generate Biphase Mark Encoded Data).

The MCASP does NOT natively support DIR-mode reception (that is, receiving in the S/PDIF format). To allow this, the MCASP can use a DIR-input to I2S-output converter implemented by an external device (that is, external DIR component). To facilitate reception in this case, the TDM mode of MCASP receivers logic is extended to support a non-standard 384-slot TDM stream.

14.1.4 Functional Block Diagram

Figure 14-1 shows the major blocks of the McASP. The McASP has independent receive/transmit clock generators and frame sync generators.

Figure 14-1. McASP Block Diagram



A McASP has up to 16 serial data pins depending of the McASP instance in use (see your device-specific data manual). When referring to AXRn data pins in this document, n will be up to 15.

B AMUTEIN is not a dedicated McASP pin, but typically comes from one of the external interrupt pins.

14.1.4.1 System Level Connections

Figure 14-2 through Figure 14-6 show examples of McASP usage in digital audio encoder/decoder systems.

Figure 14-2. McASP to Parallel 2-Channel DACs

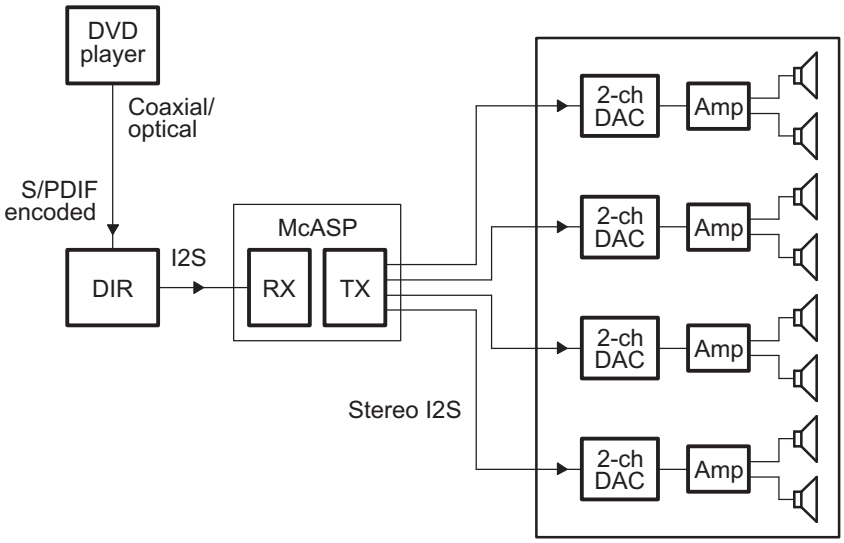


Figure 14-3. McASP to 6-Channel DAC and 2-Channel DAC

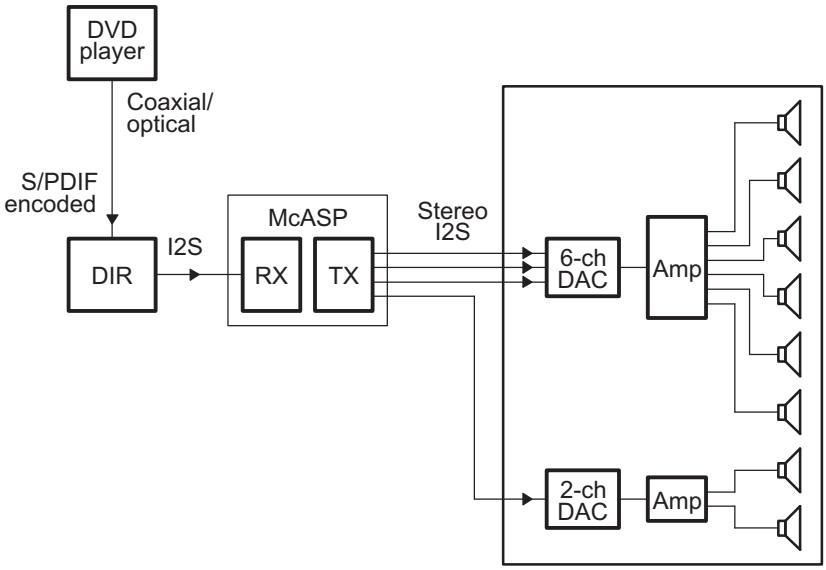


Figure 14-4. McASP to Digital Amplifier

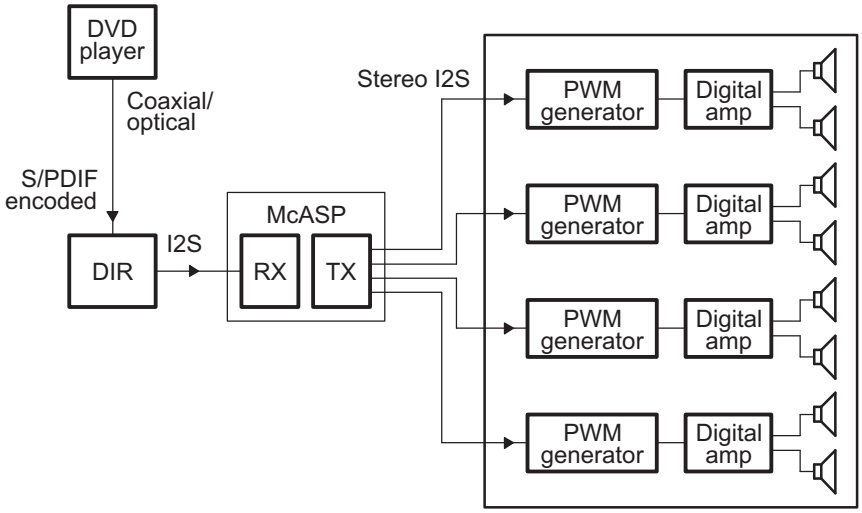


Figure 14-5. McASP as Digital Audio Encoder

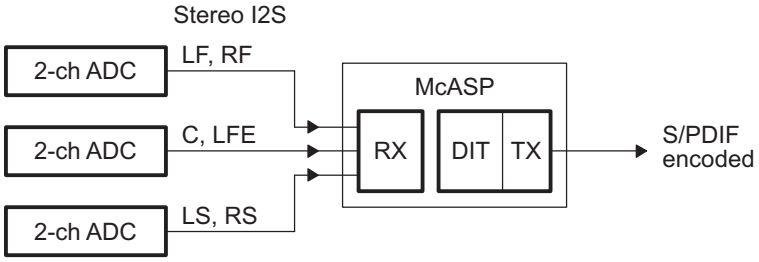
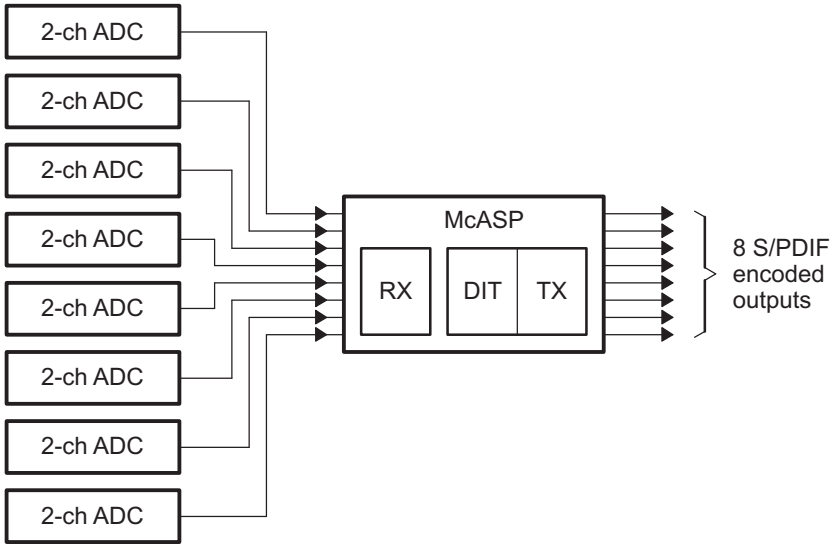


Figure 14-6. McASP as 16 Channel Digital Processor



14.1.5 Supported Use Case Statement

14.1.6 Industry Standard Compliance Statement

The McASP supports the following industry standard interfaces.

14.1.6.1 TDM Format

The McASP transmitter and receiver support the multichannel, synchronous time-division-multiplexed (TDM) format via the TDM transfer mode. Within this transfer mode, a wide variety of serial data formats are supported, including formats compatible with devices using the Inter-Integrated Sound (I2S) protocol. This section briefly discusses the TDM format and the I2S protocol.

14.1.6.1.1 TDM Format

The TDM format is typically used when communicating between integrated circuit devices on the same printed circuit board or on another printed circuit board within the same piece of equipment. For example, the TDM format is used to transfer data between the processor and one or more analog-to-digital converter (ADC), digital-to-analog converter (DAC), or S/PDIF receiver (DIR) devices.

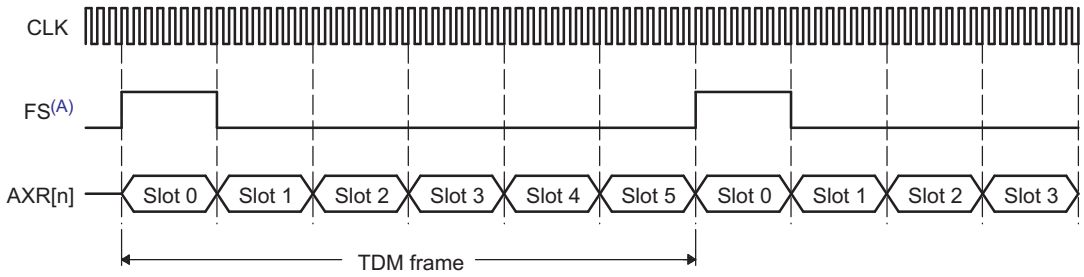
The TDM format consists of three components in a basic synchronous serial transfer: clock (CLK), data (AXRn), and the frame sync (FS). In a TDM transfer, all data bits (AXRn) are synchronous to the serial clock (ACLKX or ACLKR). The data bits are grouped into words and slots (as defined in Section 14.1.7). The "slots" are also commonly referred to as "time slots" or "channels" in TDM terminology. A frame consists of multiple slots (or channels). Each TDM frame is defined by the frame sync signal (AFSX or AFSR). Data transfer is continuous and periodic, since the TDM format is most commonly used to communicate with data converters that operate at a fixed sample rate.

There are no delays between slots. The last bit of slot N is followed immediately on the next serial clock cycle with the first bit of slot N + 1, and the last bit of the last slot is followed immediately on the next serial clock with the first bit of the first slot. However, the frame sync may be offset from the first bit of the first slot with a 0, 1, or 2-cycle delay.

It is required that the transmitter and receiver in the system agree on the number of bits per slot, since the determination of a slot boundary is not made by the frame sync signal (although the frame sync marks the beginning of slot 0 and the beginning of a new frame).

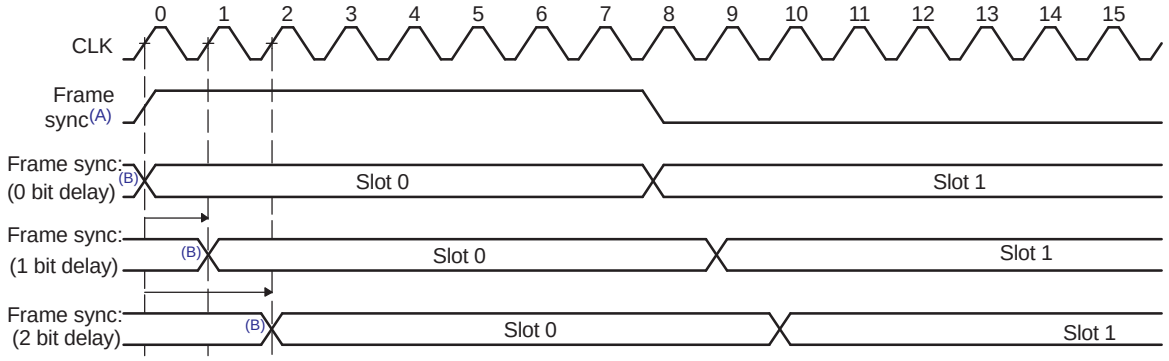
Figure 14-7 shows the TDM format. Figure 14-8 shows the different bit delays from the frame sync.

Figure 14-7. TDM Format–6 Channel TDM Example



A FS duration of slot is shown. FS duration of single bit is also supported.

Figure 14-8. TDM Format Bit Delays from Frame Sync



A FS duration of slot is shown. FS duration of single bit is also supported.

B Last bit of last slot of previous frame. No gap between this bit and the first bit of slot 0 is allowed.

In a typical audio system, one frame of data is transferred during each data converter sample period f_s. To support multiple channels, the choices are to either include more time slots per frame (thus operating with a higher bit clock rate), or to use additional data pins to transfer the same number of channels (thus operating with a slower bit clock rate).

For example, a particular six channel DAC may be designed to transfer over a single serial data pin AXRn as shown in Figure 14-7. In this case the serial clock must run fast enough to transfer a total of 6 channels within each frame period. Alternatively, a similar six channel DAC may be designed to use three serial data pins AXR[0,1,2], transferring two channels of data on each pin during each sample period. In the latter case, if the sample period remains the same, the serial clock can run three times slower than the former case. The McASP is flexible enough to support either type of DAC.

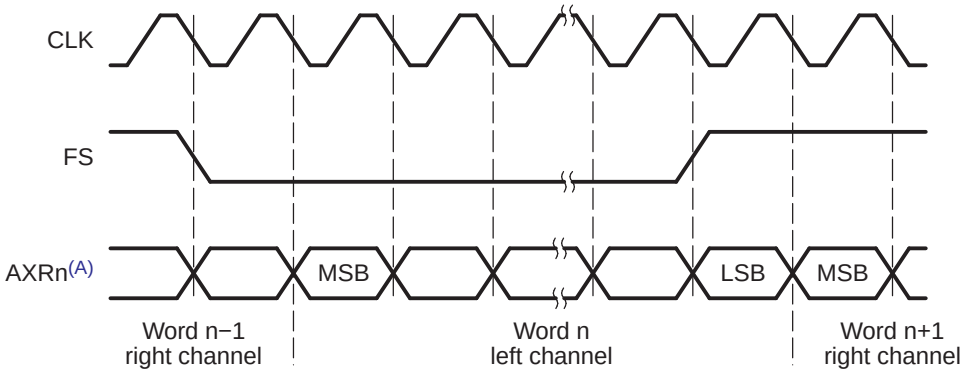
14.1.6.1.2 Inter-Integrated Sound (I2S) Format

The inter-integrated sound (I2S) format is used extensively in audio interfaces. The TDM transfer mode of the McASP supports the I2S format when configured to 2 slots per frame.

I2S format is specifically designed to transfer a stereo channel (left and right) over a single data pin AXRn. "Slots" are also commonly referred to as "channels". The frame width duration in the I2S format is the same as the slot size. The frame signal is also referred to as "word select" in the I2S format. Figure 14-9 shows the I2S protocol.

The McASP supports transfer of multiple stereo channels over multiple AXRn pins.

Figure 14-9. Inter-Integrated Sound (I2S) Format



A 1 to 16 data pins may be supported.

14.1.6.2 S/PDIF Coding Format

The McASP transmitter supports the S/PDIF format with 3.3V biphase-mark encoded output. The S/PDIF format is supported by the digital audio interface transmit (DIT) transfer mode of the McASP. This section briefly discusses the S/PDIF coding format.

14.1.6.2.1 Biphase-Mark Code (BMC)

In S/PDIF format, the digital signal is coded using the biphase-mark code (BMC). The clock, frame, and data are embedded in only one signal—the data pin AXRn. In the BMC system, each data bit is encoded into two logical states (00, 01, 10, or 11) at the pin. These two logical states form a cell. The duration of the cell, which equals to the duration of the data bit, is called a time interval. A logical 1 is represented by two transitions of the signal within a time interval, which corresponds to a cell with logical states 01 or 10. A logical 0 is represented by one transition within a time interval, which corresponds to a cell with logical states 00 or 11. In addition, the logical level at the start of a cell is inverted from the level at the end of the previous cell. Figure 14-10 and Table 14-1 show how data is encoded to the BMC format.

As shown in Figure 14-10, the frequency of the clock is twice the unencoded data bit rate. In addition, the clock is always programmed to 128 × f_s, where f_s is the sample rate (see Section 14.1.6.2.3 for details on how this clock rate is derived based on the S/PDIF format). The device receiving in S/PDIF format can recover the clock and frame information from the BMC signal.

Figure 14-10. Biphase-Mark Code (BMC)

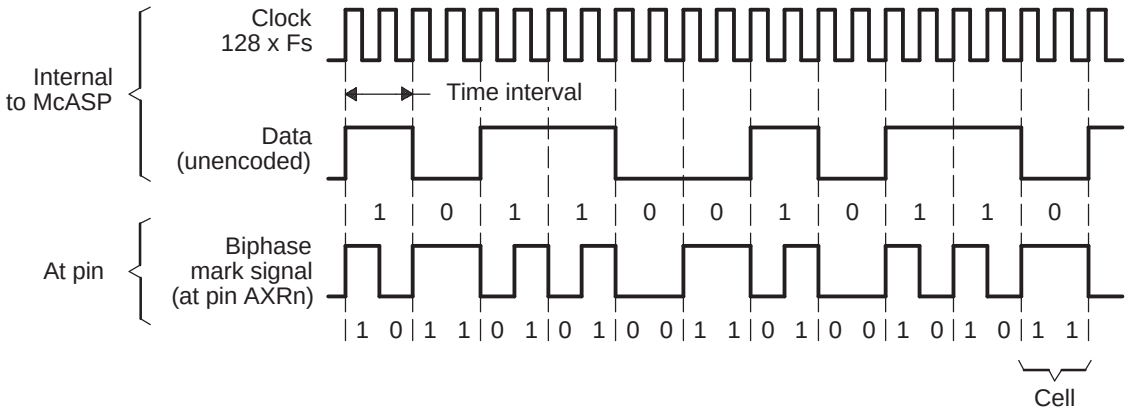


Table 14-1. Biphase-Mark Encoder

Data (Unencoded)	Previous State at Pin AXRn	BMC-Encoded Cell Output at AXRn
0	0	11
0	1	00
1	0	10
1	1	01

14.1.6.2.2 Subframe Format

Every audio sample transmitted in a subframe consists of 32 S/PDIF time intervals (or cells), numbered from 0 to 31. Figure 14-11 shows a subframe.

- **Time intervals 0-3** carry one of the three permitted preambles to signify the type of audio sample in the current subframe. The preamble is *not* encoded in BMC format, and therefore the preamble code can contain more than two consecutive 0 or 1 logical states in a row. See Table 14-2.
- **Time intervals 4-27** carry the audio sample word in linear 2s-complement representation. The most-significant bit (MSB) is carried by time interval 27. When a 24-bit coding range is used, the least-significant bit (LSB) is in time interval 4. When a 20-bit coding range is used, time intervals 8-27 carry the audio sample word with the LSB in time interval 8. Time intervals 4-7 may be used for other applications and are designated auxiliary sample bits.
- If the source provides fewer bits than the interface allows (either 20 or 24), the unused LSBs are set to logical 0. For a nonlinear PCM audio application or a data application, the main data field may carry any other information.
- **Time interval 28** carries the validity bit (V) associated with the main data field in the subframe.
- **Time interval 29** carries the user data channel (U) associated with the main data field in the subframe.
- **Time interval 30** carries the channel status information (C) associated with the main data field in the subframe. The channel status indicates if the data in the subframe is digital audio or some other type of data.
- **Time interval 31** carries a parity bit (P) such that time intervals 4-31 carry an even number of 1s and an even number of 0s (even parity). As shown in Table 14-2, the preambles (time intervals 0-3) are also defined with even parity.

Figure 14-11. S/PDIF Subframe Format

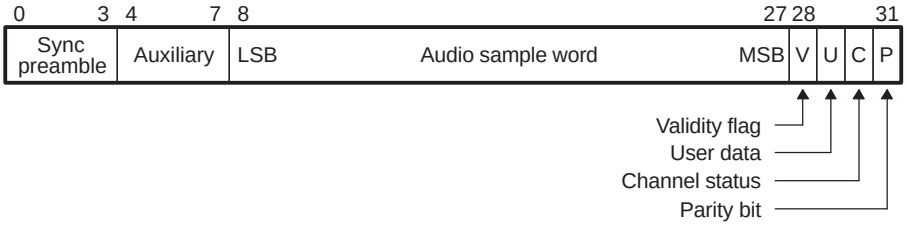


Table 14-2. Preamble Codes

Preamble Code[1]	Previous Logical State	Logical States on pin AXRn[2]	Description
B (or Z)	0	1110 1000	Start of a block and subframe 1
M (or X)	0	1110 0010	Subframe 1
W (or Y)	0	1110 0100	Subframe 2

(1) Historically, preamble codes are referred to as B, M, W. For use in professional applications, preambles are referred to as Z, X, Y, respectively.

(2) The preamble is not BMC encoded. Each logical state is synchronized to the serial clock. These 8 logical states make up time slots (cells) 0 to 3 in the S/PDIF stream.

As shown in Table 14-2, the McASP DIT only generates one polarity of preambles and it assumes the previous logical state to be 0. This is because the McASP assures an even-polarity encoding scheme when transmitting in DIT mode. If an underrun condition occurs, the DIT resynchronizes to the correct logic level on the AXRn pin before continuing with the next transmission.

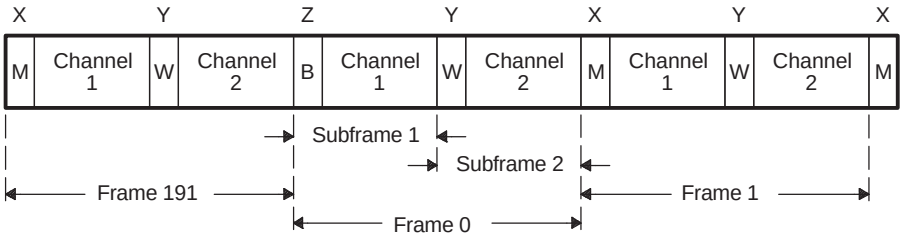
14.1.6.2.3 Frame Format

An S/PDIF frame is composed of two subframes (Figure 14-12). For linear coded audio applications, the rate of frame transmission normally corresponds exactly to the source sampling frequency f_s. The S/PDIF format clock rate is therefore 128 × f_s (128 = 32 cells/subframe × 2 clocks/cell × 2 subframes/sample). For example, for an S/PDIF stream at a 192 kHz sampling frequency, the serial clock is 128 × 192 kHz = 24.58 MHz.

In 2-channel operation mode, the samples taken from both channels are transmitted by time multiplexing in consecutive subframes. Both subframes contain valid data. The first subframe (**left** or **A** channel in stereophonic operation and **primary** channel in monophonic operation) normally starts with preamble M. However, the preamble of the first subframe changes to preamble B once every 192 frames to identify the start of the block structure used to organize the channel status information. The second subframe (**right** or **B** channel in stereophonic operation and **secondary** channel in monophonic operation) always starts with preamble W.

In single-channel operation mode in a professional application, the frame format is the same as in the 2-channel mode. Data is carried in the first subframe and may be duplicated in the second subframe. If the second subframe is not carrying duplicate data, cell 28 (validity bit) is set to logical 1.

Figure 14-12. S/PDIF Frame Format



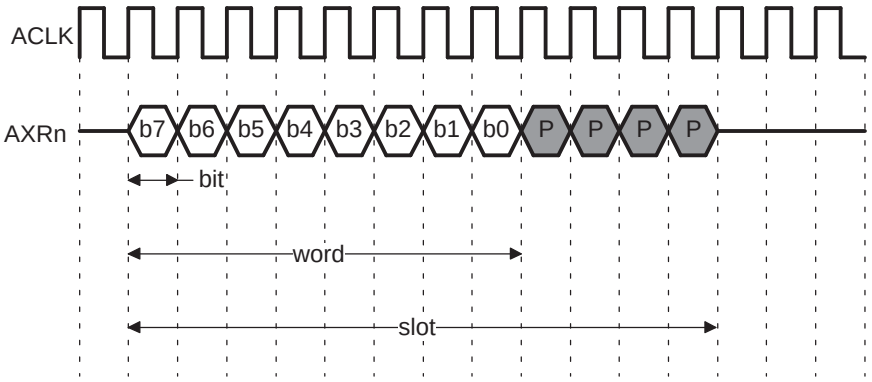
14.1.7 Definition of Terms

The serial bit stream transmitted or received by the McASP is a long sequence of 1s and 0s, either output or input on one of the audio transmit/receive pins (AXRn). However, the sequence has a hierarchical organization that can be described in terms of frames of data, slots, words, and bits.

A basic synchronous serial interface consists of three important components: clock, frame sync, and data. Figure 14-13 shows two of the three basic components—the clock (ACLK) and the data (AXRn). Figure 14-13 does not specify whether the clock is for transmit (ACLKX) or receive (ACLKR) because the definitions of terms apply to both receive and transmit interfaces. In operation, the transmitter uses ACLKX as the serial clock, and the receiver uses ACLKR as the serial clock. Optionally, the receiver can use ACLKX as the serial clock when the transmitter and receiver of the McASP are configured to operate synchronously.

Bit	A bit is the smallest entity in the serial data stream. The beginning and end of each bit is marked by an edge of the serial clock. The duration of a bit is a serial clock period. A "1" is represented by a logic high on the AXRn pin for the entire duration of the bit. A "0" is represented by a logic low on the AXRn pin for the entire duration of the bit.
Word	A word is a group of bits that make up the data being transferred between the processor and the external device. Figure 14-13 shows an 8-bit word.
Slot	A slot consists of the bits that make up the word, and may consist of additional bits used to pad the word to a convenient number of bits for the interface between the processor and the external device. In Figure 14-13, the audio data consists of only 8 bits of useful data (8-bit word), but it is padded with 4 zeros (12-bit slot) to satisfy the desired protocol in interfacing to an external device. Within a slot, the bits may be shifted in/out of the McASP on the AXRn pin either MSB or LSB first. When the word size is smaller than the slot size, the word may be aligned to the left (beginning) of the slot or to the right (end) of the slot. The additional bits in the slot not belonging to the word may be padded with 0, 1, or with one of the bits (the MSB or the LSB typically) from the data word. These options are shown in Figure 14-14.

Figure 14-13. Definition of Bit, Word, and Slot

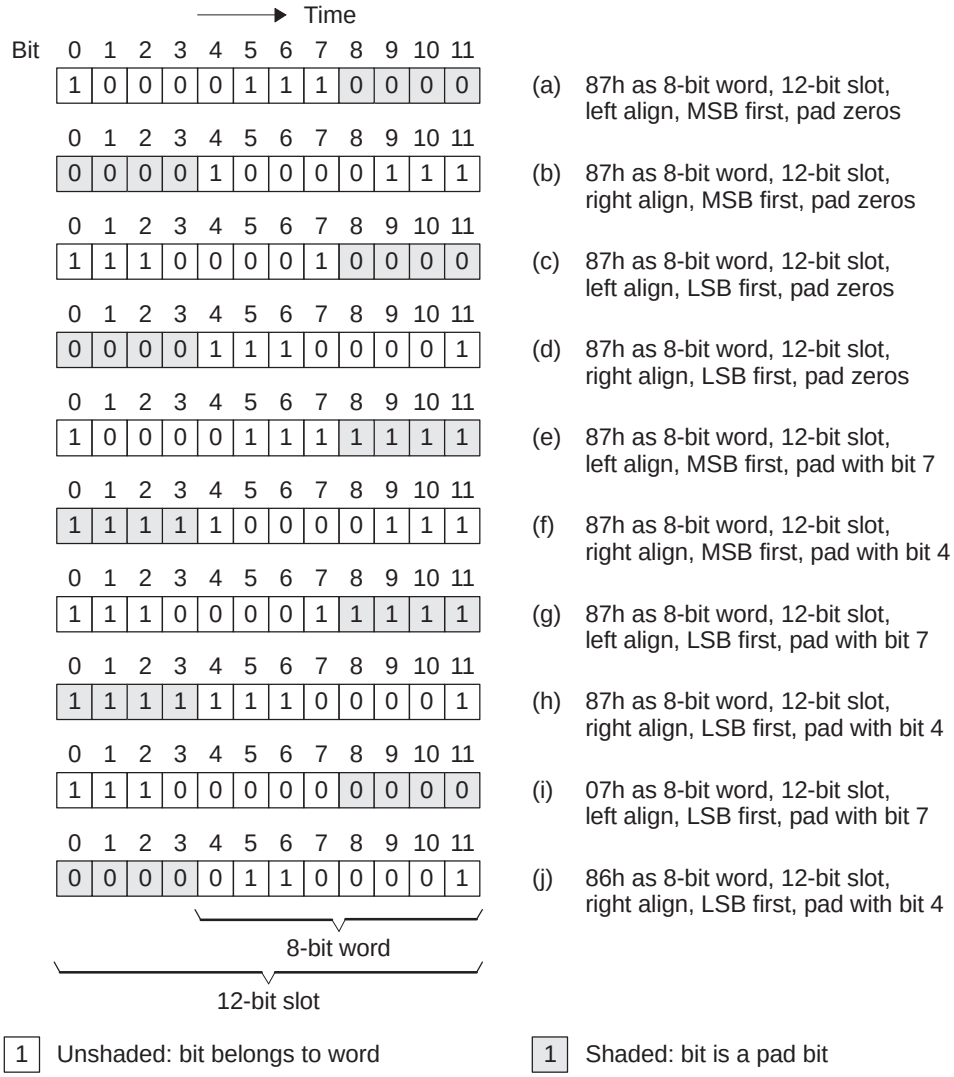


(1) b7:b0 - bits. Bits b7 to b0 form a word.

(2) P - pad bits. Bits b7 to b0, together with the four pad bits, form a slot.

(3) In this example, the data is transmitted MSB first, left aligned.

Figure 14-14. Bit Order and Word Alignment Within a Slot Examples

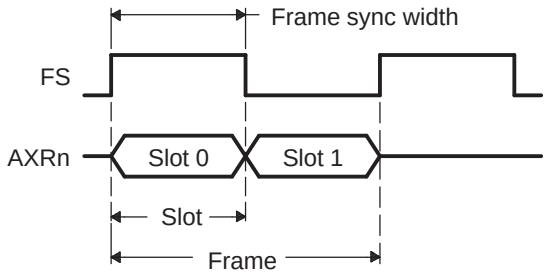


The third basic element of a synchronous serial interface is the frame synchronization signal, also referred to as frame sync (FS) in this document.

Frame	A frame contains one or multiple slots, as determined by the desired protocol. Figure 14-15 shows an example frame of data and the frame definitions. Figure 14-15 does not specify whether the frame sync (FS) is for transmit (AFSX) or receive (AFSR) because the definitions of terms apply to both receive and transmit interfaces. In operation, the transmitter uses AFSX and the receiver uses AFSR. Optionally, the receiver can use AFSX as the frame sync when the transmitter and receiver of the McASP are configured to operate synchronously.

This section only shows the generic definition of the frame sync. See Section 14.1.6 and Section 14.2.6.1 for details on the frame sync formats required for the different transfer modes and protocols (burst mode, TDM mode and I2S format, DIT mode and S/PDIF format).

Figure 14-15. Definition of Frame and Frame Sync Width



(1) In this example, there are two slots in a frame, and FS duration of slot length is shown.

Other terms used throughout the document:

TDM	Time-division multiplexed. See Section 14.1.6.1 for details on the TDM protocol.
DIR	Digital audio interface receive. The McASP does not natively support receiving in the S/PDIF format. The McASP supports I2S format output by an external DIR device.
DIT	Digital audio interface transmit. The McASP supports transmitting in S/PDIF format on up to all data pins configured as outputs.
I2S	Inter-Integrated Sound protocol, commonly used on audio interfaces. The McASP supports the I2S protocol as part of the TDM mode (when configured as a 2-slot frame).
Slot or Time Slot	For TDM format, the term time slot is interchangeable with the term slot defined in this section. For DIT format, a McASP time slot corresponds to a DIT subframe.

14.2 Architecture

14.2.1 Overview

Figure 14-1 shows the major blocks of the McASP. The McASP has independent receive/transmit clock generators and frame sync generators, error-checking logic, and up to 16 serial data pins. Refer to the device-specific data manual for the features available on your device.

All the McASP pins on the device may be individually programmed as general-purpose I/O (GPIO) if they are not used for serial port functions.

The McASP includes the following pins:

- Serializers;
  - Data pins AXRn: Up to 16 pins per McASP.
- Transmit clock generator:
  - AHCLKX: McASP transmit high-frequency master clock.
  - ACLKX: McASP transmit bit clock.
- Transmit Frame Sync Generator;
  - AFSX: McASP transmit frame sync or left/right clock (LRCLK).
- Receive clock generator;
  - AHCLKR: McASP receive high-frequency master clock.
  - ACLKR: McASP receive bit clock.
- Receive Frame Sync Generator;
  - AFSR: McASP receive frame sync or left/right clock (LRCLK).
- Mute in/out;
  - AMUTEIN: McASP mute input (from external device).
  - AMUTE: McASP mute output.

14.2.2 Clock and Frame Sync Generators

The McASP clock generators are able to produce two independent clock zones: transmit and receive clock zones. The serial clock generators may be programmed independently for the transmit section and the receive section, and may be completely asynchronous to each other. The serial clock (clock at the bit rate) may be sourced:

- Internally - by passing through two clock dividers off the internal clock source (AUXCLK).
- Externally - directly from ACLKR/X pin, which are configured as inputs. In this case, the Rx/Tx high-speed clock logic is bypassed for the XCLK/RCLK generation.
- Mixed - an external high-frequency clock is input to the McASP on either the AHCLKX or AHCLKR pins, and divided down to produce the bit rate clock.

In the internal/mixed cases, the bit rate clock is generated internally and should be driven out on the ACLKX (for transmit) or ACLKR (for receive) pins. In the internal case, an internally-generated high-frequency clock may be driven out onto the AHCLKX or AHCLKR pins to serve as a reference clock for other components in the system.

The McASP requires a minimum of a bit clock and a frame sync to operate, and provides the capability to reference these clocks from an external high-frequency master clock. In DIT mode, it is possible to use only internally-generated clocks and frame syncs.

A typical role of the MCASP frame sync signal is to carry the left/right clock (LRCLK) signal when transmitting and receiving stereo data.

14.2.2.1 Transmit Clock

The transmit bit clock, ACLKX, (Figure 14-16) may be either externally sourced from the ACLKX pin or internally generated, as selected by the CLKXM bit. If internally generated (CLKXM = 1), the clock is divided down by a programmable bit clock divider (CLKXDIV) from the transmit high-frequency master clock (AHCLKX).

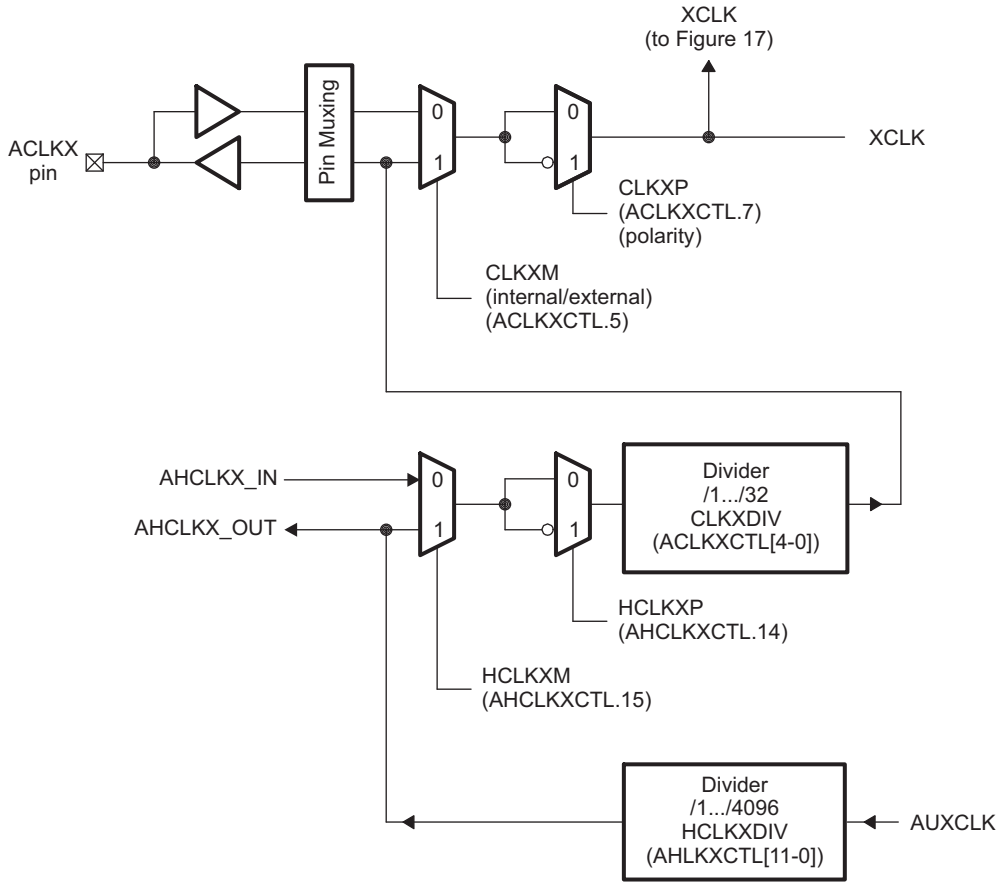
Internally, the McASP always shifts transmit data at the rising edge of the internal transmit clock, XCLK, (Figure 14-16). The CLKXP mux determines if ACLKX needs to be inverted to become XCLK. If CLKXP = 0, the CLKXP mux directly passes ACLKX to XCLK. As a result, the McASP shifts transmit data at the rising edge of ACLKX. If CLKXP = 1, the CLKX mux passes the inverted version of ACLKX to XCLK. As a result, the McASP shifts transmit data at the falling edge of ACLKX.

The transmit high-frequency master clock, AHCLKX, may be either externally sourced from the AHCLKX pin or internally generated, as selected by the HCLKXM bit. If internally generated (HCLKXM = 1), the clock is divided down by a programmable high clock divider (HCLKXDIV) from McASP internal clock source AUXCLK. The transmit high-frequency master clock may be (but is not required to be) output on the AHCLKX pin where it is available to other devices in the system.

The transmit clock configuration is controlled by the following registers:

- ACLKXCTL
- AHCLKXCTL

Figure 14-16. Transmit Clock Generator Block Diagram



(1) Refer to the device-specific date manual for multiplexing options.

14.2.2.2 Receive Clock

The MCASP receive clock generator is built on a very similar circuit to the transmit clock generator (but independent) circuit. The receiver also has the option to operate synchronously from the ACLKX and AFSX signals. This is achieved when the ASYNC bit in the transmit clock control register (ACLKXCTL) is cleared to 0 (see Figure 14-17). The receiver may be configured with different polarity (CLKRP) and frame sync data delay options from those options of the transmitter.

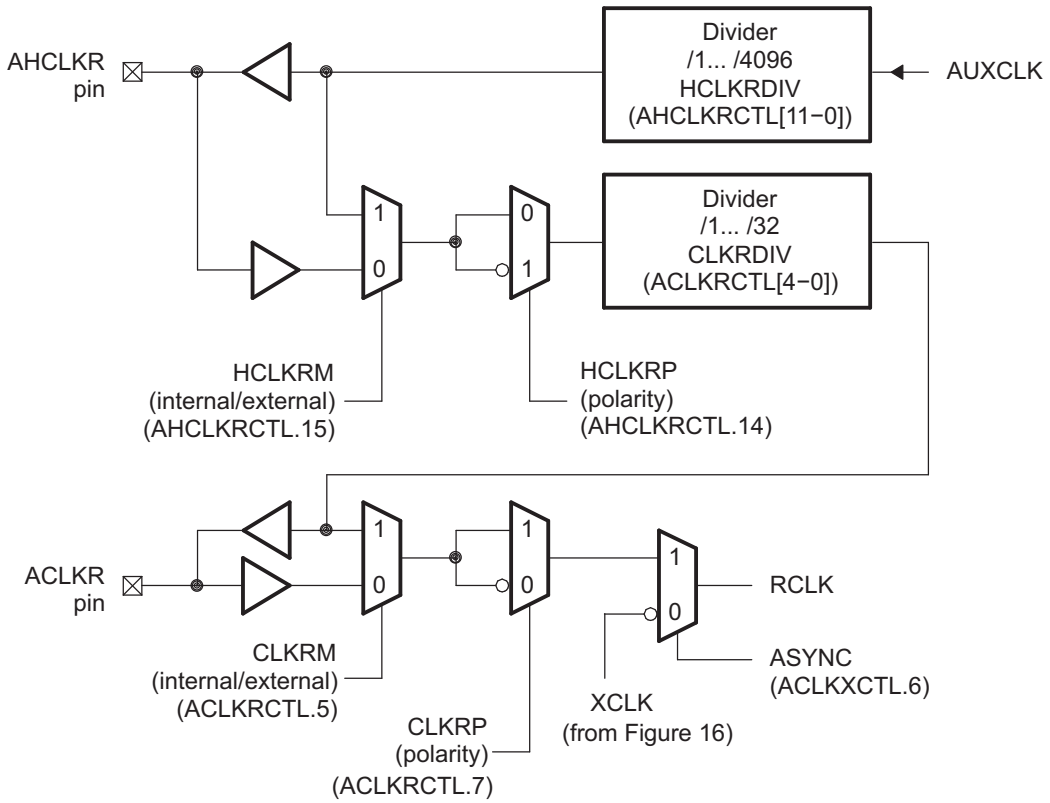
The receive clock configuration is controlled by the following registers:

- ACLKRCTL
- AHCLKRCTL

In case the receive bit clock, ACLKR, is generated internally (but asynchronously to XCLK), the CLKRM bit (from ACLKRCTL register) must be set to 1. Thus, the clock is divided down by a programmable bit clock divider (CLKRDIV bit field in ACLKRCTL register) from the source signal. If the receive high-frequency master clock, AHCLKR, is also sourced internally, the HCLKRM bit (AHCLKRCTL register) must be set to 1. Thus, the clock is divided down by a programmable high-clock divider (HCLKRDIV bit field in AHCLKRCTL register) from the MCASP internal clock source AUXCLK.

The receive high-frequency master clock, AHCLKR, may be (but is not required to be) output on the AHCLKR pin where it is available to other devices in the system. Regardless if AHCLKR is either internally generated or externally sourced, polarity of the high-frequency clock may be programmed via bit HCLKRP to be either rising or falling edge.

Figure 14-17. Receive Clock Generator Block Diagram



(1) Refer to the device-specific date manual for multiplexing options.

14.2.2.3 Frame Sync Generator

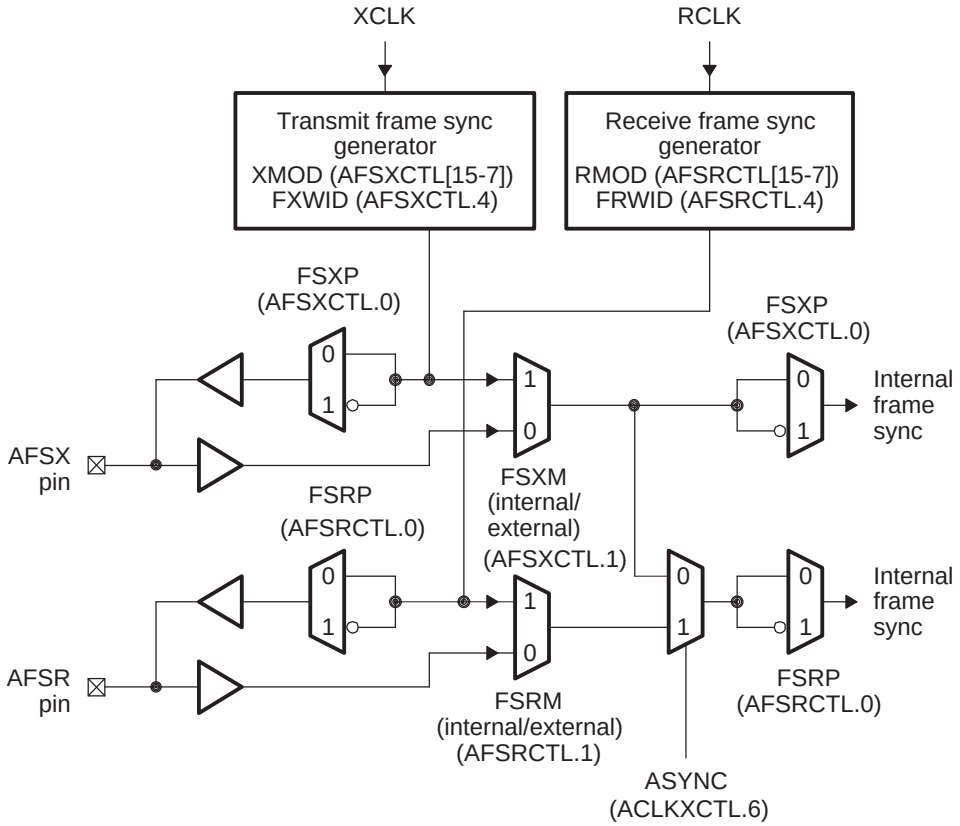
There are two different modes for frame sync: burst and TDM. A block diagram of the frame sync generator is shown in Figure 14-18. The frame sync options are programmed by the receive and transmit frame sync control registers (AFSRCTL and AFSXCTL). The options are:

- Internally-generated or externally-generated via configuring bit AFSXCTL[1] FSXM for transmit or AFSRCTL[1] FSRM for receive logic
- Frame sync polarity: rising edge or falling edge via configuring bit AFSXCTL[0] FSXP for transmit or AFSRCTL[0] FSRP for receive logic
- Frame sync width: single bit or single word via configuring bit AFSXCTL[4] FXWID for transmit or AFSRCTL[4] FRWID for receive logic
- Bit delay: 0, 1, or 2 cycles before the first data bit
- Selecting the source (AFSX or AFSR) of receiver internal frame synchronization. This is done in the same bit, ACLKXCTL[6] ASYNC, used to define the receiver internal clock source.

The transmit frame sync pin is AFSX and the receive frame sync pin is AFSR. A typical usage for these pins is to carry the left/right clock (LRCLK) signal when transmitting and receiving stereo data.

Regardless if the AFSX/AFSR is internally generated or externally sourced, the polarity of AFSX/AFSR is determined by FSXP/FSRP, respectively, to be either rising or falling edge. If FSXP/FSRP = 0, the frame sync polarity is rising edge. If FSRP/FSRP = 1, the frame sync polarity is falling edge.

Figure 14-18. Frame Sync Generator Block Diagram



14.2.2.4 Clocking Examples

Some examples of processes using the McASP clocking and frame flexibility are:

- Receive data from a DVD at 48 kHz, but output up-sampled or decoded audio at 96 kHz or 192 kHz. This could be accomplished by inputting a high-frequency master clock (for example, 512 × receive FS), receiving with an internally-generated bit clock ratio of divide-by-8, and transmitting with an internally-generated bit clock ratio of divide-by-4 or divide-by-2.
- Transmit/receive data based on one sample rate (for example, 44.1 kHz), and transmit/receive data at a different sample rate (for example, 48 kHz).

14.2.3 Memory Map

See the device-specific data manual for information describing the device memory-map.

14.2.4 Signal Descriptions

The signals used on the McASP audio interface are listed in Table 14-3.

Table 14-3. McASP Interface Signals

Pin[1]	I/O/Z	Device Reset (RESET = 0)	Description
		Transmitter Control	
AHCLKX	I/O/Z	Input	Transmit high-frequency master clock
AFSX	I/O/Z	Input	Transmit frame sync or left/right clock (LRCLK)
ACLKX	I/O/Z	Input	Transmit bit clock
		Receiver Control	
AHCLKR	I/O/Z	Input	Receive high-frequency master clock
AFSR	I/O/Z	Input	Receive frame sync or left/right clock (LRCLK)
ACLKR	I/O/Z	Input	Receive bit clock
		Mute	
AMUTE	I/O/Z	Input	Mute output
AMUTEIN	I/O/Z	Input	Mute input
		Data	
AXRn	I/O/Z	Input	TX/RX data pins. Up to 16 data pins are supported (n = 0 to 15).

(1) Due to pinout restrictions, not all interface signals are supported on all devices or all McASP instances. Refer to your device-specific data manual for details.

14.2.5 Pin Multiplexing

The McASP signals share pins with other processor functions. For detailed information on the McASP pin multiplexing and configuration, see the pin multiplexing information in the device-specific data manual.

14.2.6 Transfer Modes

14.2.6.1 Burst Transfer Mode

The McASP supports a burst transfer mode, which is useful for nonaudio data such as passing control information between two processors. Burst transfer mode uses a synchronous serial format similar to the TDM mode. The frame sync generation is not periodic or time-driven as in TDM mode, but data driven, and the frame sync is generated for each data word transferred.

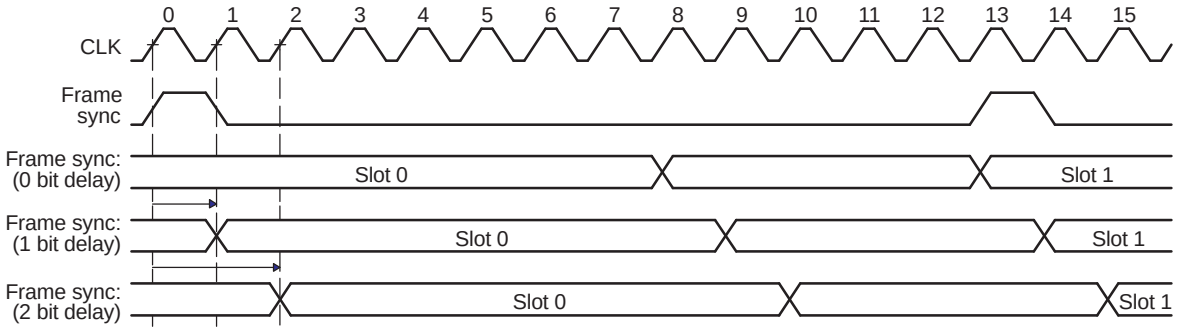
When operating in burst frame sync mode (Figure 14-19), as specified for transmit (XMOD = 0 in AFSXCTL) and receive (RMOD = 0 in AFSRCTL), one slot is shifted for each active edge of the frame sync signal that is recognized. Additional clocks after the slot and before the next frame sync edge are ignored.

In burst frame sync mode, the frame sync delay may be specified as 0, 1, or 2 serial clock cycles. This is the delay between the frame sync active edge and the start of the slot. The frame sync signal lasts for a single bit clock duration (FRWID = 0 in AFSRCTL, FXWID = 0 in AFSXCTL).

For transmit, when generating the transmit frame sync internally, the frame sync begins when the previous transmission has completed and when all the XBUF[n] (for every serializer set to operate as a transmitter) has been updated with new data.

For receive, when generating the receive frame sync internally, frame sync begins when the previous transmission has completed and when all the RBUF[n] (for every serializer set to operate as a receiver) has been read.

Figure 14-19. Burst Frame Sync Mode



The control registers must be configured as follows for the burst transfer mode. The burst mode specific bit fields are in bold face:

- PFUNC: The clock, frame, data pins must be configured for McASP function.
- PDIR: The clock, frame, data pins must be configured to the direction desired.
- PDOUT, PDIN, PDSET, PDCLR: Not applicable. Leave at default.
- GBLCTL: Follow the initialization sequence in Section 14.2.10.2 to configure this register.
- AMUTE: Not applicable. Leave at default.
- DLBCTL: If loopback mode is desired, configure this register according to Section 14.2.8.5, otherwise leave this register at default.
- DITCTL: DITEN must be left at default 0 to select non-DIT mode. Leave the register at default.
- RMASK/XMASK: Mask desired bits according to Section 14.2.7.2 and Section 14.2.8.3.
- RFMT/XFMT: Program all fields according to data format desired. See Section 14.2.8.3.
- AFSRCTL/AFSXCTL: Clear **RMOD/XMOD** bits to 0 to indicate burst mode. Clear **FRWID/FXWID** bits to 0 for single bit frame sync duration. Configure other fields as desired.
- ACLKRCTL/ACLKXCTL: Program all fields according to bit clock desired. See Section 14.2.2.
- AHCLKRCTL/AHCLKXCTL: Program all fields according to high-frequency clock desired. See Section 14.2.2.

- RTDM/XTDM: Program RTDMS0/XTDMS0 to 1 to indicate one active slot only. Leave other fields at default.
- RINTCTL/XINTCTL: Program all fields according to interrupts desired.
- RCLKCHK/XCLKCHK: Not applicable. Leave at default.
- SRCTLn: Program SRMOD to inactive/transmitter/receiver as desired. DISMOD is not applicable and should be left at default.
- DITCSRA[n], DITCSRB[n], DITUDRA[n], DITUDRB[n]: Not applicable. Leave at default.

14.2.6.2 Time-Division Multiplexed (TDM) Transfer Mode

The McASP time-division multiplexed (TDM) transfer mode supports the TDM format discussed in Section 14.1.6.1.

Transmitting data in the TDM transfer mode requires a minimum set of pins:

- ACLKX - transmit bit clock.
- AFSX - transmit frame sync (or commonly called left/right clock).
- One or more serial data pins, AXRn, whose serializers have been configured to transmit.

The transmitter has the option to receive the ACLKX bit clock as an input, or to generate the ACLKX bit clock by dividing down the AHCLKX high-frequency master clock. The transmitter can either generate AHCLKX internally or receive AHCLKX as an input. See Section 14.2.2.1.

Similarly, to receive data in the TDM transfer mode requires a minimum set of pins:

- ACLKR - receive bit clock.
- AFSR - receive frame sync (or commonly called left/right clock).
- One or more serial data pins, AXRn, whose serializers have been configured to receive.

The receiver has the option to receive the ACLKR bit clock as an input or to generate the ACLKR bit clock by dividing down the AHCLKR high-frequency master clock. The receiver can either generate AHCLKR internally or receive AHCLKR as an input. See Section 14.2.2.2 and Section 14.2.2.3.

The control registers must be configured as follows for the TDM mode. The TDM mode specific bit fields are in bold face:

- PFUNC: The clock, frame, data pins must be configured for McASP function.
- PDIR: The clock, frame, data pins must be configured to the direction desired.
- PDOUT, PDIN, PDSET, PDCLR: Not applicable. Leave at default.
- GBLCTL: Follow the initialization sequence in Section 14.2.10.2 to configure this register.
- AMUTE: Program all fields according to mute control desired.
- DLBCTL: If loopback mode is desired, configure this register according to Section 14.2.8.5, otherwise leave this register at default.
- DITCTL: DITEN must be left at default 0 to select TDM mode. Leave the register at default.
- RMASK/XMASK: Mask desired bits according to Section 14.2.7.2 and Section 14.2.8.3.
- RFMT/XFMT: Program all fields according to data format desired. See Section 14.2.8.3.
- AFSRCTL/AFSXCTL: Set **RMOD/XMOD** bits to 2-32 for TDM mode. Configure other fields as desired.
- ACLKRCTL/ACLKXCTL: Program all fields according to bit clock desired. See Section 14.2.2.
- AHCLKRCTL/AHCLKXCTL: Program all fields according to high-frequency clock desired. See Section 14.2.2.
- RTDM/XTDM: Program all fields according to the time slot characteristics desired.
- RINTCTL/XINTCTL: Program all fields according to interrupts desired.
- RCLKCHK/XCLKCHK: Program all fields according to clock checking desired.
- SRCTLn: Program all fields according to serializer operation desired.
- DITCSRA[n], DITCSRB[n], DITUDRA[n], DITUDRB[n]: Not applicable. Leave at default.

14.2.6.2.1 TDM Time Slots

TDM mode on the McASP can extend to support multiprocessor applications, with up to 32 time slots per frame. For each of the time slots, the McASP may be configured to participate or to be inactive by configuring XTDM and/or RTDM (this allows multiple processors to communicate on the same TDM serial bus).

The TDM sequencer (separate ones for transmit and receive) functions in this mode. The TDM sequencer counts the slots beginning with the frame sync. For each slot, the TDM sequencer checks the respective bit in either XTDM or RTDM to determine if the McASP should transmit/receive in that time slot.

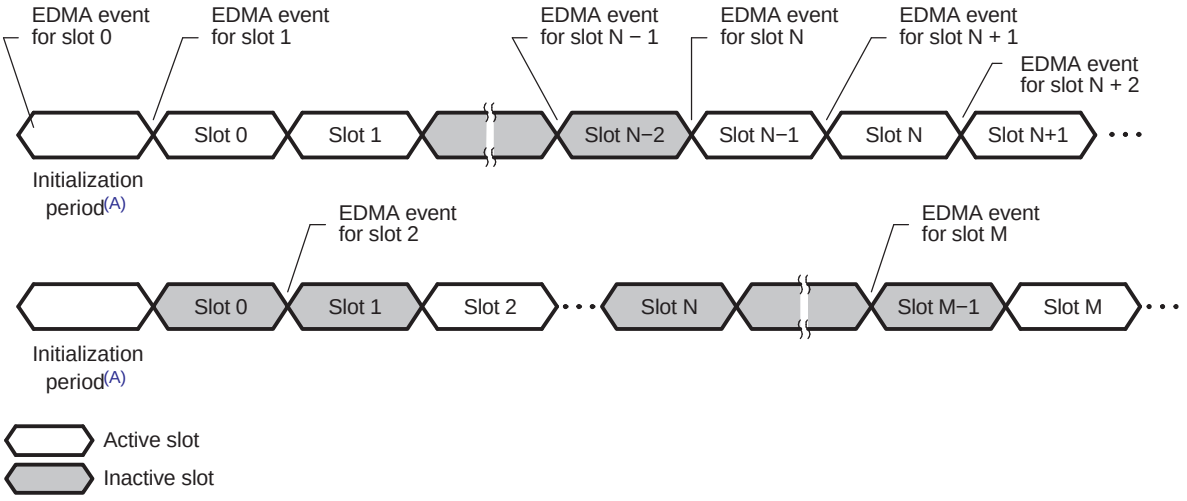
If the transmit/receive bit is active, the McASP functions normally during that time slot; otherwise, the McASP is inactive during that time slot; no update to the buffer occurs, and no event is generated. Transmit pins are automatically set to a high-impedance state, 0, or 1 during that slot, as determined by bit DISMOD in SRCTL[n].

Figure 14-20 shows when the transmit DMA event AXEVT is generated. See Section 14.2.8.1.1 for details on data ready and the initialization period indication. The transmit DMA event for an active time slot (slot N) is generated during the previous time slot (slot N - 1), regardless if the previous time slot (slot N - 1) is active or inactive.

During an active transmit time slot (slot N), if the next time slot (slot N + 1) is configured to be active, the copy from XRBUF[n] to XRSR[n] generates the DMA event for time slot N + 1. If the next time slot (slot N + 1) is configured to be inactive, then the DMA event will be delayed to time slot M - 1. In this case, slot M is the next active time slot. The DMA event for time slot M is generated during the first bit time of slot M - 1.

The receive DMA request generation does not need this capability, since the receive DMA event is generated after data is received in the buffer (looks back in time). If a time slot is disabled, then no data is copied to the buffer for that time slot and no DMA event is generated.

Figure 14-20. Transmit DMA Event (AXEVT) Generation in TDM Time Slots



A See Section 14.2.10.2, step 7a.

14.2.6.2.2 Special 384 Slot TDM Mode for Connection to External DIR

The McASP receiver also supports a 384 time slot TDM mode (DIR mode), to support S/PDIF, AES-3, IEC-60958 receiver ICs whose natural block (block corresponds to McASP frame) size is 384 samples. The advantage to using the 384 time slot TDM mode is that interrupts may be generated synchronous to the S/PDIF, AES-3, IEC-60958, such as the last slot interrupt.

The receive TDM time slot register (RTDM) should be programmed to all 1s during reception of a DIR block. Other TDM functionalities (for example, inactive slots) are not supported (only the slot counter counts the 384 subframes in a block).

To receive data in the DIR mode, the following pins are typically needed:

- ACLKR - receive bit clock.
- AFSR - receive frame sync (or commonly called left/right clock). In this mode, AFSR should be connected to a DIR which outputs a start of block signal, instead of LRCLK.
- One or more serial data pins, AXRn, whose serializers have been configured to receive.

For this special DIR mode, the control registers can be configured just as for TDM mode, except set RMOD in AFSRCTL to 384 to receive 384 time slots.

14.2.6.3 Digital Audio Interface Transmit (DIT) Transfer Mode

In addition to the TDM and burst transfer modes, which are suitable for transmitting audio data between ICs inside the same system, the digital audio interface transmit (DIT) transfer mode of the McASP also supports transmission of audio data in the S/PDIF, AES-3, or IEC-60958 format. These formats are designed to carry audio data between different systems through an optical or coaxial cable. The DIT mode only applies to serializers configured as transmitters, not receivers. Refer to Section 14.1.6.2 for a description of the S/PDIF format.

14.2.6.3.1 Transmit DIT Encoding

The McASP operation in DIT mode is basically identical to the 2 time slot TDM mode, but the data transmitted is output as a biphase mark encoded bit stream, with preamble, channel status, user data, validity, and parity automatically stuffed into the bit stream by the McASP. The McASP includes separate validity bits for even/odd subframes and two 384-bit RAM modules to hold channel status and user data bits.

The transmit TDM time slot register (XTDM) should be programmed to all 1s during DIT mode. TDM functionality is not supported in DIT mode, except that the TDM slot counter counts the DIT subframes.

To transmit data in the DIT mode, the following pins are typically needed:

- AHCLKX - transmit high-frequency master clock.
- One or more serial data pins, AXRn, whose serializers have been configured to transmit.

AHCLKX is optional (the internal clock source may be used instead), but if used as a reference, the processor provides a clock check circuit that continually monitors the AHCLKX input for stability.

If the McASP is configured to transmit in the DIT mode on more than one serial data pin, the bit streams on all pins will be synchronized. In addition, although they will carry unique audio data, they will carry the same channel status, user data, and validity information.

The actual 24-bit audio data must always be in bit positions 23-0 after passing through the first three stages of the transmit format unit.

For left-aligned Q31 data, the following transmit format unit settings process the data into right aligned 24-bit audio data ready for transmission:

- XROT = 010 (rotate right by 8 bits).
- XRVRS = 0 (no bit reversal, LSB first).
- XMASK = FFFF FF00h-FFFF 0000h (depending upon whether 24, 23, 22, 21, 20, 19, 18, 17, or 16 valid audio data bits are present).
- XPAD = 00 (pad extra bits with 0).

For right-aligned data, the following transmit format unit settings process the data into right aligned 24-bit audio data ready for transmission:

- XROT = 000 (rotate right by 0 bits).
- XRVRS = 0 (no bit reversal, LSB first).
- XMASK = 00FF FFFFh to 0000 FFFFh (depending upon whether 24, 23, 22, 21, 20, 19, 18, 17, or 16 valid audio data bits are present).
- XPAD = 00 (pad extra bits with 0).

14.2.6.3.2 Transmit DIT Clock and Frame Sync Generation

The DIT transmitter only works in the following configuration:

- In transmit frame control register (AFSXCTL):
  - Internally-generated transmit frame sync, FSXM = 1.
  - Rising-edge frame sync, FSXP = 0.
  - Bit-width frame sync, FXWID = 0.
  - 384-slot TDM, XMOD = 1 1000 0000b.
- In transmit clock control register (ACLKXCTL), ASYNC = 1.
- In transmit bitstream format register (XFMT), XSSZ = 1111 (32-bit slot size).

All combinations of AHCLKX and ACLKX are supported.

This is a summary of the register configurations required for DIT mode. The DIT mode specific bit fields are in bold face:

- PFUNC: The data pins (AXRn) must be configured for McASP function. If AHCLKX is used, it must also be configured for McASP function.
- PDIR: The data pins (AXRn) must be configured as outputs. If AHCLKX is used as an input reference, it should be configured as input. If internal clock source AUXCLK is used as the reference clock, it may be output on the AHCLKX pin by configuring AHCLKX as an output.
- PDOUT, PDIN, PDSET, PDCLR: Not applicable for DIT operation. Leave at default.
- GBLCTL: Follow the initialization sequence in Section 14.2.10.2 to configure this register.
- AMUTE: Program all fields according to mute control desired.
- DLBCTL: Not applicable. Loopback is not supported for DIT mode. Leave at default.
- DITCTL: **DITEN** bit must be set to 1 to enable DIT mode. Configure other bits as desired.
- RMASK: Not applicable. Leave at default.
- RFMT: Not applicable. Leave at default.
- AFSRCTL: Not applicable. Leave at default.
- ACLKRCTL: Not applicable. Leave at default.
- AHCLKRCTL: Not applicable. Leave at default.
- RTDM: Not applicable. Leave at default.
- RINTCTL: Not applicable. Leave at default.
- RCLKCHK: Not applicable. Leave at default.
- **XMASK**: Mask desired bits according to the discussion in this section, depending upon left-aligned or right-aligned internal data.

- **XFMT: XDATDLY** = 0. **XRVRS** = 0. **XPAD** = 0. **XPBIT** = default (not applicable). **XSSZ** = Fh (32-bit slot). XBUSEL = configured as desired. **XROT** bit is configured according to the discussion in this section, either 0 or 8-bit rotate.
- **AFSXCTL**: Configure the bits according to the discussion in this section.
- **ACLKXCTL**: **ASYNC** = 1. Program CLKXDIV bits to obtain the bit clock rate desired. Configure CLKXP and CLKXM bits as desired, because CLKX is not actually used in the DIT protocol.
- **AHCLKXCTL**: Program all fields according to high-frequency clock desired.
- **XTDM**: Set to FFFF FFFFh for all active slots for DIT transfers.
- XINTCTL: Program all fields according to interrupts desired.
- XCLKCHK: Program all fields according to clock checking desired.
- SRCTLn: Set **SRMOD** = 1 (transmitter) for the DIT pins. DISMOD field is don't care for DIT mode.
- **DITCSRA[n], DITCSRB[n]**: Program the channel status bits as desired.
- **DITUDRA[n], DITUDRB[n]**: Program the user data bits as desired.

14.2.6.3.3 DIT Channel Status and User Data Register Files

The channel status registers (DITCSRA*n* and DITCSRB*n*) and user data registers (DITUDRA*n* and DITUDRB*n*) are not double buffered. Typically the programmer uses one of the synchronizing interrupts, such as last slot, to create an event at a safe time so the register may be updated. In addition, the CPU reads the transmit TDM slot counter to determine which word of the register is being used.

It is a requirement that the software avoid writing to the word of user data and channel status that are being used to encode the current time slot; otherwise, it will be indeterminate whether the old or new data is used to encode the bitstream.

The DIT subframe format is defined in Section 14.1.6.2.2. The channel status information (C) and User Data (U) are defined in these DIT control registers:

- DITCSRA0 to DITCSRA5: The 192 bits in these six registers contain the channel status information for the LEFT channel within each frame.
- DITCSRB0 to DITCSRB5: The 192 bits in these six registers contain the channel status information for the RIGHT channel within each frame.
- DITUDRA0 to DITUDRA5: The 192 bits in these six registers contain the user data information for the LEFT channel within each frame.
- DITUDRB0 to DITUDRB5: The 192 bits in these six registers contain the user data information for the RIGHT channel within each frame.

The S/PDIF block format is shown in Figure 14-12. There are 192 frames within a block (frame 0 to frame 191). Within each frame there are two subframes (subframe 1 and 2 for left and right channels, respectively). The channel status and user data information sent on each subframe is summarized in Table 14-4.

Table 14-4. Channel Status and User Data for Each DIT Block

Frame	Subframe	Preamble	Channel Status defined in:	User Data defined in:
Defined by DITCSRA0, DITCSRB0, DITUDRA0, DITUDRB0				
0	1 (L)	B	DITCSRA0[0]	DITUDRA0[0]
0	2 (R)	W	DITCSRB0[0]	DITUDRB0[0]
1	1 (L)	M	DITCSRA0[1]	DITUDRA0[1]
1	2 (R)	W	DITCSRB0[1]	DITUDRB0[1]
2	1 (L)	M	DITCSRA0[2]	DITUDRA0[2]
2	2 (R)	W	DITCSRB0[2]	DITUDRB0[2]
…	…	…	…	…
31	1 (L)	M	DITCSRA0[31]	DITUDRA0[31]
31	2 (R)	W	DITCSRB0[31]	DITUDRB0[31]
Defined by DITCSRA1, DITCSRB1, DITUDRA1, DITUDRB1				
32	1 (L)	M	DITCSRA1[0]	DITUDRA1[0]
32	2 (R)	W	DITCSRB1[0]	DITUDRB1[0]
…	…	…	…	…
63	1 (L)	M	DITCSRA1[31]	DITUDRA1[31]
63	2 (R)	W	DITCSRB1[31]	DITUDRB1[31]
Defined by DITCSRA2, DITCSRB2, DITUDRA2, DITUDRB2				
64	1 (L)	M	DITCSRA2[0]	DITUDRA2[0]
64	2 (R)	W	DITCSRB2[0]	DITUDRB2[0]
…	…	…	…	…
95	1 (L)	M	DITCSRA2[31]	DITUDRA2[31]
95	2 (R)	W	DITCSRB2[31]	DITUDRB2[31]
Defined by DITCSRA3, DITCSRB3, DITUDRA3, DITUDRB3				
96	1 (L)	M	DITCSRA3[0]	DITUDRA3[0]
96	2 (R)	W	DITCSRB3[0]	DITUDRB3[0]
…	…	…	…	…
127	1 (L)	M	DITCSRA3[31]	DITUDRA3[31]
127	2 (R)	W	DITCSRB3[31]	DITUDRB3[31]
Defined by DITCSRA4, DITCSRB4, DITUDRA4, DITUDRB4				
128	1 (L)	M	DITCSRA4[0]	DITUDRA4[0]
128	2 (R)	W	DITCSRB4[0]	DITUDRB4[0]
…	…	…	…	…
159	1 (L)	M	DITCSRA4[31]	DITUDRA4[31]
159	2 (R)	W	DITCSRB4[31]	DITUDRB4[31]
Defined by DITCSRA5, DITCSRB5, DITUDRA5, DITUDRB5				
160	1 (L)	M	DITCSRA5[0]	DITUDRA5[0]
160	2 (R)	W	DITCSRB5[0]	DITUDRB5[0]
…	…	…	…	…
191	1 (L)	M	DITCSRA5[31]	DITUDRA5[31]
191	2 (R)	W	DITCSRB5[31]	DITUDRB5[31]

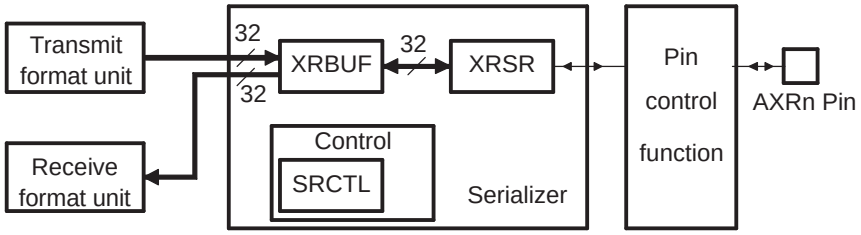
14.2.7 General Architecture

14.2.7.1 Serializers

Figure 14-21 shows the block diagram of the serializer and its interface to other units within the McASP. The serializers take care of shifting serial data in and out of the McASP. Each serializer consists of a shift register (XRSR), a data buffer (XRBUF), a control register (SRCTL), and logic to support the data alignment options of the McASP. For each serializer (XRSRn), there is a dedicated serial data pin (AXRn) and a dedicated control register (SRCTL[n]). The control register allows the serializer to be configured as a transmitter, receiver, or as inactive. When configured as a transmitter the serializer shifts out data to the serial data pin AXRn. When configured as a receiver, the serializer shifts in data from the AXRn pin. The serializer is clocked from the transmit/receive section clock (ACLKX/ACLKR) if configured to transmit/receive respectively.

All serializers that are configured to transmit operate in lock-step. Similarly, all serializers that are configured to receive also operate in lock-step. This means that at most there are two zones per McASP, one for transmit and one for receive.

Figure 14-21. Individual Serializer and Connections Within McASP



For receive, data is shifted in through the AXRn pin to the shift register XRSR. Once the entire slot of data is collected in the XRSR, the data is copied to the data buffer XRBUF. The data is now ready to be read by the processor through the RBUF register, which is an alias of the XRBUF for receive. When the processor reads from the RBUF, the McASP passes the data from RBUF through the receive format unit and returns the formatted data to the processor.

For transmit, the processor services the McASP by writing data into the XBUF register, which is an alias of the XRBUF for transmit. The data automatically passes through the transmit format unit before actually reaching the XRBUF register in the serializer. The data is then copied from XRBUF to XRSR, and shifted out from the AXRn synchronously to the serial clock.

In DIT mode, in addition to the data, the serializer shifts out other DIT-specific information accordingly (preamble, user data, etc.).

The serializer configuration is controlled by SRCTL[n]. A serializer n is configured as inactive via setting bitfield SRCTLn[1:0] SRMOD to 0. For a transmitting serializer, the SRCTLn[3:2] DISMOD bitfield defines the AXRn pin output state during inactive slots (logic high, logic low, or 3-state). A serializer n is configured in a transmit function via setting bitfield SRCTLn[1:0] SRMOD to 1. A serializer n is configured in a receive function via setting bitfield SRCTLn[1:0] SRMOD to 2h.

14.2.7.2 Format Unit

The McASP has two data formatting units, one for transmit and one for receive. These units automatically remap the data bits within the transmitted and received words between a natural format for the processor (such as a Q31 representation) and the required format for the external serial device (such as "I2S format"). During the remapping process, the format unit also can mask off certain bits or perform sign extension.

Since all transmitters share the same data formatting unit, the McASP only supports one transmit format at a time. For example, the McASP will not transmit in "I2S format" on serializer 0, while transmitting "Left Justified" on serializer 1. Likewise, the receiver section of the McASP only supports one data format at a time, and this format applies to all receiving serializers. However, the McASP can transmit in one format while receiving in a completely different format.

This formatting unit consists of three stages:

- Bit mask and pad (masks off bits, performs sign extension)
- Rotate right (aligns data within word)
- Bit reversal (selects between MSB first or LSB first)

The MCASP transmitter supports serial formats of:

- Slot (or Time slot) size = 8, 12, 16, 20, 24, 28, 32 bits
- Word size ≤ Slot size
- Alignment: when more bits/slot than bits/words, then:
  - Left aligned = word shifted first, remaining bits are pad
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot
- Order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB
  - LSB: least-significant bit of word is shifted out last, last bit is MSB

Hardware support for these serial formats comes from the programmable options in the bitstream format register, (R/X)FMT:

- XRVRS: bit reverse (1) or no bit reverse (0)
- XROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits
- XSSZ: receive slot size of 8, 12, 16, 20, 24, 28, or 32 bits

Figure 14-22 shows a block diagram of the receive formatting unit, and Figure 14-23 shows the transmit formatting unit. Note that the order in which data flows through the three stages is different between the transmit and receive formatting units.

As shown in Figure 14-23, the data to the transmit format unit (TFU) can come from the configuration port (CFG) or the data port (DATA). The selection is made through the RFMT[3] RBUSEL bit. According to port type selected, data transfer has different behavior.

Figure 14-22. Receive Format Unit

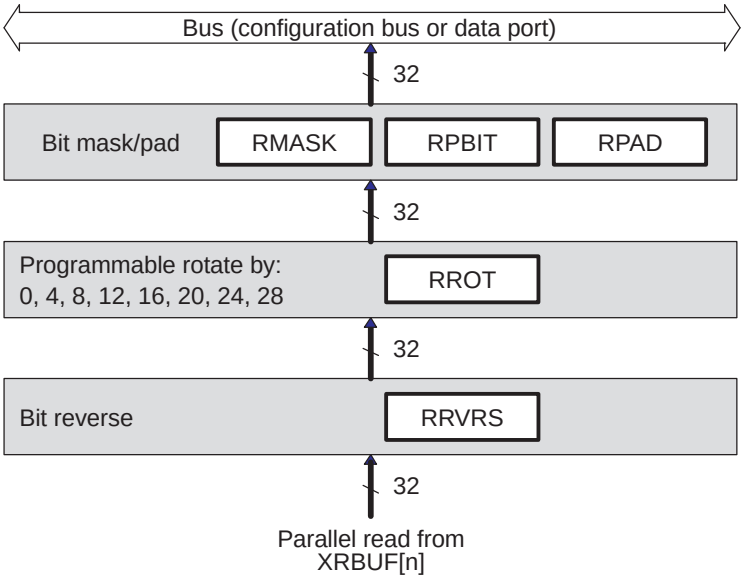
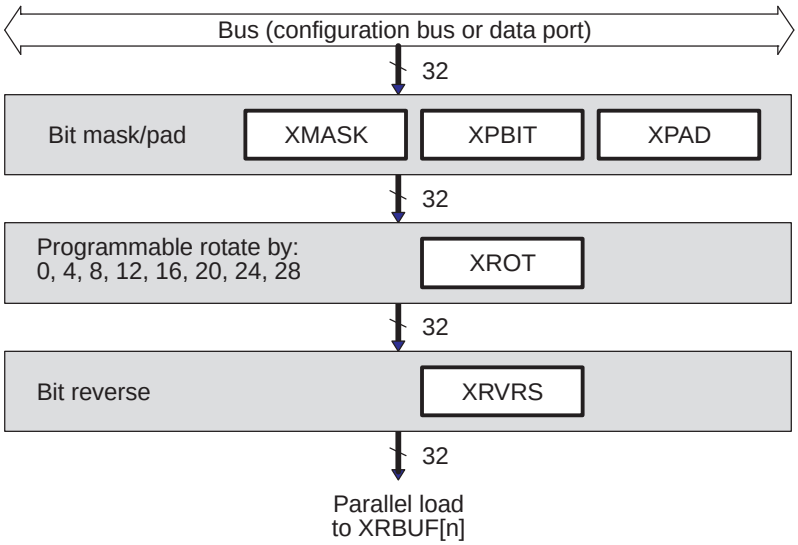


Figure 14-23. Transmit Format Unit



The bit mask and pad stage includes a full 32-bit mask register, allowing selected individual bits to either pass through the stage unchanged, or be masked off. The bit mask and pad then pad the value of the masked off bits by inserting either a 0, a 1, or one of the original 32 bits as the pad value. The last option allows for sign-extension when the sign bit is selected to pad the remaining bits.

The rotate right stage performs bitwise rotation by a multiple of 4 bits (between 0 and 28 bits), programmable by the (R/X)FMT register. Note that this is a rotation process, not a shifting process, so bit 0 gets shifted back into bit 31 during the rotation.

The bit reversal stage either passes all 32 bits directly through, or swaps them. This allows for either MSB or LSB first data formats. If bit reversal is not enabled, then the McASP will naturally transmit and receive in an LSB first order.

Finally, note that the (R/X)DATDLY bits in (R/X)FMT also determine the data format. For example, the difference between I2S format and left-justified is determined by the delay between the frame sync edge and the first data bit of a given time slot. For I2S format, (R/X)DATDLY should be set to a 1-bit delay, whereas for left-justified format, it should be set to a 0-bit delay.

The combination of all the options in (R/X)FMT means that the McASP supports a wide variety of data formats, both on the serial data lines, and in the internal processor representation.

Section 14.2.8.3 provides more detail and specific examples. The examples use internal representation in integer and Q31 notation, but other fractional notations are also possible.

14.2.7.3 State Machine

The receive and transmit sections have independent state machines. Each state machine controls the interactions between the various units in the respective section. In addition, the state machine keeps track of error conditions and serial port status.

No serial transfers can occur until the respective state machine is released from reset. See initialization sequence for details (Section 14.2.10).

The receive state machine is controlled by the RFMT register, and it reports the McASP status and error conditions in the RSTAT register. Similarly, the transmit state machine is controlled by the XFMT register, and it reports the McASP status and error conditions in the XSTAT register.

14.2.7.4 TDM Sequencer

There are separate TDM sequencers for the transmit section and the receive section. Each TDM sequencer keeps track of the slot count. In addition, the TDM sequencer checks the bits of (R/X)TDM and determines if the McASP should receive/transmit in that time slot.

If the McASP should participate (transmit/receive bit is active) in the time slot, the McASP functions normally. If the McASP should not participate (transmit/receive bit is inactive) in the time slot, no transfers between the XRBUF and XRSR registers in the serializer would occur during that time slot. In addition, the serializers programmed as transmitters place their data output pins in a predetermined state (logic low, high, or high impedance) as programmed by each serializer control register (SRCTL). Refer also to Section 14.2.6.2 for details on how DMA event or interrupt generations are handled during inactive time slots in TDM mode.

The receive TDM sequencer is controlled by register RTDM and reports current receive slot to RSLOT. The transmit TDM sequencer is controlled by register XTDM and reports current transmit slot to XSLOT.

14.2.7.5 Clock Check Circuit

A common source of error in audio systems is a serial clock failure due to instabilities in the off-chip DIR circuit. To detect a clock error quickly, a clock-check circuit is included in the McASP for both transmit and receive clocks, since both may be sourced from off chip.

The clock check circuit can detect and recover from transmit and receive clock failures. See Section 14.2.8.4.6 for implementation and programming details.

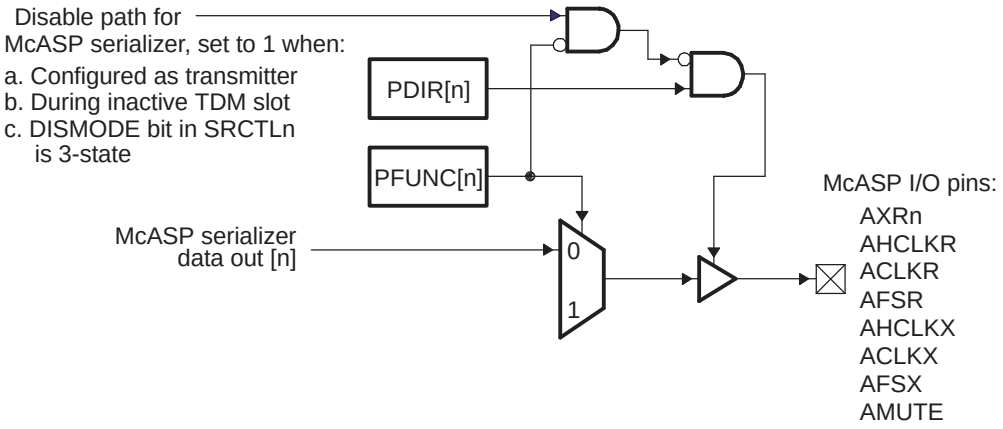
14.2.7.6 Pin Function Control

All McASP pins except AMUTEIN are bidirectional input/output pins. In addition, these bidirectional pins function either as McASP or general-purpose I/O (GPIO) pins. The following registers control the pin functions:

- Pin function register (PFUNC): selects pin to function as McASP or GPIO.
- Pin direction register (PDIR): selects pin to be input or output.
- Pin data input register (PDIN): shows data input at the pin.
- Pin data output register (PDOUT): data to be output at the pin if the pin is configured as GPIO output (PFUNC[n] = 1 and PDIR[n] = 1). Not applicable when the pin is configured as McASP pin (PFUNC[n] = 0).
- Pin data set register (PDSET): alias of PDOUT. Writing a 1 to PDSET[n] sets the respective PDOUT[n] to 1. Writing a 0 has no effect. Applicable only when the pin is configured as GPIO output (PFUNC[n] = 1 and PDIR[n] = 1).
- Pin data clear register (PDCLR): alias of PDOUT. Writing a 1 to PDCLR[n] clears the respective PDOUT[n] to 0. Writing a 0 has no effect. Applicable only when the pin is configured as GPIO output (PFUNC[n] = 1 and PDIR[n] = 1).

See the register descriptions in Section 14.3 for details on the mapping of each McASP pin to the register bits. Figure 14-24 shows the pin control block diagram.

Figure 14-24. McASP I/O Pin Control Block Diagram



14.2.7.6.1 McASP Pin Control-Transmit and Receive

You must correctly set the McASP GPIO registers PFUNC and PDIR, even when McASP pins are used for their serial port (non-GPIO) function.

Serial port functions include:

- Clock pins (ACLKX, ACLKR, AHCLKX, AHCLKR, AFSX, AFSR) used as clock inputs and outputs.
- Serializer data pins (AXRn) used to transmit or receive.
- AMUTE used as a mute output signal.

When using these pins in their serial port function, you must clear PFUNC[n] to 0 for each pin.

Also, certain outputs require PDIR[n] = 1, such as clock pins used as clock outputs, serializer data pins used to transmit, and AMUTE used as mute output.

Clock inputs and serializers configured to receive must have PDIR[n] = 0.

PFUNC and PDIR do not control the AMUTEIN signal, it is usually tied to a device level interrupt pin (consult device datasheet). If used as a mute input, this pin needs to be configured as an input in the appropriate peripheral.

Finally, there is an important advantage to having separate control of pin direction (by PDIR), and the choice of internal versus external clocking (by CLKRM/CLKXM). Depending on the specific device and usage, you might select an external clock (CLKRM = 0), while enabling the internal clock divider, and the clock pin as an output in the PDIR register (PDIR[ACLKR] = 1). In this case, the bit clock is an output (PDIR[ACLKR] = 1) and, therefore, routed to the ACLKR pin. However, because CLKRM = 0, the bit clock is then routed back to the McASP module as an "external" clock source. This may result in less skew between the clock inside the McASP and the clock in the external device, thus producing more balanced setup and hold times for a particular system. As a result, this may allow a higher serial clock rate interface.

14.2.8 Operation

This section discusses the operation of the McASP.

14.2.8.1 Data Transmission and Reception

The processor services the McASP by writing data to the XBUF register(s) for transmit operations, and by reading data from the RBUF register(s) for receive operations. The McASP sets status flag and notifies the processor whenever data is ready to be serviced. Section 14.2.8.1.1 discusses data ready status in detail.

The XBUF and RBUF registers can be accessed through one of the two peripheral ports of the device:

- The data port (DAT): This port is dedicated for data transfers on the device.
- The configuration bus (CFG): This port is used for both data transfers and peripheral configuration control on the device.

Section 14.2.8.1.2 and Section 14.2.8.1.3 discuss how to perform transfers through the data port and the configuration bus.

Either the CPU or the DMA can be used to service the McASP through any of these two peripheral ports. The CPU and DMA usages are discussed in Section 14.2.8.1.4 and Section 14.2.12.2.

14.2.8.1.1 Data Ready Status and Event/Interrupt Generation

14.2.8.1.1.1 Transmit Data Ready

The transmit data ready flag XDATA bit in the XSTAT register reflects the status of the XBUF register. The XDATA flag is set when data is transferred from the XRBUF[n] buffers to the XRSR[n] shift registers, indicating that the XBUF is empty and ready to accept new data from the processor. The transmit data ready event is individually indicated per serializer in its corresponding control register SRCTLn[4] XRDY status bit. When this bit is set to 1, it notifies to the host that this serializer Tx buffer must be serviced (written). When XBUFn is written to by the host, the SRCTLn[4] XRDY is deasserted to 0. As XDATA global flag is an OR-event of all active serializers XRDY flags, it indicates to software the moment, when write service operation has to be initiated by the MCASP host (XDATA = 1). The XRDY flags have to be sequentially scanned by user software to determine which serializer TXBUFn register has to be currently written. Once all requested TXBUFn are written, the serializers control XRDY flags are cleared to 0. As a consequence, XDATA flag is deasserted to 0 to indicate to software that write operation is completed for all serializers. The global XDATA flag can be cleared when the XSTAT[5] XDATA bit is written to 1 or once TXBUFn registers of all the serializers, that have previously raised their XRDY flags, are written with corresponding active slot data by the host.

Whenever XDATA is set, an DMA event AXEVT is automatically generated to notify the DMA of the XBUF empty status. An interrupt AXINT is also generated if XDATA interrupt is enabled in the XINTCTL register (See Section 14.2.11.1 for details).

For DMA requests, the McASP does not require XSTAT to be read between DMA events. This means that even if XSTAT already has the XDATA flag set to 1 from a previous request, the next transfer triggers another DMA request.

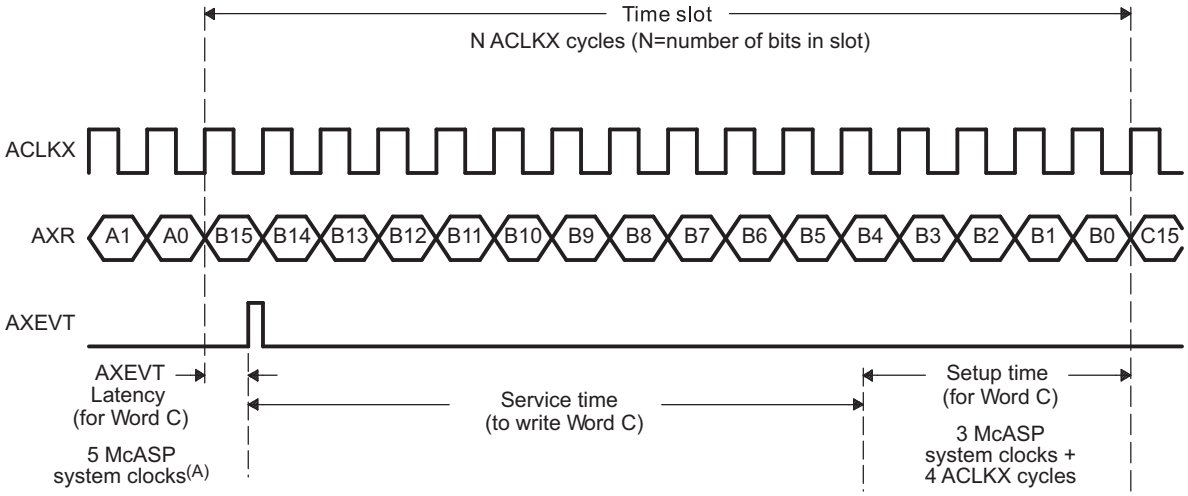
Since all serializers act in lockstep, only one DMA event is generated to indicate that all active transmit serializers are ready to be written to with new data.

Figure 14-25 shows the timing details of when AXEVT is generated at the McASP boundary. In this example, as soon as the last bit (bit A0) of Word A is transmitted, the McASP sets the XDATA flag and generates an AXEVT event. However, it takes up to 5 McASP system clocks (AXEVT Latency) before AXEVT is active at the McASP boundary. Upon AXEVT, the processor can begin servicing the McASP by writing Word C into the XBUF (Processor Service Time). The processor must write Word C into the XBUF no later than the setup time required by the McASP (Setup Time).

The maximum Processor Service Time (Figure 14-25) can be calculated as:

Processor Service Time = Time Slot - AXEVT Latency - Setup Time

Figure 14-25. Processor Service Time Upon Transmit DMA Event (AXEVT)



A Refer to the device-specific data manual for the McASP system clock source. This is not the same as AUXCLK.

Example 14-1. Processor Service Time Calculation for Transmit DMA Event (AXEVT)

The following is an example to show how to calculate Processor Service Time. Assume the following setup:

- Device: C64x+ DSP at 300 MHz.
- McASP transmits in I2S format at 192 kHz frame rate. Assume slot size is 32 bit.

With the above setup, we obtain the following parameters corresponding to Figure 14-25:

- Calculation of McASP system clock cycle:
  - C64x+ DSP uses SYSCLK2 as the McASP system clock. It runs at 150 MHz (half of device frequency).
  - Therefore, McASP system clock cycle = 1/150 MHz = 6.7 ns.
- Calculation of ACLKX clock cycle:
  - This example has two 32-bit slots per frame, for a total of 64 bits per frame.
  - ACLKX clock cycle is (1/192 kHz)/64 = 81.4 ns.
- Time Slot between AXEVT events:
  - For I2S format, McASP generates two AXEVT events per 192 kHz frame.
  - Therefore, Time Slot between AXEVT events is (1/192 kHz)/2 = 2604 ns.
- AXEVT Latency:
  = 5 McASP system clocks
  = 6.7 ns × 5 = 33.5 ns
- Setup Time
  = 3 McASP system clocks + 4 ACLKX cycles
  = (6.7 ns × 3) + (81.4 ns × 4)
  = 345.7 ns
- Processor Service Time
  = Time Slot - AXEVT Latency - Setup Time
  = 2604 ns - 33.5 ns - 345.7 ns
  = 2225 ns

14.2.8.1.1.2 Receive Data Ready

Similarly, the receive data ready flag, RDATA, in the RSTAT register reflects the data ready status of XRBUFn buffers for all of the active slot receiving serializers. The RDATA flag is set whenever data is transferred from a receiving serializer shift register XRSRn to its corresponding XRBUFn data buffer. Thus, the RDATA bit indicates the global event that some of the receivers data buffer, RXBUFn, already contains received data (that is, a buffer is full) and is ready to transfer it to the host (MPU). The receive data ready event is individually indicated per serializer in its corresponding control register SRCTLn [5] RRDY status bit. When this bit is set to 1, it notifies to host that this serializer Rx buffer must be serviced (read). When RXBUF[n] is read from the host, the SRCTLn [5] RRDY is deasserted to 0. As RDATA global flag is an OR-event of all active serializers RRDY flags, it indicates to software the moment, when read service operation has to be initiated by the MCASP host (RDATA = 1). The RRDY flags have to be sequentially scanned by user software to determine which serializer RXBUFn register has to be currently read. Once all requested RXBUFn are read, the serializers control RRDY flags are cleared to 0. As a consequence, RDATA flag is deasserted to 0, to indicate to software that read operation is completed for all n serializers. The global RDATA flag can be cleared when the RSTAT[5] RDATA bit is written to 1 or once RXBUFn registers of all the serializers, that have previously raised their RRDY flags, are read by the host. (See Section 14.2.11.2 for details).

For DMA requests, the McASP does not require RSTAT to be read between DMA events. This means that even if RSTAT already has the RDATA flag set to 1 from a previous request, the next transfer triggers another DMA request.

Since all serializers act in lockstep, only one DMA event is generated to indicate that all active receive serializers are ready to receive new data.

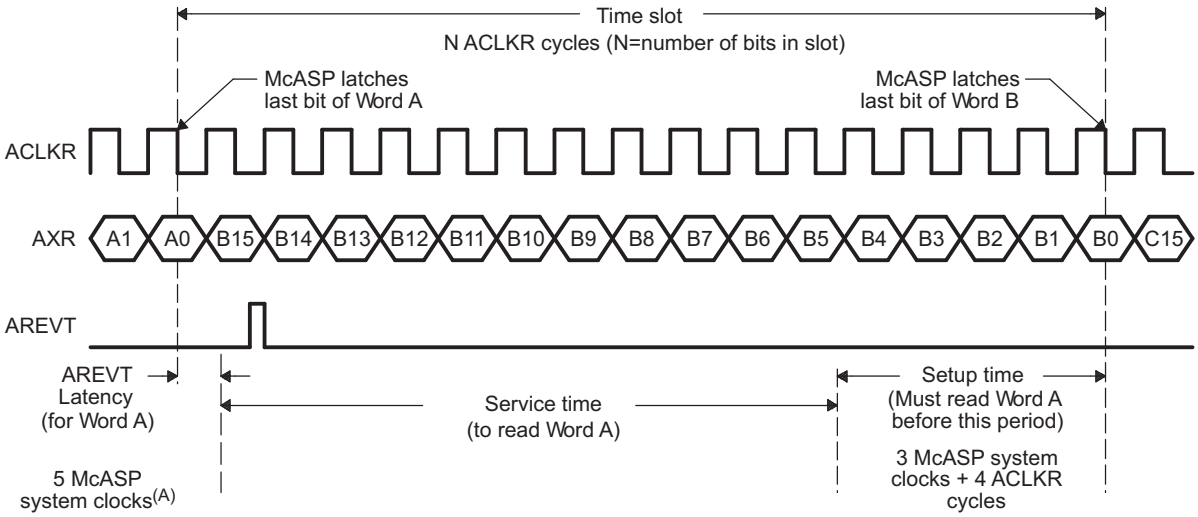
Figure 14-26 shows the timing details of when AREVT is generated at the McASP boundary. In this example, as soon as the last bit (bit A0) of Word A is received, the McASP sets the RDATA flag and generates an AREVT event. However, it takes up to 5 McASP system clocks (AREVT Latency) before AREVT is active at the McASP boundary. Upon AREVT, the processor can begin servicing the McASP by reading Word A from the RBUF (Processor Service Time). The processor must read Word A from the XBUF no later than the setup time required by the McASP (Setup Time).

The maximum Processor Service Time (Figure 14-26) can be calculated as:

Processor Service Time = Time Slot - AREVT Latency - Setup Time

The Processor Service Time calculation for receive is similar to the calculation for transmit. See Example 14-1 for Processor Service Time calculation using transmit as an example.

Figure 14-26. Processor Service Time Upon Receive DMA Event (AREVT)



A The device uses SYSCLK2 as the McASP system clock source.

14.2.8.1.2 Transfers Through the Data Port (DAT)

NOTE: The data port (DAT) addresses are located in L3 and L4 tables in the device data sheet for the appropriate data peripheral registers. To perform internal transfers through the data port, clear XBUSEL/RBUSEL bit to 0 in the respective XFMT/RFMT registers. Failure to do so will result in software malfunction.

Typically, you will access the McASP XRBUF registers through the data port. To access through the data port, simply have the CPU or DMA access the XRBUF through its data port location. Refer to the device-specific data manual for the exact memory address. Through the data port, the DMA/CPU can service all the serializers through a single address. The McASP automatically cycles through the appropriate serializers.

For transmit operations through the data port, the DMA/CPU should write to the same XBUF data port address to service all of the active transmit serializers. In addition, the DMA/CPU should write to the XBUF for all active transmit serializers in incremental (although not necessarily consecutive) order. For example, if serializers 0, 4, and 5, are set up as active transmitters, the DMA/CPU should write to the XBUF data port address four times with data for serializers 0, 4, and 5, upon each transmit data ready event. This exact servicing order must be followed so that data appears in the appropriate serializers.

Similarly, for receive operations through the data port, the DMA/CPU should read from the same RBUF data port address to service all of the active receive serializers. In addition, reads from the active receive serializers through the data port return data in incremental (although not necessarily consecutive) order. For example, if serializers 1, 2, and 3, are set up as active receivers, the DMA/CPU should read from the RBUF data port address four times to obtain data for serializers 1, 2, and 3, in this exact order, upon each receive data ready event.

When transmitting, the DMA/CPU must write data to each serializer configured as "active" (active slot in XTDM register) and "transmit" (Tx enabled in SCRCTLn register) within each time slot. Failure to do so results in a buffer underrun condition (Section 14.2.8.4.2). Similarly, when receiving, data must be read from each serializer configured as "active" (active slot in RTDM register) and "receive" (Rx enabled in SCRCTLn register) within each time slot. Failure to do results in a buffer overrun condition (Section 14.2.8.4.3).

To perform internal transfers through the data port, clear XBUSEL/RBUSEL bit to 0 in the respective XFMT/RFMT registers.

14.2.8.1.3 Transfers Through the Configuration Bus (CFG)

NOTE: To perform internal transfers through the configuration bus, set XBUSEL/RBUSEL bit to 1 in the respective XFMT/RFMT registers. Failure to do so will result in software malfunction.

In this method, the DMA/CPU accesses the XRBUF registers through the configuration bus address. The exact XRBUF register address for any particular serializer is determined by adding the offset for that particular serializer to the base address for the particular McASP (found in the device-specific data manual). XRBUF for the serializers configured as transmitters is given the name XBUF*n*. For example, the XRBUF associated with transmit serializer 2 is named XBUF2. Similarly, XRBUF for the serializers configured as receivers is given the name RBUF*n*.

Accessing the XRBUF registers through the data port is different because the CPU/DMA only needs to access one single address. When accessing through the configuration bus, the CPU/DMA must provide the exact XBUF*n* or RBUF*n* address for each access.

When transmitting, DMA/CPU must write data to each serializer configured as "active" and "transmit" within each time slot. Failure to do so results in a buffer underrun condition (Section 14.2.8.4.2). Similarly when receiving, data must be read from each serializer configured as "active" and "receive" within each time slot. Failure to do results in a buffer overrun condition (Section 14.2.8.4.3).

To perform internal transfers through the configuration bus, set XBUSEL/RBUSEL bit to 1 in the respective XFMT/RFMT registers.

14.2.8.1.4 Using the CPU for McASP Servicing

The CPU can be used to service the McASP through interrupt (upon AXINT/ARINT interrupts) or through polling the XDATA bit in the XSTAT register and the RDATA bit in the RSTAT register. As discussed in Section 14.2.8.1.2 and Section 14.2.8.1.3, the CPU can access XRBUF serializer buffer through either the data port or through the configuration bus.

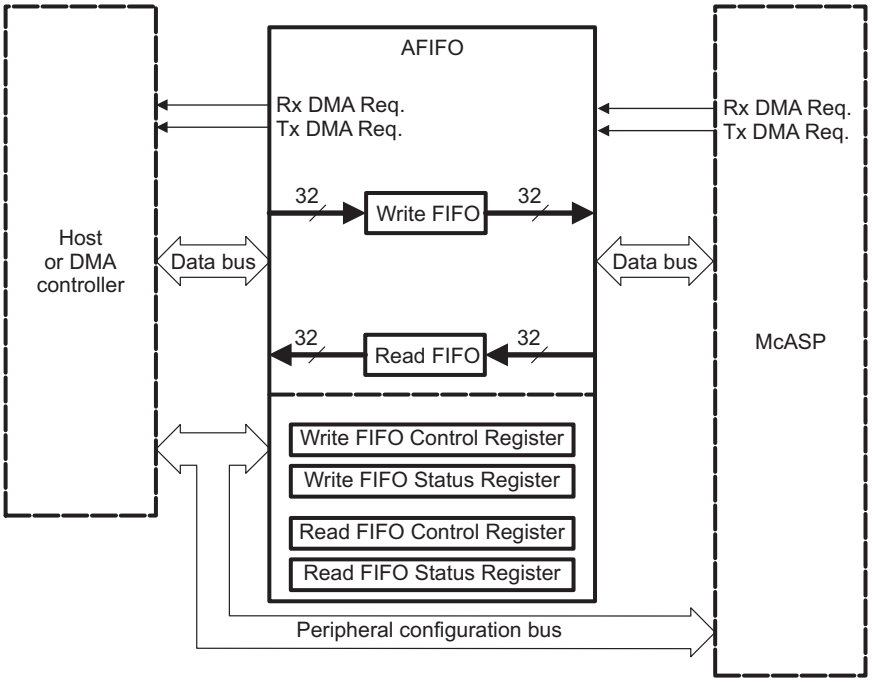
To use the CPU to service the McASP through interrupts, the XDATA/RDATA bit must be enabled in the respective XINTCTL/RINTCTL registers, to generate interrupts AXINT/ARINT to the CPU upon data ready.

14.2.8.2 McASP Audio FIFO (AFIFO)

The AFIFO contains two FIFOs: one Read FIFO (RFIFO), and one Write FIFO (WFIFO). To ensure backward compatibility with existing software, both the Read and Write FIFOs are disabled by default. See Figure 14-27 for a high-level block diagram of the AFIFO.

The AFIFO may be enabled/disabled and configured via the WFIFOCTL and RFIFOCTL registers. Note that if the Read or Write FIFO is to be enabled, it must be enabled prior to initializing the receive/transmit section of the McASP (see Section 14.2.10.2 for details).

Figure 14-27. McASP Audio FIFO (AFIFO) Block Diagram



14.2.8.2.1 AFIFO Data Transmission

When the Write FIFO is disabled, transmit DMA requests pass through directly from the McASP to the host/DMA controller. Whether the WFIFO is enabled or disabled, the McASP generates transmit DMA requests as needed; the AFIFO is “invisible” to the McASP.

When the Write FIFO is enabled, transmit DMA requests from the McASP are sent to the AFIFO, which in turn generates transmit DMA requests to the host/DMA controller.

If the Write FIFO is enabled, upon a transmit DMA request from the McASP, the WFIFO writes *WNUMDMA* 32-bit words to the McASP if and when there are at least *WNUMDMA* words in the Write FIFO. If there are not, the WFIFO waits until this condition has been satisfied. At that point, it writes *WNUMDMA* words to the McASP. (See description for WFIFOCTL.WNUMDMA in Section 14.3.43.)

If the host CPU writes to the Write FIFO, independent of a transmit DMA request, the WFIFO will accept host writes until full. After this point, excess data will be discarded.

Note that when the WFIFO is first enabled, it will immediately issue a transmit DMA request to the host. This is because it begins in an empty state, and is therefore ready to accept data.

14.2.8.2.1.1 Transmit DMA Event Pacer

The AFIFO may be configured to delay making a transmit DMA request to the host until the Write FIFO has enough space for a specified number of words. In this situation, the number of transmit DMA requests to the host or DMA controller is reduced.

If the Write FIFO has space to accept *WNUMEVT* 32-bit words, it generates a transmit DMA request to the host and then waits for a response. Once *WNUMEVT* words have been written to the FIFO, it checks again to see if there is space for *WNUMEVT* 32-bit words. If there is space, it generates another transmit DMA request to the host, and so on. In this fashion, the Write FIFO will attempt to stay filled.

Note that if transmit DMA event pacing is desired, WFIFOCTL.WNUMEVT should be set to a non-zero integer multiple of the value in WFIFOCTL.WNUMDMA. If transmit DMA event pacing is not desired, then the value in WFIFOCTL.WNUMEVT should be set equal to the value in WFIFOCTL.WNUMDMA.

14.2.8.2.2 AFIFO Data Reception

When the Read FIFO is disabled, receive DMA requests pass through directly from McASP to the host/DMA controller. Whether the RFIFO is enabled or disabled, the McASP generates receive DMA requests as needed; the AFIFO is “invisible” to the McASP.

When the Read FIFO is enabled, receive DMA requests from the McASP are sent to the AFIFO, which in turn generates receive DMA requests to the host/DMA controller.

If the Read FIFO is enabled and the McASP makes a receive DMA request, the RFIFO reads *RNUMDMA* 32-bit words from the McASP, if and when the RFIFO has space for *RNUMDMA* words. If it does not, the RFIFO waits until this condition has been satisfied; at that point, it reads *RNUMDMA* words from the McASP. (See description for RFIFOCTL.RNUMDMA in Section 14.3.45.)

If the host CPU reads the Read FIFO, independent of a receive DMA request, and the RFIFO at that time contains less than *RNUMEVT* words, those words will be read correctly, emptying the FIFO.

14.2.8.2.2.1 Receive DMA Event Pacer

The AFIFO may be configured to delay making a receive DMA request to the host until the Read FIFO contains a specified number of words. In this situation, the number of receive DMA requests to the host or DMA controller is reduced.

If the Read FIFO contains at least *RNUMEVT* 32-bit words, it generates a receive DMA request to the host and then waits for a response. Once *RNUMEVT* 32-bit words have been read from the RFIFO, the RFIFO checks again to see if it contains at least another *RNUMEVT* words. If it does, it generates another receive DMA request to the host, and so on. In this fashion, the Read FIFO will attempt to stay empty.

Note that if receive DMA event pacing is desired, RFIFOCTL.RNUMEVT should be set to a non-zero integer multiple of the value in RFIFOCTL.RNUMDMA. If receive DMA event pacing is not desired, then the value in RFIFOCTL.RNUMEVT should be set equal to the value in RFIFOCTL.RNUMDMA.

14.2.8.2.3 Arbitration Between Transmit and Receive DMA Requests

If both the WFIFO and the RFIFO are enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete.

If only the WFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete.

If only the RFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the receive DMA request. Once a transfer is in progress, it is allowed to complete.

14.2.8.3 Formatter

14.2.8.3.1 Transmit Bit Stream Data Alignment

The McASP transmitter supports serial formats of:

- Slot or time slot size = 8, 12, 16, 20, 24, 28, 32 bits.
- Word size ≤ Slot size.
- Alignment when more bits/slot than bits/words, then:
  - Left aligned = word shifted first, remaining bits are pad.
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot.
- Order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB.
  - LSB: least-significant bit of word is shifted out last, last bit is MSB.

Hardware support for these serial formats comes from the programmable options in the transmit bitstream format register (XFMT):

- XRVRS: bit reverse (1) or no bit reverse (0).
- XROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits.
- XSSZ: transmit slot size of 8, 12, 16, 20, 24, 28, or 32 bits.

XSSZ should always be programmed to match the slot size of the serial stream. The word size is not directly programmed into the McASP, but rather is used to determine the rotation needed in the XROT field.

Table 14-5 and Figure 14-28 show the XRVRS and XROT fields for each serial format and for both integer and Q31 fractional internal representations.

This discussion assumes that all slot size (SLOT in Table 14-5) and word size (WORD in Table 14-5) options are multiples of 4, since the transmit rotate right unit only supports rotation by multiples of 4. However, the bit mask/pad unit does allow for any number of significant digits. For example, a Q31 number may have 19 significant digits (word) and be transmitted in a 24-bit slot; this would be formatted as a word size of 20 bits and a slot size of 24 bits. However, it is possible to set the bit mask unit to only pass the 19 most-significant digits (program the mask value to FFFF E000h). The digits that are not significant can be set to a selected pad value, which can be any one of the significant digits, a fixed value of 0, or a fixed value of 1.

The transmit bit mask/pad unit operates on data as an initial step of the transmit format unit (see Figure 14-23), and the data is aligned in the same representation as it is written to the transmitter by the processor (typically Q31 or integer).

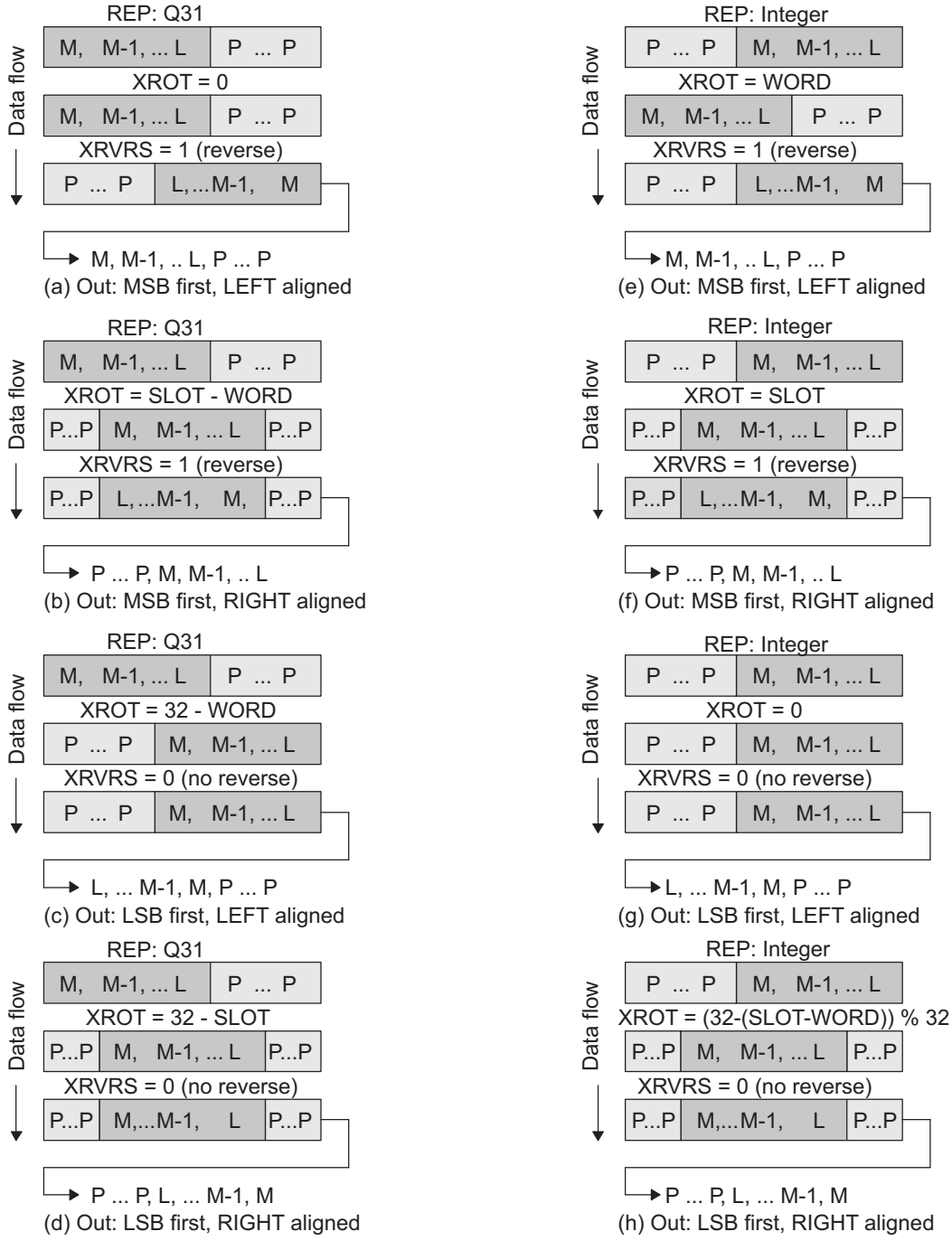
Table 14-5. Transmit Bitstream Data Alignment

Figure 14-28	Bit Stream Order	Bit Stream Alignment	Internal Numeric Representation	XFMT Bit	
				XROT[1]	XRVRS
(a)[2]	MSB first	Left aligned	Q31 fraction	0	1
(b)	MSB first	Right aligned	Q31 fraction	SLOT - WORD	1
(c)	LSB first	Left aligned	Q31 fraction	32 - WORD	0
(d)	LSB first	Right aligned	Q31 fraction	32 - SLOT	0
(e)[2]	MSB first	Left aligned	Integer	WORD	1
(f)	MSB first	Right aligned	Integer	SLOT	1
(g)	LSB first	Left aligned	Integer	0	0
(h)	LSB first	Right aligned	Integer	(32 - (SLOT - WORD)) % 32	0

(1) WORD = Word size rounded up to the nearest multiple of 4; SLOT = slot size; % = modulo operator

(2) To transmit in I2S format, use MSB first, left aligned, and also select XDATDLY = 01 (1 bit delay)

Figure 14-28. Data Flow Through Transmit Format Unit, Illustrated



14.2.8.3.2 Receive Bit Stream Data Alignment

The McASP receiver supports serial formats of:

- Slot or time slot size = 8, 12, 16, 20, 24, 28, 32 bits.
- Word size ≤ Slot size.
- Alignment when more bits/slot than bits/words, then:
  - Left aligned = word shifted first, remaining bits are pad.
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot.
- Order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB.
  - LSB: least-significant bit of word is shifted out last, last bit is MSB.

Hardware support for these serial formats comes from the programmable options in the receive bitstream format register (RFMT):

- RRVRS: bit reverse (1) or no bit reverse (0).
- RROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits.
- RSSZ: receive slot size of 8, 12, 16, 20, 24, 28, or 32 bits.

RSSZ should always be programmed to match the slot size of the serial stream. The word size is not directly programmed into the McASP, but rather is used to determine the rotation needed in the RROT field.

Table 14-6 and Figure 14-29 show the RRVRS and RROT fields for each serial format and for both integer and Q31 fractional internal representations.

This discussion assumes that all slot size and word size options are multiples of 4; since the receive rotate right unit only supports rotation by multiples of 4. However, the bit mask/pad unit does allow for any number of significant digits. For example, a Q31 number may have 19 significant digits (word) and be transmitted in a 24-bit slot; this would be formatted as a word size of 20 bits and a slot size of 24 bits. However, it is possible to set the bit mask unit to only pass the 19 most-significant digits (program the mask value to FFFF E000h). The digits that are not significant can be set to a selected pad value, which can be any one of the significant digits, a fixed value of 0, or a fixed value of 1.

The receive bit mask/pad unit operates on data as the final step of the receive format unit (see Figure 14-22), and the data is aligned in the same representation as it is read from the receiver by the processor (typically Q31 or integer).

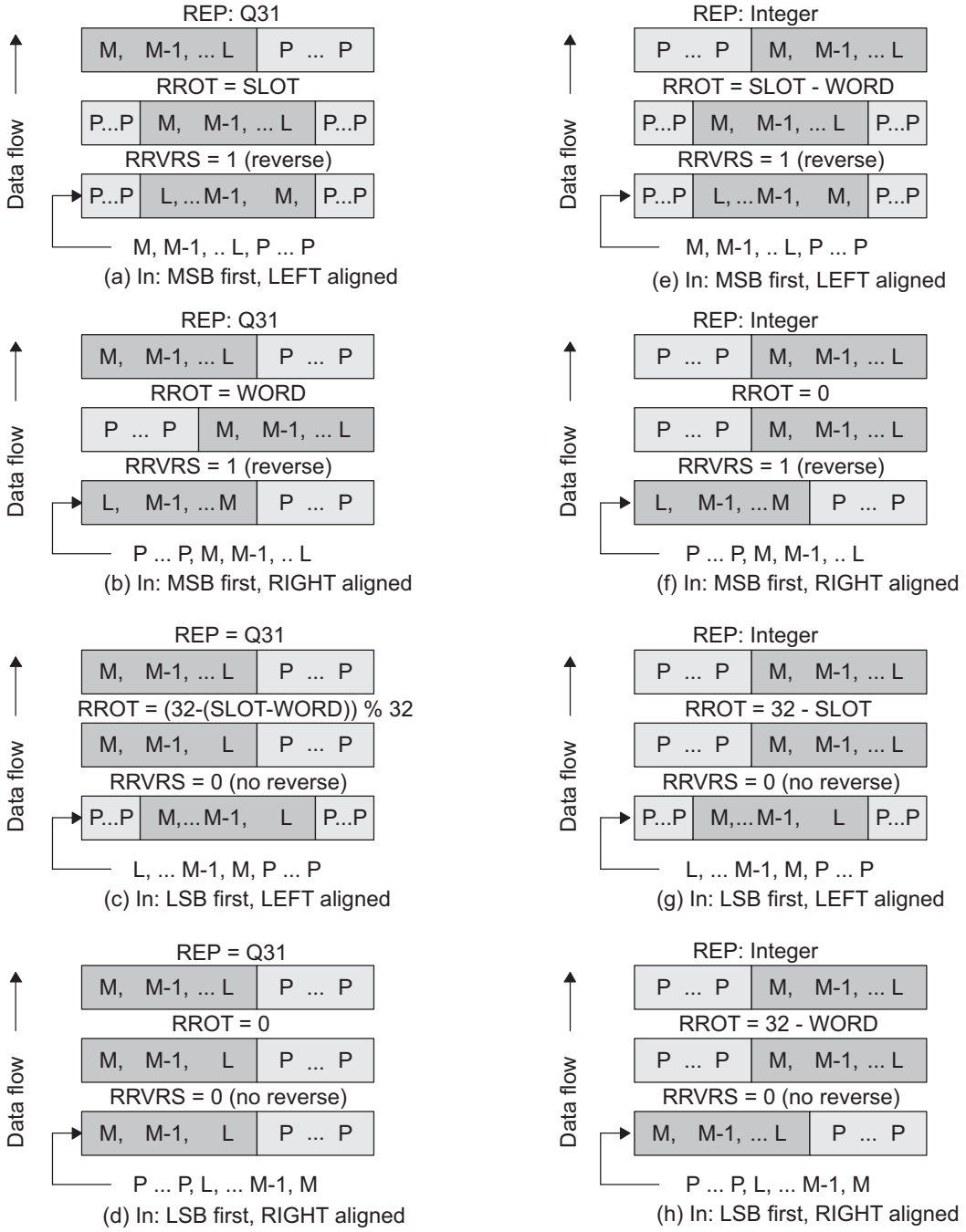
Table 14-6. Receive Bitstream Data Alignment

Figure 14-29	Bit Stream Order	Bit Stream Alignment	Internal Numeric Representation	RFMT Bit	
				RROT[1]	RRVRS
(a)[2]	MSB first	Left aligned	Q31 fraction	SLOT	1
(b)	MSB first	Right aligned	Q31 fraction	WORD	1
(c)	LSB first	Left aligned	Q31 fraction	(32 - (SLOT - WORD)) % 32	0
(d)	LSB first	Right aligned	Q31 fraction	0	0
(e)[2]	MSB first	Left aligned	Integer	SLOT - WORD	1
(f)	MSB first	Right aligned	Integer	0	1
(g)	LSB first	Left aligned	Integer	32 - SLOT	0
(h)	LSB first	Right aligned	Integer	32 - WORD	0

(1) WORD = Word size rounded up to the nearest multiple of 4; SLOT = slot size; % = modulo operator

(2) To transmit in I2S format, select MSB first, left aligned, and also select RDATDLY = 01 (1 bit delay)

Figure 14-29. Data Flow Through Receive Format Unit, Illustrated



14.2.8.4 Error Handling and Management

To support the design of a robust audio system, the McASP includes error-checking capability for the serial protocol, data underrun, and data overrun. In addition, the McASP includes a timer that continually measures the high-frequency master clock every 32 AHCLKX/AHCLKR clock cycles. The timer value can be read to get a measurement of the clock frequency and has a minimum and maximum range setting that can set an error flag if the master clock goes out of a specified range.

Upon the detection of any one or more errors (software selectable), or the assertion of the AMUTEIN input pin, the AMUTE output pin may be asserted to a high or low level to immediately mute the audio output. In addition, an interrupt may be generated if desired, based on any one or more of the error sources.

14.2.8.4.1 Unexpected Frame Sync Error

An unexpected frame sync occurs when:

- In burst mode, when the next active edge of the frame sync occurs early such that the current slot will not be completed by the time the next slot is scheduled to begin.
- In TDM mode, a further constraint is that the frame sync must occur exactly during the correct bit clock (not a cycle earlier or later) and only before slot 0. An unexpected frame sync occurs if this condition is not met.

When an unexpected frame sync occurs, there are two possible actions depending upon when the unexpected frame sync occurs:

1. Early: An early unexpected frame sync occurs when the McASP is in the process of completing the current frame and a new frame sync is detected (not including overlap that occurs due to a 1 or 2 bit frame sync delay). When an early unexpected frame sync occurs:
   - Error interrupt flag is set (XSYNCERR, if an unexpected transmit frame sync occurs; RSYNCERR, if an unexpected receive frame sync occurs).
   - Current frame is not resynchronized. The number of bits in the current frame is completed. The next frame sync, which occurs after the current frame is completed, will be resynchronized.
2. Late: A late unexpected frame sync occurs when there is a gap or delay between the last bit of the previous frame and the first bit of the next frame. When a late unexpected frame sync occurs (as soon as the gap is detected):
   - Error interrupt flag is set (XSYNCERR, if an unexpected transmit frame sync occurs; RSYNCERR, if an unexpected receive frame sync occurs).
   - Resynchronization occurs upon the arrival of the next frame sync.

Late frame sync is detected the same way in both burst mode and TDM mode; however, in burst mode, late frame sync is not meaningful and its interrupt enable should not be set.

14.2.8.4.2 Buffer Underrun Error - Transmitter

A buffer underrun can only occur for serializers programmed to be transmitters. A buffer underrun occurs when the serializer is instructed by the transmit state machine to transfer data from XRBUF[n] to XRSR[n], but XRBUF[n] has not yet been written with new data since the last time the transfer occurred. When this occurs, the transmit state machine sets the XUNDRN flag.

An underrun is checked only once per time slot. The XUNDRN flag is set when an underrun condition occurs. Once set, the XUNDRN flag remains set until the processor explicitly writes a 1 to the XUNDRN bit to clear the XUNDRN bit.

In DIT mode, a pair of BMC zeros is shifted out when an underrun occurs (four bit times at 128 × fs). By shifting out a pair of zeros, a clock may be recovered on the receiver. To recover, reset the McASP and start again with the proper initialization.

In TDM mode, during an underrun case, a long stream of zeros are shifted out causing the DACs to mute. To recover, reset the McASP and start again with the proper initialization.

14.2.8.4.3 Buffer Overrun Error - Receiver

A buffer overrun can only occur for serializers programmed to be receivers. A buffer overrun occurs when the serializer is instructed to transfer data from XRSR[n] to XRBUF[n], but XRBUF[n] has not yet been read by either the DMA or the processor. When this occurs, the receiver state machine sets the ROVRN flag. However, the individual serializer writes over the data in the XRBUF[n] register (destroying the previous sample) and continues shifting.

An overrun is checked only once per time slot. The ROVRN flag is set when an overrun condition occurs. It is possible that an overrun occurs on one time slot but then the processor catches up and does not cause an overrun on the following time slots. However, once the ROVRN flag is set, it remains set until the processor explicitly writes a 1 to the ROVRN bit to clear the ROVRN bit.

14.2.8.4.4 DMA Error - Transmitter

A transmit DMA error, as indicated by the XDMAERR flag in the XSTAT register, occurs when the DMA (or CPU) writes more words to the DAT port of the McASP than it should. For each DMA event, the DMA should write exactly as many words as there are serializers enabled as transmitters.

XDMAERR indicates that the DMA (or CPU) wrote too many words to the McASP for a given transmit DMA event. Writing too few words results in a transmit underrun error setting XUNDRN in XSTAT.

While XDMAERR occurs infrequently, an occurrence indicates a serious loss of synchronization between the McASP and the DMA or CPU. You should reinitialize both the McASP transmitter and the DMA to resynchronize them.

14.2.8.4.5 DMA Error - Receiver

A receive DMA error, as indicated by the RDMAERR flag in the RSTAT register, occurs when the DMA (or CPU) reads more words from the DAT port of the McASP than it should. For each DMA event, the DMA should read exactly as many words as there are serializers enabled as receivers.

RDMAERR indicates that the DMA (or CPU) read too many words from the McASP for a given receive DMA event. Reading too few words results in a receiver overrun error setting ROVRN in RSTAT.

While RDMAERR occurs infrequently, an occurrence indicates a serious loss of synchronization between the McASP and the DMA or CPU. You should reinitialize both the McASP receiver and the DMA to resynchronize them.

14.2.8.4.6 Clock Failure Detection

14.2.8.4.6.1 Clock-Failure Check Startup

It is expected, initially, that the clock-failure circuits will generate an error until at least one measurement has been taken. Therefore, the clock failure interrupts, clock switch, and mute functions should not immediately be enabled, but be enabled only after a specific startup procedure. The startup procedure is:

1. For the transmit clock failure check:
   (a) Configure transmit clock failure detect logic (XMIN, XMAX, XPS) in the transmit clock check control register (XCLKCHK).
   (b) Clear transmit clock failure flag (XCKFAIL) in the transmit status register (XSTAT).
   (c) Wait until first measurement is taken (> 32 AHCLKX clock periods).
   (d) Verify no clock failure is detected.
   (e) Repeat steps b–d until clock is running and is no longer issuing clock failure errors.
   (f) After the transmit clock is measured and falls within the acceptable range, the following may be enabled:
      (i) transmit clock failure interrupt enable bit (XCKFAIL) in the transmitter interrupt control register (XINTCTL).
      (ii) mute option (XCKFAIL) in the mute control register (AMUTE).
2. For the receive clock failure check:
   (a) Configure receive clock failure detect logic (RMIN, RMAX, RPS) in the receive clock check control register (RCLKCHK).
   (b) Clear receive clock failure flag (RCKFAIL) in the receive status register (RSTAT).
   (c) Wait until first measurement is taken (> 32 AHCLKR clock periods).
   (d) Verify no clock failure is detected.
   (e) Repeat steps b–d until clock is running and is no longer issuing clock failure errors.
   (f) After the receive clock is measured and falls within the acceptable range, the following may be enabled:
      (i) receive clock failure interrupt enable bit (RCKFAIL) in the receiver interrupt control register (RINTCTL).
      (ii) mute option (RCKFAIL) in the mute control register (AMUTE).

14.2.8.4.6.2 Transmit Clock Failure Check and Recovery

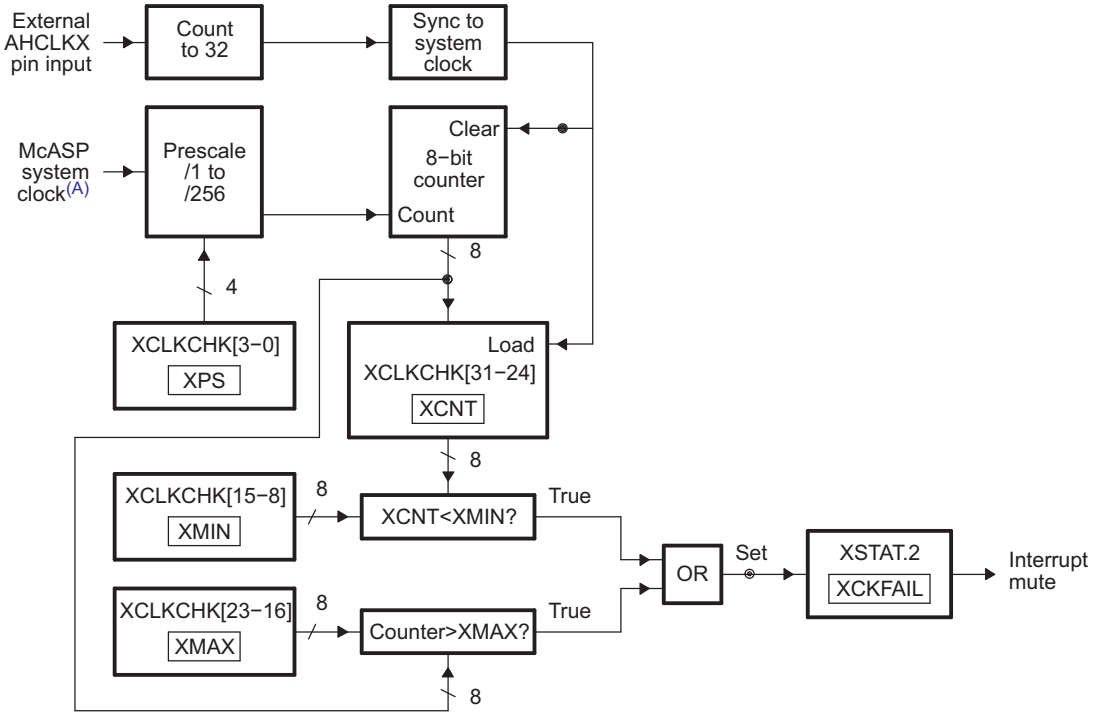
The transmit clock failure check circuit (Figure 14-30) works off both the internal McASP system clock and the external high-frequency serial clock (AHCLKX). It continually counts the number of system clocks for every 32 high rate serial clock (AHCLKX) periods, and stores the count in XCNT of the transmit clock check control register (XCLKCHK) every 32 high rate serial clock cycles.

The logic compares the count against a user-defined minimum allowable boundary (XMIN), and automatically flags an interrupt (XCKFAIL in XSTST) when an out-of-range condition occurs. An out-of-range minimum condition occurs when the count is smaller than XMIN. The logic continually compares the current count (from the running system clock counter) against the maximum allowable boundary (XMAX). This is in case the external clock completely stops, so that the counter value is not copied to XCNT. An out-of-range maximum condition occurs when the count is greater than XMAX. Note that the XMIN and XMAX fields are 8-bit unsigned values, and the comparison is performed using unsigned arithmetic.

An out-of-range count may indicate either that an unstable clock was detected, or that the audio source has changed and a new sample rate is being used.

In order for the transmit clock failure check circuit to operate correctly, the high-frequency serial clock divider must be taken out of reset regardless if AHCLKX is internally generated or externally sourced.

Figure 14-30. Transmit Clock Failure Detection Circuit Block Diagram



A Refer to the device data manual for the McASP system clock source. This is not the same as AUXCLK.

If a clock failure is detected, the transmit clock failure flag (XCKFAIL) in XSTAT is set. This causes an interrupt if the transmit clock failure interrupt enable bit (XCKFAIL) in XINTCTL is set.

14.2.8.4.6.3 Receive Clock Failure Check and Recovery

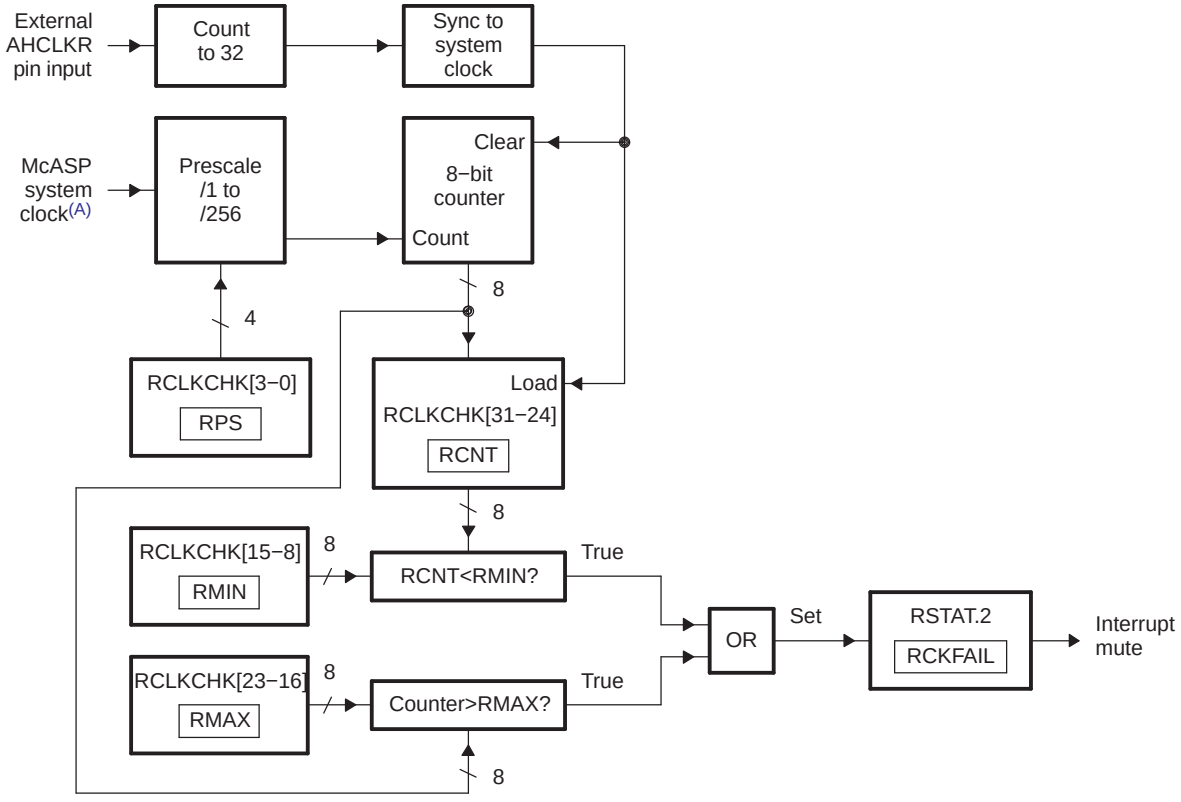
The receive clock failure check circuit (Figure 14-31) works off both the internal McASP system clock and the external high-frequency serial clock (AHCLKR). It continually counts the number of system clocks for every 32 high rate serial clock (AHCLKR) periods, and stores the count in RCNT of the receive clock check control register (RCLKCHK) every 32 high rate serial clock cycles.

The logic compares the count against a user-defined minimum allowable boundary (RMIN) and automatically flags an interrupt (RCKFAIL in RSTAT) when an out-of-range condition occurs. An out-of-range minimum condition occurs when the count is smaller than RMIN. The logic continually compares the current count (from the running system clock counter) against the maximum allowable boundary (RMAX). This is in case the external clock completely stops, so that the counter value is not copied to RCNT. An out-of-range maximum condition occurs when the count is greater than RMAX. Note that the RMIN and RMAX fields are 8-bit unsigned values, and the comparison is performed using unsigned arithmetic.

An out-of-range count may indicate either that an unstable clock was detected or that the audio source has changed and a new sample rate is being used.

In order for the receive clock failure check circuit to operate correctly, the high-frequency serial clock divider must be taken out of reset regardless if AHCLKR is internally generated or externally sourced.

Figure 14-31. Receive Clock Failure Detection Circuit Block Diagram



A Refer to device data manual for the McASP system clock source. This is not the same as AUXCLK.

14.2.8.5 Loopback Modes

The McASP features a digital loopback mode (DLB) that allows testing of the McASP code in TDM mode with a single processor device. In loopback mode, output of the transmit serializers is connected internally to the input of the receive serializers. Therefore, you can check the receive data against the transmit data to ensure that the McASP settings are correct. Digital loopback mode applies to TDM mode only (2 to 32 slots in a frame). It does not apply to DIT mode (XMOD = 180h) or burst mode (XMOD = 0).

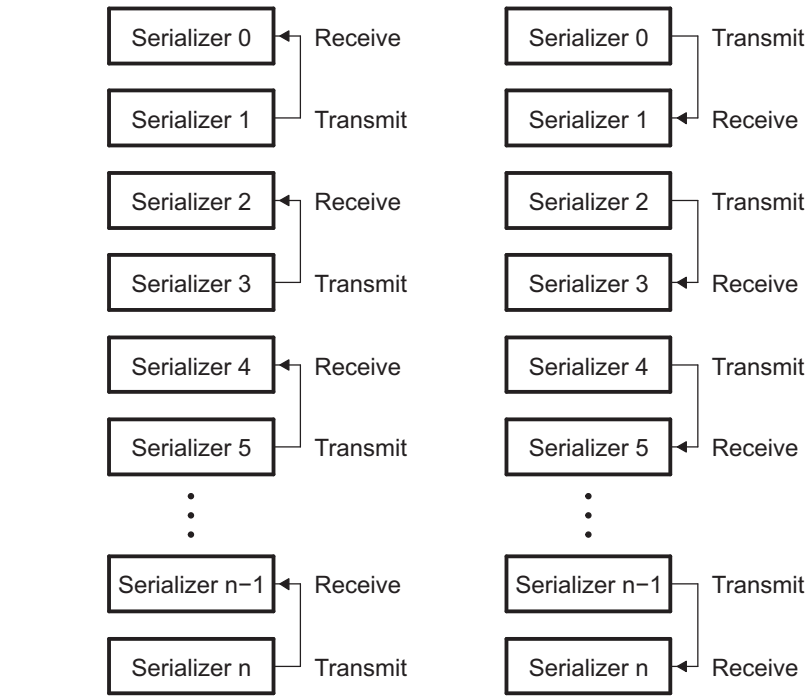
Figure 14-32 shows the basic logical connection of the serializers in loopback mode. Two types of loopback connections are possible, selected by the ORD bit in the digital loopback control register (DLBCTL) as follows:

- ORD = 0: Outputs of odd serializers are connected to inputs of even serializers. If this mode is selected, you should configure odd serializers to be transmitters and even serializers to be receivers.
- ORD = 1: Outputs of even serializers are connected to inputs of odd serializers. If this mode is selected, you should configure even serializers to be transmitters and odd serializers to be receivers.

Data can be externally visible at the I/O pin of the transmit serializer if the pin is configured as a McASP output pin by setting the corresponding PFUNC bit to 0 and PDIR bit to 1.

In loopback mode, the transmit clock and frame sync are used by both the transmit and receive sections of the McASP. The transmit and receive sections operate synchronously. This is achieved by setting the MODE bit of the DLBCTL register to 01b and the ASYNC bit of the ACLKXCTL register to 0.

Figure 14-32. Serializers in Loopback Mode



(a) DLBEN = 1 (loopback enabled) and ORD = 0 (even receive, odd transmit)

(b) DLBEN = 1 (loopback enabled) and ORD = 1 (odd receive, even transmit)

14.2.8.5.1 Loopback Mode Configurations

This is a summary of the settings required for digital loopback mode for TDM format:

- The DLBEN bit in DLBCTL must be set to 1 to enable loopback mode.
- The MODE bits in DLBCTL must be set to 01b for both the transmit and receive sections to use the transmit clock and frame sync generator.
- The ORD bit in DLBCTL must be programmed appropriately to select odd or even serializers to be transmitters or receivers. The corresponding serializers must be configured accordingly.
- The ASYNC bit in ACLKXCTL must be cleared to 0 to ensure synchronous transmit and receive operations.
- RMOD field in AFSRCTL and XMOD field in AFSXCTL must be set to 2h to 20h to indicate TDM mode. Loopback mode does not apply to DIT or burst mode.

14.2.9 Reset Considerations

The McASP has two reset sources: software reset and hardware reset.

14.2.9.1 Software Reset Considerations

The transmitter and receiver portions of the McASP may be put in reset through the global control register (GBLCTL). Note that a valid serial clock must be supplied to the desired portion of the McASP (transmit and/or receive) in order to assert the software reset bits in GBLCTL. see Section 14.2.10.2for details on how to ensure reset has occurred.

The entire McASP module may also be reset through the Power and Sleep Controller (PSC). Note that from the McASP perspective, this reset appears as a hardware reset to the entire module.

14.2.9.2 Hardware Reset Considerations

When the McASP is reset due to device reset, the entire serial port (including the transmitter and receiver state machines, and other registers) is reset.

14.2.10 Setup and Initialization

This section discusses steps necessary to use the McASP module.

14.2.10.1 Considerations When Using a McASP

The following is a list of things to be considered for systems using a McASP:

14.2.10.1.1 Clocks

For each receive and transmit section:

- External or internal generated bit clock and high frequency clock?
- If internally generated, what is the bit clock speed and the high frequency clock speed?
- Clock polarity?
- External or internal generated frame sync?
- If internally generated, what is frame sync speed?
- Frame sync polarity?
- Frame sync width?
- Transmit and receive sync or asynchronous?

14.2.10.1.2 Data Pins

For each pin of each McASP:

- McASP or GPIO?
- Input or output?

14.2.10.1.3 Data Format

For each transmit and receive data:

- Internal numeric representation (integer, Q31 fraction)?
- I2S or DIT (transmit only)?
- Time slot delay (0, 1, or 2 bit)?
- Alignment (left or right)?
- Order (MSB first, LSB first)?
- Pad (if yes, pad with what value)?
- Slot size?
- Rotate?
- Mask?

14.2.10.1.4 Data Transfers

- Internal: DMA or CPU?
- External: TDM or burst?
- Bus: configuration bus (CFG) or data port (DAT)?

14.2.10.2 Transmit/Receive Section Initialization

You must follow the following steps to properly configure the McASP. If external clocks are used, they should be present prior to the following initialization steps.

1. Reset McASP to default values by setting GBLCTL = 0. Poll the bits to ensure that the active reset value (0) is successfully latched into the GBLCTL register.
2. Configure all McASP registers except GBLCTL in the following order:
   (a) Receive registers: RMASK, RFMT, AFSRCTL, ACLKRCTL, AHCLKRCTL, RTDM, RINTCTL, RCLKCHK. If external clocks AHCLKR and/or ACLKR are used, they must be running already for proper synchronization of the GBLCTL register.
   (b) Transmit registers: XMASK, XFMT, AFSXCTL, ACLKXCTL, AHCLKXCTL, XTDM, XINTCTL, XCLKCHK. If external clocks AHCLKX and/or ACLKX are used, they must be running already for proper synchronization of the GBLCTL register.
   (c) Serializer registers: SRCTL[n].
   (d) Global registers: Registers PFUNC, PDIR, DITCTL, DLBCTL, AMUTE. Note that PDIR should only be programmed after the clocks and frames are set up in the steps above. This is because the moment a clock pin is configured as an output in PDIR, the clock pin starts toggling at the rate defined in the corresponding clock control register. Therefore you must ensure that the clock control register is configured appropriately before you set the pin to be an output. A similar argument applies to the frame sync pins. Also note that the reset state for the transmit high-frequency clock divide register (HCLKXDIV) is divide-by-1, and the divide-by-1 clocks are not gated by the transmit high-frequency clock divider reset enable (XHCLKRST).
   (e) DIT registers: For DIT mode operation, set up registers DITCSRA[n], DITCSRB[n], DITUDRA[n], and DITUDRB[n].
3. Start the respective high-frequency serial clocks AHCLKX and/or AHCLKR. This step is necessary even if external high-frequency serial clocks are used:
   (a) Take the respective internal high-frequency serial clock divider(s) out of reset by setting the RHCLKRST bit for the receiver and/or the XHCLKRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be held at 0.
   (b) Software must read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.

4. Start the respective serial clocks ACLKX and/or ACLKR. This step can be skipped if external serial clocks are used and they are running:
   (a) Take the respective internal serial clock divider(s) out of reset by setting the RCLKRST bit for the receiver and/or the XCLKRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
   (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
5. Setup data acquisition as required:
   (a) If DMA is used to service the McASP, set up data acquisition as desired and start the DMA in this step, before the McASP is taken out of reset.
   (b) If CPU interrupt is used to service the McASP, enable the transmit and/ or receive interrupt as required.
   (c) If CPU polling is used to service the McASP, no action is required in this step.
6. Activate serializers.
   (a) Before starting, clear the respective transmitter and receiver status registers by writing XSTAT = FFFFh and RSTAT = FFFFh.
   (b) Take the respective serializers out of reset by setting the RSRCLR bit for the receiver and/or the XSRCLR bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
   (c) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
7. Verify that all transmit buffers are serviced. Skip this step if the transmitter is not used. Also, skip this step if time slot 0 is selected as inactive (special cases, see Figure 14-20, second waveform). As soon as the transmit serializer is taken out of reset, XDATA in the XSTAT register is set, indicating that XBUF is empty and ready to be serviced. The XDATA status causes an DMA event AXEVT to be generated, and can cause an interrupt AXINT to be generated if it is enabled in the XINTCTL register.
   (a) If DMA is used to service the McASP, the DMA automatically services the McASP upon receiving AXEVT. Before proceeding in this step, you should verify that the XDATA bit in the XSTAT is cleared to 0, indicating that all transmit buffers are already serviced by the DMA.
   (b) If CPU interrupt is used to service the McASP, interrupt service routine is entered upon the AXINT interrupt. The interrupt service routine should service the XBUF registers. Before proceeding in this step, you should verify that the XDATA bit in XSTAT is cleared to 0, indicating that all transmit buffers are already serviced by the CPU.
   (c) If CPU polling is used to service the McASP, the XBUF registers should be written to in this step.
8. Release state machines from reset.
   (a) Take the respective state machine(s) out of reset by setting the RSMRST bit for the receiver and/or the XSMRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
   (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
9. Release frame sync generators from reset. Note that it is necessary to release the internal frame sync generators from reset, even if an external frame sync is being used, because the frame sync error detection logic is built into the frame sync generator.
   (a) Take the respective frame sync generator(s) out of reset by setting the RFRST bit for the receiver, and/or the XFRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
   (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.

10. Upon the first frame sync signal, McASP transfers begin. The McASP synchronizes to an edge on the frame sync pin, not the level on the frame sync pin. This makes it easy to release the state machine and frame sync generators from reset.
    (a) For example, if you configure the McASP for a rising edge transmit frame sync, then you do not need to wait for a low level on the frame sync pin before releasing the McASP transmitter state machine and frame sync generators from reset.

14.2.10.3 Separate Transmit and Receive Initialization

In many cases, it is desirable to separately initialize the McASP transmitter and receiver. For example, you may delay the initialization of the transmitter until the type of data coming in on the receiver is recognized. Or a change in the incoming data stream on the receiver may necessitate a reinitialization of the transmitter.

In this case, you may still follow the sequence outlined in Section 14.2.10.2, but use it for each section (transmit, receive) individually. The GBLCTL register is aliased to RGBLCTL and XGBLCTL to facilitate separate initialization of transmit and receive sections.

14.2.10.4 Importance of Reading Back GBLCTL

In Section 14.2.10.2, steps 3b, 4b, 6c, 8b, and 9b state that GBLCTL should be read back until the bits that were written are successfully latched. This is important, because the transmitter and receiver state machines run off of the respective bit clocks, which are typically about tens to hundreds of times slower than the processor's internal bus clock. Therefore, it takes many cycles between when the processor writes to GBLCTL (or RGBLCTL and XGBLCTL), and when the McASP actually recognizes the write operation. If you skip this step, then the McASP may never see the reset bits in the global control registers get asserted and de-asserted; resulting in an uninitialized McASP.

Therefore, the logic in McASP has been implemented such that once the processor writes GBLCTL, RGBLCTL, or XGBLCTL, the resulting write is not visible by reading back GBLCTL until the McASP has recognized the change. This typically requires two bit clocks plus two processor bus clocks to occur.

Also, if the bit clocks can be completely stopped, any software that polls GBLCTL should be implemented with a time-out. If GBLCTL does not have a time-out, and the bit clock stops, the changes written to GBLCTL will not be reflected until the bit clock restarts.

Finally, please note that while RGBLCTL and XGBLCTL allow separate changing of the receive and transmit halves of GBLCTL, they also immediately reflect the updated value (useful for debug purposes). Only GBLCTL can be used for the read back step.

14.2.10.5 Synchronous Transmit and Receive Operation (ASYNC = 0)

When ASYNC = 0 in ACLKXCTL, the transmit and receive sections operate synchronously from the transmit section clock and transmit frame sync signals (Figure 14-16). The receive section may have a different (but compatible in terms of slot size) data format.

When ASYNC = 0, the receive frame sync generator is internally disabled. If the AFSX pin is configured as an output, it serves as the frame sync signal for both transmit and receive. The AFSR pin should not be used because the transmit frame sync generator output, which is used by both the transmitter and the receiver when ASYNC = 0, is not propagated to the AFSR pin (Figure 14-18).

When ASYNC = 0, the transmit and receive sections must share some common settings, since they both use the same clock and frame sync signals:

- DITEN = 0 in DITCTL (TDM mode is enabled).
- The total number of bits per frame must be the same (that is, RSSZ × RMOD must equal to XSSZ × XMOD).
- Both transmit and receive should either be specified as burst or TDM mode, but not mixed.
- The settings in ACLKRCTL are irrelevant.
- FSXM must match FSRM.
- FXWID must match FRWID.

For all other settings, the transmit and receive sections may be programmed independently.

14.2.10.6 Asynchronous Transmit and Receive Operation (ASYNC = 1)

When ASYNC = 1 in ACLKXCTL, the transmit and receive sections operate completely independently and have separate clock and frame sync signals (Figure 14-16, Figure 14-17, and Figure 14-18). The events generated by each section come asynchronously.

14.2.11 Interrupts

14.2.11.1 Transmit Data Ready Interrupt

The transmit data ready interrupt (XDATA) is generated if XDATA is 1 in the XSTAT register and XDATA is also enabled in XINTCTL. Section 14.2.8.1.1 provides details on when XDATA is set in the XSTAT register.

A transmit start of frame interrupt (XSTAFRM) is triggered by the recognition of transmit frame sync. A transmit last slot interrupt (XLAST) is a qualified version of the data ready interrupt (XDATA). It has the same behavior as the data ready interrupt, but is further qualified by having the data requested belonging to the last slot (the slot that just ended was next-to-last TDM slot, current slot is last slot).

14.2.11.2 Receive Data Ready Interrupt

The receive data ready interrupt (RDATA) is generated if RDATA is 1 in the RSTAT register and RDATA is also enabled in RINTCTL. Section 14.2.8.1.2 provides details on when RDATA is set in the RSTAT register.

A receiver start of frame interrupt (RSTAFRM) is triggered by the recognition of a receiver frame sync. A receiver last slot interrupt (RLAST) is a qualified version of the data ready interrupt (RDATA). It has the same behavior as the data ready interrupt, but is further qualified by having the data in the buffer come from the last TDM time slot (the slot that just ended was last TDM slot).

14.2.11.3 Error Interrupts

Upon detection, the following error conditions generate interrupt flags:

- In the receive status register (RSTAT):
  - Receiver overrun (ROVRN).
  - Unexpected receive frame sync (RSYNCERR).
  - Receive clock failure (RCKFAIL).
  - Receive DMA error (RDMAERR).
- In the transmit status register (XSTAT):
  - Transmit underrun (XUNDRN).
  - Unexpected transmit frame sync (XSYNCERR).
  - Transmit clock failure (XCKFAIL).
  - Transmit DMA error (XDMAERR).

Each interrupt source also has a corresponding enable bit in the receive interrupt control register (RINTCTL) and transmit interrupt control register (XINTCTL). If the enable bit is set in RINTCTL or XINTCTL, an interrupt is requested when the interrupt flag is set in RSTAT or XSTAT. If the enable bit is not set, no interrupt request is generated. However, the interrupt flag may be polled.

14.2.11.4 Audio Mute (AMUTE) Function

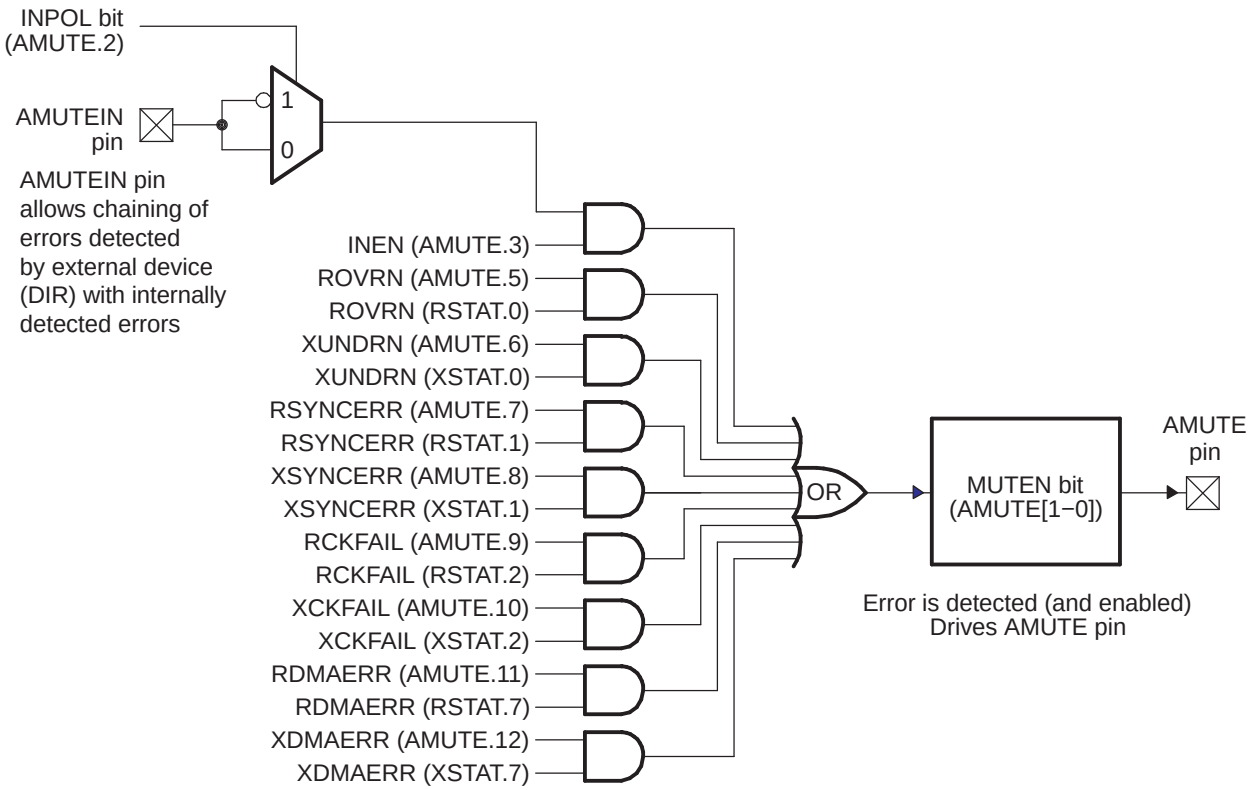
The McASP includes an automatic audio mute function (Figure 14-33) that asserts in hardware the AMUTE pin to a preprogrammed output state, as selected by the MUTEN bit in the audio mute control register (AMUTE). The AMUTE pin is asserted when one of the interrupt flags is set or an external device issues an error signal on the AMUTEIN input. Typically, the AMUTEIN input is shared with a device interrupt pin (for example EXT_INT4).

The AMUTEIN input allows the on-chip logic to consider a mute input from other devices in the system, so that all errors may be considered. The AMUTEIN input has a programmable polarity to allow it to adapt to different devices, as selected by the INPOL bit in AMUTE, and it must be enabled explicitly.

In addition to the external AMUTEIN input, the AMUTE pin output may be asserted when one of the error interrupt flags is set and its mute function is enabled in AMUTE.

When one or more of the errors is detected and enabled, the AMUTE pin is driven to an active state that is selected by MUTEN in AMUTE. The active polarity of the AMUTE pin is programmable by MUTEN (and the inactive polarity is the opposite of the active polarity). The AMUTE pin remains driven active until software clears all the error interrupt flags that are enabled to mute, and until the AMUTEIN is inactive.

Figure 14-33. Audio Mute (AMUTE) Block Diagram



14.2.11.5 Multiple Interrupts

This only applies to interrupts and not to DMA requests. The following terms are defined:

- **Active Interrupt Request:** a flag in RSTAT or XSTAT is set and the interrupt is enabled in RINTCTL or XINTCTL.
- **Outstanding Interrupt Request:** An interrupt request has been issued on one of the McASP transmit/receive interrupt ports, but that request has not yet been serviced.
- **Serviced:** The CPU writes to RSTAT or XSTAT to clear one or more of the active interrupt request flags.

The first interrupt request to become active for the transmitter with the interrupt flag set in XSTAT and the interrupt enabled in XINTCTL generates a request on the McASP transmit interrupt port AXINT.

If more than one interrupt request becomes active in the same cycle, a single interrupt request is generated on the McASP transmit interrupt port. Subsequent interrupt requests that become active while the first interrupt request is outstanding do not immediately generate a new request pulse on the McASP transmit interrupt port.

The transmit interrupt is serviced with the CPU writing to XSTAT. If any interrupt requests are active after the write, a new request is generated on the McASP transmit interrupt port.

The receiver operates in a similar way, but using RSTAT, RINTCTL, and the McASP receive interrupt port ARINT.

One outstanding interrupt request is allowed on each port, so a transmit and a receive interrupt request may both be outstanding at the same time.

14.2.12 EDMA Event Support

14.2.12.1 EDMA Events

There are six EDMA events.

14.2.12.2 Using the DMA for McASP Servicing

The most typical scenario is to use the DMA to service the McASP through the data port, although the DMA can also service the McASP through the configuration bus. Two possibilities exist for using the DMA events to service the McASP:

1. Use **AXEVT/AREVT**: Triggered upon each XDATA/RDATA transition from 0 to 1.
2. Use **AXEVTO/AREVTO** and **AXEVTE/AREVTE**: Alternating AXEVT/AREVT events for odd/even slots. Upon AXEVT/AREVT, AXEVTO/AREVTO is triggered if the event is for an odd channel, and AXEVTE/AREVTE is triggered if the event is for an even channel.

NOTE: Check the device-specific data manual to see if AXEVTO/AREVTO and AXEVTE/AREVTE are supported. These are optional.

Figure 14-34 and Figure 14-35 show an example audio system with six audio channels (LF, RF, LS, RS, C, and LFE) transmitted from three AXRn pins on the McASP. Figure 14-34 and Figure 14-35 show when events AXEVT, AXEVTO, and AXEVTE are triggered. Figure 14-34 and Figure 14-35 also apply for the receive audio channels and show when events AREVT, AREVTO, and AREVTE are triggered.

You can either use the DMA to service the McASP upon events AXEVT and AREVT (Figure 14-34) or upon events AXEVTO, AREVTO, AXEVTE, and AREVTE (Figure 14-35).

Figure 14-34. DMA Events in an Audio Example–Two Events (Scenario 1)

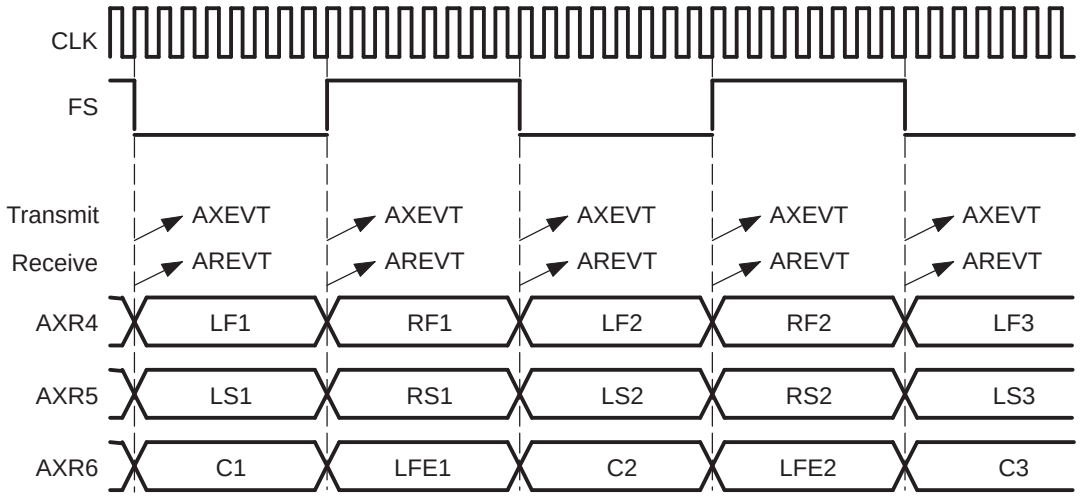
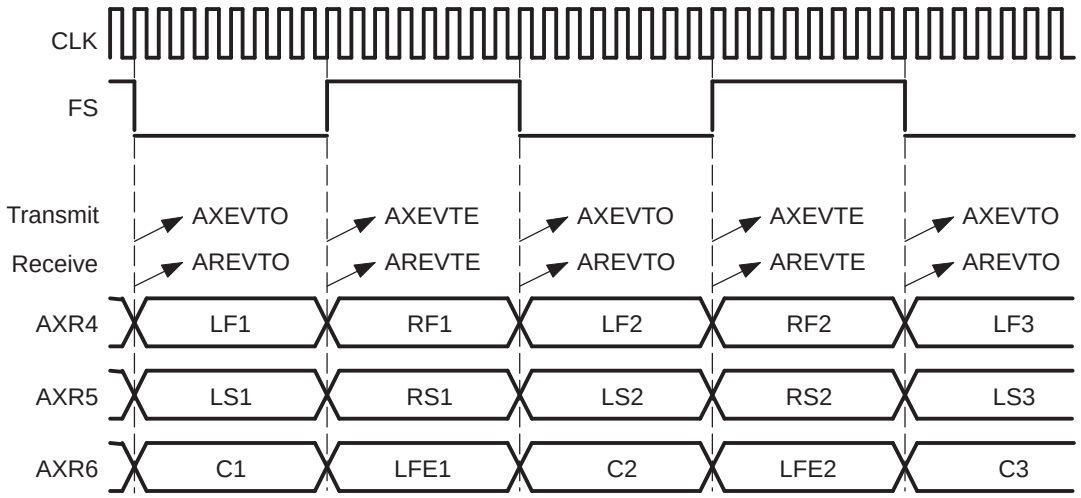


Figure 14-35. DMA Events in an Audio Example–Four Events (Scenario 2)



In scenario 1 (Figure 14-34), a DMA event AXEVT/AREVT is triggered on each time slot. In the example, AXEVT is triggered for each of the transmit audio channel time slot (Time slot for channels LF, LS, and C; and time slot for channels RF, RS, LFE). Similarly, AREVT is triggered for each of the receive audio channel time slot. Scenario 1 allows for the use of a single DMA to transmit all audio channels, and a single DMA to receive all audio channels.

In scenario 2 (Figure 14-35), two alternating DMA events are triggered for each time slot. In the example, AXEVTE (even) is triggered for the time slot for the even audio channels (LF, LS, C) and AXEVTO (odd) is triggered for the time slot for the odd audio channels (RF, RS, LFE). AXEVTO and AXEVTE alternate in time. The same is true in the receive direction with the use of AREVTO and AREVTE. This scenario allows for the use of two DMA channels (odd and even) to transmit all audio channels, and two DMA channels to receive all audio channels.

Here are some guidelines on using the different DMA events:

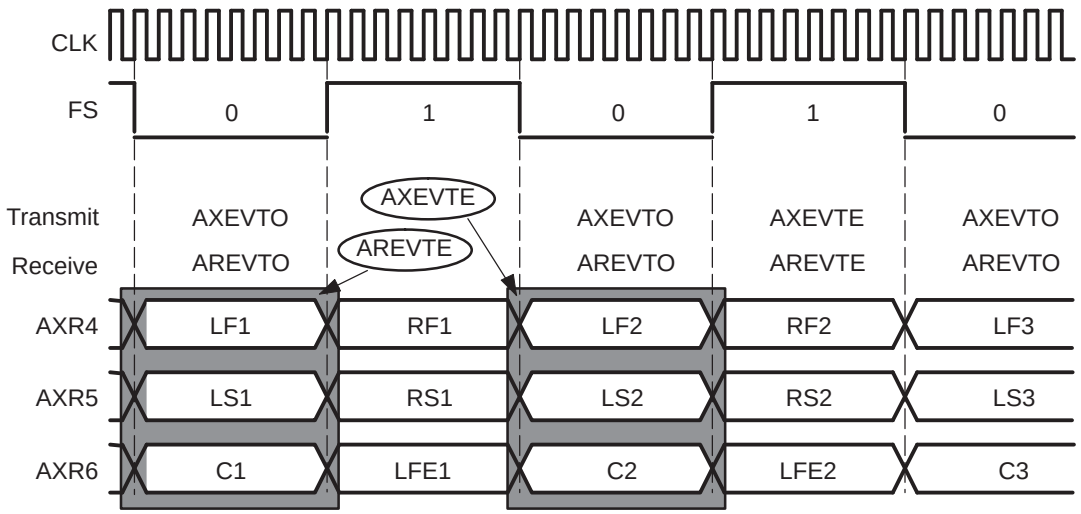
- Either use AXEVT, or the combination of AXEVTO and AXEVTE, to service the McASP. Never use all three at the same time. Similarly for receive, either use AREVT, or the combination of AREVTO and AREVTE.
- The McASP generates transmit DMA events independently from receive DMA events; therefore, separate schemes can be used for transmit and receive DMA. For example, scenario 1 could be used for the transmit data (AXEVT) and scenario 2 could be used for the receive data (AREVTO, AREVTE), and conversely.

Note the difference between DMA event generation and the CPU interrupt generation. DMA events are generated automatically upon data ready; whereas CPU interrupt generation needs to be enabled in the XINTCTL/RINTCTL register.

In Figure 14-35, scenario 2, each transmit DMA request is for data in the next time slot, while each receive DMA request is for data in the previous time slot. For example, Figure 14-36 shows a circled AXEVTE event for an even time slot transmit DMA request. The transmitter always requests a DMA transfer for data it will need to transmit during the next time slot. So, in this example, the circled event AXEVTE is a request for data for samples LF2, LS2, and C2.

On the other hand, the circled AREVTE event is an even time slot receive DMA request. The receiver always requests a DMA transfer for data it received during the previous time slot. In this example, the circled event AREVTE is a request for samples LF1, LS1, and C1.

Figure 14-36. DMA Events in an Audio Example



14.2.13 Power Management

The McASP can be placed in reduced power modes to conserve power during periods of low activity.

14.2.14 Emulation Considerations

NOTE: The receive buffer registers (RBUF*n*) and transmit buffer registers (XBUF*n*) should not be accessed by the emulator when the McASP is running. Such an access will cause the RRDY/XRDY bit in the serializer control register *n* (SRCTL*n*) to be updated.

The McASP does not support the emulation suspend. If the McASP is placed into a suspended state, software needs to ensure that the registers/internal logic is reset to its default values by performing a module power cycle before restarting the initialization sequence.

14.3 McASP Registers

Table 14-7 lists the control registers for the McASP. For the base address of these registers, see Table 1-12. The control registers are accessed through the configuration bus of the device. The receive buffer registers (RBUF*n*) and transmit buffer registers (XBUF*n*) can also be accessed through the data port of the device, as listed in Table 14-8.

Control registers for the McASP Audio FIFO (AFIFO) are summarized in Table 14-9. Note that the AFIFO Write FIFO (WFIFO) and Read FIFO (RFIFO) have independent control and status registers. The AFIFO control registers are accessed through the peripheral configuration port.

Table 14-7. McASP Registers Accessed Through Configuration Bus

Offset	Acronym	Register Description	Section
0h	REV	Revision identification register	Section 14.3.1
10h	PFUNC	Pin function register	Section 14.3.2
14h	PDIR	Pin direction register	Section 14.3.3
18h	PDOUT	Pin data output register	Section 14.3.4
1Ch	PDIN	Read returns: Pin data input register	Section 14.3.5
1Ch	PDSET	Writes affect: Pin data set register (alternate write address: PDOUT)	Section 14.3.6
20h	PDCLR	Pin data clear register (alternate write address: PDOUT)	Section 14.3.7
44h	GBLCTL	Global control register	Section 14.3.8
48h	AMUTE	Audio mute control register	Section 14.3.9
4Ch	DLBCTL	Digital loopback control register	Section 14.3.10
50h	DITCTL	DIT mode control register	Section 14.3.11
60h	RGBLCTL	Receiver global control register: Alias of GBLCTL, only receive bits are affected - allows receiver to be reset independently from transmitter	Section 14.3.12
64h	RMASK	Receive format unit bit mask register	Section 14.3.13
68h	RFMT	Receive bit stream format register	Section 14.3.14
6Ch	AFSRCTL	Receive frame sync control register	Section 14.3.15
70h	ACLKRCTL	Receive clock control register	Section 14.3.16
74h	AHCLKRCTL	Receive high-frequency clock control register	Section 14.3.17
78h	RTDM	Receive TDM time slot 0-31 register	Section 14.3.18
7Ch	RINTCTL	Receiver interrupt control register	Section 14.3.19
80h	RSTAT	Receiver status register	Section 14.3.20
84h	RSLOT	Current receive TDM time slot register	Section 14.3.21
88h	RCLKCHK	Receive clock check control register	Section 14.3.22
8Ch	REVTCTL	Receiver DMA event control register	Section 14.3.23
A0h	XGBLCTL	Transmitter global control register. Alias of GBLCTL, only transmit bits are affected - allows transmitter to be reset independently from receiver	Section 14.3.24
A4h	XMASK	Transmit format unit bit mask register	Section 14.3.25
A8h	XFMT	Transmit bit stream format register	Section 14.3.26
ACh	AFSXCTL	Transmit frame sync control register	Section 14.3.27
B0h	ACLKXCTL	Transmit clock control register	Section 14.3.28
B4h	AHCLKXCTL	Transmit high-frequency clock control register	Section 14.3.29
B8h	XTDM	Transmit TDM time slot 0-31 register	Section 14.3.30
BCh	XINTCTL	Transmitter interrupt control register	Section 14.3.31
C0h	XSTAT	Transmitter status register	Section 14.3.32
C4h	XSLOT	Current transmit TDM time slot register	Section 14.3.33
C8h	XCLKCHK	Transmit clock check control register	Section 14.3.34
CCh	XEVTCTL	Transmitter DMA event control register	Section 14.3.35
100h	DITCSRA0	Left (even TDM time slot) channel status register (DIT mode) 0	Section 14.3.37
104h	DITCSRA1	Left (even TDM time slot) channel status register (DIT mode) 1	Section 14.3.37

Table 14-7. McASP Registers Accessed Through Configuration Bus (continued)

Offset	Acronym	Register Description	Section
108h	DITCSRA2	Left (even TDM time slot) channel status register (DIT mode) 2	Section 14.3.37
10Ch	DITCSRA3	Left (even TDM time slot) channel status register (DIT mode) 3	Section 14.3.37
110h	DITCSRA4	Left (even TDM time slot) channel status register (DIT mode) 4	Section 14.3.37
114h	DITCSRA5	Left (even TDM time slot) channel status register (DIT mode) 5	Section 14.3.37
118h	DITCSRB0	Right (odd TDM time slot) channel status register (DIT mode) 0	Section 14.3.38
11Ch	DITCSRB1	Right (odd TDM time slot) channel status register (DIT mode) 1	Section 14.3.38
120h	DITCSRB2	Right (odd TDM time slot) channel status register (DIT mode) 2	Section 14.3.38
124h	DITCSRB3	Right (odd TDM time slot) channel status register (DIT mode) 3	Section 14.3.38
128h	DITCSRB4	Right (odd TDM time slot) channel status register (DIT mode) 4	Section 14.3.38
12Ch	DITCSRB5	Right (odd TDM time slot) channel status register (DIT mode) 5	Section 14.3.38
130h	DITUDRA0	Left (even TDM time slot) channel user data register (DIT mode) 0	Section 14.3.39
134h	DITUDRA1	Left (even TDM time slot) channel user data register (DIT mode) 1	Section 14.3.39
138h	DITUDRA2	Left (even TDM time slot) channel user data register (DIT mode) 2	Section 14.3.39
13Ch	DITUDRA3	Left (even TDM time slot) channel user data register (DIT mode) 3	Section 14.3.39
140h	DITUDRA4	Left (even TDM time slot) channel user data register (DIT mode) 4	Section 14.3.39
144h	DITUDRA5	Left (even TDM time slot) channel user data register (DIT mode) 5	Section 14.3.39
148h	DITUDRB0	Right (odd TDM time slot) channel user data register (DIT mode) 0	Section 14.3.40
14Ch	DITUDRB1	Right (odd TDM time slot) channel user data register (DIT mode) 1	Section 14.3.40
150h	DITUDRB2	Right (odd TDM time slot) channel user data register (DIT mode) 2	Section 14.3.40
154h	DITUDRB3	Right (odd TDM time slot) channel user data register (DIT mode) 3	Section 14.3.40
158h	DITUDRB4	Right (odd TDM time slot) channel user data register (DIT mode) 4	Section 14.3.40
15Ch	DITUDRB5	Right (odd TDM time slot) channel user data register (DIT mode) 5	Section 14.3.40
180h	SRCTL0	Serializer control register 0	Section 14.3.36
184h	SRCTL1	Serializer control register 1	Section 14.3.36
188h	SRCTL2	Serializer control register 2	Section 14.3.36
18Ch	SRCTL3	Serializer control register 3	Section 14.3.36
190h	SRCTL4	Serializer control register 4	Section 14.3.36
194h	SRCTL5	Serializer control register 5	Section 14.3.36
198h	SRCTL6	Serializer control register 6	Section 14.3.36
19Ch	SRCTL7	Serializer control register 7	Section 14.3.36
1A0h	SRCTL8	Serializer control register 8	Section 14.3.36
1A4h	SRCTL9	Serializer control register 9	Section 14.3.36
1A8h	SRCTL10	Serializer control register 10	Section 14.3.36
1ACh	SRCTL11	Serializer control register 11	Section 14.3.36
1B0h	SRCTL12	Serializer control register 12	Section 14.3.36
1B4h	SRCTL13	Serializer control register 13	Section 14.3.36
1B8h	SRCTL14	Serializer control register 14	Section 14.3.36
1BCh	SRCTL15	Serializer control register 15	Section 14.3.36
200h	XBUF0	Transmit buffer register for serializer 0	Section 14.3.41
204h	XBUF1	Transmit buffer register for serializer 1	Section 14.3.41
208h	XBUF2	Transmit buffer register for serializer 2	Section 14.3.41
20Ch	XBUF3	Transmit buffer register for serializer 3	Section 14.3.41
210h	XBUF4	Transmit buffer register for serializer 4	Section 14.3.41
214h	XBUF5	Transmit buffer register for serializer 5	Section 14.3.41
218h	XBUF6	Transmit buffer register for serializer 6	Section 14.3.41
21Ch	XBUF7	Transmit buffer register for serializer 7	Section 14.3.41
220h	XBUF8	Transmit buffer register for serializer 8	Section 14.3.41

Table 14-7. McASP Registers Accessed Through Configuration Bus (continued)

Offset	Acronym	Register Description	Section
224h	XBUF9	Transmit buffer register for serializer 9	Section 14.3.41
228h	XBUF10	Transmit buffer register for serializer 10	Section 14.3.41
22Ch	XBUF11	Transmit buffer register for serializer 11	Section 14.3.41
230h	XBUF12	Transmit buffer register for serializer 12	Section 14.3.41
234h	XBUF13	Transmit buffer register for serializer 13	Section 14.3.41
238h	XBUF14	Transmit buffer register for serializer 14	Section 14.3.41
23Ch	XBUF15	Transmit buffer register for serializer 15	Section 14.3.41
280h	RBUF0	Receive buffer register for serializer 0	Section 14.3.42
284h	RBUF1	Receive buffer register for serializer 1	Section 14.3.42
288h	RBUF2	Receive buffer register for serializer 2	Section 14.3.42
28Ch	RBUF3	Receive buffer register for serializer 3	Section 14.3.42
290h	RBUF4	Receive buffer register for serializer 4	Section 14.3.42
294h	RBUF5	Receive buffer register for serializer 5	Section 14.3.42
298h	RBUF6	Receive buffer register for serializer 6	Section 14.3.42
29Ch	RBUF7	Receive buffer register for serializer 7	Section 14.3.42
2A0h	RBUF8	Receive buffer register for serializer 8	Section 14.3.42
2A4h	RBUF9	Receive buffer register for serializer 9	Section 14.3.42
2A8h	RBUF10	Receive buffer register for serializer 10	Section 14.3.42
2ACh	RBUF11	Receive buffer register for serializer 11	Section 14.3.42
2B0h	RBUF12	Receive buffer register for serializer 12	Section 14.3.42
2B4h	RBUF13	Receive buffer register for serializer 13	Section 14.3.42
2B8h	RBUF14	Receive buffer register for serializer 14	Section 14.3.42
2BCh	RBUF15	Receive buffer register for serializer 15	Section 14.3.42

Table 14-8. McASP Registers Accessed Through Data Port

Hex Address	Register Name	Register Description
Read Accesses	RBUF[1]	Receive buffer data peripheral register address. Cycles through receive serializers, skipping over transmit serializers and inactive serializers. Starts at the lowest serializer at the beginning of each time slot. DAT BUS only if XBUSEL = 0. See the device specific data manual for address location.
Write Accesses	XBUF[1]	Transmit buffer data peripheral register address. Cycles through transmit serializers, skipping over receive and inactive serializers. Starts at the lowest serializer at the beginning of each time slot. DAT BUS only if RBUSEL = 0. See the device specific data manual for address location.

(1) RBUF and XBUF are at the same address location. Reads access RBUF and writes access XBUF.

Table 14-9. McASP AFIFO Registers Accessed Through Peripheral Configuration Port

Offset	Acronym	Register Description	Section
1000h	WFIFOCTL	Write FIFO control register	Section 14.3.43
1004h	WFIFOSTS	Write FIFO status register	Section 14.3.44
1008h	RFIFOCTL	Read FIFO control register	Section 14.3.45
100Ch	RFIFOSTS	Read FIFO status register	Section 14.3.46

14.3.1 Revision Identification Register (REV)

The revision identification register (REV) contains identification data for the peripheral. The REV is shown in Figure 14-37 and described in Table 14-10.

Figure 14-37. Revision Identification Register (REV)

31	0
REV	
R-4430 7302h	

LEGEND: R = Read only; -*n* = value after reset

Table 14-10. Revision Identification Register (REV) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4430 7302h	Identifies revision of peripheral.

14.3.2 Pin Function Register (PFUNC)

The pin function register (PFUNC) specifies the function of AXRn, ACLKX, AHCLKX, AFSX, ACLKR, AHCLKR, and AFSR pins as either a McASP pin or a general-purpose input/output (GPIO) pin. The PFUNC is shown in Figure 14-38 and described in Table 14-11.

CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

Figure 14-38. Pin Function Register (PFUNC)

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-11. Pin Function Register (PFUNC) Field Descriptions

Bit	Field	Value	Description
31	AFSR		Determines if AFSR pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
30	AHCLKR		Determines if AHCLKR pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
29	ACLKR		Determines if ACLKR pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
28	AFSX		Determines if AFSX pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
27	AHCLKX		Determines if AHCLKX pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
26	ACLKX		Determines if ACLKX pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
25	AMUTE		Determines if AMUTE pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]		Determines if AXRn pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.

14.3.3 Pin Direction Register (PDIR)

The pin direction register (PDIR) specifies the direction of AXRn, ACLKX, AHCLKX, AFSX, ACLKR, AHCLKR, and AFSR pins as either an input or an output pin. The PDIR is shown in Figure 14-39 and described in Table 14-12. Refer to the device-specific data manual for the number of pins available on your device.

Regardless of the pin function register (PFUNC) setting, each PDIR bit must be set to 1 for the specified pin to be enabled as an output and each PDIR bit must be cleared to 0 for the specified pin to be an input.

For example, if the McASP is configured to use an internally-generated bit clock and the clock is to be driven out to the system, the PFUNC bit must be cleared to 0 (McASP function) and the PDIR bit must be set to 1 (an output).

When AXRn is configured to transmit, the PFUNC bit must be cleared to 0 (McASP function) and the PDIR bit must be set to 1 (an output). Similarly, when AXRn is configured to receive, the PFUNC bit must be cleared to 0 (McASP function) and the PDIR bit must be cleared to 0 (an input).

CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

Figure 14-39. Pin Direction Register (PDIR)

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-12. Pin Direction Register (PDIR) Field Descriptions

Bit	Field	Value	Description
31	AFSR		Determines if AFSR pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
30	AHCLKR		Determines if AHCLKR pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
29	ACLKR		Determines if ACLKR pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
28	AFSX		Determines if AFSX pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
27	AHCLKX		Determines if AHCLKX pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
26	ACLKX		Determines if ACLKX pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
25	AMUTE		Determines if AMUTE pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]		Determines if AXRn pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.

14.3.4 Pin Data Output Register (PDOUT)

The pin data output register (PDOUT) holds a value for data out at all times, and may be read back at all times. The value held by PDOUT is not affected by writing to PDIR and PFUNC. However, the data value in PDOUT is driven out onto the McASP pin only if the corresponding bit in PFUNC is set to 1 (GPIO function) and the corresponding bit in PDIR is set to 1 (output).

When reading data, it returns the corresponding PDOUT[n] bit value and does not return the input from I/O pin. When writing data, it writes to the corresponding PDOUT[n] bit.

The PDOUT is shown in Figure 14-40 and described in Table 14-13.

PDOUT has these aliases or alternate addresses:

- PDSET - when written to at this address, writing a 1 to a bit in PDSET sets the corresponding bit in PDOUT to 1; writing a 0 has no effect and keeps the bits in PDOUT unchanged.
- PDCLR - when written to at this address, writing a 1 to a bit in PDCLR clears the corresponding bit in PDOUT to 0; writing a 0 has no effect and keeps the bits in PDOUT unchanged.

There is only one set of data out bits, PDOUT[31-0]. The other registers, PDSET and PDCLR, are just different addresses for the same control bits, with different behaviors during writes.

CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

Figure 14-40. Pin Data Output Register (PDOUT)

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-13. Pin Data Output Register (PDOUT) Field Descriptions

Bit	Field	Value	Description
31	AFSR		Determines drive on AFSR output pin when the corresponding PFUNC[31] and PDIR[31] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
30	AHCLKR		Determines drive on AHCLKR output pin when the corresponding PFUNC[30] and PDIR[30] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
29	ACLKR		Determines drive on ACLKR output pin when the corresponding PFUNC[29] and PDIR[29] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
28	AFSX		Determines drive on AFSX output pin when the corresponding PFUNC[28] and PDIR[28] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
27	AHCLKX		Determines drive on AHCLKX output pin when the corresponding PFUNC[27] and PDIR[27] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
26	ACLKX		Determines drive on ACLKX output pin when the corresponding PFUNC[26] and PDIR[26] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
25	AMUTE		Determines drive on AMUTE output pin when the corresponding PFUNC[25] and PDIR[25] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]		Determines drive on AXR[n] output pin when the corresponding PFUNC[n] and PDIR[n] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.

14.3.5 Pin Data Input Register (PDIN)

The pin data input register (PDIN) holds the I/O pin state of each of the McASP pins. PDIN allows the actual value of the pin to be read, regardless of the state of PFUNC and PDIR. The value after reset for registers 1 through 15 and 24 through 31 depends on how the pins are being driven. The PDIN is shown in Figure 14-41 and described in Table 14-14.

CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

Figure 14-41. Pin Data Input Register (PDIN)

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-14. Pin Data Input Register (PDIN) Field Descriptions

Bit	Field	Value	Description
31	AFSR		Logic level on AFSR pin.
		0	Pin is logic low.
		1	Pin is logic high.
30	AHCLKR		Logic level on AHCLKR pin.
		0	Pin is logic low.
		1	Pin is logic high.
29	ACLKR		Logic level on ACLKR pin.
		0	Pin is logic low.
		1	Pin is logic high.
28	AFSX		Logic level on AFSX pin.
		0	Pin is logic low.
		1	Pin is logic high.
27	AHCLKX		Logic level on AHCLKX pin.
		0	Pin is logic low.
		1	Pin is logic high.
26	ACLKX		Logic level on ACLKX pin.
		0	Pin is logic low.
		1	Pin is logic high.
25	AMUTE		Logic level on AMUTE pin.
		0	Pin is logic low.
		1	Pin is logic high.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]		Logic level on AXR[n] pin.
		0	Pin is logic low.
		1	Pin is logic high.

14.3.6 Pin Data Set Register (PDSET)

The pin data set register (PDSET) is an alias of the pin data output register (PDOUT) for writes only. Writing a 1 to the PDSET bit sets the corresponding bit in PDOUT and, if PFUNC = 1 (GPIO function) and PDIR = 1 (output), drives a logic high on the pin. PDSET is useful for a multitasking system because it allows you to set to a logic high only the desired pin(s) within a system without affecting other I/O pins controlled by the same McASP. The PDSET is shown in Figure 14-42 and described in Table 14-15.

CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

Figure 14-42. Pin Data Set Register (PDSET)

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-15. Pin Data Set Register (PDSET) Field Descriptions

Bit	Field	Value	Description
31	AFSR		Allows the corresponding AFSR bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[31] bit is set to 1.
30	AHCLKR		Allows the corresponding AHCLKR bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[30] bit is set to 1.
29	ACLKR		Allows the corresponding ACLKR bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[29] bit is set to 1.
28	AFSX		Allows the corresponding AFSX bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[28] bit is set to 1.
27	AHCLKX		Allows the corresponding AHCLKX bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[27] bit is set to 1.
26	ACLKX		Allows the corresponding ACLKX bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[26] bit is set to 1.
25	AMUTE		Allows the corresponding AMUTE bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[25] bit is set to 1.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]		Allows the corresponding AXR[n] bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[n] bit is set to 1.

14.3.7 Pin Data Clear Register (PDCLR)

The pin data clear register (PDCLR) is an alias of the pin data output register (PDOUT) for writes only. Writing a 1 to the PDCLR bit clears the corresponding bit in PDOUT and, if PFUNC = 1 (GPIO function) and PDIR = 1 (output), drives a logic low on the pin. PDCLR is useful for a multitasking system because it allows you to clear to a logic low only the desired pin(s) within a system without affecting other I/O pins controlled by the same McASP. The PDCLR is shown in Figure 14-43 and described in Table 14-16.

CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

Figure 14-43. Pin Data Clear Register (PDCLR)

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-16. Pin Data Clear Register (PDCLR) Field Descriptions

Bit	Field	Value	Description
31	AFSR		Allows the corresponding AFSR bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[31] bit is cleared to 0.
30	AHCLKR		Allows the corresponding AHCLKR bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[30] bit is cleared to 0.
29	ACLKR		Allows the corresponding ACLKR bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[29] bit is cleared to 0.
28	AFSX		Allows the corresponding AFSX bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[28] bit is cleared to 0.
27	AHCLKX		Allows the corresponding AHCLKX bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[27] bit is cleared to 0.
26	ACLKX		Allows the corresponding ACLKX bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[26] bit is cleared to 0.
25	AMUTE		Allows the corresponding AMUTE bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[25] bit is cleared to 0.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]		Allows the corresponding AXR[n] bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[n] bit is cleared to 0.

14.3.8 Global Control Register (GBLCTL)

The global control register (GBLCTL) provides initialization of the transmit and receive sections. The GBLCTL is shown in Figure 14-44 and described in Table 14-17.

The bit fields in GBLCTL are synchronized and latched by the corresponding clocks (ACLKX for bits 12-8 and ACLKR for bits 4-0). Before GBLCTL is programmed, you must ensure that serial clocks are running. If the corresponding external serial clocks, ACLKX and ACLKR, are not yet running, you should select the internal serial clock source in AHCLKXCTL, AHCLKRCTL, ACLKXCTL, and ACLKRCTL before GBLCTL is programmed. Also, after programming any bits in GBLCTL you should not proceed until you have read back from GBLCTL and verified that the bits are latched in GBLCTL.

Figure 14-44. Global Control Register (GBLCTL)

31	16
Reserved	
R-0	

15	13	12	11	10	9	8
Reserved		XFRST	XSMRST	XSRCLR	XHCLKRST	XCLKRST
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	5	4	3	2	1	0
Reserved		RFRST	RSMRST	RSRCLR	RHCLKRST	RCLKRST
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-17. Global Control Register (GBLCTL) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	XFRST		Transmit frame sync generator reset enable bit.
		0	Transmit frame sync generator is reset.
		1	Transmit frame sync generator is active. When released from reset, the transmit frame sync generator begins counting serial clocks and generating frame sync as programmed.
11	XSMRST		Transmit state machine reset enable bit.
		0	Transmit state machine is held in reset. AXRn pin state: If PFUNC[n] = 0 and PDIR[n] = 1; then the serializer drives the AXRn pin to the state specified for inactive time slot (as determined by DISMOD bits in SRCTL).
		1	Transmit state machine is released from reset. When released from reset, the transmit state machine immediately transfers data from XRBUF[n] to XRSR[n]. The transmit state machine sets the underrun flag (XUNDRN) in XSTAT, if XRBUF[n] have not been preloaded with data before reset is released. The transmit state machine also immediately begins detecting frame sync and is ready to transmit. Transmit TDM time slot begins at slot 0 after reset is released.
10	XSRCLR		Transmit serializer clear enable bit. By clearing then setting this bit, the transmit buffer is flushed to an empty state (XDATA = 1). If XSMRST = 1, XSRCLR = 1, XDATA = 1, and XBUF is not loaded with new data before the start of the next active time slot, an underrun will occur.
		0	Transmit serializers are cleared.
		1	Transmit serializers are active. When the transmit serializers are first taken out of reset (XSRCLR changes from 0 to 1), the transmit data ready bit (XDATA) in XSTAT is set to indicate XBUF is ready to be written.
9	XHCLKRST		Transmit high-frequency clock divider reset enable bit.
		0	Transmit high-frequency clock divider is held in reset and passes through its input as divide-by-1.
		1	Transmit high-frequency clock divider is running.
8	XCLKRST		Transmit clock divider reset enable bit.
		0	Transmit clock divider is held in reset. When the clock divider is in reset, it passes through a divide-by-1 of its input.
		1	Transmit clock divider is running.

Table 14-17. Global Control Register (GBLCTL) Field Descriptions (continued)

Bit	Field	Value	Description
7-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	RFRST		Receive frame sync generator reset enable bit.
		0	Receive frame sync generator is reset.
		1	Receive frame sync generator is active. When released from reset, the receive frame sync generator begins counting serial clocks and generating frame sync as programmed.
3	RSMRST		Receive state machine reset enable bit.
		0	Receive state machine is held in reset.
		1	Receive state machine is released from reset. When released from reset, the receive state machine immediately begins detecting frame sync and is ready to receive. Receive TDM time slot begins at slot 0 after reset is released.
2	RSRCLR		Receive serializer clear enable bit. By clearing then setting this bit, the receive buffer is flushed.
		0	Receive serializers are cleared.
		1	Receive serializers are active.
1	RHCLKRST		Receive high-frequency clock divider reset enable bit.
		0	Receive high-frequency clock divider is held in reset and passes through its input as divide-by-1.
		1	Receive high-frequency clock divider is running.
0	RCLKRST		Receive clock divider reset enable bit.
		0	Receive clock divider is held in reset. When the clock divider is in reset, it passes through a divide-by-1 of its input.
		1	Receive clock divider is running.

14.3.9 Audio Mute Control Register (AMUTE)

The audio mute control register (AMUTE) controls the McASP audio mute (AMUTE) output pin. The value after reset for register 4 depends on how the pins are being driven. The AMUTE is shown in Figure 14-45 and described in Table 14-18.

Figure 14-45. Audio Mute Control Register (AMUTE)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved			XDMAERR	RDMAERR	XCKFAIL	RCKFAIL	XSYNCERR
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
RSYNCERR	XUNDRN	ROVRN	INSTAT	INEN	INPOL	MUTEN	
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-18. Audio Mute Control Register (AMUTE) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	XDMAERR		If transmit DMA error (XDMAERR), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of transmit DMA error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of transmit DMA error, AMUTE is active and is driven according to MUTEN bit.
11	RDMAERR		If receive DMA error (RDMAERR), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of receive DMA error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of receive DMA error, AMUTE is active and is driven according to MUTEN bit.
10	XCKFAIL		If transmit clock failure (XCKFAIL), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of transmit clock failure is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of transmit clock failure, AMUTE is active and is driven according to MUTEN bit
9	RCKFAIL		If receive clock failure (RCKFAIL), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of receive clock failure is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of receive clock failure, AMUTE is active and is driven according to MUTEN bit.
8	XSYNCERR		If unexpected transmit frame sync error (XSYNCERR), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of unexpected transmit frame sync error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of unexpected transmit frame sync error, AMUTE is active and is driven according to MUTEN bit.
7	RSYNCERR		If unexpected receive frame sync error (RSYNCERR), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of unexpected receive frame sync error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of unexpected receive frame sync error, AMUTE is active and is driven according to MUTEN bit.
6	XUNDRN		If transmit underrun error (XUNDRN), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of transmit underrun error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of transmit underrun error, AMUTE is active and is driven according to MUTEN bit.

Table 14-18. Audio Mute Control Register (AMUTE) Field Descriptions (continued)

Bit	Field	Value	Description
5	ROVRN		If receiver overrun error (ROVRN), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of receiver overrun error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of receiver overrun error, AMUTE is active and is driven according to MUTEN bit.
4	INSTAT		Status of mute in pin, determines drive on AXRn pin when PFUNC[n] and PDIR[n] bits are set to 1.
		0	AMUTEIN pin is inactive.
		1	AMUTEIN pin is active. Audio mute in error is detected.
3	INEN		Drive AMUTE active when AMUTEIN error is active (INSTAT = 1).
		0	Drive is disabled. AMUTEIN is ignored by AMUTE.
		1	Drive is enabled (active). INSTAT = 1 drives AMUTE active.
2	INPOL		Audio mute in (AMUTEIN) polarity select bit.
		0	Polarity is active high. A high on AMUTEIN sets INSTAT to 1.
		1	Polarity is active low. A low on AM UTEIN sets INSTAT to 1.
1-0	MUTEN	0-3h	AMUTE pin enable bit (unless overridden by GPIO registers).
		0	AMUTE pin is disabled, pin goes to tri-state condition.
		1h	AMUTE pin is driven high if error is detected.
		2h	AMUTE pin is driven low if error is detected.
		3h	Reserved

14.3.10 Digital Loopback Control Register (DLBCTL)

The digital loopback control register (DLBCTL) controls the internal loopback settings of the McASP in TDM mode. The DLBCTL is shown in Figure 14-46 and described in Table 14-19. Note that loopback is NOT supported if McASP is configured in DIT mode.

Figure 14-46. Digital Loopback Control Register (DLBCTL)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		MODE		ORD	DLBEN
R-0		R/W-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-19. Digital Loopback Control Register (DLBCTL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3-2	MODE	0-3h	Loopback generator mode bits. Applies only when loopback mode is enabled (DLBEN = 1).
		0	Default and reserved on loopback mode (DLBEN = 1). When in non-loopback mode (DLBEN = 0), MODE should be left at default (00). When in loopback mode (DLBEN = 1), MODE = 00 is reserved and is not applicable.
		1h	MODE must be set to 01 when McASP operates in loopback mode (DLBEN = 1). This is necessary to allow transmit clock and frame sync generators to be used by both transmit and receive sections.
		2h-3h	Reserved.
1	ORD		Loopback order bit when loopback mode is enabled (DLBEN = 1).
		0	Odd serializers N + 1 transmit to even serializers N that receive. The corresponding serializers must be programmed properly.
		1	Even serializers N transmit to odd serializers N + 1 that receive. The corresponding serializers must be programmed properly.
0	DLBEN		Loopback mode enable bit.
		0	Loopback mode is disabled (normal McASP operation)
		1	Loopback mode is enabled (TDM mode only).

14.3.11 Digital Mode Control Register (DITCTL)

The DIT mode control register (DITCTL) controls DIT operations of the McASP. The DITCTL is shown in Figure 14-47 and described in Table 14-20.

Figure 14-47. Digital Mode Control Register (DITCTL)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		VB	VA	Rsvd	DITEN
R-0		R/W-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-20. Digital Mode Control Register (DITCTL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3	VB		Valid bit for odd time slots (DIT right subframe).
		0	V bit is 0 during odd DIT subframes.
		1	V bit is 1 during odd DIT subframes.
2	VA		Valid bit for even time slots (DIT left subframe).
		0	V bit is 0 during even DIT subframes.
		1	V bit is 1 during even DIT subframes.
1	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
0	DITEN		DIT mode enable bit. DITEN should only be changed while the XSMRST bit in GBLCTL is in reset (and for startup, XSRCLR also in reset). However, it is not necessary to reset the XCLKRST or XHCLKRST bits in GBLCTL to change DITEN.
		0	DIT mode is disabled. Transmitter operates in TDM or burst mode.
		1	DIT mode is enabled. Transmitter operates in DIT encoded mode.

14.3.12 Receiver Global Control Register (RGBLCTL)

Alias of the global control register (GBLCTL). Writing to the receiver global control register (RGBLCTL) affects only the receive bits of GBLCTL (bits 4-0). Reads from RGBLCTL return the value of GBLCTL. RGBLCTL allows the receiver to be reset independently from the transmitter. The RGBLCTL is shown in Figure 14-48 and described in Table 14-21. See Section 14.3.8 for a detailed description of GBLCTL.

Figure 14-48. Receiver Global Control Register (RGBLCTL)

31	16
Reserved	
R-0	

15	13	12	11	10	9	8
Reserved		XFRST	XSMRST	XSRCLR	XHCLKRST	XCLKRST
R-0		R-0	R-0	R-0	R-0	R-0

7	5	4	3	2	1	0
Reserved		RFRST	RSMRST	RSRCLR	RHCLKRST	RCLKRST
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-21. Receiver Global Control Register (RGBLCTL) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	XFRST	x	Transmit frame sync generator reset enable bit. A read of this bit returns the XFRST bit value of GBLCTL. Writes have no effect.
11	XSMRST	x	Transmit state machine reset enable bit. A read of this bit returns the XSMRST bit value of GBLCTL. Writes have no effect.
10	XSRCLR	x	Transmit serializer clear enable bit. A read of this bit returns the XSRCLR bit value of GBLCTL. Writes have no effect.
9	XHCLKRST	x	Transmit high-frequency clock divider reset enable bit. A read of this bit returns the XHCLKRST bit value of GBLCTL. Writes have no effect.
8	XCLKRST	x	Transmit clock divider reset enable bit. A read of this bit returns the XCLKRST bit value of GBLCTL. Writes have no effect.
7-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	RFRST		Receive frame sync generator reset enable bit. A write to this bit affects the RFRST bit of GBLCTL.
		0	Receive frame sync generator is reset.
		1	Receive frame sync generator is active.
3	RSMRST		Receive state machine reset enable bit. A write to this bit affects the RSMRST bit of GBLCTL.
		0	Receive state machine is held in reset.
		1	Receive state machine is released from reset.
2	RSRCLR		Receive serializer clear enable bit. A write to this bit affects the RSRCLR bit of GBLCTL.
		0	Receive serializers are cleared.
		1	Receive serializers are active.
1	RHCLKRST		Receive high-frequency clock divider reset enable bit. A write to this bit affects the RHCLKRST bit of GBLCTL.
		0	Receive high-frequency clock divider is held in reset and passes through its input as divide-by-1.
		1	Receive high-frequency clock divider is running.
0	RCLKRST		Receive clock divider reset enable bit. A write to this bit affects the RCLKRST bit of GBLCTL.
		0	Receive clock divider is held in reset. When the clock divider is in reset, it passes through a divide-by-1 of its input.
		1	Receive clock divider is running.

14.3.13 Receive Format Unit Bit Mask Register (RMASK)

The receive format unit bit mask register (RMASK) determines which bits of the received data are masked off and padded with a known value before being read by the CPU or DMA. The RMASK is shown in Figure 14-49 and described in Table 14-22.

Figure 14-49. Receive Format Unit Bit Mask Register (RMASK)

31	0
RMASK[*n*]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 14-22. Receive Format Unit Bit Mask Register (RMASK) Field Descriptions

Bit	Field	Value	Description
31-0	RMASK[31-0]		Receive data mask *n* enable bit.
		0	Corresponding bit of receive data (after passing through reverse and rotate units) is masked out and then padded with the selected bit pad value (RPAD and RPBIT bits in RFMT).
		1	Corresponding bit of receive data (after passing through reverse and rotate units) is returned to CPU or DMA.

14.3.14 Receive Bit Stream Format Register (RFMT)

The receive bit stream format register (RFMT) configures the receive data format. The RFMT is shown in Figure 14-50 and described in Table 14-23.

Figure 14-50. Receive Bit Stream Format Register (RFMT)

31	18	17	16
Reserved		RDATDLY	
R-0		R/W-0	

15	14	13	12	8	7	4	3	2	0
RRVRS	RPAD		RPBIT		RSSZ		RBUSEL	RROT	
R/W-0	R/W-0		R/W-0		R/W-0		R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-23. Receive Bit Stream Format Register (RFMT) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
17-16	RDATDLY	0-3h	Receive frame sync delay of AXRn.
		0	0-bit delay. The first receive data bit, AXRn, occurs in same ACLKR cycle as the receive frame sync (AFSR).
		1h	1-bit delay. The first receive data bit, AXRn, occurs one ACLKR cycle after the receive frame sync (AFSR).
		2h	2-bit delay. The first receive data bit, AXRn, occurs two ACLKR cycles after the receive frame sync (AFSR).
		3h	Reserved.
15	RRVRS		Receive serial bitstream order.
		0	Bitstream is LSB first. No bit reversal is performed in receive format bit reverse unit.
		1	Bitstream is MSB first. Bit reversal is performed in receive format bit reverse unit.
14-13	RPAD	0-3h	Pad value for extra bits in slot not belonging to the word. This field only applies to bits when RMASK[n] = 0.
		0	Pad extra bits with 0.
		1h	Pad extra bits with 1.
		2h	Pad extra bits with one of the bits from the word as specified by RPBIT bits.
		3h	Reserved.
12-8	RPBIT	0-1Fh	RPBIT value determines which bit (as read by the CPU or DMA from RBUF[n]) is used to pad the extra bits. This field only applies when RPAD = 2h.
		0	Pad with bit 0 value.
		1h-1Fh	Pad with bit 1 to bit 31 value.

Table 14-23. Receive Bit Stream Format Register (RFMT) Field Descriptions (continued)

Bit	Field	Value	Description
7-4	RSSZ	0-Fh	Receive slot size.
		0-2h	Reserved
		3h	Slot size is 8 bits.
		4h	Reserved
		5h	Slot size is 12 bits.
		6h	Reserved
		7h	Slot size is 16 bits.
		8h	Reserved
		9h	Slot size is 20 bits.
		Ah	Reserved
		Bh	Slot size is 24 bits
		Ch	Reserved
		Dh	Slot size is 28 bits.
		Eh	Reserved
		Fh	Slot size is 32 bits.
3	RBUSEL		Selects whether reads from serializer buffer XRBUF[n] originate from the configuration bus (CFG) or the data (DAT) port.
		0	Reads from XRBUF[n] originate on data port. Reads from XRBUF[n] on configuration bus are ignored.
		1	Reads from XRBUF[n] originate on configuration bus. Reads from XRBUF[n] on data port are ignored.
2-0	RROT	0-7h	Right-rotation value for receive rotate right format unit.
		0	Rotate right by 0 (no rotation).
		1h	Rotate right by 4 bit positions.
		2h	Rotate right by 8 bit positions.
		3h	Rotate right by 12 bit positions.
		4h	Rotate right by 16 bit positions.
		5h	Rotate right by 20 bit positions.
		6h	Rotate right by 24 bit positions.
		7h	Rotate right by 28 bit positions.

14.3.15 Receive Frame Sync Control Register (AFSRCTL)

The receive frame sync control register (AFSRCTL) configures the receive frame sync (AFSR). The AFSRCTL is shown in Figure 14-51 and described in Table 14-24.

Figure 14-51. Receive Frame Sync Control Register (AFSRCTL)

31	16
Reserved	
R-0	

15	7	6	5	4	3	2	1	0
RMOD		Reserved		FRWID	Reserved		FSRM	FSRP
R/W-0		R-0		R/W-0	R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-24. Receive Frame Sync Control Register (AFSRCTL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-7	RMOD	0-1FFh	Receive frame sync mode select bits.
		0	Burst mode.
		1h	Reserved.
		2h-20h	2-slot TDM (I2S mode) to 32-slot TDM.
		21h-17Fh	Reserved.
		180h	384-slot TDM (external DIR IC inputting 384-slot DIR frames to McASP over I2S interface).
		181h-1FFh	Reserved.
6-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	FRWID		Receive frame sync width select bit indicates the width of the receive frame sync (AFSR) during its active period.
		0	Single bit.
		1	Single word. Single word is not supported if RMOD is set to burst mode.
3-2	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
1	FSRM		Receive frame sync generation select bit.
		0	Externally-generated receive frame sync.
		1	Internally-generated receive frame sync.
0	FSRP		Receive frame sync polarity select bit.
		0	A rising edge on receive frame sync (AFSR) indicates the beginning of a frame.
		1	A falling edge on receive frame sync (AFSR) indicates the beginning of a frame.

14.3.16 Receive Clock Control Register (ACLKRCTL)

The receive clock control register (ACLKRCTL) configures the receive bit clock (ACLKR) and the receive clock generator. The ACLKRCTL is shown in Figure 14-52 and described in Table 14-25.

Figure 14-52. Receive Clock Control Register (ACLKRCTL)

31						16
Reserved						
R-0						
15	**8**	**7**	**6**	**5**	**4**	**0**
Reserved		CLKRP	Rsvd	CLKRM	CLKRDIV	
R-0		R/W-0	R-0	R/W-1	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

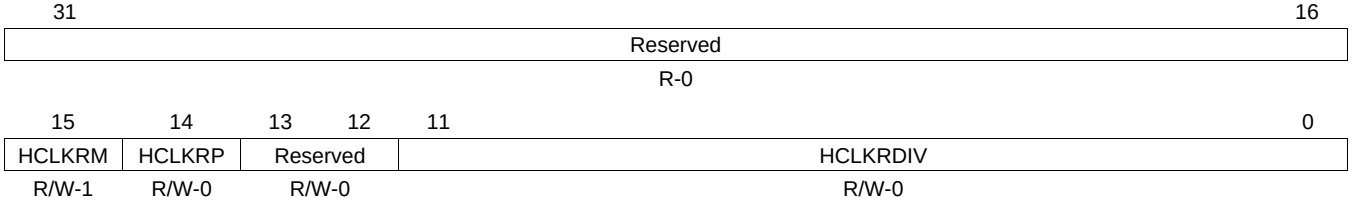
Table 14-25. Receive Clock Control Register (ACLKRCTL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7	CLKRP		Receive bitstream clock polarity select bit.
		0	Falling edge. Receiver samples data on the falling edge of the serial clock, so the external transmitter driving this receiver must shift data out on the rising edge of the serial clock.
		1	Rising edge. Receiver samples data on the rising edge of the serial clock, so the external transmitter driving this receiver must shift data out on the falling edge of the serial clock.
6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	CLKRM		Receive bit clock source bit. Note that this bit does not have any effect, if ACLKXCTL.ASYNC = 0.
		0	External receive clock source from ACLKR pin.
		1	Internal receive clock source from output of programmable bit clock divider.
4-0	CLKRDIV	0-1Fh	Receive bit clock divide ratio bits determine the divide-down ratio from AHCLKR to ACLKR. Note that this bit does not have any effect, if ACLKXCTL.ASYNC = 0.
		0	Divide-by-1.
		1h	Divide-by-2.
		2h-1Fh	Divide-by-3 to divide-by-32.

14.3.17 Receive High-Frequency Clock Control Register (AHCLKRCTL)

The receive high-frequency clock control register (AHCLKRCTL) configures the receive high-frequency master clock (AHCLKR) and the receive clock generator. The AHCLKRCTL is shown in Figure 14-53 and described in Table 14-26.

Figure 14-53. Receive High-Frequency Clock Control Register (AHCLKRCTL)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

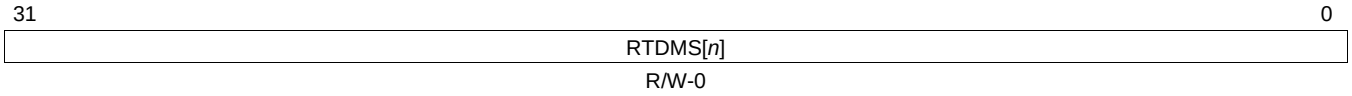
Table 14-26. Receive High-Frequency Clock Control Register (AHCLKRCTL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15	HCLKRM		Receive high-frequency clock source bit.
		0	External receive high-frequency clock source from AHCLKR pin.
		1	Internal receive high-frequency clock source from output of programmable high clock divider.
14	HCLKRP		Receive bitstream high-frequency clock polarity select bit.
		0	AHCLKR is not inverted before programmable bit clock divider. In the special case where the receive bit clock (ACLKR) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKRDIV = 0 in ACLKRCTL), AHCLKR is directly passed through to the ACLKR pin.
		1	AHCLKR is inverted before programmable bit clock divider. In the special case where the receive bit clock (ACLKR) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKRDIV = 0 in ACLKRCTL), AHCLKR is directly passed through to the ACLKR pin.
13-12	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
11-0	HCLKRDIV	0-FFFh	Receive high-frequency clock divide ratio bits determine the divide-down ratio from AUXCLK to AHCLKR.
		0	Divide-by-1.
		1h	Divide-by-2.
		2h-FFFh	Divide-by-3 to divide-by-4096.

14.3.18 Receive TDM Time Slot Register (RTDM)

The receive TDM time slot register (RTDM) specifies which TDM time slot the receiver is active. The RTDM is shown in Figure 14-54 and described in Table 14-27.

Figure 14-54. Receive TDM Time Slot Register (RTDM)



LEGEND: R/W = Read/Write; -*n* = value after reset

Table 14-27. Receive TDM Time Slot Register (RTDM) Field Descriptions

Bit	Field	Value	Description
31-0	RTDMS[31-0]		Receiver mode during TDM time slot *n*.
		0	Receive TDM time slot *n* is inactive. The receive serializer does not shift in data during this slot.
		1	Receive TDM time slot *n* is active. The receive serializer shifts in data during this slot.

14.3.19 Receiver Interrupt Control Register (RINTCTL)

The receiver interrupt control register (RINTCTL) controls generation of the McASP receive interrupt (RINT). When the register bit(s) is set to 1, the occurrence of the enabled McASP condition(s) generates RINT. The RINTCTL is shown in Figure 14-55 and described in Table 14-28. See Section 14.3.20 for a description of the interrupt conditions.

Figure 14-55. Receiver Interrupt Control Register (RINTCTL)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
RSTAFRM	Reserved	RDATA	RLAST	RDMAERR	RCKFAIL	RSYNCERR	ROVRN
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-28. Receiver Interrupt Control Register (RINTCTL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7	RSTAFRM		Receive start of frame interrupt enable bit.
		0	Interrupt is disabled. A receive start of frame interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive start of frame interrupt generates a McASP receive interrupt (RINT).
6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	RDATA		Receive data ready interrupt enable bit.
		0	Interrupt is disabled. A receive data ready interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive data ready interrupt generates a McASP receive interrupt (RINT).
4	RLAST		Receive last slot interrupt enable bit.
		0	Interrupt is disabled. A receive last slot interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive last slot interrupt generates a McASP receive interrupt (RINT).
3	RDMAERR		Receive DMA error interrupt enable bit.
		0	Interrupt is disabled. A receive DMA error interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive DMA error interrupt generates a McASP receive interrupt (RINT).
2	RCKFAIL		Receive clock failure interrupt enable bit.
		0	Interrupt is disabled. A receive clock failure interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive clock failure interrupt generates a McASP receive interrupt (RINT).
1	RSYNCERR		Unexpected receive frame sync interrupt enable bit.
		0	Interrupt is disabled. An unexpected receive frame sync interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. An unexpected receive frame sync interrupt generates a McASP receive interrupt (RINT).
0	ROVRN		Receiver overrun interrupt enable bit.
		0	Interrupt is disabled. A receiver overrun interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receiver overrun interrupt generates a McASP receive interrupt (RINT).

14.3.20 Receiver Status Register (RSTAT)

The receiver status register (RSTAT) provides the receiver status and receive TDM time slot number. If the McASP logic attempts to set an interrupt flag in the same cycle that the CPU writes to the flag to clear it, the McASP logic has priority and the flag remains set. This also causes a new interrupt request to be generated. The RSTAT is shown in Figure 14-56 and described in Table 14-29.

Figure 14-56. Receiver Status Register (RSTAT)

31						9	8
Reserved							RERR
R-0							R/W-0

7	6	5	4	3	2	1	0
RDMAERR	RSTAFRM	RDATA	RLAST	RTDMSLOT	RCKFAIL	RSYNCERR	ROVRN
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Writing a 1 clears this bit; writing a 0 has no effect.; -*n* = value after reset

Table 14-29. Receiver Status Register (RSTAT) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8	RERR		RERR bit always returns a logic-OR of:ROVRN | RSYNCERR | RCKFAIL | RDMAERR Allows a single bit to be checked to determine if a receiver error interrupt has occurred.
		0	No errors have occurred.
		1	An error has occurred.
7	RDMAERR		Receive DMA error flag. RDMAERR is set when the CPU or DMA reads more serializers through the data port in a given time slot than were programmed as receivers. Causes a receive interrupt (RINT), if this bit is set and RDMAERR in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	Receive DMA error did not occur.
		1	Receive DMA error did occur.
6	RSTAFRM		Receive start of frame flag. Causes a receive interrupt (RINT), if this bit is set and RSTAFRM in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	No new receive frame sync (AFSR) is detected.
		1	A new receive frame sync (AFSR) is detected.
5	RDATA		Receive data ready flag. Causes a receive interrupt (RINT), if this bit is set and RDATA in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	No new data in RBUF.
		1	Data is transferred from XRSR to RBUF and ready to be serviced by the CPU or DMA. When RDATA is set, it always causes a DMA event (AREVT).
4	RLAST		Receive last slot flag. RLAST is set along with RDATA, if the current slot is the last slot in a frame. Causes a receive interrupt (RINT), if this bit is set and RLAST in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	Current slot is not the last slot in a frame.
		1	Current slot is the last slot in a frame. RDATA is also set.
3	RTDMSLOT		Returns the LSB of RSLOT. Allows a single read of RSTAT to determine whether the current TDM time slot is even or odd.
		0	Current TDM time slot is odd.
		1	Current TDM time slot is even.
2	RCKFAIL		Receive clock failure flag. RCKFAIL is set when the receive clock failure detection circuit reports an error (see *Clock Failure Detection*). Causes a receive interrupt (RINT), if this bit is set and RCKFAIL in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	Receive clock failure did not occur.
		1	Receive clock failure did occur.

Table 14-29. Receiver Status Register (RSTAT) Field Descriptions (continued)

Bit	Field	Value	Description
1	RSYNCERR		Unexpected receive frame sync flag. RSYNCERR is set when a new receive frame sync (AFSR) occurs before it is expected. Causes a receive interrupt (RINT), if this bit is set and RSYNCERR in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	Unexpected receive frame sync did not occur.
		1	Unexpected receive frame sync did occur.
0	ROVRN		Receiver overrun flag. ROVRN is set when the receive serializer is instructed to transfer data from XRSR to RBUF, but the former data in RBUF has not yet been read by the CPU or DMA. Causes a receive interrupt (RINT), if this bit is set and ROVRN in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	Receiver overrun did not occur.
		1	Receiver overrun did occur.

14.3.21 Current Receive TDM Time Slot Registers (RSLOT)

The current receive TDM time slot register (RSLOT) indicates the current time slot for the receive data frame. The RSLOT is shown in Figure 14-57 and described in Table 14-30.

Figure 14-57. Current Receive TDM Time Slot Registers (RSLOT)

31	16
Reserved	
R-0	

15	9	8	0
Reserved		RSLOTCNT	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 14-30. Current Receive TDM Time Slot Registers (RSLOT) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8-0	RSLOTCNT	0-17Fh	Current receive time slot count. Legal values: 0 to 383 (17Fh). TDM function is not supported for > 32 time slots. However, TDM time slot counter may count to 383 when used to receive a DIR block (transferred over TDM format). In I2S mode, this should only be read by the DSP since frame sync initializes it to zero.

14.3.22 Receive Clock Check Control Register (RCLKCHK)

The receive clock check control register (RCLKCHK) configures the receive clock failure detection circuit. The RCLKCHK is shown in Figure 14-58 and described in Table 14-31.

Figure 14-58. Receive Clock Check Control Register (RCLKCHK)

31	24	23	16
RCNT		RMAX	
R-0		R/W-0	

15	8	7	4	3	0
RMIN		Reserved		RPS	
R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-31. Receive Clock Check Control Register (RCLKCHK) Field Descriptions

Bit	Field	Value	Description
31-24	RCNT	0-FFh	Receive clock count value (from previous measurement). The clock circuit continually counts the number of system clocks for every 32 receive high-frequency master clock (AHCLKR) signals, and stores the count in RCNT until the next measurement is taken.
23-16	RMAX	0-FFh	Receive clock maximum boundary. This 8-bit unsigned value sets the maximum allowed boundary for the clock check counter after 32 receive high-frequency master clock (AHCLKR) signals have been received. If the current counter value is greater than RMAX after counting 32 AHCLKR signals, RCKFAIL in RSTAT is set. The comparison is performed using unsigned arithmetic.
15-8	RMIN	0-FFh	Receive clock minimum boundary. This 8-bit unsigned value sets the minimum allowed boundary for the clock check counter after 32 receive high-frequency master clock (AHCLKR) signals have been received. If RCNT is less than RMIN after counting 32 AHCLKR signals, RCKFAIL in RSTAT is set. The comparison is performed using unsigned arithmetic.
7-4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3-0	RPS	0-Fh	Receive clock check prescaler value.
		0	McASP system clock divided by 1.
		1h	McASP system clock divided by 2.
		2h	McASP system clock divided by 4.
		3h	McASP system clock divided by 8.
		4h	McASP system clock divided by 16.
		5h	McASP system clock divided by 32.
		6h	McASP system clock divided by 64.
		7h	McASP system clock divided by 128.
		8h	McASP system clock divided by 256.
		9h-Fh	Reserved.

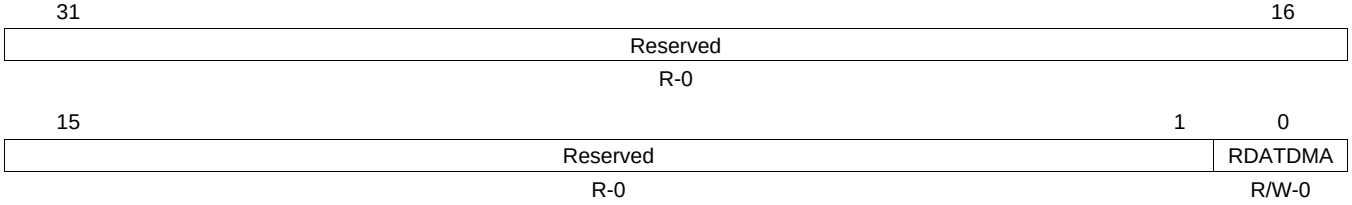
14.3.23 Receiver DMA Event Control Register (REVTCTL)

The receiver DMA event control register (REVTCTL) contains a disable bit for the receiver DMA event. The REVTCTL is shown in Figure 14-59 and described in Table 14-32.

NOTE: Device-specific registers

Accessing REVTCTL not implemented on a specific device may cause improper device operation.

Figure 14-59. Receiver DMA Event Control Register (REVTCTL)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-32. Receiver DMA Event Control Register (REVTCTL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
0	RDATDMA		Receive data DMA request enable bit. If writing to this bit, always write the default value of 0.
		0	Receive data DMA request is enabled.
		1	Reserved

14.3.24 Transmitter Global Control Register (XGBLCTL)

Alias of the global control register (GBLCTL). Writing to the transmitter global control register (XGBLCTL) affects only the transmit bits of GBLCTL (bits 12-8). Reads from XGBLCTL return the value of GBLCTL. XGBLCTL allows the transmitter to be reset independently from the receiver. The XGBLCTL is shown in Figure 14-60 and described in Table 14-33. See Section 14.3.8 for a detailed description of GBLCTL.

Figure 14-60. Transmitter Global Control Register (XGBLCTL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved			XFRST	XSMRST	XSRCLR	XHCLKRST	XCLKRST
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved			RFRST	RSMRST	RSRCLR	RHCLKRST	RCLKRST
R-0			R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-33. Transmitter Global Control Register (XGBLCTL) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0-FFh	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	XFRST		Transmit frame sync generator reset enable bit. A write to this bit affects the XFRST bit of GBLCTL.
		0	Transmit frame sync generator is reset.
		1	Transmit frame sync generator is active.
11	XSMRST		Transmit state machine reset enable bit. A write to this bit affects the XSMRST bit of GBLCTL.
		0	Transmit state machine is held in reset.
		1	Transmit state machine is released from reset.
10	XSRCLR		Transmit serializer clear enable bit. A write to this bit affects the XSRCLR bit of GBLCTL.
		0	Transmit serializers are cleared.
		1	Transmit serializers are active.
9	XHCLKRST		Transmit high-frequency clock divider reset enable bit. A write to this bit affects the XHCLKRST bit of GBLCTL.
		0	Transmit high-frequency clock divider is held in reset and passes through its input as divide-by-1.
		1	Transmit high-frequency clock divider is running.
8	XCLKRST		Transmit clock divider reset enable bit. A write to this bit affects the XCLKRST bit of GBLCTL.
		0	Transmit clock divider is held in reset. When the clock divider is in reset, it passes through a divide-by-1 of its input.
		1	Transmit clock divider is running.
7-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	RFRST	x	Receive frame sync generator reset enable bit. A read of this bit returns the RFRST bit value of GBLCTL. Writes have no effect.
3	RSMRST	x	Receive state machine reset enable bit. A read of this bit returns the RSMRST bit value of GBLCTL. Writes have no effect.
2	RSRCLR	x	Receive serializer clear enable bit. A read of this bit returns the RSRSCLR bit value of GBLCTL. Writes have no effect.
1	RHCLKRST	x	Receive high-frequency clock divider reset enable bit. A read of this bit returns the RHCLKRST bit value of GBLCTL. Writes have no effect.
0	RCLKRST	x	Receive clock divider reset enable bit. A read of this bit returns the RCLKRST bit value of GBLCTL. Writes have no effect.

14.3.25 Transmit Format Unit Bit Mask Register (XMASK)

The transmit format unit bit mask register (XMASK) determines which bits of the transmitted data are masked off and padded with a known value before being shifted out the McASP. The XMASK is shown in Figure 14-61 and described in Table 14-34.

Figure 14-61. Transmit Format Unit Bit Mask Register (XMASK)

31	0
XMASK[*n*]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 14-34. Transmit Format Unit Bit Mask Register (XMASK) Field Descriptions

Bit	Field	Value	Description
31-0	XMASK[31-0]		Transmit data mask *n* enable bit.
		0	Corresponding bit of transmit data (before passing through reverse and rotate units) is masked out and then padded with the selected bit pad value (XPAD and XPBIT bits in XFMT), which is transmitted out the McASP in place of the original bit.
		1	Corresponding bit of transmit data (before passing through reverse and rotate units) is transmitted out the McASP.

14.3.26 Transmit Bit Stream Format Register (XFMT)

The transmit bit stream format register (XFMT) configures the transmit data format. The XFMT is shown in Figure 14-62 and described in Table 14-35.

Figure 14-62. Transmit Bit Stream Format Register (XFMT)

31	18	17	16
Reserved		XDATDLY	
R-0		R/W-0	

15	14	13	12	8	7	4	3	2	0
XRVRS	XPAD		XPBIT		XSSZ		XBUSEL	XROT	
R/W-0	R/W-0		R/W-0		R/W-0		R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-35. Transmit Bit Stream Format Register (XFMT) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
17-16	XDATDLY	0-3h	Transmit frame sync delay of AXRn.
		0	0-bit delay. The first transmit data bit, AXRn, occurs in same ACLKX cycle as the transmit frame sync (AFSX).
		1h	1-bit delay. The first transmit data bit, AXRn, occurs one ACLKX cycle after the transmit frame sync (AFSX).
		2h	2-bit delay. The first transmit data bit, AXRn, occurs two ACLKX cycles after the transmit frame sync (AFSX).
		3h	Reserved.
15	XRVRS		Transmit serial bitstream order.
		0	Bitstream is LSB first. No bit reversal is performed in transmit format bit reverse unit.
		1	Bitstream is MSB first. Bit reversal is performed in transmit format bit reverse unit.
14-13	XPAD	0-3h	Pad value for extra bits in slot not belonging to word defined by XMASK. This field only applies to bits when XMASK[n] = 0.
		0	Pad extra bits with 0.
		1h	Pad extra bits with 1.
		2h	Pad extra bits with one of the bits from the word as specified by XPBIT bits.
		3h	Reserved.
12-8	XPBIT	0-1Fh	XPBIT value determines which bit (as written by the CPU or DMA to XBUF[n]) is used to pad the extra bits before shifting. This field only applies when XPAD = 2h.
		0	Pad with bit 0 value.
		1-1Fh	Pad with bit 1 to bit 31 value.

Table 14-35. Transmit Bit Stream Format Register (XFMT) Field Descriptions (continued)

Bit	Field	Value	Description
7-4	XSSZ	0-Fh	Transmit slot size.
		0-2h	Reserved.
		3h	Slot size is 8 bits.
		4h	Reserved.
		5h	Slot size is 12 bits.
		6h	Reserved.
		7h	Slot size is 16 bits.
		8h	Reserved.
		9h	Slot size is 20 bits.
		Ah	Reserved.
		Bh	Slot size is 24 bits.
		Ch	Reserved.
		Dh	Slot size is 28 bits.
		Eh	Reserved.
		Fh	Slot size is 32 bits.
3	XBUSEL		Selects whether writes to serializer buffer XRBUF[n] originate from the configuration bus (CFG) or the data (DAT) port.
		0	Writes to XRBUF[n] originate from the data port. Writes to XRBUF[n] from the configuration bus are ignored with no effect to the McASP.
		1	Writes to XRBUF[n] originate from the configuration bus. Writes to XRBUF[n] from the data port are ignored with no effect to the McASP.
2-0	XROT	0-7h	Right-rotation value for transmit rotate right format unit.
		0	Rotate right by 0 (no rotation).
		1h	Rotate right by 4 bit positions.
		2h	Rotate right by 8 bit positions.
		3h	Rotate right by 12 bit positions.
		4h	Rotate right by 16 bit positions.
		5h	Rotate right by 20 bit positions.
		6h	Rotate right by 24 bit positions.
		7h	Rotate right by 28 bit positions.

14.3.27 Transmit Frame Sync Control Register (AFSXCTL)

The transmit frame sync control register (AFSXCTL) configures the transmit frame sync (AFSX). The AFSXCTL is shown in Figure 14-63 and described in Table 14-36.

Figure 14-63. Transmit Frame Sync Control Register (AFSXCTL)

31	16
Reserved	
R-0	

15	7	6	5	4	3	2	1	0
XMOD		Reserved		FXWID	Reserved		FSXM	FSXP
R/W-0		R-0		R/W-0	R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-36. Transmit Frame Sync Control Register (AFSXCTL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-7	XMOD	0-1FFh	Transmit frame sync mode select bits.
		0	Burst mode.
		1h	Reserved.
		2h-20h	2-slot TDM (I2S mode) to 32-slot TDM.
		21h-17Fh	Reserved.
		180h	384-slot DIT mode.
		181h-1FFh	Reserved.
6-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	FXWID		Transmit frame sync width select bit indicates the width of the transmit frame sync (AFSX) during its active period.
		0	Single bit.
		1	Single word. Single word is not supported if XMOD is set to burst mode.
3-2	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
1	FSXM		Transmit frame sync generation select bit.
		0	Externally-generated transmit frame sync.
		1	Internally-generated transmit frame sync.
0	FSXP		Transmit frame sync polarity select bit.
		0	A rising edge on transmit frame sync (AFSX) indicates the beginning of a frame.
		1	A falling edge on transmit frame sync (AFSX) indicates the beginning of a frame.

14.3.28 Transmit Clock Control Register (ACLKXCTL)

The transmit clock control register (ACLKXCTL) configures the transmit bit clock (ACLKX) and the transmit clock generator. The ACLKXCTL is shown in Figure 14-64 and described in Table 14-37.

Figure 14-64. Transmit Clock Control Register (ACLKXCTL)

31	16
Reserved	
R-0	

15	8	7	6	5	4	0
Reserved		CLKXP	ASYNC	CLKXM	CLKXDIV	
R-0		R/W-0	R/W-1	R/W-1	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-37. Transmit Clock Control Register (ACLKXCTL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7	CLKXP		Transmit bitstream clock polarity select bit.
		0	Rising edge. External receiver samples data on the falling edge of the serial clock, so the transmitter must shift data out on the rising edge of the serial clock.
		1	Falling edge. External receiver samples data on the rising edge of the serial clock, so the transmitter must shift data out on the falling edge of the serial clock.
6	ASYNC		Transmit/receive operation asynchronous enable bit.
		0	Synchronous. Transmit clock and frame sync provides the source for both the transmit and receive sections. Note that in this mode, the receive bit clock is an inverted version of the transmit bit clock.
		1	Asynchronous. Separate clock and frame sync used by transmit and receive sections.
5	CLKXM		Transmit bit clock source bit.
		0	External transmit clock source from ACLKX pin.
		1	Internal transmit clock source from output of programmable bit clock divider.
4-0	CLKXDIV	0-1Fh	Transmit bit clock divide ratio bits determine the divide-down ratio from AHCLKX to ACLKX.
		0	Divide-by-1.
		1h	Divide-by-2.
		2h-1Fh	Divide-by-3 to divide-by-32.

14.3.29 Transmit High-Frequency Clock Control Register (AHCLKXCTL)

The transmit high-frequency clock control register (AHCLKXCTL) configures the transmit high-frequency master clock (AHCLKX) and the transmit clock generator. The AHCLKXCTL is shown in Figure 14-65 and described in Table 14-38.

Figure 14-65. Transmit High-Frequency Clock Control Register (AHCLKXCTL)

31					16
Reserved					
R-0					

15	14	13	12	11	0
HCLKXM	HCLKXP	Reserved		HCLKXDIV	
R/W-1	R/W-0	R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-38. Transmit High-Frequency Clock Control Register (AHCLKXCTL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15	HCLKXM		Transmit high-frequency clock source bit.
		0	External transmit high-frequency clock source from AHCLKX pin.
		1	Internal transmit high-frequency clock source from output of programmable high clock divider.
14	HCLKXP		Transmit bitstream high-frequency clock polarity select bit.
		0	AHCLKX is not inverted before programmable bit clock divider. In the special case where the transmit bit clock (ACLKX) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKXDIV = 0 in ACLKXCTL), AHCLKX is directly passed through to the ACLKX pin.
		1	AHCLKX is inverted before programmable bit clock divider. In the special case where the transmit bit clock (ACLKX) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKXDIV = 0 in ACLKXCTL), AHCLKX is directly passed through to the ACLKX pin.
13-12	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
11-0	HCLKXDIV	0-FFFh	Transmit high-frequency clock divide ratio bits determine the divide-down ratio from AUXCLK to AHCLKX.
		0	Divide-by-1.
		1h	Divide-by-2.
		2h-FFFh	Divide-by-3 to divide-by-4096.

14.3.30 Transmit TDM Time Slot Register (XTDM)

The transmit TDM time slot register (XTDM) specifies in which TDM time slot the transmitter is active. TDM time slot counter range is extended to 384 slots (to support SPDIF blocks of 384 subframes). XTDM operates modulo 32, that is, XTDMS specifies the TDM activity for time slots 0, 32, 64, 96, 128, etc. The XTDM is shown in Figure 14-66 and described in Table 14-39.

Figure 14-66. Transmit TDM Time Slot Register (XTDM)

31	0
XTDMS[*n*]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 14-39. Transmit TDM Time Slot Register (XTDM) Field Descriptions

Bit	Field	Value	Description
31-0	XTDMS[31-0]		Transmitter mode during TDM time slot *n*.
		0	Transmit TDM time slot *n* is inactive. The transmit serializer does not shift out data during this slot.
		1	Transmit TDM time slot *n* is active. The transmit serializer shifts out data during this slot according to the serializer control register (SRCTL).

14.3.31 Transmitter Interrupt Control Register (XINTCTL)

The transmitter interrupt control register (XINTCTL) controls generation of the McASP transmit interrupt (XINT). When the register bit(s) is set to 1, the occurrence of the enabled McASP condition(s) generates XINT. The XINTCTL is shown in Figure 14-67 and described in Table 14-40. See Section 14.3.32 for a description of the interrupt conditions.

Figure 14-67. Transmitter Interrupt Control Register (XINTCTL)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
XSTAFRM	Reserved	XDATA	XLAST	XDMAERR	XCKFAIL	XSYNCERR	XUNDRN
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-40. Transmitter Interrupt Control Register (XINTCTL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7	XSTAFRM		Transmit start of frame interrupt enable bit.
		0	Interrupt is disabled. A transmit start of frame interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit start of frame interrupt generates a McASP transmit interrupt (XINT).
6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	XDATA		Transmit data ready interrupt enable bit.
		0	Interrupt is disabled. A transmit data ready interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit data ready interrupt generates a McASP transmit interrupt (XINT).
4	XLAST		Transmit last slot interrupt enable bit.
		0	Interrupt is disabled. A transmit last slot interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit last slot interrupt generates a McASP transmit interrupt (XINT).
3	XDMAERR		Transmit DMA error interrupt enable bit.
		0	Interrupt is disabled. A transmit DMA error interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit DMA error interrupt generates a McASP transmit interrupt (XINT).
2	XCKFAIL		Transmit clock failure interrupt enable bit.
		0	Interrupt is disabled. A transmit clock failure interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit clock failure interrupt generates a McASP transmit interrupt (XINT).
1	XSYNCERR		Unexpected transmit frame sync interrupt enable bit.
		0	Interrupt is disabled. An unexpected transmit frame sync interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. An unexpected transmit frame sync interrupt generates a McASP transmit interrupt (XINT).
0	XUNDRN		Transmitter underrun interrupt enable bit.
		0	Interrupt is disabled. A transmitter underrun interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmitter underrun interrupt generates a McASP transmit interrupt (XINT).

14.3.32 Transmitter Status Register (XSTAT)

The transmitter status register (XSTAT) provides the transmitter status and transmit TDM time slot number. If the McASP logic attempts to set an interrupt flag in the same cycle that the CPU writes to the flag to clear it, the McASP logic has priority and the flag remains set. This also causes a new interrupt request to be generated. The XSTAT is shown in Figure 14-68 and described in Table 14-41.

Figure 14-68. Transmitter Status Register (XSTAT)

31							9	8
Reserved								XERR
R-0								R/W-0
7	6	5	4	3	2	1		0
XDMAERR	XSTAFRM	XDATA	XLAST	XTDMSLOT	XCKFAIL	XSYNCERR		XUNDRN
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R-0	R/W1C-0	R/W1C-0		R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Writing a 1 clears this bit; writing a 0 has no effect; -*n* = value after reset

Table 14-41. Transmitter Status Register (XSTAT) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8	XERR		XERR bit always returns a logic-OR of:XUNDRN | XSYNCERR | XCKFAIL | XDMAERR Allows a single bit to be checked to determine if a transmitter error interrupt has occurred.
		0	No errors have occurred.
		1	An error has occurred.
7	XDMAERR		Transmit DMA error flag. XDMAERR is set when the CPU or DMA writes more serializers through the data port in a given time slot than were programmed as transmitters. Causes a transmit interrupt (XINT), if this bit is set and XDMAERR in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	Transmit DMA error did not occur.
		1	Transmit DMA error did occur.
6	XSTAFRM		Transmit start of frame flag. Causes a transmit interrupt (XINT), if this bit is set and XSTAFRM in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	No new transmit frame sync (AFSX) is detected.
		1	A new transmit frame sync (AFSX) is detected.
5	XDATA		Transmit data ready flag. Causes a transmit interrupt (XINT), if this bit is set and XDATA in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	XBUF is written and is full.
		1	Data is copied from XBUF to XRSR. XBUF is empty and ready to be written. XDATA is also set when the transmit serializers are taken out of reset. When XDATA is set, it always causes a DMA event (AXEVT).
4	XLAST		Transmit last slot flag. XLAST is set along with XDATA, if the current slot is the last slot in a frame. Causes a transmit interrupt (XINT), if this bit is set and XLAST in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	Current slot is not the last slot in a frame.
		1	Current slot is the last slot in a frame. XDATA is also set.
3	XTDMSLOT		Returns the LSB of XSLOT. Allows a single read of XSTAT to determine whether the current TDM time slot is even or odd.
		0	Current TDM time slot is odd.
		1	Current TDM time slot is even.
2	XCKFAIL		Transmit clock failure flag. XCKFAIL is set when the transmit clock failure detection circuit reports an error (see *Clock Failure Detection*). Causes a transmit interrupt (XINT), if this bit is set and XCKFAIL in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	Transmit clock failure did not occur.
		1	Transmit clock failure did occur.

Table 14-41. Transmitter Status Register (XSTAT) Field Descriptions (continued)

Bit	Field	Value	Description
1	XSYNCERR		Unexpected transmit frame sync flag. XSYNCERR is set when a new transmit frame sync (AFSX) occurs before it is expected. Causes a transmit interrupt (XINT), if this bit is set and XSYNCERR in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	Unexpected transmit frame sync did not occur.
		1	Unexpected transmit frame sync did occur.
0	XUNDRN		Transmitter underrun flag. XUNDRN is set when the transmit serializer is instructed to transfer data from XBUF to XRSR, but XBUF has not yet been serviced with new data since the last transfer. Causes a transmit interrupt (XINT), if this bit is set and XUNDRN in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	Transmitter underrun did not occur.
		1	Transmitter underrun did occur. For details on McASP action upon underrun conditions, see *Buffer Underrun Error - Transmitter*.

14.3.33 Current Transmit TDM Time Slot Register (XSLOT)

The current transmit TDM time slot register (XSLOT) indicates the current time slot for the transmit data frame. The XSLOT is shown in Figure 14-69 and described in Table 14-42.

Figure 14-69. Current Transmit TDM Time Slot Register (XSLOT)

31	16
Reserved	
R-0	

15	10	9	0
Reserved		XSLOTCNT	
R-0		R-17Fh	

LEGEND: R = Read only; -*n* = value after reset

Table 14-42. Current Transmit TDM Time Slot Register (XSLOT) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
9-0	XSLOTCNT	0-17Fh	Current transmit time slot count. Legal values: 0 to 383 (17Fh). During reset, this counter value is 383 so the next count value, which is used to encode the first DIT group of data, will be 0 and encodes the B preamble. TDM function is not supported for >32 time slots. However, TDM time slot counter may count to 383 when used to transmit a DIT block.

14.3.34 Transmit Clock Check Control Register (XCLKCHK)

The transmit clock check control register (XCLKCHK) configures the transmit clock failure detection circuit. The XCLKCHK is shown in Figure 14-70 and described in Table 14-43.

Figure 14-70. Transmit Clock Check Control Register (XCLKCHK)

31	24	23	16
XCNT		XMAX	
R-0		R/W-0	

15	8	7	4	3	0
XMIN		Reserved		XPS	
R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-43. Transmit Clock Check Control Register (XCLKCHK) Field Descriptions

Bit	Field	Value	Description
31-24	XCNT	0	Transmit clock count value (from previous measurement). The clock circuit continually counts the number of system clocks for every 32 transmit high-frequency master clock (AHCLKX) signals, and stores the count in XCNT until the next measurement is taken.
23-16	XMAX	0-FFh	Transmit clock maximum boundary. This 8-bit unsigned value sets the maximum allowed boundary for the clock check counter after 32 transmit high-frequency master clock (AHCLKX) signals have been received. If the current counter value is greater than XMAX after counting 32 AHCLKX signals, XCKFAIL in XSTAT is set. The comparison is performed using unsigned arithmetic.
15-8	XMIN	0-FFh	Transmit clock minimum boundary. This 8-bit unsigned value sets the minimum allowed boundary for the clock check counter after 32 transmit high-frequency master clock (AHCLKX) signals have been received. If XCNT is less than XMIN after counting 32 AHCLKX signals, XCKFAIL in XSTAT is set. The comparison is performed using unsigned arithmetic.
7-4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3-0	XPS	0-Fh	Transmit clock check prescaler value.
		0	McASP system clock divided by 1.
		1h	McASP system clock divided by 2.
		2h	McASP system clock divided by 4.
		3h	McASP system clock divided by 8.
		4h	McASP system clock divided by 16.
		5h	McASP system clock divided by 32.
		6h	McASP system clock divided by 64.
		7h	McASP system clock divided by 128.
		8h	McASP system clock divided by 256.
		9h-Fh	Reserved.

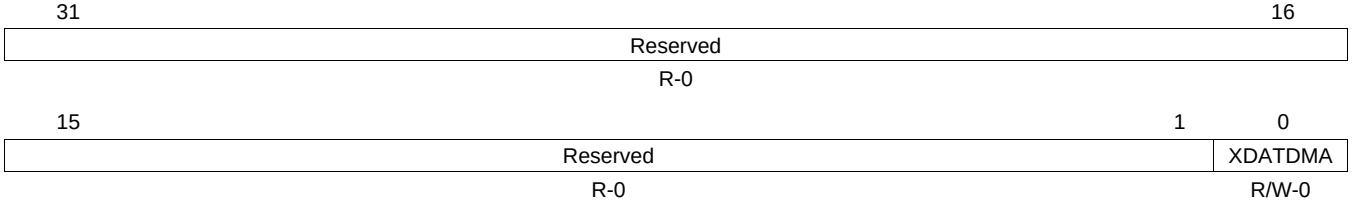
14.3.35 Transmitter DMA Event Control Register (XEVTCTL)

The transmitter DMA event control register (XEVTCTL) contains a disable bit for the transmit DMA event. The XEVTCTL is shown in Figure 14-71 and described in Table 14-44.

NOTE: Device-specific registers

Accessing REVTCTL not implemented on a specific device may cause improper device operation.

Figure 14-71. Transmitter DMA Event Control Register (XEVTCTL)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-44. Transmitter DMA Event Control Register (XEVTCTL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
0	XDATDMA		Transmit data DMA request enable bit. If writing to this bit, always write the default value of 0.
		0	Transmit data DMA request is enabled.
		1	Reserved

14.3.36 Serializer Control Registers (SRCTL*n*)

Each serializer on the McASP has a serializer control register (SRCTL). There are up to 16 serializers per McASP. The SRCTL is shown in Figure 14-72 and described in Table 14-45.

NOTE: Device-specific registers

Accessing SRCTL*n* not implemented on a specific device may cause improper device operation.

Figure 14-72. Serializer Control Registers (SRCTL*n*)

31	16
Reserved	
R-0	

15	6	5	4	3	2	1	0
Reserved		RRDY	XRDY	DISMOD		SRMOD	
R-0		R-0	R-0	R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-45. Serializer Control Registers (SRCTL*n*) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	RRDY		Receive buffer ready bit. RRDY indicates the current receive buffer state. Always reads 0 when programmed as a transmitter or as inactive. If SRMOD bit is set to receive (2h), RRDY switches from 0 to 1 whenever data is transferred from XRSR to RBUF.
		0	Receive buffer (RBUF) is empty.
		1	Receive buffer (RBUF) contains data and needs to be read before the start of the next time slot or a receiver overrun occurs.
4	XRDY		Transmit buffer ready bit. XRDY indicates the current transmit buffer state. Always reads 0 when programmed as a receiver or as inactive. If SRMOD bit is set to transmit (1h), XRDY switches from 0 to 1 when XSRCLR in GBLCTL is switched from 0 to 1 to indicate an empty transmitter. XRDY remains set until XSRCLR is forced to 0, data is written to the corresponding transmit buffer, or SRMOD bit is changed to receive (2h) or inactive (0).
		0	Transmit buffer (XBUF) contains data.
		1	Transmit buffer (XBUF) is empty and needs to be written before the start of the next time slot or a transmit underrun occurs.
3-2	DISMOD	0-3h	Serializer pin drive mode bit. Drive on pin when in inactive TDM slot of transmit mode or when serializer is inactive. This field only applies if the pin is configured as a McASP pin (PFUNC = 0).
		0	Drive on pin is 3-state.
		1h	Reserved.
		2h	Drive on pin is logic low.
		3h	Drive on pin is logic high.
1-0	SRMOD	0-3h	Serializer mode bit.
		0	Serializer is inactive.
		1h	Serializer is transmitter.
		2h	Serializer is receiver.
		3h	Reserved.

14.3.37 DIT Left Channel Status Registers (DITCSRA0-DITCSRA5)

The DIT left channel status registers (DITCSRA) provide the status of each left channel (even TDM time slot). Each of the six 32-bit registers (Figure 14-73) can store 192 bits of channel status data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register file in time, if a different set of data need to be sent.

Figure 14-73. DIT Left Channel Status Registers (DITCSRA0-DITCSRA5)

31	0
DITCSRA[*n*]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

14.3.38 DIT Right Channel Status Registers (DITCSRB0-DITCSRB5)

The DIT right channel status registers (DITCSRB) provide the status of each right channel (odd TDM time slot). Each of the six 32-bit registers (Figure 14-74) can store 192 bits of channel status data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register file in time, if a different set of data need to be sent.

Figure 14-74. DIT Right Channel Status Registers (DITCSRB0-DITCSRB5)

31	0
DITCSRB[*n*]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

14.3.39 DIT Left Channel User Data Registers (DITUDRA0-DITUDRA5)

The DIT left channel user data registers (DITUDRA) provides the user data of each left channel (even TDM time slot). Each of the six 32-bit registers (Figure 14-75) can store 192 bits of user data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register in time, if a different set of data need to be sent.

Figure 14-75. DIT Left Channel User Data Registers (DITUDRA0-DITUDRA5)

31	0
DITUDRA[*n*]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

14.3.40 DIT Right Channel User Data Registers (DITUDRB0-DITUDRB5)

The DIT right channel user data registers (DITUDRB) provides the user data of each right channel (odd TDM time slot). Each of the six 32-bit registers (Figure 14-76) can store 192 bits of user data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register in time, if a different set of data need to be sent.

Figure 14-76. DIT Right Channel User Data Registers (DITUDRB0-DITUDRB5)

31	0
DITUDRB[*n*]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

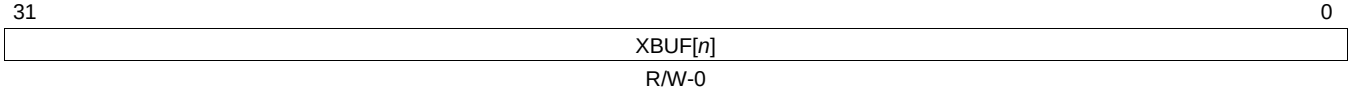
14.3.41 Transmit Buffer Registers (XBUF*n*)

The transmit buffers for the serializers (XBUF) hold data from the transmit format unit. For transmit operations, the XBUF (Figure 14-77) is an alias of the XRBUF in the serializer.

NOTE: Device-specific registers

Accessing XBUF registers not implemented on a specific device may cause improper device operation.

Figure 14-77. Transmit Buffer Registers (XBUFn)



LEGEND: R/W = Read/Write; -*n* = value after reset

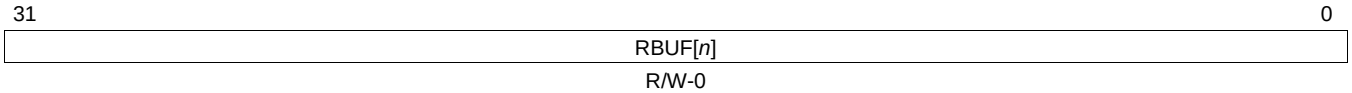
14.3.42 Receive Buffer Registers (RBUF*n*)

The receive buffers for the serializers (RBUF) hold data from the serializer before the data goes to the receive format unit. For receive operations, the RBUF (Figure 14-78) is an alias of the XRBUF in the serializer.

NOTE: Device-specific registers

Accessing RBUF registers not implemented on a specific device may cause improper device operation.

Figure 14-78. Receive Buffer Registers (RBUF*n*)



LEGEND: R/W = Read/Write; -*n* = value after reset

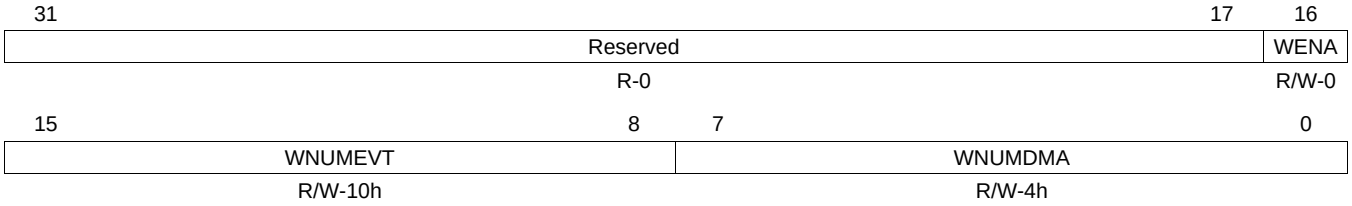
14.3.43 Write FIFO Control Register (WFIFOCTL)

Note that this register is only available if the Write FIFO is configured. The Write FIFO control register (WFIFOCTL) is shown in Figure 14-79 and described in Table 14-46.

NOTE: The WNUMEVT and WNUMDMA values must be set prior to enabling the Write FIFO.

If the Write FIFO is to be enabled, it must be enabled prior to taking the McASP out of reset.

Figure 14-79. Write FIFO Control Register (WFIFOCTL)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

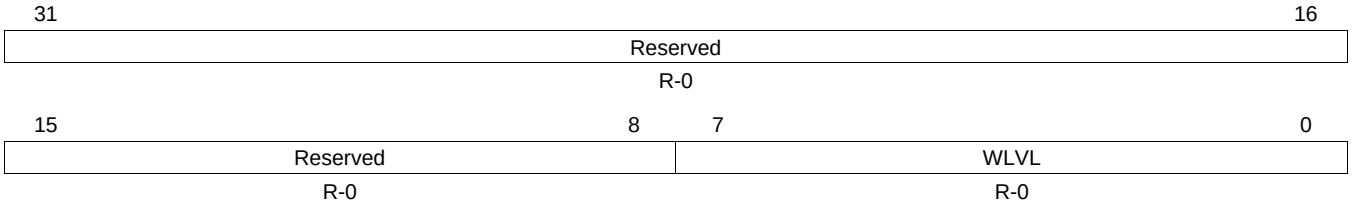
Table 14-46. Write FIFO Control Register (WFIFOCTL) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	WENA		Write FIFO enable bit.
		0	Write FIFO is disabled. The WLVL bit in the Write FIFO status register (WFIFOSTS) is reset to 0 and pointers are initialized, that is, the Write FIFO is “flushed.”
		1	Write FIFO is enabled. If Write FIFO is to be enabled, it must be enabled prior to taking McASP out of reset.
15-8	WNUMEVT	0-FFh	Write word count per DMA event (32-bit). When the Write FIFO has space for at least WNUMEVT words of data, then an AXEVT (transmit DMA event) is generated to the host/DMA controller. This value should be set to a non-zero integer multiple of the number of serializers enabled as transmitters. This value must be set prior to enabling the Write FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		3h-40h	3 to 64 words
		41h-FFh	Reserved
7-0	WNUMDMA	0-FFh	Write word count per transfer (32-bit words). Upon a transmit DMA event from the McASP, WNUMDMA words are transferred from the Write FIFO to the McASP. This value must equal the number of McASP serializers used as transmitters. This value must be set prior to enabling the Write FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		3h-10h	3-16 words
		11h-FFh	Reserved

14.3.44 Write FIFO Status Register (WFIFOSTS)

Note that this register is only available if the Write FIFO is configured. The Write FIFO status register (WFIFOSTS) is shown in Figure 14-80 and described in Table 14-47.

Figure 14-80. Write FIFO Status Register (WFIFOSTS)



LEGEND: R = Read only; -*n* = value after reset

Table 14-47. Write FIFO Status Register (WFIFOSTS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	WLVL	0-FFh	Write level (read-only). Number of 32-bit words currently in the Write FIFO.
		0	0 words currently in Write FIFO.
		1h	1 word currently in Write FIFO.
		2h	2 words currently in Write FIFO.
		3h-40h	3 to 64 words currently in Write FIFO.
		41h-FFh	Reserved

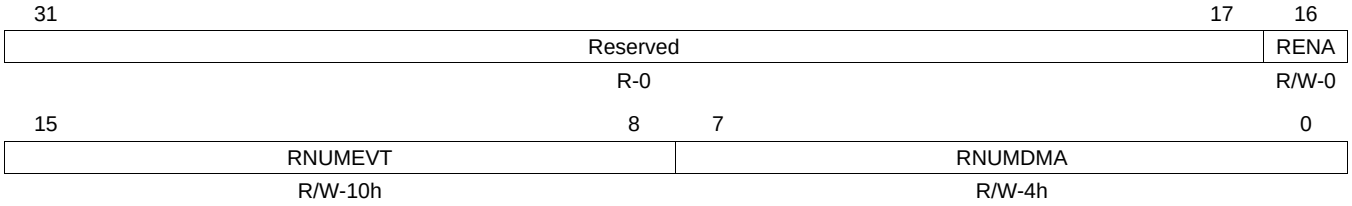
14.3.45 Read FIFO Control Register (RFIFOCTL)

Note that this register is only available if the Read FIFO is configured. The Read FIFO control register (RFIFOCTL) is shown in Figure 14-81 and described in Table 14-48.

NOTE: The RNUMEVT and RNUMDMA values must be set prior to enabling the Read FIFO.

If the Read FIFO is to be enabled, it must be enabled prior to taking the McASP out of reset.

Figure 14-81. Read FIFO Control Register (RFIFOCTL)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

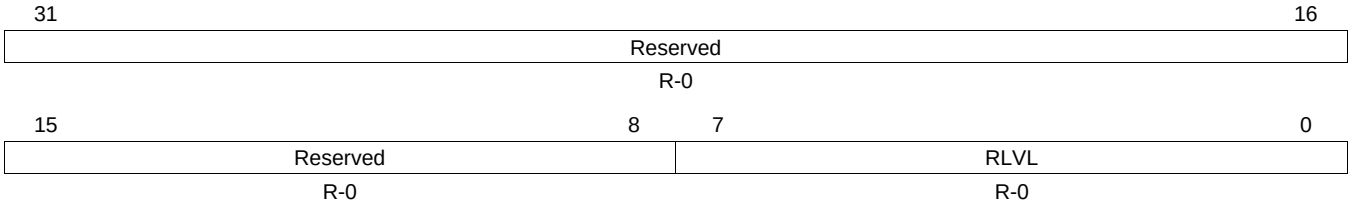
Table 14-48. Read FIFO Control Register (RFIFOCTL) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	RENA		Read FIFO enable bit.
		0	Read FIFO is disabled. The RLVL bit in the Read FIFO status register (RFIFOSTS) is reset to 0 and pointers are initialized, that is, the Read FIFO is “flushed.”
		1	Read FIFO is enabled. If Read FIFO is to be enabled, it must be enabled prior to taking McASP out of reset.
15-8	RNUMEVT	0-FFh	Read word count per DMA event (32-bit). When the Read FIFO contains at least RNUMEVT words of data, then an AREVT (receive DMA event) is generated to the host/DMA controller. This value should be set to a non-zero integer multiple of the number of serializers enabled as receivers. This value must be set prior to enabling the Read FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		3h-40h	3 to 64 words
		41h-FFh	Reserved
7-0	RNUMDMA	0-FFh	Read word count per transfer (32-bit words). Upon a receive DMA event from the McASP, the Read FIFO reads RNUMDMA words from the McASP. This value must equal the number of McASP serializers used as receivers. This value must be set prior to enabling the Read FIFO.
		0	0 words
		1	1 word
		2	2 words
		3h-10h	3-16 words
		11h-FFh	Reserved

14.3.46 Read FIFO Status Register (RFIFOSTS)

Note that this register is only available if the Read FIFO is configured. The Read FIFO status register (RFIFOSTS) is shown in Figure 14-82 and described in Table 14-49.

Figure 14-82. Read FIFO Status Register (RFIFOSTS)



LEGEND: R = Read only; -*n* = value after reset

Table 14-49. Read FIFO Status Register (RFIFOSTS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RLVL	0-FFh	Read level (read-only). Number of 32-bit words currently in the Read FIFO.
		0	0 words currently in Read FIFO.
		1h	1 word currently in Read FIFO.
		2h	2 words currently in Read FIFO.
		3h-40h	3 to 64 words currently in Read FIFO.
		41h-FFh	Reserved

Multichannel Buffered Serial Port (McBSP)

This chapter describes the multichannel buffered serial port (McBSP).

Topic **Page**

15.1 Introduction

15.1.1 Overview

The multichannel buffered serial port (McBSP) provides a full duplex direct serial interface between the host chip and other devices in a system such as application chips, codecs. It can accommodate to a wide range of peripherals and protocols thanks to its high level of versatility.

15.1.2 Features

The main features of the McBSP modules are:

- OCP-IP2.0 compliant interface (32-bit data bus)
- 512-byte internal buffer for transmit operation (fixed size)
- 512-byte internal buffer for receive operation (fixed size)
- Auto-gating clocks for power saving
- Power Management interface
- Full-duplex communication
- Buffered transmission and reception that allows a continuous data stream
- Independent clocking and framing for reception and for transmission
- Capability to generate interrupts or DMA requests on internal events
- 128 channels for transmission and for reception
- Multi-channel selection modes that allow to enable or block transfers in each one of the channels
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices (I2S compliant devices)
- Support for external generation of clock signals and frame-synchronization (frame-synchronization) signals
- A programmable sample rate generator for internal generation and control of clock signals and frame-synchronization signals
- Programmable polarity for frame-synchronization pulses and for clock signals
- A wide selection of data sizes: 8, 12, 16, 20, 24, and 32 bits
- Bit reordering (send/receive LSB)
- Status bits for flagging exception/error conditions
- Full/Half cycle mode on both transmit and receive side

The McBSP supports dual phase frames in order to provide I2S fully compliant capabilities. The limitation on dual phase frame is that the number of words per phase must be set to 1 for both first and second phase.

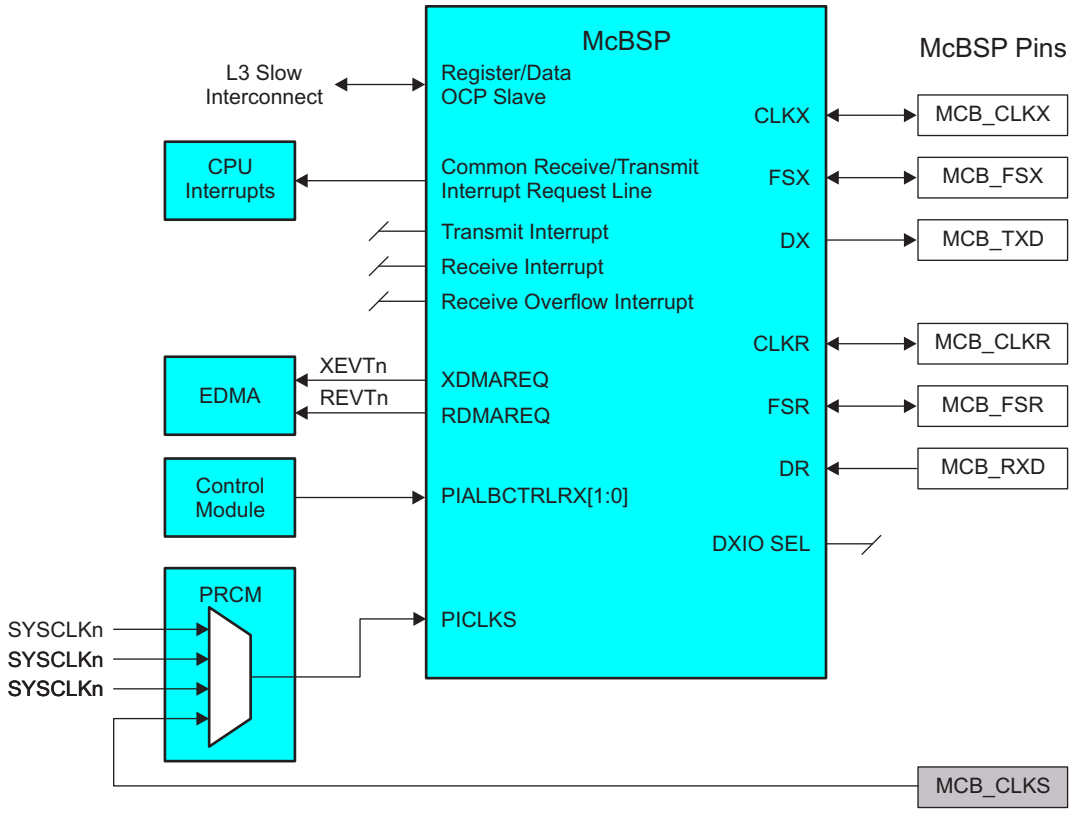
Four operating modes are defined for the module:

- **Active mode:** the module is running synchronously on the interface clock, interrupts and DMA requests can be generated according to the configuration and the external signals.
- **Smart Idle mode:** the module is in a waiting state, interface/functional clocks can be stopped, no interrupt can be generated, a wake-up signal can be generated according the configuration and external signals.
- **Force Idle mode:** the module has no activity, interface clock can be stopped, no interrupts and DMA requests can be generated, and the wake-up feature is inhibited. The software must disable the McBSP module (receive/transmit parts) prior to enter Force Idle mode.
- **No Idle mode:** the module is active but cannot respond to host interface requests.

The Smart/Force Idle modes are configured within the module and activated on request by the host processor through system interface sideband signals.

15.1.3 Block Diagram

Figure 15-1. McBSP Block Diagram



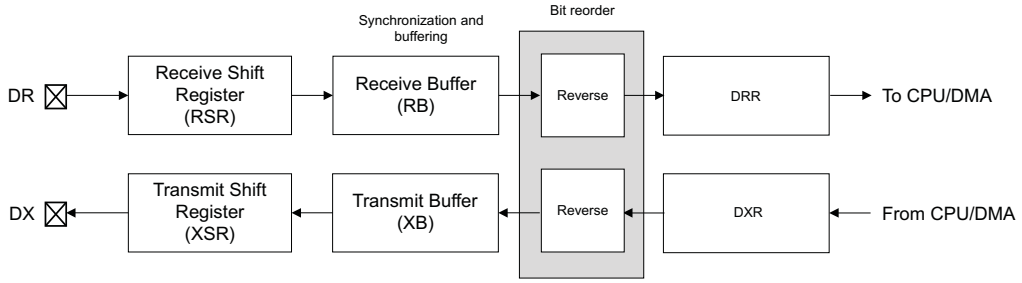
Note: The number of SYSCLKn varies per device implementation.
See Control Module Chapter for details.

15.2 Architecture

15.2.1 McBSP Data Transfer Process

Figure 15-2 shows the McBSP data transfer paths. McBSP receive operations and transmit operations are buffered (up to 512-byte buffers organized on 32-bit words are used). Input/output registers are 32-bit long words.

Figure 15-2. McBSP Data Transfer Paths



15.2.1.1 Data Transfer Process for 8/12/16/20/24/32-bit Long Words

Receive data arrives on the McBSP.DR pin and is shifted into receive shift register. When a full word is received, the content of the shift register is copied into receive buffer under condition receive buffer is not full. When the receive buffer threshold is reached, the McBSP asserts DMA or interrupt request and the receive buffer content is then transferred to the host (the CPU or the DMA controller reads the data receive register DRR_REG).

Transmit data is written by the CPU or the DMA controller to data transmit register DXR_REG using the byte enable input (when a byte is not enabled the byte value in the memory will contain the previous written value). If there is no previous data in transmit shift register, the value from the transmit buffer is copied to transmit shift register; otherwise, the content is copied to transmit shift register when the last bit of the previous data is shifted out on the McBSP.DX pin.

Note that the byte enable is used for transmit data register DXR_REG only and the value is ignored when the CPU accesses are targeting other registers. When writing to DXR_REG, the bytes are independently enabled by the byte enable pattern (for example, byte enable is “0001” then only the less significant byte will be written).

15.2.1.2 Bit Reordering (option to transfer LSB first)

Generally, the McBSP transmits or receives all data with the most significant bit (MSB) first. However, certain data protocols require the least significant bit (LSB) to be transferred first.

If you set XREVERSE = 01b in XCR2_REG[4:3], the bit ordering of the data words is reversed (LSB first) before being sent to the serial port.

If you set RREVERSE = 01b in RCR2_REG[4:3], the bit ordering of the data words is reversed during reception.

This feature is available for all the data formats from 8 up to 32-bit data length.

15.2.1.3 Clocking and Framing Data

This section explains basic concepts and terminology important for understanding how McBSP data transfers are timed and delimited.

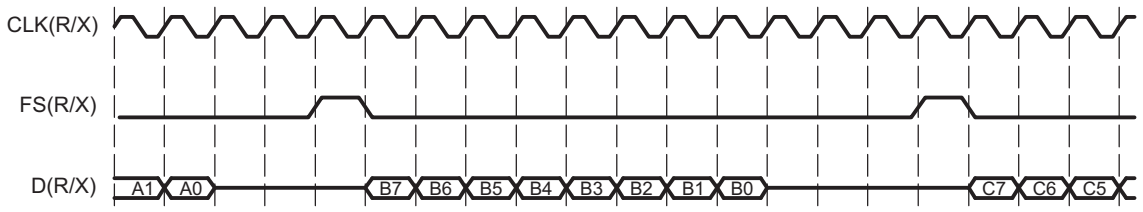
15.2.1.3.1 Clocking

Data is shifted one bit at a time from the McBSP.DR pin to the receive shift registers or from the transmit shift registers to the McBSP.DX pin. The time for each bit transfer is controlled by the rising or falling edge of a clock signal.

The receive clock signal (CLKR) controls bit transfers from the McBSP.DR pin to the receive shift registers. The transmit clock signal (CLKX) controls bit transfers from the transmit shift registers to the McBSP.DX pin. CLKR or CLKX can be derived from a pin at the boundary of the McBSP (McBSP.CLKR and McBSP.CLKX respectively) or derived from inside the McBSP. The polarities of CLKR and CLKX are programmable.

Figure 15-3 gives an example in which the clock signal controls the timing of each bit transfer on the pin.

Figure 15-3. Clock Signal Control of Bit Transfer Timing



The McBSP is constrained to operate at an internal functional frequency up to 55 MHz while CPU interface (OCP compliant interface) can operate at a frequency up to 110 MHz. When driving CLKX or CLKR at the pin, choose an appropriate input clock frequency. When using the internal sample rate generator for CLKX/CLKR/CLKS, choose an appropriate input clock frequency (up to 110 MHz) and divide down value (CLKGDV). Please see the device datasheet for maximum McBSP CLKX cycle time supported on your device.

15.2.1.3.2 Serial Words

Bits traveling between a shift register (RSR or XSR) and a data pin (McBSP.DR or McBSP.DX) are transferred in a group called a serial word. You can define how many bits are in a word.

Bits coming in on the McBSP.DR pin are held in RSR until RSR holds a full serial word. Only then is the word passed to the receive buffer RB (and ultimately to the DRR_REG register).

During transmission, XSR accept new data from transmit buffer XB after a full serial word has been passed from XSR to the McBSP.DX pin.

In the example in Figure 15-3, an 8–bit word size was defined (see bits 7 through 0 of word B being transferred).

15.2.1.3.3 Frames and Frame Synchronization

One or more words are transferred in a group called a frame. You can define how many words are in a frame. All of the words in a frame are sent in a continuous stream. However, there can be pauses between frame transfers. The McBSP uses frame–synchronization signals to determine when each frame is received/transmitted. When a pulse occurs on a frame–synchronization signal, the McBSP begins receiving/transmitting a frame of data. When the next pulse occurs, the McBSP receives/transmits the next frame, and so on. Pulses on the receive frame–synchronization (FSR) signal initiate frame transfers on McBSP.DR. Pulses on the transmit frame–synchronization (FSX) signal initiate frame transfers on McBSP.DX. FSR or FSX can be derived from a pin at the boundary of the McBSP (McBSP.FSR and McBSP.FSX respectively) or derived from inside the McBSP. In the example in Figure 15-3, a one–word frame is transferred when a frame–synchronization pulse occurs. In McBSP operation, the inactive–to–active transition of the frame–synchronization signal indicates the start of the next frame. For this reason, the frame–synchronization signal may be high for an arbitrary number of clock cycles. Only after the signal is recognized to have gone inactive, and then active again, does the next frame synchronization occurs.

15.2.1.3.4 Detecting Frame–Synchronization Pulses, Even in Reset State

The McBSP can generate receive and transmit interrupts to the CPU to indicate specific events in the McBSP. To facilitate detection of frame synchronization, these interrupts can be sent in response to frame–synchronization pulses. Unlike other serial port interrupt modes, this mode can operate while the associated portion of the serial port is in reset (such as activating RINT when the receiver is in reset). In this case, FSRM/FSXM and FSRP/FSXP still select the appropriate source and polarity of frame synchronization. Thus, even when the serial port is in the reset state, these signals are synchronized to the CPU clock and then sent to the CPU in the form of RINT and XINT at the point at which they feed the receiver and transmitter of the serial port. Consequently, a new frame–synchronization pulse can be detected, and after this occurs the CPU can take the serial port out of reset safely.

15.2.1.3.5 Ignoring Frame–Synchronization Pulses

The McBSP ignores transmit and/or receive frame–synchronization pulses while the frame transfer has been already started by a previous frame–synchronization pulse (unexpected frame–synchronization pulses). The McBSP does not support features like re-transmit or re-receive of an erroneous frame or word. The receiver or transmitter ignores frame–synchronization pulses until the desired frame length or number of words is reached. For more details on unexpected frame–synchronization pulses, see Section 15.2.3.2 and Section 15.2.3.5.

15.2.1.3.6 Frame Frequency

The frame frequency is determined by the period between frame–synchronization pulses and is defined as shown in the following equation:

Frame frequency = Clock frequency / (Number of clock cycles between frame – synchronization pulses)

The frame frequency can be increased by decreasing the time between frame–synchronization pulses (limited only by the number of bits per frame). As the frame transmit frequency increases, the inactivity period between the data packets for adjacent transfers decreases to zero.

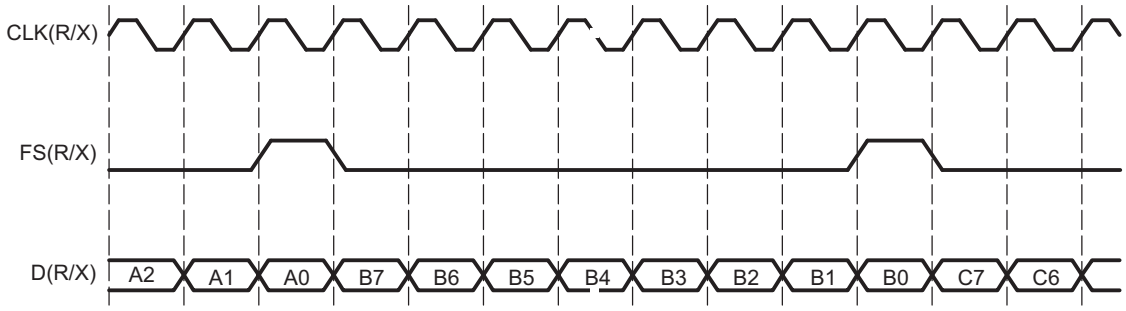
15.2.1.3.7 Maximum Frame Frequency

The minimum number of clock cycles between frame synchronization pulses is equal to the number of bits transferred per frame. The maximum frame frequency is defined as shown in the following equation:

Maximum frame frequency = Clock frequency / Number of bits per frame

Figure 15-4 shows the McBSP operating at maximum packet frequency. At maximum packet frequency, the data bits in consecutive packets are transmitted contiguously with no inactivity between bits.

Figure 15-4. McBSP Operating at Maximum Packet Frequency



If there is a 1–bit data delay as shown in this figure, the frame–synchronization pulse overlaps the last bit transmitted in the previous frame. Effectively, this permits a continuous stream of data, back-to-back transfers.

For XDATDLY = 0 (0-bit data delay), the first bit of data is transmitted asynchronously to the internal transmit clock signal (CLKX).

15.2.1.4 Frame Phases (dual phase frame I2S support)

The McBSP allows you to configure each frame to contain one or two phases. The support for dual phase frames is required in order to provide I2S fully compliant capabilities. The limitation on dual phase frame support is that the number of words per phase must be set to 1 for both first and second phase.

The number of bits per word can be specified differently for each of the two phases of a frame, allowing greater flexibility in structuring data transfers. For example, a user might define a frame as consisting of one phase containing one word of 16 bits, followed by a second phase consisting of one word of 32 bits. This configuration allows the user to compose frames for custom applications such as I2S protocol.

15.2.1.4.1 Number of Phases, Words, and Bits per Frame

Table 15-1 shows which bit fields in the receive control registers (RCR1_REG and RCR2_REG) and in the transmit control registers (XCR1_REG and XCR2_REG) determine the number of phases per frame, the number of words per frame, and the number of bits per word for each phase, for both receiver and transmitter. The maximum number of words per frame is limited to 2 when using dual-phase frames (one word for each phase), and to 128 for a single–phase frame. The number of bits per word can be 8, 12, 16, 20, 24, or 32 bits.

The following legend applies to the table:

- RPHASE = RCR2_REG[15]
- XPHASE = XCR2_REG[15]
- RFRLEN1 = RCR1_REG[14:8] (cleared to 0 if using dual-phase)
- RFRLEN2 = RCR2_REG[14:8] (cleared to 0 if using dual-phase)
- XFRLEN1 = XCR1_REG[14:8] (cleared to 0 if using dual-phase)
- XFRLEN2 = XCR2_REG[14:8] (cleared to 0 if using dual-phase)
- RWDLEN1 = RCR1_REG[7:5]
- RWDLEN2 = RCR2_REG[7:5]
- XWDLEN1 = XCR1_REG[7:5]
- XWDLEN2 = XCR2_REG[7:5]

Table 15-1. Phases, Words and Bits Per Frame Control Bits

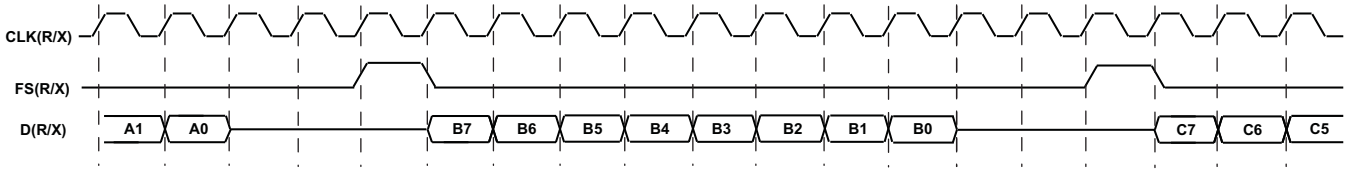
Operation	Number of Phases
Reception	1 (RPHASE = 0)
Reception	2 (RPHASE = 1)
Transmission	1 (XPHASE = 0)
Transmission	2 (XPHASE = 1)

15.2.1.4.2 Single-phase Frame Example

Figure 15-5 shows an example of a single-phase data frame containing one 8-bit word. Because the transfer is configured for one data bit delay, the data on the McBSP.DX and McBSP.DR pins are available one clock cycle after FS(R/X) goes active. The figure makes the following assumptions:

- RPHASE (RCR2_REG[15]) and XPHASE (XCR2_REG[15]) = 0: Single-phase frame
- RFRLEN1 (RCR1_REG[14:8]) and XFRLEN1 (XCR1_REG[14:8]) = 0: 1 word per frame
- RWDLEN1 (RCR1_REG[7:5]) and XWDLEN1 (XCR1_REG[7:5]) = 0: 8-bit word length
- RFRLEN2 (RCR2_REG[14:8]) and XWDLEN2 (XCR2_REG[14:8]) are ignored
- CLKRP (PCR_REG[0]) and CLKXP (PCR_REG[1]) = 0: Receive data clocked on falling edge; transmit data clocked on rising edge
- FSRP (PCR_REG[2]) and FSXP (PCR_REG[3]) = 0: Active–high frame–synchronization signals
- RDATDLY (RCR2_REG[1:0]) and XDARDLY (XCR2_REG[1:0]) = 01b: 1–bit data delay

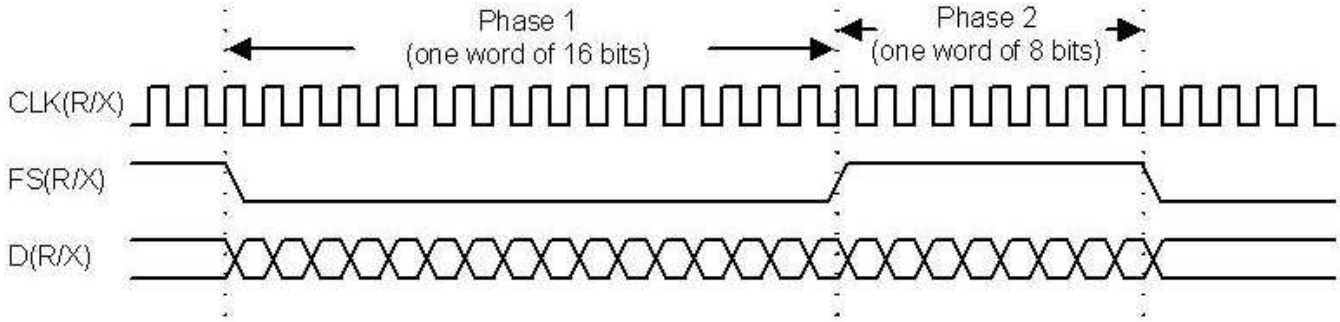
Figure 15-5. Single-Phase Frame for a McBSP Data Transfer



15.2.1.4.3 Dual-Phase Frame Example

Figure 15-6 shows an example of a frame where the first phase consists of one word of 16 bits, followed by a second phase of one word of 8 bits. The entire bit stream in the frame is contiguous. There are no gaps between words/phases.

Figure 15-6. Dual-Phase Frame for a McBSP Data Transfer



15.2.1.5 McBSP Reception

This section explains the fundamental process of reception in the McBSP.

Figure 15-7 and Figure 15-8 show how reception occurs in the McBSP. Figure 15-7 shows the physical path for the data. Figure 15-8 is a timing diagram showing signal activity for one possible reception scenario. A description of the process follows the figures.

Figure 15-7. McBSP Reception Physical Data Path

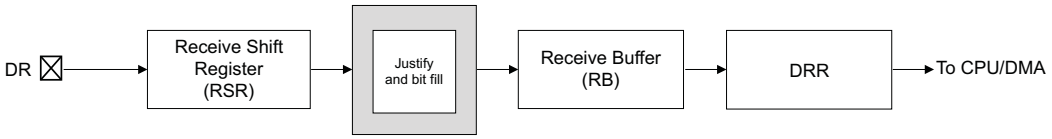
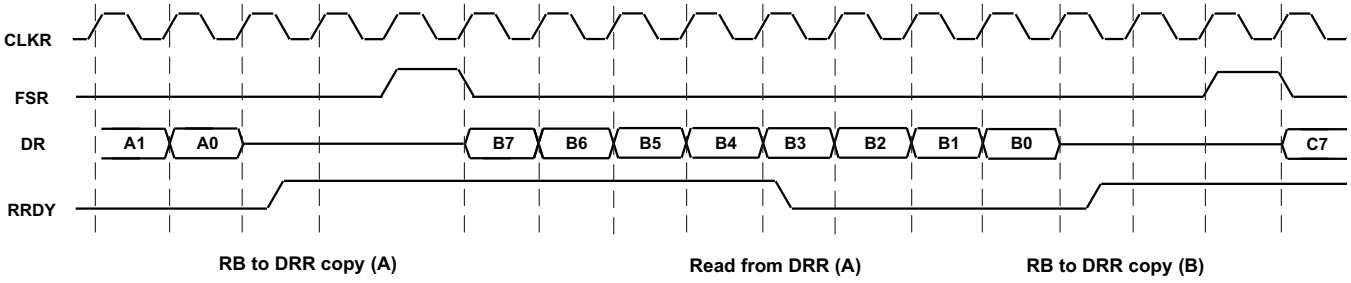


Figure 15-8. McBSP Reception Signal Activity



RRDY: Status of receiver ready bit (high is 1)

The following process describes how data travels from the McBSP.DR pin to the CPU or to the DMA controller:

1. The McBSP waits for a receive frame-synchronization pulse on internal FSR.
2. When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the RDATDLY bits of RCR2_REG[1:0] register. In the preceding timing diagram a 1-bit data delay is selected.
3. The McBSP accepts data bits on the McBSP.DR pin and shifts them into the receive shift register.
4. When a full word is received, the McBSP copies the contents of the receive shift register to the receive buffer, provided that RB is not full.
5. When the programmed receive threshold is reached the McBSP asserts the receiver ready bit (RRDY) in SPCR1_REG. This indicates that receive data is ready to be read by the CPU or the DMA controller by accessing DRR_REG register. The data copied from RB to DRR_REG is justified and bit filled according to the RJUST bits.
6. The CPU or the DMA controller reads the data from the data receive register. When the receive buffer is empty, RRDY bit is cleared.

When activity is not properly timed, errors can occur. See the following topics for more details:

- Overrun in the Receiver
- Underflow in the receiver
- Unexpected Receive Frame–Synchronization Pulse

15.2.1.6 McBSP Transmission

This section explains the fundamental process of transmission in the McBSP. Figure 15-9 and Figure 15-10 show how transmission occurs in the McBSP.

Figure 15-9 shows the physical path for the data. Figure 15-10 is a timing diagram showing signal activity for one possible transmission scenario. A description of the process follows the figures.

Figure 15-9. McBSP Transmission Physical Data Path

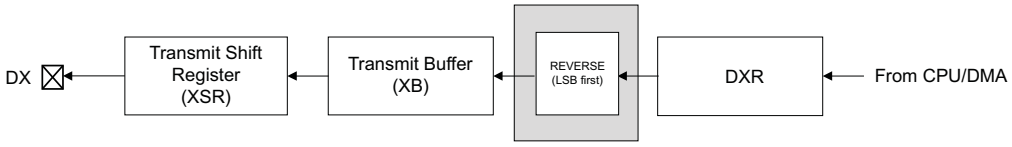
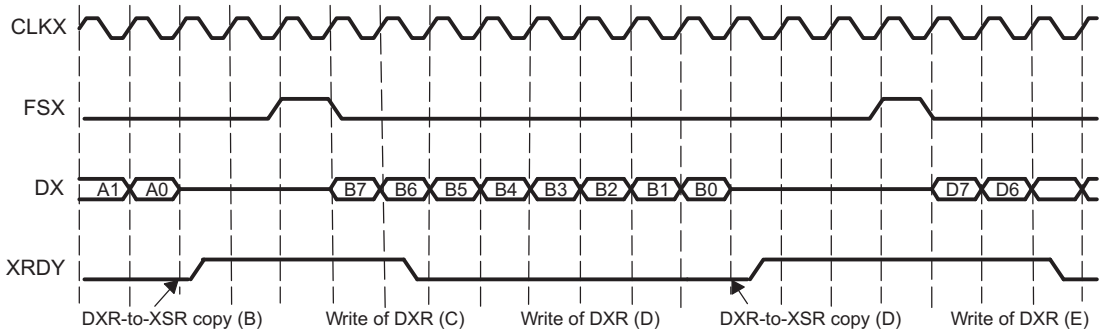


Figure 15-10. McBSP Transmission Signal Activity



The following process shows how.....

1. The CPU or the DMA controller writes data to the data transmit register. When the transmit buffer is reached the transmitter ready bit (XRDY) is cleared in SPCR2_REG[1] register to indicate that the transmitter is not ready for new data.
2. When new data arrives in DXR_REG register, the McBSP copies the content of the data transmit register to the transmit buffer. In addition, the Transmit Ready bit (XRDY) is set as long as the buffer contains at least the transmit threshold number of free locations. This indicates that the transmitter is ready to accept new data from the CPU or the DMA controller.
3. The McBSP waits for a transmit frame–synchronization pulse on internal FSX.
4. When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the XCR2_REG[1:0] register XDATDLY bits. In the preceding timing diagram, a 1-bit data delay is selected.
5. The McBSP shifts data bits from the transmit shift register to the McBSP.DX pin.

When activity is not properly timed, errors can occur. See the following topics for more details:

- Underflow in the transmitter
- Overflow in the transmitter
- Unexpected transmit frame-synchronization pulse

15.2.1.7 Enable/Disable the Transmit and Receive Processes

The McBSP has the option to stop-resume the transmit/receive process while the module is in functional mode (out of transmit/receive reset).

When the transmit/receive disable XDISABLE/RDISABLE bit in XCCR_REG/RCCR_REG register is set, the McBSP will stop the transmit/receive operation at the next frame boundary (frame corruption avoided).

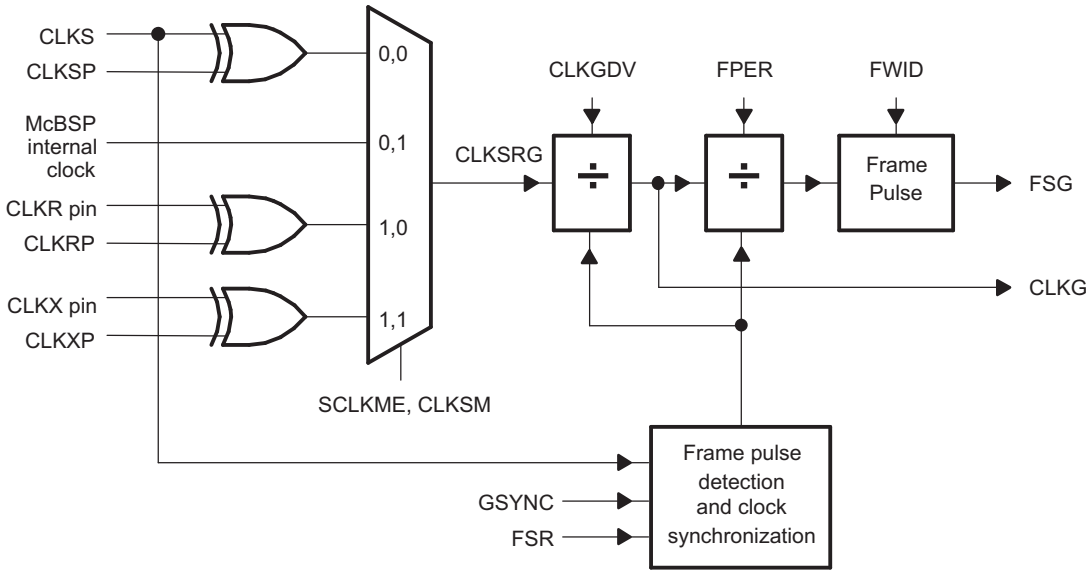
As soon as the XDISABLE/RDISABLE bit is cleared, the transmit/receive process will resume at the next frame boundary.

Note that it is not recommended to use this mechanism together with the possibility to interrogate the transmit/receive buffer status register (XBUFFSTAT/RBUFFSTAT indicating the occupied/available buffer locations), since this register is a CPU (OCP) clock synchronous register and does not reflect the exact number of occupied/free locations available on the functional clock domain.

15.2.2 McBSP Sample Rate Generator

The McBSP contains a sample rate generator that can be used to generate an internal data clock (CLKG) and an internal frame–synchronization signal (FSG). CLKG can be used for bit shifting on the data receive (McBSP.DR) pin and/or the data transmit (McBSP.DX) pin. FSG can be used to initiate frame transfers on McBSP.DR and/or McBSP.DX. Figure 15-11 is a conceptual block diagram of the sample rate generator.

Figure 15-11. Sample Rate Generator Block Diagram



The source clock for the sample rate generator (labeled CLKSRG in the diagram) can be supplied by either the FCLK functional clock (OCP interface clock), or by an external pin (McBSP.CLKS, McBSP.CLKX, or McBSP.CLKR). The source is selected with the SCLKME bit of the PCR_REG[7] register and the CLKSM bit of the SRGR2_REG[13] register.

If a pin is used, the polarity of the incoming signal can be inverted with the appropriate polarity bit (CLKSP bit of SRGR2_REG[14] register, CLKXP bit of PCR_REG register, or CLKRP bit of PCR_REG[0] register).

The sample rate generator has a three-stage clock divider that gives CLKG and FSG programmability.

The three stages provide:

- Clock divide-down. The source clock is divided according to the SRGR1_REG[7:0] register CLKGDV bits to produce CLKG .
- Frame period divide-down. CLKG is divided according to the SRGR2_REG[11:0] register FPER bit field to control the period from the start of a frame–pulse to the start of the next pulse.
- Frame-synchronization pulse-width countdown. CLKG cycles are counted according to the SRGR1_REG[15:8] register FWID bits to control the width of each frame-synchronization pulse.

Note: The McBSP cannot operate at an internal functional frequency faster than 55 MHz. Choose an input clock frequency and a CLKGDV value such that CLKG is less than or equal to 55 MHz. Please see the device datasheet for maximum McBSP CLKX cycle time supported on your device.

In addition to the three-stage clock divider, the sample rate generator has a frame-synchronization pulse detection and clock synchronization module that allows synchronization of the clock divide down with an incoming frame-synchronization pulse on the McBSP.FSR pin. This feature is enabled or disabled with the SRGR2_REG[15] register GSYNC bit.

CLKG is used as source in order to generate the output clocks CLKX/CLKR when the CLKXM/CLKRM indicates that the clock is an output. The output CLKX/CLKR is generated according to the clock polarity setting given by the CLKXP/CLKRP as follows: when falling edge, the CLKX/CLKR is CLKG inverted.

15.2.2.1 Clock Generation in the Sample Rate Generator

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both. Use of the sample rate generator to drive clocking is controlled by the clock mode bits (CLKRM and CLKXM) in the pin control register (PCR_REG[8] and PCR_REG[9] respectively) and polarity mode bits (CLKRP and CLKXP).

When a clock mode bit is set to 1 (CLKRM = 1 for reception, CLKXM = 1 for transmission), the corresponding data clock (CLKR for reception, CLKX for transmission) is driven by the internal sample rate generator output clock (CLKG) according with the polarity setting.

The effects of CLKRM = 1 and CLKXM = 1 on the McBSP are partially affected by the use of the digital loop back mode, the analog loop back mode and by the synchronous receive/transmit setting, respectively, as described in Table 15-2. The analog loop back mode is selected with the SPCR1_REG[15] register ALB bit. The digital loop back mode is selected with the DLB bit of the XCCR_REG register. The synchronous mode is controlled by the ALBRXCTRL input pin.

When using the sample rate generator as a clock source, make sure the sample rate generator is enabled (GRST = 1).

Table 15-2. Effects of DLB and ALB Bits on Clock Modes

Mode Bit Settings		Effect
CLKRM = 1	DLB = 0 & ALB = 0 (Digital & analog loop back mode disabled)	CLKR is an output pin driven by the sample rate generator output clock (CLKG).
	DLB = 0 & ALB = 1 (Analog loop back mode enabled)	CLKR is an output pin driven by the sample rate generator output clock (CLKG). The receive functional part internal clock is driven by CLKX input pin CLKXI. The source of CLKX depends on the CLKXM bit. The receive frame synchronization is driven by FSX input pin (FSXI). The receive data is driven by the DX loop back pin (DXI).
	DLB = 1 & ALB = 0 (Digital loop back mode enabled)	CLKR is not driven. The sample rate generator and the frame synchronization generator need to be enabled. The internal transmit and receive clocks are driven by SRG (CLKG having the appropriate CLKXP polarity). The transmit and receive frame synchronization signals are driven by FSG (having the appropriate FSXP polarity). The transmit data is connected to the receive input data. Note that in digital loop back mode no serial link activity will be seen by the remote device because CLKREN, CLKXEN, FSREN, FSRXEN, DXEN are not active.
	DLB = 1 & ALB = 1 (reserved mode)	Undefined functionality.

Table 15-2. Effects of DLB and ALB Bits on Clock Modes (continued)

Mode Bit Settings		Effect
CLKXM = 1	DLB = 0 & ALB = 0 (Digital & analog loop back mode disabled)	CLKX is an output pin driven by the sample rate generator output clock (CLKG).
	DLB = 0 & ALB = 1 (Analog loop back mode enabled)	CLKX is an output pin driven by the sample rate generator output clock (CLKG).
	DLB = 1 & ALB = 0 (Digital loop back mode enabled)	CLKX is not driven. The sample rate generator and the frame synchronization generator need to be enabled. The internal transmit and receive clocks are driven by SRG (CLKG having the appropriate CLKXP polarity). The transmit and receive frame synchronization signals are driven by FSG (having the appropriate FSXP polarity). The transmit data is connected to the receive input data. Note that in digital loop back mode no serial link activity will be seen by the remote device because CLKREN, CLKXEN, FSREN, FSRXEN, DXEN are not active.
	DLB = 1 & ALB = 1 (reserved mode)	Undefined functionality.
	ALBRXCTRL[0] = 1 (synchronous mode DLB = 0 & ALB = 0)	CLKX is an output pin driven by the sample rate generator output clock (CLKG). CLKR is connected to the CLKX.

15.2.2.2 Frame Synchronization Generation in the Sample Rate Generator

The sample rate generator can produce a frame–synchronization signal (FSG) for use by the receiver, the transmitter, or both.

If you want the receiver to use FSG for frame synchronization, make sure PCR_REG[10] register FSRM bit = 1. (When FSRM = 0, receive frame synchronization is supplied via the McBSP.FSR pin.)

If you want the transmitter to use FSG for frame synchronization, you must set:

- FSXM = 1 in PCR_REG[11] register: This indicates that transmit frame synchronization is supplied by the McBSP itself rather than from the McBSP.FSX pin
- FSGM = 1 in SRGR2_REG[12] register: This indicates that when FSXM = 1, transmit frame synchronization is supplied by the sample rate generator. (When FSGM = 0 and FSXM = 1, the transmit frame-synchronization signal (FSX) is generated when transmit buffer is not empty. When FSGM = 0, FPER and FWID are used to determine the frame synchronization period and width (external FSX is gated by the buffer empty condition).

In either case, the sample rate generator must be enabled (SPCR2_REG[6] register GRST bit = 1) and the frame–synchronization logic in the sample rate generator must be enabled (SPCR2_REG[7] register FRST bit = 0).

15.2.2.2.1 Choosing the Width of the Frame-Synchronization Pulse

Each pulse on FSG has a programmable width. You program the SRGR1_REG[15:8] register FWID bits, and the resulting pulse width is (FWID + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

15.2.2.2.2 Controlling the Period Between the Starting Edges of the Frame-Synchronization Pulses

You can control the amount of time from the starting edge of one FSG pulse to the starting edge of the next FSG pulse. This period is controlled in one of two ways, depending on the configuration of the sample rate generator:

- If the sample rate generator is using an external input clock and SRGR2_REG[15] register GSYNC bit = 1, FSG pulses in response to an inactive–to–active transition on the McBSP.FSR pin. Thus, an external device controls the frame–synchronization period.
- Otherwise, you program the FPER bits of SRGR2_REG[11:0] register, and the resulting frame–synchronization period is (FPER + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

15.2.2.2.3 Keeping FSG Synchronized to an External Clock

When an external signal is selected to drive the sample rate generator, the GSYNC bit of SRGR2_REG15] register and the McBSP.FSR pin can be used to configure the timing of FSG pulses.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive–to–active transition on the McBSP.FSR pin triggers a resynchronization of CLKG and generation of FSG.

15.2.2.3 Synchronizing Sample Rate Generator Outputs to an External Clock

The sample rate generator can produce a clock signal (CLKG) and a frame–synchronization signal (FSG) based on an input clock signal that is either the FCLK functional clock signal or a signal at the McBSP.CLKS, McBSP.CLKR, or McBSP.CLKX pin. When an external clock is selected to drive the sample rate generator, the GSYNC bit of SRGR2_REG[15] register and the McBSP.FSR pin can be used to control the timing of CLKG and the pulsing of FSG relative to the chosen input clock.

Make GSYNC = 1 when you want the McBSP and an external device to divide down the input clock with the same phase relationship. If GSYNC = 1:

- An inactive–to–active transition on the McBSP.FSR pin triggers a resynchronization of CLKG and a pulsing of FSG.
- CLKG always begins with a high state after synchronization.
- FSR is always detected at the same edge of the input clock signal that generates CLKG, no matter how long the FSR pulse is.
- The SRGR2_REG[11:0] register FPER bits are ignored because the frame–synchronization period on FSG is determined by the arrival of the next frame–synchronization pulse on the McBSP.FSR pin.

If GSYNC = 0, CLKG runs freely and is not resynchronized, and the frame–synchronization period on FSG is determined by FPER.

15.2.2.3.1 Operating the Transmitter Synchronously with the Receiver

When GSYNC = 1, the transmitter can operate synchronously with the receiver, provided that:

- FSX is programmed to be driven by FSG (FSGM = 1 in SRGR2_REG[12] register and FSXM = 1 in PCR_REG[11] register). If the input FSR has appropriate timing so that it can be sampled by the falling edge of CLKG, it can be used, instead, by setting FSXM = 0 and connecting FSR to FSX externally.
- The sample rate generator clock drives the transmit and receive clocking (PCR_REG[8] CLKRM bit and PCR_REG[9] CLKXM bit are set to 1). Therefore, the CLK(R/X) pin must not be driven by any other driving source.

15.2.2.3.2 Synchronization Examples

Figure 15-12 and Figure 15-13 show the clock and frame–synchronization operation with various polarities of CLKS (the chosen input clock) and FSR. These figures assume FWID = 0 in SRGR1_REG[15:8] register, for an FSG pulse that is one CLKG cycle wide. The FPER bits of SRGR2_REG[11:0] register are not programmed; the period from the start of a frame–synchronization pulse to the start of the next pulse is determined by the arrival of the next inactive–to–active transition on the McBSP.FSR pin. Each of the figures shows what happens to CLKG when it is initially synchronized and GSYNC = 1, and when it is not initially synchronized and GSYNC = 1. Figure 15-13 has a slower CLKG frequency (it has a larger divide–down value in the CLKGDV bits of SRGR1_REG[7:0] register).

Figure 15-12. CLKG Synchronization and FSG Generation (GSYNC = 1 and CLKGDV = 1)

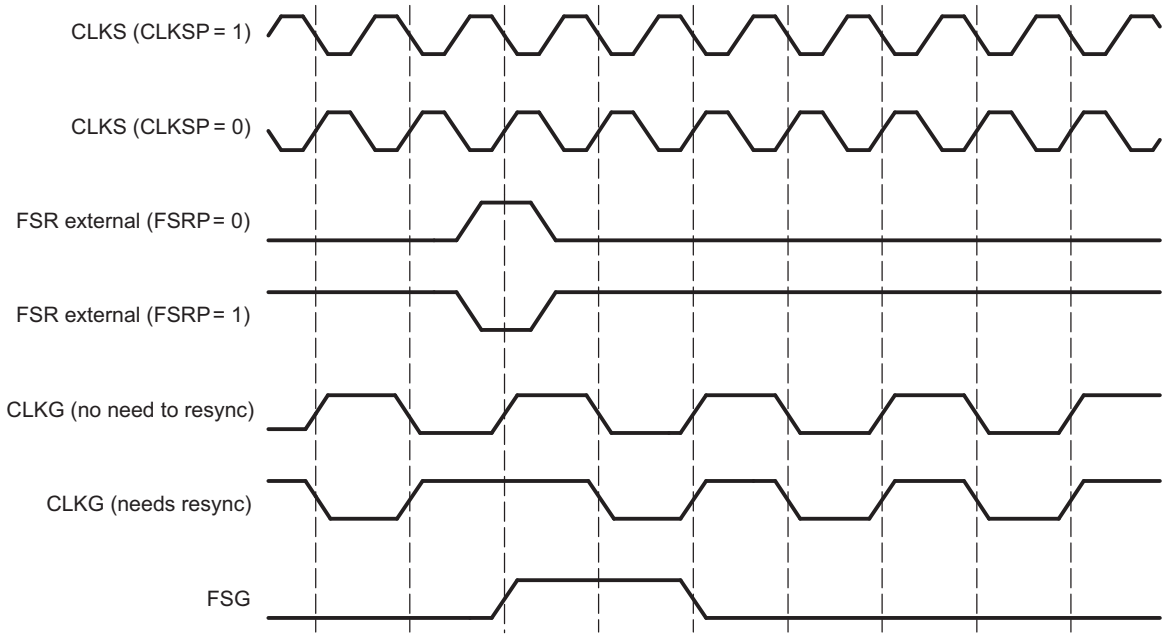
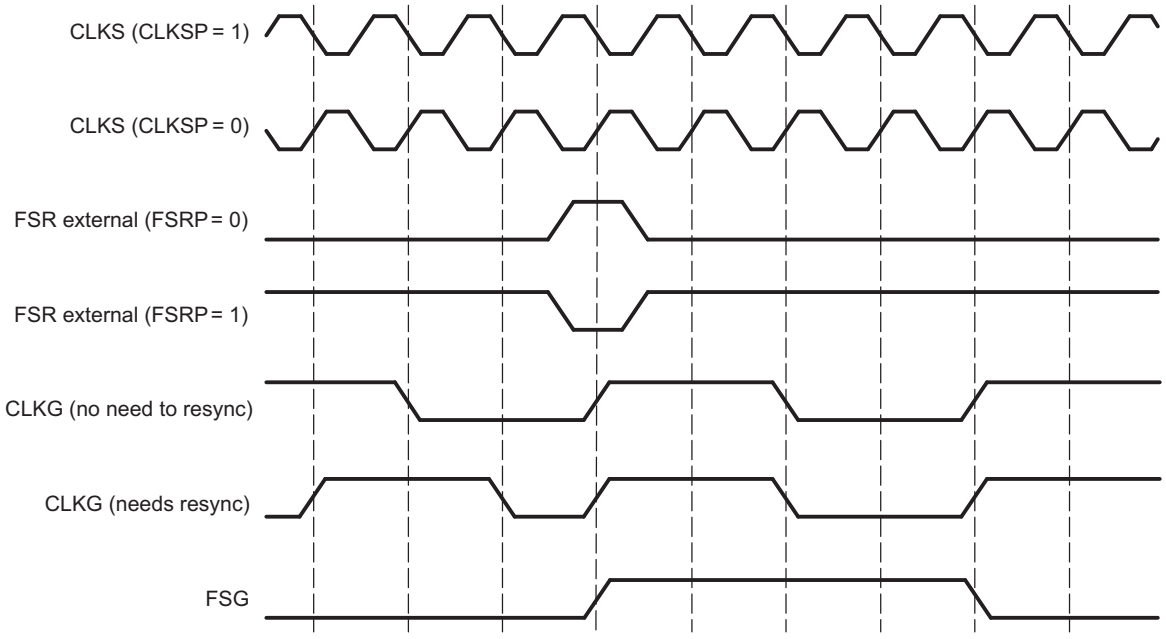


Figure 15-13. CLKG Synchronization and FSG Generation (GSYNC = 1 and CLKGDV = 3h)



15.2.3 McBSP Exception/Error Conditions

There are several serial port events that can constitute a system error. Any of this error conditions may be sources of an interrupt:

- Receiver Overrun

(ROVFLSTAT bit is set to 1 in IRQSTATUS register, and legacy mode RFULL bit is set to 1 in SPCR1_REG register)

This occurs when receive buffer is full and RSR(s) are full with another new word shifted in from McBSP.DR. Therefore, ROVFLSTAT (RFULL) indicates an error condition wherein any new data that can arrive at this time on McBSP.DR replaces the contents of the RSR, and the previous word is lost. The RSR continue to be overwritten as long as new data arrives on McBSP.DR and DRR_REG register is not read.

- Unexpected Receive Frame-Synchronization Pulse

(RSYNCERR bit is set to 1 in IRQSTATUS register, and legacy mode RSYNCERR bit is set to 1 in SPCR1_REG register)

This occurs during reception when an unexpected frame–synchronization pulse arrives. An unexpected frame-synchronization pulse is one that is supposed to begin the next frame transfer before all the bits of the current frame have been received. Such a pulse is ignored by the receiver, but sets the RSYNCERR bit in SPCR1_REG.

- Receiver Underflow

(RUNDFLSTAT bit is set to 1 in IRQSTATUS register)

This occurs when DMA controller or CPU reads data from an empty receive buffer.

- Transmitter Underflow

(XUNDFLSTAT bit is set to 1 in IRQSTATUS register, and legacy mode XEMPTY bit is cleared to 0 in SPCR2_REG register)

If a new frame–synchronization signal arrives when transmit buffer is empty, the previous data in the XSR is sent again. This procedure continues for every new frame–synchronization pulse that arrives until DXR_REG register is loaded with new data (and the XB buffer will be no longer empty).

- Unexpected Transmit Frame-Synchronization Pulse

(XSYNCERR bit is set to 1 in IRQSTATUS register, and legacy mode XSYNCERR bit is set to 1 in SPCR2_REG register)

This occurs during transmission when an unexpected frame–synchronization pulse arrives. An unexpected frame–synchronization pulse is one that is supposed to begin the next frame transfer before all the bits of the current frame have been transferred. Such a pulse is ignored by the transmitter, but sets the XSYNCERR bit in SPCR2_REG.

- Transmitter Overflow

(XOVFLSTAT bit is set to 1 in IRQSTATUS register)

This occurs when DMA controller or CPU write data to a full transmit buffer.

15.2.3.1 Overrun in the Receiver

(ROVFLSTAT bit set to 1 in IRQSTATUS register, and legacy mode RFULL bit set to 1 in SPCR1_REG register)

Indicates that the receiver has experienced overrun and is in an error condition. Receive overrun is set when all of the following conditions are met:

- DRR_REG is not read even if the RRDY bit in IRQSTATUS register is set (legacy mode: RRDY bit in SPCR1_REG register) and DMA or interrupt request has been asserted.
- RB is full
- RSR is full

As previously described, data arriving on McBSP.DR is continuously shifted into. Once a complete word is shifted into the RSR, an RSR–to–RB copy can occur only if the RB is not full.

Either of the following events clears the legacy mode RFULL bit and allows subsequent transfers to be read properly:

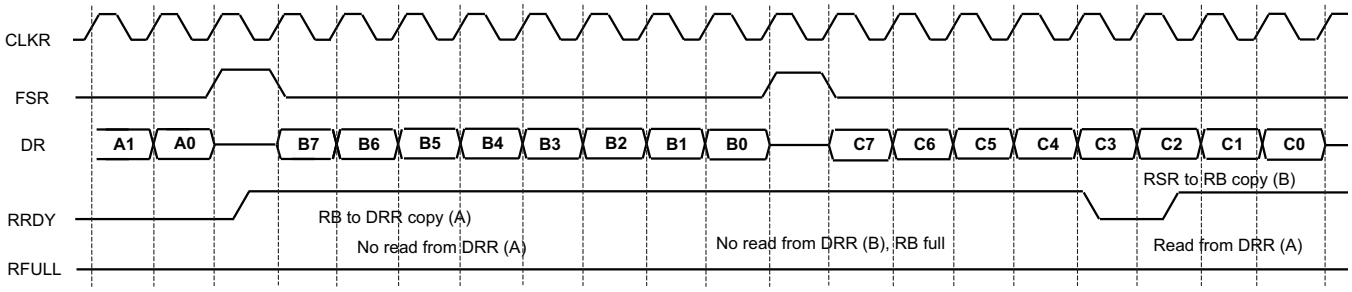
- The CPU or DMA controller reads DRR_REG.
- The receiver is reset individually (SPCR1_REG[0] register RRST bit = 0) or as part of an Global reset.

Another frame–synchronization pulse is required to restart the receiver.

According to the IRQENABLE register setting, this condition can generate the COMMONIRQ line to be asserted low. Writing 1 to the corresponding bit in status register will clear the interrupt.

Figure 15-14 shows the receive overrun condition.

Figure 15-14. Overrun in the McBSP Receiver



15.2.3.2 Unexpected Receive Frame-Synchronization Pulse

15.2.3.2.1 Possible Responses to Receive Frame-Synchronization Pulses

If a frame–synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame–synchronization pulse, and the receiver sets the receive frame–synchronization error bit RSYNCERR in IRQSTATUS (and the legacy RSYNCERR bit SPCR1_REG[3]) register.

According to the IRQENABLE register settings this condition can generate the COMMONIRQ line to be asserted low. Writing 1 to the corresponding bit in status register will clear the interrupt.

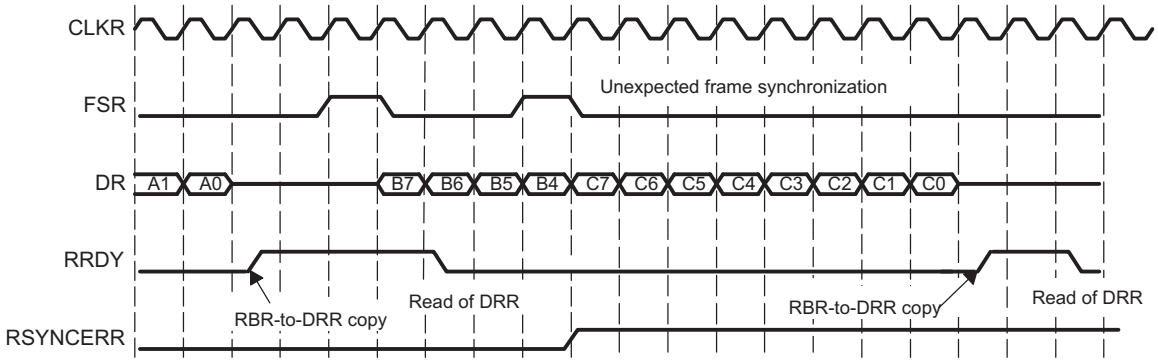
Using the legacy mode, RSYNCERR bit in SPCR1_REG can be cleared only by a receiver reset or by a write of 0 to this bit. If you want the McBSP to notify the CPU of receive frame–synchronization errors, you can set the legacy mode receive interrupt with the SPCR1_REG[5:4] register RINTM bits. When RINTM = 11b, the McBSP sends a receive interrupt (legacy mode RINT) request to the CPU each time that RSYNCERR is set.

15.2.3.2.2 Example of an Unexpected Receive Frame-Synchronization Pulse

Figure 15-15 shows an unexpected receive frame–synchronization pulse during normal operation of the serial port, with time intervals between data packets.

Note that the unexpected receive frame–synchronization pulse does not influence the data receive process, being ignored by the data receive state machine.

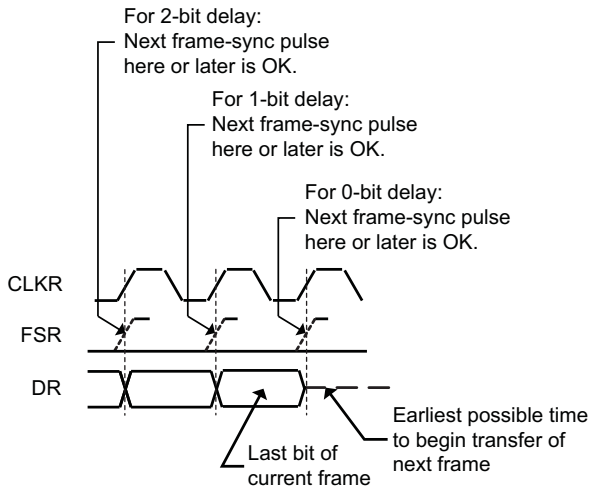
Figure 15-15. Unexpected Frame-Synchronization Pulse During a McBSP Reception



15.2.3.2.3 Preventing Unexpected Receive Frame-Synchronization Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKR cycles, depending on the value of the RCR2_REG[1:0] RDATDLY bits. For each possible data delay, Figure 15-16 shows when a new frame–synchronization pulse on FSR can safely occur relative to the last bit of the current frame.

Figure 15-16. Proper Positioning of Receive Frame-Synchronization Pulses



15.2.3.3 Underflow in the Receiver

The McBSP indicates a receiver underflow condition by setting the RUNDFLSTAT bit in IRQSTATUS register. This error occurs when DMA controller or CPU reads data from an empty receive buffer (this may happen only if the CPU or DMA controller does not respect the DMA length, does not wait for DMA request or does not check the buffer status before reading data. According to the IRQENABLE register settings this condition can generate the COMMONIRQ line to be asserted low. Writing 1 to the corresponding bit in status register will clear the interrupt.

15.2.3.4 Underflow in the Transmitter

The McBSP indicates a transmitter empty (or underflow) condition by setting the XUNDFLSTAT bit in IRQSTATUS register. Also the legacy mode XEMPTY bit in SPCR2_REG[2] register is cleared. Either of the following events activates XEMPTY (XEMPTY = 0):

- DXR_REG has not been loaded and transmit buffer is empty, and all bits of the data word in the XSR have been shifted out on the McBSP.DX pin.
- The transmitter is reset (by forcing XRST = 0 in SPCR2_REG[0] register, or by an Global reset) and is then restarted.

XEMPTY is deactivated (XEMPTY = 1) when a new word in DXR_REG is transferred to XB. If FSXM = 1 in PCR_REG[11] and FSGM = 0 in SRGR2_REG[12], the transmit frame-synchronization signal (FSX) is generated when transmit buffer is not empty. When FSGM = 0, FPER and FWID are used to determine the frame synchronization period and width (external FSX is gated by the buffer empty condition). Otherwise, the transmitter waits for the next frame–synchronization pulse before sending out the next frame on McBSP.DX.

When the transmitter is taken out of reset (XRST = 1), it is in a transmitter ready state (SPCR2_REG[1] register XRDY bit = 1) and transmitter empty (XEMPTY = 0) state. If DXR_REG is loaded by the CPU or the DMA controller before internal FSX goes active high, a valid XB–to–XSR transfer occurs. This allows for the first word of the first frame to be valid even before the transmit frame–synchronization pulse is generated or detected. Alternatively, if a transmit frame–synchronization pulse is detected before DXR_REG is loaded, zeros are output on McBSP.DX.

The XUNDFLSTAT in IRQSTATUS register indicates a real underflow condition, in which the frame is corrupted due to lack of data availability during transmit process. According to the IRQENABLE register settings this condition can generate the COMMONIRQ line to be asserted low. Writing 1 to the corresponding bit in status register will clear the interrupt.

15.2.3.5 Unexpected Transmit Frame-Synchronization Pulse

15.2.3.5.1 Possible Responses to Transmit Frame-Synchronization Pulses

If a frame–synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame–synchronization pulse, and the transmitter sets the transmit frame–synchronization error bit XSYNCERR in IRQSTATUS (and the legacy XSYNCERR bit in SPCR2_REG[3]) register.

According to the IRQENABLE register settings this condition can generate the COMMONIRQ line to be asserted low. Writing 1 to the corresponding bit in status register will clear the interrupt.

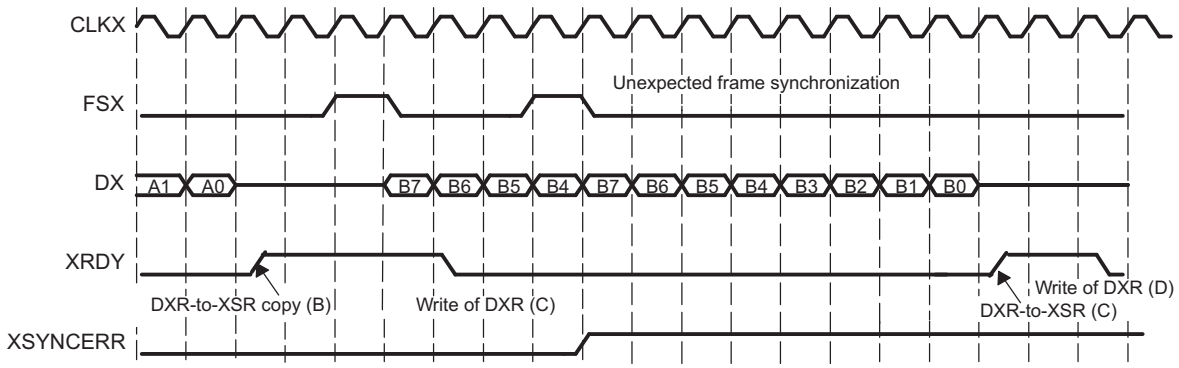
Using the legacy mode, XSYNCERR bit in SPCR2_REG can be cleared only by a transmitter reset or by a write of 0 to this bit. If you want the McBSP to notify the CPU of frame–synchronization errors, you can set a special transmit interrupt mode with the SPCR2_REG[5:4] register XINTM bits. When XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU each time that XSYNCERR is set.

15.2.3.5.2 Example of Unexpected Transmit Frame-Synchronization Pulse

Figure 15-17 shows an unexpected transmit frame–synchronization pulse during normal operation of the serial port with intervals between the data packets.

Note that the unexpected transmit frame–synchronization pulse does not influence the data transmit process, being ignored by the data transmit state machine.

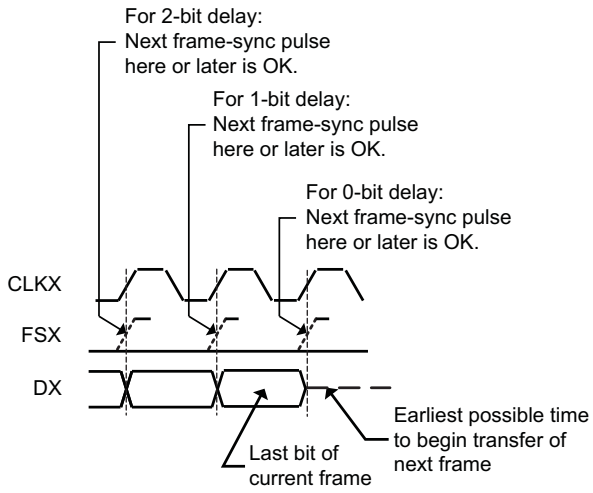
Figure 15-17. Unexpected Frame-Synchronization Pulse During a McBSP Transmission



15.2.3.5.3 Preventing Unexpected Transmit Frame-Synchronization Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKX cycles, depending on the value in the XCR2_REG[1:0] register XDATDLY bits. For each possible data delay, Figure 15-18 shows when a new frame–synchronization pulse on FSX can safely occur relative to the last bit of the current frame.

Figure 15-18. Proper Positioning of Transmit Frame-Synchronization Pulses



15.2.3.6 Overflow in the Transmitter

The McBSP indicates a transmitter overflow condition by setting the XOVFLSTAT bit in IRQSTATUS[11] register. This error occurs when DMA controller or CPU write data to a full transmit buffer (this may happen only if the CPU or DMA controller does not respect the DMA length, does not wait for DMA request or does not check the buffer status before writing data. According to the IRQENABLE register settings this condition can generate the COMMONIRQ line to be asserted low. Writing 1 to the corresponding bit in status register will clear the interrupt.

15.2.4 McBSP DMA Configuration

The McBSP receive and transmit data DMA requests are active after the receive RRST and transmit XRST are released. After reset the default DMA threshold (and length) is one.

The receive and transmit DMA requests can be individually disabled by setting zero the RDMAEN, XDMAEN bits in RCCR_REG respectively XCCR_REG. When disabling the DMA the DMA request line is deasserted even if a DMA transfer is pending and the DMA state machine is not reset.

The DMA threshold and length configuration is done through THRSH1_REG and THRSH2_REG registers as follows:

- (THRSH1_REG + 1) value represents the required receive DMA request length (the length of the transfer is the same as the threshold value plus one). As long as the RB occupied locations level is above or equal to the THRSH1_REG value plus one, the DMA request will be asserted. After transferring the configured (THRSH1_REG + 1) number of words, the receive DMA request (McBSP.REVNT) will be deasserted and reasserted as soon as the conditions are met again.
- (THRSH2_REG + 1) value represents the required transmit DMA request length (the length of the transfer is the same as the threshold value plus one). As long as the XB free locations level is above or equal to the THRSH2_REG value plus one the DMA request will be asserted. After transferring the configured (THRSH2_REG + 1) number of words, the transmit DMA request (McBSP.XEVNT) will be deasserted and reasserted as soon as the conditions are met again.

Note that the CPU may decide not to use the DMA to transfer the data. In this case the DMA must be disabled (or the DMA request can be ignored by CPU) and the common interrupt line can be used. The RRDY bit for receive and XRDY bit for transmit will indicate when the threshold values are reached. Also, by reading the receive buffer status RBUFFSTAT_REG register and transmit buffer status XBUFFSTAT_REG register the CPU may decide to transfer data even if the threshold is not reached. This mechanism is useful on the last transfer on receive side when the threshold value is bigger than the occupied locations inside the receive buffer and the CPU needs to read this data. Since no interrupt or DMA request is asserted the only option in this case is to read the receive buffer status register value and to transfer the remaining data without using the DMA or interrupt indication.

15.2.5 Multichannel Selection Modes

15.2.5.1 Channels, Blocks, and Partitions

A McBSP channel is a time slot for shifting in/out the bits of one serial word. The McBSP supports up to 128 channels for reception and 128 channels for transmission. In the receiver and in the transmitter, the 128 available channels are divided into 8 blocks that each contains 16 contiguous channels:

Table 15-3. Channels, Block, Partitions

Block 0: Channels 0–15	Block 4: Channels 64–79
Block 1: Channels 16–31	Block 5: Channels 80–95
Block 2: Channels 32–47	Block 6: Channels 96–111
Block 3: Channels 48–63	Block 7: Channels 112–127

The blocks are assigned to partitions according to the selected partition mode. In the 2–partition mode, you assign one even–numbered block (0, 2, 4, or 6) to partition A and one odd–numbered block (1, 3, 5, or 7) to partition B. In the 8–partition mode, blocks 0 through 7 are automatically assigned to partitions, A through H, respectively.

The number of partitions for reception and the number of partitions for transmission are independent. For example, it is possible to use two receive partitions (A and B) and eight transmit partitions (A–H).

15.2.5.2 Multichannel Selection

When a McBSP uses a time–division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices, the McBSP may need to receive and/or transmit on only a few channels. To save memory and bus bandwidth, you can use a multichannel selection mode to prevent data flow in some of the channels.

Each channel partition has a dedicated channel enable register. If the appropriate multichannel selection mode is on, each bit in the register controls whether data flow is allowed or prevented in one of the channels that is assigned to that partition.

The McBSP has one receive multichannel selection mode and three transmit multichannel selection modes.

15.2.5.3 Configuring a Frame for Multichannel Selection

Before you enable a multichannel selection mode, make sure you properly configure the data frame:

- Select a single–phase frame (RCR2_REG[15] register RPHASE bit and XCR2_REG[15] register XPHASE bit = 0). Each frame represents a TDM data stream.
- Set a frame length (in RCR1_REG[14:8] register RFRLEN1 bit field and in XCR1_REG[14:8] register XFRLEN1 bit field) that includes the highest–numbered channel to be used. For example, if you plan to use channels 0, 15, and 39 for reception, the receive frame length must be at least 40 (RFRLEN1 = 39). If XFRLEN1 = 39 in this case, the receiver creates 40 time slots per frame but only receives data during time slots 0, 15, and 39 of each frame.

15.2.5.4 Using Eight Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use eight partitions or two partitions. If you choose the 8–partition mode (RMCME = 1 for reception, XMCME = 1 for transmission), McBSP channels are activated in the following order: A, B, C, D, E, F, G, H.

In response to a frame–synchronization pulse, the receiver or transmitter begins with the channels in partition A and then continues with the other partitions in order until the complete frame has been transferred. When the next frame–synchronization pulse occurs, the next frame is transferred, beginning with the channels in partition A.

In the 8–partition mode, the (R/X)PABLK and (R/X)PBBLK bits are ignored and the 16–channel blocks are assigned to the partitions as shown in the following tables. These assignments cannot be changed. The tables also show the registers used to control the channels in the partitions.

Figure 15-19 shows an example of the McBSP using the 8–partition mode. In response to a frame–synchronization pulse, the McBSP begins a frame transfer with partition A and then activates B, C, D, E, F, G, and H to complete a 128–word frame.

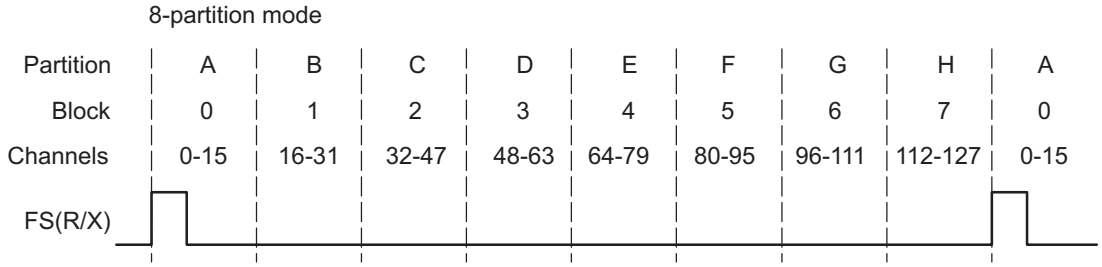
Table 15-4. Eight Partitions – Receive Channel Assignment and Control

Receive Partition	Assigned Block of Receive Channels	Register Used for Channel Control
A	Block 0: channels 0 through 15	RCERA_REG
B	Block 1: channels 16 through 31	RCERB_REG
C	Block 2: channels 32 through 47	RCERC_REG
D	Block 3: channels 48 through 63	RCERD_REG
E	Block 4: channels 64 through 79	RCERE_REG
F	Block 5: channels 80 through 95	RCERF_REG
G	Block 6: channels 96 through 111	RCERG_REG
H	Block 7: channels 112 through 127	RCERH_REG

Table 15-5. Eight Partitions – Transmit Channel Assignment and Control

Transmit Partition	Assigned Block of Receive Channels	Register Used for Channel Control
A	Block 0: channels 0 through 15	XCERA_REG
B	Block 1: channels 16 through 31	XCERB_REG
C	Block 2: channels 32 through 47	XCERC_REG
D	Block 3: channels 48 through 63	XCERD_REG
E	Block 4: channels 64 through 79	XCERE_REG
F	Block 5: channels 80 through 95	XCERF_REG
G	Block 6: channels 96 through 111	XCERG_REG
H	Block 7: channels 112 through 127	XCERH_REG

Figure 15-19. McBSP Data Transfer in the 8-Partition Mode



15.2.5.5 Receive Multichannel Selection Mode

The MCR1_REG[0] register RMCM bit determines whether all channels or only selected channels are enabled for reception. When RMCM = 0, all 128 receive channels are enabled and cannot be disabled. When RMCM = 1, the receive multichannel selection mode is enabled. In this mode:

- Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCER[A,H]_REG). The way channels are assigned to the RCER[A,H]_REG registers depends on the number of receive channel partitions (two or eight), as defined by the MCR1_REG[9] register RMCME bit.
- If a receive channel is disabled, any bits received in that channel are not transferred to the receive buffer (RB), and as a result, does not set the receiver ready bit (RRDY). Therefore, no DMA synchronization event (REVT) is generated and, if the receiver interrupt mode depends on RRDY (RINTM = 00b), no interrupt is generated.

As an example of how the McBSP behaves in the receive multichannel selection mode, suppose you enable only channels 0, 15, and 39 and that the frame length is 40. The McBSP:

- Accepts bits shifted in from the McBSP.DR pin in channel 0
- Ignores bits received in channels 1–14
- Accepts bits shifted in from the McBSP.DR pin in channel 15
- Ignores bits received in channels 16–38
- Accepts bits shifted in from the McBSP.DR pin in channel 39

15.2.5.6 Using Two Partitions (legacy only)

For multichannel selection operation in the receiver and/or the transmitter, you can use two partitions or eight partitions. If you choose the 2–partition mode (RMCME = 0 for reception, XMCME = 0 for transmission), McBSP channels are activated using an alternating scheme. In response to a frame–synchronization pulse, the receiver or transmitter begins with the channels in partition A and then alternates between partitions B and A until the complete frame has been transferred. When the next frame–synchronization pulse occurs, the next frame is transferred beginning with the channels in partition A.

15.2.5.6.1 Assigning Blocks to Partitions A and B

For reception, any two of the eight receive–channel blocks can be assigned to receive partitions A and B, which means up to 32 receive channels can be enabled at any given point in time. Similarly, any two of the eight transmit–channel blocks (up 32 enabled transmit channels) can be assigned to transmit partitions A and B.

For reception:

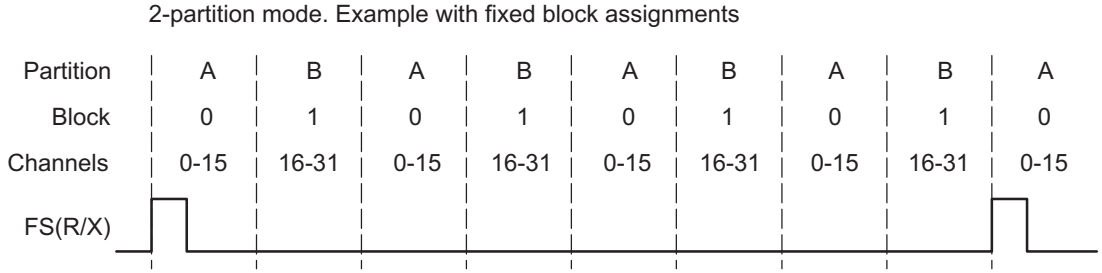
- Assign an even–numbered channel block (0, 2, 4, or 6) to receive partition A by writing to the RPABLK bits. In the receive multichannel selection mode, the channels in this partition are controlled by receive channel enable register A (RCERA_REG).
- Assign an odd–numbered block (1, 3, 5, or 7) to receive partition B with the RPBBLK bits. In the receive multichannel selection mode, the channels in this partition are controlled by receive channel enable register B (RCERB_REG).

For transmission:

- Assign an even–numbered channel block (0, 2, 4, or 6) to transmit partition A by writing to the XPABLK bits. In one of the transmit multichannel selection modes, the channels in this partition are controlled by transmit channel enable register A (XCERA_REG).
- Assign an odd–numbered block (1, 3, 5, or 7) to transmit partition B with the XPBBLK bits. In one of the transmit multichannel selection modes, the channels in this partition are controlled by transmit channel enable register B (XCERB_REG).

Figure 15-20 shows an example of alternating between the channels of partition A and the channels of partition B. Channels 0–15 have been assigned to partition A, and channels 16–31 have been assigned to partition B. In response to a frame–synchronization pulse, the McBSP begins a frame transfer with partition A and then alternates between partitions B and A until the complete frame is transferred.

Figure 15-20. Alternating Between Partitions A and B Channels



15.2.5.7 Transmit Multichannel Selection Modes

The MCR2_REG[1:0] register XMCM bits determine whether all channels or only selected channels are enabled and unmasked for transmission. The McBSP has three transmit multichannel selection modes (XMCM = 01b, XMCM = 10b, and XMCM = 11b), which are described in Table 15-6.

As an example of how the McBSP behaves in a transmit multichannel selection mode, suppose that XMCM = 01b (all channels disabled unless individually enabled) and that you have enabled only channels 0, 15, and 39. Suppose also that the frame length is 40. The McBSP:

- Shifts data to the McBSP.DX pin in channel 0
- Places the McBSP.DX pin in the high impedance state in channels 1–14
- Shifts data to the McBSP.DX pin in channel 15
- Places the McBSP.DX pin in the high impedance state in channels 16–38
- Shifts data to the McBSP.DX pin in channel 39

Table 15-6. Selecting a Transmit Multichannel Selection Mode with the XMCM Bit Field

XMCM	Transmit Multichannel Selection Mode
00b	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.
01b	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCER[A,H]_REG). If enabled, a channel in this mode is also unmasked.
	The MCR2_REG[9] register XMCME bit determines whether 32 channels or 128 channels are selectable in the XCER[A,H]_REG registers.
10b	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCER[A,H]_REG).
	The MCR2_REG[9] register XMCME bit determines whether 32 channels or 128 channels are selectable in the XCER[A,H]_REG registers.
11b	This mode is used for symmetric transmission and reception.
	All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (RCER[A,H]_REG). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCER[A,H]_REG).
	The MCR2_REG[9] register XMCME bit determines whether 32 channels or 128 channels are selectable in RCER[A,H]_REG registers and XCER[A,H]_REG registers.

15.2.5.7.1 Disabling/Enabling Versus Masking/Unmasking

For transmission, a channel can be:

- Enabled and unmasked (transmission can begin and can be completed
- Enabled but masked (transmission can begin but cannot be completed)
- Disabled (transmission cannot occur)

Table 15-7. McBSP Channel Control Options

Enabled channel	A channel that can begin transmission by passing data from the data transmit register (DXR_REG) to the transmit shift register (XSR) through XB.
Masked channel	A channel that cannot complete transmission. The McBSP.DX pin is held in the high impedance state; data cannot be shifted out on the McBSP.DX pin.
	In systems where symmetric transmit and receive provide software benefits, this feature allows transmit channels to be disabled on a shared serial bus. A similar feature is not needed for reception because multiple receptions cannot cause serial bus contention.
Disabled channel	A channel that is not enabled. A disabled channel is also masked.
	Because no DXR–to–XB copy occurs, the SPCR2_REG[1] register XRDY bit is not set. Therefore, no DMA synchronization event (XEVT) is generated, and if the transmit interrupt mode depends on XRDY (XINTM = 00b in SPCR2_REG[5:4]), no interrupt is generated.
	The SPCR2_REG[2] register XEMPTY bit is not affected.
Unmasked channel	A channel that is not masked. Data in the XSR(s) is shifted out on the McBSP.DX pin.

15.2.5.7.2 Activity on McBSP Pins for Different Values of XMCM Bit

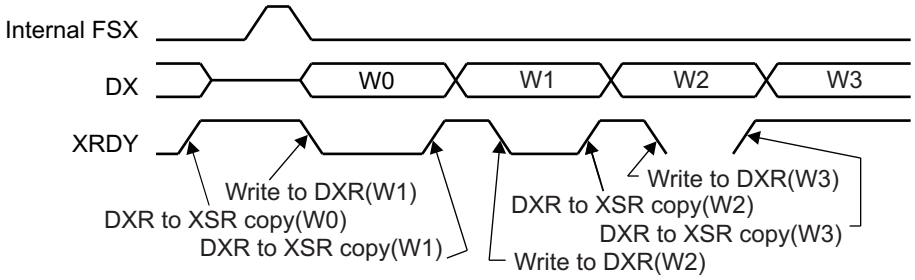
Figure 15-21 shows the activity on the McBSP pins for the various XMCM values. In all cases, the transmit frame is configured as follows:

- XPHASE = 0: Single–phase frame (required for multichannel selection modes)
- XFRLEN1 = 0000011b: 4 words per frame
- XWDLEN1 = 000b: 8 bits per word
- XMCME = 0: 2–partition mode (only partitions A and B used)

In the case where XMCM = 11b, transmission and reception are symmetric, which means the corresponding bits for the receiver (RPHASE, RFRLEN1, RWDLEN1, and RMCME) must have the same values as XPHASE, XFRLEN1, and XWDLEN1, respectively. In Figure 15-21, the arrows showing where the various events occur are only sample indications. Wherever possible, there is a time window in which these events can occur.

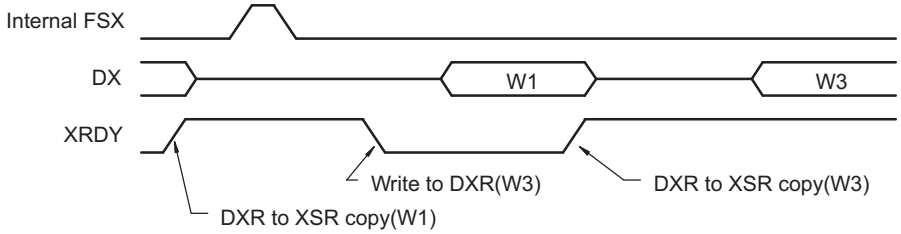
Figure 15-21. Activity on McBSP Pins for the Possible Values of XMCM Bit

(a) XMCM = 00b: All channels enabled and unmasked.



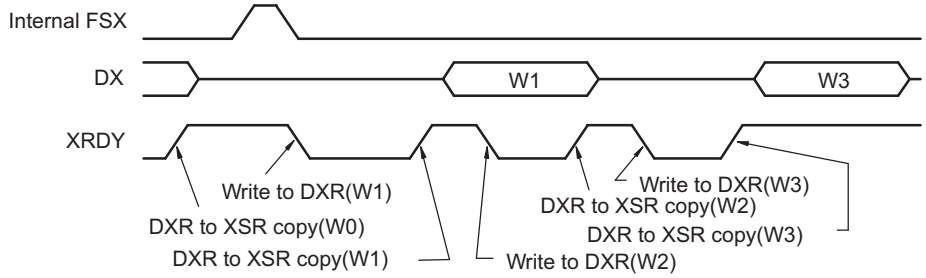
Words W0, W1, W2, W3 are written to the transmit buffer and W0, W1, W2, W3 from the transmit buffer are transferred by McBSP.DX

(b) XMCM = 01b, XPABLK = 00b, XCERA = 1010b: Only channels 1 and 3 enabled and unmasked



Words W1, W3 are written to the transmit buffer and W1, W3 from the transmit buffer are transferred by McBSP.DX

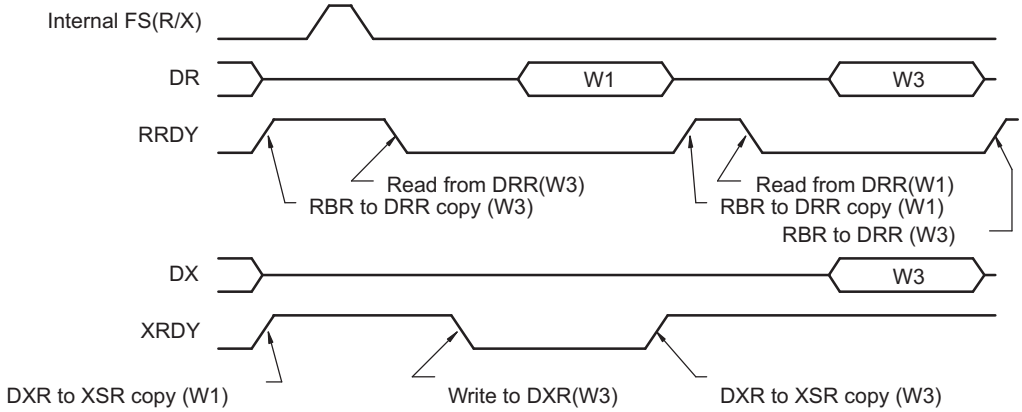
(c) XMCM = 10b, XPABLK = 00b, XCERA = 1010b: All channels enabled, only 1 and 3 unmasked



Words W0, W1, W2, W3 are written to the transmit buffer but only W1 and W3 from the transmit buffer are transferred by McBSP.DX

(d) XMCM = 11b, RPABLK = 00b, XPABLK = X, RCERA = 1010b, XCERA = 1000b

Receive channels: 1 and 3 enabled; Transmit channels: 1 and 3 enabled, but only 3 unmasked



Words W1, W3 are written to the transmit buffer but only W3 from the transmit buffer is transferred by McBSP.DX

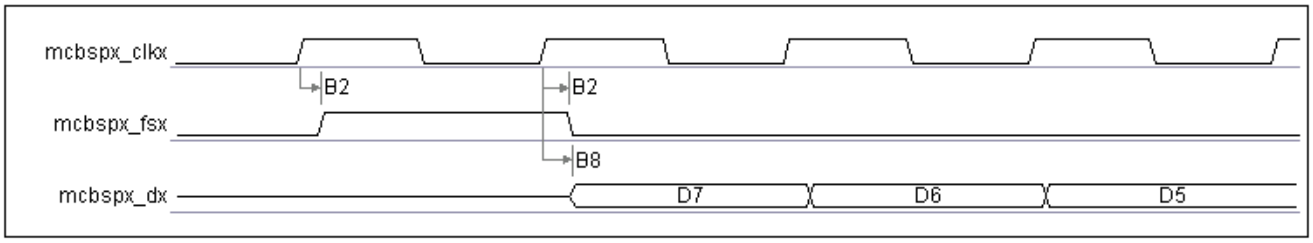
15.2.6 McBSP Full/Half Cycle Modes

The following figures are for data delay set to 1.

15.2.6.1 Transmit Full Cycle Mode

When configured in full cycle mode (XCCR[11] register, XFULL_CYCLE bit field), the FSX is sampled on the configured CLKX edge and the data is driven on the same configured edge:

Figure 15-22. Transmit Full Cycle Mode

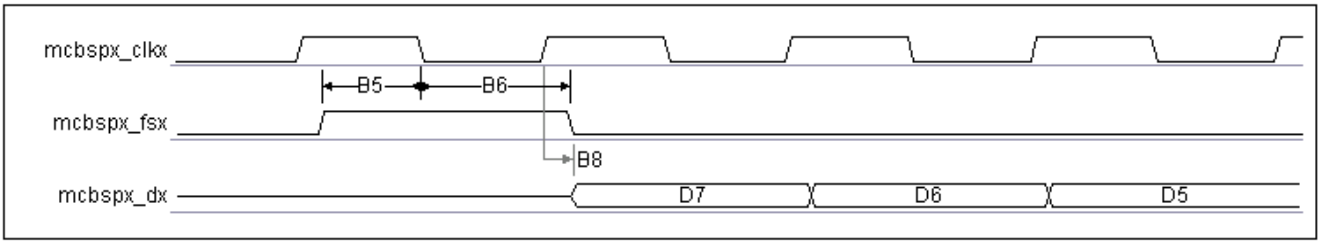


Data driven on positive CLKX edge, FSX sampled on positive CLKX edge.

15.2.6.2 Transmit Half Cycle Mode

When configured in half cycle mode (XCCR[11] register, XFULL_CYCLE bit field), the FSX is sampled on the opposite configured CLKX edge and the data is driven on the next configured edge:

Figure 15-23. Transmit Half Cycle Mode

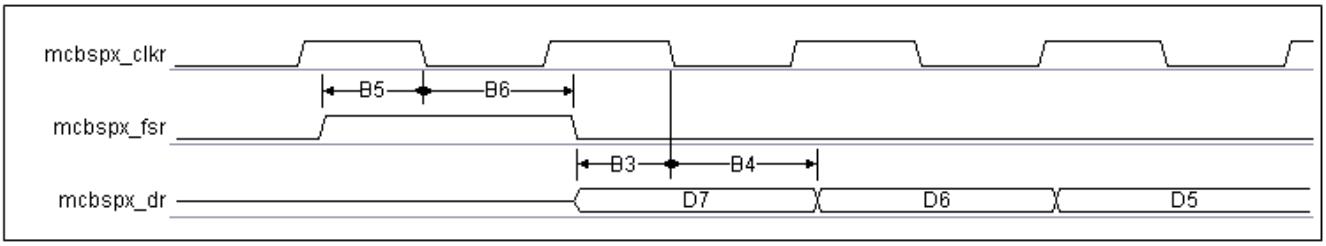


Data driven on positive CLKX edge, FSX sampled on negative CLKX edge.

15.2.6.3 Receive Full Cycle Mode

When configured in full cycle mode (RCCR[11] register, RFULL_CYCLE bit field), the FSR is sampled on the configured CLKR edge and the data is sampled on the same configured edge:

Figure 15-24. Receive Full Cycle Mode

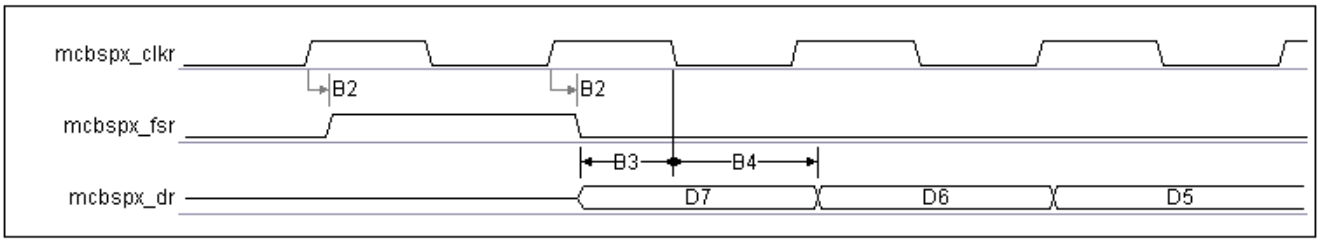


Data sampled on negative CLKR edge, FSR sampled on negative CLKR edge.

15.2.6.4 Receive Half Cycle Mode

When configured in half cycle mode (RCCR[11] register, RFULL_CYCLE bit field), the FSR is sampled on the opposite configured CLKR edge and the data is sampled on the next configured edge:

Figure 15-25. Receive Half Cycle Mode



Data sampled on negative CLKR edge, FSR sampled on positive CLKR edge.

15.2.7 Power Management

15.2.7.1 ForceIdle Behavior

When configured in ForceIdle the module should respond immediately (according to the OCP guideline description) to a McBSP.MIDLEREQ, regardless of the internal state of the module. Entering in this mode, the module will freeze all the internal activity when the clocks are switched off by the power management external module. If the functional part, transmit or receive, is running within this period of time, the internal state of the module will not be idle (FSM states, processes, etc.), and when the module exits from the ForceIdle state unexpected behavior may happen in both receiver and transmitter. In order to avoid this both receive and transmit parts need to be disabled by software prior to MIdleReq assertion.

15.2.7.2 SmartIdle Behavior

When configured in SmartIdle, the sources for wake-up generation are a subset of the interrupt sources. The wakeup sources are enabled by setting the corresponding bit in WAKEUPEN register.

For receive WAKEUP there are 4 possible configuration scenarios:

- **RRDYEN** - The McBSP asserts the McBSP.WAKEUP request when the RB reaches the high threshold (THRSH1_REG + 1) register value. If the corresponding bit is set in IRQENABLE register, McBSP sends an interrupt (McBSP.COMMONIRQ) request to the CPU when exiting from idle mode (interrupt will be asserted once the RRDY bit changes from 0 to 1, indicating that receive data is ready to be read).
- **REOFEN** - The McBSP asserts the McBSP.WAKEUP request at the end of the frame. If the corresponding bit is set in IRQENABLE register McBSP sends an interrupt (McBSP.COMMONIRQ) request to the CPU when exit from idle mode.
- **RSYNCERREN** - The McBSP asserts the McBSP.WAKEUP request when an unexpected receive frame-synchronization pulse is detected. If the corresponding bit is set in IRQENABLE register, McBSP sends an interrupt (McBSP.COMMONIRQ) request o the CPU when exiting from idle mode (interrupt will be asserted once the RSYNCERR bit changes from 0 to 1, indicating that a receive error occurred).

For transmit WAKEUP there are four possible configuration scenarios:

- **XRDYEN** - The McBSP asserts the McBSP.WAKEUP request when the XB reaches the high threshold (THRSH2_REG + 1) register value. If the corresponding bit is set in IRQENABLE register, McBSP sends an interrupt (McBSP.COMMONIRQ) request to the CPU when exiting from idle mode (interrupt will be asserted once the XRDY bit changes from 0 to 1, indicating that transmit buffer data is ready to accept new data).
- **XEOFEN** - The McBSP asserts the McBSP.WAKEUP request at the end of the frame. If the corresponding bit is set in IRQENABLE register, the McBSP sends an interrupt (McBSP.COMMONIRQ) request to the CPU when exiting from idle mode.
- **XFSXEN** - The McBSP sends a McBSP.WAKEUP request when a transmit frame-synchronization pulse is detected while the module is in idle mode. If the corresponding bit is set in IRQENABLE register, the McBSP sends an interrupt (McBSP.COMMONIRQ) request to the CPU when exiting from idle mode.
- **XSYNCERREN** - The McBSP asserts the McBSP.WAKEUP request when an unexpected transmit frame-synchronization pulse is detected. If the corresponding bit is set in IRQENABLE register, the McBSP sends an interrupt (McBSP.COMMONIRQ) request to the CPU when exiting from idle mode (interrupt will be asserted once the XSYNCERR bit changes from 0 to 1, indicating that a transmit error occurred).

15.2.7.2.1 Analysis of the Receiver Smart Idle Behavior

The analysis of the power mode behavior is depicted in Table 15-8.

Table 15-8. Analysis of the Receiver Smart Idle Behavior

CLKRM	CLKXM	Behavior
0	0	McBSP is configured as a slave, (functional clocks are provided from outside). The module will acknowledge the SmartIdle mode request as soon as there is no pending DMA, interrupt request or transmit buffer threshold synchronization (only when wake-up event is set on transmit threshold reached), regardless of the CLOCKACTIVITY settings or receive and transmit activity.
0	1	McBSP is configured as a transmit master (the source clock can be CLKS, CLKR or OCP). When OCP clock is used as source clock and CLOCKACTIVITY is indicating that the OCP clock will be switched off, or CLKS clock is used as source clock and CLOCKACTIVITY is indicating that the functional clock is switched off, the McBSP will not acknowledge the SmartIdle request unless: • the transmit part is disabled (XDISABLE) or under software reset (XRST) and the receive part is not using the transmit loop-back clock (ALBCTRLRX[0] pin is asserted). • both transmit and receive parts are disabled (XDISABLE/RDISABLE) or under software reset (XRST/RRST) The SmartIdle acknowledge is asserted as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached) and the pending transmit and/or receive frames where completed in case of transmit and/or receive disable. When OCP or CLKS is used as source clock and the CLOCKACTIVITY indicates that the corresponding clock will not be switched off, the module will acknowledge the SmartIdle mode request as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached). When CLKR is used as source (functional clock is provided from outside) then the module will acknowledge the SmartIdle mode request as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached), regardless of the CLOCKACTIVITY settings.
1	0	McBSP is configured as a receive master (the source clock can be CLKS, CLKX or OCP). When OCP clock is used as source clock and CLOCKACTIVITY is indicating that the OCP clock will be switched off, or CLKS clock is used as source clock and CLOCKACTIVITY is indicating that the functional clock is switched off, the McBSP will not acknowledge the SmartIdle request unless: • the receive part is disabled (RDISABLE) or under software reset (RRST). The acknowledge is asserted as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached) and the pending transmit and/or receive frames where completed in case of transmit and/or receive disable. When OCP or CLKS is used as source clock and the CLOCKACTIVITY indicates that the corresponding clock will not be switched off, the module will acknowledge the SmartIdle mode request as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached). When CLKX is used as source (functional clock is provided from outside) then the module will acknowledge the SmartIdle mode request as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached), regardless of the CLOCKACTIVITY settings.

Table 15-8. Analysis of the Receiver Smart Idle Behavior (continued)

CLKRM	CLKXM	Behavior
1	1	McBSP is configured as transmit and receive master (the source clock can be CLKS or OCP). When OCP clock is used as source clock and CLOCKACTIVITY is indicating that the OCP clock will be switched off, or CLKS clock is used as source clock and CLOCKACTIVITY is indicating that the functional clock is switched off, the McBSP will not acknowledge the SmartIdle request unless: • both transmit and receive parts are disabled (XDISABLE/RDISABLE) or under software reset (XRST/RRST) The SmartIdle acknowledge is asserted as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached) and the pending transmit and/or receive frames where completed in case of transmit and/or receive disable. Note that no wakeup event will be available in this mode since the entire McBSP and remote device activity will be frozen. When OCP or CLKS is used as source clock and the CLOCKACTIVITY indicates that the corresponding clock will not be switched off, the module will acknowledge the SmartIdle mode request as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached).

Notes:

- The RFSREN/XFSXEN mode is suitable for wake-up generation when both clocks (functional and OCP) are switched off and McBSP.FSR/McBSP.FSX is configured as input. The frame synchronization-pulse is asynchronously detected during idle.
- The RSYNCERREN/XSYNCERREN mode can be used to wake-up the McBSP only by a remote module implementing such a feature, to use this type of error in order to trigger a wake-up. This mode requires functional clock to be active.
- When McBSP.FSR/McBSP.FSX is configured as an output, the McBSP.FSR/ McBSP.FSX wake-up generation makes no sense (the module cannot be in SmartIdle mode).
- Detection of RSYNCERR/XSYNCERR during idle mode can be used only when McBSP.FSR/ McBSP.FSX is configured as an input and the remote system knows to assert such an error in order to trigger the wake-up of the McBSP.
- The module does not implement IRQ assertion when configured as GPIO; also a wake-up capability in this mode is not available.

15.2.8 Programming Model

15.2.8.1 McBSP Initialization Procedure

The serial port initialization procedure is as follows:

1. Clear SPCR1_REG[0] register RRST bit, SPCR2_REG[7] register FRST bit, and SPCR2_REG[0] register XRST bit to 0. If coming out of a global reset, this step is not required.
2. While the serial port is in the reset state, program only the McBSP configuration registers (not the data registers) as required.
3. Wait for two clock cycles. This ensures proper internal synchronization
4. Set SPCR1_REG[0] register RRST bit and SPCR2_REG[0] register XRST bit to 1 to enable the serial port. Make sure that as you set these reset bits, you do not modify any of the other bits in SPCR1_REG and SPCR2_REG registers. Otherwise, you would change the configuration set in step 2.
5. Set up data acquisition as required (such as writing to DXR_REG register).
6. Set SPCR2_REG[7] register FRST bit to 1 if internally generated frame synchronization is required.
7. Wait for two clock cycles for the receiver and transmitter to become active.

Alternatively, on write (step 1 or 5), the transmitter and receiver can be placed in or taken out of reset by modifying the desired bit.

The previous procedure for reset/initialization can be applied in general when the receiver or transmitter has to be reset during its normal operation, and also when the sample rate generator is not used for either operation.

Notes:

- The necessary duration of the active-low period of XRST or RRST is at least two CLKR/CLKX cycles.
- The appropriate bits in serial port configuration registers (SPCR1_REG, PCR_REG, RCR1_REG, RCR2_REG, XCR1_REG, XCR2_REG, SRGR1_REG and SRGR2_REG) should only be modified when the affected portion of the serial port is in its reset state.
- In most cases, the data transmit register (DXR_REG) should be loaded by the CPU or the DMA controller only when the transmitter is enabled (XRST = 1). An exception to this rule is when these registers are used for loop back internal data.
- The bits of the channel control registers (MCR1_REG, MCR2_REG, RCER{A-H}_REG and XCER{A-H}_REG) can be modified at any time as long as they are not being used by the current reception/transmission in a multichannel selection mode
- The sample rate generator is reset by clearing SPCR2_REG[6] register GRST bit to 0.

15.2.8.2 Reset and Initialization Procedure for the Sample Rate Generator

To reset and initialize the sample rate generator:

1. Place the McBSP/sample rate generator in reset. During a Global reset, the sample rate generator, the receiver, and the transmitter reset bits (GRST, RRST, and XRST) are automatically forced to 0. Otherwise, during normal operation, the sample rate generator can be reset by making GRST = 0 in SPCR2_REG[6] register, provided that CLKG and/or FSG is not used by any portion of the McBSP. Depending on your system you may also want to reset the receiver (RRST = 0 in SPCR1_REG[0]) and reset the transmitter (SPCR2_REG[0] register XRST bit = 0).
2. Program the registers that affect the sample rate generator. Program the sample rate generator registers (SPCR1_REG and SPCR2_REG) as required for your application. If necessary, other control registers can be loaded with desired values provided the respective portion of the McBSP (the receiver or transmitter) is in reset. After the sample rate generator registers are programmed, wait 2 CLKSRG cycles. This ensures proper synchronization internally.
3. Enable the sample rate generator (take it out of reset). In SPCR2_REG[6], set GRST bit to 1 to enable the sample rate generator. After the sample rate generator is enabled, wait two CLKG cycles for the sample rate generator logic to stabilize. On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to (input clock frequency)/(CLKGDV + 1) where the input clock is selected with the SCLKME bit of PCR_REG[7] register and the SRGR2_REG[13] register CLKSM bit in one of the following configurations:

Table 15-9. Input Clock Selection for Sample Rate Generator

SCLKME bit	CLKSM bit	Input Clock for Sample Rate Generator
0	0	Signal on McBSP.CLKS pin
0	1	McBSP_FCLK clock
1	0	Signal on McBSP.CLKR pin
1	1	Signal on McBSP.CLKX pin

4. If necessary, enable the receiver and/or the transmitter. If necessary, remove the receiver and/or transmitter from reset by setting RRST and/or XRST = 1.
5. Necessary, enable the frame–synchronization logic of the sample rate generator. After the required data acquisition setup is done (DXR_REG is loaded with data), set GRST = 1 in SPCR2_REG[6] if an internally generated frame–synchronization pulse is required. FSG is generated with an active–high edge after the programmed number of CLKG clocks (FPER + 1) has elapsed.

15.2.8.3 Data Transfer DMA Request Configuration

To configure the McBSP receive/transmit data DMA requests, perform the following procedure:

1. Write the receive THRSH1_REG register with the required receive DMA request length (the length of the transfer is the same as the threshold value + 1). As long as the RB occupied locations level is above or equal to the THRSH1_REG value + 1, the DMA request will be asserted. After transferring the configured THRSH1_REG + 1 number of words, the receive DMA request (McBSP.REVNT) will be deasserted and reasserted as soon as the conditions are met again.
   Note that in case of a number of transfers that exceed the number of the programmed DMA length the McBSP will respond to the command, and will perform the transfer regardless of the receive buffer empty condition. When the receive buffer is empty a data transfer access will trigger a receive underflow interrupt, if enabled by RUNDFL_EN bit in IRQENSTAT_REG register.
2. Write the transmit THRSH2_REG register with the required transmit DMA request length (the length of the transfer is the same as the threshold value + 1). As long as the XB free locations level is above or equal to the THRSH2_REG value + 1, the DMA request will be asserted. After transferring the configured THRSH2_REG + 1 number of words, the transmit DMA request (McBSP.XEVNT) will be deasserted and reasserted as soon as the conditions are met again.
   Note that in case of a number of transfers that exceed the number of the programmed DMA length the McBSP will respond to the command, and will perform the transfer regardless of the transmit buffer full condition. When the transmit buffer is full a data transfer access will trigger a transmit overflow interrupt, if enabled by XOVFL_EN bit in IRQENSTAT_REG register

15.2.8.4 Interrupt Configuration

The McBSP offers two interrupt schemes:

- OCP compliant interrupt request scheme using a common receive/transmit interrupt request line.
- The legacy interrupt compliant scheme using 3 interrupt lines: one for receive, one for transmit and the receive overflow interrupt line.

The OCP compliant interrupt line can be configured by using the IRQENABLE register. When the IRQSTATUS bit is set and the corresponding IRQENABLE bit is set to 1, the interrupt line is asserted. Writing one to a bit in IRQSTATUS register will clear the bit.

There are several conditions, which may be configured to generate an interrupt as follows:

- Transmit Buffer Overflow (XOVFLSTAT bit is set to 1 when transmit buffer overflow; the data which is written while overflow condition is discarded).
- Transmit Buffer Underflow (XUNDFLSTAT bit is set to 1 when the transmit data buffer is empty new data is required to be transmitted).
- Transmit Buffer Threshold Reached (XRDY bit is set to 1 when the transmit buffer free locations are equal or above the (THRSH2_REG + 1) value).
- Transmit End of Frame (XEOF is set to 1 when a complete frame was transmitted).
- Transmit Frame Synchronization (XFSX bit is set to 1 when a new transmit frame synchronization is asserted).
- Transmit Frame Synchronization Error (XSYNCERR is set to 1 when a transmit frame synchronization error is detected).
- Receive Buffer Overflow (ROVFLSTAT bit is set to 1 when receive buffer overflow; the data which is written while overflow condition is discarded).
- Receive Buffer Underflow (RUNDFLSTAT bit is set to 1 when read operation is performed to the receive data register while receive buffer is empty; data read while underflow condition is undefined).
- Receive Buffer Threshold Reached (RRDY bit is set to 1 when the receive buffer occupied locations are equal or above the (THRSH1_REG + 1) value).
- Receive End of Frame (REOF is set to 1 when a complete frame was received).
- Receive Frame Synchronization (RFSR bit is set to 1 when a new receive frame synchronization is asserted).
- Receive Frame Synchronization Error (RSYNCERR is set to 1 when a receive frame synchronization error is detected).

15.2.8.5 Receiver Configuration

To configure the McBSP receiver, perform the following procedure:

1. Place the McBSP/receiver in reset.
2. Program the McBSP registers for the desired receiver operation.
3. Take the receiver out of reset.

15.2.8.5.1 Programming the McBSP Registers for the Desired Receiver Configuration

The following is a list of important tasks to be performed when you are configuring the McBSP receiver. Each task corresponds to one or more McBSP register bit fields. Note that SRG is Sample Rate Generator.

Global Behavior

1. Set the receiver pins to operate as McBSP pins.
2. Enable/disable the digital loop back mode.
3. Enable/disable the analog loop back mode.
4. Enable/disable the synchronous transmit-receive mode.
5. Enable/disable the receive multichannel selection mode.

Data behavior:

1. Choose 1 or 2 phases for the receive frame.
2. Set the receive word length(s).
3. Set the receive frame length (one word per phase if dual-phase selected).
4. Enable/disable the receive frame–synchronization ignore function.
5. Set the receive data delay.
6. Set the receive sign–extension and justification mode.
7. Set the receive interrupt mode.

Frame–synchronization behavior:

1. Set the receive frame–synchronization mode.
2. Set the receive frame–synchronization polarity.
3. Set the sample rate generator (SRG) frame–synchronization period and pulse width.

Clock behavior:

1. Set the receive clock mode.
2. Set the receive clock polarity.
3. Set the SRG clock divide–down value.
4. Set the SRG clock synchronization mode.
5. Set the SRG clock mode (choose an input clock).
6. Set the SRG input clock polarity.

15.2.8.5.2 Resetting and Enabling the Receiver

The first step of the receiver configuration procedure is to reset the receiver, and the last step is to enable the receiver (to take it out of reset).

The serial port can be reset in the following two ways:

- A global reset places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed, GRST, FRST, RRST and XRST bits = 0, keeping the entire serial port in the reset state.
- The serial port transmitter and receiver can be reset directly using the RRST and XRST bits in the SPCR1_REG register. The sample rate generator can be reset directly using the GRST bit in SPCR2_REG[6] register.

15.2.8.5.3 Set the Receiver Pins to Operate as McBSP Pins

The RIOEN bit (PCR_REG[12]) determines whether the receiver pins are McBSP pins or general–purpose I/O pins.

Refer to Section 15.2.8.7 for further information.

15.2.8.5.4 Enable/Disable the Synchronous Transmit-Receive Mode

The ALBCTRLRX input pin is used to configure the synchronous transmit-receive mode. The ALBCTRLRX[0/1] input pin selection is determined by programming the Control Module AUD_CTRL Register.

The ALBCTRLRX [0] controls the multiplexer which connects the functional receive input clock (CLKR is connected to the CLKX input pin CLKXI when ALBCTRLRX[0] = 1).

The ALBCTRLRX[1] controls the multiplexer which connects the receive frame synchronization (FSR is connected to the FSX input pin FSXI when ALBCTRLRX[1] = 1).

15.2.8.5.5 Enable/Disable the Analog Loop Back Mode

The ALB bit (SPCR1_REG[15]) determines whether the analog loop-back mode is on or off.

In the analog loop-back mode, the receive signals are connected internally through multiplexers to the corresponding transmit loop back signals (DR is connected to transmit loop back data on DXI pin, FSR is connected to FRX input pin FSXI, and CLKR is connected to the CLKX input pin CLKXI). This mode allows testing of serial port; the McBSP receives the data it transmits. This loop back mode is done through pads testing also the IO buffers.

15.2.8.5.6 Enable/Disable the Digital Loop Back Mode

The DLB bit (XCCR_REG) determines whether the digital loop-back mode is on or off. In the digital loop-back mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals (DR is connected to transmit data on DX output pin DXO, FSR is connected to FRX output pin FSXO, and CLKR is connected to the CLKX output pin CLKXO). This mode allows testing of serial port; the McBSP receives the data it transmits. This loop back mode is not done through pads, all output signals being disabled (CLKREN, CLKXEN, FSREN, FSRXEN, DXEN are not active).

Note that in digital loop back mode the sample rate generator and the frame synchronization generator need to be enabled in order to generate the CLKX and FSX signals.

15.2.8.5.7 Enable/Disable the Receive Multichannel Selection Mode

The RMCM bit (MCR1_REG[0]) determines whether the receive multichannel selection mode is on or off.

15.2.8.5.8 Choose 1 or 2 Phases for the Receive Frame

The RPHASE bit (RCR2_REG[15]) determines whether the receive data frame has one or two phases. When dual-phase is selected the number of words per phase must be set to 1.

15.2.8.5.9 Set the Receive Word Length(s)

The RWDLEN1 (RCR1_REG[7:5]) and RWDLEN2 (RCR2_REG[7:5]) bit fields determine how many bits are in each serial word in phase 1 and in phase 2, respectively, of the receive data frame.

Each frame can have one or two phases, depending on the value that you load into the RPHASE bit. If a single–phase frame is selected, RWDLEN1 selects the length for every serial word received in the frame. If a dual–phase frame is selected, RWDLEN1 and RWDLEN2 must be cleared to 0 (one word per phase).

15.2.8.5.10 Set the Receive Frame Length

The RFRLEN1 (RCR1_REG[14:8]) and RFRLEN2 (RCR2_REG[14:8]) bit fields determine how many serial words are in phase 1 and in phase 2, respectively, of the receive data frame.

The receive frame length is the number of serial words in the receive frame. Each frame can have one or two phases, depending on value that you load into the RPHASE bit.

If a single–phase frame is selected (RPHASE = 0), the frame length is equal to the length of phase 1. If a dual–phase frame is selected (RPHASE = 1), the frame length is the length is two (one word for phase 1 plus the one word for phase 2).

The 7–bit RFRLEN1 field allow up to 128 words per phase when single-phase frame. See Table 15-10 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame–synchronization pulse.

Program the RFRLEN fields with [w minus 1], where w represents the number of words per phase. For the example, if you want a phase length of 128 words in phase 1, load 127 into RFRLEN1.

Table 15-10. How to Calculate the Length of the Receive Frame

RPHASE	RFRLEN1	RFRLEN2	Frame Length
0	0 <= RFRLEN1 <= 127	Don't care	(RFRLEN1 + 1) words
1	RFRLEN1 = 0	RFRLEN2 = 0	2 words

15.2.8.5.11 Set the Receive Reverse Mode

The RREVERSE bit field (RCR2_REG[4:3]) determines whether reverse (LSB first) data transfer option is chosen for McBSP reception.

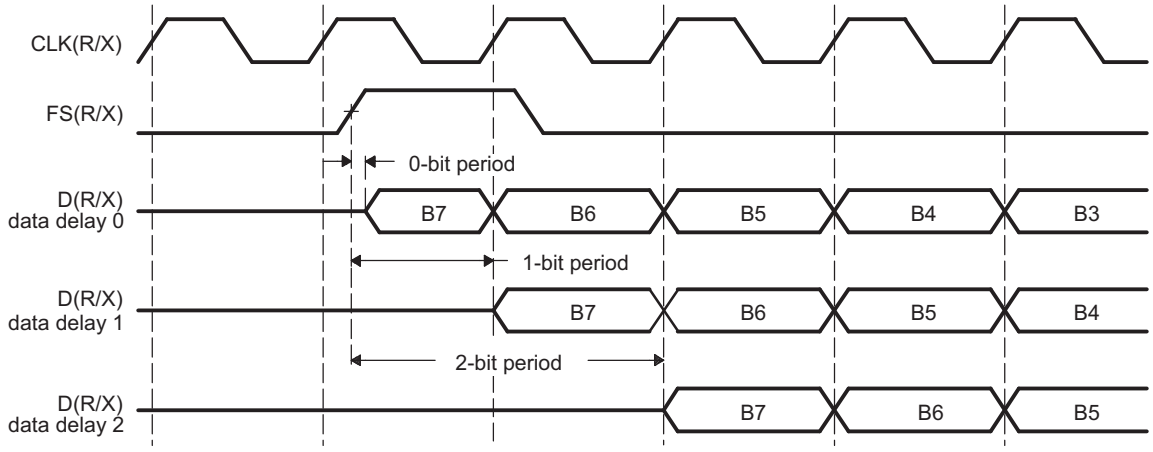
15.2.8.5.12 Set the Receive Data Delay

The RDATDLY bit field (RCR2_REG[1:0]) determines the length of the data delay for the receive frame.

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay.

RDATDLY specifies the data delay for reception. The range of programmable data delay is 0 to 2 bit–clocks (RDATDLY = 00b–10b), as shown in Figure 15-26. In this figure, the data transferred is an 8–bit value with bits labeled B7, B6, B5, and so on. Typically a 1–bit delay is selected, because data often follows a 1–cycle active frame–synchronization pulse.

Figure 15-26. Range of Programmable Data Delay



0-bit data delay:

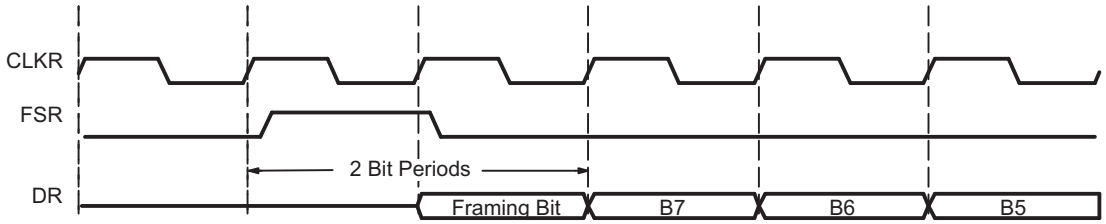
Normally, a frame–synchronization pulse is detected or sampled with respect to an edge of internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data may be received or transmitted. However, in the case of 0–bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception, this problem is solved because receive data is sampled on the first falling edge of CLKR where an active–high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus on McBSP.DX. The transmitter then asynchronously detects the frame–synchronization signal (FSX) going active high and immediately starts driving the first bit to be transmitted on the McBSP.DX pin.

2-bit delay:

A data delay of two bit periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of two bits (framing bit appears after a 1–bit delay and data appears after a 2–bit delay), the serial port essentially discards the framing bit from the data stream, as shown in Figure 15-27. In this figure, the data transferred is an 8–bit value with bits labeled B7, B6, B5, and so on.

Figure 15-27. 2-Bit Data Delay Used to Skip a Framing Bit



15.2.8.5.13 Set the Receive Sign-Extension and Justification Mode

The RJUST bit field (SPCR1_REG[14:13]) determines whether data received by the McBSP is sign–extended or not and how it is justified.

RJUST bit field selects whether data in RB is right– or left–justified (with respect to the MSB) in DRR_REG register and whether unused bits in DRR_REG are filled with zeroes or with sign bits.

The following tables show the effects of various RJUST values. Table 15-11 shows the effect on an example 12–bit receive–data value ABCh, Table 15-12 shows the effect on an example 20–bit receive–data value ABCDEh.

Table 15-11. Example: Use of RJUST Bit Field with 12-Bit Data Value ABCh

RJUST	Justification	Extension	Value in DRR_REG
00b	Right	Zero fill MSBs	0000 0ABCh
01b	Right	Sign extend data into MSBs	FFFF FABCh
10b	Left	Zero fill LSBs	ABC0 0000h
11b	Reserved	Reserved	Reserved

Table 15-12. Example: Use of RJUST Bit Field with 20-Bit Data Value ABCDE

RJUST	Justification	Extension	Value in DRR_REG
00b	Right	Zero fill MSBs	000A BCDEh
01b	Right	Sign extend data into MSBs	FFFA BCDEh
10b	Left	Zero fill LSBs	ABCD E000h
11b	Reserved	Reserved	Reserved

15.2.8.5.14 Set the Receive Interrupt Mode (legacy only)

The RINTM bit field (SPCR1_REG[5:4]) determines which event generates a receive interrupt request to the CPU. The receive interrupt (RINT) informs the CPU of changes to the serial port status. Four options exist for configuring this interrupt.

- RINTM = 00: RINT generated when the RRDY bit (SPCR1_REG[1]) changes from 0 to 1. Interrupt on every serial word by tracking the RRDY bit in the SPCR1_REG register. Regardless of the value of RINTM, RRDY can be read to detect the RRDY = 1 condition.
- RINTM = 01: RINT generated by an end–of–frame condition in the receive multichannel selection mode. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.
- RINTM = 10: RINT generated by a new receive frame–synchronization pulse. Interrupt on detection of receive frame–synchronization pulses. This generates an interrupt even when the receiver is in its reset state. This is done by synchronizing the incoming frame–synchronization pulse to the CPU clock and sending it to the CPU via RINT.
- RINTM = 11: RINT generated when RSYNCERR is set. Interrupt on frame–synchronization error. Regardless of the value of RINTM, RSYNCERR can be read to detect this condition. For information on using RSYNCERR, see Section 15.2.3.2.

The McBSP provides also a common interrupt line McBSP.COMMONIRQ, which can be used by setting the IRQENABLE register. All the above settings have equivalent enable bits in the IRQENABLE register in order to enable the common interrupt line (RRDYEN is equivalent with RINTM = 00, REOFEN is equivalent with RINTM = 01 (the interrupt is generated by an end-of-frame condition regardless of the multichannel selection mode), RFSREN is equivalent with RINTM = 10 and RSYNCERREN is equivalent with RINTM = 11 setting). This interrupt line has its own status register.

15.2.8.5.15 Set the Receive Frame-Synchronization Mode

FSRM bit (PCR_REG[10]), GSYNC bit (SRGR2_REG[15]), ALB bit (SPCR1_REG[15]) and DLB bit field (XCCR_REG) are used to determine the source for receive frame synchronization and the function of the McBSP.FSR pin.

Table 15-13 shows how you can select various sources to provide the receive frame–synchronization signal and the effect on the McBSP.FSR pin. The polarity of the signal on the McBSP.FSR pin is determined by the FSRP bit.

In digital loop-back mode (DLB = 1), the transmit frame–synchronization signal is used as the receive frame–synchronization signal.

Also in the analog loop back mode, the internal receive clock signal (CLKR), the internal receive frame–synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

In the digital loop-back mode (DLB = 1) or analog loop-back mode (ALB = 1), the transmit frame synchronization signal is used as the receive frame synchronization signal.

Table 15-13. FSRM and GSYNC Effects on Frame-Synchronization Signal and McBSP.FSR Pin

FSRM	GSYNC	Source of Receive Frame Synchronization	McBSP.FSR Pin Status
0	0 or 1	An external frame synchronization signal enters the McBSP through the McBSP.FSR pin. The signal is then inverted as determined by FSRP before being used as internal FSR.	Input
1	0	Internal FSR is driven by the sample rate generator frame–synchronization signal (FSG).	Output. FSG is inverted as determined by FSRP before being driven out on the McBSP.FSR pin.
1	1	Internal FSR is driven by the sample rate generator frame–synchronization signal (FSG).	Input. The external frame–synchronization input on the McBSP.FSR pin is used to synchronize CLKG and generate FSG pulses.

15.2.8.5.16 Set the Receive Frame-Synchronization Polarity

The FSRP bit (PCR_REG[2]) determines whether frame–synchronization pulses are active high or active low on the McBSP.FSR pin.

Receive frame–synchronization pulses can be generated internally by the sample rate generator or driven by an external source. The source of frame synchronization is selected by programming the PCR_REG[10] register FSRM mode bit. FSR is also affected by the SRGR2_REG[15] register GSYNC bit. For information about the effects of FSRM and GSYNC, see the Frame Synchronization Generation in the Sample Rate Generator section. Similarly, receive clocks can be selected to be inputs or outputs by programming the PCR_REG[8] register CLKRM mode bit (see the Clock Generation in the Sample Rate Generator section).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame–synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the McBSP.DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from an external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of the internal clock, CLK(R/X). Similarly, data on the McBSP.DX pin is output on the rising edge of internal CLKX.

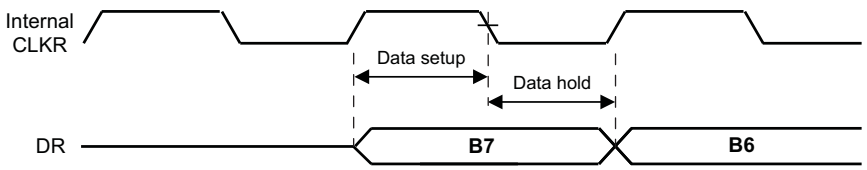
FSRP, FSXP, CLKRP, and CLKXP bits in the pin control register (PCR_REG) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame–synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active–low frame–synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active–high frame–synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling–edge triggered input clock on CLKX is inverted to a rising–edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising–edge triggered) clock, internal CLKX, is inverted before being sent out on the McBSP.CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal CLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising–edge triggered input clock on CLKR is inverted to a falling–edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling–edge triggered clock is inverted to a rising–edge triggered clock before being sent out on the McBSP.CLKR pin.

Note that CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. Figure 15-28 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 15-28. Data Externally Clocked on a Rising Edge and Sampled on a Falling Edge



15.2.8.5.17 Set the SRG Frame-Synchronization Period and Pulse Width

FPER bit field (SRGR2_REG[11:0]) is used to set the SRG frame-synchronization period and FWID bit field (SRGR1_REG[15:8]) is used to set the SRG pulse width. The sample rate generator can produce a clock signal, CLKG, and a frame–synchronization signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

On FSG, the period from the start of a frame–synchronization pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame–synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

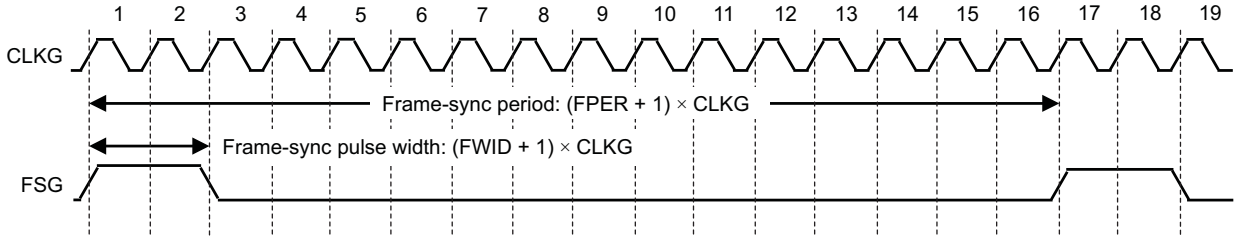
Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down–counters. The 12–bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8–bit FWID counter counts down from the programmed value (255 maximum) to 0.

Figure 15-29 shows a frame–synchronization period of 16 CLKG periods (FPER = 15 or 0000 1111b) and a frame–synchronization pulse with an active width of 2 CLKG periods (FWID = 1).

When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when GRST = 1 and FSGM = 1, a frame–synchronization pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

Figure 15-29. Frame of 16 CLKG Periods and Active Width of 2 CLKG Periods



15.2.8.5.18 Set the Receive Clock Mode

CLKRM bit (PCR_REG[8]), ALB bit (SPCR1_REG[15]) and DLB bit field (XCCR_REG) are used to set the receive clock mode.

Table 15-14 shows how you can select various sources to provide the receive clock signal and affect the McBSP.CLKR pin. The CLKRP bit determines the polarity of the signal on the McBSP.CLKR pin.

In the digital loop-back mode (DLB = 1) or analog loop-back mode (ALB = 1), the transmit clock signal is used as the receive clock signal.

Table 15-14. CLKRM Effect on Receive Clock Signal and McBSP.CLKR Pin

CLKRM	Source of Receive Clock	McBSP.CLKR Pin Status
0	The McBSP.CLKR pin is an input driven by an external clock. The external clock signal is inverted as determined by CLKRP before being used.	Input
1	The sample rate generator clock (CLKG) drives internal CLKR.	Output. CLKG, inverted as determined by CLKRP, is driven out on the McBSP.CLKR pin.

15.2.8.5.19 Set the Receive Clock Polarity

CLKRP bit (PCR_REG[0]) is used to set the receive clock polarity.

Receive frame–synchronization pulses can be generated internally by the sample rate generator or driven by an external source. The source of frame synchronization is selected by programming the PCR_REG[10] register FSRM mode bit. FSR is also affected by the GSYNC bit in SRGR2_REG[15]. Similarly, receive clocks can be selected to be inputs or outputs by programming the PCR_REG[8] register CLKRM mode bit.

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame–synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the McBSP.DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the McBSP.DX pin is output on the rising edge of internal CLKX. FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR_REG) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame–synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active–low frame–synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active–high frame–synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling–edge triggered input clock on CLKX is inverted to a rising–edge triggered clock before being sent to the transmitter. If CLKXP = 1 and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising–edge triggered) clock, internal CLKX, is inverted before being sent out on the McBSP.CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal CLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising–edge triggered input clock on CLKR is inverted to a falling–edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling–edge triggered clock is inverted to a rising–edge triggered clock before being sent out on the McBSP.CLKR pin.

Note that CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge.

15.2.8.5.20 Set the SRG Clock Divide-Down Value

CLKGDV bit field (SRGR1_REG[7:0]) is used to set the sample rate generator clock divide-down value.

The first divider stage generates the serial data bit clock from the input clock. This divider stage utilizes a counter, preloaded by CLKGDV, that contains the divide ratio value.

The output of the first divider stage is the data bit clock, which is output as CLKG and which serves as the input for the second and third stages of the divider.

CLKG has a frequency equal to 1/(CLKGDV + 1) of sample rate generator input clock. Thus, the sample generator input clock frequency is divided by a value between 1 and 256. The CLKG duty cycle is 50%.

15.2.8.5.21 Set the SRG Clock Synchronization Mode

GSYNC bit (SRGR2_REG[15]) is used to set the SRG clock synchronization mode.

15.2.8.5.22 Set the SRG Clock Mode (choose an input clock)

SCLKME bit (PCR_REG[7]) and CLKSM bit (SRGR2_REG[13]) are used to set the SRG clock mode.

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both, but CLKG is derived from an input clock.

15.2.8.5.23 Set the SRG Input Clock Polarity

CLKSP bit (SRGR2_REG[14]), CLKXP bit (PCR_REG[1]) and CLKRP bit (PCR_REG[0]) are used to set the SRG input clock polarity.

The sample rate generator can produce a clock signal (CLKG) and a frame–synchronization signal (FSG) for use by the receiver, the transmitter, or both. To produce CLKG and FSG, the sample rate generator must be driven by an input clock signal derived from the McBSP_FCLK clock or from an external clock on the McBSP.CLKS, McBSP.CLKX, or McBSP.CLKR pin. If you use a pin, choose a polarity for that pin by using the appropriate polarity bit (CLKSP for the McBSP.CLKS pin, CLKXP for the McBSP.CLKX pin, CLKRP for the McBSP.CLKR pin). The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

15.2.8.6 Transmitter Configuration

To configure the McBSP transmitter, perform the following procedure:

1. Place the McBSP transmitter in reset
2. Program the McBSP registers for the desired transmitter
3. Take the transmitter out of reset

These three steps are detailed in the following subsections.

15.2.8.6.1 Programming the McBSP Registers for the Desired Transmitter Operation

The following is a list of important tasks to be performed when you are configuring the McBSP transmitter. Each task corresponds to one or more McBSP register bit fields.

Global behavior:

1. Set the transmitter pins to operate as McBSP pins.
2. Enable/disable the digital loop back mode.
3. Enable/disable the analog loop back mode.
4. Enable/disable the synchronous transmit-receive mode.
5. Enable/disable transmit multichannel selection.

Data behavior:

1. Choose 1 or 2 phases for the transmit frame.
2. Set the transmit word length(s).
3. Set the transmit frame length (one word per phase if dual-phase selected).
4. Enable/disable the transmit frame–synchronization ignore function.
5. Set the transmit data delay.
6. Set the transmit DXENA mode.
7. Set the transmit interrupt mode.

Frame-synchronization behavior:

1. Set the transmit frame–synchronization mode.
2. Set the transmit frame–synchronization polarity.
3. Set the SRG frame–synchronization period and pulse width.

Clock behavior:

1. Set the transmit clock mode.
2. Set the transmit clock polarity.
3. Set the SRG clock divide–down value.
4. Set the SRG clock synchronization mode.
5. Set the SRG clock mode (choose an input clock).
6. Set the SRG input clock polarity.

15.2.8.6.2 Resetting and Enabling the Transmitter

The first step of the transmitter configuration procedure is to reset the transmitter, and the last step is to enable the transmitter (to take it out of reset). The serial port can be reset in the following two ways:

1. A global reset places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed, GRST, FRST, RRST and XRST bits = 0, keeping the entire serial port in the reset state.
2. The serial port transmitter and receiver can be reset directly using the RRST and XRST bits in the SPCR1_REG register. The sample rate generator can be reset directly using the GRST bit in SPCR2_REG register.

15.2.8.6.3 Set the Transmitter Pins to Operate as McBSP Pins

The XIOEN bit (PCR_REG[13]) determines whether the transmitter pins are McBSP pins or general–purpose I/O pins.

Refer to Section 15.2.8.7 for further information.

15.2.8.6.4 Enable/Disable the Digital Loop Back Mode

The DLB bit (XCCR_REG) determines whether the digital loop-back mode is on or off. In the digital loop-back mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals (DR is connected to transmit data on DX output pin DXO, FSR is connected to FRX output pin FSXO, and CLKR is connected to the CLKX output pin CLKXO). This mode allows testing of serial port; the McBSP receives the data it transmits. This loop back mode is not done through pads, all output signals being disabled (CLKREN, CLKXEN, FSREN, FSRXEN, DXEN are not active).

Note that in digital loop back mode the sample rate generator and the frame synchronization generator need to be enabled in order to generate the CLKX and FSX signals.

15.2.8.6.5 Enable/Disable the Transmit Multichannel Selection

The XMCM bit field (MCR2_REG[1:0]) determines whether the transmit multichannel selection mode is on or off.

15.2.8.6.6 Choose 1 or 2 Phases for the Transmit Frame

The XPHASE bit (XCR2_REG[15]) determines whether the transmit data frame has one or two phases.

15.2.8.6.7 Set the Transmit Word Length(s)

The XWDLEN1 (XCR1_REG[7:5]) and XWDLEN2 (XCR2_REG[7:5]) bit fields determine how many bits are in each serial word in phase 1 and in phase 2, respectively, of the transmit data frame.

Each frame can have one or two phases, depending on the value that you load into the XPHASE bit. If a single–phase frame is selected, XWDLEN1 selects the length for every serial word received in the frame. If a dual–phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame and XWDLEN2 determines the word length in phase 2 of the frame.

15.2.8.6.8 Set the Transmit Frame Length

The XFRLEN1 (XCR1_REG[14:8]) and XFRLEN2 (XCR2_REG[14:8]) bit fields determine how many serial words are in phase 1 and in phase 2, respectively, of the transmit data frame.

The transmit frame length is the number of serial words in the transmit frame. Each frame can have one or two phases, depending on value that you load into the XPHASE bit. If a single–phase frame is selected (XPHASE = 0), the frame length is equal to the length of phase 1. If a dual–phase frame is selected (XPHASE = 1), the frame length is 2, the length of phase 1 (one word) plus the length of phase 2 (one word).

The 7–bit XFRLEN fields allow up to 128 words per phase. See Table 15-15 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame–synchronization pulse.

Program the XFRLEN fields with [w minus 1], where w represents the number of words per phase. For the example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.

Table 15-15. How to Calculate the Length of the Transmit Frame

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	0 <= XFRLEN1 <= 127	Don't care	(XFRLEN1 + 1) words
1	XFRLEN1= 0	XFRLEN2 = 0	2 words

15.2.8.6.9 Set the Transmit Reverse Mode

The XREVERSE bit field (XCR2_REG[4:3]) determines whether reverse (LSB first) data transfer option is chosen for McBSP reception.

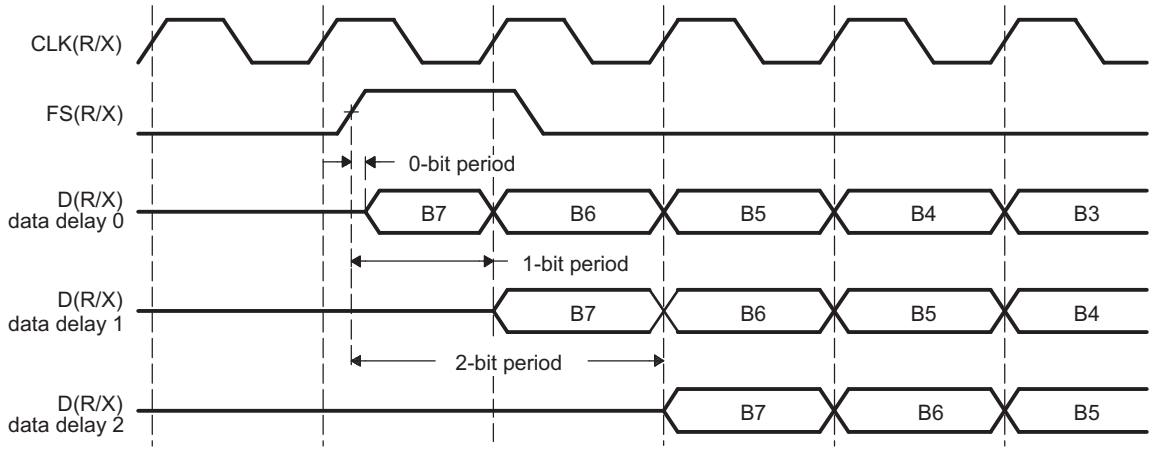
15.2.8.6.10 Set the Transmit Data Delay

The XDATDLY bit field (XCR2_REG[1:0]) determines the length of the data delay for the transmit frame.

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay.

XDATDLY specifies the data delay for reception. The range of programmable data delay is 0 to 2 bit–clocks (XDATDLY = 00b–10b), as shown in Figure 15-30. In this figure, the data transferred is an 8–bit value with bits labeled B7, B6, B5, and so on. Typically a 1–bit delay is selected, because data often follows a 1–cycle active frame–synchronization pulse.

Figure 15-30. Range of Programmable Data Delay

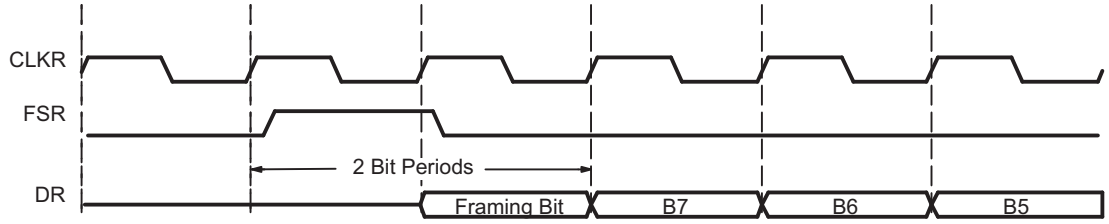


0-bit data delay:

Normally, a frame–synchronization pulse is detected or sampled with respect to an edge of internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data may be received or transmitted. However, in the case of 0–bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception, this problem is solved because receive data is sampled on the first falling edge of CLKR where an active–high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus on McBSP.DX. The transmitter then asynchronously detects the frame–synchronization signal (FSX) going active high and immediately starts driving the first bit to be transmitted on the McBSP.DX pin.

Figure 15-31. 2-Bit Data Delay Used to Skip a Framing Bit



15.2.8.6.11 Set the Transmit DXENA Mode

DXENA bit (SPRC1_REG[7]) is used to set the transmit DXENA (DX delay enable) mode.

The DXENA bit controls the delay enabler on the McBSP.DX pin. Set DXENA to enable an extra delay for turn–on time. This bit does not control the data itself, so only the first bit is delayed (the delay is given by a combinatorial delay buffer). The inserted delay: 7 ns, 14 ns (default), 20 ns or 28 ns can be set using the DXENDLY bit field in XCCR_REG register. If you tie together the McBSP.DX pins of multiple McBSPs, make sure DXENA = 1 to avoid having more than one McBSP transmitting on the data line at one time.

15.2.8.6.12 Set the Transmit Interrupt Mode (legacy only)

The XINTM bit field (SPCRX_REG[5:4]) determines which event generates a transmit interrupt request to the CPU.

The transmit interrupt (XINT) informs the CPU of changes to the serial port status. Four options exist for configuring this interrupt.

- XINTM = 00: XINT generated when the XRDY bit (SPCR2_REG[1]) changes from 0 to 1. Interrupt on every serial word by tracking the XRDY bit in the SPCR2_REG register. Regardless of the value of XINTM, XRDY can be read to detect the XRDY = 1 condition.
- XINTM = 01: XINT generated by an end–of–frame condition in the transmit multichannel selection mode. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.
- XINTM = 10: XINT generated by a new transmit frame–synchronization pulse. Interrupt on detection of transmit frame–synchronization pulses. This generates an interrupt even when the transmitter is in its reset state. This is done by synchronizing the incoming frame–synchronization pulse to the CPU clock and sending it to the CPU via XINT.
- XINTM = 11: XINT generated when XSYNCERR is set. Interrupt on frame–synchronization error. Regardless of the value of XINTM, XSYNCERR can be read to detect this condition.

The McBSP provides also a common interrupt line McBSP.COMMONIRQ, which can be used by setting the IRQENABLE register. All the above settings have equivalent enable bits in the IRQENABLE register in order to enable the common interrupt line (XDYEN is equivalent with XINTM = 00, XEOFEN is equivalent with XINTM = 01 (the interrupt is generated by an end-of-frame condition regardless of the multichannel selection mode), XFSXEN is equivalent with XINTM = 10 and XSYNCERREN is equivalent with XINTM = 11 setting). This interrupt line has its own status register.

15.2.8.6.13 Set the Transmit Frame-Synchronization Mode

FSXM bit (PCR_REG[11]) and FSGM bit (SRGR2_REG[12]) are used to set the transmit frame-synchronization mode.

Table 15-16 shows how FSXM and FSGM select the source of transmit frame–synchronization pulses. The three choices are:

1. External frame–synchronization input
2. Sample rate generator frame–synchronization signal (FSG)
3. Sample rate generator frame–synchronization signal (FSG) gated by the transmit buffer XB empty condition.

Table 15-16 also shows the effect of each bit setting on the McBSP.FSX pin. The FSXP bit determines the polarity of the signal on the McBSP.FSX pin.

Table 15-16. How FSXM and FSGM Select the Source of Transmit Frame-Synchronization Pulses

FSXM	FSGM	Source of Transmit Frame Synchronization	McBSP.FSX Pin Status
0	0 or 1	An external frame–synchronization signal enters the McBSP through the McBSP.FSX pin. The signal is then inverted by FSXP before being used as internal FSX.	Input
1	1	Internal FSX is driven by the sample rate generator frame–synchronization signal (FSG).	Output. FSG is inverted by FSXP before being driven out on McBSP.FSX pin.
1	0	A XB empty condition causes the McBSP not to generate a transmit frame–synchronization pulse. The frame synchronization is generated taking into account the FWID and FPER as long as the transmit buffer is not empty. When the buffer is empty the generated frame synchronization signal is gated.	Output. The generated frame–synchronization pulse is inverted as determined by FSXP before being driven out on McBSP.FSX pin.

If the sample rate generator creates a frame–synchronization signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the McBSP.FSR pin.

In the digital loop-back mode (DLB = 1) or analog loop-back mode (ALB = 1), the transmit frame synchronization signal is used as the receive frame synchronization signal.

15.2.8.6.14 Set the Transmit Frame-Synchronization Polarity

The FSXP bit (PCR_REG[3]) determines whether frame–synchronization pulses are active high or active low on the McBSP.FSX pin.

Transmit frame–synchronization pulses can be generated internally by the sample rate generator or driven by an external source. The source of frame synchronization is selected by programming the PCR_REG[11] register FSXM mode bit. FSX is also affected by the FSGM bit (SRGR2_REG[12]). Similarly, transmit clocks can be selected to be inputs or outputs by programming the PCR_REG[9] register CLKXM mode bit.

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame–synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the McBSP.DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the McBSP.DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR_REG) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame–synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active–low frame–synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected and the polarity bit FS(R/X)P = 1, the internal active–high frame–synchronization signals are inverted before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling–edge triggered input clock on CLKX is inverted to a rising–edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising–edge triggered) clock, internal CLKX, is inverted before being sent out on the McBSP.CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal CLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising–edge triggered input clock on CLKR is inverted to a falling–edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling–edge triggered clock is inverted to a rising–edge triggered clock before being sent out on the McBSP.CLKR pin.

Note that CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge.

15.2.8.6.15 Set the SRG Frame-Synchronization Period and Pulse Width

FPER bit field (SRGR2_REG[11:0]) is used to set the SRG frame-synchronization period and FWID bit field (SRGR1_REG[15:8]) is used to set the SRG pulse width. The sample rate generator can produce a clock signal, CLKG, and a frame–synchronization signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

On FSG, the period from the start of a frame–synchronization pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame–synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

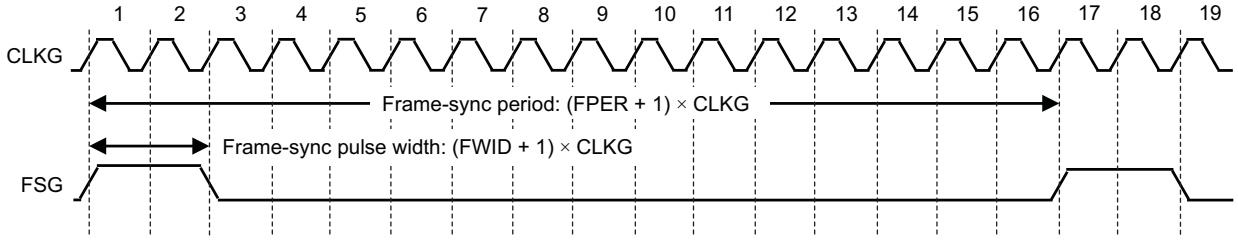
Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down–counters. The 12–bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8–bit FWID counter counts down from the programmed value (255 maximum) to 0.

Figure 15-32 shows a frame–synchronization period of 16 CLKG periods (FPER = 15 or 0000 1111b) and a frame–synchronization pulse with an active width of 2 CLKG periods (FWID = 1).

When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when GRST = 1 and FSGM = 1, a frame–synchronization pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

Figure 15-32. Frame of 16 CLKG Periods and Active Width of 2 CLKG Periods



15.2.8.6.16 Set the Transmit Clock Mode

CLKXM bit (PCR_REG[9]) is used to set the transmit clock mode

Table 15-17 shows how the CLKXM bit selects the transmit clock and the corresponding status of the McBSP.CLKX pin. The CLKXP bit determines the polarity of the signal on the McBSP.CLKX pin.

If the sample rate generator creates a clock signal (CLKG) that is derived from an external input clock, the GSYNC bit determines whether CLKG is kept synchronized with pulses on the McBSP.FSR pin.

In the digital loop-back mode (DLB = 1) or analog loop-back mode (ALB = 1), the transmit frame synchronization signal is used as the receive frame synchronization signal.

Table 15-17. CLKXM Bit Effect on Transmit Clock and McBSP.CLKX Pin

CLKXM	Source of Transmit Clock	McBSP.CLKX Pin Status
0	Internal CLKX is driven by an external clock on the McBSP.CLKX pin. CLKX is inverted as determined by CLKXP before being used.	Input
1	Internal CLKX is driven by the sample rate generator clock, CLKG.	Output. CLKG, inverted as determined by CLKXP, is driven out on McBSP.CLKX.

15.2.8.6.17 Set the Transmit Clock Polarity

CLKXP bit (PCR_REG[1]) is used to set the transmit clock polarity.

Transmit frame–synchronization pulses can be either generated internally by the sample rate generator or driven by an external source. The source of frame synchronization is selected by programming the PCR_REG[11] register FSXM mode bit. FSX is also affected by the FSGM bit (SRGR2_REG[12]). Similarly, transmit clocks can be selected to be inputs or outputs by programming the PCR_REG[9] register CLKXM mode bit.

When FSR and FSX are inputs (FSXM = FSRM = 0, external frame–synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the McBSP.DR pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the McBSP.DX pin is output on the rising edge of internal CLKX. FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR_REG) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame–synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active–low frame–synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active–high frame–synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling–edge triggered input clock on CLKX is inverted to a rising–edge triggered clock before being sent to the transmitter. If CLKXP = 1 and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising–edge triggered) clock, internal CLKX, is inverted before being sent out on the McBSP.CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal CLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising–edge triggered input clock on CLKR is inverted to a falling–edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling–edge triggered clock is inverted to a rising–edge triggered clock before being sent out on the McBSP.CLKR pin. Note that CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge.

15.2.8.6.18 Set the SRG Clock Divide-Down Value

CLKGDV bit field (SRGR1_REG[7:0]) is used to set the sample rate generator clock divide-down value.

The first divider stage generates the serial data bit clock from the input clock. This divider stage utilizes a counter, preloaded by CLKGDV, that contains the divide ratio value.

The output of the first divider stage is the data bit clock, which is output as CLKG and which serves as the input for the second and third stages of the divider.

CLKG has a frequency equal to 1/(CLKGDV + 1) of sample rate generator input clock. Thus, the sample generator input clock frequency is divided by a value between 1 and 256. The CLKG duty cycle is 50%.

15.2.8.6.19 Set the SRG Clock Synchronization Mode

GSYNC bit (SRGR2_REG[15]) is used to set the SRG clock synchronization mode.

15.2.8.6.20 Set the SRG Clock Mode (choose an input clock)

SCLKME bit (PCR_REG[7]) and CLKSM bit (SRGR2_REG[13]) are used to set the SRG clock mode.

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both, but CLKG is derived from an input clock.

15.2.8.6.21 Set the SRG Input Clock Polarity

CLKSP bit (SRGR2_REG[14]), CLKXP bit (PCR_REG[1]) and CLKRP bit (PCR_REG[0]) are used to set the SRG input clock polarity.

The sample rate generator can produce a clock signal (CLKG) and a frame–synchronization signal (FSG) for use by the receiver, the transmitter, or both. To produce CLKG and FSG, the sample rate generator must be driven by an input clock signal derived from the McBSP_FCLK clock or from an external clock on the McBSP.CLKS, McBSP.CLKX, or McBSP.CLKR pin. If you use a pin, choose a polarity for that pin by using the appropriate polarity bit (CLKSP for the McBSP.CLKS pin, CLKXP for the McBSP.CLKX pin, CLKRP for the McBSP.CLKR pin). The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

15.2.8.7 General-Purpose I/O on the McBSP Pins (legacy only)

Table 15-18 summarizes how to use the McBSP pins as general–purpose I/O pins. All of the bits mentioned in the table except XRST and RRST bits are in the pin control register (PCR_REG). XRST and RRST are in the serial port control registers (SPCR2_REG[0] and SPCR1_REG[0], respectively.

To use receiver pins McBSP.CLKR, McBSP.FSR, and McBSP.DR as general–purpose I/O pins rather than as serial port pins, you must set two conditions:

1. The receiver of the serial port is in reset (SPCR1_REG[0] RRST bit = 0).
2. General–purpose I/O is enabled for the serial port receiver (RIOEN = 1 in PCR_REG[12]).

The McBSP.CLKR and McBSP.FSR pins can be individually configured as either input or output pins with the CLKRM and FSRM bits, respectively. The McBSP.DR pin can only be an input pin. The following table shows, which bits in PCR_REG are used to read from/write to these pins.

For the transmitter pins McBSP.CLKX, McBSP.FSX, and McBSP.DX, you must meet two conditions:

1. The transmitter of the serial port is in reset (XRST = 0 in SPCR2_REG[0]).
2. General–purpose I/O is enabled for the serial port transmitter (XIOEN = 1 in PCR_REG[12]).

The McBSP.CLKX and McBSP.FSX pins can be individually configured as input or output pins with the CLKXM and FSXM bits, respectively. The McBSP.DX pin can only be an output pin.

For the McBSP.CLKS pin, all of the reset and I/O enable conditions must be met:

1. Both the receiver and transmitter of the serial port are in reset (RRST = 0 and XRST = 0).
2. General–purpose I/O is enabled for both the receiver and the transmitter (RIOEN = 1 and XIOEN = 1).

The McBSP.CLKS pin can only be an input pin. To read the status of the signal on the McBSP.CLKS pin, read the CLKS_STAT bit in PCR_REG.

Table 15-18. Using McBSP Pins for General-Purpose I/O

Pin	General–Purpose Use Enabled by This Bit Combination	Selected as Input When ...	Input Value Read From This Bit
McBSP.CLKX	XRST = 0 XIOEN = 1	CLKXM = 0	CLKXP
McBSP.FSX	XRST = 0 XIOEN = 1	FSXM = 0	FSXP
McBSP.DX	XRST = 0 XIOEN = 1	Never	Does not apply
McBSP.CLKR	RRST = 0 RIOEN = 1	CLKRM = 0	CLKRP
McBSP.FSR	RRST = 0 RIOEN = 1	FSRM = 0	FSRP
McBSP.DR	RRST = 0 RIOEN = 1	Always	DR_STAT
McBSP.CLKS	RRST = XRST = 0 RIOEN = XIOEN = 1	Always	CLKS_STAT

15.3 McBSP Registers

Table 15-19 lists the registers for the McBSP.

Table 15-19. McBSP Registers

Address Offset	Acronym	Register Name	Section
0h	REVNB	Revision Number Register	Section 15.3.1
10h	SYSCONFIG_REG	System Configuration Register	Section 15.3.2
20h	EOI	End of Interrupt Register	Section 15.3.3
24h	IRQSTATUS_RAW	Interrupt Status Raw Register	Section 15.3.4
28h	IRQSTATUS	Interrupt Status Register	Section 15.3.5
2Ch	IRQENABLE_SET	Interrupt Enable Set Register	Section 15.3.6
30h	IRQENABLE_CLR	Interrupt Enable Clear Register	Section 15.3.7
34h	DMARXENABLE_SET	DMA Rx Enable Set Register	Section 15.3.8
38h	DMATXENABLE_SET	DMA Tx Enable Set Register	Section 15.3.9
3Ch	DMARXENABLE_CLR	DMA Rx Enable Clear Register	Section 15.3.10
40h	DMATXENABLE_CLR	DMA Tx Enable Clear Register	Section 15.3.11
48h	DMARXWAKE_EN	DMA Rx Wake Enable Register	Section 15.3.12
4Ch	DMATXWAKE_EN	DMA Tx Wake Enable Register	Section 15.3.13
B4h	XBUFFSTAT_REG	McBSP Transmit Buffer Status Register	Section 15.3.50
100h	DRR_REG	McBSP Data Receive Register	Section 15.3.14
108h	DXR_REG	McBSP Data Transmit Register	Section 15.3.15
110h	SPCR2_REG	McBSP Serial Port Control Register 2	Section 15.3.16
114h	SPCR1_REG	McBSP Serial Port Control Register 1	Section 15.3.17
118h	RCR2_REG	McBSP Receive Control Register 2	Section 15.3.18
11Ch	RCR1_REG	McBSP Receive Control Register 1	Section 15.3.19
120h	XCR2_REG	McBSP Transmit Control Register 2	Section 15.3.20
124h	XCR1_REG	McBSP Transmit Control Register 1	Section 15.3.21
128h	SRGR2_REG	McBSP Sample Rate Generator Register 2	Section 15.3.22
12Ch	SRGR1_REG	McBSP Sample Rate Generator Register 1	Section 15.3.23
130h	MCR2_REG	McBSP Multichannel Register 2	Section 15.3.24
134h	MCR1_REG	McBSP Multichannel Register 1	Section 15.3.25
138h	RCERA_REG	McBSP Receive Channel Enable Register Partition A	Section 15.3.26
13Ch	RCERB_REG	McBSP Receive Channel Enable Register Partition B	Section 15.3.27
140h	XCERA_REG	McBSP Transmit Channel Enable Register Partition A	Section 15.3.28
144h	XCERB_REG	McBSP Transmit Channel Enable Register Partition B	Section 15.3.29
148h	PCR_REG	McBSP Pin Control Register	Section 15.3.30
14Ch	RCERC_REG	McBSP Receive Channel Enable Register Partition C	Section 15.3.31
150h	RCERD_REG	McBSP Receive Channel Enable Register Partition D	Section 15.3.32
154h	XCERC_REG	McBSP Transmit Channel Enable Register Partition C	Section 15.3.33
158h	XCERD_REG	McBSP Transmit Channel Enable Register Partition D	Section 15.3.34
15Ch	RCERE_REG	McBSP Receive Channel Enable Register Partition E	Section 15.3.35
160h	RCERF_REG	McBSP Receive Channel Enable Register Partition F	Section 15.3.36
164h	XCERE_REG	McBSP Transmit Channel Enable Register Partition E	Section 15.3.37
168h	XCERF_REG	McBSP Transmit Channel Enable Register Partition F	Section 15.3.38
16Ch	RCERG_REG	McBSP Receive Channel Enable Register Partition G	Section 15.3.39
170h	RCERH_REG	McBSP Receive Channel Enable Register Partition H	Section 15.3.40
174h	XCERG_REG	McBSP Transmit Channel Enable Register Partition G	Section 15.3.41
178h	XCERH_REG	McBSP Transmit Channel Enable Register Partition H	Section 15.3.42

Table 15-19. McBSP Registers (continued)

Address Offset	Acronym	Register Name	Section
190h	THRSH2_REG	McBSP Transmit Buffer Threshold Register (DMA or IRQ Trigger)	Section 15.3.43
194h	THRSH1_REG	McBSP Receive Buffer Threshold Register (DMA or IRQ Trigger)	Section 15.3.44
1A0h	IRQSTATATUS	McBSP Interrupt Status Register (OCP Compliant IRQ Line)	Section 15.3.45
1A4h	IRQENABLE	McBSP Interrupt Enable Register (OCP Compliant IRQ Line)	Section 15.3.46
1A8h	WAKEUPEN	McBSP Wakeup Enable Register	Section 15.3.47
1ACh	XCCR_REG	McBSP Transmit Configuration Control Register	Section 15.3.48
1B0h	RCCR_REG	McBSP Receive Configuration Control Register	Section 15.3.49
1B8h	RBUFFSTAT_REG	McBSP Receive Buffer Status Register	Section 15.3.51
1C0h	STATUS_REG	McBSP Status Register	Section 15.3.52

15.3.1 Revision Number Register (REVNB)

The Revision Number (REVNB) register is shown in Figure 15-33 and described in Table 15-20.

This read-only register contains the hard-coded revision number of the module. A write to this register has no effect.

Figure 15-33. Revision Number Register (REVNB)

31	30	29	28	27	16
SCHEME		Reserved		FUNC	
R-1		R-1		R-8h	

15	11	10	8	7	6	5	0
RTL		MAJOR		CUSTOM		MINOR	
R-0		R-0		R-0		R-37h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-20. Revision Number Register (REVNB) Field Descriptions

Bits	Field Name	Value	Description
31-30	SCHEME	1h	Used to distinguish between old Scheme and current. Spare bit to encode future schemes.
29-28	Reserved	0	Reads return 0x1.
27-16	FUNC	8h	Function: Indicates a software compatible module family.
15-11	RTL	0h	RTL version.
10-8	MAJOR	0h	Major Revision.
7-6	CUSTOM	0h	Custom ID.
5-0	MINOR	37h	Minor Revision.

15.3.2 System Configuration Register (SYSCONFIG_REG)

The System Configuration (SYSCONFIG_REG) register is shown in Figure 15-34 and described in Table 15-21.

Figure 15-34. System Configuration Register (SYSCONFIG_REG)

31	16
Reserved	
R-0	

15	10	9	8
Reserved		CLOCKACTIVITY	
		R/W-0	

7	5	4	3	2	1	0
Reserved		SIDLEMODE		ENAWAKEUP	SOFTRESET	Reserved
R-0		R/W-0		R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-21. System Configuration Register (SYSCONFIG_REG) Field Descriptions

Bits	Field Name	Value	Description
31-10	Reserved	0	Reserved.
9-8	CLOCKACTIVITY	Bit 9	OCP interface clock.
		0	OCP clock can be switched-off.
		1	OCP clock is maintained during wake up period.
		Bit 8	Functional clock.
		0	Functional clock can be switched-off.
		1	Functional clock is maintained during wake up period.
		0	OCP clock can be switched-off. Functional clock can be switched-off.
		1h	OCP clock can be switched-off. Functional clock is maintained during wake up period.
		2h	OCP clock is maintained during wake up period. Functional clock can be switched-off.
		3h	OCP clock is maintained during wake up period. Functional clock is maintained during wake up period.
7-5	Reserved	0	Reserved.
4-3	SIDLEMODE		Slave interface power management, req/ack control
		0	Force-idle. An idle request is acknowledged unconditionally.
		1h	No-idle. An idle request is never acknowledged.
		2h	Smart-idle. Acknowledgment to an idle request is given based on the internal activity of the module.
		3h	Smart Idle Wakeup.
2	ENAWAKEUP		WakeUp feature control.
		0	WakeUp is disabled.
		1	WakeUp capability is enabled.
1	SOFTRESET		McBSPLP global software reset.
		0	Read: Reset done, no pending action.
			Write: No action.
		1	Read: Reset (software or other) ongoing.
			Write: Initiate software reset.
0	Reserved	0	Reserved.

15.3.3 End of Interrupt Register (EOI)

The End of Interrupt (EOI) register is shown in Figure 15-35 and described in Table 15-22.

Figure 15-35. End of Interrupt Register (EOI)

31	1	0
Reserved		LINE_NUMBER
R-0		W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-22. End of Interrupt Register (EOI) Field Descriptions

Bits	Field Name	Value	Description
31-1	Reserved	0	Reserved.
0	LINE_NUMBER		Software End Of Interrupt (EOI) control. Write number of interrupt output. Write a value of 0 to force the output interrupt to re-assert if any enabled interrupts are pending.

15.3.4 Interrupt Status Raw Register (IRQSTATUS_RAW)

This register provides core status information for interrupt handling, showing all active events (enabled and not enabled ones). The fields are read-write. Writing a 1 to a bit will set it to 1, for example, trigger the IRQ when also enabled (mostly for debug). Writing a 0 will have no effect, for example, the register value will not be modified. Only enabled, active events will trigger an actual interrupt request on the IRQ output line.

Interrupts can either be serviced using the highlander standard IRQSTATUS, IRQSTATUS_RAW, IRQENABLE_SET, and IRQENABLE_CLR set of registers, or the legacy IRQSTATUS_REG and IRQENABLE_REG registers, but not both in the same system.

The Interrupt Status Raw (IRQSTATUS_RAW) register is shown in Figure 15-36 and described in Table 15-23.

Figure 15-36. Interrupt Status Raw Register (IRQSTATUS_RAW)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved	XEMPTYEOF	Reserved	XOVFLSTAT	XUNDFLSTAT	XRDY	XEOF	XFSX
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
XSYNCERR	Reserved	ROVFLSTAT	RUNDFLSTAT	RRDY	REOF	RFSR	RSYNCERR
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-23. Interrupt Status Raw Register (IRQSTATUS_RAW) Field Descriptions

Bits	Field Name	Value	Description
31-15	Reserved	0	Reserved.
14	XEMPTYEOF		Transmit Buffer Empty at end of frame (XEMPTYEOF is set to 1 when a complete frame was transmitted and the transmit buffer is empty). Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	XEMPTYEOF is NOT set to when a complete frame was transmitted and the transmit buffer is empty. Writing a 0 will have no effect.
		1	XEMPTYEOF is set to 1 when a complete frame was transmitted and the transmit buffer is empty. Writing 1 to this bit will set it to 1.
13	Reserved	0	Reserved.
12	XOVFLSTAT		Transmit Buffer Overflow (XOVFLSTAT bit is set to 1 when transmit buffer overflow; the data which is written while overflow condition is discarded). Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	XEMPTYEOF is NOT set to when a complete frame was transmitted and the transmit buffer is empty . Writing a 0 will have no effect.
		1	XEMPTYEOF is set to 1 when a complete frame was transmitted and the transmit buffer is empty. Writing 1 to this bit will set it to 1.
11	XUNDFLSTAT		Transmit Buffer Underflow (XUNDFLSTAT bit is set to 1 when the transmit data buffer is empty new data is required to be transmitted). Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	The transmit data buffer is NOT empty new data is required to be transmitted. Writing a 0 will have no effect.
		1	The transmit data buffer is empty new data is required to be transmitted. Writing 1 to this bit will set it to 1.
10	XRDY		Transmit Buffer Threshold Reached (XRDY bit is set to 1 when the transmit buffer free locations are equal or above the THRSH2_REG value). Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	Transmit buffer occupied locations are below the THRSH2_REG value). Writing a 0 will have no effect.
		1	Transmit buffer occupied locations are equal or above the THRSH2_REG value). Writing 1 to this bit will set it to 1.
9	XEOF		Transmit End Of Frame (XEOF is set to 1 when a complete frame was transmitted). Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	Complete frame was NOT transmitted. Writing a 0 will have no effect.
		1	Complete frame was transmitted. Writing 1 to this bit will set it to 1.
8	XFSX		Transmit Frame Synchronization (XFSX bit is set to 1 when a new transmit frame synchronization is asserted). Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	New transmit frame synchronization is NOT asserted. Writing a 0 will have no effect.
		1	New transmit frame synchronization is asserted. Writing 1 to this bit will set it to 1.
7	XSYNCERR		Transmit Frame Synchronization Error (XSYNCERR is set to 1 when a transmit frame synchronization error is detected). Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	Transmit frame synchronization error is NOT detected. Writing a 0 will have no effect.
		1	Transmit frame synchronization error is detected. Writing 1 to this bit will set it to 1.
6	Reserved	0	Reserved.
5	ROVFLSTAT		Receive Buffer Overflow (ROVFLSTAT bit is set to 1 when receive buffer overflow; the data which is written while overflow condition is discarded). Writing 1 to this bit will set it to 1. Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	Receive buffer NOT overflow . Writing a 0 will have no effect.
		1	Receive buffer overflow; Writing 1 to this bit will set it to 1.
4	RUNDFLSTAT		Receive Buffer Underflow (RUNDFLSTAT bit is set to 1 when read operation is performed to the receive data register while receive buffer is empty; data read while underflow condition is undefined). Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	Read operation is performed to the receive data register while receive buffer is NOT empty. Writing a 0 will have no effect.
		1	Read operation is performed to the receive data register while receive buffer is empty; Writing 1 to this bit will set it to 1.

Table 15-23. Interrupt Status Raw Register (IRQSTATUS_RAW) Field Descriptions (continued)

Bits	Field Name	Value	Description
3	RRDY		Receive Buffer Threshold Reached (RRDY bit is set to 1 when the receive buffer occupied locations are equal or above the THRSH1_REG value). Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	Receive buffer occupied locations are below the THRSH1_REG value). Writing a 0 will have no effect.
		1	Receive buffer occupied locations are equal or above the THRSH1_REG value). Writing 1 to this bit will set it to 1.
2	REOF		Receive End Of Frame (REOF is set to 1 when a complete frame was received). Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	Complete frame was NOT received. Writing a 0 will have no effect.
		1	Complete frame was received; Writing 1 to this bit will set it to 1.
1	RFSR		Receive Frame Synchronization (RFSR bit is set to 1 when a new receive frame synchronization is asserted). Wr Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	New receive frame synchronization is NOT asserted. Writing a 0 will have no effect.
		1	New receive frame synchronization is asserted; Writing 1 to this bit will set it to 1.
0	RSYNCERR		Receive Frame Synchronization Error (RSYNCERR is set to 1 when a receive frame synchronization error is detected). Writing 1 to this bit will set it to 1. Writing a 0 will have no effect.
		0	Receive frame synchronization error is NOT detected. Writing a 0 will have no effect.
		1	Receive frame synchronization error is detected. Writing 1 to this bit will set it to 1.

15.3.5 Interrupt Status Register (IRQSTATUS)

This register provides core status information for interrupt handling, showing all active and enabled events and masking the others. The fields are read-write. Writing a 1 to a bit will clear it to 0, for example, clear the IRQ. Writing a 0 will have no effect, for example, the register value will not be modified. Only enabled, active events will trigger an actual interrupt request on the IRQ output line.

Interrupts can either be serviced using the highlander standard IRQSTATUS, IRQSTATUS_RAW, IRQENABLE_SET, and IRQENABLE_CLR set of registers, or the legacy IRQSTATUS_REG and IRQENABLE_REG registers, but not both in the same system.

The Interrupt Status (IRQSTATUS) register is shown in Figure 15-37 and described in Table 15-24.

Figure 15-37. Interrupt Status Register (IRQSTATUS)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	XEMPTYEOF	Reserved	XOVFLSTAT	XUNDFLSTAT	XRDY	XEOF	XFSX
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
XSYNCERR	Reserved	ROVFLSTAT	RUNDFLSTAT	RRDY	REOF	RFSR	RSYNCERR
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-24. Interrupt Status Register (IRQSTATUS) Field Descriptions

Bits	Field Name	Value	Description
31-15	Reserved	0	Reserved.
14	XEMPTYEOF		Transmit Buffer Empty at end of frame (XEMPTYEOF is set to 1 when a complete frame was transmitted and the transmit buffer is empty).
		0	XEMPTYEOF is NOT set to when a complete frame was transmitted and the transmit buffer is empty.
		1	XEMPTYEOF is set to 1 when a complete frame was transmitted and the transmit buffer is empty. Writing 1 to this bit clears the bit.
13	Reserved	0	Reserved.
12	XOVFLSTAT		Transmit Buffer Overflow (XOVFLSTAT bit is set to 1 when transmit buffer overflow; the data which is written while overflow condition is discarded). Writing 1 to this bit clears the bit.
		0	Transmit buffer NOT overflow.
		1	Transmit buffer overflow; Writing 1 to this bit clears the bit.
11	XUNDFLSTAT		Transmit Buffer Underflow (XUNDFLSTAT bit is set to 1 when the transmit data buffer is empty new data is required to be transmitted). Writing 1 to this bit clears the bit.
		0	The transmit data buffer is NOT empty new data is required to be transmitted.
		1	The transmit data buffer is empty new data is required to be transmitted. Writing 1 to this bit clears the bit.
10	XRDY		Transmit Buffer Threshold Reached (XRDY bit is set to 1 when the transmit buffer free locations are equal or above the THRSH2_REG value). Writing 1 to this bit clears the bit.
		0	Transmit buffer occupied locations are below the THRSH2_REG value).
		1	Transmit buffer occupied locations are equal or above the THRSH2_REG value). Writing 1 to this bit clears the bit.
9	XEOF		Transmit End Of Frame (XEOF is set to 1 when a complete frame was transmitted). Writing 1 to this bit clears the bit.
		0	Complete frame was NOT transmitted.
		1	Complete frame was transmitted; Writing 1 to this bit clears the bit.

Table 15-24. Interrupt Status Register (IRQSTATUS) Field Descriptions (continued)

Bits	Field Name	Value	Description
8	XFSX		Transmit Frame Synchronization (XFSX bit is set to 1 when a new transmit frame synchronization is asserted). Writing 1 to this bit clears the bit.
		0	New transmit frame synchronization is NOT asserted.
		1	New transmit frame synchronization is asserted; Writing 1 to this bit clears the bit.
7	XSYNCERR		Transmit Frame Synchronization Error (XSYNCERR is set to 1 when a transmit frame synchronization error is detected). Writing 1 to this bit clears the bit.
		0	Transmit frame synchronization error is NOT detected.
		1	Transmit frame synchronization error is detected. Writing 1 to this bit clears the bit.
6	Reserved	0	Reserved.
5	ROVFLSTAT		Receive Buffer Overflow (ROVFLSTAT bit is set to 1 when receive buffer overflow; the data which is written while overflow condition is discarded). Writing 1 to this bit clears the bit.
		0	Receive buffer NOT overflow.
		1	Receive buffer overflow; Writing 1 to this bit clears the bit.
4	RUNDFLSTAT		Receive Buffer Underflow (RUNDFLSTAT bit is set to 1 when read operation is performed to the receive data register while receive buffer is empty; data read while underflow condition is undefined). Writing 1 to this bit clears the bit.
		0	Read operation is performed to the receive data register while receive buffer is NOT empty.
		1	Read operation is performed to the receive data register while receive buffer is empty; Writing 1 to this bit clears the bit.
3	RRDY		Receive Buffer Threshold Reached (RRDY bit is set to 1 when the receive buffer occupied locations are equal or above the THRSH1_REG value). Writing 1 to this bit clears the bit.
		0	Receive buffer occupied locations are below the THRSH1_REG value).
		1	Receive buffer occupied locations are equal or above the THRSH1_REG value). Writing 1 to this bit clears the bit.
2	REOF		Receive End Of Frame (REOF is set to 1 when a complete frame was received). Writing 1 to this bit clears the bit.
		0	Complete frame was NOT received.
		1	Complete frame was received; Writing 1 to this bit clears the bit.
1	RFSR		Receive Frame Synchronization (RFSR bit is set to 1 when a new receive frame synchronization is asserted). Writing 1 to this bit clears the bit.
		0	New receive frame synchronization is NOT asserted.
		1	New receive frame synchronization is asserted; Writing 1 to this bit clears the bit.
0	RSYNCERR		Receive Frame Synchronization Error (RSYNCERR is set to 1 when a receive frame synchronization error is detected). Writing 1 to this bit clears the bit.
		0	Receive frame synchronization error is NOT detected.
		1	Receive frame synchronization error is detected. Writing 1 to this bit clears the bit.

15.3.6 Interrupt Enable Set Register (IRQENABLE_SET)

All 1-bit fields enable a specific interrupt event to trigger an interrupt request. Writing a 1 to a bit will enable the field. Writing a 0 will have no effect, for example, the register value will not be modified.

Interrupts can either be serviced using the highlander standard IRQSTATUS, IRQSTATUS_RAW, IRQENABLE_SET, and IRQENABLE_CLR set of registers, or the legacy IRQSTATUS_REG and IRQENABLE_REG registers, but not both in the same system.

The Interrupt Enable Set (IRQENABLE_SET) register is shown in Figure 15-38 and described in Table 15-25.

Figure 15-38. Interrupt Enable Set Register (IRQENABLE_SET)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved	XEMPTYEOF	Reserved	XOVFLSTAT	XUNDFLSTAT	XRDY	XEOF	XFSX
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
XSYNCERR	Reserved	ROVFLSTAT	RUNDFLSTAT	RRDY	REOF	RFSR	RSYNCERR
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 15-25. Interrupt Enable Set Register (IRQENABLE_SET) Field Descriptions

Bits	Field Name	Value	Description
31-15	Reserved	0	Reserved.
14	XEMPTYEOF		Transmit Buffer Empty at end of frame (XEMPTYEOF is set to 1 when a complete frame was transmitted and the transmit buffer is empty).
		0	XEMPTYEOF is NOT set to when a complete frame was transmitted and the transmit buffer is empty.
		1	XEMPTYEOF is set to 1 when a complete frame was transmitted and the transmit buffer is empty. Writing 1 to this bit clears the bit.
13	Reserved	0	Reserved.
12	XOVFLSTAT		Transmit Buffer Overflow (XOVFLSTAT bit is set to 1 when transmit buffer overflow; the data which is written while overflow condition is discarded). Writing 1 to this bit clears the bit.
		0	Transmit buffer NOT overflow.
		1	Transmit buffer overflow; Writing 1 to this bit clears the bit.
11	XUNDFLSTAT		Transmit Buffer Underflow (XUNDFLSTAT bit is set to 1 when the transmit data buffer is empty new data is required to be transmitted). Writing 1 to this bit clears the bit.
		0	The transmit data buffer is NOT empty new data is required to be transmitted.
		1	The transmit data buffer is empty new data is required to be transmitted. Writing 1 to this bit clears the bit.
10	XRDY		Transmit Buffer Threshold Reached (XRDY bit is set to 1 when the transmit buffer free locations are equal or above the THRSH2_REG value). Writing 1 to this bit clears the bit.
		0	Transmit buffer occupied locations are below the THRSH2_REG value).
		1	Transmit buffer occupied locations are equal or above the THRSH2_REG value). Writing 1 to this bit clears the bit.
9	XEOF		Transmit End Of Frame (XEOF is set to 1 when a complete frame was transmitted). Writing 1 to this bit clears the bit.
		0	Complete frame was NOT transmitted.
		1	Complete frame was transmitted; Writing 1 to this bit clears the bit.
8	XFSX		Transmit Frame Synchronization (XFSX bit is set to 1 when a new transmit frame synchronization is asserted). Writing 1 to this bit clears the bit.
		0	New transmit frame synchronization is NOT asserted.
		1	New transmit frame synchronization is asserted; Writing 1 to this bit clears the bit.

Table 15-25. Interrupt Enable Set Register (IRQENABLE_SET) Field Descriptions (continued)

Bits	Field Name	Value	Description
7	XSYNCERR		Transmit Frame Synchronization Error (XSYNCERR is set to 1 when a transmit frame synchronization error is detected). Writing 1 to this bit clears the bit.
		0	Transmit frame synchronization error is NOT detected.
		1	Transmit frame synchronization error is detected. Writing 1 to this bit clears the bit.
6	Reserved	0	Reserved.
5	ROVFLSTAT		Receive Buffer Overflow (ROVFLSTAT bit is set to 1 when receive buffer overflow; the data which is written while overflow condition is discarded). Writing 1 to this bit clears the bit.
		0	Receive buffer NOT overflow.
		1	Receive buffer overflow; Writing 1 to this bit clears the bit.
4	RUNDFLSTAT		Receive Buffer Underflow (RUNDFLSTAT bit is set to 1 when read operation is performed to the receive data register while receive buffer is empty; data read while underflow condition is undefined). Writing 1 to this bit clears the bit.
		0	Read operation is performed to the receive data register while receive buffer is NOT empty.
		1	Read operation is performed to the receive data register while receive buffer is empty; Writing 1 to this bit clears the bit.
3	RRDY		Receive Buffer Threshold Reached (RRDY bit is set to 1 when the receive buffer occupied locations are equal or above the THRSH1_REG value). Writing 1 to this bit clears the bit.
		0	Receive buffer occupied locations are below the THRSH1_REG value).
		1	Receive buffer occupied locations are equal or above the THRSH1_REG value). Writing 1 to this bit clears the bit.
2	REOF		Receive End Of Frame (REOF is set to 1 when a complete frame was received). Writing 1 to this bit clears the bit.
		0	Complete frame was NOT received.
		1	Complete frame was received; Writing 1 to this bit clears the bit.
1	RFSR		Receive Frame Synchronization (RFSR bit is set to 1 when a new receive frame synchronization is asserted). Writing 1 to this bit clears the bit.
		0	New receive frame synchronization is NOT asserted.
		1	New receive frame synchronization is asserted; Writing 1 to this bit clears the bit.
0	RSYNCERR		Receive Frame Synchronization Error (RSYNCERR is set to 1 when a receive frame synchronization error is detected). Writing 1 to this bit clears the bit.
		0	Receive frame synchronization error is NOT detected.
		1	Receive frame synchronization error is detected. Writing 1 to this bit clears the bit.

15.3.7 Interrupt Enable Clear Register (IRQENABLE_CLR)

All 1-bit fields clear a specific interrupt event. Writing a 1 to a bit will disable the interrupt field. Writing a 0 will have no effect, for example, the register value will not be modified.

Interrupts can either be serviced using the highlander standard IRQSTATUS, IRQSTATUS_RAW, IRQENABLE_SET, and IRQENABLE_CLR set of registers, or the legacy IRQSTATUS_REG and IRQENABLE_REG registers, but not both in the same system.

The Interrupt Enable Clear (IRQENABLE_CLR) register is shown in Figure 15-39 and described in Table 15-26.

Figure 15-39. Interrupt Enable Clear Register (IRQENABLE_CLR)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	XEMPTYEOF	Reserved	XOVFLSTAT	XUNDFLSTAT	XRDY	XEOF	XFSX
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
XSYNCERR	Reserved	ROVFLSTAT	RUNDFLSTAT	RRDY	REOF	RFSR	RSYNCERR
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-26. Interrupt Enable Clear Register (IRQENABLE_CLR) Field Descriptions

Bits	Field Name	Value	Description
31-15	Reserved	0	Reserved.
14	XEMPTYEOF		Transmit Buffer Empty at end of frame (XEMPTYEOF is set to 1 when a complete frame was transmitted and the transmit buffer is empty).
		0	XEMPTYEOF is NOT set to when a complete frame was transmitted and the transmit buffer is empty.
		1	XEMPTYEOF is set to 1 when a complete frame was transmitted and the transmit buffer is empty. Writing 1 to this bit clears the bit.
13	Reserved	0	Reserved.
12	XOVFLSTAT		Transmit Buffer Overflow (XOVFLSTAT bit is set to 1 when transmit buffer overflow; the data which is written while overflow condition is discarded). Writing 1 to this bit clears the bit.
		0	Transmit buffer NOT overflow.
		1	Transmit buffer overflow; Writing 1 to this bit clears the bit.
11	XUNDFLSTAT		Transmit Buffer Underflow (XUNDFLSTAT bit is set to 1 when the transmit data buffer is empty new data is required to be transmitted). Writing 1 to this bit clears the bit.
		0	The transmit data buffer is NOT empty new data is required to be transmitted.
		1	The transmit data buffer is empty new data is required to be transmitted. Writing 1 to this bit clears the bit.
10	XRDY		Transmit Buffer Threshold Reached (XRDY bit is set to 1 when the transmit buffer free locations are equal or above the THRSH2_REG value). Writing 1 to this bit clears the bit.
		0	Transmit buffer occupied locations are below the THRSH2_REG value).
		1	Transmit buffer occupied locations are equal or above the THRSH2_REG value). Writing 1 to this bit clears the bit.
9	XEOF		Transmit End Of Frame (XEOF is set to 1 when a complete frame was transmitted). Writing 1 to this bit clears the bit.
		0	Complete frame was NOT transmitted.
		1	Complete frame was transmitted; Writing 1 to this bit clears the bit.

Table 15-26. Interrupt Enable Clear Register (IRQENABLE_CLR) Field Descriptions (continued)

Bits	Field Name	Value	Description
8	XFSX		Transmit Frame Synchronization (XFSX bit is set to 1 when a new transmit frame synchronization is asserted). Writing 1 to this bit clears the bit.
		0	New transmit frame synchronization is NOT asserted.
		1	New transmit frame synchronization is asserted; Writing 1 to this bit clears the bit.
7	XSYNCERR		Transmit Frame Synchronization Error (XSYNCERR is set to 1 when a transmit frame synchronization error is detected). Writing 1 to this bit clears the bit.
		0	Transmit frame synchronization error is NOT detected.
		1	Transmit frame synchronization error is detected. Writing 1 to this bit clears the bit.
6	Reserved	0	Reserved.
5	ROVFLSTAT		Receive Buffer Overflow (ROVFLSTAT bit is set to 1 when receive buffer overflow; the data which is written while overflow condition is discarded). Writing 1 to this bit clears the bit.
		0	Receive buffer NOT overflow.
		1	Receive buffer overflow; Writing 1 to this bit clears the bit.
4	RUNDFLSTAT		Receive Buffer Underflow (RUNDFLSTAT bit is set to 1 when read operation is performed to the receive data register while receive buffer is empty; data read while underflow condition is undefined). Writing 1 to this bit clears the bit.
		0	Read operation is performed to the receive data register while receive buffer is NOT empty.
		1	Read operation is performed to the receive data register while receive buffer is empty; Writing 1 to this bit clears the bit.
3	RRDY		Receive Buffer Threshold Reached (RRDY bit is set to 1 when the receive buffer occupied locations are equal or above the THRSH1_REG value). Writing 1 to this bit clears the bit.
		0	Receive buffer occupied locations are below the THRSH1_REG value).
		1	Receive buffer occupied locations are equal or above the THRSH1_REG value). Writing 1 to this bit clears the bit.
2	REOF		Receive End Of Frame (REOF is set to 1 when a complete frame was received). Writing 1 to this bit clears the bit.
		0	Complete frame was NOT received.
		1	Complete frame was received; Writing 1 to this bit clears the bit.
1	RFSR		Receive Frame Synchronization (RFSR bit is set to 1 when a new receive frame synchronization is asserted). Writing 1 to this bit clears the bit.
		0	New receive frame synchronization is NOT asserted.
		1	New receive frame synchronization is asserted; Writing 1 to this bit clears the bit.
0	RSYNCERR		Receive Frame Synchronization Error (RSYNCERR is set to 1 when a receive frame synchronization error is detected). Writing 1 to this bit clears the bit.
		0	Receive frame synchronization error is NOT detected.
		1	Receive frame synchronization error is detected. Writing 1 to this bit clears the bit.

15.3.8 DMA Rx Enable Set Register (DMARXENABLE_SET)

The 1-bit field enables a Receive DMA request. Writing a 1 to this field will set it to 1. Writing a 0 will have no effect, for example, the register value is not modified. Note that the RCCR_REG.RDMAEN field is the global (slave) DMA enabler, and that it is disabled by default. The field RCCR_REG.RDMAEN should also be set to 1 to enable a Receive DMA Request.

The DMA Rx Enable Set (DMARXENABLE_SET) register is shown in Figure 15-40 and described in Table 15-27.

Figure 15-40. DMA Rx Enable Set Register (DMARXENABLE_SET)

31	1	0
Reserved		DMARX_ENABLE_SET
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-27. DMA Rx Enable Set Register (DMARXENABLE_SET) Field Descriptions

Bits	Field Name	Value	Description
31-1	Reserved	0	Reserved.
0	DMARX_ENABLE_SET		Receive DMA channel enable set.

15.3.9 DMA Tx Enable Set Register (DMATXENABLE_SET)

The 1-bit field enables a Transmit DMA request. Writing a 1 to this field will set it to 1. Writing a 0 will have no effect, for example, the register value is not modified. Note that the XCCR_REG.XDMAEN field is the global (slave) DMA enabler, and that it is disabled by default. The field XCCR_REG.XDMAEN should also be set to 1 to enable a Transmit DMA Request.

The DMA Tx Enable Set (DMATXENABLE_SET) register is shown in Figure 15-41 and described in Table 15-28.

Figure 15-41. DMA Tx Enable Set Register (DMATXENABLE_SET)

31	1	0
Reserved		DMATX_ENABLE_SET
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-28. DMA Tx Enable Set Register (DMATXENABLE_SET) Field Descriptions

Bits	Field Name	Value	Description
31-1	Reserved	0	Reserved.
0	DMATX_ENABLE_SET		Transmit DMA channel enable set.

15.3.10 DMA Rx Enable Clear Register (DMARXENABLE_CLR)

The 1-bit field disables a Receive DMA request. Writing a 1 to a bit will clear it to 0. Another result of setting to '1' the DMARX_ENABLE_CLEAR field, is the reset of the DMA RX request and wakeup lines. Writing a 0 will have no effect, for example, the register value is not modified.

The DMA Rx Enable Clear (DMARXENABLE_CLR) register is shown in Figure 15-42 and described in Table 15-29.

Figure 15-42. DMA Rx Enable Clear Register (DMARXENABLE_CLR)

31	1	0
Reserved		DMARX_ENABLE_CLEAR
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-29. DMA Rx Enable Clear Register (DMARXENABLE_CLR) Field Descriptions

Bits	Field Name	Value	Description
31-1	Reserved	0	Reserved.
0	DMARX_ENABLE_CLEAR		Receive DMA channel enable clear.

15.3.11 DMA Tx Enable Clear Register (DMATXENABLE_CLR)

The 1-bit field disables a Transmit DMA request. Writing a 1 to a bit will clear it to 0. Another result of setting to '1' the DMATX_ENABLE_CLEAR field, is the reset of the DMA TX request and wakeup lines. Writing a 0 will have no effect, for example, the register value is not modified.

The DMA Tx Enable Clear (DMATXENABLE_CLR) register is shown in Figure 15-43 and described in Table 15-30.

Figure 15-43. DMA Tx Enable Clear Register (DMATXENABLE_CLR)

31	1	0
Reserved		DMATX_ENABLE_CLEAR
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-30. DMA Tx Enable Clear Register (DMATXENABLE_CLR) Field Descriptions

Bits	Field Name	Value	Description
31-1	Reserved	0	Reserved.
0	DMATX_ENABLE_CLEAR		Transmit DMA channel enable clear.

15.3.12 DMA Rx Wake Enable Register (DMARXWAKE_EN)

All 1-bit fields listed above enable a specific (synchronous) DMA request source to generate an asynchronous wakeup (on the appropriate wakeup line). Note that the SYSCONFIG_REG.ENAWAKEUP field is the global (slave) wakeup enabler, and that it is disabled by default.

The DMA Rx Wake Enable (DMARXWAKE_EN) register is shown in Figure 15-44 and described in Table 15-31.

Figure 15-44. DMA Rx Wake Enable Register (DMARXWAKE_EN)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved	XEMPTYEOFEN	Reserved			XRDYEN	XEOFEN	XFSXEN
R-0	R/W-0	R-0			R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
XSYNCERREN	Reserved			RRDYEN	REOFEN	RFSREN	RSYNCERREN
R/W-0	R-0			R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-31. DMA Rx Wake Enable Register (DMARXWAKE_EN) Field Descriptions

Bits	Field Name	Value	Description
31-15	Reserved	0	Reserved.
14	XEMPTYEOFEN		Transmit Buffer Empty at End Of Frame enable bit.
		0	Transmit Buffer Empty at End Of Frame WK enable is NOT active.
		1	Transmit Buffer Empty at End Of Frame WK enable is active.
13-11	Reserved	0	Reserved.
10	XRDYEN		Transmit Buffer Threshold Reached WK enable bit.
		0	Transmit Buffer Threshold WK enable is NOT active.
		1	Transmit Buffer Threshold WK enable is active.
9	XEOFEN		Transmit End Of Frame WK enable bit.
		0	Transmit End Of Frame WK enable is NOT active.
		1	Transmit End Of Frame WK enable is active.
8	XFSXEN		Transmit Frame Synchronization WK enable bit.
		0	Transmit Frame Synchronization WK enable is NOT active.
		1	Transmit Frame Synchronization WK enable is active.
7	XSYNCERREN		Transmit Frame Synchronization Error WK enable bit.
		0	Transmit Frame Synchronization Error WK enable is NOT active.
		1	Transmit Frame Synchronization Error WK enable is active.
6-4	Reserved	0	Reserved.
3	RRDYEN		Receive Buffer Threshold wakeup enable bit.
		0	Receive Buffer Threshold WK enable is NOT active.
		1	Receive Buffer Threshold WK enable is active.
2	REOFEN		Receive End Of Frame WK enable bit.
		0	Receive End Of Frame WK enable is NOT active.
		1	Receive End Of Frame WK enable is active
1	RFSREN		Receive Frame Synchronization WK enable bit.
		0	Receive Frame Synchronization WK enable is NOT active.
		1	Receive Frame Synchronization WK enable is active.

Table 15-31. DMA Rx Wake Enable Register (DMARXWAKE_EN) Field Descriptions (continued)

Bits	Field Name	Value	Description
0	RSYNCERREN		Receive Frame Synchronization Error WK enable bit.
		0	Receive Frame Synchronization Error WK enable is NOT active.
		1	Receive Frame Synchronization Error WK enable is active.

15.3.13 DMA Tx Wake Enable Register (DMATXWAKE_EN)

All 1-bit fields listed above enable a specific (synchronous) DMA request source to generate an asynchronous wakeup (on the appropriate wakeup line). Note that the SYSCONFIG_REG.ENAWAKEUP field is the global (slave) wakeup enabler, and that it is disabled by default.

The DMA Tx Wake Enable (DMATXWAKE_EN) register is shown in Figure 15-45 and described in Table 15-32.

Figure 15-45. DMA Tx Wake Enable Register (DMATXWAKE_EN)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved	XEMPTYEOFEN	Reserved			XRDYEN	XEOFEN	XFSXEN
R-0	R/W-0	R-0			R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
XSYNCERREN	Reserved			RRDYEN	REOFEN	RFSREN	RSYNCERREN
R/W-0	R-0			R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-32. DMA Tx Wake Enable Register (DMATXWAKE_EN) Field Descriptions

Bits	Field Name	Value	Description
31-15	Reserved	0	Reserved.
14	XEMPTYEOFEN		Transmit Buffer Empty at End Of Frame enable bit.
		0	Transmit Buffer Empty at End Of Frame WK enable is NOT active.
		1	Transmit Buffer Empty at End Of Frame WK enable is active.
13-11	Reserved	0	Reserved.
10	XRDYEN		Transmit Buffer Threshold Reached WK enable bit.
		0	Transmit Buffer Threshold WK enable is NOT active.
		1	Transmit Buffer Threshold WK enable is active.
9	XEOFEN		Transmit End Of Frame WK enable bit.
		0	Transmit End Of Frame WK enable is NOT active.
		1	Transmit End Of Frame WK enable is active.
8	XFSXEN		Transmit Frame Synchronization WK enable bit.
		0	Transmit Frame Synchronization WK enable is NOT active.
		1	Transmit Frame Synchronization WK enable is active.
7	XSYNCERREN		Transmit Frame Synchronization Error WK enable bit.
		0	Transmit Frame Synchronization Error WK enable is NOT active.
		1	Transmit Frame Synchronization Error WK enable is active.
6-4	Reserved	0	Reserved.
3	RRDYEN		Receive Buffer Threshold wakeup enable bit.
		0	Receive Buffer Threshold WK enable is NOT active.
		1	Receive Buffer Threshold WK enable is active.
2	REOFEN		Receive End Of Frame WK enable bit.
		0	Receive End Of Frame WK enable is NOT active.
		1	Receive End Of Frame WK enable is active.
1	RFSREN		Receive Frame Synchronization WK enable bit.
		0	Receive Frame Synchronization WK enable is NOT active.
		1	Receive Frame Synchronization WK enable is active.

Table 15-32. DMA Tx Wake Enable Register (DMATXWAKE_EN) Field Descriptions (continued)

Bits	Field Name	Value	Description
0	RSYNCERREN		Receive Frame Synchronization Error WK enable bit.
		0	Receive Frame Synchronization Error WK enable is NOT active.
		1	Receive Frame Synchronization Error WK enable is active.

15.3.14 McBSP Data Receive Register (DRR_REG)

The McBSP_DRR_REG register is shown in Figure 15-46 and described in Table 15-33.

Figure 15-46. McBSP_DRR_REG

31	0
DRR	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 15-33. McBSP_DRR_REG Field Descriptions

Bits	Field Name	Value	Description
31-0	DRR	0	Data receive register.

15.3.15 McBSP Data Transmit Register (DXR_REG)

The McBSP_DXR_REG register is shown in Figure 15-47 and described in Table 15-34.

Figure 15-47. McBSP_DRR_REG

31	0
DXR	
W-0	

LEGEND: W = Write only; -*n* = value after reset

Table 15-34. McBSP_DXR_REG Field Descriptions

Bits	Field Name	Value	Description
31-0	DXR	0	Data transmit register.

15.3.16 McBSP Serial Port Control Register 2 (SPCR2_REG)

The McBSP_SPCR2_REG register is shown in Figure 15-48 and described in Table 15-35.

Figure 15-48. McBSP_SPCR2_REG

31	16
Reserved	
R-0	

15	10	9	8
Reserved		FREE	SOFT
R-0		R/W-0	R/W-0

7	6	5	4	3	2	1	0
FRST	GRST	XINTM		XSYNCERR	XEMPTY	XRDY	XRST
R/W-0	R/W-0	R/W-0		R/W-0	R-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-35. McBSP_SPCR2_REG Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	FREE		Free Running Mode (When this bit is set, the module ignores the Msuspend input)
		0	Free running mode is disabled
		1	Free running mode is enabled
8	SOFT		Soft bit.
		0	SOFT Mode is disabled: the McBSP module stops its activity immediately following MSuspend assertion.
		1	SOFT Mode is enabled: the McBSP module freezes its state after completion of the current operation when MSuspend is asserted.
7	FRST		Frame-Synchronization Generator Reset.
		0	Frame-Synchronization logic is reset. Frame-Synchronization signal FSG is not generated by the sample-rate generator.
		1	Frame-Synchronization signal FSG is generated after (FPER+1) number of CLKG clocks; all frame counters are loaded with their programmed values.
6	GRST		Sample-Rate Generator Reset.
		0	SRG is reset.
		1	SRG is pulled out of reset. CLKG is driven as per programmed value in SRG registers (SRGR[1,2])
5-4	XINTM		Transmit Interrupt Mode (legacy).
		0	Transmit interrupt is driven by XRDY.
		1h	Transmit interrupt generated by end-of-frame.
		2h	Transmit interrupt generated by a new frame synchronization.
		3h	Transmit interrupt generated by XSYNCERR.
3	XSYNCERR		Transmit Synchronization Error. (Writing 0 to this bit clear the legacy transmit interrupt if asserted due to XSYNCERROR condition).
		0	No synchronization error.
		1	Synchronization error detected by McBSP.
2	XEMPTY		Transmit Shift Register XSR Empty.
		0	XSR is empty.
		1	XSR is not empty.
1	XRDY		Transmitter ready.
		0	Transmitter is not ready.
		1	Transmitter is ready for new data in DXR.
0	XRST		Transmitter reset. This resets and enables the transmitter.
		0	The serial port transmitter is disabled and in reset state.
		1	The serial port transmitter is enabled.

15.3.17 McBSP Serial Port Control Register 1 (SPCR1_REG)

The McBSP_SPCR1_REG register is shown in Figure 15-49 and described in Table 15-36.

Figure 15-49. McBSP_SPCR1_REG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
ALB	RJUST		Reserved				
R/W-0	R/W-0		R-0				

7	6	5	4	3	2	1	0
DXENA	Reserved	RINTM		RSYNCERR	RFULL	RRDY	RRST
R/W-0	R-0	R/W-0		R/W-0	R-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-36. McBSP_SPCR1_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15	ALB		Analog loopback Mode.
		0	Analog loopback mode disabled.
		1	Analog loopback mode enabled.
14-13	RJUST		Receive Sign-Extension and Justification Mode.
		0	Right-justify and zero-fill MSBs in DRR.
		1h	Right-justify and sign-extend MSBs in DRR.
		2h	Left-justify and zero-fill LSBs in DRR.
		3h	Reserved.
12-8	Reserved	0	Reserved
7	DXENA		DX Enabler.
		0	DX enabler is off.
		1	DX enabler is on.
6	Reserved	0	Reserved
5-4	RINTM		Receive Interrupt Mode (legacy).
		0	Receive Interrupt driven by RRDY (end of word) and end of frame in A-bis mode.
		1h	Receive Interrupt generated by end-of-block or end-of-frame in multichannel operation.
		2h	Receive Interrupt generated by a new frame synchronization.
		3h	Receive Interrupt generated by RSYNCERR.
3	RSYNCERR		Receive Synchronization Error. (Writing 0 to this bit clear the legacy receive interrupt if asserted due to RSYNCERROR condition).
		0	No synchronization error.
		1	Synchronization error detected by McBSP.
2	RFULL		Receive Shift Register (RSR [1,2]) Full.
		0	RB is not in overrun condition.
		1	DRR is not read, RB is full and RSR is also full with new word.
1	RRDY		Receiver Ready.
		0	Receiver is not ready.
		1	Receiver is ready with data to be read from DRR.
0	RRST		Receiver reset. This resets and enables the receiver.
		0	The serial port receiver is disabled and in reset state.
		1	The serial port receiver is enabled.

15.3.18 McBSP Receive Control Register 2 (RCR2_REG)

The McBSP_RCR2_REG register is shown in Figure 15-50 and described in Table 15-37.

Figure 15-50. McBSP_RCR2_REG

31	16
Reserved	
R-0	

15	14	8	7	5	4	3	2	1	0
RPHASE	RFRLEN2		RWDLEN2		RREVERSE		Reserved	RDATDLY	
R/W-0	R/W-0		R/W-0		R/W-0		R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-37. McBSP_RCR2_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RPHASE		Receive Phases.
		0	Single-phase frame.
		1	Dual-phase frame.
14-8	RFRLEN2		Receive Frame Length 2
			Single-phase frame selected: RFRLEN2 = don't care.
			Dual-phase frame selected: RFRLEN2 = 000 0000 - 1 word per second phase (other values are reserved).
7-5	RWDLEN2		Receive Word Length 2.
		0	8 bits
		1h	12 bits
		2h	16 bits
		3h	20 bits
		4h	24 bits
		5h	32 bits
		6h-7h	Reserved (do not use)
4-3	RREVERSE		Receive reverse mode.
		0	Data transfer starts with MSB first.
		1h	Data transfer starts with LSB first.
		2h-3h	Reserved (do not use)
2	Reserved	0	Reserved
1-0	RDATDLY		Receive Data Delay
		0	0-bit data delay
		1h	1-bit data delay
		2h	2-bit data delay
		3h	Reserved

15.3.19 McBSP Receive Control Register 1 (RCR1_REG)

The McBSP_RCR1_REG register is shown in Figure 15-51 and described in Table 15-38.

Figure 15-51. McBSP_RCR1_REG

31 15	14 8	7 5	4 0
Reserved	RFRLEN1	RWDLEN1	Reserved
R-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 15-38. McBSP_RCR1_REG Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14-8	RFRLEN1		Receive Frame Length 1. Single-phase frame selected: RFRLEN1 = 000 0000 - 1 word per frame RFRLEN1 = 000 0001 - 2 words per frame. RFRLEN1 = 111 1111 - 128 words per frame. Dual-phase frame selected: RFRLEN1 = 000 0000 - 1 word per phase (other values are reserved).
7-5	RWDLEN1		Receive Word Length 1
		0	8 bits
		1h	12 bits
		2h	16 bits
		3h	20 bits
		4h	24 bits
		5h	32 bits
		6h-7h	Reserved (do not use).
4-0	Reserved	0	Reserved

15.3.20 McBSP Transmit Control Register 2 (XCR2_REG)

The McBSP_XCR2_REG register is shown in Figure 15-52 and described in Table 15-39.

Figure 15-52. McBSP_XCR2_REG

31	16
Reserved	
R-0	

15	14	8	7	5	4	3	2	1	0
XPHASE	XFRLEN2		XWDLEN2		XREVERSE		Reserved	XDATDLY	
R/W-0	R/W-0		R/W-0		R/W-0		R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-39. McBSP_XCR2_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	XPHASE		Transmit Phases.
		0	Single-phase frame.
		1	Dual-phase frame.
14-8	XFRLEN2		Transmit Frame Length 2.
			Single-phase frame selected: XFRLEN2 = don't care.
			Dual-phase frame selected: XFRLEN2 = 000 0000 - 1 word per second phase (other values are reserved).
7-5	XWDLEN2		Transmit Word Length 2
		0	8 bits
		1h	12 bits
		2h	16 bits
		3h	20 bits
		4h	24 bits
		5h	32 bits
		6h-7h	Reserved (do not use)
4-3	XREVERSE		Transmit reverse mode.
		0	Data transfer starts with MSB first.
		1h	Data transfer starts with LSB first.
		2h-3h	Reserved (do not use).
2	Reserved	0	Reserved
1-0	XDATDLY		Transmit Data Delay
		0	0-bit data delay
		1h	1-bit data delay
		2h	2-bit data delay
		3h	Reserved

15.3.21 McBSP Transmit Control Register 1 (XCR1_REG)

The McBSP_XCR1_REG register is shown in Figure 15-53 and described in Table 15-40.

Figure 15-53. McBSP_XCR1_REG

31 15	14 8	7 5	4 0
Reserved	XFRLEN1	XWDLEN1	Reserved
R-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 15-40. McBSP_XCR1_REG Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14-8	XFRLEN1		Transmit Frame Length 1 Single-phase frame selected: XFRLEN1 = 000 0000 - 1 word per frame. XFRLEN1 = 000 0001 - 2 words per frame. XFRLEN1 = 111 1111 - 128 words per frame. Dual-phase frame selected: XFRLEN1 = 000 0000 - 1 word per phase (other values are reserved).
7-5	XWDLEN1		Transmit Word Length 1
		0	8 bits
		1h	12 bits
		2h	16 bits
		3h	20 bits
		4h	24 bits
		5h	32 bits
		6h-7h	Reserved (do not use)
4-0	Reserved	0	Reserved

15.3.22 McBSP Sample Rate Generator Register 2 (SRGR2_REG)

The McBSP_SRGR2_REG register is shown in Figure 15-54 and described in Table 15-41.

Figure 15-54. McBSP_SRGR2_REG

31					16
Reserved					
R-0					
15	14	13	12	11	0
GSYNC	CLKSP	CLKSM	FSGM	FPER	
R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 15-41. McBSP_SRGR2_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	GSYNC		Sample Rate Generator (SRG) Synchronization
			Only used when the external clock (CLKS) drives the SRG clock (CLKSM=0).
		0	The SRG clock (CLKG) is free running.
		1	The SRG clock (CLKG) is running, but CLKG is resynchronized and frame-synchronization signal (FSG) is generated only after detecting the receive frame-synchronization signal (FSR).
			Also, frame period, FPER, is a don't care because the period is dictated by the external frame-synchronization pulse.
14	CLKSP		CLKS Polarity Clock Edge Select
			Only used when the external clock CLKS drives the SRG clock (CLKSM=0).
		0	Rising edge of CLKG and FSG.
		1	Falling edge of CLKG and FSG.
13	CLKSM		McBSP SRG Clock Mode
		0	SCLKME = 0: SRG clock derived from the CLKS pin.
			SCLKME = 1: SRG clock derived from the the CLKRI pin.
		1	SCLKME = 0: SRG clock derived from the OCP clock.
			SCLKME = 1: SRG clock derived from the CLKXI pin.
12	FSGM		Sample Rate Generator Transmit Frame-Synchronization Mode. Used when FSXM=1 in the PCR.
		0	Transmit frame-synchronization signal (FSX) is generated when transmit buffer is not empty.
			When FSGM = 0, FPER and FWID are used to determine the frame synchronization period and width (external FSX is gated by the buffer empty condition).
		1	Transmit frame-synchronization signal driven by the SRG frame-synchronization signal, FSG.
11-0	FPER	0-FFFh	Frame Period. This value + 1 determines when the next frame-synchronization signal becomes active.
			Range: 1 to 4096 CLKG periods.

15.3.23 McBSP Sample Rate Generator Register 1 (SRGR1_REG)

The McBSP_SRGR1_REG register is shown in Figure 15-55 and described in Table 15-42.

Figure 15-55. McBSP_SRGR1_REG

31 16	15 8	7 0
Reserved	FWID	CLKGDV
R-0	R/W-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 15-42. McBSP_SRGR1_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-8	FWID	0-FFh	Frame Width. This value + 1 determines the width of the frame-synchronization pulse, FSG, during its active period. Range: 1 to 256 CLKG periods.
7-0	CLKGDV	1-FFh	Sample Rate Generator Clock Divider. This value is used as the divide-down number to generate the required SRG clock frequency. Default value is 1.

15.3.24 McBSP Multichannel Register 2 (MCR2_REG)

The McBSP_MCR2_REG register is shown in Figure 15-56 and described in Table 15-43.

Figure 15-56. McBSP_MCR2_REG

31	16
Reserved	
R-0	

15	10	9	8	7	6	5	4	2	1	0
Reserved		XMCME	XPBBLK		XPABLK		Reserved		XMCM	
R-0		R/W-0	R/W-0		R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-43. McBSP_MCR2_REG Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	XMCME		XMCME Transmit multichannel partition mode bit 0. XMCME is only applicable if channels can be individually disabled/enabled or masked/unmasked for transmission (XMCM is non-zero). XMCME determines whether only 32 channels or all 128 channels are to be individually selectable.
		0	**2-partition mode (legacy only):** Only partitions A and B are used. You can control up to 32 channels in the transmit multichannel selection mode selected with the XMCM bits. If XMCM = 01b or 10b, assign 16 channels to partition A with the XPABLK bits. Assign 16 channels to partition B with the XPBBLK bits. If XMCM = 11b (for symmetric transmission and reception), assign 16 channels to receive partition A with the RPABLK bits. Assign 16 channels to receive partition B with the RPBBLK bits You control the channels with the appropriate transmit channel enable registers: XCERA: Channels in partition A. XCERB: Channels in partition B.
		1	**8-partition mode:** All partitions (A through H) are used. You can control up to 128 channels in the transmit multichannel selection mode selected with the XMCM bits. You control the channels with the appropriate transmit channel enable registers: XCERA: Channels 0 through 15 XCERB: Channels 16 through 31 XCERC: Channels 32 through 47 XCERD: Channels 48 through 63 XCERE: Channels 64 through 79 XCERF: Channels 80 through 95 XCERG: Channels 96 through 111 XCERH: Channels 112 through 127
8-7	XPBBLK		Transmit Partition B Block (legacy only)
		0	Block 1. Channel 16 to channel 31
		1h	Block 3. Channel 48 to channel 63
		2h	Block 5. Channel 80 to channel 95
		3h	Block 7. Channel 112 to channel 127
6-5	XPABLK		Transmit Partition A Block (legacy only)
		0	Block 0. Channel 0 to channel 15
		1h	Block 2. Channel 32 to channel 47
		2h	Block 4. Channel 64 to channel 79
		3h	Block 6. Channel 96 to channel 111

Table 15-43. McBSP_MCR2_REG Field Descriptions (continued)

Bit	Field	Value	Description
4-2	Reserved	0	Reserved
1-0	XMCM		Transmit Multichannel Selection Enable
		0	All channels enabled without masking (DX is always driven during transmission of data).
		1h	All channels disabled and therefore masked by default. Required channels are selected by enabling XP(A/B)BLK and XCER(A/B) appropriately. Also, these selected channels are not masked and therefore DX is always driven.
		2h	All channels enabled, but masked. Selected channels enabled via XP(A/B)BLK and XCER(A/B) are unmasked.
		3h	All channels disabled and therefore masked by default. Required channels are selected by enabling RP(A/B)BLK and RCER(A/B) appropriately. Selected channels can be unmasked by RP(A/B)BLK and XCER(A/B). This mode is used for symmetric transmit and receive operation.

15.3.25 McBSP Multichannel Register 1 (MCR1_REG)

The McBSP_MCR1_REG register is shown in Figure 15-57 and described in Table 15-44.

Figure 15-57. McBSP_MCR1_REG

31	16
Reserved	
R-0	

15	10	9	8	7	6	5	4	1	0
Reserved		RMCME	RPBBLK		RPABLK		Reserved		RMCM
R-0		R/W-0	R/W-0		R/W-0		R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-44. McBSP_MCR1_REG Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	RMCME		RMCME Receive multichannel partition mode bit 0 . Receive multichannel partition mode determines whether only 32 channels or all 128 channels are to be individually selectable. RMCME is only applicable if channels can be individually enabled or disabled for reception (RMCM = 1).
		0	**2-partition mode (legacy only).** Only partitions A and B are used. You can control up to 32 channels in the receive multichannel selection mode (RMCM = 1). Assign 16 channels to partition A with the RPABLK bits. Assign 16 channels to partition B with the RPBBLK bits. You can control the channels with the appropriate receive channel enable registers: RCERA: Channels in partition A RCERB: Channels in partition B
		1	**8-partition mode:** All partitions (A through H) are used. Control up to 128 channels in the receive multichannel selection mode. You can control the channels with the appropriate receive channel enable registers: RCERA: Channels 0 through 15 RCERB: Channels 16 through 31 RCERC: Channels 32 through 47 RCERD: Channels 48 through 63 RCERE: Channels 64 through 79 RCERF: Channels 80 through 95 RCERG: Channels 96 through 111 RCERH: Channels 112 through 127
8-7	RPBBLK		Receive Partition B Block (legacy only)
		0	Block 1. Channel 16 to channel 31
		1h	Block 3. Channel 48 to channel 63
		2h	Block 5. Channel 80 to channel 95
		3h	Block 7. Channel 112 to channel 127
6-5	RPABLK		Receive Partition A Block (legacy only)
		0	Block 0. Channel 0 to channel 15
		1h	Block 2. Channel 32 to channel 47
		2h	Block 4. Channel 64 to channel 79
		3h	Block 6. Channel 96 to channel 111
4-1	Reserved	0	Reserved.

Table 15-44. McBSP_MCR1_REG Field Descriptions (continued)

Bit	Field	Value	Description
0	RMCM		Receive Multichannel Selection Enable
		0	All 128 channels
		1	All channels disabled by default.
			Required channels are selected by enabling RP(A/B)BLK and RCER(A/B) appropriately.

15.3.26 McBSP Receive Channel Enable Register Partition A (RCERA_REG)

The McBSP_RCERA_REG register is shown in Figure 15-58 and described in Table 15-45.

Figure 15-58. McBSP_RCERA_REG

31	16	15	0
Reserved		RCERA	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-45. McBSP_RCERA_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	RCERA		Receive Channel Enable. RCERA n = 0. Disables reception of n-th channel in an even-numbered block in partition A. RCERA n = 1. Enables reception of n-th channel in an even-numbered block in partition A.

15.3.27 McBSP Receive Channel Enable Register Partition B (RCERB_REG)

The McBSP_RCERB_REG register is shown in Figure 15-59 and described in Table 15-46.

Figure 15-59. McBSP_RCERB_REG

31	16	15	0
Reserved		RCERB	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-46. McBSP_RCERB_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	RCERB		Receive Channel Enable. RCERB n = 0. Disables reception of n-th channel in a odd-numbered block in partition B. RCERB n = 1. Enables reception of n-th channel in a odd-numbered block in partition B.

15.3.28 McBSP Transmit Channel Enable Register Partition A (XCERA_REG)

The McBSP_XCERA_REG register is shown in Figure 15-60 and described in Table 15-47.

Figure 15-60. McBSP_XCERA_REG

31	16	15	0
Reserved		XCERA	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-47. McBSP_XCERA_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	XCERA		Transmit Channel Enable. XCERA n = 0. Disables transmission of n-th channel in an even-numbered block in partition A. XCERA n = 1. Enables transmission of n-th channel in an even-numbered block in partition A.

15.3.29 McBSP Transmit Channel Enable Register Partition B (XCERB_REG)

The McBSP_XCERB_REG register is shown in Figure 15-61 and described in Table 15-48.

Figure 15-61. McBSP_XCERB_REG

31	16	15	0
Reserved		XCERB	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-48. McBSP_XCERB_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	XCERB		Transmit Channel Enable. XCERB n = 0. Disables transmission of n-th channel in an odd-numbered block in partition B. XCERB n = 1. Enables transmission of n-th channel in an odd-numbered block in partition B.

15.3.30 McBSP Pin Control Register (PCR_REG)

The McBSP_PCR_REG register is shown in Figure 15-62 and described in Table 15-49.

Figure 15-62. McBSP_PCR_REG

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	IDLE_EN	XIOEN	RIOEN	FSXM	FSRM	CLKXM	CLKRM
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
SCLKME	CLKS_STAT	DX_STAT	DR_STAT	FSXP	FSRP	CLKXP	CLKRP
R/W-0	R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-49. McBSP_PCR_REG Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved.
14	IDLE_EN		Idle enable (legacy only).
			This bit allows stopping all the clocks in the McBSP module. (legacy)
		0	The McBSP is running
		1	The clocks in the McBSP are shut off when both IDLE_EN = 1 and this power domain is in idle mode (Force idle or Smart idle)
13	XIOEN		Transmit General Purpose I/O Mode only when XRST = 0 in SPCR[1,2]. (legacy only)
		0	DX, FSX and CLKX are configured as serial port pins and do not function as general-purpose I/Os.
		1	DX pin is a general purpose output. FSX and CLKX are general purpose I/Os. These serial port pins do not perform serial port operation.
12	RIOEN		Receive General Purpose I/O Mode when RRST = 0 in SPCR[1,2]. (legacy)
		0	DR, FSR, CLKR and CLKS are configured as serial port pins and do not function as general-purpose I/Os.
		1	DR and CLKS pins are general purpose inputs; FSR and CLKR are general purpose I/Os. These serial port pins do not perform serial port operation. The CLKS pin is affected by a combination of RRST and RIOEN signals of the receiver.
11	FSXM		Transmit Frame-Synchronization Mode.
		0	Frame-synchronization signal derived from an external source.
		1	Frame synchronization is determined by the SRG frame-synchronization mode bit FSGM in SRGR2.
10	FSRM		Receive Frame-Synchronization Mode.
		0	Frame-Synchronization pulses generated by an external device. FSR is an input pin.
		1	Frame synchronization generated internally by SRG. FSR is an output pin except when GSYNC=1 in SRGR.
9	CLKXM		Transmitter Clock Mode.
			When digital loop-back mode set (McBSPi.McBSP_XCCR_REG[5] DLB = 1), CLKXM bit is ignored. The internal transmit clock (not the mcbspi_clkx pin) is driven by the internal SRG and mcbspi_clkx pin is in high-impedance.
		0	Transmitter clock is driven by an external clock with CLKX as an input pin.
		1	CLKX is an output pin and is driven by the internal SRG.

Table 15-49. McBSP_PCR_REG Field Descriptions (continued)

Bit	Field	Value	Description
8	CLKRM		Receiver Clock Mode.
		0	Case 1: Analog loop back mode not set (ALB = 0) in SPCR1:
			Receive clock (CLKR) is an input driven by an external clock.
			Case 2: Analog loop back mode set (ALB = 1) in SPCR1:
			Receive clock (not the CLKR pin) is driven by transmit clock (CLKX), which is based on the CLKXM bit in the PCR. CLKR pin is in high-impedance.
		1	Case 1: Analog loop back mode not set (ALB = 0) in SPCR1:
			CLKR is an output pin and is driven by the internal sample rate generator.
			Case 2: Analog loop back mode set (ALB = 1) in SPCR1:
			CLKR is an output pin and is driven by the transmit clock. The transmit clock is derived based on the CLKRM bit in the PCR.
7	SCLKME		SCLKME Sample rate generator input clock mode bit 0.
			The sample rate generator can produce a clock signal, CLKG. The frequency of CLKG is:
			CLKG frequency = (Input clock frequency) / (CLKGDV + 1)
			SCLKME is used in conjunction with the CLKSM bit to select the input clock:
		0	CLKSM = 0: Signal on CLKS pin.
			CLKSM = 1: McBSP_OCP clock.
		1	CLKSM = 0: Signal on CLKR pin.
			CLKSM = 1: Signal on CLKX pin.
6	CLKS_STAT		CLKS pin status. Reflects value on CLKS pin when selected as a general purpose input. (legacy)
		0	The signal on the CLKS pin is low.
		1	The signal on the CLKS pin is high.
5	DX_STAT		DX pin status. Reflects value driven on to DX pin when selected as a general purpose output. (legacy)
		0	Drive the signal on the DX pin low.
		1	Drive the signal on the DX pin high.
4	DR_STAT		DR pin status. Reflects value on DR pin when selected as a general purpose input. (legacy)
		0	The signal on DR pin is low.
		1	The signal on DR pin is high.
3	FSXP		Transmit Frame-Synchronization Polarity.
		0	Frame-synchronization pulse FSX is active high.
		1	Frame-synchronization pulse FSX is active low.
2	FSRP		Receive Frame-Synchronization Polarity.
		0	Frame-synchronization pulse FSR is active high.
		1	Frame-synchronization pulse FSR is active low.
1	CLKXP		Transmit Clock Polarity.
		0	Transmit data driven on rising edge of CLKX.
		1	Transmit data driven on falling edge of CLKX.
0	CLKRP		Receive Clock Polarity.
		0	Receive data sampled on falling edge of CLKR.
		1	Receive data sampled on rising edge of CLKR.

15.3.31 McBSP Receive Channel Enable Register Partition C (RCERC_REG)

The McBSP_RCERC_REG register is shown in Figure 15-63 and described in Table 15-50.

Figure 15-63. McBSP_RCERC_REG

31	16	15	0
Reserved		RCERC	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-50. McBSP_RCERC_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	RCERC		Receive Channel Enable. RCERC n = 0. Disables reception of n-th channel in block 2 in partition C. RCERC n = 1. Enables reception of n-th channel in block 2 in partition C.

15.3.32 McBSP Receive Channel Enable Register Partition D (RCERD_REG)

The McBSP_RCERD_REG register is shown in Figure 15-64 and described in Table 15-51.

Figure 15-64. McBSP_RCERD_REG

31	16	15	0
Reserved		RCERD	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-51. McBSP_RCERD_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0			Receive Channel Enable. RCERD n = 0. Disables reception of n-th channel in block 3 in partition D. RCERD n = 1. Enables reception of n-th channel in block 3 in partition D.

15.3.33 McBSP Transmit Channel Enable Register Partition C (XCERC_REG)

The McBSP_XCERC_REG register is shown in Figure 15-65 and described in Table 15-52.

Figure 15-65. McBSP_XCERC_REG

31	16	15	0
Reserved		XCERC	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-52. McBSP_XCERC_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	XCERC		Transmit Channel Enable. XCERC n = 0. Disables transmission of n-th channel in block 2 in partition C. XCERC n = 1. Enables transmission of n-th channel in block 2 in partition C.

15.3.34 McBSP Transmit Channel Enable Register Partition D (XCERD_REG)

The McBSP_XCERD_REG register is shown in Figure 15-66 and described in Table 15-53.

Figure 15-66. McBSP_XCERD_REG

31	16	15	0
Reserved		XCERD	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-53. McBSP_XCERD_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	XCERD		Transmit Channel Enable. XCERD n = 0. Disables transmission of n-th channel in block 3 in partition D. XCERD n = 1. Enables transmission of n-th channel in block 3 in partition D.

15.3.35 McBSP Receive Channel Enable Register Partition E (RCERE_REG)

The McBSP_RCERE_REG register is shown in Figure 15-67 and described in Table 15-54.

Figure 15-67. McBSP_RCERE_REG

31	16	15	0
Reserved		RCERE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-54. McBSP_RCERE_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	RCERE		Receive Channel Enable. RCERE n = 0. Disables reception of n-th channel in block 4 in partition E. RCERE n = 1. Enables reception of n-th channel in block 4 in partition E.

15.3.36 McBSP Receive Channel Enable Register Partition F (RCERF_REG)

The McBSP_RCERF_REG register is shown in Figure 15-68 and described in Table 15-55.

Figure 15-68. McBSP_RCERF_REG

31	16	15	0
Reserved		RCERF	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-55. McBSP_RCERF_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	RCERF		Receive Channel Enable. RCERF n = 0. Disables reception of n-th channel in block 5 in partition F. RCERF n = 1. Enables reception of n-th channel in block 5 in partition F.

15.3.37 McBSP Transmit Channel Enable Register Partition E (XCERE_REG)

The McBSP_XCERE_REG register is shown in Figure 15-69 and described in Figure 23-35.

Figure 15-69. McBSP_XCERE_REG

31	16	15	0
Reserved		XCERE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-56. McBSP_XCERE_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	XCERE		Transmit Channel Enable. XCERE n = 0. Disables transmission of n-th channel in block 4 in partition E. XCERE n = 1. Enables transmission of n-th channel in block 4 in partition E.

15.3.38 McBSP Transmit Channel Enable Register Partition F (XCERF_REG)

The McBSP_XCERF_REG register is shown in Figure 15-70 and described in Table 15-57.

Figure 15-70. McBSP_XCERF_REG

31	16	15	0
Reserved		XCERF	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-57. McBSP_XCERF_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	XCERF		Transmit Channel Enable. XCERF n = 0. Disables transmission of n-th channel in block 5 in partition F. XCERF n = 1. Enables transmission of n-th channel in block 5 in partition F.

15.3.39 McBSP Receive Channel Enable Register Partition G (RCERG_REG)

The McBSP_RCERG_REG register is shown in Figure 15-71 and described in Table 15-58.

Figure 15-71. McBSP_RCERG_REG

31	16	15	0
Reserved		RCERG	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-58. McBSP_RCERG_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	RCERG		Receive Channel Enable. RCERG n = 0. Disables reception of n-th channel in block 6 in partition G. RCERG n = 1. Enables reception of n-th channel in block 6 in partition G.

15.3.40 McBSP Receive Channel Enable Register Partition H (RCERH_REG)

The McBSP_RCERH_REG register is shown in Figure 15-72 and described in Table 15-59.

Figure 15-72. McBSP_RCERH_REG

31	16	15	0
Reserved		RCERH	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-59. McBSP_RCERH_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	RCERH		Receive Channel Enable. RCERH n = 0. Disables reception of n-th channel in block 7 in partition H. RCERH n = 1. Enables reception of n-th channel in block 7 in partition H.

15.3.41 McBSP Transmit Channel Enable Register Partition G (XCERG_REG)

The McBSP_XCERG_REG register is shown in Figure 15-73 and described in Table 15-60.

Figure 15-73. McBSP_XCERG_REG

31	16	15	0
Reserved		XCERG	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-60. McBSP_XCERG_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	XCERG		Transmit Channel Enable. XCERG n = 0. Disables transmission of n-th channel in block in partition G. XCERG n = 1. Enables transmission of n-th channel in block in partition G.

15.3.42 McBSP Transmit Channel Enable Register Partition H (XCERH_REG)

The McBSP_XCERH_REG register is shown in Figure 15-74 and described in Table 15-61.

Figure 15-74. McBSP_XCERH_REG

31	16	15	0
Reserved		XCERH	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-61. McBSP_XCERH_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	XCERH		Transmit Channel Enable. XCERH n = 0. Disables transmission of n-th channel in block in partition H. XCERH n = 1. Enables transmission of n-th channel in block in partition H.

15.3.43 McBSP Transmit Buffer Threshold Register (DMA or IRQ Trigger) (THRSH2_REG)

The McBSP_THRSH2_REG register is shown in Figure 15-75 and described in Table 15-62.

Figure 15-75. McBSP_THRSH2_REG

31	12	11	0
Reserved		XTHRESHOLD	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 15-62. McBSP_THRSH2_REG Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11-0	XTHRESHOLD		Transmit buffer threshold value. The DMA request (if enabled) of interrupt assertion (if enabled) will be triggered if the number of free locations inside transmit buffer are above or equal to the XTHRESHOLD value + 1. Also, this value (XTHRESHOLD value + 1) indicates the number of words transferred during a transmit data DMA request, if transmit DMA is enabled.

15.3.44 McBSP Receive Buffer Threshold Register (DMA or IRQ trigger) (THRSH1_REG)

The McBSP_THRSH1_REG register is shown in Figure 15-76 and described in Table 15-63.

Figure 15-76. McBSP_THRSH1_REG

31	12	11	0
Reserved		RTHRESHOLD	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 15-63. McBSP_THRSH1_REG Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved.
11-0	RTHRESHOLD		Receive buffer threshold value. The DMA request (if enabled) of interrupt assertion (if enabled) will be triggered if the number of occupied locations inside receive buffer are above or equal to the RTHRESHOLD value + 1. Also, this value (RTHRESHOLD value + 1) indicates the number of words transferred during a receive data DMA request, if receive DMA is enabled.

15.3.45 McBSP Interrupt Status Register (OCP compliant IRQ line) (IRQSTATATUS)

The McBSP_IRQSTATUS_REG register is shown in Figure 15-77 and described in Table 15-64.

Figure 15-77. McBSP_IRQSTATUS_REG

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	XEMPTYEOF	Reserved	XOVFLSTAT	XUNDFLSTAT	XRDY	XEOF	XFSX
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
XSYNCERR	Reserved	ROVFLSTAT	RUNDFLSTAT	RRDY	REOF	RFSR	RSYNCERR
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-64. McBSP_IRQSTATUS_REG Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14	XEMPTYEOF		Transmit Buffer Empty at end of frame (XEMPTYEOF is set to 1 when a complete frame was transmitted and the transmit buffer is empty).
		0	Transmit buffer NOT Empty at end of frame.
		1	Transmit buffer Empty at end of frame; Writing 1 to this bit clears the bit.
13	Reserved	0	Reserved.
12	XOVFLSTAT		Transmit Buffer Overflow (XOVFLSTAT bit is set to 1 when transmit buffer overflow; the data which is written while overflow condition is discarded).
		0	Transmit buffer NOT overflow.
		1	Transmit buffer overflow; Writing 1 to this bit clears the bit.
11	XUNDFLSTAT		Transmit Buffer Underflow (XUNDFLSTAT bit is set to 1 when the transmit data buffer is empty new data is required to be transmitted).
		0	The transmit data buffer is NOT empty new data is required to be transmitted.
		1	The transmit data buffer is empty new data is required to be transmitted. Writing 1 to this bit clears the bit.
10	XRDY		Transmit Buffer Threshold Reached (XRDY bit is set to 1 when the transmit buffer free locations are equal or above the THRSH2_REG value).
		0	Transmit buffer occupied locations are below the THRSH2_REG value).
		1	Transmit buffer occupied locations are equal or above the THRSH2_REG value). Writing 1 to this bit clears the bit.
9	XEOF		Transmit End Of Frame (XEOF is set to 1when a complete frame was transmitted).
		0	Complete frame was NOT transmitted.
		1	Complete frame was transmitted; Writing 1 to this bit clears the bit.
8	XFSX		Transmit Frame Synchronization (XFSX bit is set to 1 when a new transmit frame synchronization is asserted).
		0	New transmit frame synchronization is NOT asserted.
		1	New transmit frame synchronization is asserted; Writing 1 to this bit clears the bit.
7	XSYNCERR		Transmit Frame Synchronization Error (XSYNCERR is set to 1 when a transmit frame synchronization error is detected).
		0	Transmit frame synchronization error is NOT detected.
		1	Transmit frame synchronization error is detected. Writing 1 to this bit clears the bit.
6	Reserved	0	Reserved.

Table 15-64. McBSP_IRQSTATUS_REG Field Descriptions (continued)

Bit	Field	Value	Description
5	ROVFLSTAT		Receive Buffer Overflow (ROVFLSTAT bit is set to 1 when receive buffer overflow; the data which is written while overflow condition is discarded).
		0	Receive buffer NOT overflow.
		1	Receive buffer overflow; Writing 1 to this bit clears the bit.
4	RUNDFLSTAT		Receive Buffer Underflow (RUNDFLSTAT bit is set to 1 when read operation is performed to the receive data register while receive buffer is empty; data read while underflow condition is undefined).
		0	Read operation is performed to the receive data register while receive buffer is NOT empty.
		1	Read operation is performed to the receive data register while receive buffer is empty; Writing 1 to this bit clears the bit.
3	RRDY		Receive Buffer Threshold Reached (RRDY bit is set to 1 when the receive buffer occupied locations are equal or above the THRSH1_REG value).
		0	Receive buffer occupied locations are below the THRSH1_REG value).
		1	Receive buffer occupied locations are equal or above the THRSH1_REG value). Writing 1 to this bit clears the bit.
2	REOF		Receive End Of Frame (REOF is set to 1 when a complete frame was received). Writing 1 to this bit clears the bit.
		0	Complete frame was NOT received.
		1	Complete frame was received; Writing 1 to this bit clears the bit.
1	RFSR		Receive Frame Synchronization (RFSR bit is set to 1 when a new receive frame synchronization is asserted).
		0	New receive frame synchronization is NOT asserted.
		1	New receive frame synchronization is asserted; Writing 1 to this bit clears the bit.
0	RSYNCERR		Receive Frame Synchronization Error (RSYNCERR is set RW 0x0 to 1 when a receive frame synchronization error is detected).
		0	Receive frame synchronization error is NOT detected.
		1	Receive frame synchronization error is detected. Writing 1 to this bit clears the bit.

15.3.46 McBSP Interrupt Enable Register (OCP compliant IRQ line) (IRQENABLE)

The McBSP_IRQENABLE_REG register is shown in Figure 15-78 and described in Table 15-65.

Figure 15-78. McBSP_IRQENABLE_REG

31	30	29	28	27	26	25	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved	XEMPTYEOFEN	Reserved	XOVFLEN	XUNDFLEN	XRDYEN	XEOFEN	XFSXEN
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
XSYNCERREN	Reserved	ROVFLEN	RUNDFLEN	RRDYEN	REOFEN	RFSREN	RSYNCERREN
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-65. McBSP_IRQENABLE_REG Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved.
14	XEMPTYEOFEN		Transmit buffer empty at end of frame enable bit.
		0	Transmit buffer Empty at end of frame NOT enabled.
		1	Transmit buffer Empty at end of frame enabled.
13	Reserved	0	Reserved.
12	XOVFLEN		Transmit Buffer Overflow enable bit.
		0	Transmit Buffer Overflow NOT enabled.
		1	Transmit Buffer Overflow enabled.
11	XUNDFLEN		Transmit Buffer Underflow enable bit.
		0	Transmit Buffer Underflow NOT enabled.
		1	Transmit Buffer Underflow enabled.
10	XRDYEN		Transmit Buffer Threshold Reached enable bit.
		0	Transmit Buffer Threshold Reached NOT enabled.
		1	Transmit Buffer Threshold Reached enabled.
9	XEOFEN		Transmit End Of Frame enable bit.
		0	Transmit End Of Frame NOT enabled.
		1	Transmit End Of Frame enabled.
8	XFSXEN		Transmit Frame Synchronization enable bit.
		0	Transmit Frame Synchronization NOT enabled.
		1	Transmit Frame Synchronization enabled.
7	XSYNCERREN		Transmit Frame Synchronization Error enable bit.
		0	Transmit Frame Synchronization Error NOT enabled.
		1	Transmit Frame Synchronization Error enabled.
6	Reserved	0	Reserved.
5	ROVFLEN		Receive Buffer Overflow enable bit.
		0	Receive Buffer Overflow NOT enabled.
		1	Receive Buffer Overflow enabled.
4	RUNDFLEN		Receive Buffer Underflow enable bit.
		0	Receive Buffer Underflow NOT enabled.
		1	Receive Buffer Underflow enabled.

Table 15-65. McBSP_IRQENABLE_REG Field Descriptions (continued)

Bit	Field	Value	Description
3	RRDYEN		Receive Buffer Threshold enable bit.
		0	Receive Buffer Threshold NOT enabled.
		1	Receive Buffer Threshold enabled.
2	REOFEN		Receive End Of Frame enable bit.
		0	Receive End Of Frame NOT enabled.
		1	Receive End Of Frame enabled.
1	RFSREN		Receive Frame Synchronization enable bit.
		0	Receive Frame Synchronization NOT enabled.
		1	Receive Frame Synchronization enabled.
0	RSYNCERREN		Receive Frame Synchronization Error enable bit.
		0	Receive Frame Synchronization Error NOT enabled.
		1	Receive Frame Synchronization Error enabled.

15.3.47 McBSP Wakeup Enable Register (WAKEUPEN)

The McBSP_WAKEUPEN_REG register is shown in Figure 15-79 and described in Table 15-66.

Figure 15-79. McBSP_WAKEUPEN_REG

31	16
Reserved	
R-0	

15	14	13	11	10	9	8
Reserved	XEMPTYEOFEN	Reserved		XRDYEN	XEOFEN	XFSXEN
R-0	R/W-0	R-0		R/W-0	R/W-0	R/W-0

7	6	4	3	2	1	0
XSYNCERREN	Reserved		RRDYEN	REOFEN	RFSREN	RSYNCERREN
R/W-0	R-0		R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-66. McBSP_WAKEUPEN_REG Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved.
14	XEMPTYEOFEN		Transmit buffer empty at end of frame WK enable bit.
		0	Transmit buffer Empty at end of frame WK enable is NOT active.
		1	Transmit buffer Empty at end of frame WK enable is active.
13-11	Reserved	0	Reserved.
10	XRDYEN		Transmit Buffer Threshold Reached WK enable bit.
		0	Transmit Buffer Threshold WK enable is NOT active.
		1	Transmit Buffer Threshold WK enable is active.
9	XEOFEN		Transmit End Of Frame WK enable bit.
		0	Transmit End Of Frame WK enable is NOT active.
		1	Transmit End Of Frame WK enable is active.
8	XFSXEN		Transmit Frame Synchronization WK enable bit.
		0	Transmit Frame Synchronization WK enable is NOT active.
		1	Transmit Frame Synchronization WK enable is active.
7	XSYNCERREN		Transmit Frame Synchronization Error WK enable bit.
		0	Transmit Frame Synchronization Error WK enable is NOT active.
		1	Transmit Frame Synchronization Error WK enable is active.
6-4	Reserved	0	Reserved.
3	RRDYEN		Receive Buffer Threshold wake-up enable bit.
		0	Receive Buffer Threshold WK enable is NOT active.
		1	Receive Buffer Threshold WK enable is active.
2	REOFEN		Receive End Of Frame WK enable bit.
		0	Receive End Of Frame WK enable is NOT active.
		1	Receive End Of Frame WK enable is active.
1	RFSREN		Receive Frame Synchronization WK enable bit.
		0	Receive Frame Synchronization WK enable is NOT active.
		1	Receive Frame Synchronization WK enable is active.
0	RSYNCERREN		Receive Frame Synchronization Error WK enable bit.
		0	Receive Frame Synchronization Error WK enable is NOT active.
		1	Receive Frame Synchronization Error WK enable is active.

15.3.48 McBSP Transmit Configuration Control Register (XCCR_REG)

The McBSP_XCCR_REG register is shown in Figure 15-80 and described in Table 15-67.

Figure 15-80. McBSP_XCCR_REG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
EXTCLKGATE	PPCONNECT	DXENDLY		XFULL_CYCLE	Reserved		
R/W-0	R/W-0	R/W-1		R/W-0	R-0		

7	6	5	4	3	2	1	0
Reserved		DLB	Reserved	XDMAEN	Reserved		XDISABLE
R-0		R/W-0	R-0	R/W-1	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-67. McBSP_XCCR_REG Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15	EXTCLKGATE		External clock gating enable (CLKX and FSX master only). When this bit is set and the transmit clock and FSX are set as output, the CLKX is enabled when FSX is active plus 3 clock cycles after (clock is provided for FWID + 4 clock cycles, assuming that the FSX width, active, is FWID + 1 clock cycles); outside this window the external transmit clock is gated. The receive uses the same gated transmit clock and transmit frame synchronization signals regardless of the CLKRM/FSRM settings. When using this mode the frame synchronization signal must be active during reception of the entire frame (FWID must be programmed accordingly) to ensure the proper receive process, which requires at least 3 cycles after the frame complete to transfer the data into the receive buffer.
		0	External clock gating disabled.
		1	External clock gating enable.
14	PPCONNECT		Pair to pair connection. When set the DXENO pin is always set to 0, regardless of the frame boundary, setting the tree state buffer as output.
		0	non Pair-to-pair connection. The DX pin will go to high-impedance state when there is no frame to transmit.
		1	Pair-to-pair connection. When set, the DXENO pin is always set to 0, regardless of the frame boundary, setting the tree state buffer as output. This means the DX pin will be driven outside valid frame window. In that case, data sent by McBSP module during inactive channel are not guaranteed.
13-12	DXENDLY		When McBSPi.McBSP_SPCR1_REG[7] DXENA bit is set to 1, this field selects the added delay as follow:
		0	8 ns
		1h	14 ns (default)
		2h	20 ns
		3h	28 ns
11	XFULL_CYCLE		Transmit full cycle mode select:
		0	McBSP module operates in transmit half-cycle mode (transmit frame synchronization is sampled by the opposite edge of the clock used to drive transmit data).
		1	McBSP module operates in transmit full-cycle mode (transmit frame synchronization is sampled by the same edge of the clock used to drive transmit data).
10-6	Reserved	0	Reserved.
5	DLB		Digital Loopback. When this bit is set the transmit FSX, CLKX and DX are connected to FSR, CLKR, DR. The outputs of the McBSP (CLKR/CLKX, FSR/FSX, DX are not enabled)
		0	No DLB.
		1	DLB.
4	Reserved	0	Reserved

Table 15-67. McBSP_XCCR_REG Field Descriptions (continued)

Bit	Field	Value	Description
3	XDMAEN		Transmit DMA Enable bit. When cleared to 0 this bit will gate the external transmit DMA request, without resetting the DMA state machine. It is recommended to change this bit value only during transmit reset.
		0	Will gate the external transmit DMA request.
		1	Will NOT gate the external transmit DMA request.
2-1	Reserved	0	Reserved
0	XDISABLE		Transmit Disable bit. When this bit is set the transmit process will stop at the next frame boundary.
		0	The transmit process will NOT stop at the next frame boundary.
		1	The transmit process will stop at the next frame boundary.

15.3.49 McBSP Receive Configuration Control Register (RCCR_REG)

The McBSP_RCCR_REG register is shown in Figure 15-81 and described in Table 15-68.

Figure 15-81. McBSP_RCCR_REG

31	12	11	10	4	3	2	1	0
Reserved		RFULL_CYCLE	Reserved		RDMAEN	Reserved		RDISABLE
R-0		R/W-1	R-0		R/W-1	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 15-68. McBSP_RCCR_REG Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11	RFULL_CYCLE		Receive full cycle mode select:
		0	McBSP module operates in receive half-cycle mode (receive frame synchronization is sampled by the opposite edge of the clock used to sample receive data).
		1	McBSP module operates in receive full-cycle mode (receive frame synchronization is sampled by the same edge of the clock used to sample receive data).
10-4	Reserved	0	Reserved
3	RDMAEN		Receive DMA Enable bit. When cleared to 0 this bit will gate the external transmit DMA request, without resetting the DMA state machine. It is recommended to change this bit value only during receive reset.
		0	When cleared to 0 this bit will gate the external transmit DMA request.
		1	When set to 1 this bit will NOT gate the external transmit DMA request.
2-1	Reserved	0	Reserved
0	RDISABLE		Receive Disable bit. When this bit is set the receive process will stop at the next frame boundary.
		0	The receive process will NOT stop at the next frame boundary.
		1	When this bit is set the receive process will stop at the next frame boundary.

15.3.50 McBSP Transmit Buffer Status Registers (XBUFFSTAT_REG)

The McBSP_XBUFFSTAT_REG registers are shown in Figure 15-82 and described in Table 15-69.

Figure 15-82. McBSP_XBUFFSTAT_REG

31	8	7	0
Reserved		XBUFFSTAT	
R-0		R-80h	

LEGEND: R = Read only; -*n* = value after reset

Table 15-69. McBSP_XBUFFSTAT_REG Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	XBUFFSTAT	80h	Transmit buffer status (indicates the number of free locations inside transmit buffer). The XBUFFSTAT value reflects the buffer status on the OCP clock domain and it can be smaller than the real number of the free locations which are seen by the transmit state machine.

15.3.51 McBSP Receive Buffer Status Registers (RBUFFSTAT_REG)

The McBSP_RBUFFSTAT_REG registers are shown in Figure 15-83 and described in Table 15-70.

Figure 15-83. McBSP_RBUFFSTAT_REG

31	8	7	0
Reserved		RBUFFSTAT	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 15-70. McBSP_RBUFFSTAT_REG Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7-0	RBUFFSTAT	0	Receive Buffer Status (indicates the number of occupied locations inside receive buffer). The RBUFFSTAT value reflects the buffer status on the OCP clock domain and it can be smaller than the real number of the occupied locations which are seen by the receive state machine.

15.3.52 McBSP_STATUS_REG

The McBSP_STATUS_REG register is shown in Figure 15-84 and described in Table 15-71.

Figure 15-84. McBSP_STATUS_REG

31	1	0
Reserved		CLKMUXSTATUS
R-0		R-0

LEGEND: R = Read only; -*n* = value after reset

Table 15-71. McBSP_STATUS_REG Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	CLKMUXSTATUS		Indicates that the McBSP AUDIOBUFFER clock muxing is done after exiting SmartIdle mode. When this bit is set one the response to a different register access is delayed until the muxing process is done. In order to avoid such a situation pooling can be performed to status register in order to evaluate when McBSP is ready. Note that this information is relevant only for the McBSP having AUDIOBUFFER.

Serial Port Interface (SPI)

This chapter describes the master/slave multichannel serial port interface (SPI) module.

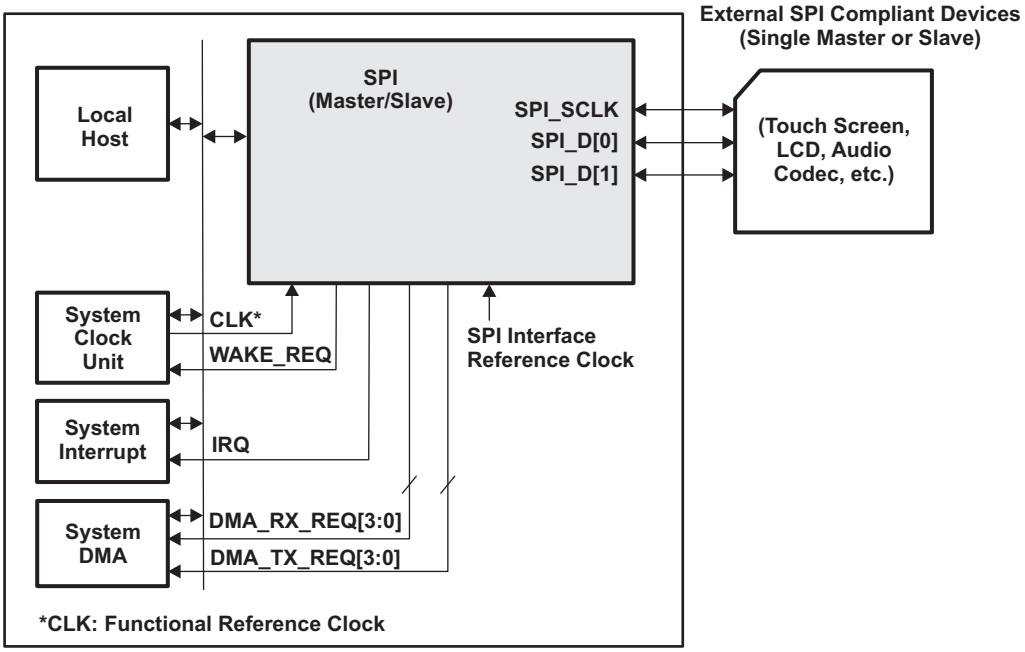
Topic **Page**

16.1 Introduction

16.1.1 Overview

SPI is a general-purpose receive/transmit master/slave controller that can interface with up to four slave external devices or one single external master. It allows a duplex, synchronous, serial communication between a CPU and SPI compliant external devices (slaves and masters). Figure 16-1 shows a high-level overview of a SPI system.

Figure 16-1. SPI System Overview



16.1.2 Features

The SPI includes these distinctive features:

- Serial clock with programmable frequency, polarity and phase
- It supports maximum frequency of 48 MHz.
- Clock frequency granularity can be changed: power of two or one single clock cycle.
- SPI enable generation programmable
- SPI enable with programmable polarity
- Wide selection of SPI word lengths ranging from 4 bits to 32 bits
- Master / Slave modes
- Master multichannel mode:
  - SPI interface provides supports for up to four SPI channels
  - SPI word Transmit / Receive slot assignment based on round robin arbitration
  - SPI configuration per channel (clock definition, enable polarity and word width)
  - Full duplex/half duplex
  - Transmit-only/receive-only/transmit-and-receive modes
  - Flexible I/O port controls per channel
  - Programmable clock granularity
- Independent DMA requests (read/write) per channel

- Single interrupt line for multiple interrupt source events
- 32-bit-wide bus, for system configuration and data transfer
- Programmable delay before the first SPI word transmitted
- No dead cycle between two successive words in slave mode
- Multiple SPI word access with a channel using a FIFO enabled
- Programmable timing control between chip select and external clock generation
- Support DMA providing 256 bit aligned addresses when FIFO is used
- Built-in FIFO available for a single channel. FIFO size of up to 64 bytes is supported.

16.2 Architecture

16.2.1 SPI Interface

This section lists the name and description of the SPI interface used for connection to external SPI compliant devices. SPI has a total of seven external SPI signals shown in Table 16-1.

Table 16-1. SPI Interface Pins

Pin	Type	Description
SPI_SCLK	I/O	SPI serial clock (output when master, input when slave)
SPI_D[0]	I/O	Can be configured as either input or output (MOSI or MISO)
SPI_D[1]	I/O	Can be configured as either input or output (MOSI or MISO)
$\overline{\text{SPI\_SCS[0]}}$	I/O	SPI chip select 0 output when master, input when slave (active low)
$\overline{\text{SPI\_SCS[1]}}$	I/O	SPI chip select 1 output when master, input when slave (active low)
$\overline{\text{SPI\_SCS[2]}}$	I/O	SPI chip select 2 output when master, input when slave (active low)
$\overline{\text{SPI\_SCS[3]}}$	I/O	SPI chip select 3 output when master, input when slave (active low)

16.2.2 SPI Transmission

This section describes the transmissions supported by SPI. The SPI protocol is a synchronous protocol that allows a master device to initiate serial communication with a slave device. Data is exchanged between these devices. A slave select line ($\overline{\text{SPI\_SCS[n]}}$) can be used to allow selection of an individual slave SPI device. Slave devices that are not selected do not interfere with SPI bus activities. Connected to multiple external devices, SPI exchanges data with a single SPI device at a time through two main modes:

- Two data pins interface mode. (See Section 16.2.2.1)
- Single data pin interface mode (recommended for half-duplex transmission). (See Section 16.2.2.2)

The flexibility of SPI allows exchanging data with several formats through programmable parameters described in Section 16.2.2.3.

16.2.2.1 Two Data Pins Interface Mode

The two data pins interface mode, allows a full duplex SPI transmission where data is transmitted (shifted out serially) and received (shifted in serially) simultaneously on separate data lines SPI_D[0] and SPI_D[1]. Data leaving the master exits on transmit serial data line also known as MOSI: MasterOutSlaveIn. Data leaving the slave exits on the receive data line also known as MISO: MasterInSlaveOut.

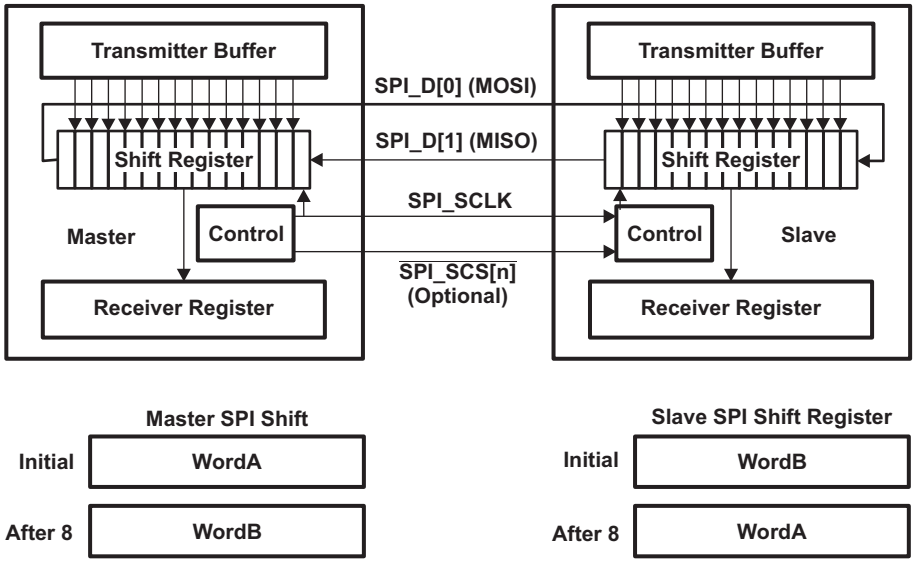
SPI has a unified SPI port control: SPI_D[1:0] can be independently configured as receive or transmit lines. The user has the responsibility to program which data line to use and in which direction (receive or transmit), according to the external slave/master connection.

The serial clock (SPI_SCLK) synchronizes shifting and sampling of the information on the two serial data lines (SPI_D[1:0]). Each time a bit is transferred out from the Master, one bit is transferred in from Slave.

Figure 16-2 shows an example of a full duplex system with a Master device on the left and a Slave device on the right. After 8 cycles of the serial clock SPI_SCLK, the WordA has been transferred from the master to the slave. At the same time, the WordB has been transferred from the slave to the master.

When referring to the master device, the control block transmits the clock SPI_SCLK and the enable signal ($\overline{\text{SPI\_SCS[n]}}$) (optional depends on MCSPI_MODULECTRL[PIN34] bit field).

Figure 16-2. SPI Full-Duplex Transmission



16.2.2.2 Single Data Pin Interface Mode

In single data pin interface mode, under software control, a single data line is used to alternatively transmit and receive data (Half duplex transmission).

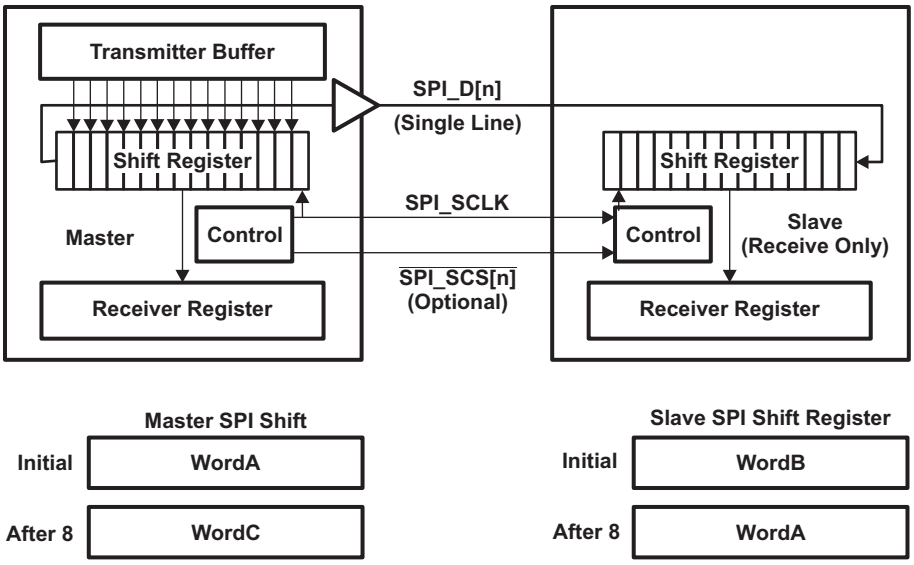
SPI has a unified SPI port control: SPI_D[1:0] can be independently configured as receive or transmit lines. The user has the responsibility to program which data line to use and in which direction (receive or transmit), according to the external slave/master connection.

As for a full duplex transmission, the serial clock (SPI_SCLK) synchronizes shifting and sampling of the information on the single serial data line.

16.2.2.2.1 Example With a Receive-Only Slave

Figure 16-3 shows a half duplex system with a Master device on the left and a receive-only Slave device on the right. Each time a bit is transferred out from the Master, one bit is transferred in the Slave. After 8 cycles of the serial clock SPI_SCLK, the WordA has been transferred from the master to the slave.

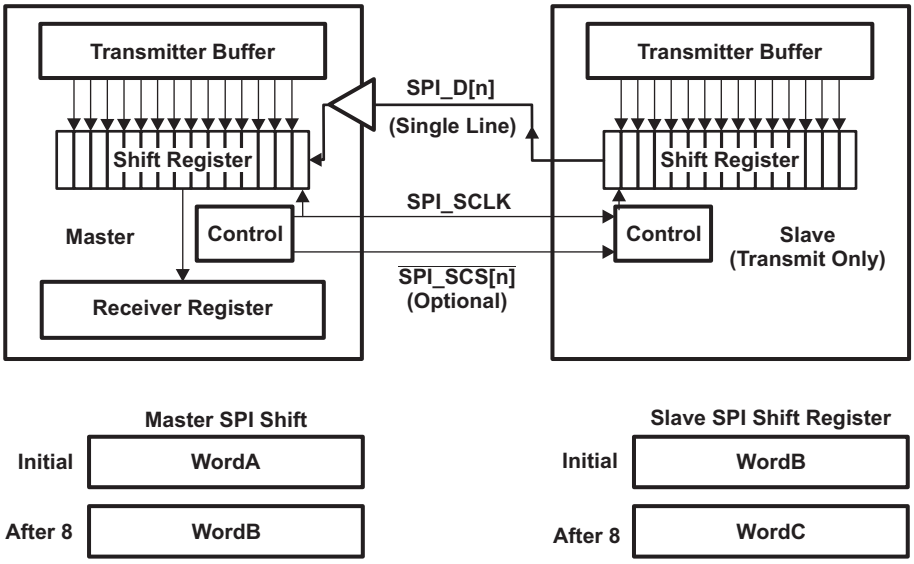
Figure 16-3. SPI Half-Duplex Transmission (Receive-only Slave)



16.2.2.2.2 Example With a Transmit-Only Slave

Figure 16-4 shows a half duplex system with a Master device on the left and a transmit-only Slave device on the right. Each time a bit is transferred out from the Slave, one bit is transferred in the Master. After 8 cycles of the serial clock SPI_SCLK, the WordA has been transferred from the slave to the master.

Figure 16-4. SPI Half-Duplex Transmission (Transmit-Only Slave)



16.2.2.3 Transfer Formats

This section describes the transfer formats supported by SPI.

The flexibility of SPI allows setting the parameters of the SPI transfer:

- SPI word length
- SPI enable generation programmable
- SPI enable assertion
- SPI enable polarity
- SPI clock frequency
- SPI clock phase
- SPI clock polarity

The consistency between SPI word length, clock phase and clock polarity of the master SPI device and the communicating slave device remains under software responsibility.

16.2.2.3.1 Programmable Word Length

SPI supports any SPI word from 4 to 32 bits long.

The SPI word length can be changed between transmissions to allow a master device to communicate with peripheral slaves having different requirements.

16.2.2.3.2 Programmable SPI Enable Generation

SPI is able to generate or not the SPI enable, if management of chip select is de-asserted a point to point connection is mandatory. Only a single master of slave device can be connected to the SPI bus.

16.2.2.3.3 Programmable SPI Enable ($\overline{\text{SPI\_SCS[n]}}$)

The polarity of the $\overline{\text{SPI\_SCS[n]}}$ signals is programmable. $\overline{\text{SPI\_SCS[n]}}$ signals can be active high or low.

The assertion of the $\overline{\text{SPI\_SCS[n]}}$ signals is programmable: $\overline{\text{SPI\_SCS[n]}}$ signals can be manually asserted or can be automatically asserted.

Two consecutive words for two different slave devices may go along with active $\overline{\text{SPI\_SCS[n]}}$ signals with different polarity.

16.2.2.3.4 Programmable SPI Clock (SPI_SCLK)

The phase and the polarity of the SPI serial clock are programmable when SPI is a SPI master device or a SPI slave device. The baud rate of the SPI serial clock is programmable when it is a SPI master.

When SPI is operating as a slave, the serial clock SPI_SCLK is an input from the master.

16.2.2.3.5 Bit Rate

In Master Mode, an internal reference clock CLKSPIREF is used as an input of a programmable divider to generate bit rate of the serial clock SPI_SCLK. Granularity of this clock divider can be changed.

16.2.2.3.6 Polarity and Phase

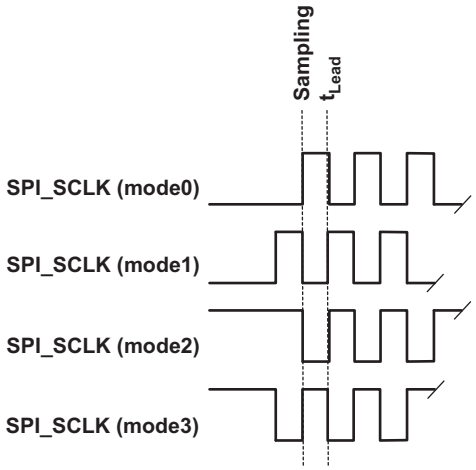
SPI supports four sub-modes of the SPI format transfer that depend on the polarity (POL) and the phase (PHA) of the SPI serial clock (SPI_SCLK). Table 16-2 and Figure 16-5 show a summary of the four sub-modes. Software selects one of four combinations of serial clock phase and polarity.

Two consecutive SPI words for two different slave devices may go along with active SPI_SCLK signal with different phase and polarity.

Table 16-2. Phase and Polarity Combinations

Polarity (POL)	Phase (PHA)	SPI Mode	Comments
0	0	mode0	SPI_SCLK active high and sampling occurs on the rising edge.
0	1	mode1	SPI_SCLK active high and sampling occurs on the falling edge.
1	0	mode2	SPI_SCLK active low and sampling occurs on the falling edge.
1	1	mode3	SPI_SCLK active low and sampling occurs on the rising edge.

Figure 16-5. Phase and Polarity Combinations



16.2.2.3.7 Transfer Format With PHA = 0

This section describes the concept of a SPI transmission with the SPI mode0 and the SPI mode2.

In the transfer format with PHA = 0, $\overline{\text{SPI\_SCS[n]}}$ is activated a half cycle of SPI_SCLK ahead of the first SPI_SCLK edge.

In both master and slave modes, SPI drives the data lines at the time of $\overline{\text{SPI\_SCS[n]}}$ is asserted.

Each data frame is transmitted starting with the MSB. At the extremity of both SPI data lines , the first bit of SPI word is valid a half cycle of SPI_SCLK after the $\overline{\text{SPI\_SCS[n]}}$ assertion.

Therefore, the first edge of the SPI_SCLK line is used by the master to sample the first data bit sent by the slave. On the same edge, the first data bit sent by the master is sampled by the slave.

On the next SPI_SCLK edge, the received data bit is shifted into the shift register, and a new data bit is transmitted on the serial data line.

This process continues for a total of pulses on the SPI_SCLK line defined by the SPI word length programmed in the master device, with data being latched on odd numbered edges and shifted on even numbered edges.

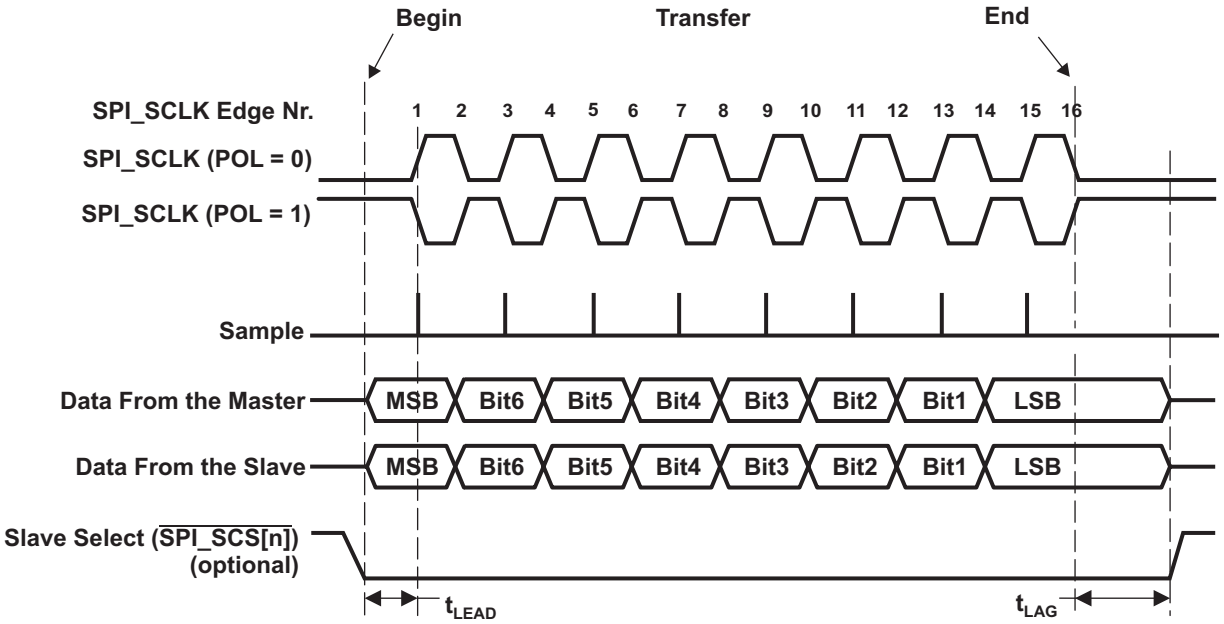
Figure 16-6 is a timing diagram of a SPI transfer for the SPI mode0 and the SPI mode2, when SPI is master or slave, with the frequency of SPI_SCLK equals to the frequency of CLKSPIREF. It should not be used as a replacement for SPI timing information and requirements.

When SPI is in slave mode, if the $\overline{\text{SPI\_SCS[n]}}$ line is not de-asserted between successive transmissions then the content of the Transmitter register is not transmitted, instead the last received SPI word is transmitted.

In master mode, the $\overline{\text{SPI\_SCS[n]}}$ line must be negated and reasserted between each successive SPI word. This is because the slave select pin freezes the data in its shift register and does not allow it to be altered if PHA bit equals 0.

In 3-pin mode without using the $\overline{\text{SPI\_SCS[n]}}$ signal, the controller provides the same waveform but with $\overline{\text{SPI\_SCS[n]}}$ forced to low state. In this mode, $\overline{\text{SPI\_SCS[n]}}$ is useless.

Figure 16-6. Full Duplex Single Transfer Format with PHA = 0



16.2.2.3.8 Transfer Format With PHA = 1

This section describes SPI full duplex transmission with the SPI mode1 and the SPI mode 3.

In the transfer format with PHA = 1, $\overline{\text{SPI\_SCS[n]}}$ is activated a delay (t_{Lead}) ahead of the first SPI_SCLK edge.

In both master and slave modes, SPI drives the data lines on the first SPI_SCLK edge.

Each data frame is transmitted starting with the MSB. At the extremity of both SPI data lines , the first bit of SPI word is valid on the next SPI_SCLK edge, a half cycle of SPI_SCLK later. It is the sampling edge for both the master and slave.

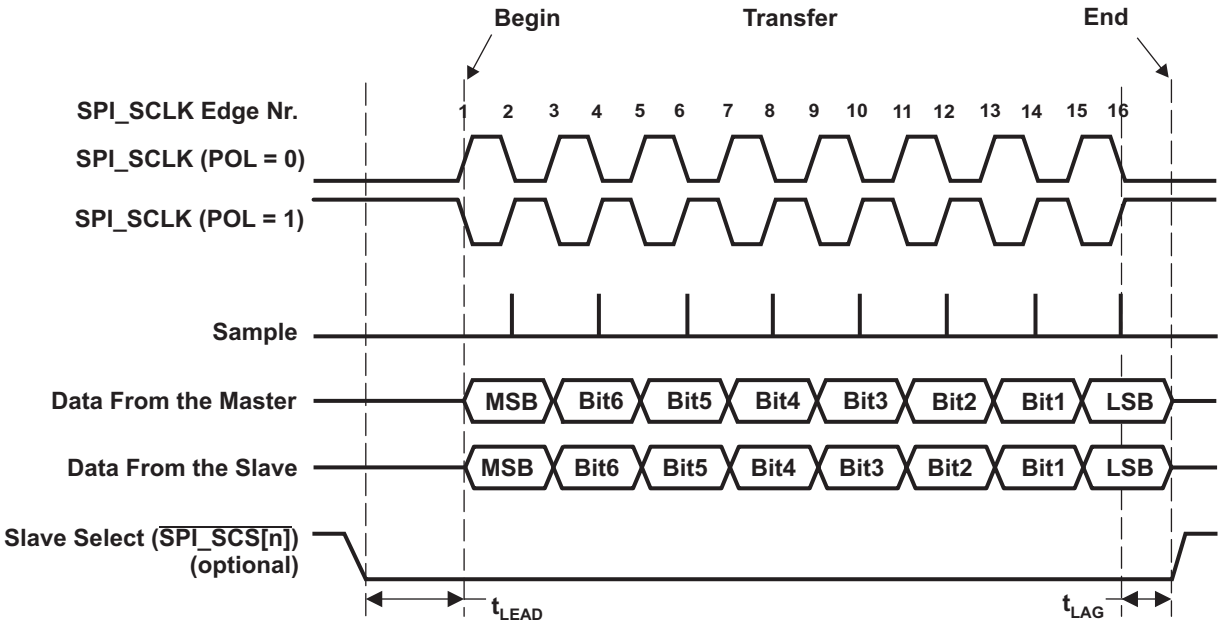
When the third edge occurs, the received data bit is shifted into the shift register. The next data bit of the master is provided to the serial input pin of the slave.

This process continues for a total of pulses on the SPI_SCLK line defined by the word length programmed in the master device, with data being latched on even numbered edges and shifted on odd numbered edges.

Figure 16-7 is a timing diagram of a SPI transfer for the SPI mode1 and the SPI mode3, when SPI is master or slave, with the frequency of SPI_SCLK equals to the frequency of CLKSPIREF. It should not be used as a replacement for SPI timing information and requirements.

The $\overline{\text{SPI\_SCS[n]}}$ line may remain active between successive transfers. In 3-pin mode without using the $\overline{\text{SPI\_SCS[n]}}$ signal, the controller provides the same waveform but with $\overline{\text{SPI\_SCS[n]}}$ forced to low state. In this mode, $\overline{\text{SPI\_SCS[n]}}$ is useless.

Figure 16-7. Full Duplex Single Transfer Format With PHA = 1



16.2.3 Master Mode

SPI is in master mode when the MS bit of the register MCSPI_MODULCTRL is cleared.

In master mode SPI supports multi-channel communication with up to four independent SPI communication channel contexts. SPI initiates a data transfer on the data lines (SPI_D[1:0]) and generates clock (SPI_SCLK) and control signals ($\overline{\text{SPI\_SCS[n]}}$) to a single SPI slave device at a time.

16.2.3.1 Dedicated Resources Per Channel

In the following sections, the letter “I” indicates the channel number that can be 0, 1, 2, or 3. Each channel has the following dedicated resources:

- Its own channel enable, programmable with the bit EN of the register MCSPI_CH(I)CTRL. Disabling the channel, outside data word transmission, remains under user responsibility.
- Its own transmitter register MCSPI_TX on top of the common shift register. If the transmitter register is empty, the status bit TXS of the register MCSPI_CH(I)STAT is set.
- Its own receiver register MCSPI_RX on top of the common shift register. If the receiver register is full, the status bit RXS of the register MCSPI_CH(I)STAT is set.
- A fixed SPI ENABLE line allocation ($\overline{\text{SPI\_SCS[I]}}$ port for channel I), SPI enable management is optional.
- Its own communication configuration with the following parameters via the register (I)CONF
  - Transmit/Receive modes, programmable with the bit TRM.
  - Interface mode (Two data pins or Single data pin) and data pins assignment, both programmable with the bits IS and DPE.
  - SPI word length, programmable with the bits WL.
  - SPI_SCS[n] polarity, programmable with the bit EPOL.
  - SPI_SCS[n] kept active between words, programmable with the bit FORCE.
  - Turbo mode, programmable with the bit TURBO.
  - SPI_SCLK frequency, programmable with the bit CLKD, the granularity of clock division can be changed using CLKG bit, the clock ratio is then concatenated with MCSPI_CHCTRL[EXTCLK] value.
  - SPI_SCLK polarity, programmable with the bit POL
  - SPI_SCLK phase, programmable with the bit PHA.
  - Start bit extension enable, programmable with the bit SBE to support LoSSI transfer specification.
  - Start bit polarity, programmable with the bit SBPOL
  - Use a FIFO Buffer or not( ), programmable with FFER and FFEW, depending on transfer mode TRM.
- Two DMA requests events, read and write, to synchronize read/write accesses of the DMA controller with the activity of SPI. The DMA requests are enabled with the bits DMAR and DMAW.
- Three interrupts events

The transfers will use the latest loaded parameters of the register (I)CONF.

The configuration parameters SPI_SCS[n] polarity, Turbo mode, SPI_SCLK phase and SPI_SCLK polarity can be loaded in the (I)CONF register only when the channel is disabled. The user has the responsibility to change the other parameters of the (I)CONF register when no transfer occurs on the SPI interface.

16.2.3.2 Interrupt Events in Master Mode

In master mode, the interrupt events related to the transmitter register state are TX_empty and TX_underflow. The interrupt event related to the receiver register state is RX_full.

16.2.3.2.1 TX_empty

The event TX_empty is activated when a channel is enabled and its transmitter register becomes empty (transient event). Enabling channel automatically raises this event, except for the Master receive only mode. (See Section 16.2.3.5). When FIFO buffer is enabled (MCSPI_CH(I)CONF[FFEW] set to 1), the TX_empty is asserted as soon as there is enough space in buffer to write a number of byte defined by MCSPI_XFERLEVEL[AEL].

Transmitter register must be loaded to remove the source of the interrupt and the TX_empty interrupt status bit must be cleared for interrupt line de-assertion (if event enabled as interrupt source) . (See Section 16.2.5).

When FIFO is enabled, no new TX_empty event will be asserted as soon as CPU has not performed the number of write into transmitter register defined by MCSPI_XFERLEVEL[AEL]. It is the responsibility of CPU to perform the right number of writes.

16.2.3.2.2 TX_underflow

The event TX_underflow is activated when the channel is enabled and if the transmitter register or FIFO is empty (not updated with new data) at the time of shift register assignment.

The TX_underflow is a harmless warning in master mode.

To avoid having TX_underflow event at the beginning of a transmission, the event TX_underflow is not activated when no data has been loaded into the transmitter register since channel has been enabled.

To avoid having TX_underflow event the Transmitter register must be loaded seldom.

TX_underflow interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

Note: Be careful, when more than one channel have a FIFO enable bit field FFER or FFEW set, the module force no use of FIFO, software must take care that only one enabled channel is configured to use the FIFO buffer.

16.2.3.2.3 RX_ full

The event RX_full is activated when channel is enabled and receiver register becomes filled (transient event). When FIFO buffer is enabled (MCSPI_CH(I)CONF[FFER] set to 1), the RX_full is asserted as soon as there is a number of bytes holds in buffer to read defined by MCSPI_XFERLEVEL[AFL].

Receiver register must be read to remove source of interrupt and RX_full interrupt status bit must be cleared for interrupt line de-assertion (if event enabled as interrupt source).

When FIFO is enabled, no new RX_full event will be asserted as soon as CPU has not performed the number of read into receive register defined by MCSPI_XFERLEVEL[AFL]. It is the responsibility of CPU to perform the right number of reads.

16.2.3.2.4 End of Word Count

The event end of word count (EOW) is activated when channel is enabled and configured to use the built-in FIFO. This interrupt is raised when the controller had performed the number of transfer defined in MCSPI_XFERLEVEL[WCNT] register. If the value was programmed to 0000h, the counter is not enabled and this interrupt is not generated.

The End of Word count interrupt also indicates that the SPI transfer is halt on channel using the FIFO buffer as soon as MCSPI_XFERLEVEL[WCNT] is not reloaded and channel re-enabled.

End of Word interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

16.2.3.3 Master Transmit and Receive Mode

This mode is programmable per channel (bits TRM of the register (I)CONF).

The channel access to the shift registers, for transmission/reception, is based on its transmitter and receiver register state and round robin arbitration.

The channel that meets the rules below is included in the round robin list of already active channels scheduled for transmission and/or reception. The arbiter skips the channel that does not meet the rules and search for the next following enabled channel, in rotation.

Rule 1: Only enabled channels (bit EN of the register MCSPI_CH(I)CTRL), can be scheduled for transmission and/or reception.

Rule 2: An enabled channel can be scheduled if its transmitter register is not empty (bit TXS of the register MCSPI_CH(I)STAT) or its FIFO is not empty in case of buffer use for the corresponding channel (bit FFE of the register MCSPI_CH(I)STAT), that is updated with new data, at the time of shift register assignment. If the transmitter register or FIFO is empty, at the time of shift register assignment, the event TX_underflow is activated and the next enabled channel with new data to transmit is scheduled. (See also transmit only mode).

Rule 3: An enabled channel can be scheduled if its receive register is not full (bit RXS of the register MCSPI_CH(I)STAT)) or its FIFO is not full in case of buffer use for the corresponding channel (bit FFF of the register MCSPI_CH(I)STAT) at the time of shift register assignment. (See also receive only mode). Therefore the receiver register of FIFO cannot be overwritten. The RX_overflow bit, in the MCSPI_IRQSTATUS register is never set in this mode.

On completion of SPI word transfer (bit EOT of the register MCSPI_CH(I)STAT is set) the updated transmitter register of the next scheduled channel is loaded into the shift register. This bit is meaningless when using the Buffer for this channel. The serialization (transmit and receive) starts according to the channel communication configuration. On serialization completion the received data is transferred to the channel receive register.

The built-in FIFO is available in this mode and can be configured in one data direction Transmit or Receive, then the FIFO is seen as a unique FFNBYTE bytes buffer, or it can also be configured in both data direction Transmit and Receive, then the FIFO is split into two separate FFNBYTE/2 bytes buffer with their own address space management, in this last case the definition of AEL and AFL levels is based on FFNBYTE/2 bytes and is under CPU responsibility.

16.2.3.4 Master Transmit-Only Mode

This mode avoids the CPU to read the receiver register (minimizing data movement) when only transmission is meaningful.

The master transmit only mode is programmable per channel (bits TRM of the register (I)CONF).

In master transmit only mode, transmission starts after data is loaded into the transmitter register.

Rule 1 and **Rule 2**, defined in Section 16.2.3.3, are applicable in this mode.

Rule 3, defined in Section 16.2.3.3, is not applicable: In master transmit only mode, the receiver register or FIFO state “full” does not prevent transmission, and the receiver register is always overwritten with the new SPI word. This event in the receiver register is not significant when only transmission is meaningful. So, the RX_overflow bit, in the MCSPI_IRQSTATUS register is never set in this mode.

The SPI module automatically disables the RX_full interrupt status. The corresponding interrupt request and DMA Read request are not generated in master transmit only mode.

The status of the serialization completion is given by the bit EOT of the register MCSPI_CH(I)STAT. This bit is meaningless when using the Buffer for this channel.

The built-in FIFO is available in this mode and can be configured with FFEW bit field in the MCSPI_CH(I)CONF register, then the FIFO is seen as a unique FFNBYTE bytes buffer.

16.2.3.5 Master Receive-Only Mode

This mode avoids the CPU to refill the transmitter register (minimizing data movement) when only reception is meaningful.

The master receive mode is programmable per channel (bits TRM of the register (I)CONF).

The master receive only mode enables channel scheduling only on empty state of the receiver register.

Rule 1 and **Rule 3**, defined in Section 16.2.3.3, are applicable in this mode.

Rule 2, defined in Section 16.2.3.3, is not applicable: In master receive only mode, after the first loading of the transmitter register of the enabled channel, the transmitter register state is maintained as full. The content of the transmitter register is always loaded into the shift register, at the time of shift register assignment. So, after the first loading of the transmitter register, the bits TX_empty and TX_underflow, in the MCSPI_IRQSTATUS register are never set in this mode.

The status of the serialization completion is given by the bit EOT of the register MCSPI_CH(I)STAT. The bit RX_full in the MCSPI_IRQSTATUS register is set when a received data is loaded from the shift register to the receiver register. This bit is meaningless when using the Buffer for this channel.

The built-in FIFO is available in this mode and can be configured with FFER bit field in the MCSPI_CH(I)CONF register, then the FIFO is seen as a unique FFNBYTE bytes buffer.

16.2.3.6 Single-Channel Master Mode

When the SPI is configured as a master device with a single enabled channel, the assertion of the SPIM_CSX signal can be controlled in two different ways:

- In 3 pin mode : MCSPI_MODULCTRL[1] PIN34 and MCSPI_MODULCTRL[0] SINGLE bits are set to 1, the controller transmit SPI word as soon as transmit register or FIFO is not empty.
- In 4 pin mode : MCSPI_MODULCTRL[1] PIN34 bit is cleared to 0 and MCSPI_MODULCTRL[0] SINGLE bit is set to 1, SPI_SCS[n] assertion/deassertion controlled by Software. (See Section 16.2.3.6.1) using the MCSPI_CHxCONF[20] FORCE bit.

16.2.3.6.1 Programming Tips When Switching to Another Channel

When a single channel is enabled and data transfer is ongoing:

- Wait for completion of the SPI word transfer (bit EOT of the register MCSPI_CH(I)STAT is set) before disabling the current channel and enabling a different channel.
- Disable the current channel first, and then enable the other channel.

16.2.3.6.2 Keep $\overline{\text{SPI\_SCS[n]}}$ Active Mode (Force $\overline{\text{SPI\_SCS[n]}}$)

Continuous transfers are manually allowed by keeping the $\overline{\text{SPI\_SCS[n]}}$ signal active for successive SPI words transfer. Several sequences (configuration/enable/disable of the channel) can be run without deactivating the $\overline{\text{SPI\_SCS[n]}}$ line. This mode is supported by all channels and any master sequence can be used (transmit-receive, transmit-only, receive-only).

Keeping the $\overline{\text{SPI\_SCS[n]}}$ active mode is supported when:

- A single channel is used (MCSPI_MODULCTRL[Single] bit is set to 1).
- Transfer parameters of the transfer are loaded in the configuration register (MCSPI_CH(I)CONF) in the appropriate channel.

  The state of the $\overline{\text{SPI\_SCS[n]}}$ signal is programmable.
  - Writing 1 into the bit FORCE of the register MCSPI_CH(I)CONF drives high the $\overline{\text{SPI\_SCS[n]}}$ line when MCSPI_CHCONF(I)[EPOL] is cleared to 0, and drives it low when MCSPI_CHCONF(I)[EPOL] is set.
  - Writing 0 into the bit FORCE of the register MCSPI_CH(I)CONF drives low the $\overline{\text{SPI\_SCS[n]}}$ line when MCSPI_CHCONF(I)[EPOL] is cleared to 0, and drives it high when MCSPI_CHCONF(I)[EPOL] is set.
- A single channel is enabled (MCSPI_CH(I)CTRL[En] set to 1) . The first enabled channel activates the $\overline{\text{SPI\_SCS[n]}}$ line.

Once the channel is enabled, the SPI_SCS[n] signal is activated with the programmed polarity.

As in multi-channel master mode, the start of the transfer depends on the status of the transmitter register, the status of the receiver register and the mode defined by the bits TRM in the configuration register (transmit only, receive only or transmit and receive) of the enabled channel.

The status of the serialization completion of each SPI word is given by the bit EOT of the register MCSPI_CH(I)STAT. The bit RX_full in the MCSPI_IRQSTATUS register is set when a received data is loaded from the shift register to the receiver register.

A change in the configuration parameters is propagated directly on the SPI interface. If the SPI_SCS[n] signal is activated the user must insure that the configuration is changed only between SPI words, in order to avoid corrupting the current transfer.

NOTE: The SPI_SCS[n] polarity, the SPI_SCLK phase and SPI_SCLK polarity must not be modified when the SPI_SCS[n] signal is activated. The Transmit/Receive mode, programmable with the bit TRM can be modified only when the channel is disabled. The channel can be disabled and enabled while the SPI_SCS[n] signal is activated.

The delay between SPI words that requires the connected SPI slave device to switch from one configuration (Transmit only for instance) to another (receive only for instance) must be handled under software responsibility.

At the end of the last SPI word, the channel must be deactivated (MCSPI_CH(I)CTRL[En] is cleared to 0) and the SPI_SCS[n] can be forced to its inactive state (MCSPI_CH(I)CONF[Force]).

Figure 16-8 and Figure 16-9 show successive transfers with SPI_SCS[n] kept active low with a different configuration for each SPI word in respectively single data pin interface mode and two data pins interface mode. The arrows indicate when the channel is disabled before a change in the configuration parameters and enabled again.

Figure 16-8. Continuous Transfers With SPI_SCS[n] Maintained Active (Single-Data-Pin Interface Mode)

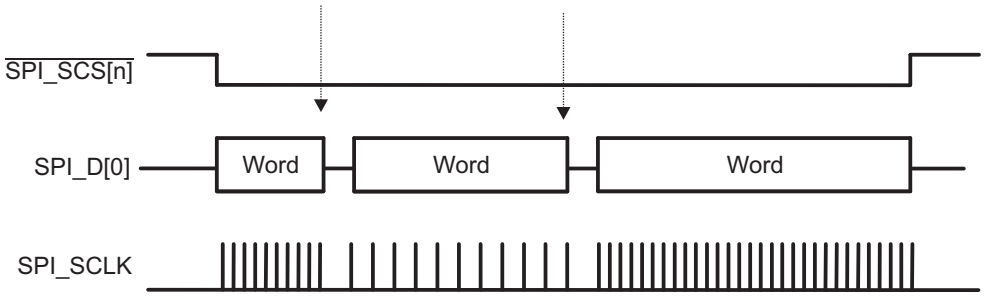
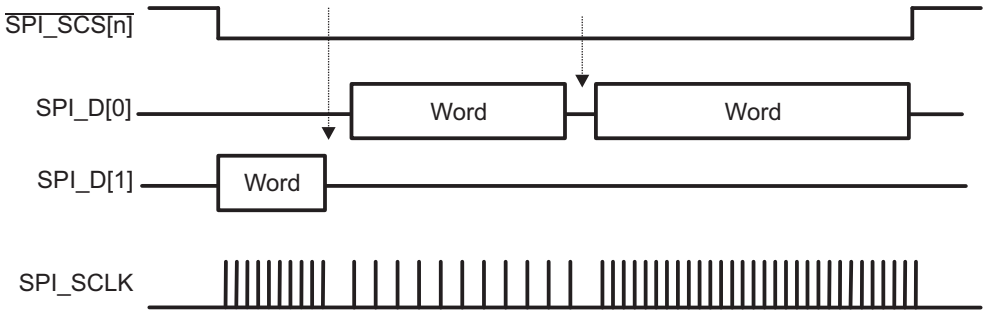


Figure 16-9. Continuous Transfers With SPI_SCS[n] Maintained Active (Dual-Data-Pin Interface Mode)



NOTE: The turbo mode is also supported for the Keep $\overline{\text{SPI\_SCS[n]}}$ active mode when the following conditions are met:

- A single channel will be explicitly used (MCSPI_MODULCTRL[Single] bit is set to 1).
- The turbo mode is enabled in the configuration of the channel (Turbo bit of the register (i)CONF).

16.2.3.6.3 Turbo Mode

The purpose of the Turbo mode is to improve the throughput of the SPI interface when a single channel is enabled, by allowing transfers until the shift register and the receiver register are full.

This mode is programmable per channel (bit Turbo of the register (I)CONF). When several channels are enabled, the bit Turbo of the registers MCSPI_CH(I)CONF has no effect, and the channel access to the shift registers remains as described in Section 16.2.3.3.

In Turbo mode, **Rule 1** and **Rule 2** defined in Section 16.2.3.3 are applicable, but Rule 3 is not applicable. An enabled channel can be scheduled if its receive register is full (bit RXS of the register MCSPI_CH(I)STAT) at the time of shift register assignment until the shift register is full.

The receiver register cannot be overwritten in turbo mode. In consequence the RX_overflow bit, in MCSPI_IRQSTATUS register is never set in this mode.

16.2.3.7 Start Bit Mode

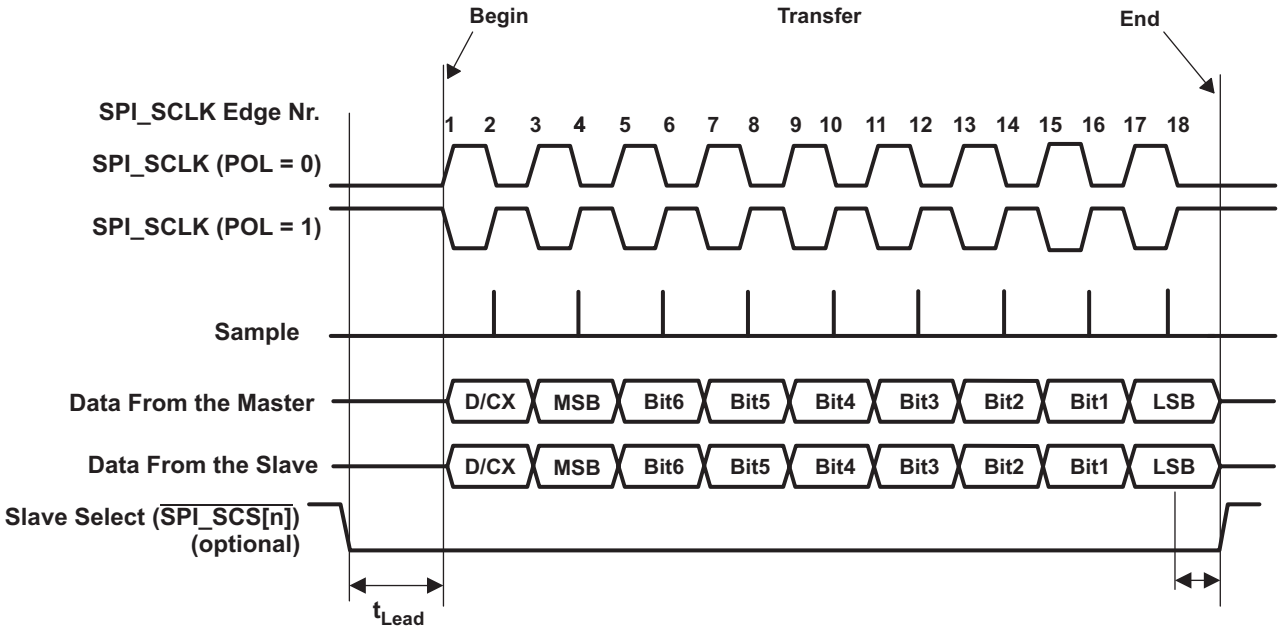
The purpose of the start bit mode is to add an extended bit before the SPI word transmission specified by word length WL (see Figure 16-10). This feature is only available in master mode. This mode is compliant with write command/data format as specified in LoSSI protocol.

This mode is programmable per channel (Start bit enable (SBE) bit of the register MCSPI_CH(I)CONF).

The polarity of the extended bit is programmable per channel and it indicates whether the next SPI word must be handled as a command when SBPOL is cleared to 0 or as a data or a parameter when SBPOL is set to 1. Moreover, start bit polarity SBPOL can be changed dynamically during start bit mode transfer without disabling the channel for reconfiguration, in this case you have the responsibility to configure the SBPOL bit before writing the SPI word to be transmitted in TX register.

The start bit mode could be used at the same time as turbo mode and/or manual chip select mode. In this case, only one channel could be used, no round-robin arbitration is possible.

Figure 16-10. Extended SPI Transfer With Start Bit PHA = 1

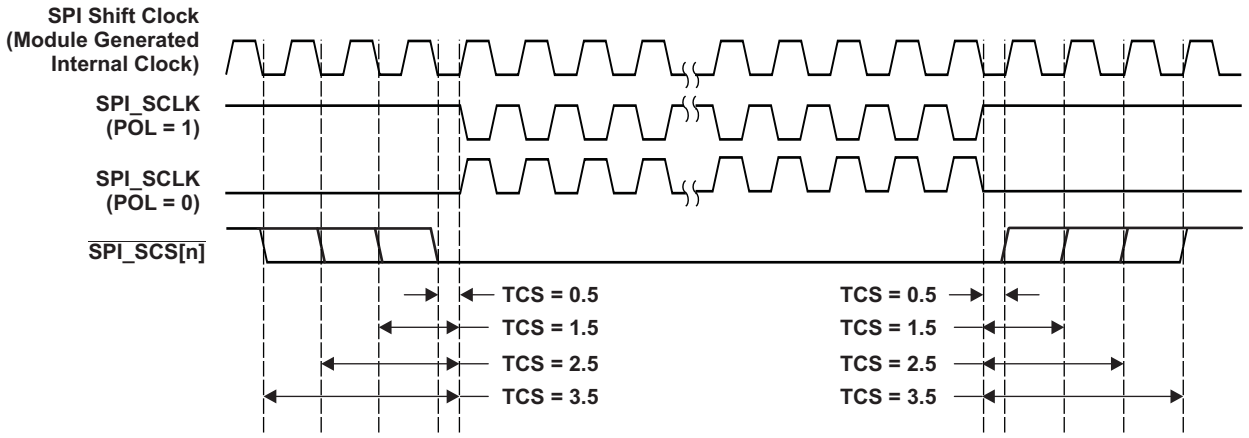


16.2.3.8 Chip-Select Timing Control

The chip select timing control is only available in master mode with automatic chip select generation (FORCE bit field is cleared to 0), to add a programmable delay between chip select assertion and first clock edge or chip select removal and last clock edge. The option is available only in 4 pin mode MCSPI_MODULCTRL[1] PIN34 is cleared to 0.

This mode is programmable per channel (TCS bit of the register MCSPI_CH(I)CONF). Figure 16-11 shows the chip-select SPI_SCS[n] timing control.

Figure 16-11. Chip-Select SPI_SCS[n] Timing Controls



NOTE: Because of the design implementation for transfers using a clock divider ratio set to 1 (clock bypassed), a half cycle must be added to the value between chip-select assertion and the first clock edge with PHA = 1 or between chip-select removal and the last clock edge with PHA = 0.

With an odd clock divider ratio which occurs when granularity is one clock cycle, that means that MCSPI_CH(I)CONF[CLKG] is set to 1 and MCSPI_CH(I)CONF[CLKD] has an even value, the clock duty cycle is not 50%, then one of the high level or low level duration is selected to be added to TCS delay.

Table 16-3 summarizes all delays between chip select and first (setup) or last (hold) clock edge.

In 3-pin mode this option is useless, the chip select SPI_SCS[n] is forced to low state.

T_ref = CLKSPIREF period in ns

F_{ratio} = SPI clock division ratio

Table 16-3. Chip Select ↔ Clock Edge Delay Depending on Configuration

Clock Ratio F_{ratio}	Clock Phase PHA	Chip Select ↔ Clock Edge Delay	
		Setup	Hold
1	0	T_ref × (TCS + ½)	T_ref × (TCS + 1)
	1	T_ref × (TCS + 1)	T_ref × (TCS + ½)
Even ≥ 2	x	T_ref × F_{ratio} × (TCS + ½)	T_ref × F_{ratio} × (TCS + ½)
Odd ≥ 3 (only with MCSPI_CH(I)CONF[CLKG] set to 1	0	T_ref × [{F_{ratio} × TCS) + (F_{ratio} + ½)]	T_ref × [{F_{ratio} × TCS) + (F_{ratio} + ½)]
	1	T_ref × [{F_{ratio} × TCS) + (F_{ratio} - ½)]	T_ref × [{F_{ratio} × TCS) + (F_{ratio} - ½)]

The clock divider ratio depends on divider granularity MCSPI_CH(I)CONF[CLKG]:

- MCSPI_CH(I)CONF[CLKG] = 0 : granularity is power of 2.
  $F_{ratio} = 2^{MCSPI\_CH(I)CONF[CLKD]}$
- MCSPI_CH(I)CONF[CLKG] = 1 : EXTCLK concatenated with CLKD + 1.
  F_{ratio} = MCSPI_CH(I)CNTRL[EXTCLK].MCSPI_CH(I)CONF[CLKD] + 1

16.2.3.9 Clock Ratio Granularity

By default the clock division ratio is defined by the register MCSPI_CH(I)CONF[CLKD] with power of 2 granularity leading to a clock division in range 1 to 32768, in this case the duty cycle is always 50%. With bit MCSPI_CH(I)CONF[CLKG] the clock division granularity can be changed to one clock cycle, in that case the register MCSPI_CH(I)CTRL[EXTCLK] is concatenated with MCSPI_CH(I)CONF[CLKD] to give a 12-bit width division ratio in range 1 to 4096.

When granularity is one clock cycle (MCSPI_CH(I)CONF[CLKG] set to 1), for odd value of clock ratio the clock duty cycle is kept to 50-50 using falling edge of clock reference CLKSPIREF.

Table 16-4. CLKSPIO High/Low Time Computation

Clock Ratio F_{ratio}	CLKSPIO High Time	CLKSPIO Low Time
1	T_{high}_ref	T_{low}_ref
Even ≥ 2	t_ref × (F_{ratio}/2)	t_ref × (F_{ratio}/2)
Odd ≥ 3	t_ref × (F_{ratio}/2)	t_ref × (F_{ratio}/2)

T_ref = CLKSPIREF period in ns. T_{high}_ref = CLKSPIREF high Time period in ns. T_{low}_ref = CLKSPIREF low Time period in ns. F_{ratio} = SPI clock division ratio

F_{ratio} = MCSPI_CH(I)CTRL[EXTCLK].MCSPI_CH(I)CONF[CLKD] + 1

For odd ratio value the duty cycle is calculated as:

Duty_cycle = ½

Granularity examples: With a clock source frequency of 48 MHz:

Table 16-5. Clock Granularity Examples

MCSPI_CH(I) CTRL	MCSPI_CH(I) CONF	MCSPI_CH(I) CONF		MCSPI_CH(I) CONF	MCSPI_CH(I) CONF	Thigh (ns)	Tlow (ns)	Tperiod (ns)	Duty Cycle	Fout (MHz)
EXTCLK	CLKD	CLKG	F_{ratio}	PHA	POL					
X	0	0	1	X	X	10.4	10.4	20.8	50-50	48
X	1	0	2	X	X	20.8	20.8	41.6	50-50	24
X	2	0	4	X	X	41.6	41.6	83.2	50-50	12
X	3	0	8	X	X	83.2	83.2	166.4	50-50	6
0	0	1	1	X	X	10.4	10.4	20.8	50-50	48
0	1	1	2	X	X	20.8	20.8	41.6	50-50	24
0	2	1	3	1	0	31,2	31,2	62.4	50-50	16
0	2	1	3	1	1	31,2	31,2	62.4	50-50	16
0	3	1	4	X	X	41.6	41.6	83.2	50-50	12
5	0	1	81	1	0	842,4	842,4	1684.8	50-50	0.592
5	7	1	88	X	X	915.2	915.2	1830.4	50-50	0.545

16.2.3.10 FIFO Buffer Management (Optional USEFIFO = 1)

The SPI controller has a built-in FFNBYTE bytes buffer in order to unload DMA or interrupt handler and improve data throughput. The instantiation of this Buffer is optional it depends on a generic parameter USEFIFO, the FIFO is instantiated when it is set to 1. Allowed FIFO depth up to 64 bytes is supported and is defined by generic parameter FFNBYTE. When the FIFO is not instantiated writes in registers MCSPI_XFERLEVEL, MCSPI_CH(I)CONF[FFER] and MCSPI_CH(I)CONF[FFEW] have no functional effect, nevertheless read back is allowed to check written value.

This buffer can be used by only one channel at once and is selected by setting MCSPI_CH(I)CONF[FFER] or MCSPI_CH(I)CONF[FFEW] to 1.

If several channel are selected and several FIFO enable bit fields set to 1, the controller forces buffer not to be used, it is the responsibility of the driver to set only one FIFO enable bit field.

The buffer can be used in the modes defined:

- Master or Slave mode.
- Transmit only, Receive only or Transmit/Receive mode.
- Single channel or turbo mode, or in normal round robin mode. In round robin mode the buffer is used by only one channel.
- Every word length MCSPI_CH(I)CONF[WL] are supported.

Two levels AEL and AFL located in MCSPI_XFERLEVEL register rule the buffer management. The granularity of these levels is one byte, then it is not aligned with SPI word length. It is the responsibility of the driver to set these values as a multiple of SPI word length defined in MCSPI_CH(I)CONF[WL]. The number of byte written in the FIFO depends on word length (see Table 16-6).

Table 16-6. FIFO Writes, Word Length Relationship

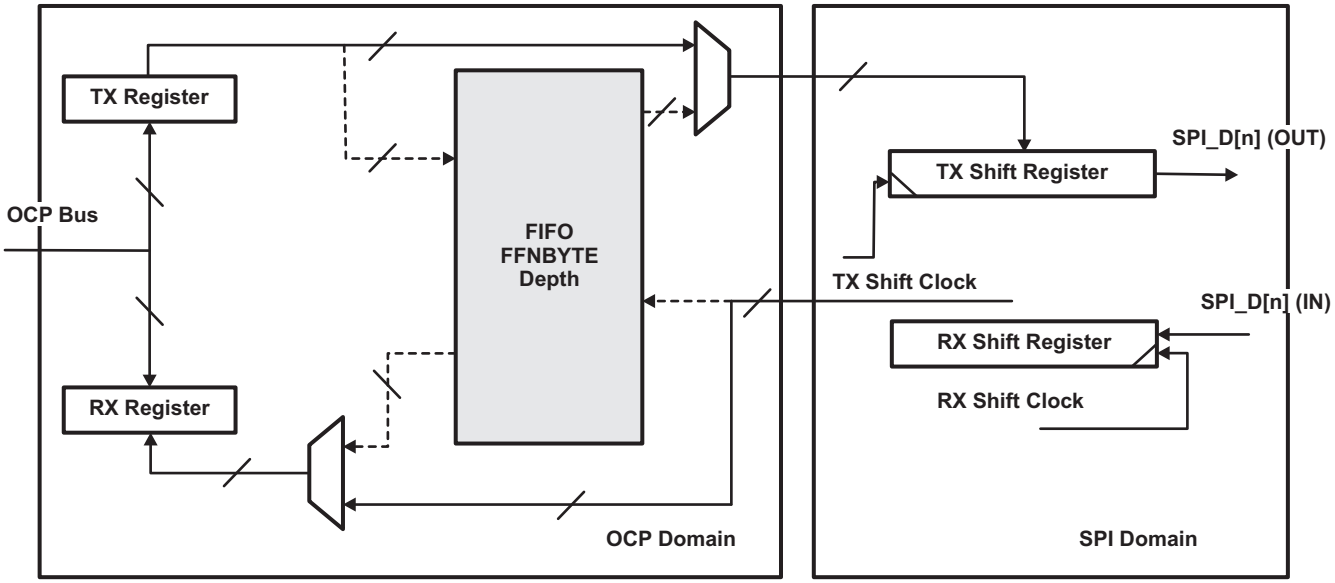
	SPI Word Length WL		
	3 ≤ WL ≤ 7	**8 ≤ WL ≤ 15**	**16 ≤ WL ≤ 31**
Number of byte written in the FIFO	1 byte	2 bytes	4 byte

16.2.3.10.1 Split FIFO

The FIFO can be split into two parts when the module is configured in transmit/receive mode MCSPI_CH(I)CONF[TRM] is cleared to 0 and MCSPI_CH(I)CONF[FFER] and MCSPI_CH(I)CONF[FFEW] asserted. Then system can access a FFNBYTE/2 byte depth FIFO per direction.

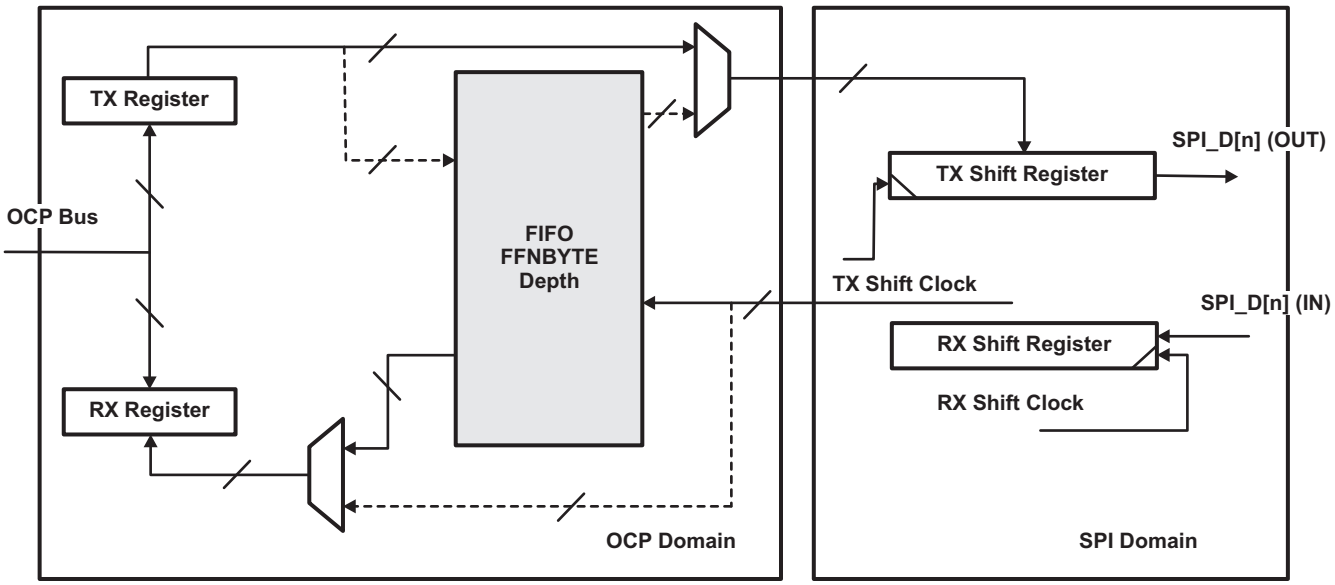
The FIFO buffer pointers are reset when the corresponding channel is enabled or FIFO configuration changes.

Figure 16-12. Transmit/Receive Mode With No FIFO Used



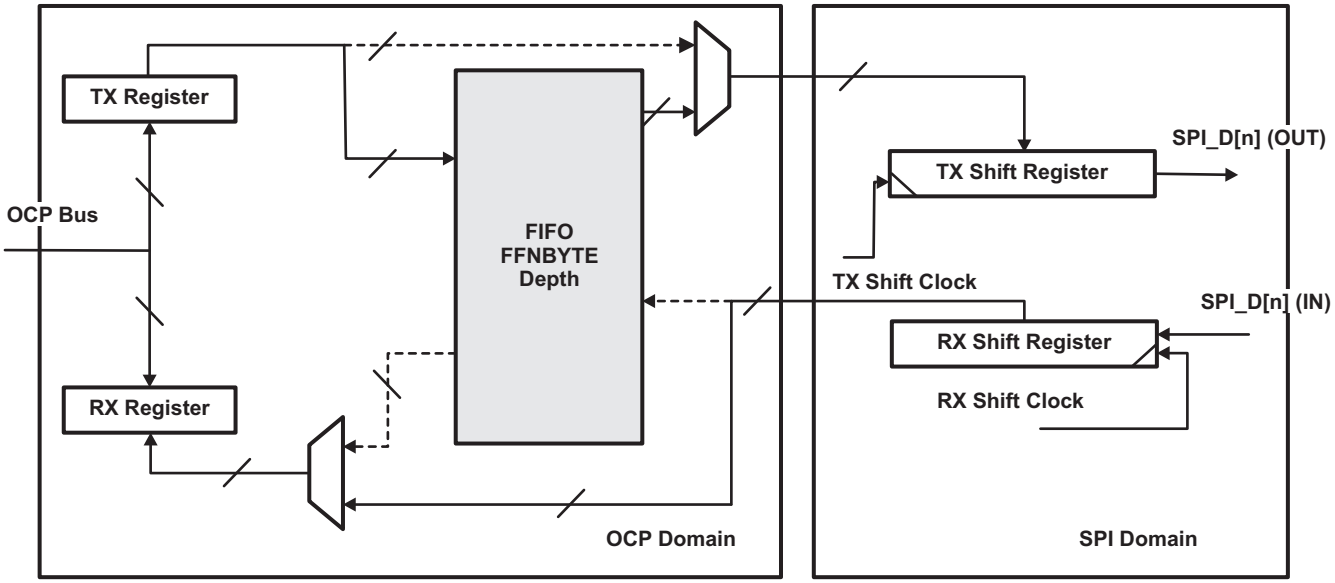
Configuration:
MCSPI_CH(i)CONF[TRM]= 0 (Transmit/receive mode)
MCSPI_CH(i)CONF[FFER] = 0 (FIFO disabled on receive path)
MCSPI_CH(i)CONF[FFEW] = 0 (FIFO disabled on transmit path)

Figure 16-13. Transmit/Receive Mode With Only Receive FIFO Enabled



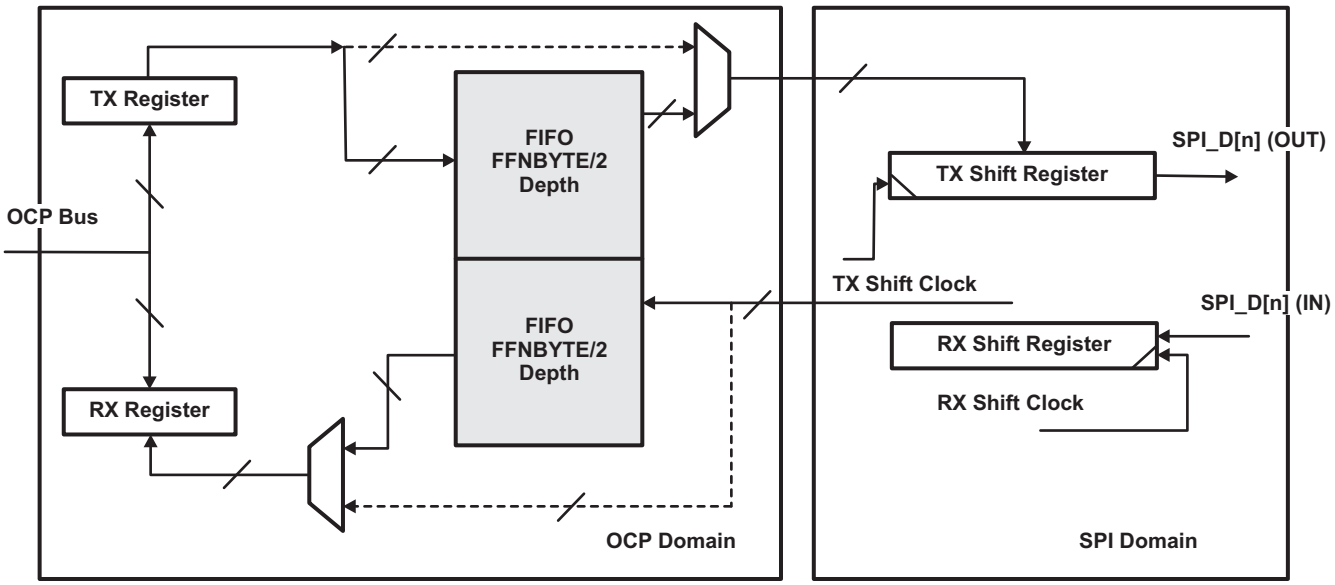
Configuration:
MCSPI_CH(i)CONF[TRM] = 0 (Transmit/receive mode)
MCSPI_CH(i)CONF[FFER] = 1 (FIFO enabled on receive path)
MCSPI_CH(i)CONF[FFEW] = 0 (FIFO disabled on transmit path)

Figure 16-14. Transmit/Receive Mode With Only Transmit FIFO Used



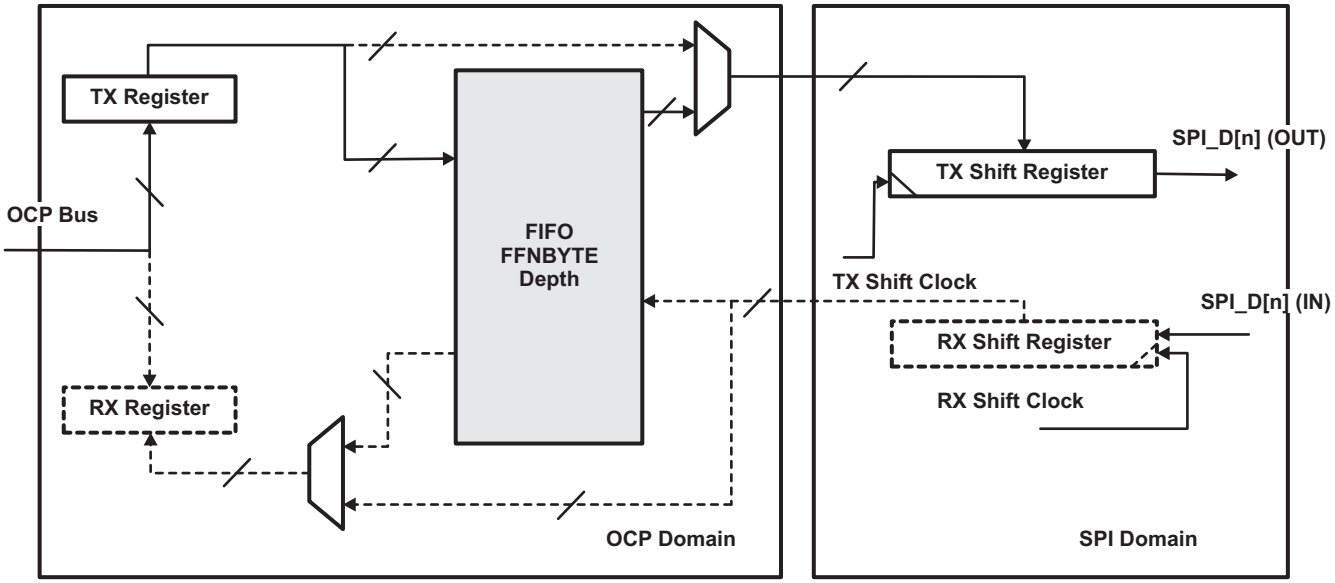
Configuration:
MCSPI_CH(i)CONF[TRM] = 0 (Transmit/receive mode)
MCSPI_CH(i)CONF[FFER] = 0 (FIFO disabled on receive path)
MCSPI_CH(i)CONF[FFEW] = 1 (FIFO enabled on transmit path)

Figure 16-15. Transmit/Receive Mode With Both FIFO Direction Used



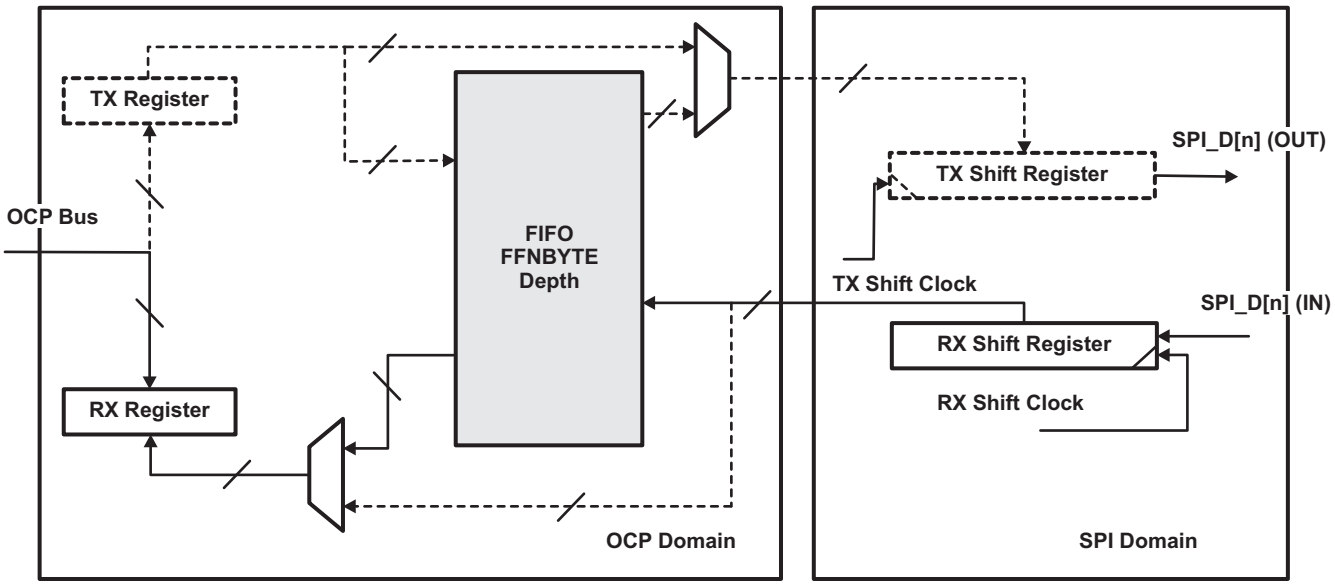
Configuration:
MCSPI_CH(i)CONF[TRM] = 0 (Transmit/receive mode)
MCSPI_CH(i)CONF[FFER] = 1 (FIFO enabled on receive path)
MCSPI_CH(i)CONF[FFEW] = 0 (FIFO disabled on transmit path)

Figure 16-16. Transmit-Only Mode With FIFO Used



Configuration:
MCSPI_CH(i)CONF[TRM] = 2h (Transmit-only mode)
MCSPI_CH(i)CONF[FFER] = 1 (FIFO enabled on transmit path)
MCSPI_CH(i)CONF[FFEW] (not applicable)

Figure 16-17. Receive-Only Mode With FIFO Used



Configuration:
MCSPI_CH(i)CONF[TRM] = 1 (Receive-only mode)
MCSPI_CH(i)CONF[FFER] = 1 (FIFO enabled on receive path)
MCSPI_CH(i)CONF[FFEW] (not applicable)

16.2.3.10.2 Buffer Almost Full

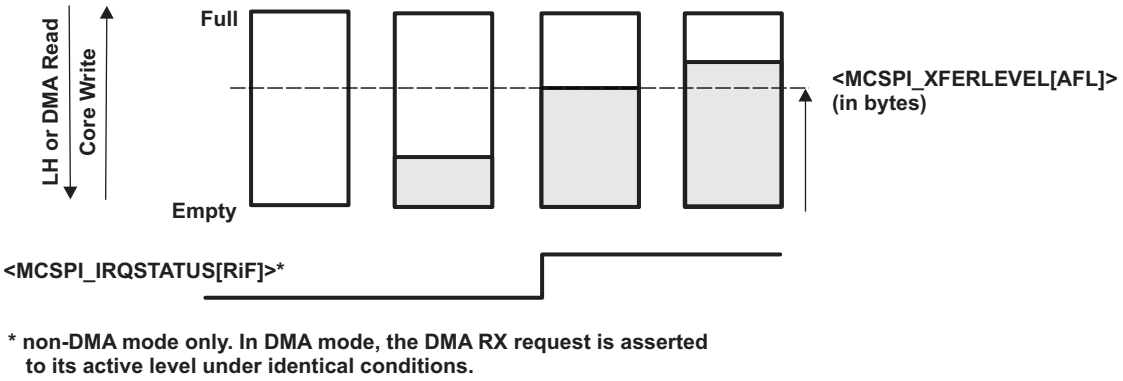
The bit field MCSPI_XFERLEVEL[AFL] is needed when the buffer is used to receive SPI word from a slave (MCSPI_CH(I)CONF[FFER] must be set to 1) and register width depends on the generic parameter FFNBYTE value. It defines the almost full buffer status.

When FIFO pointer reaches this level, an interrupt or a DMA request is sent to the CPU to enable the system to read AFL + 1 bytes from receive register. Be careful, AFL + 1 must correspond to a multiple value of MCSPI_CH(I)CONF[WL].

When DMA is used, the request is de-asserted after the first receive register read.

No new request will be asserted again as long as system has not performed the right number of read accesses.

Figure 16-18. Buffer Almost Full Level (AFL)



*** non-DMA mode only. In DMA mode, the DMA RX request is asserted to its active level under identical conditions.**

.

NOTE: SPI_IRQSTATUS register bits are not available in DMA mode. In DMA mode, the SPIm_DMA_RX*n* request is asserted on the same conditions than the SPI_IRQSTATUS RX*n*_FULL flag.

16.2.3.10.3 Buffer Almost Empty

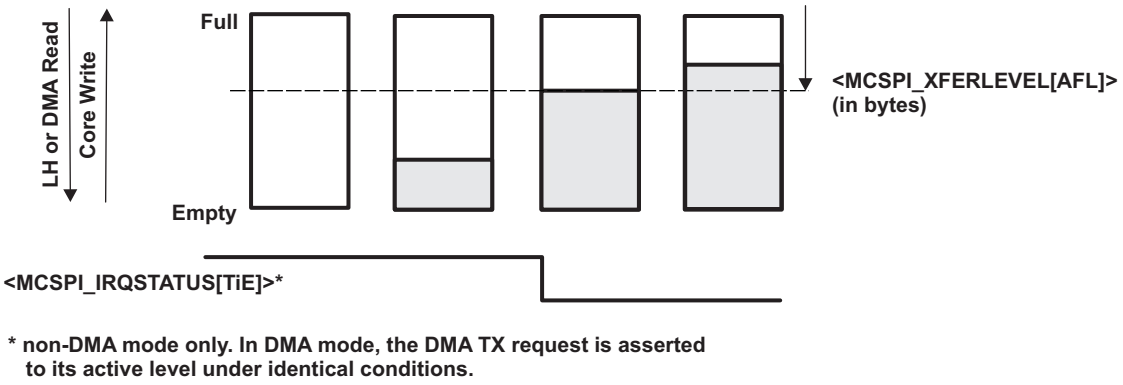
The bit field MCSPI_XFERLEVEL[AEL] is needed when the buffer is used to transmit SPI word to a slave (MCSPI_CH(I)CONF[FFEW] bit must be set to 1) and register width depends on the generic parameter FFNBYTE value. It defines the almost empty buffer status.

When FIFO pointer has not reached this level, an interrupt or a DMA request is sent to the CPU to enable the system to write AEL + 1 bytes to transmit register. Be careful, AEL + 1 must correspond to a multiple value of MCSPI_CH(I)CONF[WL].

When DMA is used, the request is de-asserted after the first transmit register write.

No new request will be asserted again as long as system has not performed the right number of write accesses.

Figure 16-19. Buffer Almost Empty Level (AEL)



.

16.2.3.10.4 End of Transfer Management

When the FIFO buffer is enabled for a channel, the user shall previously configure the MCSPI_XFERLEVEL register the AEL and AFL levels and especially the WCNT bit field to define the number of Spi word to be transferred using the FIFO before enabling the channel.

This counter allows the controller to stop the transfer correctly after a defined number of Spi word transfer. If WNCT is cleared to 0, the counter is not used and the user must stop the transfer manually by disabling the channel, in this case the user doesn't know how many SPI transfers have been done. For receive transfer, software shall poll the corresponding FFE bit field and read the Receive register to empty the FIFO buffer.

When End Of Word count interrupt is generated, the user can disable the channel and poll on MCSPI_CH(I)STAT[FFE] register to know if SPI word is still there in FIFO buffer and read last words.

NOTE: If the FIFO buffer is enabled, the module is configured to use the WCNT feature in the master mode, and the software uses the TX_EMPTY interrupt to load data into the FIFO, then the WCNT value must be a multiple of the Almost Empty Level (AEL).

16.2.3.10.5 Multiple SPI Word Access

The CPU has the ability to perform multiple SPI word access to the receive or transmit registers within a single 32-bit OCP access by setting the bit field MCSPI_MODULCTRL[MOA] to 1 under specific conditions:

- The channel selected has the FIFO enable.
- Only FIFO sense enabled support the kind of access.
- The bit field MCSPI_MODULCTRL[MOA] is set to 1.
- Only 32-bit OCP access and data width can be performed to receive or transmit registers, for other

kind of access the CPU must de-assert MCSPI_MODULCTRL[MOA] bit fields.

- The Level MCSPI_XFERLEVEL[AEL] and MCSPI_XFERLEVEL[AFL] must be 32-bit aligned , it means that AEL[0] = AEL[1] = 1 or AFL[0] = AFL[1] = 1.
- If MCSPI_XFERLEVEL[WCNT] is used it must be configured according to SPI word length.
- The word length of SPI words allows to perform multiple SPI access, that means that MCSPI_CH(I)CONF[WL] < 16

Number of SPI word access depending on SPI word length:

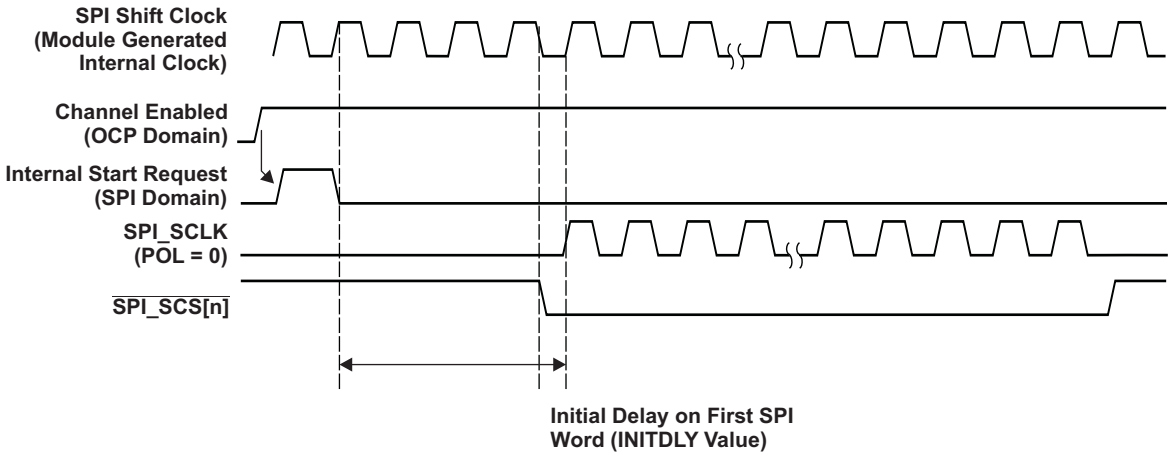
- 3 ≤ WL ≤ 7, SPI word length smaller or equal to byte length, four SPI words accessed per 32-bit OCP read/write. If word count is used (MCSPI_XFERLEVEL[WCNT]), set the bit field to WCNT[0] = WCNT[1] = 0
- 8 ≤ WL ≤ 15, SPI word length greater than byte or equal to 16-bit length, two SPI words accessed per 32-bit OCP read/write. If word count is used (MCSPI_XFERLEVEL[WCNT]), set the bit field to WCNT[0] = 0
- 16 ≤ WL multiple SPI word access not applicable.

16.2.3.11 First SPI Word Delayed

The SPI controller has the ability to delay the first SPI word transfer to give time for system to complete some parallel processes or fill the FIFO in order to improve transfer bandwidth. This delay is applied only on first SPI word after SPI channel enabled and first write in Transmit register. It is based on output clock frequency.

This option is meaningful in master mode and single channel mode MCSPI_MODULECTRL[SINGLE] asserted.

Figure 16-20. Master Single Channel Initial Delay



Few delay values are available: No delay, 4/8/16/32 SPI cycles.

Its accuracy is half cycle in clock bypass mode and depends on clock polarity and phase.

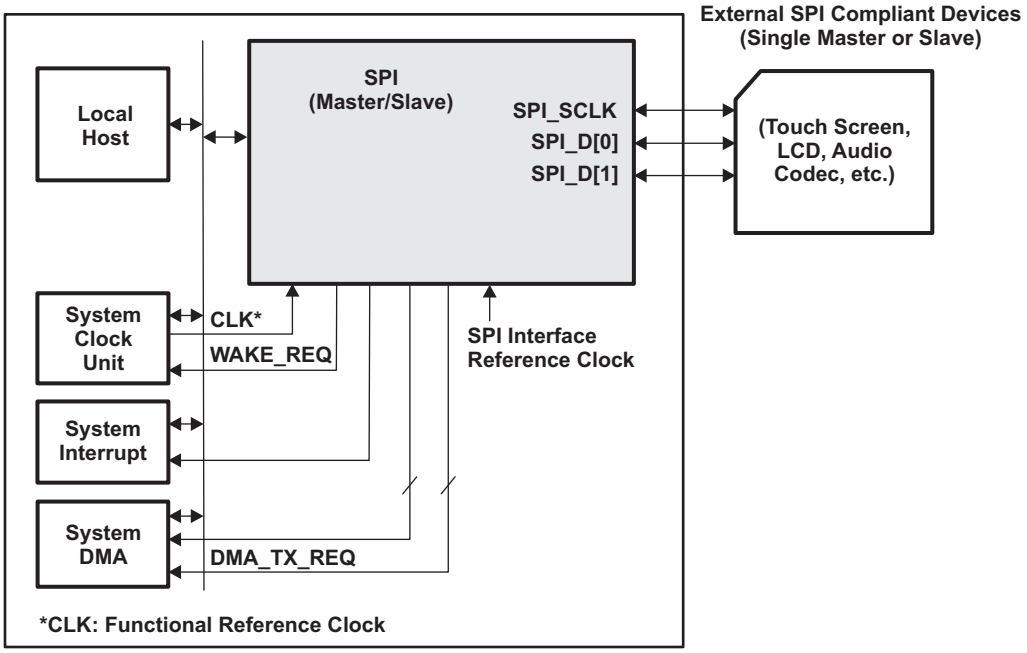
16.2.3.12 3-Pin or 4-Pin Mode

External SPI bus interface can be configured to use a restricted set of pin using the bit field MCSPI_MODULCTRL[1] PIN34 and depending on targeted application:

- If MCSPI_MODULECTRL[34] is cleared to 0 (default value), the controller is in 4-pin mode using the SPI pins: SPI_SCLK, SPI_D[0], SPI_D[1], and SPI_SCS[n].
- If MCSPI_MODULECTRL[34] is set to 1, the controller is in 3-pin mode using the SPI pins: SPI_SCLK, SPI_D[0], and SPI_D[1].

In this mode it is mandatory to put the controller in single channel master mode (MCSPI_MODULECTRL[SINGLE] asserted) and to connect only one SPI device on the bus.

Figure 16-21. 3-Pin Mode System Overview



In 3-pin mode all options related to chip select management are useless:

- MCSPI_CHxCONF[EPOL]
- MCSPI_CHxCONF[TCS0]
- MCSPI_CHxCONF[FORCE]

The chip select pin $\overline{\text{SPI\_SCS[n]}}$ is forced to 0 in this mode.

16.2.4 Slave Mode

SPI is in slave mode when the bit MS of the register MCSPI_MODULCTRL is set.

In slave mode, SPI can be connected to up to 4 external SPI master devices. SPI handles transactions with a single SPI master device at a time.

In slave mode, SPI initiates data transfer on the data lines (SPI_D[1:0]) when it receives an SPI clock (SPI_SCLK) from the external SPI master device.

The controller is able to work with or without a chip select $\overline{\text{SPI\_SCS[n]}}$ depending on MCSPI_MODULCTRL[1] PIN34 bit setting. It also support transfers without dead cycle between two successive words.

16.2.4.1 Dedicated Resources

In slave mode, enabling a channel that is not channel 0 has no effect. Only channel 0 can be enabled. The channel 0, in slave mode has the following resources:

- Its own channel enable, programmable with the bit EN of the register MCSPI_CH0CTRL. This channel should be enabled before transmission and reception. Disabling the channel, outside data word transmission, remains under user responsibility.
- Any of the 4 ports $\overline{\text{SPI\_SCS[3:0]}}$ can be used as a slave SPI device enable. This is programmable with the SPIENSLV bits of the register MCSPI_CH0CONF.
- Its own transmitter register MCSPI_TX on top of the common shift register. If the transmitter register is empty, the status bit TXS of the register MCSPI_CH0STAT is set. When SPI is selected by an external master (active signal on the $\overline{\text{SPI\_SCS[n]}}$ port assigned to channel 0), the transmitter register content of channel0 is always loaded in shift register whether it has been updated or not. The transmitter register should be loaded before SPI is selected by a master.
- Its own receiver register MCSPI_RX on top of the common shift register. If the receiver register is full, the status bit RXS of the register MCSPI_CH0STAT is set.

NOTE: The transmitter register and receiver registers of the other channels are not used. Read from or Write in the registers of a channel that is not channel 0 has no effect.

- Its own communication configuration with the following parameters via the register MCSPI_CH0CONF:
  - Transmit/Receive modes, programmable with the bit TRM.
  - Interface mode (Two data pins or Single data pin) and data pins assignment, both programmable with the bits IS and DPE.
  - SPI word length, programmable with the bits WL.
  - $\overline{\text{SPI\_SCS[n]}}$ polarity, programmable with the bit EPOL.
  - SPI_SCLK polarity, programmable with the bit POL.
  - SPI_SCLK phase, programmable with the bit PHA.
  - Use a FIFO buffer or not, programmable with FFER and FFEW, depending on transfer mode TRM.

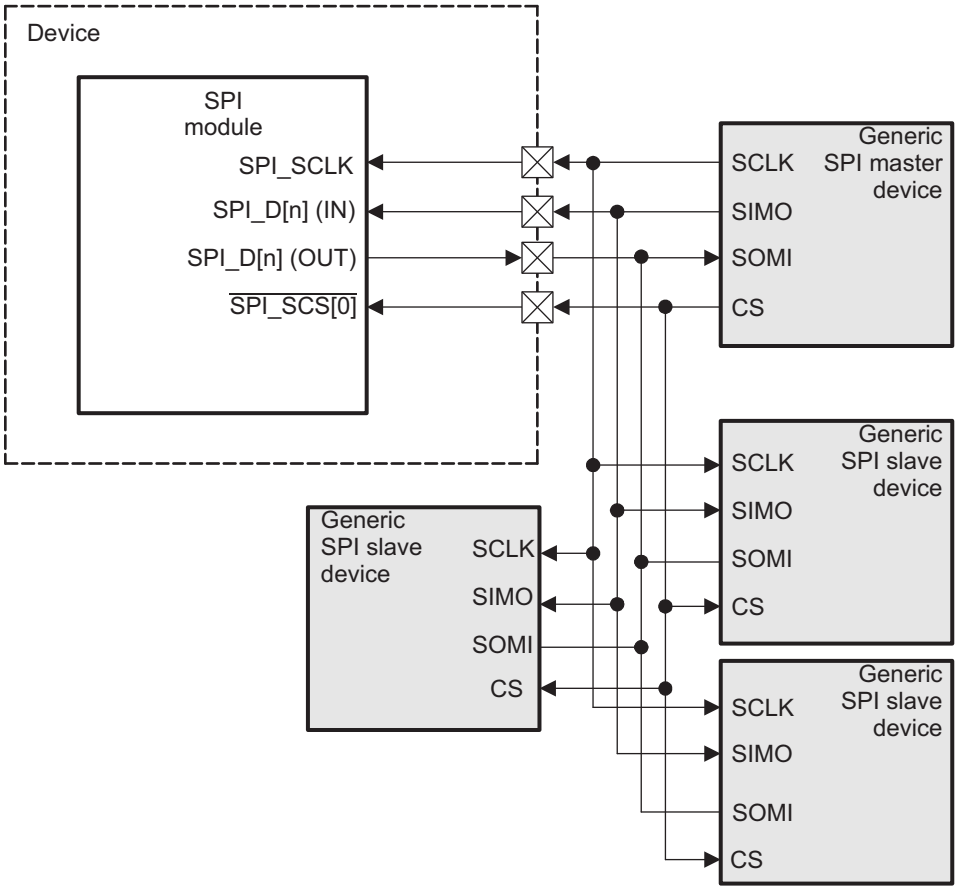
The SPI_SCLK frequency of a transfer is controlled by the external SPI master connected to SPI. The bits CLKD0 of the register 0CONF are not used in slave mode.

NOTE: The configuration of the channel can be loaded in the 0CONF register only when the channel is disabled.

- Two DMA requests events, read and write, to synchronize read/write accesses of the DMA controller with the activity of SPI. The DMA requests are enabled with the bits DMAR and DMAW of the register 0CONF.
- Four interrupts events.

Figure 16-22 shows an example of four slaves wired on a single master device.

Figure 16-22. Example of SPI Slave with One Master and Multiple Slave Devices on Channel 0



16.2.4.2 Interrupt Events in Slave Mode

The interrupt events related to the transmitter register state are TX_empty and TX_underflow. The interrupt events related to the receiver register state are RX_full and RX_overflow.

16.2.4.2.1 TX_EMPTY

The event TX_empty is activated when the channel is enabled and its transmitter register becomes empty. Enabling channel automatically raises this event. When FIFO buffer is enabled (MCSPI_CH(I)CONF[FFEW] set to 1), the TX_empty is asserted as soon as there is enough space in buffer to write a number of byte defined by MCSPI_XFERLEVEL[AEL].

Transmitter register must be load to remove source of interrupt and TX_empty interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

When FIFO is enabled, no new TX_empty event will be asserted as soon as Local Host has not performed the number of write into transmitter register defined by MCSPI_XFERLEVEL[AEL]. It is the responsibility of Local Host to perform the right number of writes.

16.2.4.2.2 TX_UNDERFLOW

The event TX_underflow is activated when channel is enabled and if the transmitter register or FIFO (if use of buffer is enabled) is empty (not updated with new data) when an external master device starts a data transfer with SPI (transmit and receive).

When FIFO is enabled the data emitted while underflow event is raised is not the last data written in the FIFO.

The TX_underflow indicates an error (data loss) in slave mode.

To avoid having TX_underflow event at the beginning of a transmission, the event TX_underflow is not activated when no data has been loaded into the transmitter register since channel has been enabled.

TX_underflow interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

16.2.4.2.3 RX_FULL

The event RX_full is activated when channel is enabled and receiver becomes filled (transient event). When FIFO buffer is enabled (MCSPI_CH(I)CONF[FFER] set to 1), the RX_full is asserted as soon as there is a number of bytes holds in buffer to read defined by MCSPI_XFERLEVEL[AFL].

Receiver register must be read to remove source of interrupt and RX_full interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

When FIFO is enabled, no new RX_full event will be asserted as soon as Local Host has not performed the number of read into receive register defined by MCSPI_XFERLEVEL[AFL]. It is the responsibility of Local Host to perform the right number of reads.

16.2.4.2.4 RX_OVERFLOW

The RX0_OVERFLOW event is activated in slave mode in either transmit-and-receive or receive-only mode, when a channel is enabled and the SPI_RX*n* register or FIFO is full when a new SPI word is received. The SPI_RX*n* register is always overwritten with the new SPI word. If the FIFO is enabled data within the FIFO are overwritten, it must be considered as corrupted. The RX0_OVERFLOW event should not appear in slave mode using the FIFO.

The RX0_OVERFLOW indicates an error (data loss) in slave mode.

The SPI_IRQSTATUS[3] RX0_OVERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

16.2.4.2.5 End of Word Count

The event EOW (End Of Word count) is activated when channel is enabled and configured to use the built-in FIFO. This interrupt is raised when the controller had performed the number of transfer defined in MCSPI_XFERLEVEL[WCNT] register. If the value was programmed to 0000h, the counter is not enabled and this interrupt is not generated.

The End of Word count interrupt also indicates that the SPI transfer is halt on channel using the FIFO buffer as soon as MCSPI_XFERLEVEL[WCNT] is not reloaded and channel re-enabled.

End of Word interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

16.2.4.3 Slave Transmit-and-Receive Mode

The slave transmit and receive mode is programmable (bits TRM cleared to 0 in the register MCSPI_CH(I)CONF).

After the channel is enabled, transmission and reception proceed with interrupt and DMA request events.

In slave transmit and receive mode, transmitter register should be loaded before SPI is selected by an external SPI master device.

Transmitter register or FIFO (if use of buffer enabled) content is always loaded in shift register whether it has been updated or not. The event TX_underflow is activated accordingly, and does not prevent transmission.

On completion of SPI word transfer (bit EOT of the register MCSPI_CH(I)STAT is set) the received data is transferred to the channel receive register. This bit is meaningless when using the Buffer for this channel.

The built-in FIFO is available in this mode and can be configured in one data direction Transmit or Receive, then the FIFO is seen as a unique FFNBYTE bytes buffer, or it can also be configured in both data direction Transmit and Receive, then the FIFO is split into two separate FFNBYTE/2 bytes buffer with their own address space management, in this last case the definition of AEL and AFL levels is based on FFNBYTE bytes and is under Local Host responsibility.

16.2.4.4 Slave Receive-Only Mode

The slave receive mode is programmable (bits TRM set to 01 in the register (I)CONF).

In receive only mode, the Transmitter register should be loaded before SPI is selected by an external SPI master device. Transmitter register or FIFO (if use of buffer enabled) content is always loaded in shift register whether it has been updated or not. The event TX_underflow is activated accordingly, and does not prevent transmission.

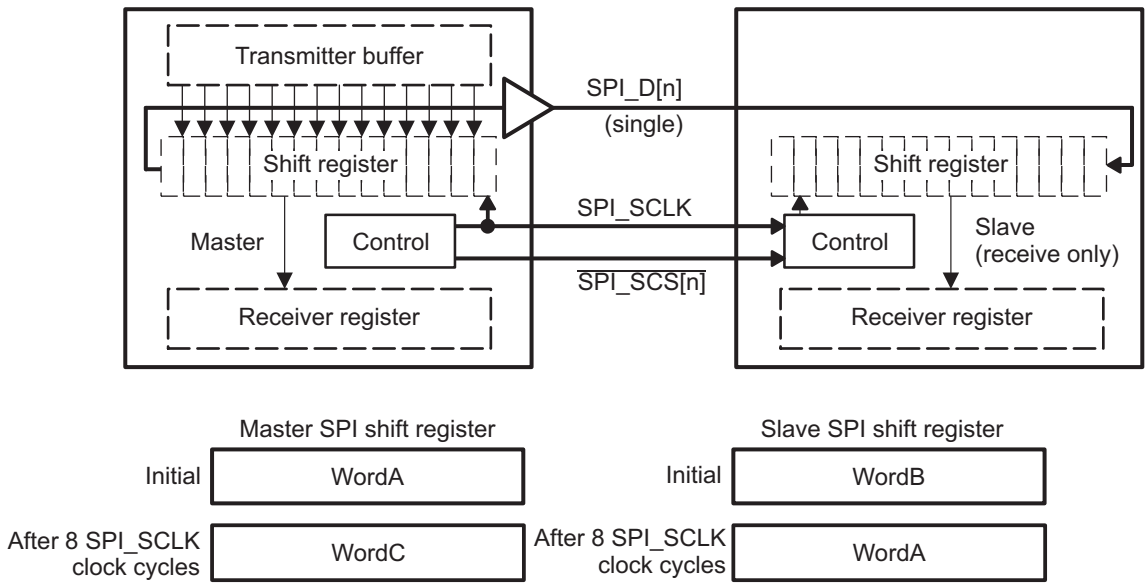
When an SPI word transfer completes (the SPIm.SPI_CH*n*STAT0[2] EOT bit (with *n* = 0) is set to 1), the received data is transferred to the channel receive register.

To use SPI as a slave receive only device with MCSPI_CH(I)CONF[TRM] = 00, the user has the responsibility of the inhibition of the TX_empty and TX_underflow interrupts and DMA write requests due to the transmitter register sate.

On completion of SPI word transfer (bit EOT of the register MCSPI_CH(I)STAT is set) the received data is transferred to the channel receive register. This bit is meaningless when using the Buffer for this channel. The built-in FIFO is available in this mode and can be configured with FFER bit field in the MCSPI_CH(I)CONF register, then the FIFO is seen as a unique FFNBYTE bytes buffer.

Figure 16-23 shows an example of a half-duplex system with a master device on the left and a receive-only slave device on the right. Each time a bit transfers out from the master, 1 bit transfers in from the slave. After eight cycles of the serial clock spim_clk, Word A transfers from the master to the slave.

Figure 16-23. SPI Half-Duplex Transmission (Receive-Only Slave)



16.2.4.5 Slave Transmit-Only Mode

The slave transmit only mode is programmable (TRM bit set to 10 in the register MCSPI_CH(I)CONF). This mode avoids the CPU to read the receiver register (minimizing data movement) when only transmission is meaningful.

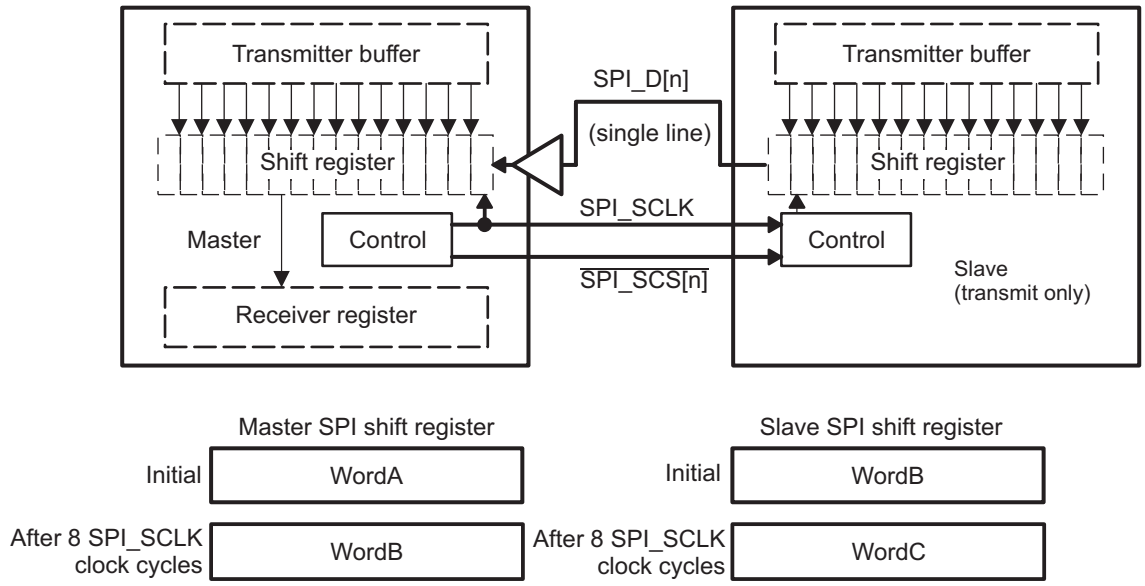
To use SPI as a slave transmit only device with (I)CONF[TRM] = 00, the user should inhibit the RX_full and RX_overflow interrupts and DMA read requests due to the receiver register sate.

On completion of SPI word transfer the bit EOT of the register MCSPI_CH(I)STAT is set. This bit is meaningless when using the buffer for this channel.

The built-in FIFO is available in this mode and can be configured with FFER bit field in the MCSPI_CH(I)CONF register, then the FIFO is seen as a unique FFNBYTE bytes buffer.

Figure 16-24 shows a half-duplex system with a master device on the left and a transmit-only slave device on the right. Each time a bit transfers out from the slave device, 1 bit transfers in the master. After eight cycles of the serial clock spim_clk, WordB transfers from the slave to the master.

Figure 16-24. SPI Half-Duplex Transmission (Transmit-Only Slave)



16.2.5 Interrupts

According to its transmitter register state and its receiver register state each channel can issue interrupt events if they are enabled.

The interrupt events are listed in the Section 16.2.3.2 and in Section 16.2.4.2.

Each interrupt event has a status bit, in the MCSPI_IRQSTATUS register, which indicates service is required, and an interrupt enable bit, in the MCSPI_IRQENABLE register, which enables the status to generate hardware interrupt requests.

When an interrupt occurs and later a mask is applied on it (IRQENABLE), the interrupt line is not asserted again even if the interrupt source has not been serviced.

SPI supports interrupt driven operation and polling.

16.2.5.1 Interrupt-Driven Operation

Alternatively, an interrupt enable bit, in the MCSPI_IRQENABLE register, can be set to enable each of the events to generate interrupt requests when the corresponding event occurs. Status bits are automatically set by hardware logic conditions.

When an event occurs (the single interrupt line is asserted), the CPU must:

- Read the MCSPI_IRQSTATUS register to identify which event occurred,
- Read the receiver register that corresponds to the event, to remove the source of an RX_full event, or Write into the transmitter register that corresponds to the event, to remove the source of a TX_empty event. No action is needed to remove the source of the events TX_underflow and RX_overflow.
- Write a 1 into the corresponding bit of MCSPI_IRQSTATUS register to clear the interrupt status, and release the interrupt line.

The interrupt status bit should always be reset after channel enabling and before event are enabled as interrupt source.

16.2.5.2 Polling

When the interrupt capability of an event is disabled in the MCSPI_IRQENABLE register, the interrupt line is not asserted and:

- The status bits in the MCSPI_IRQSTATUS register can be polled by software to detect when the corresponding event occurs.
- Once the expected event occurs, CPU must: Read the receiver register that corresponds to the event, to remove the source of an RX_full event, or write into the transmitter register that corresponds to the event, to remove the source of a TX_empty event. No action is needed to remove the source of the events TX_underflow and RX_overflow.
- Writing a 1 into the corresponding bit of MCSPI_IRQSTATUS register clears the interrupt status and does not affect the interrupt line state.

16.2.6 DMA Requests

SPI can be interfaced with a DMA controller. At system level, the advantage is to discharge the local host of the data transfers.

According to its transmitter register state, its receiver register state or FIFO level (if use of buffer for the channel) each channel can issue DMA requests if they are enabled.

The DMA requests need to be disabled in order to get TX and RX interrupts, in order to define either the end of the transfer or the transfer of the last words for the modes listed below:

- Master Transmit On
- Master normal receive only mode
- Master turbo receive only mode
- Slave Transmit Only

There are 2 DMA request lines per channel. The management of DMA requests differ according to use of FIFO buffer or not:

16.2.6.1 FIFO Buffer Disabled

The DMA Read request line is asserted when the channel is enabled and a new data is available in the receive register of the channel. DMA Read request can be individually masked with the bit DMAR of the register MCSPI_CH(I)CONF. The DMA Read request line is de-asserted (OCP compliant) on read completion of the receive register of the channel.

The DMA Write request line is asserted when the channel is enabled and the transmitter register of the channel is empty. DMA Write request can be individually masked with the bit DMAW of the register MCSPI_CH(I)CONF. The DMA Write request line is deasserted (OCP compliant) on load completion of the transmitter register of the channel.

Only one SPI word can be transmitted/received per OCP bus access to write/read the transmit or receive register.

16.2.6.2 FIFO Buffer Enabled

The DMA Read request line is asserted when the channel is enabled and a number of bytes defined in MCSPI_XFERLEVEL[AFL] bit field is hold in FIFO buffer for the receive register of the channel. DMA Read request can be individually masked with the bit DMAR of the register MCSPI_CH(I)CONF. The DMA Read request line is de-asserted (OCP compliant) on the first SPI word read completion of the receive register of the channel. No new DMA request will be asserted again as soon as user has not performed the right number of read accesses defined by MCSPI_XFERLEVEL[AFL] it is under user responsibility.

The DMA Write request line is asserted when the channel is enabled and the number of bytes hold in FIFO buffer is below the level defined by the MCSPI_XFERLEVEL[AEL] bit field. DMA Write request can be individually masked with the bit DMAW of the register MCSPI_CH(I)CONF. The DMA Write request line is de-asserted (OCP compliant) on load completion of the first SPI word in the transmitter register of the channel. No new DMA request will be asserted again as soon as user has not performed the right number of write accesses defined by MCSPI_XFERLEVEL[AEL] it is under user responsibility.

Only one SPI word can be transmitted/received per OCP bus access to write/read the transmit or receive FIFO.

16.2.6.3 DMA 256 Bit Aligned Addresses

The controller has two registers, MCSPI_DAFTX and MCSPI_DAFRX, used only with an enabled channel that manages the FIFO to be compliant the a DMA handler providing only 256-bit aligned addresses.

This features is activated when the bit field MCSPI_MODULCTRL[8] FDAA is set to 1 and only one enabled channel has its bit field MCSPI_CH(I)CONF[27] FFE(I)W or MCSPI_CH(I)CONF[28] FFE(I)R enabled.

In this case, the registers MCSPI_TX(I) and MCSPI_RX(I) are not used and data are managed through registers MCSPI_DAFTX and MCSPI_DAFRX.

16.2.7 Emulation Mode

The MReqDebug input differentiates a regular access of a processor (application access), from an emulator access.

Application access: MReqDebug = 0

In functional mode, the consequences of a read of a receiver register MCSPI_RX(i) are the following:

- The source of an RXi_Full event in the MCSPI_IRQSTATUS register is removed, if it was enabled in the MCSPI_IRQENABLE register.
- The RXiS status bit in the MCSPI_IRQSTATUS register is cleared.
- In master mode, depending on the round robin arbitration, and the transmitter register state, the channel may access to the shift register for transmission/reception.

Emulator access: MReqDebug = 1

In emulation mode, SPI behavior is the same as in functional mode but a read of a receiver register MCSPI_RX(i) is not intrusive:

- MCSPI_RX(i) is still considered as not read. When the FIFO buffer is enabled, pointers are not updated.
- The source of an RXi_Full event in the MCSPI_IRQSTATUS register is not removed. The RXiS status bit in the MCSPI_CH(i)STAT register is held steady.

In emulation mode, as in functional mode, based on the ongoing data transfers, the status bits of the MCSPI_CH(i)STAT register may be optionally updated, the interrupt and DMA request lines may be optionally asserted.

16.2.8 Power Saving Management

Independently of the module operational modes (Transmit and/or Receive), two modes of operations are defined from a power management perspective: normal and idle modes.

The two modes are temporally fully exclusive.

SPI is compliant with the profile “IdleReq / SIdleAck / Swakup” in the Idle mode.

16.2.8.1 Normal Mode

Both the OCP clock and SPI clock (CLKSPIREF) provided to SPI must be active for both master and slave modes. The auto-gating of the module OCP clock and SPI clock occurs when the following conditions are met:

- The bit AutoIdle of the register MCSPI_SYSCONFIG is set.
- In master mode, there is no data to transmit or receive in all channels.
- In slave mode, the SPI is not selected by the external SPI master device and no OCP accesses.

Autogating of the module OCP clock and SPI clock stops when the following conditions are met:

- In master mode, an OCP access occurs.
- In slave mode, an OCP access occurs or SPI is selected by an external SPI master device.

16.2.8.2 Idle Mode

The OCP clock and SPI clock provided to SPI may be switched off on system power manager request and switched back on module request.

SPI is compliant with the power management handshaking protocol: idle request from the system power manager, idle acknowledgment from SPI.

The idle acknowledgment in response to an idle request from the system power manager varies according to a programmable mode in the MCSPI_SYSCONFIG register: No idle mode, force idle mode, and smart idle mode.

- When programmed for no idle mode (the bit SIdleMode of the register MCSPI_SYSCONFIG is set to 01), the module ignores the system power manager request, and behaves normally, as if the request was not asserted.
- When programmed for smart idle mode (the bit SIdleMode of the register MCSPI_SYSCONFIG is set to 10), the module acknowledges the system power manager request according to its internal state.
- When programmed for force idle mode (the bit SIdleMode of the register MCSPI_SYSCONFIG is cleared to 00), the module acknowledges the system power manager request unconditionally.

The OCP clock will be optionally switched off, during the smart idle mode period, if the bit ClockActivity of the register MCSPI_SYSCONFIG is set.

The SPI clock will be optionally switched off, during the smart idle mode period, if the second bit ClockActivity of the register MCSPI_SYSCONFIG is set.

SPI assumes that both clocks may be switched off whatever the value set in the field ClockActivity of the register MCSPI_SYSCONFIG.

16.2.8.2.1 Transitions from Normal Mode to Smart-Idle Mode

The module detects an idle request when the synchronous signal IdleReq is asserted.

When IdleReq is asserted, any access to the module will generate an error as long as the OCP clock is alive.

When configured as a slave device, SPI responds to the idle request by asserting the SIdleAck signal (idle acknowledgement) only after completion of the current transfer ($\overline{\text{SPI\_SCS[n]}}$) slave selection signal deasserted by the external master) and if interrupt or DMA request lines are not asserted.

As a master device, SPI responds to the idle request by asserting the SIdleAck signal (idle acknowledgement) only after completion of all the channel data transfers and if interrupt or DMA request lines are not asserted.

As long as SIdleAck is not asserted, if an event occurs, the module can still generate an interrupt or a DMA request after IdleReq assertion. In this case, the module ignores the idle request and SIdleAck will not get asserted: The system power manager will abort the power mode transition procedure. It is then the responsibility of the system to de-assert IdleReq before attempting to access the module.

When SIdleAck is asserted, the module does not assert any new interrupt or DMA request.

16.2.8.2.2 Transition From Smart-Idle Mode to Normal mode

SPI detects the end of the idle period when the idle request signal (IdleReq) is deasserted.

Upon IdleReq de-assertion, the module switches back to normal mode and de-asserts SIdleAck signal. The module is fully operational.

16.2.8.2.3 Force-Idle Mode

Force-idle mode is enabled as follows:

- The bit SIdleMode of the register MCSPI_SYSCONFIG is cleared to 00 (Force Idle).

  The force idle mode is an idle mode where SPI responds unconditionally to the idle request by asserting the SIdleAck signal and by deasserting unconditionally the interrupt and DMA request lines if asserted.

  The transition from normal mode to idle mode does not affect the interrupt event bits of the MCSPI_IRQSTATUS register.

  In force-idle mode, the module is supposed to be disabled at that time, so the interrupt and DMA request lines are likely deasserted. OCP clock and SPI clock provided to SPI can be switched off.

  An idle request during an SPI data transfer can lead to an unexpected and unpredictable result, and is under software responsibility.

  Any access to the module in force idle mode will generate an error as long as the OCP clock is alive and IdleReq is asserted.

The module exits the force idle mode when:

- The idle request signal (IdleReq) is de-asserted.

  Upon IdleReq de-assertion, the module switches back to normal mode and de-asserts SIdleAck signal. The module is fully operational. The interrupt and DMA request lines are optionally asserted a clock cycle later.

16.2.9 System Test Mode

SPI is in system test mode (SYSTEST) when the bit System_Test of the register MCSPI_MODULCTRL is set.

The SYSTEST mode is used to check in a very simple manner the correctness of the system interconnect either internally to interrupt handler, or power manager, or externally to SPI I/Os.

I/O verification can be performed in SYSTEST mode by toggling the outputs and capturing the logic state of the inputs. (See MCSPI_SYST register definition in Section 16.3.10)

16.2.10 Reset

16.2.10.1 Internal Reset Monitoring

The module is reset by the hardware when an active-low reset signal, synchronous to the OCP interface clock is asserted on the input pin RESETN.

This hardware reset signal has a global reset action on the module. All configuration registers and all state machines are reset, in all clock domains.

Additionally, the module can be reset by software through the bit SoftReset of the register MCSPI_SYSCONFIG. This bit has exactly the same action on the module logic as the hardware RESETN signal. The register MCSPI_SYSCONFIG is not sensitive to software reset. The SoftReset control bit is active high. The bit is automatically reset to 0 by the hardware.

A global ResetDone status bit is provided in the status register MCSPI_SYSSTATUS. This bit is set to 1 when all the different clock domains resets (OCP domain and SPI domains) have been released (logical AND).

The global ResetDone status bit can be monitored by the software to check if the module is ready-to-use following a reset (either hardware or software).

The clock CLKSPIREF must be provided to the module, in order to allow the ResetDone status bit to be set.

When used in slave mode, the clock CLKSPIREF is needed only during the reset phase. The clock CLKSPIREF can be switched off after the ResetDone status is set.

16.2.10.2 Reset values of registers

The reset values of registers and signals are described in Section 16.3.

16.2.11 Access to Data Registers

This section details the supported data accesses (read or write) from/to the data receiver registers MCSPI_RX(i) and data transmitter registers MCSPI_TX(i).

Supported access:

SPI supports only one SPI word per register (receiver or transmitter) and does not support successive 8-bit or 16-bit accesses for a single SPI word.

The SPI word received is always right justified on LSB of the 32-bit register MCSPI_RX(i), and the SPI word to transmit is always right justified on LSB of the 32-bit register MCSPI_TX(i).

The upper bits, above SPI word length, are ignored and the content of the data registers is not reset between the SPI data transfers.

The coherence between the number of bits of the SPI Word, the number of bits of the access and the enabled byte remains under the user's responsibility. Only aligned accesses are supported.

In Master mode, data should not be written in the transmit register when the channel is disabled.

16.2.12 Programming Aid

16.2.12.1 Module Initialization

- Hard or soft reset.
- Read MCSPI_SYSSTATUS.
- Check if reset is done.
- Module configuration: (a) Write into MCSPI_MODULCTRL (b) Write into MCSPI_SYSCONFIG.
- Before the ResetDone bit is set, the clocks CLK and CLKSPIREF must be provided to the module.
- To avoid hazardous behavior, it is advised to reset the module before changing from MASTER mode to SLAVE mode or from SLAVE mode to MASTER mode.

16.2.12.2 SPI Operational Modes Configuration

The selection of the working mode is done through the MCSPI_CHxCONF register (where x = 0, 1, 2 and 3). Table 16-7 through Table 16-9 list the possible operating modes and their configurations.

Table 16-7. SPI Receive Mode Initialization

Step	Register/Bit Field/Programming Model	Value
Set receive mode.	MCSPI_CHxCONF[13:12] TRM	0x1
Set the word length.	MCSPI_CHxCONF[11:7] WL	0x8
Clock initialization and channel enabling	MCSPI_MODULCTRL[2] MS MCSPI_CHxCTRL[0] EN	0x0 0x1
Channels activated low during active state	MCSPI_CHxCONF[6] EPOL	0x1
Clock held high during active state	MCSPI_CHxCONF[1] POL	0x0
Data latched on odd-numbered edges of the SPI clock	MCSPI_CHxCONF[0] PHA	0x0
Reset the status bits.	MCSPI_IRQSTATUS	0x0

Table 16-8. SPI Transmit Mode Initialization

Step	Register/Bit Field/Programming Model	Value
Set transmit mode.	MCSPI_CHxCONF[13:12] TRM	0x2
Set the word length.	MCSPI_CHxCONF[11:7] WL	0x8
Clock initialization and channel enabling	MCSPI_MODULCTRL[2] MS MCSPI_CHxCTRL[0] EN	0x0 0x1
Channels activated low during active state	MCSPI_CHxCONF[6] EPOL	0x1
Clock held high during active state	MCSPI_CHxCONF[1] POL	0x0
Data latched on odd-numbered edges of the SPI clock	MCSPI_CHxCONF[0] PHA	0x0
Reset the status bits.	MCSPI_IRQSTATUS	0x0

Table 16-9. SPI Transmit-and-Receive Mode Initialization

Step	Register/Bit Field/Programming Model	Value
Set transmit and receive mode.	MCSPI_CHxCONF[13:12] TRM	0x0
Set the word length.	MCSPI_CHxCONF[11:7] WL	0x8
Clock initialization and channel enabling	MCSPI_MODULCTRL[2] MS MCSPI_CHxCTRL[0] EN	0x0 0x1
Channels activated low during active state	MCSPI_CHxCONF[6] EPOL	0x1
Clock held high during active state	MCSPI_CHxCONF[1] POL	0x0
Data latched on odd-numbered edges of the SPI clock	MCSPI_CHxCONF[0] PHA	0x0

Table 16-9. SPI Transmit-and-Receive Mode Initialization (continued)

Step	Register/Bit Field/Programming Model	Value
Reset the status bits.	MCSPI_IRQSTATUS	0x0

16.2.12.2.1 Common Transfer Procedures Without FIFO – Polling Method

16.2.12.2.1.1 Receive-Only Procedure – Polling Method

Table 16-10 lists the receive-only procedure using the polling method. The SPI is acting as slave.

Table 16-10. Receive-Only Procedure – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode	See Table 16-7.	
Start the channel.	MCSPI_CHxCTRL[0] EN	0x1
Wait until receive register is full?	MCSPI_RXx	= 0x1
Stop the channel.	MCSPI_CHxCTRL[0] EN	0x0

16.2.12.2.1.2 Receive-Only Procedure – Interrupt Method

Table 16-11 lists the receive-only procedure using the interrupt method. The SPI is acting as slave. Channel 0 is used. For the rest channels (1, 2 and 3) the same logic applies.

Table 16-11. Receive-Only Procedure – Interrupt Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 16-7.	
Start the channel.	MCSPI_CHxCTRL[0] EN	0x1
Enable the interrupt for the receiver register.	MCSPI_IRQENABLE[2] RX0_FULL_ENABLE	0x1
Read the status register.	MCSPI_IRQSTATUS[2] RX0_FULL	0x0
Disable the interrupt.	MCSPI_IRQENABLE[2] RX0_FULL_ENABLE	0x0
Stop the channel.	MCSPI_CHxCTRL[0] EN	0x0
Read the receiver register.	MCSPI_RX0	xxxx

16.2.12.2.1.3 Transmit-Only Procedure – Polling Method

Table 16-12 lists the transmit-only procedure using the polling method. The SPI is acting as master.

Table 16-12. Transmit-Only Procedure – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 16-8	
Start the channel.	MCSPI_CHxCTRL[0] EN	0x1
Wait until end of transfer?	MCSPI_CHxSTAT[2:1]	= 0x2
Stop the channel.	MCSPI_CHxCTRL[0] EN	0x0

16.2.12.2.1.4 Transmit-and-Receive Procedure – Polling Method

Table 16-12 lists the transmit-and-receive procedure using the polling method. The SPI is acting as master and slave.

Table 16-13. Transmit-and-Receive Procedure – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 16-9.	
Start the channel.	MCSPI_CHxCTRL[0] EN	0x1
Wait until transmit/receive word?	MCSPI_CHxSTAT[2:0]	= 0x3
Stop the channel.	MCSPI_CHxCTRL[0] EN	0x0

16.2.12.2.2 Common Transfer Procedures With FIFO – Polling Method

When using FIFO the SPI module can start the transfer only after the first write request is released by writing the MCSPI_TXx register (where x = 0, 1, 2 and 3). The first write request can be managed by the IRQ routine or DMA handler. The end of transfer is more complex and depends on the transfer type. See Table 16-12 through Table 16-19.

In multi-channel master mode, be careful not to overwrite the bits of the other channels when initializing the MCSPI_IRQSTATUS and MCSPI_IRQENABLE registers.

16.2.12.2.2.1 Receive-Only Procedure With Word Count – Polling Method

Table 16-14. Receive-Only Procedure With Word Count – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 16-7.	
Start the channel.	MCSPI_CHxCTRL[0] EN	0x1
Wait until end of word count?	MCSPI_IRQSTATUS[17] EOW	= 0x1
Stop the channel.	MCSPI_CHxCTRL[0] EN	0x0
Read from the receiver register.	MCSPI_RXx	xxxx

16.2.12.2.2.2 Transmit-Only Procedure With and Without Word Count – Polling Method

Table 16-15. Transmit-Only Procedure Without Word Count – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 16-8.	
Start the channel.	MCSPI_CHxCTRL[0] EN	0x1
Wait until end of word count?	MCSPI_IRQSTATUS[17] EOW	= 0x1
Wait until end of transfer?	MCSPI_CHxSTAT[2] EOT MCSPI_CHxSTAT[3] TXFFE	= 0x1 = 0x1
Stop the channel.	MCSPI_CHxCTRL[0] EN	0x0

16.2.12.2.2.3 Transmit-Only Procedure With and Without Word Count – Interrupt Method

Table 16-16. Transmit-Only Procedure With Word Count – Interrupt Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 16-8.	
Start the channel.	MCSPI_CHxCTRL[0] EN	0x1
Enable the interrupt for the transmit register.	MCSPI_IRQENABLE[4] TX1_EMPTY_ENABLE	0x1
End of word count.	MCSPI_IRQSTATUS[17] EOW	= 0x1
End of transfer	MCSPI_CHxSTAT[2] EOT MCSPI_CHxSTAT[3] TXFFE	= 0x1 = 0x1
Clear the interrupt.	MCSPI_IRQENABLE[17] EOWKE	= 0x0

Table 16-16. Transmit-Only Procedure With Word Count – Interrupt Method (continued)

Step	Register/Bit Field/Programming Model	Value
Disable the interrupt for the transmit register.	MCSPI_IRQENABLE[4] TX1_EMPTY_ENABLE	0x0
Stop the channel.	MCSPI_CHxCTRL[0] EN	0x0

16.2.12.2.2.4 Transmit-and-Receive Procedure With Word Count – Polling Method

Table 16-17. Transmit-and-Receive Procedure With Word Count – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 16-9.	
Start the channel.	MCSPI_CHxCTRL[0] EN	0x1
Wait until end of word count?	MCSPI_IRQSTATUS[17] EOW	= 0x1
Stop the channel.	MCSPI_CHxCTRL[0] EN	0x0
Read from the receiver register.	MCSPI_RXx	xxxx

16.2.12.2.2.5 Transmit-and-Receive Procedure with Word Count – Interrupt Method

Table 16-18. Transmit-and-Receive Procedure With Word Count – Interrupt Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 16-9.	
Start the channel.	MCSPI_CHxCTRL[0] EN	0x1
Enable the interrupt for the receiver register.	MCSPI_IRQENABLE[2] RX0_FULL_ENABLE	0x1
Enable the interrupt for the transmit register.	MCSPI_IRQENABLE[4] TX1_EMPTY_ENABLE	0x1
End of word count?	MCSPI_IRQSTATUS[17] EOW	= 0x1
Clear the interrupt.	MCSPI_IRQENABLE[17] EOWKE	= 0x0
Disable the interrupt for the receiver register.	MCSPI_IRQENABLE[2] RX0_FULL_ENABLE	0x0
Disable the interrupt for the transmit register.	MCSPI_IRQENABLE[4] TX1_EMPTY_ENABLE	0x0
Stop the channel.	MCSPI_CHxCTRL[0] EN	0x0
Read from the receiver register.	MCSPI_RXx	xxxx

16.2.12.2.2.6 Transmit-and-Receive Procedure Without Word Count – Polling Method

Table 16-19. Transmit-and-Receive Procedure Without Word Count – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 16-9.	
Start the channel.	MCSPI_CHxCTRL[0] EN	0x1
Wait until end of transfer?	MCSPI_CHxSTAT[2] EOT MCSPI_CHxSTAT[3] TXFFE	= 0x1 = 0x1
Stop the channel.	MCSPI_CHxCTRL[0] EN	0x0
Read from the receiver register.	MCSPI_RXx	xxxx

16.2.13 Interrupt and DMA Events

SPI has two DMA requests (Rx and Tx) per channel. It also has one interrupt line for all the interrupt requests.

16.3 SPI Registers

Table 16-20 lists the SPI registers. For the base address of these registers, see Table 1-12.

Table 16-20. SPI Registers

Offset Address	Acronym	Register Name	Section
0h	MCSPI_HL_REV	McSPI IP revision register	Section 16.3.1
4h	MCSPI_HL_HWINFO	McSPI IP hardware information register	Section 16.3.2
10h	MCSPI_HL_SYSCONFIG	McSPI IP system configuration register	Section 16.3.3
100h	MCSPI_REVISION	McSPI revision register	Section 16.3.4
110h	MCSPI_SYSCONFIG	McSPI system configuration register	Section 16.3.5
114h	MCSPI_SYSSTATUS	McSPI system status register	Section 16.3.6
118h	MCSPI_IRQSTATUS	McSPI interrupt status register	Section 16.3.7
11Ch	MCSPI_IRQENABLE	McSPI interrupt enable register	Section 16.3.8
120h	MCSPI_WAKEUPENABLE	McSPI wakeup enable register	Section 16.3.9
124h	MCSPI_SYST	McSPI system test register	Section 16.3.10
128h	MCSPI_MODULCTRL	McSPI module control register	Section 16.3.11
12Ch	MCSPI_CH0CONF	McSPI channel 0 configuration register	Section 16.3.12
130h	MCSPI_CH0STAT	McSPI channel 0 status register	Section 16.3.13
134h	MCSPI_CH0CTRL	McSPI channel 0 control register	Section 16.3.14
138h	MCSPI_TX0	McSPI channel 0 FIFO transmit buffer register	Section 16.3.15
13Ch	MCSPI_RX0	McSPI channel 0 FIFO receive buffer register	Section 16.3.16
140h	MCSPI_CH1CONF	McSPI channel 1 configuration register	Section 16.3.12
144h	MCSPI_CH1STAT	McSPI channel 1 status register	Section 16.3.13
148h	MCSPI_CH1CTRL	McSPI channel 1 control register	Section 16.3.14
14Ch	MCSPI_TX1	McSPI channel 1 FIFO transmit buffer register	Section 16.3.15
150h	MCSPI_RX1	McSPI channel 1 FIFO receive buffer register	Section 16.3.16
154h	MCSPI_CH2CONF	McSPI channel 2 configuration register	Section 16.3.12
158h	MCSPI_CH2STAT	McSPI channel 2 status register	Section 16.3.13
15Ch	MCSPI_CH2CTRL	McSPI channel 2 control register	Section 16.3.14
160h	MCSPI_TX2	McSPI channel 2 FIFO transmit buffer register	Section 16.3.15
164h	MCSPI_RX2	McSPI channel 2 FIFO receive buffer register	Section 16.3.16
168h	MCSPI_CH3CONF	McSPI channel 3 configuration register	Section 16.3.12
16Ch	MCSPI_CH3STAT	McSPI channel 3 status register register	Section 16.3.13
170h	MCSPI_CH3CTRL	McSPI channel 3 control register	Section 16.3.14
174h	MCSPI_TX3	McSPI channel 3 FIFO transmit buffer register	Section 16.3.15
178h	MCSPI_RX3	McSPI channel 3 FIFO receive buffer register	Section 16.3.16
17Ch	MCSPI_XFERLEVEL	McSPI transfer levels register	Section 16.3.17
180h	MCSPI_DAFTX	DMA address aligned FIFO transmitter register	Section 16.3.18
1A0h	MCSPI_DAFRX	DMA address aligned FIFO receiver register	Section 16.3.19

16.3.1 McSPI IP Revision Register (MCSPI_HL_REV)

The McSPI IP revision register (MCSPI_HL_REV) holds the IP revision identifier information. The MCSPI_HL_REV is shown in Figure 16-25 and described in Table 16-21.

Figure 16-25. McSPI IP Revision Register (MCSPI_HL_REV)

31	30	29	28	27	16
SCHEME		Reserved		FUNC	
R-1		R-0		R-30h	

15	11	10	8	7	6	5	0
R_RTL		X_MAJOR		CUSTOM		Y_MINOR	
R-0		R-0		R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 16-21. McSPI IP Revision Register (MCSPI_HL_REV) Field Descriptions

Bit	Field	Value	Description
31-30	SCHEME	0-3h	Identifies scheme.
29-28	Reserved	0	Reserved for module specific status information. Read returns 0.
27-16	FUNC	0-FFFh	Indicates a software compatible module family.
15-11	R_RTL	0-1Fh	RTL version (R).
10-8	X_MAJOR	0-7h	Major revision (X).
7-6	CUSTOM	0-3h	Indicates a special version for a particular device.
5-0	Y_MINOR	0-3Fh	Minor revision (Y).

16.3.2 McSPI IP Hardware Information Register (MCSPI_HL_HWINFO)

The McSPI IP hardware information register (MCSPI_HL_HWINFO) provides information about the IP hardware configuration. The MCSPI_HL_HWINFO is shown in Figure 16-26 and described in Table 16-22.

Figure 16-26. McSPI IP Hardware Information Register (MCSPI_HL_HWINFO)

31	16
Reserved	
R-0	

15	6	5	1	0
Reserved		FFNBYTE		USEFIFO
R-0		R-4h		R-1

LEGEND: R = Read only; -*n* = value after reset

Table 16-22. McSPI IP Hardware Information Register (MCSPI_HL_HWINFO) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved.
5-1	FFNBYTE		FIFO number of byte generic parameter. Defines the value of FFNBYTE generic parameter.
		0	Reserved
		1h	FIFO 16 bytes depth
		2h	FIFO 32 bytes depth
		3h	Reserved
		4h	FIFO 64 bytes depth
		5h-7h	Reserved
		8h	FIFO 128 bytes depth
		9h-Fh	Reserved
		10h	FIFO 256 bytes depth
		11h-1Fh	Reserved
0	USEFIFO		Use of a FIFO enable.
		0	FIFO not implemented in design
		1	FIFO and its management implemented in design with depth defined by FFNBYTE generic.

16.3.3 McSPI IP System Configuration Register (MCSPI_HL_SYSCONFIG)

The McSPI IP system configuration register (MCSPI_HL_SYSCONFIG) allows control of the clock management. The MCSPI_HL_SYSCONFIG is shown in Figure 16-27 and described in Table 16-23.

Figure 16-27. McSPI IP System Configuration Register (MCSPI_HL_SYSCONFIG)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		IDLEMODE		FREEEMU	SOFTRESET
R-0		R/W-2h		R/W-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 16-23. McSPI IP System Configuration Register (MCSPI_HL_SYSCONFIG) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved.
3-2	IDLEMODE		Configuration of the local target state management mode. The target can handle read/write transactions as long as it is out of IDLE state.
		0	Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally,that is, regardless of the IP module's internal requirements.
		1h	No-idle mode: local target never enters idle state.
		2h	Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module shall not generate (IRQ- or DMA-request-related) wakeup events.
		3h	Smart-idle wakeup-capable mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module may generate (IRQ- or DMA-request-related) wakeup events when in idle state. Mode is only relevant if the appropriate IP module "swakeup" output(s) is (are) implemented.
1	FREEEMU		Sensitivity to emulation (debug) suspend input signal.
		0	IP module is sensitive to emulation suspend.
		1	IP module is not sensitive to emulation suspend.
0	SOFTRESET		Software reset.
		Write 0	No action
		Read 0	Reset done, no pending action
		Write 1	Initiate software reset
		Read 1	Reset (software or other) ongoing

16.3.4 McSPI Revision Register (MCSPI_REVISION)

The McSPI revision register (MCSPI_REVISION) contains the hard-coded RTL revision number. The MCSPI_REVISION is shown in Figure 16-28 and described in Table 16-24.

Figure 16-28. McSPI Revision Register (MCSPI_REVISION)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		REV	
R-0		R-x	

LEGEND: R = Read only; -*n* = value after reset

Table 16-24. McSPI Revision Register (MCSPI_REVISION) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7-0	REV	0-FFh	Identifies revision of peripheral.

16.3.5 McSPI System Configuration Register (MCSPI_SYSCONFIG)

The McSPI system configuration register (MCSPI_SYSCONFIG) allows control of various parameters of the module interface. It is not sensitive to software reset. The MCSPI_SYSCONFIG is shown in Figure 16-29 and described in Table 16-25.

Figure 16-29. McSPI System Configuration Register (MCSPI_SYSCONFIG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R/W-0															

15	14	13	12	11	10	9	8
Reserved						CLOCKACTIVITY	
R/W-0						R/W-0	

7	6	5	4	3	2	1	0
Reserved			SIDLEMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE
R/W-0			R/W-2h		R/W-1	R/W-0	R/W-1

LEGEND: R/W = Read/Write; -*n* = value after reset

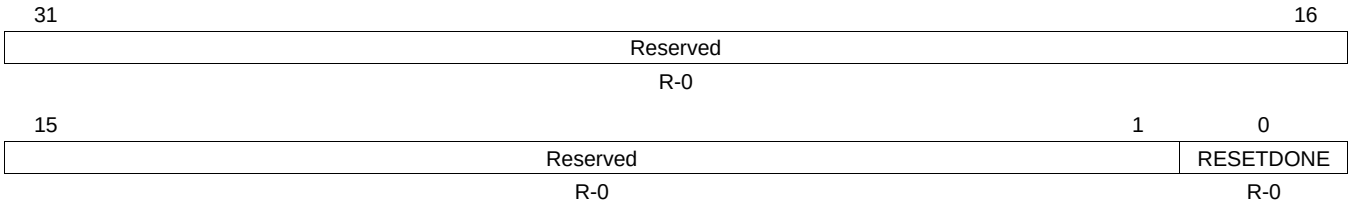
Table 16-25. McSPI System Configuration Register (MCSPI_SYSCONFIG) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads returns 0
9-8	CLOCKACTIVITY		Clocks activity during wake-up mode period.
		0	OCP and Functional clocks may be switched off.
		1h	OCP clock is maintained. Functional clock may be switched-off.
		2h	Functional clock is maintained. OCP clock may be switched-off.
		3h	OCP and Functional clocks are maintained.
7-5	Reserved	0	Reads returns 0
4-3	SIDLEMODE		Power management
		0	If an idle request is detected, the McSPI acknowledges it unconditionally and goes in Inactive mode. Interrupt, DMA requests are unconditionally de-asserted.
		1h	If an idle request is detected, the request is ignored and keeps on behaving normally.
		2h	Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements.
		3h	Reserved
2	ENAWAKEUP		Enable Wakeup.
		0	Wakeup capability is disabled.
		1	Wakeup capability is enabled.
1	SOFTRESET		Software reset. During reads it always returns 0.
		0	(write) Normal mode
		1	(write) Set this bit to 1 to trigger a module reset. The bit is automatically reset by the hardware.
0	AUTOIDLE		Internal OCP Clock gating strategy
		0	OCP clock is free-running
		1	Automatic OCP clock gating strategy is applied, based on the OCP interface activity

16.3.6 McSPI System Status Register (MCSPI_SYSSTATUS)

The McSPI system status register (MCSPI_SYSSTATUS) provides status information about the module excluding the interrupt status information. The MCSPI_SYSSTATUS is shown in Figure 16-30 and described in Table 16-26.

Figure 16-30. McSPI System Status Register (MCSPI_SYSSTATUS)



LEGEND: R = Read only; -*n* = value after reset

Table 16-26. McSPI System Status Register (MCSPI_SYSSTATUS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved for module specific status information. Read returns 0.
0	RESETDONE		Internal Reset Monitoring
		0	Internal module reset is on-going
		1	Reset completed

16.3.7 McSPI Interrupt Status Register (MCSPI_IRQSTATUS)

The McSPI interrupt status register (MCSPI_IRQSTATUS) regroups all the status of the module internal events that can generate an interrupt. The MCSPI_IRQSTATUS is shown in Figure 16-31 and described in Table 16-27.

NOTE: In SYSTEST mode, the bits of this register have no meaning and always read 0.

Figure 16-31. McSPI Interrupt Status Register (MCSPI_IRQSTATUS)

31							18	17	16
Reserved								EOW	Rsvd
R/W-0								R/W-0	R-0

15	14	13	12	11	10	9	8
Rsvd	RX3_FULL	TX3_UNDERFLOW	TX3_EMPTY	Reserved	RX2_FULL	TX2_UNDERFLOW	TX2_EMPTY
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Rsvd	RX1_FULL	TX1_UNDERFLOW	TX1_EMPTY	RX0_OVERFLOW	RX0_FULL	TX0_UNDERFLOW	TX0_EMPTY
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 16-27. McSPI Interrupt Status Register (MCSPI_IRQSTATUS) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reads returns 0
17	EOW		End of word (EOW) count event when a channel is enabled using the FIFO buffer and the channel has sent the number of McSPI words defined by the MCSPI_XFERLEVEL[WCNT].
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.
16	Reserved	0	Reserved
15	Reserved	0	Reads returns 0
14	RX3_FULL		Receiver register is full or almost full. Only when Channel 3 is enabled. This bit indicate FIFO almost full status when built-in FIFO is used for receive register (MCSPI_CH3CONF[FFE3R] is set).
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.
13	TX3_UNDERFLOW		Transmitter register underflow. Only when Channel 3 is enabled. The transmitter register is empty (not updated by Host or DMA with new data) before its time slot assignment. Exception: No TX_underflow event when no data has been loaded into the transmitter register since channel has been enabled.
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.

Table 16-27. McSPI Interrupt Status Register (MCSPI_IRQSTATUS) Field Descriptions (continued)

Bit	Field	Value	Description
12	TX3_EMPTY		Transmitter register is empty or almost empty. This bit indicate FIFO almost full status when built-in FIFO is used for transmit register (MCSPI_CH3CONF[FFE3W] is set).
			Note: Enabling the channel automatically raises this event.
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.
11	Reserved	0	Reads returns 0
10	RX2_FULL		Receiver register full or almost full. Channel 2 This bit indicate FIFO almost full status when built-in FIFO is used for receive register (MCSPI_CH3CONF[FFE2R] is set).
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.
9	TX2_UNDERFLOW		Transmitter register underflow. Channel 2
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.
8	TX2_EMPTY		Transmitter register empty or almost empty. Channel 2. This bit indicate FIFO almost full status when built-in FIFO is used for transmit register (MCSPI_CH3CONF[FFE2W] is set).
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.
7	Reserved	0	Reads returns 0
6	RX1_FULL		Receiver register full or almost full. Channel 1. This bit indicate FIFO almost full status when built-in FIFO is use for receive register (MCSPI_CH3CONF[FFE1R] is set).
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.
5	TX1_UNDERFLOW		Transmitter register underflow. Channel 1.
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.
4	TX1_EMPTY		Transmitter register empty or almost empty. Channel 1. This bit indicate FIFO almost full status when built-in FIFO is use for transmit register (MCSPI_CH3CONF[FFE1W] is set).
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.
3	RX0_OVERFLOW		Receiver register overflow (slave mode only). Channel 0.
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.

Table 16-27. McSPI Interrupt Status Register (MCSPI_IRQSTATUS) Field Descriptions (continued)

Bit	Field	Value	Description
2	RX0_FULL		Receiver register full or almost full. Channel 0. Receiver register full or almost full. Channel 0
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.
1	TX0_UNDERFLOW		Transmitter register underflow. Channel 0.
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.
0	TX0_EMPTY		Transmitter register empty or almost empty. Channel 0. This bit indicate FIFO almost full status when built-in FIFO is use for transmit register (MCSPI_CH3CONF[FFE0W] is set).
		Write 0	Event status bit is unchanged.
		Read 0	Event false.
		Write 1	Event status bit is reset.
		Read 1	Event is pending.

16.3.8 McSPI Interrupt Enable Register (MCSPI_IRQENABLE)

This McSPI interrupt enable register (MCSPI_IRQENABLE) enables/disables the module internal sources of interrupt, on an event-by-event basis. The MCSPI_IRQENABLE is shown in Figure 16-32 and described in Table 16-28.

Figure 16-32. McSPI Interrupt Enable Register (MCSPI_IRQENABLE)

31							18	17	16
Reserved								EOWKE	Rsvd
R/W-0								R/W-0	R-0

15	14	13	12	11	10	9	8
Rsvd	RX3_FULL_ ENABLE	TX3_UNDERFLOW_ ENABLE	TX3_EMPTY_ ENABLE	Reserved	RX2_FULL_ ENABLE	TX2_UNDERFLOW_ ENABLE	TX2_EMPTY_ ENABLE
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Rsvd	RX1_FULL_ ENABLE	TX1_UNDERFLOW_ ENABLE	TX1_EMPTY_ ENABLE	RX0_OVERFLOW_ ENABLE	RX0_FULL_ ENABLE	TX0_UNDERFLOW_ ENABLE	TX0_EMPTY_ ENABLE
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 16-28. McSPI Interrupt Enable Register (MCSPI_IRQENABLE) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reads return 0
17	EOWKE		End of word count interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
16	Reserved		Reserved
15	Reserved	0	Reads return 0
14	RX3_FULL_ ENABLE		MCSPI_RX3 receiver register full or almost full interrupt enable (channel 3).
		0	Interrupt is disabled.
		1	Interrupt is enabled.
13	TX3_UNDERFLOW_ ENABLE		MCSPI_TX3 transmitter register underflow interrupt enable (channel 3).
		0	Interrupt is disabled.
		1	Interrupt is enabled.
12	TX3_EMPTY_ ENABLE		MCSPI_TX3 transmitter register empty or almost empty interrupt enable (channel 3).
		0	Interrupt is disabled.
		1	Interrupt is enabled.
11	Reserved	0	Reads return 0
10	RX2_FULL_ ENABLE		MCSPI_RX2 receiver register full or almost full interrupt enable (channel 2).
		0	Interrupt is disabled.
		1	Interrupt is enabled.
9	TX2_UNDERFLOW_ ENABLE		MCSPI_TX2 transmitter register underflow interrupt enable (channel 2).
		0	Interrupt is disabled.
		1	Interrupt is enabled.
8	TX2_EMPTY_ ENABLE		MCSPI_TX2 transmitter register empty or almost empty interrupt enable (channel 2).
		0	Interrupt is disabled.
		1	Interrupt is enabled.
7	Reserved	0	Reads return 0

Table 16-28. McSPI Interrupt Enable Register (MCSPI_IRQENABLE) Field Descriptions (continued)

Bit	Field	Value	Description
6	RX1_FULL_ ENABLE		MCSPI_RX1 receiver register full or almost full interrupt enable (channel 1)
		0	Interrupt is disabled.
		1	Interrupt is enabled.
5	TX1_UNDERFLOW_ ENABLE		MCSPI_TX1 transmitter register underflow interrupt enable (channel 1).
		0	Interrupt is disabled.
		1	Interrupt is enabled.
4	TX1_EMPTY_ ENABLE		MCSPI_TX1 transmitter register empty or almost empty interrupt enable (channel 1).
		0	Interrupt is disabled.
		1	Interrupt is enabled.
3	RX0_OVERFLOW_ ENABLE		MCSPI_RX0 receivier register overflow interrupt enable (channel 0).
		0	Interrupt is disabled.
		1	Interrupt is enabled.
2	RX0_FULL_ ENABLE		MCSPI_RX0 receiver register full or almost full interrupt enable (channel 0).
		0	Interrupt is disabled.
		1	Interrupt is enabled.
1	TX0_UNDERFLOW_ ENABLE		MCSPI_TX0 transmitter register underflow interrupt enable (channel 0).
		0	Interrupt is disabled.
		1	Interrupt is enabled.
0	TX0_EMPTY_ ENABLE		MCSPI_TX0 transmitter register empty or almost empty interrupt enable (channel 0).
		0	Interrupt is disabled.
		1	Interrupt is enabled.

16.3.9 McSPI Wakeup Enable Register (MCSPI_WAKEUPENABLE)

The McSPI wakeup enable register (MCSPI_WAKEUPENABLE) allows you to enable/disable the module internal sources of wakeup on an event-by-event basis. The MCSPI_WAKEUPENABLE is shown in Figure 16-33 and described in Table 16-29.

Figure 16-33. McSPI Wakeup Enable Register (MCSPI_WAKEUPENABLE)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															WKEN
R-0															R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-29. McSPI Wakeup Enable Register (MCSPI_WAKEUPENABLE) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Read returns 0.
0	WKEN		Wakeup functionality in slave mode when an active control signal is detected on the $\overline{\text{SPI\_SCS[n]}}$ line programmed in MCSPI_CH(i)CONF[SPIENSLV].
		0	The event is not allowed to wakeup the system, even if MCSPI_SYSCONFIG[ENAWAKEUP] is set.
		1	The event is allowed to wakeup the system if MCSPI_SYSCONFIG[ENAWAKEUP] is set.

16.3.10 McSPI System Test Register (MCSPI_SYST)

This McSPI system test register (MCSPI_SYST) is used to configure the system interconnect either internally to the peripheral bus or externally to the device I/O pads, when the module is configured in the system test (SYSTEST) mode. The MCSPI_SYST is shown in Figure 16-34 and described in Table 16-30.

Figure 16-34. McSPI System Test Register (MCSPI_SYST)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R/W-0															

15	14	13	12	11	10	9	8
Reserved				SSB	SPIENDIR	SPIDATDIR1	SPIDATDIR0
R/W-0				R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved	SPICLK	SPIDAT_1	SPIDAT_0	SPIEN_3	SPIEN_2	SPIEN_1	SPIEN_0
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 16-30. McSPI System Test Register (MCSPI_SYST) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads returns 0
11	SSB		Set status bit. This bit must be cleared prior attempting to clear a status bit of the MCSPI_ IRQSTATUS register.
		0	No action. Writing 0 does not clear already set status bits.
			This bit must be cleared prior attempting to clear a status bit of the <MCSPI_IRQSTATUS> register.
		1	Writing 1 sets to 1 all status bits contained in the MCSPI_IRQSTATUS register.
			Writing 1 into this bit sets to 1 all status bits contained in the <MCSPI_IRQSTATUS> register.
10	SPIENDIR		Sets the direction of the $\overline{\text{SPI\_SCS}}$[3:0] lines and SPI_SCLK line.
		0	Output (as in master mode).
		1	Input (as in slave mode).
9	SPIDATDIR1		Sets the direction of the SPI_D[1].
		0	Output
		1	Input
8	SPIDATDIR0		Sets the direction of the SPI_D[0].
		0	Output
		1	Input
7	Reserved	0	Reserved
6	SPICLK		SPI_SCLK line (signal data value)
			If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the CLKSPI line (high or low), and a write into this bit has no effect.
			If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the CLKSPI line is driven high or low according to the value written into this register.
5	SPIDAT_1		SPI_D[1] line (signal data value)
			If MCSPI_SYST[SPIDATDIR1] = 0 (output mode direction), the SPI_D[1] line is driven high or low according to the value written into this register.
			If MCSPI_SYST[SPIDATDIR1] = 1 (input mode direction), this bit returns the value on the SPI_D[1] line (high or low), and a write into this bit has no effect.
4	SPIDAT_0		SPI_D[0] line (signal data value)
			If MCSPI_SYST[SPIDATDIR0] = 0 (output mode direction), the SPI_D[0] line is driven high or low according to the value written into this register.
			If MCSPI_SYST[SPIDATDIR0] = 1 (input mode direction), this bit returns the value on the SPI_D[0] line (high or low), and a write into this bit has no effect.

Table 16-30. McSPI System Test Register (MCSPI_SYST) Field Descriptions (continued)

Bit	Field	Value	Description
3	SPIEN_3		$\overline{\text{SPI\_SCS}}$[3] line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the $\overline{\text{SPI\_SCS}}$[3] line is driven high or low according to the value written into this register. If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the $\overline{\text{SPI\_SCS}}$[3] line (high or low), and a write into this bit has no effect.
2	SPIEN_2		$\overline{\text{SPI\_SCS}}$[2] line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the $\overline{\text{SPI\_SCS}}$[2] line is driven high or low according to the value written into this register. If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the $\overline{\text{SPI\_SCS}}$[2] line (high or low), and a write into this bit has no effect.
1	SPIEN_1		$\overline{\text{SPI\_SCS}}$[1] line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the $\overline{\text{SPI\_SCS}}$[1] line is driven high or low according to the value written into this register. If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the $\overline{\text{SPI\_SCS}}$[1] line (high or low), and a write into this bit has no effect.
0	SPIEN_0		$\overline{\text{SPI\_SCS}}$[0] line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the $\overline{\text{SPI\_SCS}}$[0] line is driven high or low according to the value written into this register. If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the $\overline{\text{SPI\_SCS}}$[0] line (high or low), and a write into this bit has no effect.

16.3.11 McSPI Module Control Register (MCSPI_MODULCTRL)

This McSPI module control register (MCSPI_MODULCTRL) is used to configure the serial port interface. The MCSPI_MODULCTRL is shown in Figure 16-35 and described in Table 16-31.

Figure 16-35. McSPI Module Control Register (MCSPI_MODULCTRL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R/W-0															

15	14	13	12	11	10	9	8
Reserved							FDAA
R/W-0							R/W-0

7	6	5	4	3	2	1	0
MOA	INITDLY			SYSTEM_TEST	MS	PIN34	SINGLE
R/W-0	R/W-0			R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 16-31. McSPI Module Control Register(MCSPI_MODULCTRL) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	
8	FDAA		FIFO DMA Address 256-bit aligned. This register is used when a FIFO is managed by the module and DMA connected to the controller provides only 256 bit aligned address. If this bit is set the enabled channel which uses the FIFO has its datas managed through MCSPI_DAFTX and MCSPI_DAFRX registers instead of MCSPI_TX(i) and MCSPI_RX(i) registers.
		0	FIFO data managed by MCSPI_TX(i) and MCSPI_RX(i) registers.
		1	FIFO data managed by MCSPI_DAFTX and MCSPI_DAFRX registers.
7	MOA		Multiple word ocp access. This register can only be used when a channel is enabled using a FIFO. It allows the system to perform multiple SPI word access for a single 32-bit OCP word access. This is possible for WL < 16.
		0	Multiple word access disabled
		1	Multiple word access enabled with FIFO
6-4	INITDLY		Initial SPI delay for first transfer. This register is an option only available in single master mode, The controller waits for a delay to transmit the first SPI word after channel enabled and corresponding TX register filled. This delay is based on SPI output frequency clock, No clock output provided to the boundary and chip select is not active in 4 pin mode within this period.
		0	No delay for first SPI transfer
		1h	The controller wait 4 SPI bus clock
		2h	The controller wait 8 SPI bus clock
		3h	The controller wait 16 SPI bus clock
		4h	The controller wait 32 SPI bus clock
		5h-Fh	Reserved
3	SYSTEM_TEST		Enables the system test mode
		0	Functional mode
		1	System test mode (SYSTEST)
2	MS		Master/ Slave
		0	Master - The module generates the SPI_SCLK and $\overline{\text{SPI\_SCS[3:0]}}$
		1	Slave - The module receives the SPI_SCLK and $\overline{\text{SPI\_SCS[3:0]}}$
1	PIN34		Pin mode selection. This register is used to configure the SPI pin mode, in master or slave mode. If asserted, the controller only use SPI_D[0], SPI_D[1], and SPI_SCLK clock pin for SPI transfers.
		0	$\overline{\text{SPI\_SCS[n]}}$ is used as a chip select.
		1	$\overline{\text{SPI\_SCS[n]}}$ is not used. In this mode all related option to chip select have no meaning.

Table 16-31. McSPI Module Control Register(MCSPI_MODULCTRL) Field Descriptions (continued)

Bit	Field	Value	Description
0	SINGLE		Single channel / Multichannel (master mode only) For master mode, the user has to set this bit to 0 for Multi or 1 for Single. For Slave mode, this bit has to be set to 0.
		0	More than one channel will be used in master mode.
		1	Only one channel will be used in master mode. This bit must be set in Force $\overline{\text{SPI\_SCS[n]}}$ mode.

16.3.12 McSPI Channel (i) Configuration Register (MCSPI_CH(i)CONF)

The McSPI channel *i* configuration register (MCSPI_CH(i)CONF) is used to configure channel*i*. The (MCSPI_CH(i)CONF) is shown in Figure 16-36 and described in Table 16-32.

Figure 16-36. McSPI Channel (*i*) Configuration Register (MCSPI_CH(i)CONF)

31	30	29	28	27	26	25	24
Reserved		CLKG	FFER	FFEW	TCS		SBPOL
R/W-0		R/W-0	R/W-0	R/W-0	R/W-0		R/W-0
23	22	21	20	19	18	17	16
SBE	SPIENSLV		FORCE	TURBO	IS	DPE1	DPE0
R/W-0	R/W-0		R/W-0	R/W-0	R/W-1	R/W-1	R/W-0
15	14	13	12	11			8
DMAR	DMAW	TRM		WL			
R/W-0	R/W-0	R/W-0		R/W-0			
7	6	5			2	1	0
WL	EPOL	CLKD				POL	PHA
R/W-0	R/W-0	R/W-0				R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-32. McSPI Channel (*i*) Configuration Register (MCSPI_CH(i)CONF) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Read returns 0
29	CLKG		Clock divider granularity. This register defines the granularity of channel clock divider: power of two or one clock cycle granularity.
			When this bit is set the register MCSPI_CHCTRL[EXTCLK] must be configured to reach a maximum of 4096 clock divider ratio. Then The clock divider ratio is a concatenation of MCSPI_CHCONF[CLKD] and MCSPI_CHCTRL[EXTCLK] values.
		0	Clock granularity of power of 2
		1	1 clock cycle granularity
28	FFER		FIFO enabled for receive. Only one channel can have this bit set.
		0	The buffer is not used to receive data.
		1	The buffer is used to receive data.
27	FFEW		FIFO enabled for transmit. Only one channel can have this bit set.
		0	The buffer is not used to transmit data.
		1	The buffer is used to transmit data.
26-25	TCS		Chip select time control. This 2-bit field defines the number of interface clock cycles between CS toggling and first or last edge of SPI clock.
		0	0.5 clock cycles
		1h	1.5 clock cycles
		2h	2.5 clock cycles
		3h	3.5 clock cycles
24	SBPOL		Start bit polarity.
		0	Start bit polarity is held to 0 during SPI transfer.
		1	Start bit polarity is held to 1 during SPI transfer.
23	SBE		Start bit enable for SPI transfer.
		0	Default SPI transfer length as specified by WL bit field.
		1	Start bit D/CX added before SPI transfer. Polarity is defined by MCSPI_CH(i)CONF[SBPOL].

Table 16-32. McSPI Channel *(i)* Configuration Register (MCSPI_CH(i)CONF) Field Descriptions (continued)

Bit	Field	Value	Description
22-21	SPIENSLV		Channel 0 only and slave mode only: SPI slave select signal detection.
		0	Detection enabled only on $\overline{\text{SPI\_SCS[0]}}$
		1h	Detection enabled only on $\overline{\text{SPI\_SCS[1]}}$
		2h	Detection enabled only on $\overline{\text{SPI\_SCS[2]}}$
		3h	Detection enabled only on $\overline{\text{SPI\_SCS[3]}}$
20	FORCE		Manual $\overline{\text{SPI\_SCS[n]}}$ assertion to keep $\overline{\text{SPI\_SCS[n]}}$ active between SPI words. (single channel master mode only)
		0	Writing 0 into this bit drives the $\overline{\text{SPI\_SCS[n]}}$ line when MCSPI_CHCONF(i)[EPOL] = 0, and drives it high when MCSPI_CHCONF(i)[EPOL] = 1.
		1	Writing 1 into this bit drives the $\overline{\text{SPI\_SCS[n]}}$ line when MCSPI_CHCONF(i)[EPOL] = 0, and drives it low when MCSPI_CHCONF(i)[EPOL] = 1.
19	TURBO		Turbo mode.
		0	Turbo is deactivated (recommended for single SPI word transfer).
		1	Turbo is activated to maximize the throughput for multi-SPI word transfers.
18	IS		Input select
		0	Data line 0 (SPI_D[0]) selected for reception.
		1	Data line 1 (SPI_D[1]) selected for reception.
17	DPE1		Transmission enable for data line 1 (SPIDATAGZEN[1])
		0	Data line 1 (SPI_D[1]) selected for transmission
		1	No transmission on data line 1 (SPI_D[1])
16	DPE0		Transmission enable for data line 0 (SPIDATAGZEN[0])
		0	Data line 0 (SPI_D[0]) selected for transmission
		1	No transmission on data line 0 (SPI_D[0])
15	DMAR		DMA read request. The DMA read request line is asserted when the channel is enabled and new data is available in the receive register of the channel. The DMA read request line is deasserted on read completion of the receive register of the channel.
		0	DMA read request is disabled.
		1	DMA read request is enabled.
14	DMAW		DMA write request. The DMA write request line is asserted when the channel is enabled and the MCSPI_TX*n* register of the channel is empty. The DMA write request line is deasserted on load completion of the MCSPI_TX*n* register of the channel.
		0	DMA write request is disabled.
		1	DMA write request is enabled.
13-12	TRM		Transmit/receive modes.
		0	Transmit and receive mode
		1h	Receive-only mode
		2h	Transmit-only mode
		3h	Reserved

Table 16-32. McSPI Channel *(i)* Configuration Register (MCSPI_CH(i)CONF) Field Descriptions (continued)

Bit	Field	Value	Description
11-7	WL		SPI word length.
		0-2h	Reserved
		3h	The SPI word is 4-bits long.
		4h	The SPI word is 5-bits long
		5h	The SPI word is 6-bits long
		6h	The SPI word is 7-bits long
		7h	The SPI word is 8-bits long
		8h	The SPI word is 9-bits long
		9h	The SPI word is 10-bits long
		Ah	The SPI word is 11-bits long
		Bh	The SPI word is 12-bits long
		Ch	The SPI word is 13-bits long
		Dh	The SPI word is 14-bits long
		Eh	The SPI word is 15-bits long
		Fh	The SPI word is 16-bits long
		10h	The SPI word is 17-bits long
		11h	The SPI word is 18-bits long
		12h	The SPI word is 19-bits long
		13h	The SPI word is 20-bits long
		14h	The SPI word is 21-bits long
		15h	The SPI word is 22-bits long
		16h	The SPI word is 23-bits long
		17h	The SPI word is 24-bits long
		18h	The SPI word is 25-bits long
		19h	The SPI word is 26-bits long
		1Ah	The SPI word is 27-bits long
		1Bh	The SPI word is 28-bits long
		1Ch	The SPI word is 29-bits long
		1Dh	The SPI word is 30-bits long
		1Eh	The SPI word is 31-bits long
		1Fh	The SPI word is 32-bits long
6	EPOL		$\overline{\text{SPI\_SCS[n]}}$ polarity
		0	$\overline{\text{SPI\_SCS[n]}}$ is held high during the active state.
		1	$\overline{\text{SPI\_SCS[n]}}$ is held low during the active state.

Table 16-32. McSPI Channel *(i)* Configuration Register (MCSPI_CH(i)CONF) Field Descriptions (continued)

Bit	Field	Value	Description
5-2	CLKD		Frequency divider for SPI_SCLK. (only when the module is a Master SPI device). A programmable clock divider divides the SPI reference clock (CLKSPIREF) with a 4-bit value, and results in a new clock SPI_SCLK available to shift-in and shift-out data. By default the clock divider ratio has a power of two granularity when MCSPI_CHCONF[CLKG] is cleared, Otherwise this register is the 4 LSB bit of a 12-bit register concatenated with clock divider extension MCSPI_CHCTRL[EXTCLK] register.
			The value description below defines the clock ratio when MCSPI_CHCONF[CLKG] is cleared to 0.
		0	Divide by 1
		1h	Divide by 2
		2h	Divide by 4
		3h	Divide by 8
		4h	Divide by 16
		5h	Divide by 32
		6h	Divide by 64
		7h	Divide by 128
		8h	Divide by 256
		9h	Divide by 512
		Ah	Divide by 1024
		Bh	Divide by 2048
		Ch	Divide by 4096
		Dh	Divide by 8192
		Eh	Divide by 16384
		Fh	Divide by 32768
1	POL		SPI_SCLK polarity
		0	SPI_SCLK is held high during the active state
		1	SPI_SCLK is held low during the active state
0	PHA		SPI_SCLK phase
		0	Data are latched on odd numbered edges of SPI_SCLK
		1	Data are latched on even numbered edges of SPI_SCLK

Table 16-33. Data Lines Configurations

			TRMi		
ISi	DPEi1	DPEi0	Transmit and Receive	Receive Only	Transmit Only
0	0	0	Supported	Supported	Supported
0	0	1	Supported	Supported	Supported
0	1	0	Supported	Supported	Supported
0	1	1	Not supported (unpredictable result)	Supported	Not supported (unpredictable result)
1	0	0	Supported	Supported	Supported
1	0	1	Supported	Supported	Supported
1	1	0	Supported	Supported	Supported
1	1	1	Not supported (unpredictable result)	Supported	Not supported (unpredictable result)

16.3.13 McSPI Channel (*i*) Status Register (MCSPI_CH(i)STAT)

The McSPI channel *i* status register register (MCSPI_CH(i)STAT) provides status information about the McSPI channel *i* FIFO transmit buffer register (MCSPI_TX*n*) and the McSPI channel *i* FIFO receive buffer register (MCSPI_RX*n*) of channel *i*. The (MCSPI_CH(i)STAT) is shown in Figure 16-37 and described in Table 16-34.

Figure 16-37. McSPI Channel *(i)* Status Register (MCSPI_CH(i)STAT)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved	RXFFF	RXFFE	TXFFF	TXFFE	EOT	TXS	RXS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 16-34. McSPI Channel *(i)* Status Register (MCSPI_CH(i)STAT) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Read returns 0
6	RXFFF		Channel *i* FIFO receive buffer full status.
		0	FIFO receive buffer is not full.
		1	FIFO receive buffer is full.
5	RXFFE		Channel *i* FIFO receive buffer empty status.
		0	FIFO receive buffer is not empty.
		1	FIFO receive buffer is empty.
4	TXFFF		Channel *i* FIFO transmit buffer full status.
		0	FIFO transmit buffer is not full.
		1	FIFO transmit buffer is full.
3	TXFFE		Channel *i* FIFO transmit buffer empty status.
		0	FIFO transmit buffer is not empty.
		1	FIFO transmit buffer is empty.
2	EOT		Channel *i* end-of-transfer status. The definitions of beginning and end of transfer vary with master versus slave and the transfer format (transmit/receive mode, turbo mode).
		0	This flag is automatically cleared when the shift register is loaded with the data from the transmitter register (beginning of transfer).
		1	This flag is automatically set to one at the end of an SPI transfer.
1	TXS		Channel *i* transmitter register status. The bit is cleared when the host writes the most significant byte of the SPI word in the MCSPI_TX(i) register. The bit is set when enabling the channel *i* , and also when the SPI word is transferred from the MCSPI_TX(i) register to the shift register.
		0	Register is full.
		1	Register is empty.
0	RXS		Channel *i* receiver register status. The bit is cleared when enabling the channel *i*, and also when the host reads the most significant byte of the received SPI word from the MCSPI_RX(i) register. The bit is set when the received SPI word is transferred from the shift register to the MCSPI_RX(i) register.
		0	Register is empty.
		1	Register is full.

16.3.14 McSPI Channel *(i)* Control Register (MCSPI_CH(I)CTRL)

The McSPI channel *i* control register (MCSPI_CH*i*CTRL) is used to enable channel *i*. The MCSPI_CH*i*CTRL is shown in Figure 16-38 and described in Table 16-35.

Figure 16-38. McSPI Channel *(i)* Control Register (MCSPI_CH(I)CTRL)

31			16
Reserved			
R-0			
15	**8**	**7**	**1**	**0**
EXTCLK		Reserved		EN
R/W-0		R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-35. McSPI Channel *(i)* Control Register (MCSPI_CH(I)CTRL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-8	EXTCLK		Clock ratio extension. Used to concatenate with the CLKD bit field in MCSPI_CH*n*CONF for clock ratio only when granularity is 1 clock cycle (CLKG bit in MCSPI_CH*n*CONF set to 1). Then the maximum value reached is a 4096 clock divider ratio.
		0	Clock ratio is CLKD + 1
		1h	Clock ratio is CLKD + 1 + 16
		...	...
		FFh	Clock ratio is CLKD + 1 + 4080
7-1	Reserved	0	Reserved
0	EN		Channel *n* enable.
		0	Channel *n* is not active.
		1	Channel *n* is active.

16.3.15 McSPI Channel *(i)* Transmit Register (MCSPI_TX*(i)*)

The McSPI channel *i* transmit register (MCSPI_TX*(i)*) contains a single McSPI word to transmit on the serial link. The (MCSPI_TX*(i)*) is shown in Figure 16-39 and described in Table 16-36.

- Little endian host access SPI 8 bit word on 0; big endian host accesses on 3h.
- The SPI words are transferred with MSB first.

Figure 16-39. McSPI Channel *(i)* Transmit Register (MCSPI_TX*(i)*)

31	0
TDATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 16-36. McSPI Channel *(i)* Transmit Register (MCSPI_TX*(i)*) Field Descriptions

Bit	Field	Value	Description
31-0	TDATA	0-FFFF FFFFh	Channel *i* data to transmit.

16.3.16 McSPI Channel *(i)* Receive Register (MCSPI_RX*(i)*)

The McSPI channel *i* FIFO receive buffer register (MCSPI_RX*(i)*) contains a single McSPI word received through the serial link. The (MCSPI_RX*(i)*) is shown in Figure 16-40 and described in Table 16-37.

- Little endian host access SPI 8 bit word on 0; big endian host accesses on 3h.

Figure 16-40. McSPI Channel *(i)* Receive Register (MCSPI_RX*(i)*)

31	0
RDATA	
R-0	

LEGEND: R = Read only; -*n* = value after reset

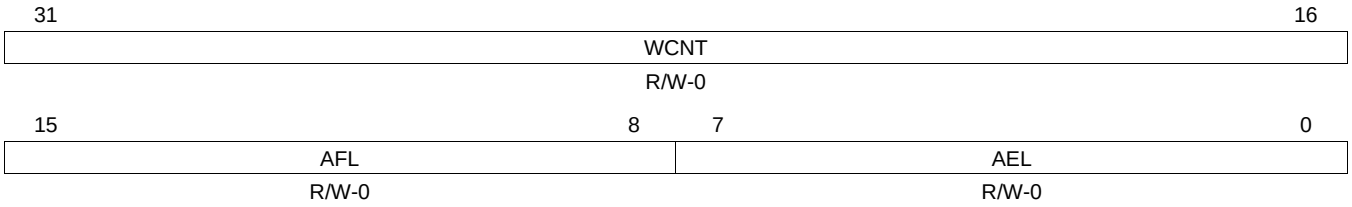
Table 16-37. McSPI Channel *(i)* Receive Register (MCSPI_RX*(i)*) Field Descriptions

Bit	Field	Value	Description
31-0	RDATA	0-FFFF FFFFh	Channel *i* received data.

16.3.17 McSPI Transfer Levels Register (MCSPI_XFERLEVEL)

The McSPI transfer levels register (MCSPI_XFERLEVEL) provides the transfer levels needed while using the FIFO buffer during transfer. The MCSPI_XFERLEVEL is shown in Figure 16-41 and described in Table 16-38.

Figure 16-41. McSPI Transfer Levels Register (MCSPI_XFERLEVEL)



LEGEND: R/W = Read/Write; -*n* = value after reset

Table 16-38. McSPI Transfer Levels Register (MCSPI_XFERLEVEL) Field Descriptions

Bit	Field	Value	Description
31-16	WCNT		SPI word counter. Holds the programmable value of the number of SPI words to be transferred on the channel that is using the FIFO buffer. When the transfer has started, a read back of this register returns the current SPI word transfer index.
		0	Counter not used
		1	1 SPI word
		...	...
		FFFEh	65534 SPI word
		FFFFh	65535 SPI word
15-8	AFL		Buffer almost full. Holds the programmable almost full level value used to determine almost full buffer condition. If you want an interrupt or a DMA read request to be issued during a receive operation when the data buffer holds at least *n* bytes, then the buffer MCSPI_XFERLEVEL[AFL] must be set with n - 1.
		0	1 byte
		1	2 bytes
		...	...
		FFh	256 bytes
7-0	AEL		Buffer almost empty. Holds the programmable almost empty level value used to determine almost empty buffer condition. If you want an interrupt or a DMA write request to be issued during a transmit operation when the data buffer is able to receive *n* bytes, then the buffer MCSPI_XFERLEVEL[AEL] must be set with n - 1.
		0	1 byte
		1	2 bytes
		...	...
		FFh	256 bytes

16.3.18 DMA Address Aligned FIFO Transmitter Register (MCSPI_DAFTX)

The DMA address aligned FIFO transmitter register (MCSPI_DAFTX) contains the SPI words to transmit on the serial link when the FIFO used and the DMA address is aligned on 256 bits. MCSPI_DAFTX is an image of one of MCSPI_TX*(i)* register corresponding to the channel that has its FIFO enabled. The MCSPI_DAFTX is shown in Figure 16-42 and described in Table 16-39.

- Little endian host access SPI 8 bit word on 0; big endian host accesses on 3h.
- The SPI words are transferred with MSB first.

Figure 16-42. DMA Address Aligned FIFO Transmitter Register (MCSPI_DAFTX)

31	0
DAFTDATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 16-39. DMA Address Aligned FIFO Transmitter Register (MCSPI_DAFTX) Field Descriptions

Bit	Field	Value	Description
31-0	DAFTDATA	0-FFFF FFFFh	FIFO data to transmit with DMA 256 bit aligned address. This register is only is used when MCSPI_MODULCTRL[FDAA] is set to 1 and only one of the MCSPI_CH(i)CONF[FFEW] of enabled channels is set. If these conditions are not respected, any access to this register returns a null value.

16.3.19 DMA Address Aligned FIFO Receiver Register (MCSPI_DAFRX)

The DMA address aligned FIFO receiver register (MCSPI_DAFRX) contains the SPI words to receive on the serial link when the FIFO used and the DMA address is aligned on 256 bits. MCSPI_DAFRX is an image of one of MCSPI_RX*(i)* register corresponding to the channel that has its FIFO enabled. The MCSPI_DAFRX is shown in Figure 16-43 and described in Table 16-40.

- Little endian host access SPI 8 bit word on 0; big endian host accesses on 3h.

Figure 16-43. DMA Address Aligned FIFO Receiver Register (MCSPI_DAFRX)

31	0
DAFRDATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 16-40. DMA Address Aligned FIFO Receiver Register (MCSPI_DAFRX) Field Descriptions

Bit	Field	Value	Description
31-0	DAFRDATA	0-FFFF FFFFh	FIFO received data with DMA 256 bit aligned address. This register is only is used when MCSPI_MODULCTRL[FDAA] is set to 1 and only one of the MCSPI_CH(i)CONF[FFER] of enabled channels is set. If these conditions are not respected, any access to this register returns a null value.

Peripheral Component Interconnect Express (PCIe)

This chapter describes the PCIe Subsystem (PCIESS). The PCIESS provides a high-speed glueless serial interconnect to peripherals utilizing high bandwidth applications.

Topic **Page**

17.1 Introduction

17.1.1 Overview

The PCI Express (PCIe) module is a multi-lane I/O interconnect that provides low pin count, high reliability, and high-speed data transfer at rates of up to 5.0 Gbps per lane per direction, for serial links on backplanes and printed wiring boards. It is a 3rd Generation I/O Interconnect technology succeeding PCI and ISA bus that is designed to be used as a general-purpose serial I/O interconnect in multiple market segments, including desktop, mobile, server, storage and embedded communications. It is also used as a bridge to other interconnects like SATA, USB2/3.0, GbE MAC, and so on.

The PCIESS contains the Synopsys DesignWare core (DWC) PCIe Dual Mode core and Texas Instruments SERDES PHY. The Dual Mode (DM) core operates as either Endpoint (EP) mode or Root Complex (RC) Port mode. The core supports a single In-Port and a single Out-Port. The role of the PCIe core, EP or RC, is configured based on the value sampled from the BOOTMODE[4:0] pins and by application software configuring PCIE_CFG[PCIE_DEVTYPE] register.

The incumbent design, PCI, is a parallel bus architecture that is increasingly difficult to scale-up in bandwidth, which is usually performed by increasing the number of data signal lines. More signal lines result in difficult clock-to-data skew management, creating complex PCB layout rules that make cost-effective implementations in the FR4 technology difficult. Increasing the number of signal lines also increases the power dissipation. The PCIe architecture was developed to help minimize I/O bus bottlenecks within systems and to provide the necessary bandwidth for high-speed, chip-to-chip, and board-to-board communications within a system. It is designed to replace the PCI-based shared, parallel-bus-signaling technology that is approaching its practical performance limits while simplifying the interface design. It includes cost, performance, and scalability advantages ensuring a long life span existence to applications, system designs and investments.

PCIe is a serial-based technology which uses low-voltage differential data signaling/lines (LVDS) to reduce the number of data signal lines and high-frequency clock signals in a point-to-point interconnect arrangement between two devices. It also serves to eliminate multiple host presences on the same bus.

17.1.2 Features

The PCIESS on the device supports an interface width of ×2 lanes. The following features are included:

- 250 MHz functional clock frequency operation (PIPE clock frequency)
- Supports a single bi-directional Link interface (that is, a single Ingress and a single Egress ports). Can directly connect to one device only with the possibility of 1 or 2 bi-directional lanes.
- Operates at a raw speed of 2.5Gbps or 5.0 Gbps per Lane per Direction
- Maximum outbound payload size of 128 bytes
- Maximum inbound payload size of 256 bytes
- Maximum remote read request size of 256 bytes
- Ultra-low transmit and receive latency
- Support for dynamic-width conversion
- Automatic Lane reversal
- Polarity inversion on receive
- Single Virtual Channel (VC)
- Single Traffic Class (TC)
- Automatic credit management
- Single Function in Endpoint (EP) mode
- ECRC generation and checking
- PCI Device Power Management with the exception of D3 cold with Vaux
- PCI Express Active State Power Management (ASPM) state L0s and L1
- PCI Express Link Power Management states except L2 state
- PCI Express Advanced Error Reporting

- PCI Express messages for both transmit and receive
- Filtering for Posted, Non-Posted, and Completion traffic
- Configurable BAR filtering, I/O filtering, configuration filtering and Completion lookup/timeout
- Access to configuration space registers and external application memory mapped registers through BAR0 and through configuration access
- Legacy Interrupts reception (Root Complex (RC)) and generation (EP)
- MSI generation and reception
- PHY Loopback in RC mode

17.1.3 Features Not Supported

The following features are not supported by the PCIESS:

- Multiple VCs
- Multiple TCs
- Outbound transactions involving less than 4 bytes
- Function Level Reset
- PCI Express beacon for in-band wake
- Built-in hardware support for hot plug
- Vendor Messaging
- IO access in inbound direction
- Addressing modes other than incremental for burst transactions
- Using ×2 link as two ×1 links
- Auxiliary Power to maintain controller state to come out D3cold state
- Link state L2 support

17.1.4 Functional Block Diagram

The PCIESS is shown in Figure 17-1.

17.1.4.1 PCIe Core

The PCI Express core implements the three PCI Express protocol layers (Transaction Layer, Data Link Layer, and the MAC portion of the Physical Layer). The PCIe core is a Dual Mode core allowing it to operate either as an RC or as an EP. As an EP, it can operate as a Legacy Endpoint or native PCIe Endpoint. The role it assumes is based on the state of the PCIe Configuration register (within the Control Module) bit field PCIE_CFG [1:0] PCIE_DEVTYPE = 00b/ 01b/ 10b = EP/ Legacy Endpoint/ RC respectively. The user software or BOOTROM is required to initialize this field. During times PCIe is used as a Boot Device, then the state of Bootmode Pins will determine the Boot Type used (in this case it will be PCIe) and the address size to be used which is either 32bit or 64bit addressing. The bootloader software configures the PCIe core to operate in EP mode. No RC Bootmode is supported.

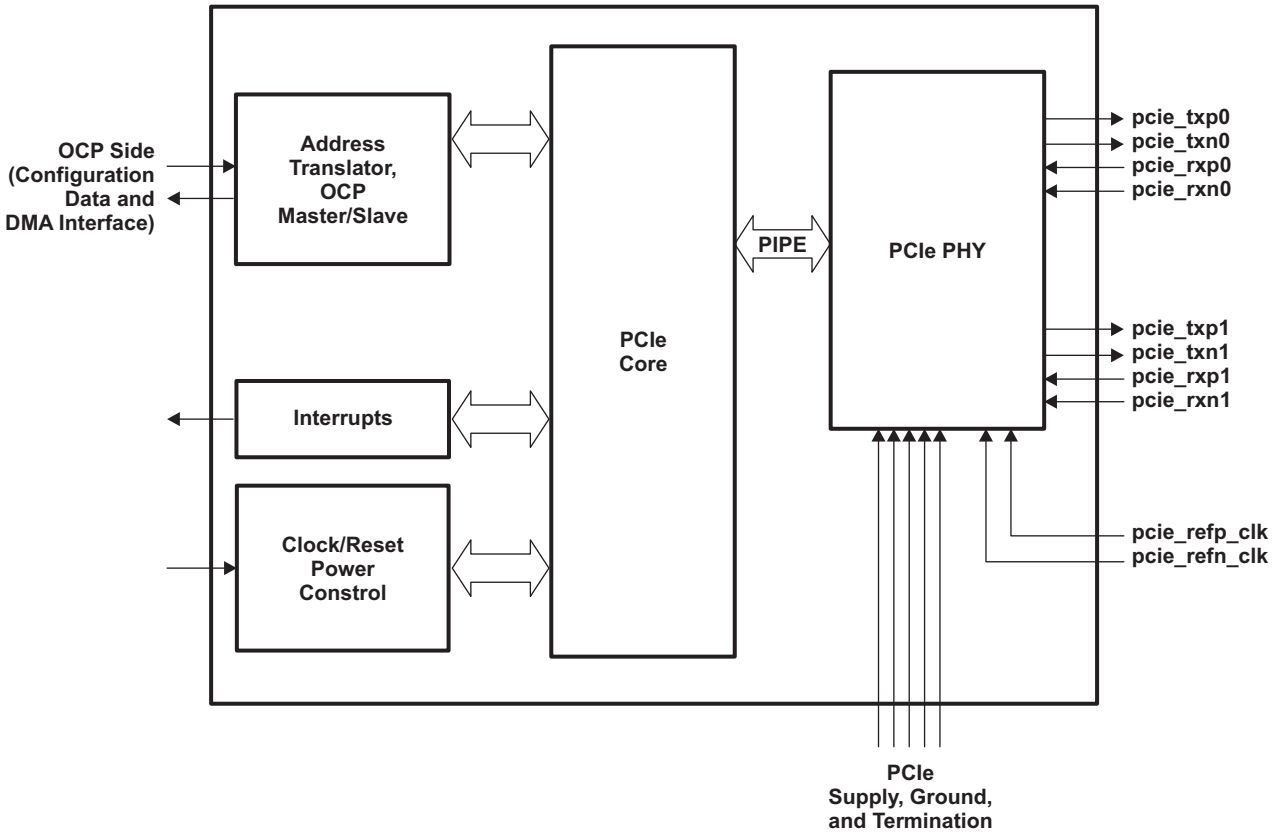
17.1.4.2 PCIe PHY

The PCI PHY (SERDES) contains the analog portion of the PHY which is the transmission line channel that is used to transmit and receive data. It contains a phase locked loop, analog transceiver, phase interpolation based clock/data recovery, parallel to serial converter, serial to parallel converter, scrambler, configuration and test logic.

17.1.4.3 OCP (Configuration and DMA Access Interface)

CPU side access, Configuration Registers, Data Access, and Remote EP DMA access is performed through the Configuration and DMA Access interface. The maximum amount of data burst access made through this port is 128 bytes.

Figure 17-1. PCIe Subsystem (PCIESS) Block Diagram



17.1.4.4 Clock, Reset, Power Control Logic

Several clock domains exist within PCIESS. These clocks are functional clocks used by the PCIe controller and interface bridges as well as receive and transmit clocks used to clock data in and out respectively. The clocks required for clocking data and PHY functional clocks are generated by the PHY through the supplied external input 100 MHz differential clock with no more than 300ppm tolerance. The PCIe controller functional clock frequency is generated from the internal PLL and should be programmed to generate a clock frequency of 250MHz.

The PCIESS supports the Conventional Reset mechanism that is specified within the PCI Express Specification. The reset shown on the block diagram pertains to a hardware reset (cold or warm reset).

In addition to automatic power down mode exercised and entered by the hardware (Active State Power Management) when no activity is present, PCIESS supports higher level power down modes controlled (activated and deactivated) by application software.

NOTE: Link state L2 is not supported.

17.1.4.5 Interrupts

The PCIESS is capable of generating multiple interrupts events via the four interrupt lines (INTA/B/C/D – Legacy Interrupts Combined, MSI, Error, and PM/Reset), connected to the Interrupt Controller. The user software is required to acknowledge the serviced interrupt by writing to the corresponding vector onto the EOI register.

17.1.4.6 PCIe Power/Ground/Termination

Several supplies, grounds, and termination exist to properly power up the PHY.

17.1.4.7 Differential Data Lines

A pair of differential data lines exists, for both transmit and receive paths, for each lane.

17.1.5 Supported Use Case Statement

The PCIe subsystem has only one interface link and this link can be used in a ×1 or ×2 lane arrangements connecting to only one single device. In other words, there exists no support for connecting to two devices in an ×1 arrangement since it has the support of a single interface link. It also means that it cannot be used as a switch. Lane 0 must always be connected and functioning, either alone or in conjunction with Lane 1, for the link to be established. Lane 1 cannot be used alone.

17.1.6 Industry Standard(s) Compliance Statement

The PCIESS complies with the following standards:

- Revision 2.0 of the PCI Express Base Specification
- Synopsys DWC PCIe Dual Core Version 3.51a
- TI SERDES 1.1.03

17.1.7 Terminology Used in this Document

The following is a brief explanation of some terms used in this document:

ADPLL— All Digital Phase Locked Loop

ASPM— Active state power management

AXI— AMBA AXI Bus Protocol

DBI— Direct Bus Interface

EP— End Point

LVDS— Low Voltage Differential Signaling

MMR— Memory Mapped Register

OCP— Open Core Protocol

OCM— On-Chip Memory

OCMC— On-Chip Memory Controller

PCI— Peripheral Component Interconnect

PCIe— PCI Express

PCIESS— PCI Express subsystem

PCISIG— PCI Special Interest Group

PIPE— Physical Interface for PCI Express

PME— Power Management Event

RC— Root Complex

TC— Traffic Class

TLP— Transaction Layer Packet

VC— Virtual Channel

VPD— Vital Product Data

17.2 Architecture

This section discusses the architecture of the PCIESS.

17.2.1 Clock Control

The PCIESS uses multiple clock domains. At the top level, there are only two functional clocks of relevance – a reference clock to the PCIE PHY and a clock for OCP interfaces of the subsystem.

The OCP interface functional clock is generated from the main PLL which in turn uses the 27-MHz crystal input. It is important that the PLL is programmed appropriately in such a way that a 250-MHz functional clock is present at SYSCLK5.

The other clock input to the PCIESS is the differential 100-MHz clock with a maximum frequency of deviation value of 300 ppm. This differential clock is used by the PHY PLL to generate the necessary functional and bit clock required by the PHY.

The PCIESS operation is completely dependent upon availability and stability of the clocks, more importantly the accuracy of the clock generated from the PLL that is inside the PHY. All registers in PCIESS are located in the clock domain that is dependent upon PLL proper configuration and lock status state prior to placing the PCIe in active state. Transactions initiated before ensuring PLL lock status will cause the PCIe subsystem to operate in an unpredictable manner. The SoC-level registers are used to enable PLL and to verify lock status.

17.2.2 Supported PCIe Transactions

All of the PCIe Transactions defined, Posted and Non-Posted, are supported except the Locked Memory Read request transaction and its subsequent completion Locked response transaction. In-bound I/O read/write transactions are also not supported.

Table 17-1. PCIe Transaction Layer Packets Supported

Transaction Packet Types	Posted/Non-posted
Memory Read	Non-posted
Memory Write	Posted
IO read (Outbound only supported in RC Mode)	Non-posted
IO Write (Outbound only supported in RC Mode)	Non-posted
Configuration Read (Type 0 and Type 1)	Non-posted
Configuration Write (Type 0 and Type 1)	Non-posted
Message Request without Data	Posted
Message Request with Data	Posted
Completion without Data	-
Completion with Data	-

17.2.3 Address Translations

PCI Express TLP transactions using Address Routing use PCIe addresses. To accommodate the mapping requirement that exists between a PCIe address and a local OCP/Internal address, a built-in hardware address translation mechanisms exists. That is, if the “Type” field of an outgoing or received TLP indicates the use of address routing, then an Outbound or Inbound address translation is required and is performed accordingly to map OCP/Internal address to PCIe address or vice-versa using hardware address translators. Address translations for Outbound and Inbound transactions are discussed below.

PCIe recognizes four address spaces, Memory, I/O, Configuration, and Message. Messages do not consume any Memory or I/O resource and no type of translation is required. I/O addressing is used by PCIe (only supported in RC configuration) in outbound access to a Legacy EP; this is not a mandatory feature by PCIe and when supported the I/O spaces is memory mapped to system memory. This implies that there exists a need to map only two types of device address spaces, Configuration and Memory address spaces. The address translators apply to translating Configuration and Memory addresses between PCIe address and OCP/Internal address.

The address space of the physical device is divided into two spaces (Range 0 and Range 1). Address space (Range 0) that is used for PCI configuration tasks, referred to as configuration space; and a second Address space (Range 1) that is used for accessing memory (non-configuration related), referred to as memory space.

NOTE: All I/O access use Address Routing. There is no-support for I/O access in Inbound direction. That is, when it comes to a transaction that performs I/O access only TLPs with Outbound transactions are supported.

17.2.3.1 Outbound Address Translations

Outgoing Transaction Layer Packets (TLP) associated with “Address Routing” would require the use of the Outbound address translator that creates a mapping between the OCP/Internal memory/address and external PCIe device memory/address on the PCIe fabric.

The PCIE Subsystem allows mapping of Physical device address to PCIe address on outgoing TLPs. This is accomplished by using outbound address translation logic. For each outbound read/write request, the address translation module within PCIESS converts an OCP address (OCP/Internal address) to a PCIE address (PCIe address) of Memory Read/Write type. The address translation logic uses information programmed within address translation registers to perform the mapping. The registers, OB_SIZE, OB_OFFSET_INDEXn, OB_OFFSETn_HI are used in conjunction with Out-bound address translator.

The physical memory range that PCIESS occupies in the devices internal address range is divided into 32 equally sized translation regions (Regions 0 to 31). These equally divided regions can be programmed to have a size of 1, 2, 4, or 8 MB and this size value is communicated with the Outbound Address Translator via the OB_SIZE register. Each such region (OCP/Internal address) can be remapped to a PCIE address range of same size as the size of translation region itself. The address translation logic identifies and extract the 5 bits (32 regions) from the OCP/Internal address and uses this value as an index to identify one of the 32 regions. The bit address position for these 5 bits depends on the size of the regions. Once the region is identified, the Address translation logic then generates PCIe base address, from the values provided within the corresponding configuration registers for that region, that is, the registers OB_OFFSET_INDEXn and OB_OFFSETn_HI [n = 0-31]. If 32-bit addressing is used, OB_OFFSETn_HI will always be programmed with zero.

Once the PCIe base address has been identified, the offset that is to be added to this base address is derived from the lower bit fields of the OCP/Internal address and the bit fields that make up this offset correspond to the size of the regions.

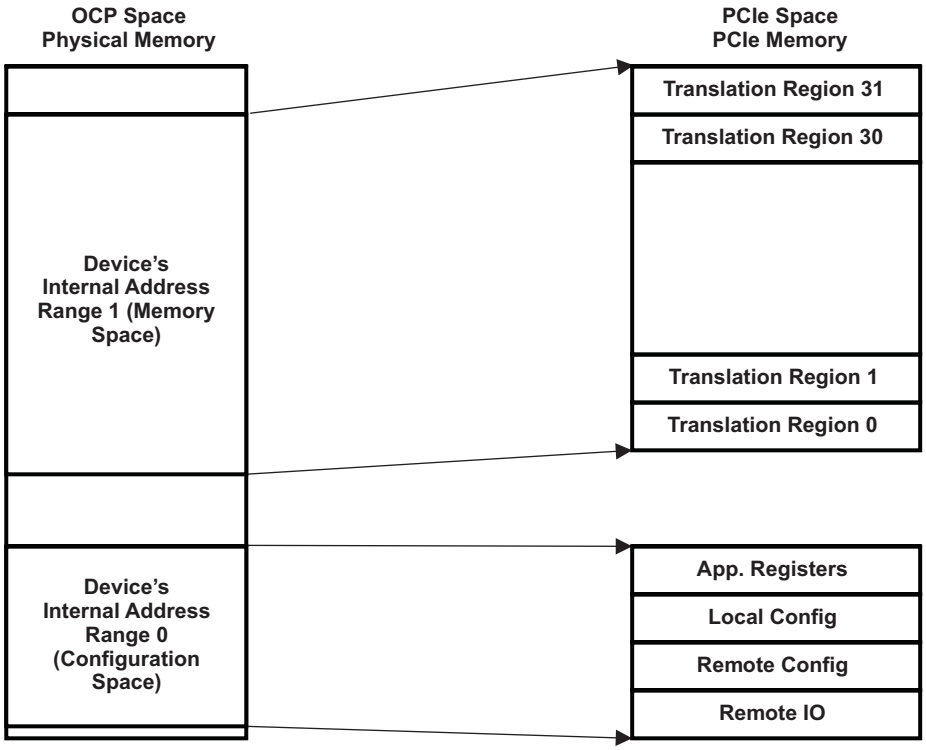
Application software is required to identify the desired physical memory that is to be accessible by the PCIe module and initialize the corresponding registers prior to enabling PCIe transactions. For Outbound translation the registers in use, initialization, and usage is discussed below.

- Outbound Size Register (OB_SIZE) — Application software initializes this register with the size value that applies to all 32 regions. OB_SIZE=0,1,2, and 3 correspond to region sizes of 1MB, 2MB, 4MB, and 8MB and the corresponding indexed regions for these sizes are identified from the OCP/Internal address. Bits[24:20], bits[25:21], bits[26:22], and bits[27:23] of the OCP/Internal address holds the Index value used to identify one of the 32 regions for the 1MB, 2MB, 4MB and 8MB regions respectively. Size of regions also affect the meaningful bit fields within OB_OFFSET_INDEXn [n=0-31] register. The lower bit fields corresponding to the size of the region are masked and not used in the mapping.
- Outbound Translation Region n Offset Low and Index Register (OB_OFFSET_INDEXn [n = 0-31]) — Application software initializes this register with the 32-bit PCIe address. The useful address bits used by the Outbound translator from this register depends upon the size of the region. If each region is 1MB, 2MB, 4MB, or 8MB, then the relevant address field bits used to create PCIe address are the

higher address bits, specifically bits[31:20], bits[31:21], bits[31:22], and bits[31:23], respectively.

- Outbound Address High Offset (OB_OFFSETn_HI [n = 0-31]) – Application software initializes this register if using 64-bit addressing with the higher 32-bit address. This register is required to be programmed with a Zero value if using 32-bit addressing.

Figure 17-2. Outbound Address Translation



The following example demonstrates the mapping of a given OCP/Internal address to a PCIe address.

Example 1:

For a given OCP/Internal Address of 9D3A_1234h; what would be the corresponding PCIe Address that would be used on an outgoing TLP Header (assume a 2MB region partition)?

For this example, further assume the followings: 64 bit addressing is to be used and for region 9, and application software has initialized OBOFFSET9_HI with 3344 5566h and OB_OFFSET9 with 56Ex xxxxh (where 'x' here is a don't care hexadecimal value). Note that bits[31:21] of OB_OFFSET9 register is the part of this register content that is meaningful for this example since the size of the regions is 2MB.

The procedure that is used to convert this OCP/Internal address to a PCIe address is as follows.

1. Extract the five bits Index and Offset from given input OCP/Internal address of 9D31 1234h: Since region size is given to be 2MB, bits [25:21] of address 9D3A 1234h is extracted to be used as an Index and this comes out to be 01001b, which is 9. Since size of all regions is 2MB, the offset address corresponds to bits[20:00] of the OCP/Internal address. Note that the offset of the OCP/Internal address, bits[20:00], is directly mapped to the PCIe address; that is, bits[20:00], and for this example that translates to XX1A 1234h. Since Index value is 9, translation registers contents for region 9 will be used.

   Index is now matched to Region 9 and Offset is 001A 1234h.
2. Generate PCIe Base address: Using the Index value identified above, 9, generate base address from register OBOFFSET9_HI (bits[63:32]) for 64-bit addressing and register OB_OFFSET9 (bits[31:20]). The PCIe Base Address for this example would results with: 3344 5566 56E0 0000h.
3. Compute translated address, PCIe address: Using the generated Base Address and Offset the PCIe address corresponding to a OCP/Internal address of 9D3A 1234h is 3344 5566 56FA 1234h.

NOTE: For this example, 2MB size region will allow unique address mapping for Physical Address values 0000 0000h to 03FF FFFFh. This allows a total of 2MB × 32 = 64MB unique Physical Address location to be mapped onto a corresponding unique virtual PCIe address. Any Physical address outside this 64MB space will not be uniquely mapped and the Physical address will be truncated to fall in within the 64MB space, which happens to be for this example. In other words, bits[31:26] of the given example Input/Physical Address is masked and did not contribute to the PCIe Address generation.

For this example, OCP/Internal addresses 9D3A 1234h, 913A 1234h, 953A 1234h, 993A 1234h, etc., are few of other OCP/Internal Physical values that will all map to the same PCIe address 3344 5566 56FA 1234h.

17.2.3.2 Inbound Address Translations

Incoming error free accepted Transaction Layer Packets (TLP) associated with “Address Routing” would require the use of the In-bound address translator to map received PCIe address to OCP/Internal address.

The PCIe Subsystem allows mapping of accepted PCI Express addresses to a Physical device address internal to the device using the Inbound Address Translation logic. For each Inbound read/write request, that results with incoming TLP, the address translation module within PCIESS can convert a PCIe address to OCP (internal/physical) address for Memory or Configuration Read/Write type transactions.

PCIESS is aware of two OCP Address Spaces (Internal/Physical Memory Spaces) within the device. The first address space, “Address Space Zero” (also known as region 0), is dedicated for local application registers, local configuration accesses, remote configuration accesses and remote IO accesses (RC only). The second, address space, “Address Space One” (also known as region 1), is dedicated for data transfer. Details on this are captured in following sections.

Address Space Zero occupies a contiguous 16K Bytes of location. The first 4KB of this 16KB Space is the Configuration Space. “Address Space One” is used for data buffering and is large in size and not necessarily contiguous. To perform a mapping of a PCIe addresses that would land within Address Space One, four dedicated Regions, (Regions [0-3]) are available for the use of the Inbound Address Translator. Note that Outbound translation has 32 regions while Inbound translation has four regions.

This means that any Accepted TLP that has found a match with one of the BARs, BAR[0-5] will be mapped to one of the two address spaces, Address Space Zero or Address Space One. BAR0 is dedicated to Address Space Zero and implicit mapping is done and no region association is required. However BARs[1-5] are dedicated to Address Space One and association with one of the four regions, Regions[0-3], is required. Note that this is valid in the context of an accepted TLP with 32-bit Addressing. If 64-bit Addressing is used, then the association of the six BARs with the Address Spaces change to a total of three since a pair of adjacent BARs concatenated is required to hold the 64-bit Address. This implies that BAR0 and BAR1 will hold 64-bit Address with BAR0 holding the low 32-bit PCIe address to match while BAR1 holds the high 32-bit PCIe address to match (associated with Address Space Zero). The same holds for Address Space One association. BARs[2-3] and BARs[4-5] will hold the accepted 64-bit address that will be associated to Address Space One where mapping takes place between the BARs and Regions[0-3].

Address Translation for Address Space Zero does not exist since the internal location is unique and is contiguous. All is needed is the PCIe Address within the received TLP PCIe address matching BAR0, for 32-bit addressing, and BARs[0:1] for 64-bit addressing.

However, Address Translation for Address Space One requires the use of one of the four Regions (Regions[0-3]) to map accepted TLPs to Internal/Physical memory address. Four region specific memory mapped registers exist and used by the Inbound Address Translator.

Association between regions (Regions[0-3]) and BARs is made via the corresponding IB_BARn [n=0-3] register. IB_STARTn_HI [n = 0-3] and IB_STARTn_LO [n = 0-3] are used to hold the start address of a 64-bit PCIe Address to match, and IB_OFFSETn [n = 0-3] is used to hold the Offset for that particular Region (note that what is referred here as offset could be viewed as the physical Memory Base Address). Note also that for 32-bit addressing, IB_STARTn_HI[0-3] is programmed to zero.

The procedure used by the In-Bound Address Translator to perform the mapping between the PCIe Address and Address Space One is as follows:

1. Extract the Offset: PCIe Address – (IB_STARTn_HI : IB_STARTn_LO)
2. Compute absolute OCP/Internal address: Add Base Address IB_OFFSETn to the Offset extracted on step 1.

NOTE: When using TLPs with 64-bit addressing a pair of adjacent BARs are concatenated to hold the 64-bit Address. This implies that BAR0 and BAR1 will hold the 64-bit Address with BAR1 holding the high 32-bit PCIe address to match and BAR0 holding the low 32-bit PCIe address to match. Same argument holds for BAR3, BAR4 and BAR5, BAR6 where BAR3 and BAR5 hold the low 32-bit of the 64-bit addresses.

Example 2:

For a given 64-bit PCIe Address of 1234 5678 ABC5 0000h, that qualifies for acceptance; what would be the corresponding mapped Internal (Physical) Device Address be? (Assume that Region 1 registers are programmed to match BAR2, BAR3 RC programmed addresses).

Further assume that application software has programmed the set of registers corresponding to Region 1 with the values shown below.

- IB_BAR1 = 2. This assignment associates Region 1 with BAR2, or BAR3 for 64-bit Addressing. This value programmed as 2 associates the match between Region 1 (IB_BAR1) and Configuration Register BAR2/BAR3.
- IB_START1_HI = 1234 5678h
- IB_START1_LO = ABC0 0000h
- IB_OFFSET1 = 3340 0000h

The OCP/Internal address is computed by Subtracting the PCIe Base Address and extracting the Offset from the PCIe Address (Address within the TLP Header) and add the resultant to the start Address of the OCP/Internal address.

Extract the Offset from the PCIe Address:

- Extracted Offset = PCIe address – (IB_STARTn_HI : IB_STARTn_LO) => Extracted Offset = 1234 5678 ABC5 0000h – 1234 5678 ABC0 0000h = 0005 0000h

Compute the absolute OCP/Internal address:

- Add the extracted Offset to the Internal Base Address:

  Internal/Physical Absolute Address = 3340 0000h + 0005 0000h = 3345 0000h

In example 2, PCIe Address of 1234 5678 ABC5 0000h is translated or mapped to OCP/Internal address, 3345 0000h.

17.2.3.2.1 BAR0 Exception for In-Bound Address Translation

The memory space covered by BAR0 in inbound direction is completely dedicated to accessing the application registers, Address Space Zero. It implies that the BAR0 cannot be remapped to any other location but to application registers. Any remote inbound access matching the BAR0 region will automatically be routed to these registers. It allows RC devices to control EP devices in absence of dedicated software running on EP. For RC and EP, this mapping of BAR0 to registers allows the message signaled interrupts to work. There is no support to disable BAR0 accesses from reaching the application registers.

17.2.3.2.2 Mapping Multiple Non-contiguous Memory Ranges to One Region

A single inbound TLP address translation region (matching RC programmed PCIe BAR Address) can be used to map accesses to multiple non-contiguous internal address ranges, within Address Space One, by ensuring that the start addresses of these regions are programmed in ascending order in the inbound start address registers.

Example 3:

In this example, two BARs (BAR1 and BAR2) are remapped to four separate locations (four regions) that are non-contiguous using 32-bit addressing. The first two regions (Region 0 and Region 1) remap accesses landing in BAR1 space and the following two regions (Region 2 and Region 3) remap accesses landing in BAR2 space. Each BAR is treated as a 32-bit BAR.

Assumed RC programmed Value of Base Address Registers in Configuration Space:

- BAR1: 1111 0000h
- BAR2: 2222 0000h

NOTE: In case there are more than one matching BAR, then the highest start address that is greater than the TLP address is the one that is considered for translation.

Table 17-2. Example Demonstrating the Mapping of Non-contiguous Memories to a Single Region

Region (IB_BAR)	BAR	IB_Start_LO	IB_Offset	TLP Address	Translated Address
0	1	1111 0000h	3333 0000h	1111 0080h	[1111 0080h - 1111 0000h] + 3333 0000h = 3333 0080h
1	1	1111 8000h	4444 0000h	1111 9000h	[1111 9000h – 1111 8000h] + 4444 0000h = 4444 1000h
2	2	2222 0000h	5555 0000h	2222 0400h	[2222 0400h – 2222 0000h] + 5555 0000h = 5555 0400h
3	2	2222 0800h	6666 0000h	2223 1000h	[2223 1000h – 2222 0800h] + 6666 0000h = 6667 0800h

17.2.3.2.3 Using BAR1 Value as Start Address

The inbound address translation for BAR1 has additional capability of using the value of BAR1 register (from PCIe Configuration Space) as the start address for inbound address translation. This feature can be activated by leaving the start address of the corresponding inbound translation region (one of the four Regions associated to BAR1) programmed with zero. When an incoming read/write access matches BAR1 and the inbound region's BAR match (IB_BARn) is set to one with start address programmed to zero, the BAR1 value is used to compute the translated address. Note that if this feature is used, only one inbound translation window is available and it will be relative to BAR1 programmed value. No other BARs or inbound translation windows can be used if BAR1 is designated as the reference for inbound address translation.

17.2.3.2.4 BAR Mask Registers

The PCI Configuration Register BARs in End Point devices are programmed by RC during the PCI Configuration process while performing enumeration, device discovery. Since the PCIESS is a Dual Mode (DM) core, prior to the start of PCI Configuration, the PCIESS EP device firmware/application software has the opportunity to modify the behavior of BAR registers prior to RC starts the enumeration process and assignment of PCIe Base Addresses. Using the BAR Mask Registers which are overlaid on the BAR Registers, the software on the EP device can configure the amount of address space that the EP will request from the RC during configuration for each of its BARs. The software can also modify the BAR types and can enable/disable the BARs. That is, the read only bit fields of the Configuration Register BARs are user programmable and is required to be programmed prior to enumeration or device discovery process begins.

Note that the BAR Mask Registers are accessible, for configuration purposes of the BARs, only when operating as EP and at the same time only when CMD_STATUS[DBI_CS2] is set/ enabled. Firmware needs to always read back the data written after modification to ensure that the write has completed since the confirmation read will not happen until the write completion insuring the register update is made prior to use. Firmware needs to clear CMD_STATUS[DBI_CS2] after initial configuration prior to RC starts enumeration.

It is important to not attempt modification of BAR Mask Registers from serial link side as unpredictable behavior may occur and the system could become unstable and even worse unusable.

17.2.3.3 Address Alignment Requirement

As per PCIe standard, no transaction can cross a 4 KB-aligned address boundary. The limit on transaction size on the internal bus interface (OCP) is 128 bytes. So, the transactions must be 128 byte or smaller on the internal bus interface. If a transaction received in inbound direction crosses a 128 byte aligned address, then the PCIESS master interface can split such transaction into multiple transactions at the 128 byte boundary.

17.2.3.4 Byte Strobe Usage Requirements

For any type of write transactions, the byte enables can only have a single unbroken string of 1s. In other words, in a transaction, if a byte's write strobe is set, then all following bytes must have write strobe set until the last byte with write enabled. "Holes" or "Zeros" in between the byte enables are not allowed.

Since the internal bus width is greater that 32-bit, the TLP size will not be 1 (PCIe counts in 32-bit units) and therefore, it is through the FBE/LBE (First/Last Byte Enable) that the actual data transfer size is controlled.

17.2.3.5 Zero-Length Read/Write Transactions

Read transactions that request zero bytes are not supported by PCIESS. PCIESS will read issue a read with FBE field of PCIe TLP set to Fh. Writes of zero bytes are supported.

17.2.3.6 Transactions Crossing 4KB Boundary

Any single transaction that reads/writes data on locations crossing a 4KB aligned address are invalid as per PCIe Standard Section 2.2.7. If such a read is issued to PCIESS Slave, it will lead to two transactions on the serial link so that PCIe protocol is not violated. But if the completions do not return in order from remote PCIe link partner, then it may result in violation.

17.2.3.7 Outbound Transactions Involving Less than 4 Bytes

The PCIESS imposes a limitation on the minimum size of data, 4-bytes wide, to be accessed in outbound transactions. It is not possible to access less than 4 bytes. This could potentially be a problem when accessing non-prefetchable memory, resulting in accesses to an undesired memory location.

17.2.3.8 Transaction Address Alignment

The PCIESS imposes a limitation of a maximum of 128 byte outbound read/write command. However, if the starting address is not aligned to an 8 byte boundary, then the maximum transaction size is reduced to 120 bytes. Unspecified behavior will occur if misaligned transactions in outbound direction are not limited to a maximum of 120 bytes.

17.2.3.9 Read Interleaving

Read interleaving refers to the process of returning split read responses from multiple transactions. This implies that read data is not guaranteed to be sent in sequential order (data for one transaction to be sent completely before the next). PCIE core is guaranteed to not interleave read responses if the outbound read command/transaction size does not exceed the max transaction size configured in the PCIE core. Currently this is configured as 128 bytes.

17.2.3.10 Endian Mode

The PCIESS is configured to operate in Little Endian mode. Most of the peripherals within the devices are also configured in Little Endian mode.

17.2.4 Address Spaces

Internal Device addressable resources, from PCIESS perspective, are categorized in to two separate spaces. That is, PCIESS has two OCP Address Spaces or Regions. The first is dedicated for local application registers, local configuration accesses, remote configuration accesses and remote IO accesses (RC only). This space is referred as Address Space Zero. The second is dedicated for data transfer and it is referred as Address Space One.

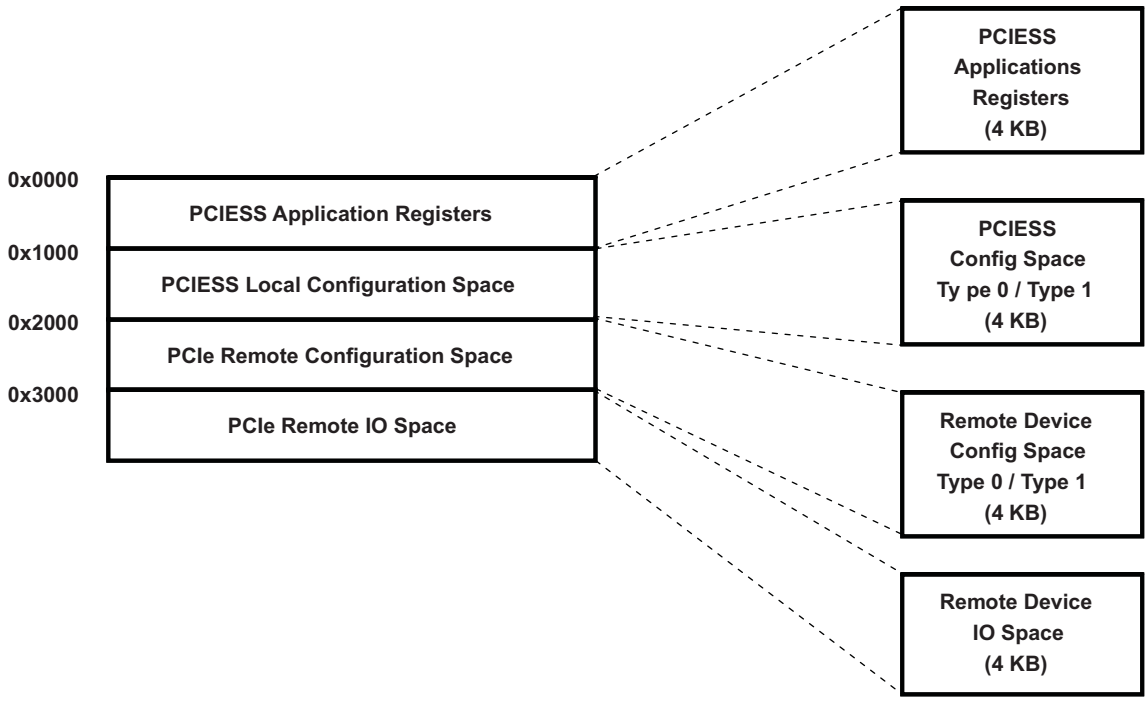
17.2.4.1 Address Spaces Zero

Address Space Zero is made of a contiguous 16 Kbytes of memory partitioned into four sections/regions with equal sizes (each 4 Kbytes long). These regions are:

1. PCIESS Application Registers
2. PCIe Local Configuration Registers
3. PCIe Remote Configuration Registers
4. PCIe IO Access Window

Figure 17-3 illustrates the relationship of the various address regions within the Address Space Zero.

Figure 17-3. Address Space Zero Relationships



Each of the four addresses space Zero regions is meant for accessing a set of registers.

1. PCIESS Application Registers – These registers are used to configure and monitor various settings within the PCIESS. These registers are specific to the application needs and are not related to the PCIE Configuration Registers. Note that all PCIESS Application registers should be accessed in 32-bit mode.

2. PCIe Local Configuration Registers – PCIESS local configuration registers are used to read the settings of configuration registers of the local PCIE Device. Prior to PCI Express configuration is complete, these registers can also be written to. Depending upon whether PCIESS is configured as RC or EP, the layout of these registers is either Config space Type 0 or Type 1. Note that all accesses on PCIe Local Configuration Registers must be made in 32-bit mode. In EP Mode, this is Type 0. In RC Mode, this is Type 1.
3. PCIe Remote Configuration Registers – Remote PCIe configuration registers are accessed by programming the bus number, device number and function number of the remote PCIE device in one of the application register and then accessing the PCI Remote configuration registers as if it were the PCIE Config space of a single PCIE function. The layout of the remote configuration registers varies based upon whether the device is an End Point or a PCIe Switch.
4. PCIe IO Access Window – A 4KB region is dedicated for remote IO accesses when PCIESS is in RC mode. Any access made on this space becomes an IO access. The actual address in the TLP gets its base address from the IO_BASE register value and an offset that is directly derived from the address of the OCP/VBUM access in this 4KB space.

None of the four address ranges described above support burst transactions. So, only single 32-bit transactions should be issued to these addresses. These addresses should also be configured as non-cacheable address space.

17.2.4.1.1 Remote Configuration and I/O Requests (Caution)

Since the remote configuration and IO transaction windows are directly mapped to internal bus space, the software must take caution not to access these spaces when there is no operational PCIe link. No response may be generated for such transactions. It is recommended that checks be built into software to avoid remote accesses in the absence of an operational link.

17.2.4.2 Address Space One

The second address space/region is used for data transfer. The BAR values setup within the PCIe Local Configuration Registers define where within the memory map of the CPU on Root Complex side are the End Points located. All locations other than what is setup in BAR registers of the End Points are on the Root Complex side.

The Address Space One is mapped to multiple devices in RC mode. Each remote device could be allocated a portion of this memory space and any transaction that is targeted to this address space gets converted to a PCIe transaction targeted to the appropriate remote PCIe device.

17.2.4.2.1 Organization of Configuration Registers

As shown in Table 17-3, the registers for various PCIe capabilities are linked to each other via address offsets specified in the registers.

Table 17-3. Register Blocks That Make Up the PCIe Configuration Registers

Offset From Start of Configuration Space	Register Block
00h	PCI-Compatible Header (Type 0/1)
40h	Power Management Capabilities Registers
50h	MSI Capabilities Registers
70h	PCI Express Capabilities Registers
100h	PCI Express Extended Capabilities Registers
700h	Port Logic Registers

17.2.5 Bus Mastering

An End Point that is capable of becoming a bus master and initiate transactions in upstream direction must have its Bus Master Enable bit set in configuration space registers (STATUS_COMMAND[BUS_MASTER]). If transactions are initiated (while assuming the role of a RC or EP) before enabling bus mastering capability, then transactions will not start until the bus master enable is set. There is no timeout or error response generated when transactions does not go out because of bus master bit not being set.

17.2.6 PCIe Loopback

The PCIe Specifications provides loopback support in two ways – through PIPE interface (Link Layer) and second through PHY loopback capability (PHY Layer).

The PIPE Interface Loopback requires for two PCIe Devices/Components to attached to each other in a Loopback Master and a Loopback Slave configuration. A Loopback Master is the component requesting Loopback. A Loopback Slave is the component looping back the data. Note that, regardless with the PCIESS role it assumes (RC or EP), it can assume a role of a Loopback Master or a Loopback Slave.

The Loopback, Loopback at the PHY Level, does not require another component.

17.2.6.1 PIPE Loopback

The procedure depends upon whether the device is operating in RC or EP Mode. In either case, the PCIESS can be loopback master or loopback slave as outlined in PCIe specifications. Note that this Loopback Mode cannot be used for looping back transactions. These modes are to be used with PCIe test equipment only for symbol level loopback.

The Loopback entry procedure when PCIESS is a Loopback Master:

RC Mode

1. Set Loopback Enable in the Port Link Control Register in Port Logic Register space.
2. The link retraining sequence must be initiated by writing to Link Retrain field in the Link Control Register in PCI Express Capabilities structure in the device's PCI Configuration Space.

EP Mode

1. Set Loopback Enable in the Port Link Control Register.
2. Force the LTSSM to be in recovery state via the Port Force Link register of the Port Logic registers.
3. Set Force Link bit to high in Port Force Link register in Port Logic Registers.

Once this is done, devices at the ends of a PCIe link enter PCIe LTSSM Loopback state. The initiator of loopback state is the loopback master and the other device is loopback slave. Note that it is not possible to send TLPs in this mode and return them via the loopback state of the other device.

The Loopback entry procedure when PCIESS is a Loopback Slave:

If the PCIESS is a Loopback slave, then the incoming serial data is routed back to the originating device from the PIPE interface as per PCIe loopback requirements. Typically, PCIe test equipment will be used as Loopback Master and it will transition PCIESS into Loopback Slave state following which the inbound transactions will be Loopback to the test equipment. There is no programming required on PCIESS to enter Loopback in slave mode. PHY support is not required to use this loopback mode.

17.2.6.2 PHY Loopback

The PHY loopback is accomplished by switching the PHY to loopback where the transmitted data is looped back to the receive path at the PHY level. This mode can be used to perform TLP loopback even if there is no link partner. It is, however, only possible to set up when PCIESS is used in RC mode. This limitation is because of the fact that link training cannot occur between two upstream ports; at least one port must be a downstream port.

The procedure is similar to the one for PIPE (Link Layer) interface but the PHY programming is used instead. This mode is entered by configuring the SERDES configuration registers. Both the Transmit and Receive paths must be set in loopback mode to enable PHY loopback mode.

Note that there are several other requirements for this to correctly work:

1. The PHY should be configured to operate in loopback mode before any transaction is sent out. Recommended approach is to set loopback before link training. Otherwise, any transactions that were not looped back will cause sequence numbers to increment on transmitter but not on receiver and all following transactions will be dropped because of sequence number mismatch.
2. The BAR0 and BAR1 values (if programmed) should not match the address for the transaction that is issued on slave interface. Otherwise, it will not get to the master port but to internal registers.
3. The memory base/limit registers should not be configured such that the address of the transaction lies in the range specified by the memory base/limit registers. Otherwise, the transaction will be discarded as a misrouted packet.
4. It is not possible to use configuration type transactions in this mode as RC cannot be a target of configuration transactions. Such transactions will be invalid and loopback will not work.

17.2.7 L3 Memory Map

The system memory mapping of the device is broken into three levels (Level 1, 2, and 3) of granularity for target address spaces allocation for easier address decoding capability. The address spaces required for PCIe usage lies within Level 3 (L3) Memory/Region. See the *Chip Level Resources* chapter or the data manual for more information.

Some processors within the device may re-map these targets to different PCIe addresses through an internal or external MMU. Processors without MMUs and other bus masters will use these OCP/Internal addresses to access L3 regions. Note that not all masters have access to all L3 regions but only those with defined connectivity. See the data manual or the Chip Level chapter for more information.

The OCP/Internal address block accessed by PCIESS is 256MB wide (within L3 region) with starting address of 2000 0000h and ending address of 2FFF FFFFh. This is also referred to Address Space One within PCIe.

Another OCP/Internal address block accessed by PCIESS is a16MB wide block (within L3 region) with starting address of 5100 0000h and ending address of 51FF FFFFh. This address region is referred to Address Space Zero within PCIe.

Two more additional registers within the Control Module (also within L3 region) PCIE_CFG and PCIE_TEST_CTRL are used by the PCIe to configure the PHY PLL, PCIe Mode of operation, Test Mode Patterns, etc.

Four more additional registers within the Power Reset and Control Module (also within L3 region) are used to control the PCIe controller reset status and release states as well as clock enable/disable capabilities. The PM_DEFAULT_PWRSTST and RM_DEFAULT_RSTCTRL registers are used to control reset related tasks while CM_DEFAULT_PCI_CLKSTCTRL and CM_DEFAULT_PCI_CLKCTRL are used to control clocking related functions.

17.2.8 Reset Considerations

The PCIESS supports the Conventional Reset mechanism that is specified within the PCI Express Specification. Both hardware and software reset are supported.

No support for the Functional Level reset is needed since the PCIESS supports a single function.

17.2.8.1 Software Reset Considerations

Software reset is issued via the transmission of TS1 Ordered Sets.

17.2.8.2 Hardware Reset Considerations

Hardware reset is caused while power is being sourced to the device. The SoC level Power On Reset acts as the power on reset for PCIESS.

17.2.8.3 PCIe Boot Capability

The device supports PCIe boot capability as an Endpoint. No support for RC bootmode. See Bootmode configuration details for the applicable settings of the Bootmode pins.

17.2.8.4 PCIe System Module Registers

Two configuration registers PCIE_CFG and PCIE_TEST_CTRL are used for configuring PCIe for operation as well as for Test mode generation. These registers exist within the Control Module space. These registers can only be accessed in Supervisory mode. See the System Module for more information. The base address for the Control Module registers is 4814 0000h.

Four registers within the Power, Reset, and Clock Management (PRCM) Module are used to control the PCIe controller reset status, reset release, and PCIe clock enable/disable capabilities. RM_DEFAULT_RSTST and RM_DEFAULT_RSTCTRL are used to control reset related control capabilities, and CM_DEFAULT_PCI_CLKSTCTRL and CM_DEFAULT_PCI_CLKCTRL are used to control clocking related tasks. The base address for the PRCM registers is 4818 0000h.

17.2.8.4.1 PCIe Configuration Register (PCIE_CFG)

The PCIE_CFG register, in the Control Module, is located at offset 640h and the register format and description are shown in Figure 17-4 and Table 17-4. The PCIE_CFG register is used to configure the PCIe role to be in RC or EP mode. It is also used to configure the PHY PLL multiplier value.

Figure 17-4. PCIe Configuration Register (PCIE_CFG)

31	16
PCIE_CFGPLL	
R/W-1C9h	

15 12	11 8	7 2	1 0
Reserved	PCIE_STSPLL	Reserved	PCIE_DEVTYPE
R-0	R-0	R-0	R/W-0

Table 17-4. PCIe Configuration Register (PCIE_CFG) Field Descriptions

Bit	Field	Description	Reset		Access[1]		
					{Lock Bit}		
					0		1
			Value	Type	Priv	Else	
31-16	PCIE_CFGPLL	PCIe PLL Configuration bits : **Bit [31]** is **Reserved** and must be kept at 0. **Bit [30:29] CLKBYP** Clock bypass. Facilitates bypassing of the PLL with **refclkp / n**. 0x0: The transmitter operates normally from PLL. 0x1, 0x2: Reserved 0x3: The PLL clock is bypassed by **refclkp / n**. **Bit [28:27] LB** Keep these bits at 0x0 (default). **Bit [26] SLEEP_PLL** Puts the PLL into low power sleep state when written to '1'. In this mode, the core of PLL remains locked to refclkp/n, but all clock distribution and associated circuits are powered down. 0x0: Normal mode 0x1: Sleep mode **Bit [25] VRANGE** Write this bit to : 0 : If PLL Output Clock Frequency >= 2.17 GHz. 1 : If PLL Output Clock Frequency < 2.17 GHz. **Bit [24] ENDIVCLK** Enable DIVCLK output : 0: Disables output of a divide-by-5 of PLL clock. 1 : Enables output of a divide-by-5 of PLL clock. **Bit [23:17] MPY** Select PLL Multiply factors between 4 and 25. MPY bits default value **(0b1100100)** can be used to enable a 25× multiplier considering a 100 MHz clock source at PLL input. 0b0010000: multiply by 4 0b0010100: multiply by 5 0b0011000: multiply by 6 0b0100000: multiply by 8 0b0100001: multiply by 8.25 0b0101000: multiply by 10 0b0110000: multiply by 12 0b0110010: multiply by 12.5 0b0111100: multiply by 15 0b1000000: multiply by 16 0b1000010: multiply by 16.5 0b1010000: multiply by 20 0b1011000: multiply by 22 0b1100100: multiply by 25 (default) **Bit [16] ENPLL** Enables the PLL. After setting this bit, it is necessary to allow 350 ns for the regulator to stabilize. Thereafter, the PLL will take no longer than 200 cycles (of **refclkp / n**) to lock to the required frequency, provided refclkp / n are stable. The LOCK bit of PCIE_STSPLL bitfield will be driven high by the digital lock detector. 0: Disable PLL. The PLL is fully powered down. 1: Enable PLL	1C9h	GWR	R/W	R	NA

(1) NA = Not Applicable; R/W = Read/Write; R = Read only

Table 17-4. PCIe Configuration Register (PCIE_CFG) Field Descriptions (continued)

Bit	Field	Description	Reset		Access[1]		
					{Lock Bit}		
					0		1
			Value	Type	Priv	Else	
15-12	Reserved	Reserved. Read returns 0	0	GWR	R	R	NA
11-8	PCIE_STSPLL	PCIe PLL Status bits : **Bits [11:10]** are Reserved and always read as 0. **Bit[9] DIVCLK** : This bit indicates if divided-by-5 PLL clock is running. DIVCLK bit is held low (divided clock is NOT running) unless both the ENPLL and ENDIVCLK bits of the bitfield PCIE_CFGPLL are set high. Note that the "divide-by-5" clock is synchronous to the refclkp / n and NOT to the recovered recive clocks. Furthermore it will be subject to frequency overshoot of < 5% until the PLL has indicated lock in LOCK status bit. 0: DIVCLK is NOT running 1: DIVCLK is running **Bit[8] LOCK** : PLL Lock status bit. Driven high asynchronously between 2048 - 3071 **refclkp / n** cycles after the PLL has locked, which will occur within 200 **refclkp / n** cycles. Whilst LOCK is low, the PLL output frequency may overshoot by no more than <5 %, provided **MPY** is NOT changed to select a lower multiplicaiton factor. The same percentage overshoot will be mirrored by the bus clocks originating from the SerDes. 0: PLL has NOT locked 1 : PLL has locked	0	GWR	R	R0	NA
7-2	Reserved	Reserved. Read returns 0	0	GWR	R	R	NA
1-0	PCIE_DEVTYPE	PCIe Module Device Type 0 Endpoint (EP) operation 1h Legacy Endpoint operation 2h Root Complex (RC) operation 3h Reserved	0	GWR	R/W	R	NA

17.2.8.4.2 PCIe Test Pattern Control Register (PCIE_TEST_CTRL)

The PCIE_TEST_CTRL register, in the Control Module, is located at offset 718h and the register format and description are shown in Figure 17-5 and Table 17-5.

Figure 17-5. PCIe Test Pattern Control Register (PCIE_TEST_CTRL)

31	16
Reserved	
R-0	

15	8	7	6	4	3	2	0
Reserved		TEST_FAIL1	TESTPATT1		TEST_FAIL0	TESTPATT0	
R-0		R-0	R-0		R-0	R/W-0	

Table 17-5. PCIe Test Pattern Control Register (PCIE_TEST_CTRL) Field Descriptions

Bit	Field	Description	Reset		Access[1]		
					{Lock Bit}		
					0		1
			Value	Type	Priv	Else	
31-8	Reserved	Reserved. Read returns 0	0	GWR	R	R	NA
7	TEST_FAIL1	Test Fail Serial to parallel converter mismatch on Ch 1	0	GWR	R	R	NA
6-4	TESTPATT1	PCIe Ch1 Test Pattern Select 000 Test Mode disabled 001 Alternating 0/1 pattern 010 PRBS 7-bit LFSR $x^7 + x^6 + 1$ feedback 011 PRBS 23-bit LFSR $x^{23} + x^{18} + 1$ feedback 100 PRBS 31-bit LFSR $x^{31} + x^{28} + 1$ feedback 101 User Defined pattern (66666h default) 110 Reserved 111 Reserved	0	GWR	R/W	R	NA
3	TEST_FAIL0	Test Fail Serial to parallel converter mismatch on Ch 0	0	GWR	R	R	NA
2-0	TESTPATT0	PCIe Ch0 Test Pattern Select 000 Test Mode disabled 001 Alternating 0/1 pattern 010 PRBS 7-bit LFSR $x^7 + x^6 + 1$ feedback 011 PRBS 23-bit LFSR $x^{23} + x^{18} + 1$ feedback 100 PRBS 31-bit LFSR $x^{31} + x^{28} + 1$ feedback 101 User Defined pattern (66666h default) 110 Reserved 111 Reserved	0	GWR	R/W	R	NA

(1) NA = Not Applicable; R/W = Read/Write; R = Read only

17.2.8.4.3 RM_DEFAULT_RSTCTRL Register

The RM_DEFAULT_RSTCTRL register, in the PRCM module, is located at offset B10h and the register format and description are shown in Figure 17-6 and Table 17-6. The RM_DEFAULT_RSTCTRL register controls the release of the default subsystem resets.

Figure 17-6. RM_DEFAULT_RSTCTRL Register

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
PCI_LRST	Reserved	Reserved	RST3	M3_RST2	M3_RST1	Reserved	Reserved
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-6. RM_DEFAULT_RSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0.
7	PCI_LRST		ACTIVE domain PCI Local reset control
		0	Reset is cleared for PCIe
		1	Reset is asserted for PCIe
6-5	Reserved	3h	Reserved. Always write the default value for future device compatibility.
4	RST3		Logic and MMU reset control
		0	Reset is cleared for the logic and MMU
		1	Reset is asserted for the logic and MMU
3	M3_RST2		Second M3 reset control
		0	Reset is cleared for the ALWON sequencer CPU2
		1	Reset is asserted for the ALWON sequencer CPU2
2	M3_RST1		First M3 reset control
		0	Reset is cleared for the ALWON sequencer CPU1
		1	Reset is cleared for the ALWON sequencer CPU1
1-0	Reserved	3h	Reserved. Always write the default value for future device compatibility.

17.2.8.4.4 RM_DEFAULT_RSTST Register

The RM_DEFAULT_RSTST register, in the PRCM module, is located at offset B14h and the register format and description are shown in Figure 17-7 and Table 17-7. The RM_DEFAULT_RSTST register logs the different reset sources of the DEFAULT domain. Each bit is set upon release of the domain reset signal. Must be cleared by software, using RM_DEFAULT_RSTCTRL register, prior to setting the status bit.

Figure 17-7. RM_DEFAULT_RSTST Register

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
PCI_LRST	Reserved	Reserved	RST3	M3_RST2	M3_RST1	Reserved	
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-7. RM_DEFAULT_RSTST Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	PCI_LRST		PCI local software reset
		0	No software reset occurred
		1	PCI has been reset upon software reset
6-5	Reserved	0	Reserved. Always write the default value for future device compatibility.
4	RST3		Logic and MMU software reset
		0	No software reset occurred
		1	Logic and MMU has been reset upon software reset
3	M3_RST2		Second M3 software reset
		0	No software reset occurred
		1	M3_2 has been reset upon software reset
2	M3_RST1		First M3 software reset
		0	No software reset occurred
		1	M3_1 has been reset upon software reset
1-0	Reserved	0	Reserved. Always write the default value for future device compatibility.

17.2.8.4.5 CM_DEFAULT_PCI_CLKSTCTRL Register

The CM_DEFAULT_PCI_CLKSTCTRL register, in the PRCM module, is located at 4818_0510h and the register format and description is shown below. This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 17-8. CM_DEFAULT_PCI_CLKSTCTRL Register

31	16
Reserved	
R-0	

15	9	8
Reserved		CLKACTIVITY_PCI_GCLK
R-0		R-0

7	2	1	0
Reserved		CLKTRCTRL	
R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-8. CM_DEFAULT_PCI_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Read returns 0
8	CLKACTIVITY_PCI_GCLK		This field indicates the state of the PCI_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Read returns 0
1-0	CLKTRCTRL		Controls the clock state transition of the PCI clock domain in DEFAULT power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wakeup transition on the domain.
		3h	Reserved

17.2.8.4.6 CM_DEFAULT_PCI_CLKCTRL Register

The CM_DEFAULT_PCI_CLKCTRL register, in the PRCM module, is located at 4818_0578h and the register format and description is shown below. This register manages the PCIe clocks.

Figure 17-9. CM_DEFAULT_PCI_CLKCTRL Register

31	19	18	17	16
Reserved		STBYST	IDLEST	
R-0		R-1	R-0	

15	2	1	0
Reserved		MODULEMODE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 17-9. CM_DEFAULT_PCI_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Read returns 0
18	STBYST		Module standby status.
		0	Module is functional (not in standby)
		1	Module is in standby
1-0	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Read returns 0
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wake up).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

17.2.9 Interrupt Support

The PCIESS provides a total of four interrupts to the device interrupt controller, that is, it generates four interrupt events aggregated. In other words, each of these interrupts has more than one underlying event that the interrupt represents. Once all pending interrupts have been cleared, the hardware EOI for the interrupt generic will be generated.

Both Message Signaled Interrupt (MSI) and Legacy Interrupt are handled by the PCIESS. When operating as an EP, the PCIESS is capable of generating MSI or Legacy Interrupt, depending on the role it is assuming. Note that one PCIe component can not generate both type of interrupts. It is either one or the other. The interrupt type an EP generates is configured during configuration time. When operating as RC, the PCIESS is capable of handling both MSI and Legacy Interrupts. This is because when operating as RC it should be able to service both PCIe endpoints as well as Legacy endpoints.

17.2.9.1 Interrupt Events and Requests

A total of four interrupt events are supported by the PCIESS. Some of the events are meaningful based on the role the PCIESS assumes (RC or EP). Table 17-10 captures the Interrupt events supported for both RC and EP modes.

Table 17-10. PCIESS Interrupt Events

Interrupt	Interrupt Description
0	PCIE Express Legacy Interrupt Mode (RC mode only)
	[0] INTA
	[1] INTB
	[2] INTC
	[3] INTD
1	MSI Interrupts 0 through 31 in PCI Express Mode (EP/RC Modes)
	[0] MSI Interrupt 0
	[1..30] MSI Interrupt 1 and so on
	[31] MSI Interrupt 31
2	Error Interrupts (EP, RC)
	[0] System Error (OR of Fatal, Nonfatal, Correctable errors)
	[1] PCIe Fatal Error
	[2] PCIe Non-fatal Error
	[3] PCIe Correctable Error
	[4] AXI Error due to fatal condition in AXI bridge
	[5] PCIe Advanced Error
3	Power Management and Reset Event Interrupts (EP, RC)
	[0] Power Management Turn-off Message Interrupt (EP only)
	[1] Power Management Ack Message Interrupt
	[2] Power Management Event Interrupt
	[3] Link Request Reset Interrupt (Hot reset or Link down)

17.2.9.2 Interrupt Generation

Since the PCIESS is capable of assuming a role of RC or EP, interrupt generation capability is dependant upon the role it is assuming.

17.2.9.2.1 Interrupt Generation in RC Mode

As per PCI Express Base Specifications, Root Complex ports only receive interrupts. There is no mechanism to generate interrupts from RC port to EP mode as per PCIe specification. However, PCIESS does support generation of interrupts from RC to EP. The behavior is similar to generation and reception of MSI interrupts in RC mode except for the fact that this functionality is enabled in EP mode as well.

The RC device can perform a memory write into the MSI IRQ register over the PCIe link to generate one of 32 EP interrupts. Note that the PCIESS will follow PCIe MSI rules and will not necessarily accumulate multiple writes to the same MSI vector. Only one of such writes is guaranteed to be processed and subsequent writes on the same vector arriving before the interrupt status is cleared may be lost.

17.2.9.2.2 Interrupt Generation in EP Mode

When PCIESS is operating as an EP, either the legacy interrupts or the MSI interrupts can be triggered to the upstream ports (eventually leading to an interrupt in RC device). As per PCIe Specifications, each PCIe function may generate only one of the Legacy or MSI interrupt types as decided during configuration period.

Legacy Interrupt Generation in EP Mode

The endpoint can trigger generation of a PCI Legacy Interrupt at the Root Complex via an in-band Assert_INTx / Deassert_INTx PCIe Message. The actual interrupt that is generated on RC port is based on the configuration of the EP that generates the interrupt and it could be one of INTA, INTB, INTC or INTD. See Interrupt related registers in configuration space registers.

To generate an interrupt, following steps are required:

1. Legacy interrupt generation should be enabled via EP_LEGACY_CONFIG register.
2. Write 1h to EP_IRQ_SET register to enable the legacy interrupt.
3. An ASSERT INTA/B/C/D message is automatically sent.
4. Write 1h to EP_IRQ_CLR register to disable the legacy interrupt by sending a DEASSERT INT A/B/C/D message.

Once an assert message has been generated, it cannot be generated again until a deassert message is generated. Thus, only one interrupt can be pending at a time. The pending status can be checked in EP_IRQ_STATUS register.

Note that the interrupt messaging mechanism makes it unfeasible to guarantee a time of delivery of the interrupt unlike in conventional designs where the interrupt line is often electrically connected to the final destination.

There is no hardware input port provided that will allow generation of legacy interrupts on the EP port.

MSI Interrupt Generation in EP Mode

MSI Interrupts are generated by a PCIe Write transaction that performs a 32-bit memory write to a pre-determined address with a pre-determined data. The PCIe system software configures the address and the data that is to be used in the memory write transaction at the time of initialization of the EP device. The MSI scheme supports multiple interrupts and each device can request up to 32 interrupt vectors even though the allotted interrupts may be less than the requested number.

To generate MSI interrupts, the following steps need to be taken:

1. Ensure that the MSI support has been enabled in the device.
2. Read the value of MSI Address Register in the local PCIe configuration space.
3. Read the value of MSI Data Register in the local PCIe configuration space.
4. Determine the number of MSI vectors allocated (and the number requested) to the device.
5. Depending upon the number of MSI interrupts allocated, issue a Memory Write transaction with the address same as MSI Address Register and Data same as MSI Data Register. In the data, the LSBs can be modified to reflect appropriate MSI event that needs to be notified to Root Complex.
6. The Memory Write transaction can also be optionally routed through the outbound address translation interface if the destination PCIe address is not directly accessible.

For more details about how MSI interrupts are expected to behave, refer to PCIe Standard Specifications.

17.2.9.3 Interrupt Reception

Since the PCIESS is capable of assuming a role of RC or EP, interrupt reception capability is dependant upon the role it is assuming.

17.2.9.3.1 Interrupt Reception in EP Mode

The PCIe specification does not have provision for End Points to receive legacy interrupts. As a result, only events other than PCIe related can cause interrupts. The MSI interrupts are not supported on EP devices as per PCIe Specification but PCIESS does support these interrupts. The MSI interrupt is generated as a result of the one of 32 events that are trigger by a write to MSI_IRQ register by the RC.

These interrupts, delivered via register writes over the serial link, could also be coming from another End Point that performs a write to the appropriate interrupt registers in the EP's BAR0 space. It is up to software designers to implement a way to determine the actual source of the interrupt.

17.2.9.3.1.1 Host Reset Request Interrupt Reception in EP Mode

When the link is down, the upstream port may request reset of the End Point. This request is terminated as an interrupt to the End Point host software. The PCIESS automatically disables LTSSM by de-asserting app_ltssm_enable bit in CMD_STATUS register and suspends LTSSM in DETECT QUIET state. All outstanding transactions are errored out on slave port and further transactions are no generated on master port. Once the transactions are completely stopped through OCP disconnect protocol, the software should issue a local reset to PCIESS. The re-initialization process may then be started.

17.2.9.3.2 Interrupt Reception in RC Mode

When PCIESS operates in RC Mode the Endpoints on the PCI fabric can be a Switch, PCIe EP or Legacy EP. For this reason, it should be able to handle both MSI and Legacy Interrupt.

17.2.9.3.2.1 MSI Interrupt Reception in RC Mode

A total of 32 MSI interrupts can be generated from one or more downstream devices. These MSI interrupts represent the MSI interrupts that have been allocated to various downstream devices during PCIe configuration/enumeration procedure. Before the End Point devices can issue MSI interrupts, the MSI address and data registers must be configured. Each End Point can either use MSI interrupts or legacy interrupts and not both at the same time. MSI interrupt have the same race condition hazard as the legacy interrupts. Hence, software drivers should take precaution.

17.2.9.3.2.2 Legacy Interrupt Reception in RC Mode

Any of the four legacy interrupts may be generated by the PCIESS. Each interrupt can originate from multiple End Point devices. The software should service these by probing the interrupt registers in each downstream device's configuration space. When all devices have been serviced, the last device serviced will send a interrupt deassert message which will clear the interrupt.

The interrupt request signal at the PCIESS boundary is a pulse signal that is triggered each time a assert interrupt message is received. The interrupt pending signal is a level signal that is high as long as the interrupt has not be serviced and the interrupt status not cleared through a register write.

Note that the traditional EOI procedure, although implemented, may not operate as expected if the deassert message arrives after the EOI has been issued. This can not be corrected but only worked around by doing a read on the interrupt register downstream before issuing EOI. If the EOI register is written to before the deassert message has reached the interrupt logic, the interrupt pending status will not have cleared and the interrupt will re-trigger.

In addition, the software drivers for downstream devices must ensure that the data transaction that is expected to complete before interrupt is triggered in RC device's CPU has completed. Since PCIe write transactions are posted, it is not necessary that a write from EP to system memory in RC has completed before a write to MSI interrupt generation register has completed. This could create a potential race condition.

It is optional to support Legacy Interrupts in PCI Express (non-legacy) devices.

17.2.10 DMA Support

The PCIESS has no built in DMA and makes use of the EDMA to move data in and out of the PCIESS with out the need of the CPU intervention. EDMA is only used when PCIESS is the initiator of a transaction and accessing other PCIe component. When another PCIe component is accessing a resource within the PCIESS, the PCIe in this case acts as a master and it makes use of the master port to directly access the necessary resource requiring no need for the EDMA usage. For details on EDMA, please consult the EDMA Peripheral Guide.

As an initiator only Cortex-A8 and EDMA can be used to move data in and out of the device resource. As a master (that is, external PCIe component accessing device resource), it is capable of accessing DDR (via DMM), Async EMIF (GPMC), C674x L2 Memory, and On-Chip Memory.

NOTE: Master/Slave reference here does not mean RC/EP mode. As an RC or EP, the PCIESS can act as both Master and Slave. It means as to who is generating the request. If the PCIESS is generating a request, it means that an external PCIe component is the actual entity generating the request and the PCIESS is now regarded as a Master. If the PCIESS is accessing external PCIe component, the PCIESS is regarded as a slave and only Cortex-A8 or EDMA is capable of moving data in and out internal resource.

17.2.10.1 DMA Support in RC Mode

When operating in Root Complex mode, the EDMA controller can perform DMA transfer between device internal resource and any remote device located on the PCI Express fabric. The memory address of such devices is available to the software via the PCI Express bus enumeration procedure. In addition, the PCIe subsystem has a provision to perform memory address translation on outbound requests. Thus, the software is able to map different memory regions in its memory map to correspond to different addresses (and different access types) on the PCI Express side.

There are bandwidth implications of using an external DMA. If the PCIe core has been programmed to establish a link in PCIe 2.5 Gbps rate, then the DMA controller that drives PCIESS Slave Port must be able to write/read data at about 85% of 2 Gbps bandwidth per PCIe link. For a PCIe link speed of 5.0 Gbps, the DMA controller must be able to provide bandwidth of about 85% of 4 Gbps per PCIe link.

In addition, the master port on PCIe port can issue read/write accesses that have been initiated by remote PCI Express device. The interconnect fabric should provide sufficient capacity to serve 85% of 2 Gbps (4 Gbps if operating in Gen2) per PCIe link in each direction.

17.2.10.2 DMA Support in EP Mode

When operating as a PCIe End Point, the device will be located in PCIe memory map at location programmed in the Base Address Registers by the PCIe Root device. In End Point mode, the PCIESS provides address translation functionality. It is possible to map IO, Config and Memory accesses originating on PCI Express side to memory accesses with different address on the OCP side. These address ranges are configurable through application registers.

The approximate data rate for each of these transactions will be about 85% of 2 Gbps (4Gbps in Gen2 mode) in each direction on each PCIe lane.

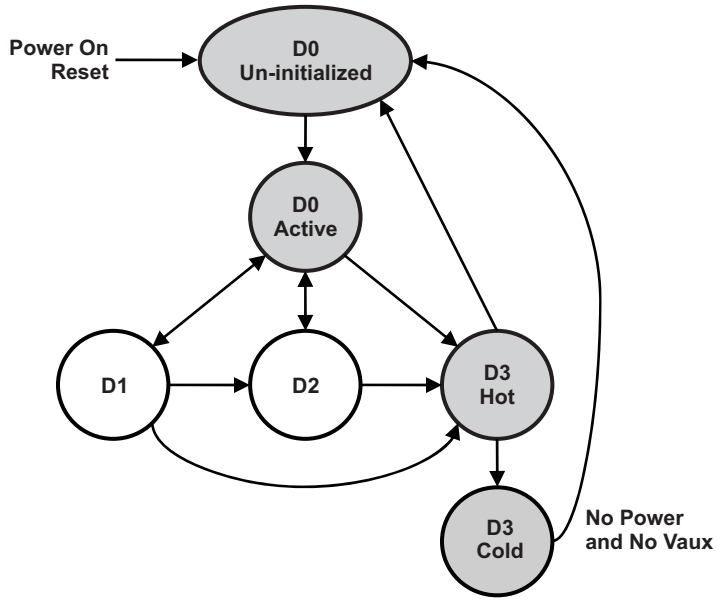
17.2.11 Power Management

PCI Express has multiple power management protocols. Some of these are invoked by the hardware solely, Advanced State Power Management (ASPM) feature, while others are activated at higher levels via software.

17.2.11.1 Device Power Management

The PCI Express protocol is compatible with all PCI power management functionality. The power states specified are D0, D1, D2, D3Hot and D3cold. All functions must support D0 and D3 states.

Figure 17-10. Device Power Management States



17.2.11.1.1 D0 Power State

Upon completion of a reset such a power-on or hot reset, the function is considered to be in D0 uninitialized state. Once the function is enumerated, configured and one or more of the memory space enable, IO space enable or bus master enable bits are set, it is considered to be in D0 active state. The D0 active state is the full-operation state of a PCI Express function.

17.2.11.1.2 D1 Power State

Support for D1 state is optional and is primarily driven by software. It is considered to be a light sleep state that provides some power savings compared to D0 state while still allowing transition to D0 state. While in the D1 state, a Function must not initiate any Request TLPs on the Link except for a PME Message. Configuration and Message Requests are the only TLPs accepted by a Function in the D1state. All other received Requests must be handled as Unsupported Requests, and all received Completions may optionally be handled as Unexpected Completions. A Function's software driver participates in the process of transitioning the Function from D0 to D1. It contributes to the process by saving any functional state (if necessary), and otherwise preparing the Function for the transition to D1. As part of this quiescence process the Function's software driver must ensure that any mid-transaction TLPs (Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to D1.

17.2.11.1.3 D2 Power State

Support for D2 state is optional in PCI Express and is primarily driven by software. It provides significant power savings while still retaining capability to transition back to previous condition. In this state, the PCI function can only initiate Power Management events and it can only respond to configuration accesses.

17.2.11.1.4 D3Hot and D3Cold Power State

All PCI Express functions are required to support D3 state. In D3hot state, power is still present while in D3cold there is no power. Devices in D3Hot state may be transitioned back to D0 state while the devices in D3Hot state need re-initialization to be brought back to D0 state.

Functions in D3hot respond to configuration space accesses as long as power and clock are supplied so that they can be returned to D0 by software. When programmed to D0, the function may return to the D0 Initialized or D0 Un-initialized state without PCIe reset being asserted. There is an option of either performing an internal reset or not performing an internal reset. If not performing an internal reset, upon completion of the D3hot to D0 Initialized state, no additional operating system intervention is required beyond writing the Power State bits. If the internal reset is performed, devices return to D0 Un-initialized and a full re-initialization is performed on the device. The full re-initialization sequence returns the device to D0 Initialized when normal operation may resume.

17.2.11.2 Link State Power Management

There are multiple states for the physical layer – L0, L0s, L1, L2 and L3 states that provide incrementally higher power savings as the states transition from L0 towards L3. PCIESS supports L0, L0s and L1 states. L3 state is power-off state and is supported by default. PCIESS does not support L2 power down states.

Some of the link power states are autonomously managed by hardware. This scheme is referred to as Active State Power Management (ASPM). Figure 17-11 illustrates the transitions between states L0, L0s and L1 that are managed by ASPM. The transitions between gray states are allowed in ASPM. These transitions can be enabled through registers in the PCI Express Configuration Space to allow either just L0s transition or both L0s and L1 transitions.

The L1 power state can also be entered via software control. When the device power state is D1, D2 or D3hot, then the link state must transition to L1 state. Figure 17-12 shows such state transition.

Figure 17-11. Transitions Between Link States

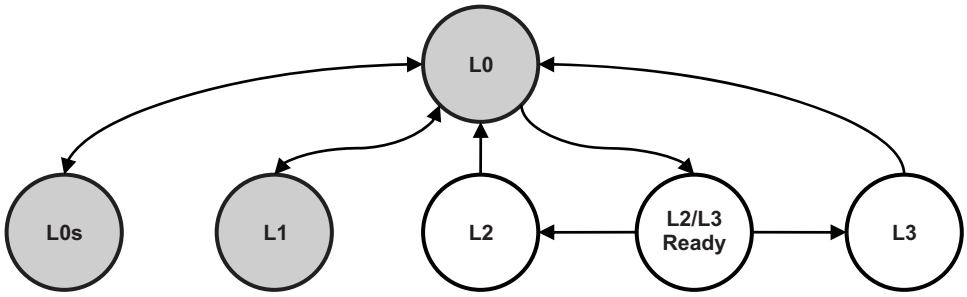
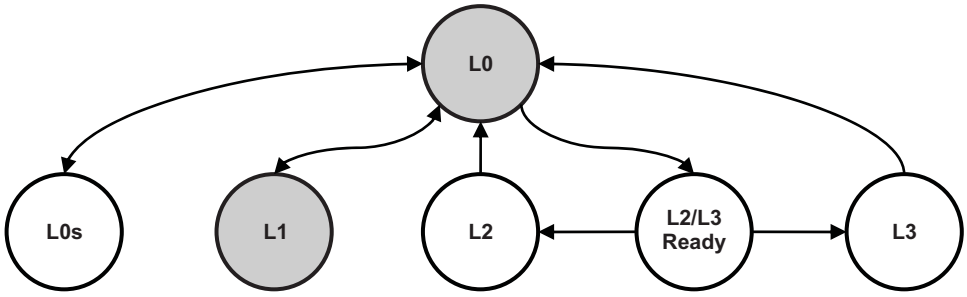


Figure 17-12. Link State Transition to L1 State



17.2.11.2.1 L0s State

L0s is a lower power state enabled by Active State Power Management. Each device can control its L0s transition on its transmitter. Receiver side is controlled by the remote device.

17.2.11.2.2 L1 State

The L1 state is reached either through ASPM timer mechanism or via the software initiated transition to a D state other than the D0 state. In L1 state, the link is in idle state and both receiver and transmitter in devices on both ends of a link are able to conserve energy.

17.2.11.2.3 L2/L3 Ready State

This state is a transitional state reached from L1 where from the link must either transition to L2 or to L3 state. The L3 state is a full power off state. The L2 state is also a power off state except that the WAKE signal can be used by End Point devices to request power and clock from the system.

17.2.11.2.4 L2 State

The L2 state is appropriate when a device needs to monitor an external event while in deep power down mode. In L2 state, auxiliary power is supplied and a minimal amount of current is drawn from the power source. Almost all of the logic is devoid of power. To recover, the wake signal is used.

17.2.11.2.5 L3 State

In this state, device is supplied no power from the PCIe fabric. There is no mechanism to communicate in L3 state. To recover, the system must re-establish power and reference clock followed by a fundamental reset.

17.2.12 Relationship Between Device and Link Power States

The device D-states are correlated to the link power states. For each D state, there are specific states that the PCIe link or interconnect can transition to. Table 17-11 shows the permissible state combinations.

Table 17-11. Device Power States and Link Power States Relations

Downstream Device State	Permissible Upstream Device State	Permissible Link State
D0	D0	L0 (required), L0s (required), L1 ASPM (optional)
D1	D0 – D1	L1
D2	D0 – D2	L1
D3hot	D0 – D3hot	L1, L2/L3 Ready
D3cold	D0 – D3cold	L2aux, L3

17.2.12.1 OCP Power Management

The OCP bus power management is accomplished by use of Idle and Standby Mode. The list of Idle/Standby states support by PCIESS are in Table 17-12

Table 17-12. Idle/Stand-by State Supported

IDLE/STANDBY Mode	Remarks
NO IDLE	Supported.
SMART IDLE	Default State
SMART IDLE w/ WAKE-UP	Supported
FORCE IDLE	Supported
NO STANDBY	Supported
SMART STANDBY	Default State
SMART STANDBY w/ WAKE	Not supported
FORCE STANDBY	Supported

Note that if Idle mode is reached in default state, the hardware will not request wakeup once in Idle mode. Software must ensure that it wakes up the hardware by exiting IDLE state in such situations. The compatibilities of various states are tabulated in Table 17-13.

Table 17-13. Relationships Between Power States and Link States

STANDBY Mode	Tested IDLE Mode(s)	Link States and transitions
NO STANDBY	NO IDLE	States: L0 and L1 Transitions: L0àL1, L1àL0
SMART STANDBY	SMART IDLE WITH WAKE	States: L0, L1 The valid transitions among these states are tested Transitions: L0àL1, L1àL0, L0àL2/L3 Ready The states L2 and L3 need fundamental reset to recover from.
SMART STANDBY	SMART IDLE	States: L0, L1, L2/L3 Ready Transitions: L0àL1, L0àL2/L3 Ready Valid transitions among these states are tested. Note that the Remote partner cannot wake up PCIESS if OCP wakeup is not enabled. A fundamental reset is always required to wake from L2 or L3 states.
SMART STANDBY	NO IDLE	States: L0, L1 Transitions: L0àL1 Only master port will enter standby. No clock gating is achievable in this configuration.
SMART STANDBY w/ WAKE	NA	Not supported
FORCE STANDBY	FORCE IDLE	States: Uninitialized Link, L0 Transitions: None It is not recommended to keep OCP in standby/idle and link in initialized state. This should be done only when the remote side is aware that the device interconnect in not operational and therefore, no transactions should be initiated.

17.3 Use Case

17.3.1 PCIe Root Complex

When the PCIESS is desired to operate as a Root Complex (RC), the following initialization sequence is recommended.

17.3.1.1 PCIe Root Complex Initialization Sequence

The initialization sequence is:

1. Configure PCIe Mode of operation to RC Mode by programming PCIE_CFG.PCIE_DEVTYPE with a value of 2h.
2. Bring PCIESS out of reset through the device level reset controller.
3. Enable and configure PCIe Clock (See PCIE_CFG Register Description). Note: The Default PLL Multiplier configuration value (PCIE_CFG.PCIE_CFGPLL = 1C9h) is most likely to be used if using a 100 MHz input clock source.
4. Wait for the PCIe PHY PLL to lock; wait for PCIe PLL Status (PCIE_CFG.PCIE_STSPLL) to change to 1h.
5. Insure that the link is idle. Disable link training by de-asserting the LTSSM Enable bit in PCIESS Control Register. Upon reset, the LTSSM Enable is de-asserted automatically by hardware, CMD_STATUS.LTSSM_EN is zero.
6. Configure core registers through OCP Slave interface address space 0.
7. Initiate link training can be initiated by asserting LTSSM Enable bit in PCIESS Control Register; programming CMD_STATUS.LTSSM_EN with a 1.
8. Insure link training completion and success by observing DEBUG0.LTSSM_STATE field change to 11h.
9. In conjunction with the system software, start bus enumeration and setup configuration space on downstream ports.
10. Continue software handshake and initialization on the remote devices. This includes setting up DMA protocols, interrupt procedures, etc.
11. With the completion of software initialization, DMA accesses can be started on various end points.

17.3.1.2 PCIe Root Complex Configuration Accesses

Configuration accesses are made by RC port to individual function in each downstream EP device to program the PCIe specific operating parameters. In particular, the configuration accesses are used to allocate memory ranges for each downstream device, configure those memory ranges as IO or Memory type, enable bus master capability on the device if necessary and also build a software database of PCIE attributes and capabilities of each downstream device. Each downstream device can be configured through the configuration access region of PCIESS. The PCIESS convert memory reads and writes on the configuration region of OCP interface into configuration access on the serial link.

17.3.1.3 PCIe Root Complex Memory Accesses

There are two types of memory accesses – the outbound memory accesses that are initiated by the DMA on the PCIESS Slave port and the inbound memory accesses that are initiated by the PCIESS Master port targeted to internal memory regions within the OCP Interconnect.

The outbound PCIe memory read and write accesses are made through the PCIESS slave port through the device memory range that is dedicated to data transfers. The read and write transactions on this region are directly mapped to PCI Express space by the PCIESS in conjunction with the outbound address translation mechanism. The completions to the bus transactions are generated by PCIESS when it receives completions from remote devices. In case of errors or timeouts, an error response is provided. For reads, the error responses span as many phases as there would be data phases if the error had not occurred.

The inbound memory accesses received by PCIESS on the serial link are initiated by remote PCIe End Points that are capable of bus mastership. Such accesses are converted to bus transactions on PCIESS master port and once a Response/Completion is received, the PCIESS sends data/response back to the PCIe bus master over the serial links. Typically, the inbound accesses will be targeted to memory space resident on the bus side of the RC port. The locations to which inbound memory accesses map are determined by software. The software must inform the remote devices about what protocol is to be followed so that the remote device will access the relevant memory regions to read or write data/control information. Not all End Points have the capability to initiate inbound accesses.

An inbound access cannot cross a 4 KB boundary as per PCIe specifications. In addition, any access that spans a 128 byte boundary in the device memory map can get split into two transactions at the 128 byte boundary.

17.3.1.4 PCIe Root Complex I/O Accesses

I/O accesses are optional in PCI Express. In PCIESS, these accesses can be made through a 4KB memory space and a programmable register. All accesses made to the 4KB space become IO accesses and the IO Base register determines the target address for such accesses. The IO accesses cannot be for more than 32-bits of data aligned at 4 byte boundary.

17.3.2 PCIe End Point

When the PCIESS is desired to operate as an Endpoint (EP), the following initialization sequence is recommended.

17.3.2.1 PCIe End Point Initialization Sequence

Upon de-assertion of reset, the PCIESS is configured as End Point by chip level setting of PCIESS inputs. Before a Root Complex is allowed to access the configuration space of the end point, the following initialization sequence should be followed:

1. Configure PCIe Mode of operation for EP Mode by programming PCIE_CFG.PCIE_DEVTYPE with a value of 0.
2. Bring PCIESS out of reset through the device level reset controller.
3. Enable and configure PCIe Clock (See PCIE_CFG Register Description). Note: The Default PLL Multiplier configuration value (PCIE_CFG.PCIE_CFGPLL= 1C9h) is most likely to be used if using a 100 MHz input clock source.
4. Wait for the PCIe PHY PLL to lock; wait for PCIe PLL Status (PCIE_CFG.PCIE_STSPLL) to change to 1h.
5. Insure that the link is idle. Disable link training by de-asserting the LTSSM Enable bit in PCIESS Control Register. Upon reset, the LTSSM Enable is de-asserted automatically by hardware, CMD_STATUS.LTSSM_EN is 0.
6. Program the configuration registers in the PCIESS to desired values.
7. Initiate link training can be initiated by asserting LTSSM Enable bit in PCIESS Control Register; programming CMD_STATUS.LTSSM_EN with a 1.
8. Insure link training completion and success by observing DEBUG0.LTSSM_STATE field change to 11h.
9. If further configuration register initialization is required, the Application Request Retry bit should be set. This will lead to incoming accesses to be responded with the retry response. This feature allows slow devices extra time before the Root Port assumes the devices to be inactive. Once programming is complete, de-assert Application Request Retry field to allow transactions from the Root Complex.
10. Once configuration setup is complete, DMA transactions can begin.
11. Inbound PCIe transactions will arrive at the master port of the PCIESS End Point. These will be responded to by the target slave devices and the PCIESS will relay the response back to the PCIE device that initiated the transaction.
12. Outbound PCIe transactions will be targeted to the slave port of the PCIESS End Point. These transactions will be serviced only if the PCIESS End Point has been given Bus Master Capability by the Root Complex.

17.3.2.2 PCIe End Point Configuration Accesses

As an endpoint, the PCIESS can only be a target of configuration accesses from upstream. Until the link is established and APP_RETRY_EN is disabled, the PCIESS will not respond to configuration accesses. When enabled, the PCIESS will respond to configuration accesses automatically and these accesses do not get relayed to the master interface on the device interconnect side.

End point is not capable to accessing configuration space of devices other than its own. The system software can initialize the read only fields in PCIESS Configuration Space via the slave port before the link training has been initiated. Once the configuration is complete by the PCI Express Root Complex, the system software should not modify the PCIESS configuration parameters. There is no explicit prevention mechanism in hardware to disallow such accesses though.

17.3.2.3 PCIe End Point Memory Accesses

There are two types of Memory Accesses – inbound and outbound memory accesses.

An inbound access is typically initiated by a Root Complex port or by another End Point that is reaching the PCIESS via a PCI Express switch that supports peer-to-peer access. In either case, the incoming PCIe transaction results in an access on the PCIESS master port. The response to the such accesses is relayed back to the originating PCIe device.

In an outbound access, a DMA module or a host CPU initiates a read/write access on the PCIESS Slave port. This request is converted into a PCIe memory read/write transaction over the PCIE link. Once the PCIESS receives completion from the remote device, it generates a completion on the OCP slave port. The software/hardware that initiates the request on the slave port must perform it in a memory region that has been determined previously through software protocols. For example, the software may get information about applicable memory regions from the software that is running on the Root Complex device.

17.3.2.4 PCIe End Point I/O Accesses

IO Accesses are not supported in PCIESS operating as a PCIe End Point.

17.4 PCIe Registers

17.4.1 Accessing Read-only Registers in Configuration Space

Some of the register fields provided in the configuration space can be written to prior to bus enumeration via the slave interface of PCIESS. Note that the hardware does not prevent modification of read only field after bus enumeration but it is strongly advised that read only fields not be written once the bus enumeration is complete.

In addition, the BAR Mask registers can also be programmed by first enabling the DBI_CS2 bit and then performing writes on the BAR registers. The BAR mask registers are overlaid on BAR registers. For the BAR mask registers to be writable, the respective BAR must first be enabled.

17.4.2 Accessing EP Application Registers from PCIe RC

The application registers are also mapped at 2K and above address, in the configuration space. The RC software can access these registers over PCIe link provided the registers are programmed to some default values by the BOOT code in the PCIESS host device that enables link training and TLP exchange.

17.4.3 Encoding of LTSSM State in DEBUG Registers

The LTSSM state value that is read from DEBUG registers is an encoded value. The literal names of the LTSSM states corresponding to the encoded values are tabulated in Table 17-14.

Table 17-14. Encoded Debug Registers Contents and Their Relation to LTSSM States

Code	LTSSM State
0h	DETECTQUIET
1h	DETECTACT
2h	POLLACTIVE
3h	POLLCOMPLIANCE
4h	POLLCONFIG
5h	PREDETECTQUIET
6h	DETECTWAIT
7h	CFGLINKWDSTART
8h	CFGLINKWDACEPT
9h	CFGLANENUMWAIT
Ah	CFGLANENUMACEPT
Bh	CFGCOMPLETE
Ch	CFGIDLE
Dh	RCVRYLOCK
Eh	RCVRYSPEED
Fh	RCVRYRCVRCFG
10h	RCVRYIDLE
11h	L0
12h	L0S
13h	L123SENDEIDLE
14h	L1IDLE
15h	L2IDLE
16h	L2WAKE
17h	DISABLEDENTRY
18h	DISABLEDIDLE
19h	DISABLED
1Ah	LPBKENTRY
1Bh	LPBKACTIVE
1Ch	LPBKEXIT
1Dh	LPBKEXITTIMEOUT
1Eh	HOTRESETENTRY
1Fh	HOTRESET

17.4.4 PCIe Application Registers

The Application registers are accessible in multiple ways depending on the point of origin of such accesses. When accessed from the OCP side (on board CPU or EDMA performing the access), these registers are accessed via the 4KB space in Address Space or Address Region Zero. When accessed from the PCIe serial link side, the application registers are mapped to BAR0 of an EP as well as RC. In other words, if a PCIe component desires access to the application registers, based on the address type used (memory or configuration) the right block of memory is accessed automatically. In addition, an RC can access these registers in a PCIESS EP via the upper 1 KB space of the PCIe configuration space. The offsets of the registers remain the same regardless with the access method used.

Table 17-15 lists the PCIe application registers. For the base address of these registers, see Table 1-11.

Table 17-15. PCIe Application Registers

Offset	Acronym	Register Name	Section
0h	PID	Peripheral Version and ID Register	Section 17.4.4.1
4h	CMD_STATUS	Command Status Register	Section 17.4.4.2
8h	CFG_SETUP	Configuration Transaction Setup Register	Section 17.4.4.3
Ch	IOBASE	IO TLP Base Register	Section 17.4.4.4
10h	TLPCFG	TLP Attribute Configuration	Section 17.4.4.5
14h	RSTCMD	Reset Command and Status Register	Section 17.4.4.6
20h	PMCMD	Power Management Command Register	Section 17.4.4.7
24h	PMCFG	Power Management Configuration Register	Section 17.4.4.8
28h	ACT_STATUS	Activity Status Register	Section 17.4.4.9
30h	OB_SIZE	Outbound Size Register	Section 17.4.4.10
34h	DIAG_CTRL	Diagnostic Control Register	Section 17.4.4.11
38h	RESERVED	RESERVED	-
3Ch	PRIORITY	CBA Transaction Priority Register	Section 17.4.4.12
50h	IRQ_EOI	End of Interrupt Register	Section 17.4.4.13
54h	MSI_IRQ	MSI Interrupt Register	Section 17.4.4.14
64h	EP_IRQ_SET	Endpoint Interrupt Request Set Register	Section 17.4.4.15
68h	EP_IRQ_CLR	Endpoint Interrupt Request Clear Register	Section 17.4.4.16
6Ch	EP_IRQ_STATUS	Endpoint Interrupt Status Register	Section 17.4.4.17
70h	GPR0	General Purpose 0 Register	Section 17.4.4.18
74h	GPR1	General Purpose 1 Register	Section 17.4.4.19
78h	GPR2	General Purpose 2 Register	Section 17.4.4.20
7Ch	GPR3	General Purpose 3 Register	Section 17.4.4.21
100h	MSI0_IRQ_STATUS_RAW	MSI 0 Interrupt Raw Status Register	Section 17.4.4.22
104h	MSI0_IRQ_STATUS	MSI 0 Interrupt Enabled Status Register	Section 17.4.4.23
108h	MSI0_IRQ_ENABLE_SET	MSI 0 Interrupt Enable Set Register	Section 17.4.4.24
10Ch	MSI0_IRQ_ENABLE_CLR	MSI 0 Interrupt Enable Clear Register	Section 17.4.4.25
180h	IRQ_STATUS_RAW	Interrupt Raw Status Register	Section 17.4.4.26
184h	IRQ_STATUS	Interrupt Enabled Status Register	Section 17.4.4.27
188h	IRQ_ENABLE_SET	Interrupt Enable Set Register	Section 17.4.4.28
18Ch	IRQ_ENABLE_CLR	Interrupt Enable Clear Register	Section 17.4.4.29
1C0h	ERR_IRQ_STATUS_RAW	ERR Interrupt Raw Status Register	Section 17.4.4.30
1C4h	ERR_IRQ_STATUS	ERR Interrupt Enabled Status Register	Section 17.4.4.31
1C8h	ERR_IRQ_ENABLE_SET	ERR Interrupt Enable Set Register	Section 17.4.4.32
1CCh	ERR_IRQ_ENABLE_CLR	ERR Interrupt Enable Clear Register	Section 17.4.4.33
1D0h	PMRST_IRQ_STATUS_RAW	Power Management and Reset Interrupt Raw Status Register	Section 17.4.4.34
1D4h	PMRST_IRQ_STATUS	Power Management and Reset Interrupt Enabled Status Register	Section 17.4.4.35

Table 17-15. PCIe Application Registers (continued)

Offset	Acronym	Register Name	Section
1D8h	PMRST_ENABLE_SET	Power Management and Reset Interrupt Enable Set Register	Section 17.4.4.36
1DCh	PMRST_ENABLE_CLR	Power Management and Reset Interrupt Enable Clear Register	Section 17.4.4.37
200h	OB_OFFSET_INDEXn	Outbound Translation Region n Offset Low and Index Register	Section 17.4.4.38
204h	OB_OFFSETn_HI	Outbound Translation Region n Offset High Register	Section 17.4.4.39
300h	IB_BAR0	Inbound Translation Bar Match 0 Register	Section 17.4.4.40
304h	IB_START0_LO	Inbound Translation 0 Start Address Low Register	Section 17.4.4.41
308h	IB_START0_HI	Inbound Translation 0 Start Address High Register	Section 17.4.4.42
30Ch	IB_OFFSET0	Inbound Translation 0 Address Offset Register	Section 17.4.4.43
310h	IB_BAR1	Inbound Translation Bar Match 1 Register	Section 17.4.4.44
314h	IB_START1_LO	Inbound Translation 1 Start Address Low Register	Section 17.4.4.45
318h	IB_START1_HI	Inbound Translation 1 Start Address High Register	Section 17.4.4.46
31Ch	IB_OFFSET1	Inbound Translation 1 Address Offset Register	Section 17.4.4.47
320h	IB_BAR2	Inbound Translation Bar Match 2 Register	Section 17.4.4.48
324h	IB_START2_LO	Inbound Translation 2 Start Address Low Register	Section 17.4.4.49
328h	IB_START2_HI	Inbound Translation 2 Start Address High Register	Section 17.4.4.50
32Ch	IB_OFFSET2	Inbound Translation 2 Address Offset Register	Section 17.4.4.51
330h	IB_BAR3	Inbound Translation Bar Match 3 Register	Section 17.4.4.52
334h	IB_START3_LO	Inbound Translation 3 Start Address Low Register	Section 17.4.4.53
338h	IB_START3_HI	Inbound Translation 3 Start Address High Register	Section 17.4.4.54
33Ch	IB_OFFSET3	Inbound Translation 3 Address Offset Register	Section 17.4.4.55
380h	PCS_CFG0	PCS Configuration 0 Register	Section 17.4.4.56
384h	PCS_CFG1	PCS Configuration 1 Register	Section 17.4.4.57
388h	PCS_STATUS	PCS Status Register	Section 17.4.4.58
390h	SERDES_CFG0	SerDes Configuration for Lane 0 Register	Section 17.4.4.59
394h	SERDES_CFG1	SerDes Configuration for Lane 1 Register	Section 17.4.4.60

17.4.4.1 PID Register

The peripheral version and ID register (PID) is described in the figure and table below.

Figure 17-13. PID Register

31	30	29	28	27	16
SCHEME		Reserved		FUNC	
R-1		R-0		R-E31h	

15	11	10	8	7	6	5	0
RTL		MAJOR		CUSTOM		MINOR	
R-1		R-1		R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-16. PID Register Field Descriptions

Bit	Field	Value	Description
31-30	SCHEME	0-3h	PID Register Format Scheme
29-28	Reserved	0	Reserved
27-16	FUNC	0-FFFh	Function code of the peripheral
15-11	RTL	0-1Fh	RTL version number (R)
10-8	MAJOR	0-7h	Major revision code (X)
7-6	CUSTOM	0-3h	Custom code
5-0	MINOR	0-3Fh	Minor revision code

17.4.4.2 CMD_STATUS Register

The command status register (CMD_STATUS) is described in the figure and table below.

Figure 17-14. CMD_STATUS Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved				OCP_STANDBY		OCP_IDLE	
R-0				R/W-2h		R/W-2h	

7	6	5	4	3	2	1	0
Reserved		DBI_CS2	APP_RETRY_EN	POSTED_WR_EN	IB_XLT_EN	OB_XLT_EN	LTSSM_EN
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-17. CMD_STATUS Register Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11-10	OCP_STANDBY	0-3h	The **OCP standby mode bitfield** defines the local clock-management behavior of the PCIe module based on Standby (master) protocol with the device PRCM. **For more information on Standby states relations to the PCIe protocol link power states, refer to** Section 17.2.12.1, *OCP Power Management* .
			0h : Force standby / The module unconditionally asserts the standby request to the PRCM module, regardless of its internal operations. The PRCM module may gate the functional and interface clocks to the module. This mode must be used carefully **(should NOT be used in normal usage but for debug purposes only)** because it does not prevent loss of data at the time the clocks are gated. /
			1h : No standby / The module never asserts the standby request to the PRCM module. This mode is safe from a module point of view because it ensures that the clocks remain active; however, it is not efficient from a power-saving perspective because it never allows the PRCM module output clocks to be gated. /
			2h : Smart standby / The module asserts the standby request based on its internal activity status. The standby signal is asserted only when all ongoing transactions are complete and the module is idled. The PRCM module can then gate the clocks to the module. Note that, this mode is NOT wake-up capable, i.e. the module can NOT generate (IRQ- or DMA request-related) wake-up events when in IDLE state. /
			3h : Reserved
9-8	OCP_IDLE	0-3h	The **OCP idle mode bitfield** defines the local clock-management behavior of the PCIe module based on Idle (Slave) protocol with the device PRCM. **For more information on Idle states relations to the PCIe protocol link power states, refer to** Section 17.2.12.1, *OCP Power Management*.
			0h : Force Idle / The module unconditionally acknowledges an IDLE request from the PRCM module, regardless of its internal operations. This mode must be used carefully **(should NOT be used in normal usage but for debug purposes only)**, because it does not prevent loss of data at the time the clock is switched off. /
			1h : No Idle / The module never acknowledges any IDLE request from the PRCM module. This mode is safe from a module point of view because it ensures the clocks remain active; however, it is not efficient from a power-saving perspective because it does not allow the PRCM module output clock to be shut off, and thus the power domain to be set to a lower power state. /
			2h : Smart Idle / The module acknowledges an IDLE request, basing its decision on its internal activity. Namely, the acknowledge signal is asserted only when all pending transactions, interrupts, or DMA requests are processed. This is the best approach to efficient system power management. Note that, this mode is NOT wake-up capable, i.e. the module can NOT generate (IRQ- or DMA request- related) wake-up events when in IDLE state. /
			3h : Smart Idle (wake-up capable) / The module acknowledges the IDLE request, basing its decision on its internal activity. Namely, the acknowledge signal is asserted only when all pending transactions, interrupts, or DMA requests are processed. This is the best approach to efficient system power management. The module may generate (IRQ- or DMA request- related) wake-up events when in IDLE state. /

Table 17-17. CMD_STATUS Register Field Descriptions (continued)

Bit	Field	Value	Description
7-6	Reserved	0	Reserved
5	DBI_CS2	0	Set to enable writing to BAR mask registers that are overlaid on BAR registers.
4	APP_RETRY_EN	0	Application Request Retry Enable—Setting this bit will enable all incoming PCIe transactions to be returned with a retry response. This feature can be used if initialization can take longer than PCIe stipulated time frame.
3	POSTED_WR_EN	0	Posted Write Enable—Setting this bit will cause the OCP master to use posted write commands. Default is zero with all OCP Master writes defaulting to non-posted.
2	IB_XLT_EN	0	Inbound Address Translation Enable—Setting this bit will enable translation of inbound memory/io read/write requests into memory read/write requests.
1	OB_XLT_EN	0	Outbound Address Translation Enable—Setting this bit will enable translation of outbound memory read/write requests into memory/io/cfg read/write requests.
0	LTSSM_EN	0	Link Transitioning Enable—Setting this bit will enable LTSSM in PCI Express Core and link negotiation with link partner will begin.

17.4.4.3 CFG_SETUP Register

The config transaction setup register (CFG_SETUP) is described in the figure and table below.

Figure 17-15. CFG_SETUP Register

31	25	24	23	16
Reserved		CFG_TYPE	CFG_BUS	
R-0		R/W-0	R/W-0	

15	13	12	8	7	3	2	0
Reserved		CFG_DEVICE		Reserved		CFG_FUNC	
R-0		R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-18. CFG_SETUP Register Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24	CFG_TYPE	0	Configuration Type for outbound configuration accesses. Set for Type 1 access and clear for Type 0 access.
23-16	CFG_BUS	0-FFh	PCIe Bus number for outbound configuration accesses
15-13	Reserved	0	Reserved
12-8	CFG_DEVICE	0-1Fh	PCIe Device number for outbound configuration accesses
7-3	Reserved	0	Reserved
2-0	CFG_FUNC	0-7h	PCIe Function number for outbound configuration accesses

17.4.4.4 IOBASE Register

The IO TLP base register (IOBASE) is described in the figure and table below.

Figure 17-16. IOBASE Register

31	16
IOBASE	
R/W-0	

15	12	11	0
IOBASE		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-19. IOBASE Register Field Descriptions

Bit	Field	Value	Description
31-12	IOBASE	0-F FFFFh	Bits 31-12 of outgoing IO TLP. RC mode only.
11-0	Reserved	0	Reserved

17.4.4.5 TLPCFG Register

The TLP attribute configuration register (TLPCFG) is described in the figure and table below.

Figure 17-17. TLPCFG Register

31	16
Reserved	
R-0	

15	2	1	0
Reserved		RELAXED	NO_SNOOP
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-20. TLPCFG Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	RELAXED	0	Enable Relaxed Ordering for all outgoing TLPs.
0	NO_SNOOP	0	Enable No Snoop attribute on all outgoing TLPs.

17.4.4.6 RSTCMD Register

The reset command and status register (RSTCMD) is described in the table and figure below.

Figure 17-18. RSTCMD Register

31	17	16
Reserved		FLUSH_N
R-0		R-1

15	1	0
Reserved		INIT_RST
R-0		W1S-0

LEGEND: R = Read only; W1S = Write 1 to set; -*n* = value after reset

Table 17-21. RSTCMD Register Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	FLUSH_N	1	Bridge Flush Status—Reads a zero when no transaction is pending. Used to ensure no pending transactions before issuing warm reset. Only applicable in PCIESS versions that use Designware core version 3.57 and newer.
15-1	Reserved	0	Reserved
0	INIT_RST	0	Write 1 to initiate a downstream hot reset sequence on downstream.

17.4.4.7 PMCMD Register

The power management command register (PMCD) is described in the figure and table below.

Figure 17-19. PMCMD Register

31	16
Reserved	
R-0	

15	2	1	0
Reserved		PM_XMT_TURNOFF	PM_XMT_PME
R-0		R/W1S -0	R/W1S -0

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set; -*n* = value after reset

Table 17-22. PMCMD Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	PM_XMT_TURNOFF	0	Write 1 to transmit a PM_TURNOFF message. Reads zero. Applicable in RC mode only.
0	PM_XMT_PME	0	Write 1 to transmit a PM_PME message. Reads zero. Applicable to EP mode only.

17.4.4.8 PMCFG Register

The power management configuration register (PMCFG) is described in the figure and table below.

Figure 17-20. PMCFG Register

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		RDY_ENTR_L23	Reserved	ENTR_L23
R-0		R-0	R-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

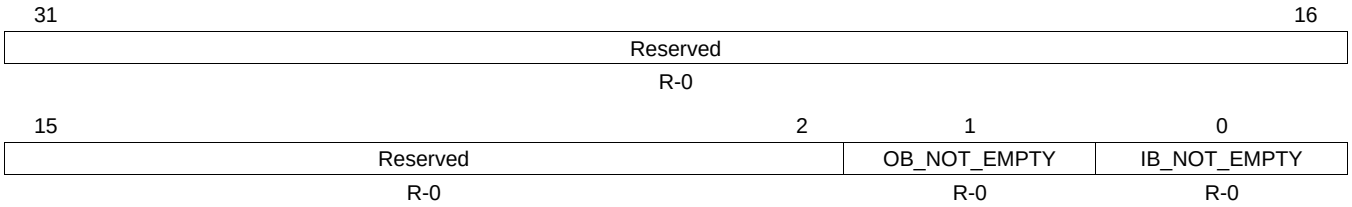
Table 17-23. PMCFG Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	RDY_ENTR_L23	0	Read L2/L3 entry readiness. Applicable to RC and EP.
1	Reserved	0	Reserved
0	ENTR_L23	0	Write one to enable entry to L2/L3 ready state. Applicable to RC and EP.

17.4.4.9 ACT_STATUS Register

The activity status register (ACT_STATUS) is described in the figure and table below.

Figure 17-21. ACT_STATUS Register



LEGEND: R = Read only; -*n* = value after reset

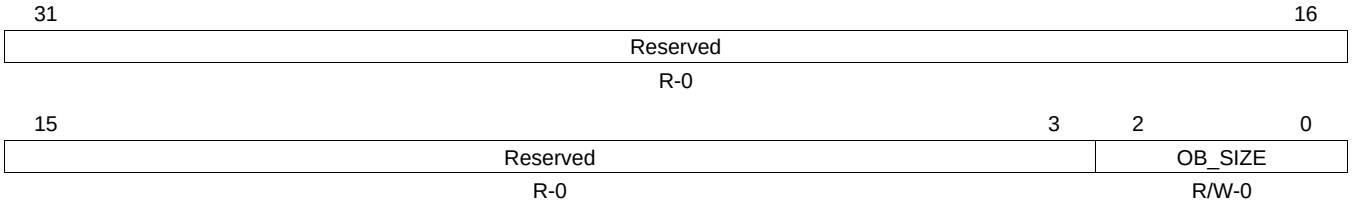
Table 17-24. ACT_STATUS Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	OB_NOT_EMPTY	0	Outbound buffers are empty if this bit is read as 0.
0	IB_NOT_EMPTY	0	Inbound buffers are empty if this bit is read as 1.

17.4.4.10 OB_SIZE Register

The outbound size register (OB_SIZE) is described in the figure and table below.

Figure 17-22. OB_SIZE Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

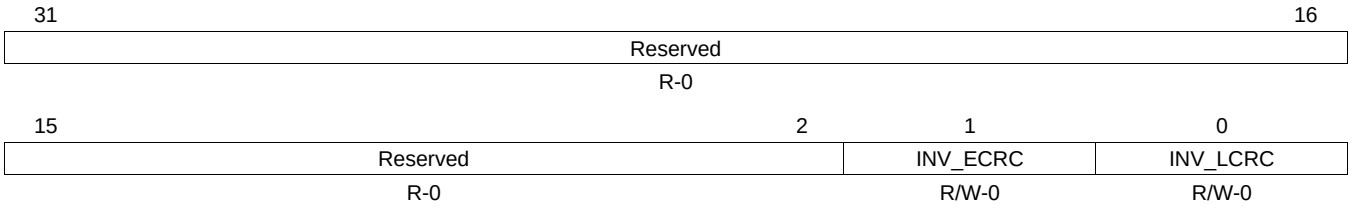
Table 17-25. OB_SIZE Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	OB_SIZE	0-7h	Write 0, 1, 2 and so on to set each outbound translation window size to 1, 2, 4 MB and so on. Applicable to RC and EP.

17.4.4.11 DIAG_CTRL Register

The diagnostic control register (DIAG_CTRL) is described in the figure and table below.

Figure 17-23. DIAG_CTRL Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

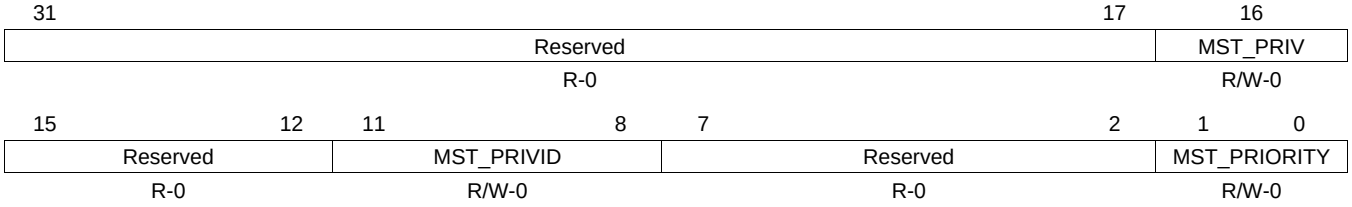
Table 17-26. DIAG_CTRL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	INV_ECRC	0	Write 1 to force inversion of LSB of ECRC for next one packet. It is self-cleared when the ECRC error has been injected on one TLP.
0	INV_LCRC	0	Write 1 to force inversion of LSB of LCRC for next one packet. It is self-cleared when the ECRC error has been injected on one TLP.

17.4.4.12 PRIORITY Register

The CBA transaction priority register (PRIORITY) is described in the figure and table below.

Figure 17-24. PRIORITY Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

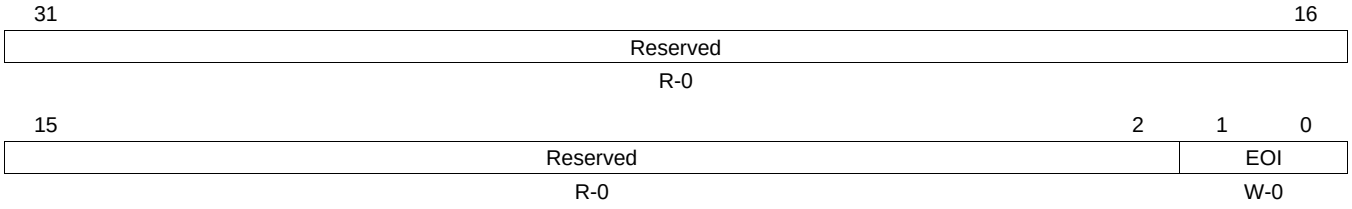
Table 17-27. PRIORITY Register Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	MST_PRIV	0	Value on master transactions
15-12	Reserved	0	Reserved
11-8	MST_PRIVID	0-Fh	Value on master transactions
7-2	Reserved	0	Reserved
1-0	MST_PRIORITY	0-3h	Priority level for each inbound transaction on the CBA master port. This field is not used for OCP interface.

17.4.4.13 IRQ_EOI Register

The end of interrupt register (IRQ_EOI) is described in the figure and table below.

Figure 17-25. IRQ_EOI Register



LEGEND: R = Read only; W = Write only; -*n* = value after reset

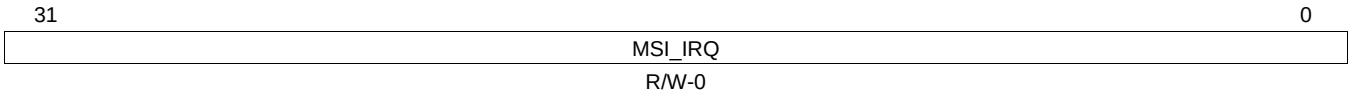
Table 17-28. EOI Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	EOI	0-3h	EOI for each interrupt. Write to indicate end-of-interrupt for the interrupt events. Write 0 to mark EOI for INTA/INTB/INTC/INTD, 1 to mark EOI for MSI interrupts and so on.

17.4.4.14 MSI_IRQ Register

The MSI interrupt IRQ register (MSI_IRQ) is described in the figure and table below.

Figure 17-26. MSI_IRQ Register



LEGEND: R/W = Read/Write; -*n* = value after reset

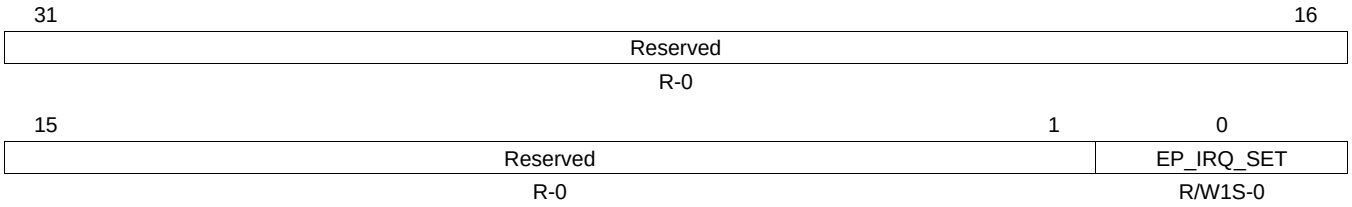
Table 17-29. MSI_IRQ Register Field Descriptions

Bit	Field	Value	Description
31-0	MSI_IRQ	0-FFFF FFFFh	This register is written to by the remote device. Writes initiated by an EP over PCIe link that target BAR0 of the RC land to this register if the offset matches. To generate MSI Interrupt 0, the EP should write 0000 0000h to this register. It will result in a pulse on bit 0 triggering the MSI interrupt from PCIESS to the external processor.

17.4.4.15 EP_IRQ_SET Register

The endpoint interrupt request set register (EP_IRQ_SET) is described in the figure and table below.

Figure 17-27. EP_IRQ_SET Register



LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set; -*n* = value after reset

Table 17-30. EP_IRQ_SET Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	EP_IRQ_SET	0	Write 1 to generate assert interrupt message. If MSI is disabled, legacy interrupt assert message will be generated. On read, a 1 indicates currently asserted interrupt.

17.4.4.16 EP_IRQ_CLR Register

The endpoint interrupt request clear register (EP_IRQ_CLR) is described in the figure and table below.

Figure 17-28. EP_IRQ_CLR Register

31	16
Reserved	
R-0	

15	1	0
Reserved		EP_IRQ_CLR
R-0		R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-31. EP_IRQ_CLR Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	EP_IRQ_CLR	0	Write 1 to generate deassert interrupt message. If MSI is disabled, legacy interrupt deassert message will be generated. On read, a 1 indicates currently asserted.

17.4.4.17 EP_IRQ_STATUS Register

The endpoint interrupt status register (EP_IRQ_STATUS) is described in the figure and table below.

Figure 17-29. EP_IRQ_STATUS Register

31	1
Reserved	
R-0	

15	1	0
Reserved		EP_IRQ_STATUS
R-0		R-0

LEGEND: R = Read only; -*n* = value after reset

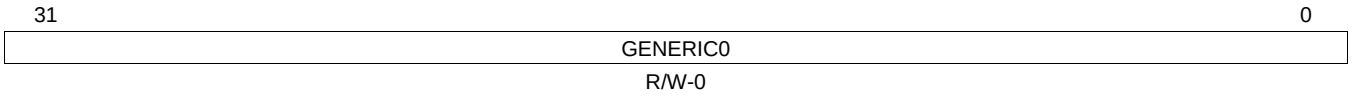
Table 17-32. EP_IRQ_STATUS Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	EP_IRQ_STATUS	0	Indicates whether interrupt for function 0 is asserted or not.

17.4.4.18 GPR0 Register

The general purpose 0 register (GPR0) is described in the figure and table below.

Figure 17-30. GPR0 Register



LEGEND: R/W = Read/Write; -*n* = value after reset

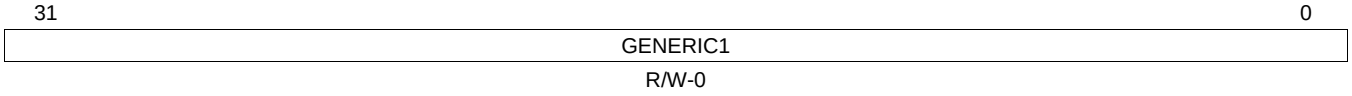
Table 17-33. GPR0 Register Field Descriptions

Bit	Field	Value	Description
31-0	GENERIC0	0-FFFF FFFFh	Generic Info field 0

17.4.4.19 GPR1 Register

The general purpose 1 register (GPR1) is described in the figure and table below.

Figure 17-31. GPR1 Register



LEGEND: R/W = Read/Write; -*n* = value after reset

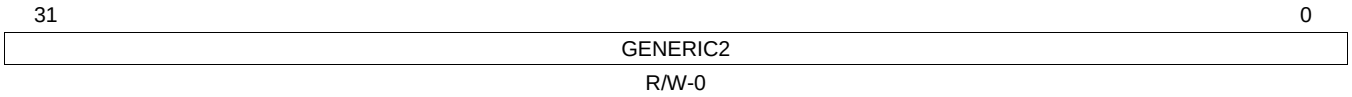
Table 17-34. GPR1 Register Field Descriptions

Bit	Field	Value	Description
31-0	GENERIC1	0-FFFF FFFFh	Generic Info field 1

17.4.4.20 GPR2 Register

The general purpose 2 register (GPR2) is described in the figure and table below.

Figure 17-32. GPR2 Register



LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-35. GPR2 Register Field Descriptions

Bit	Field	Value	Description
31-0	GENERIC2	0-FFFF FFFFh	Generic Info field 2

17.4.4.21 GPR3 Register

The general purpose 3 register (GPR3) is described in the figure and table below.

Figure 17-33. GPR3 Register

31	0
GENERIC3	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-36. GPR3 Register Field Descriptions

Bit	Field	Value	Description
31-0	GENERIC3	0-FFFF FFFFh	Generic Info field 3

17.4.4.22 MSI0_IRQ_STATUS_RAW Register

The MSI 0 interrupt raw status register (MSI0_IRQ_STATUS_RAW) is described in the figure and table below.

Figure 17-34. MSI0_STATUS_RAW Register

31	0
MSI0_RAW_STATUS	
R/W1S-0	

LEGEND: R/W = Read/Write; W1S = Write 1 to set; -*n* = value after reset

Table 17-37. MSI0_STATUS_RAW Register Field Descriptions

Bit	Field	Value	Description
31-0	MSI0_RAW_STATUS	0-FFFF FFFFh	Each bit indicates raw status of MSI vector associated with the bit. Typically, writes to this register are only done for debug purposes.

17.4.4.23 MSI0_IRQ_STATUS Register

The MSI 0 interrupt enabled status register (MSI0_IRQ_STATUS) is described in the figure and table below.

Figure 17-35. MSI0_IRQ_STATUS Register

31	0
MSI0_IRQ_STATUS	
R/W0C-0	

LEGEND: R/W = Read/Write; W0C = Write 0 to clear; -*n* = value after reset

Table 17-38. MSI0_IRQ_STATUS Register Field Descriptions

Bit	Field	Value	Description
31-0	MSI0_IRQ_STATUS	0-FFFF FFFFh	Each bit indicates raw status of MSI vector associated with the bit. Each of the bits can be written with a zero to clear the respective interrupt status bit.

17.4.4.24 MSI0_IRQ_ENABLE_SET Register

The MSI 0 interrupt enable set register (MSI0_IRQ_ENABLE_SET) is described in the figure and table below.

Figure 17-36. MSI0_IRQ_ENABLE_SET Register

31	0
MSI0_IRQ_ENABLE_SET	
R/W1S-0	

LEGEND: R/W = Read/Write; W1S = Write 1 to set; -*n* = value after reset

Table 17-39. MSI0_IRQ_ENABLE_SET Register Field Descriptions

Bit	Field	Value	Description
31-0	MSI0_IRQ_ENABLE_SET	0-FFFF FFFFh	Each bit, when written to, enables the MSI interrupt associated with the bit.

17.4.4.25 MSI0_IRQ_ENABLE_CLR Register

The MSI0 interrupt enable clear register (MSI0_IRQ_ENABLE_SET) is described in the figure and table below.

Figure 17-37. MSI0_IRQ_ENABLE_CLR Register

31	0
MSI0_IRQ_ENABLE_CLR	
R/W1C-0	

LEGEND: R/W = Read/Write; W1C = Write 1 to clear; -*n* = value after reset

Table 17-40. MSI0_IRQ_ENABLE_CLR Register Field Descriptions

Bit	Field	Value	Description
31-0	MSI0_IRQ_ENABLE_CLR	0-FFFF FFFFh	Each bit, when written to, disables the MSI interrupt associated with the bit.

17.4.4.26 IRQ_STATUS_RAW Register

The raw interrupt status register (IRQ_STATUS_RAW) is described in the figure and table below.

Figure 17-38. IRQ_STATUS_RAW Register

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		INTD	INTC	INTB	INTA
R-0		R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set; -*n* = value after reset

Table 17-41. IRQ_STATUS_RAW Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	INTD	0	Legacy Interrupt D raw status. RC mode only.
2	INTC	0	Legacy Interrupt C raw status. RC mode only.
1	INTB	0	Legacy Interrupt B raw status. RC mode only.
0	INTA	0	Legacy Interrupt A raw status. RC mode only.

17.4.4.27 IRQ_STATUS Register

The interrupt enabled status register (IRQ_STATUS) is described in the figure and table below.

Figure 17-39. IRQ_STATUS Register

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		INTD	INTC	INTB	INTA
R-0		R/W0C-0	R/W0C-0	R/W0C-0	R/W0C-0

LEGEND: R/W = Read/Write; R = Read only; W0C = Write 0 to clear; -*n* = value after reset

Table 17-42. IRQ_STATUS Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	INTD	0	Legacy Interrupt D status. Set when interrupt is active. Write Zero, '0', to clear the interrupt event. RC mode only.
2	INTC	0	Legacy Interrupt C status. Set when interrupt is active. Write Zero, '0', to clear the interrupt event. RC mode only.
1	INTB	0	Legacy Interrupt B status. Set when interrupt is active. Write Zero, '0', to clear the interrupt event. RC mode only.
0	INTA	0	Legacy Interrupt A status. Set when interrupt is active. Write Zero, '0', to clear the interrupt event. RC mode only.

17.4.4.28 IRQ_ENABLE_SET Register

The interrupt enable set register (IRQ_ENABLE_SET is described in the figure and table below.

Figure 17-40. IRQ_ENABLE_SET Register

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		INTD	INTC	INTB	INTA
R-0		R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set; -*n* = value after reset

Table 17-43. IRQ_ENABLE_SET Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	INTD	0	Legacy Interrupt D status. Set to enable the interrupt. On read, one/zero means interrupt is enabled/disabled.
2	INTC	0	Legacy Interrupt C status. Set to enable the interrupt. On read, one/zero means interrupt is enabled/disabled.
1	INTB	0	Legacy Interrupt B status. Set to enable the interrupt. On read, one/zero means interrupt is enabled/disabled.
0	INTA	0	Legacy Interrupt A status. Set to enable the interrupt. On read, one/zero means interrupt is enabled/disabled.

17.4.4.29 IRQ_ENABLE_CLR Register

The interrupt enable clear register (IRQ_ENABLE_CLR) is described in the figure and table below.

Figure 17-41. IRQ_ENABLE_CLR Register

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		INTD	INTC	INTB	INTA
R-0		R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-44. IRQ_ENABLE_CLR Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	INTD	0	Legacy Interrupt D disable. Set to enable the interrupt. On read, one/zero means interrupt is enabled/disabled.
2	INTC	0	Legacy Interrupt C disable. Set to enable the interrupt. On read, one/zero means interrupt is enabled/disabled.
1	INTB	0	Legacy Interrupt B disable. Set to enable the interrupt. On read, one/zero means interrupt is enabled/disabled.
0	INTA	0	Legacy Interrupt A disable. Set to enable the interrupt. On read, one/zero means interrupt is enabled/disabled.

17.4.4.30 ERR_IRQ_STATUS_RAW Register

The raw ERRinterrupt status register (ERR_IRQ_STATUS_RAW) is described in the figure and table below.

Figure 17-42. ERR_IRQ_STATUS_RAW Register

31	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved		ERR_AER	ERR_AXI	ERR_CORR	ERR_NONFATAL	ERR_FATAL	ERR_SYS
R-0		R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set; *-n* = value after reset

Table 17-45. ERR_IRQ_STATUS_RAW Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5	ERR_AER	0	ECRC error raw status.
4	ERR_AXI	0	AXI tag lookup fatal error raw status.
3	ERR_CORR	0	Correctable error raw status.
2	ERR_NONFATAL	0	Nonfatal error raw status.
1	ERR_FATAL	0	Fatal error raw status.
0	ERR_SYS	0	System Error (FATAL, NONFATAL, or CORRECTABLE error) raw status.

17.4.4.31 ERR_IRQ_STATUS Register

The ERR interrupt enabled status register (ERR_IRQ_STATUS) is described in the figure and table below.

Figure 17-43. ERR_IRQ_STATUS Register

31	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved		ERR_AER	ERR_AXI	ERR_CORR	ERR_NONFATAL	ERR_FATAL	ERR_SYS
R-0		R/W0C-0	R/W0C-0	R/W0C-0	R/W0C-0	R/W0C-0	R/W0C-0

LEGEND: R/W = Read/Write; R = Read only; W0C = Write 0 to clear; *-n* = value after reset

Table 17-46. ERR_IRQ_STATUS Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5	ERR_AER	0	ECRC error raw status.
4	ERR_AXI	0	AXI tag lookup fatal error raw status.
3	ERR_CORR	0	Correctable error raw status.
2	ERR_NONFATAL	0	Nonfatal error raw status.
1	ERR_FATAL	0	Fatal error raw status.
0	ERR_SYS	0	System Error (FATAL, NONFATAL, or CORRECTABLE error) raw status.

17.4.4.32 ERR_IRQ_ENABLE_SET Register

The ERR interrupt enable set register (ERR_IRQ_ENABLE_SET) is described in the figure and table below.

Figure 17-44. ERR_IRQ_ENABLE_SET Register

31	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved		ERR_AER	ERR_AXI	ERR_CORR	ERR_NONFATAL	ERR_FATAL	ERR_SYS
R-0		R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set; *-n* = value after reset

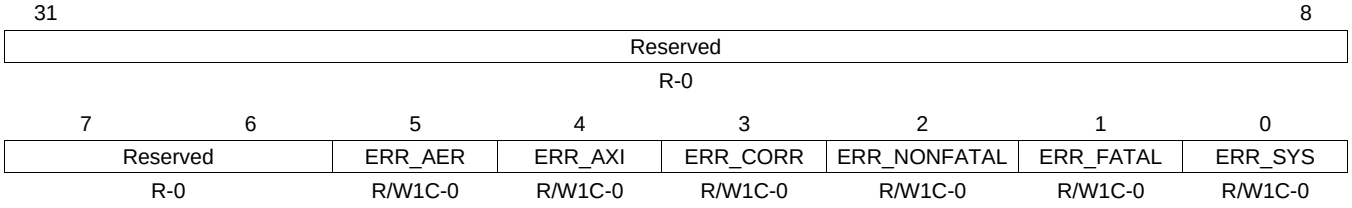
Table 17-47. ERR_IRQ_ENABLE_SET Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5	ERR_AER	0	ECRC error interrupt enable. Set to enable. On read, 1/0 means enabled/disabled, respectively.
4	ERR_AXI	0	AXI tag lookup fatal error interrupt enable. Set to enable. On read, 1/0 means enabled/disabled, respectively.
3	ERR_CORR	0	Correctable error interrupt enable. Set to enable. On read, 1/0 means enabled/disabled, respectively.
2	ERR_NONFATAL	0	Nonfatal error interrupt enable. Set to enable. On read, 1/0 means enabled/disabled, respectively.
1	ERR_FATAL	0	Fatal error interrupt enable. Set to enable. On read, 1/0 means enabled/disabled, respectively.
0	ERR_SYS	0	System Error (FATAL, NONFATAL, or CORRECTABLE error) interrupt enable. Set to enable. On read, 1/0 means enabled/disabled, respectively.

17.4.4.33 ERR_IRQ_ENABLE_CLR Register

The ERR interrupt enable clear register (ERR_IRQ_ENABLE_CLR) is described in the figure and table below.

Figure 17-45. ERR_IRQ_ENABLE_CLR Register



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-48. ERR_IRQ_ENABLE_CLR Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5	ERR_AER	0	ECRC error interrupt disable. Set to disable. On read, 1/0 means enabled/disabled, respectively.
4	ERR_AXI	0	AXI tag lookup fatal error interrupt disable. Set to disable. On read, 1/0 means enabled/disabled, respectively.
3	ERR_CORR	0	Correctable error interrupt disable. Set to disable. On read, 1/0 means enabled/disabled, respectively.
2	ERR_NONFATAL	0	Nonfatal error interrupt disable. Set to disable. On read, 1/0 means enabled/disabled, respectively.
1	ERR_FATAL	0	Fatal error interrupt disable. Set to disable. On read, 1/0 means enabled/disabled, respectively.
0	ERR_SYS	0	System Error (FATAL, NONFATAL, or CORRECTABLE error) interrupt disable. Set to disable. On read, 1/0 means enabled/disabled, respectively.

17.4.4.34 PMRST_IRQ_STATUS_RAW Register

The power management and reset interrupt status register (PMRST_IRQ_STATUS_RAW) is described in the figure and table below.

Figure 17-46. PMRST_IRQ_STATUS_RAW Register

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		LINK_RST_REQ	PM_PME	PM_TO_ACK	PM_TURNOFF
R-0		R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set; -*n* = value after reset

Table 17-49. PMRST_IRQ_STATUS_RAW Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	LINK_RST_REQ	0	Link Request Reset interrupt raw status
2	PM_PME	0	Power Management PME message received interrupt raw status
1	PM_TO_ACK	0	Power Management ACK received interrupt raw status
0	PM_TURNOFF	0	Power Management Turnoff message received raw status

17.4.4.35 PMRST_IRQ_STATUS Register

The power management and reset interrupt enabled status register (PMRST_IRQ_STATUS) is described in the figure and table below.

Figure 17-47. PMRST_IRQ_STATUS Register

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		LINK_RST_REQ	PM_PME	PM_TO_ACK	PM_TURNOFF
R-0		R/W0C-0	R/W0C-0	R/W0C-0	R/W0C-0

LEGEND: R/W = Read/Write; R = Read only; W0C = Write 0 to clear; -*n* = value after reset

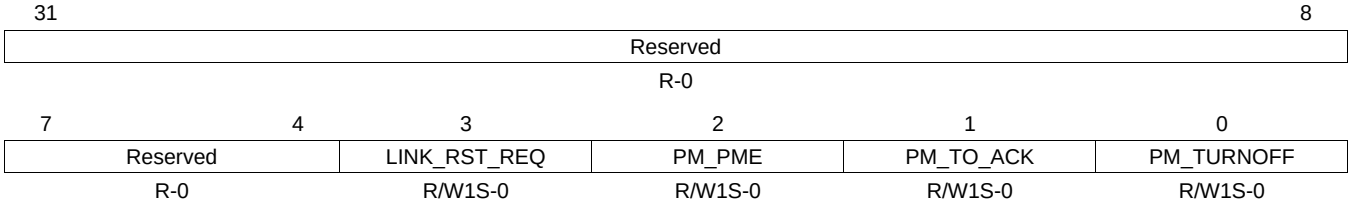
Table 17-50. PMRST_IRQ_STATUS Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	LINK_RST_REQ	0	Link Request Reset interrupt raw status
2	PM_PME	0	Power Management PME message received interrupt raw status
1	PM_TO_ACK	0	Power Management ACK received interrupt raw status
0	PM_TURNOFF	0	Power Management Turnoff message received raw status

17.4.4.36 PMRST_ENABLE_SET Register

The power management and reset interrupt enabled set register (PMRST_ENABLE_SET) is described in the figure and table below.

Figure 17-48. PMRST_ENABLE_SET Register



LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set; -*n* = value after reset

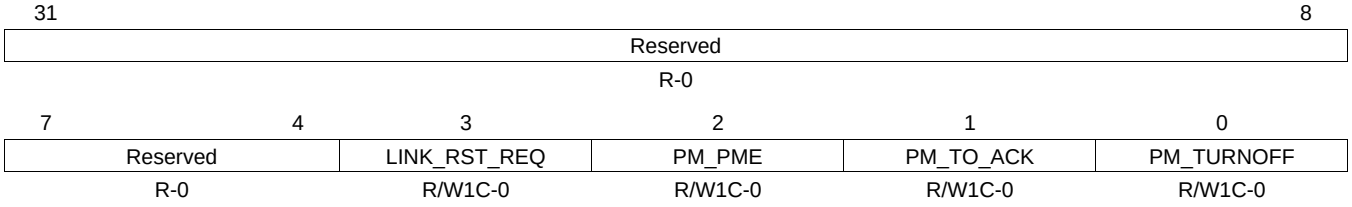
Table 17-51. PMRST_ENABLE_SET Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	LINK_RST_REQ	0	Link Request Reset interrupt enable. Set to enable the interrupt. Read 1 means interrupt is enabled.
2	PM_PME	0	Power Management PME message received interrupt enable. Set to enable the interrupt. Read 1 means interrupt is enabled.
1	PM_TO_ACK	0	Power Management ACK received interrupt enable. Set to enable the interrupt. Read 1 means interrupt is enabled.
0	PM_TURNOFF	0	Power Management Turnoff message received enable. Set to enable the interrupt. Read 1 means interrupt is enabled.

17.4.4.37 PMRST_ENABLE_CLR Register

The power management and reset interrupt enabled clear register (PMRST_ENABLE_CLR) is described in the figure and table below.

Figure 17-49. PMRST_ENABLE_CLR Register



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

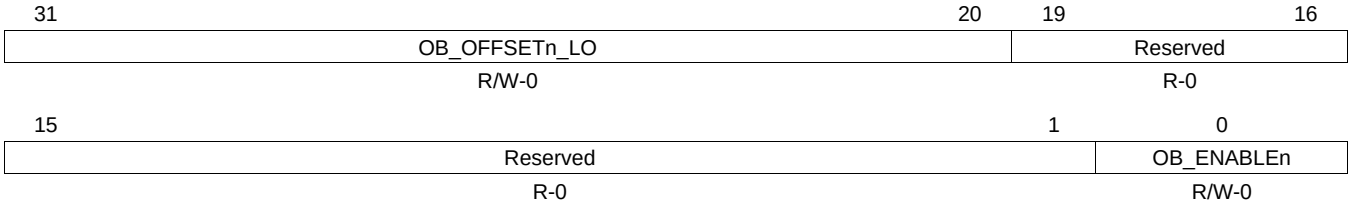
Table 17-52. PMRST_ENABLE_CLR Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	LINK_RST_REQ	0	Link Request Reset interrupt disable. Set to disable the interrupt. Read 1 means interrupt is enabled.
2	PM_PME	0	Power Management PME message received interrupt disable. Set to disable the interrupt. Read 1 means interrupt is enabled.
1	PM_TO_ACK	0	Power Management ACK received interrupt disable. Set to disable the interrupt. Read 1 means interrupt is enabled.
0	PM_TURNOFF	0	Power Management Turnoff message received disable. Set to disable the interrupt. Read 1 means interrupt is enabled.

17.4.4.38 OB_OFFSET_INDEXn Register

The outbound translation region N offset low and index register (OB_OFFSET_INDEXn) is described in the figure and table below.

Figure 17-50. OB_OFFSET_INDEXn Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-53. OB_OFFSET_INDEXn Register Field Descriptions

Bit	Field	Value	Description
31-20	OB_OFFSETn_LO	0-FFFh	Offset bits 31-20 for translation region N (N = 0-31)
19-1	Reserved	0	Reserved
0	OB_ENABLEn	0	Enable translation region N (N = 0-31)

17.4.4.39 OB_OFFSETn_HI Register

The outbound translation region N offset high register (OB_OFFSETn_HI) is described in the figure and table below.

Figure 17-51. OB_OFFSETn_HI Register

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

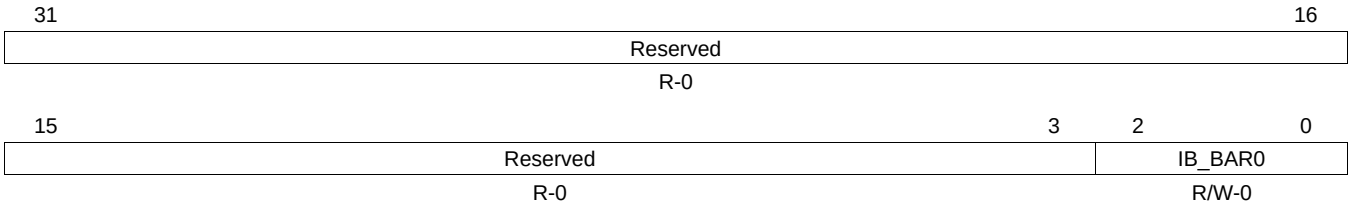
Table 17-54. OB_OFFSETn_HI Register Field Descriptions

Bit	Field	Value	Description
31-0	OB_OFFSETn_HI	0-FFFF FFFFh	Offset bits 31-0 for translation region N (N = 0-31)

17.4.4.40 IB_BAR0 Register

The inbound translation bar match 0 register (IB_BAR0) is described in the figure and table below.

Figure 17-52. IB_BAR0 Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

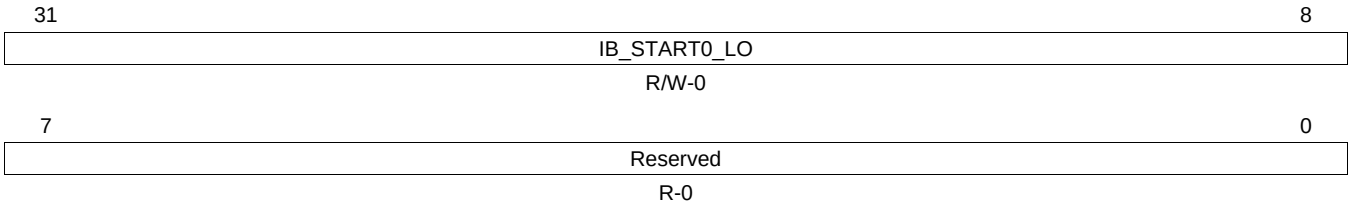
Table 17-55. IB_BAR0 Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	IB_BAR0	0-7h	BAR number to match for inbound translation region 0

17.4.4.41 IB_START0_LO Register

The inbound translation 0 start address low register (IB_START0_LO) is described in the figure and table below.

Figure 17-53. IB_START0_LO Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-56. IB_START0_LO Register Field Descriptions

Bit	Field	Value	Description
31-8	IB_START0_LO	0-FF FFFFh	Start address bits 31-8 for inbound translation region 0.
7-0	Reserved	0	Reserved

17.4.4.42 IB_START0_HI Register

The inbound translation 0 start address high register (IB_START0_HI) is described in the figure and table below.

Figure 17-54. IB_START0_HI Register

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

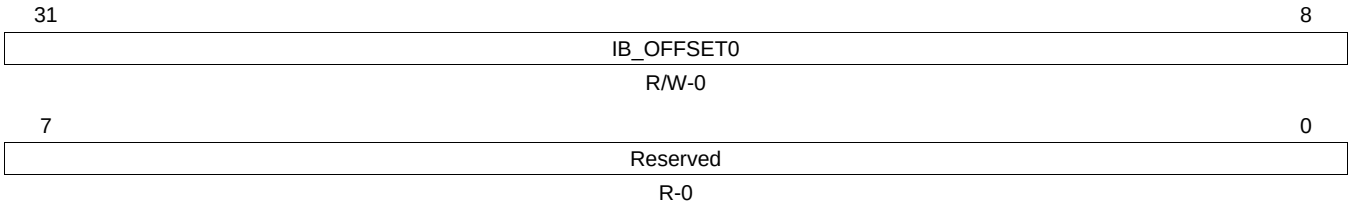
Table 17-57. IB_START0_HI Register Field Descriptions

Bit	Field	Value	Description
31-0	IB_START0_HI	0-FFFF FFFFh	Start address bits 63-0 for inbound translation region 0.

17.4.4.43 IB_OFFSET0 Register

The inbound translation 0 address offset register (IB_OFFSET0) is described in the figure and table below.

Figure 17-55. IB_OFFSET0 Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

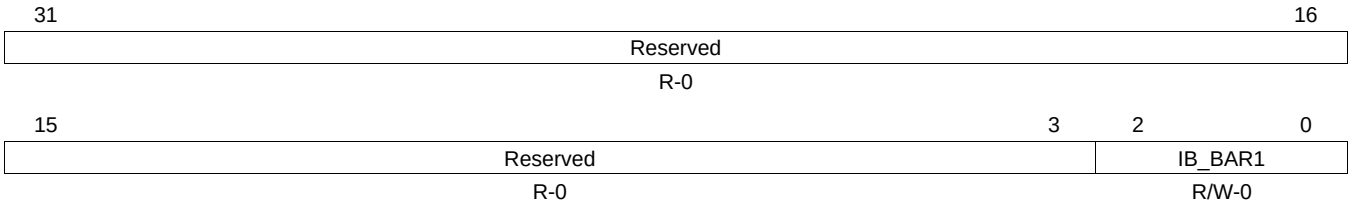
Table 17-58. IB_OFFSET0 Register Field Descriptions

Bit	Field	Value	Description
31-8	IB_OFFSET0	0-FF FFFFh	Offset address bits 31-8 for inbound translation region 0.
7-0	Reserved	0	Reserved

17.4.4.44 IB_BAR1 Register

The inbound translation bar match 1 register (IB_BAR1) is described in the figure and table below.

Figure 17-56. IB_BAR1 Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

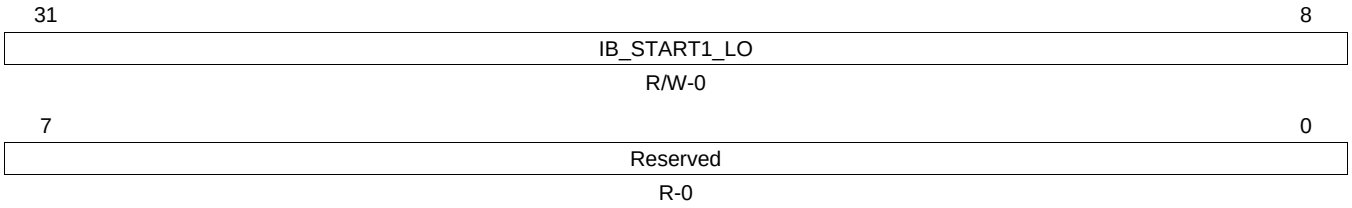
Table 17-59. IB_BAR1 Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	IB_BAR1	0-7h	BAR number to match for inbound translation region 1

17.4.4.45 IB_START1_LO Register

The inbound translation 1 start address low register (IB_START1_LO) is described in the figure and table below.

Figure 17-57. IB_START1_LO Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

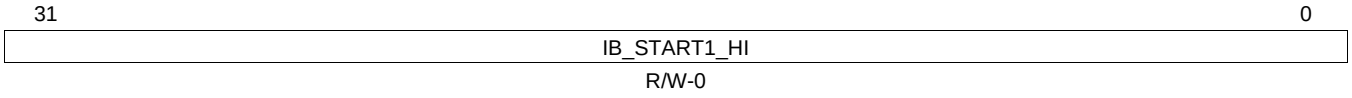
Table 17-60. IB_START1_LO Register Field Descriptions

Bit	Field	Value	Description
31-8	IB_START1_LO	0-FF FFFFh	Start address bits 31-8 for inbound translation region 1
7-0	Reserved	0	Reserved

17.4.4.46 IB_START1_HI Register

The inbound translation 1 start address high register (IB_START1_HI) is described in the figure and table below.

Figure 17-58. IB_START1_HI Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

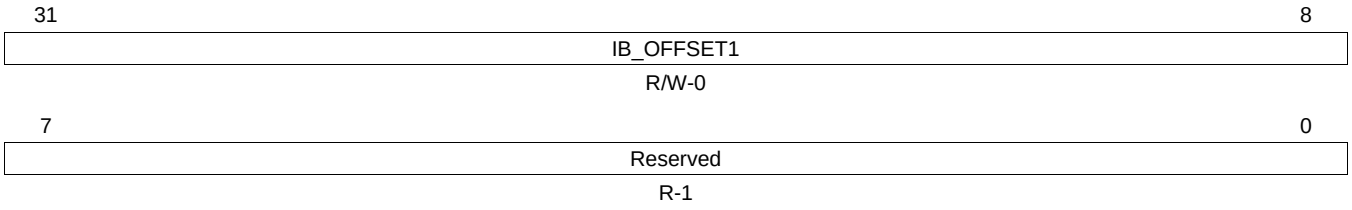
Table 17-61. IB_START1_HI Register Field Descriptions

Bit	Field	Value	Description
31-0	IB_START1_HI	0-FFFF FFFFh	Start address bits 63-0 for inbound translation region 1

17.4.4.47 IB_OFFSET1 Register

The inbound translation 1 address offset register (IB_OFFSET1) is described in the figure and table below.

Figure 17-59. IB_OFFSET1 Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

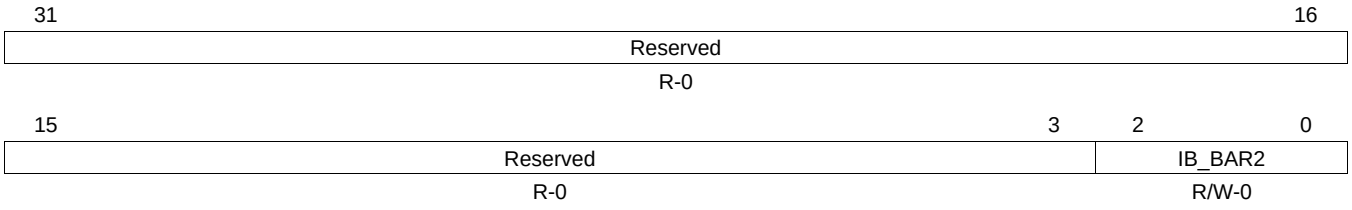
Table 17-62. IB_OFFSET 1 Register Field Descriptions

Bit	Field	Value	Description
31-8	IB_OFFSET1	0-FF FFFFh	Offset address bits 31-8 for inbound translation region 1
7-0	Reserved	0	Reserved

17.4.4.48 IB_BAR2 Register

The inbound translation bar match 2 register (IB_BAR2) is described in the figure and table below.

Figure 17-60. IB_BAR2 Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

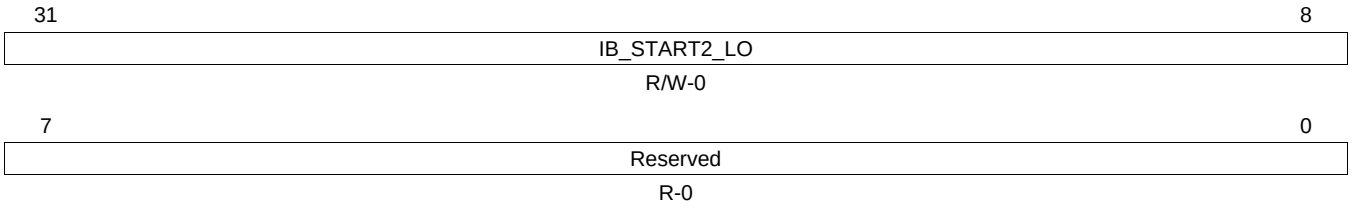
Table 17-63. IB_BAR2 Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	IB_BAR2	0-7h	BAR number to match for inbound translation region 2

17.4.4.49 IB_START2_LO Register

The inbound translation 2 start address low register (B_START2_LO) is described in the figure and table below.

Figure 17-61. IB_START2_LO Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

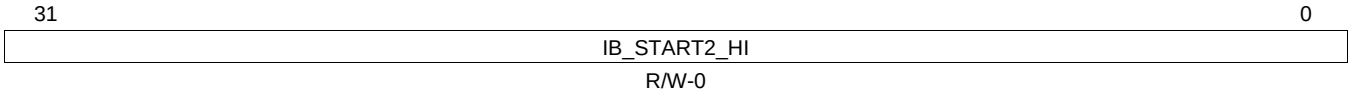
Table 17-64. IB_START2_LO Register Field Descriptions

Bit	Field	Value	Description
31-8	IB_START2_LO	0-FF FFFFh	Start address bits 31-8 for inbound translation region 2
7-0	Reserved	0	Reserved

17.4.4.50 IB_START2_HI Register

The inbound translation 2 start address low register (B_START2_HI) is described in the figure and table below.

Figure 17-62. IB_START2_HI Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

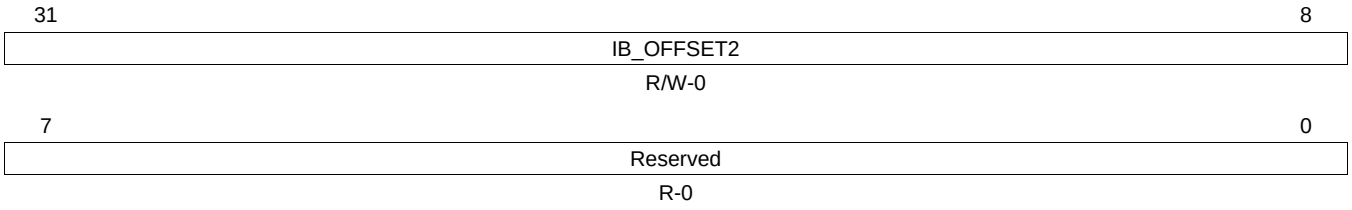
Table 17-65. IB_START2_HI Register Field Descriptions

Bit	Field	Value	Description
31-0	IB_START2_HI	0-FFFF FFFFh	Start address bits 63-0 for inbound translation region 2

17.4.4.51 IB_OFFSET2 Register

The inbound translation 2 address offset register (IB_OFFSET2) is described in the figure and table below.

Figure 17-63. IB_OFFSET2 Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

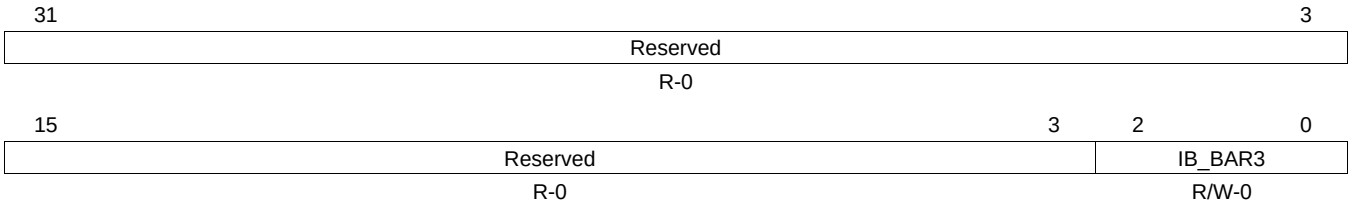
Table 17-66. IB_OFFSET2 Register Field Descriptions

Bit	Field	Value	Description
31-8	IB_OFFSET2	0-FF FFFFh	Offset address bits 31-8 for inbound translation region 2
7-0	Reserved	0	Reserved

17.4.4.52 IB_BAR3 Register

The inbound translation bar match 3 register (B_BAR3) is described in the figure and table below.

Figure 17-64. IB_BAR3 Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

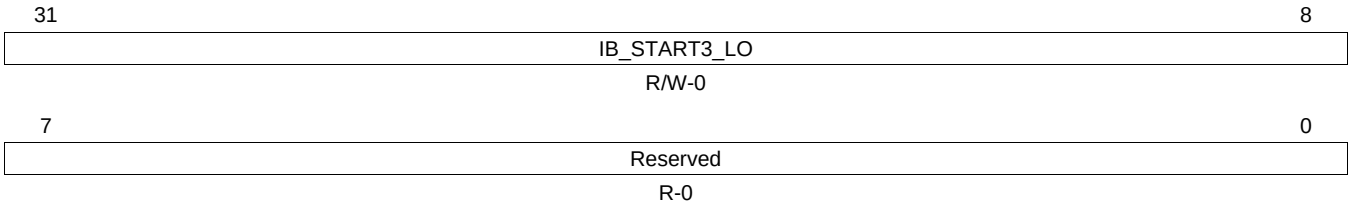
Table 17-67. IB_BAR3 Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	IB_BAR3	0-7h	BAR number to match for inbound translation region 3

17.4.4.53 IB_START3_LO Register

The inbound translation 3 start address low register (IB_START3_LO) is described in the figure and table below.

Figure 17-65. IB_START3_LO Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

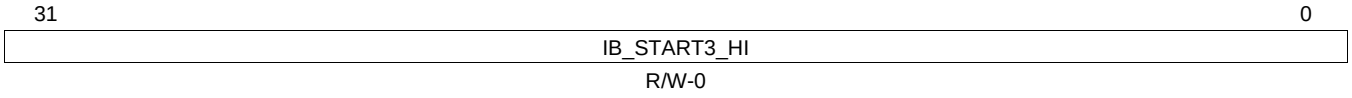
Table 17-68. IB_START3_LO Register Field Descriptions

Bit	Field	Value	Description
31-8	IB_START3_LO	0-FF FFFFh	Start address bits 31-8 for inbound translation region 3
7-0	Reserved	0	Reserved

17.4.4.54 IB_START3_HI Register

The inbound translation 3 start address high register (IB_START3_HI) is described in the figure and table below.

Figure 17-66. IB_START3_HI Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

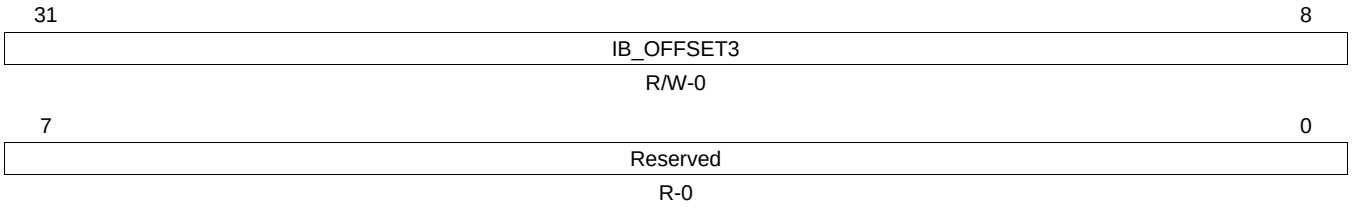
Table 17-69. IB_START3_HI Register Field Descriptions

Bit	Field	Value	Description
31-0	IB_START3_HI	0-FFFF FFFFh	Start address bits 63-0 for inbound translation region 3

17.4.4.55 IB_OFFSET3 Register

The inbound translation 3 address offset register (IB_OFFSET3) is described in the figure and table below.

Figure 17-67. IB_OFFSET3 Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

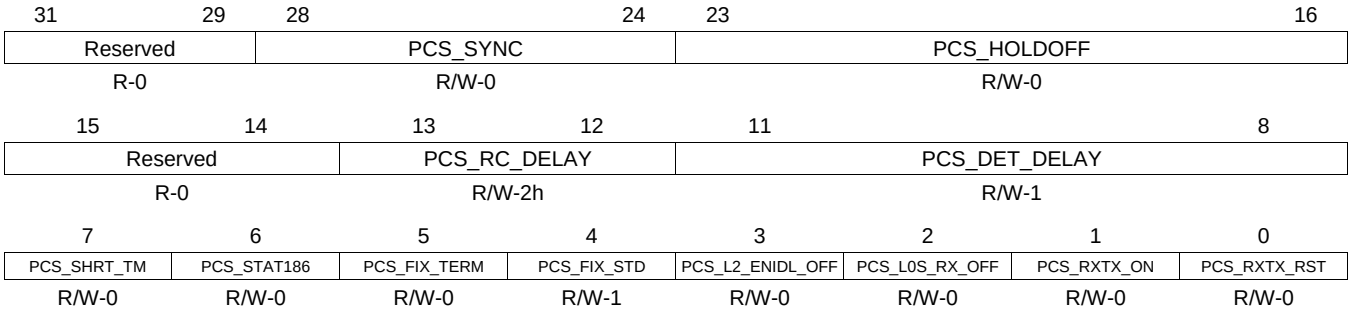
Table 17-70. IB_OFFSET3 Register Field Descriptions

Bit	Field	Value	Description
31-8	IB_OFFSET3	0-FF FFFFh	Offset address bits 31-8 for inbound translation region 3
7-0	Reserved	0	Reserved

17.4.4.56 PCS_CFG0 Register

The PCS configuration 0 (PCS_CFG0) register is described in the figure and table below

Figure 17-68. PCS_CFG0 Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

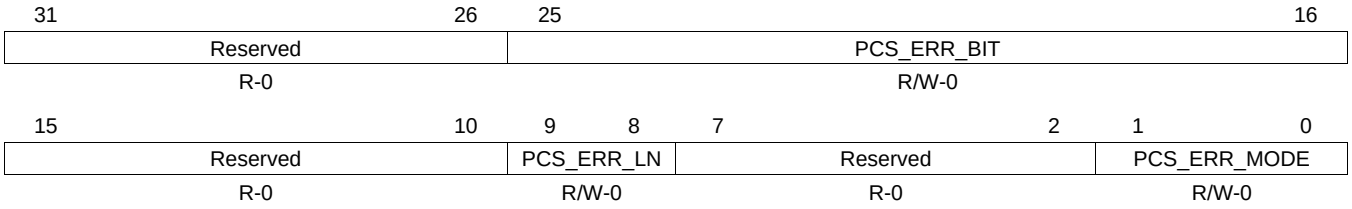
Table 17-71. PCS_CFG0 Register Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved
28-24	PCS_SYNC	0-1Fh	Receiver Lock/Sync Control
23-16	PCS_HOLDOFF	0-FFh	Receiver Initialization Hold Off Control
15-14	Reserved	0	Reserved
13-12	PCS_RC_DELAY	0-3h	Rate Change Delay
11-8	PCS_DET_DELAY	0-Fh	Detection Delay
7	PCS_SHRT_TM	0	Enable short times for debug purposes.
6	PCS_STAT186	0	Enable PIPE Spec 1.86 for PHY status behavior
5	PCS_FIX_TERM	0	Fed term output to 3'b100 during reset.
4	PCS_FIX_STD	0	Fix std output to 2'b10
3	PCS_L2_ENIDL_OFF	0	Deassert ENIDL during L2 state.
2	PCS_L0S_RX_OFF	0	Deassert Rx Enable in L0s state.
1	PCS_RXTX_ON	0	RX and TX on during reset. TX also on in P1 state.
0	PCS_RXTX_RST	0	RX and TX on during reset

17.4.4.57 PCS_CFG1 Register

The PCS configuration 1 (PCS_CFG1) register is described in the figure and table below.

Figure 17-69. PCS_CFG1 Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

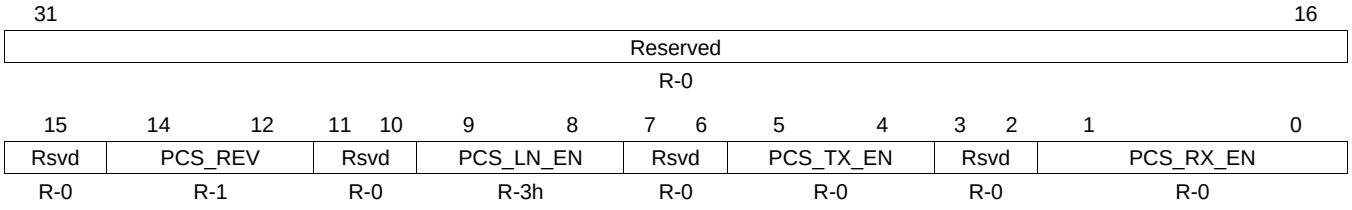
Table 17-72. PCS_CFG1 Register Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	PCS_ERR_BIT	0-3FFh	Error Bit enable
15-10	Reserved	0	Reserved
9-8	PCS_ERR_LN	0-3h	Error Lane enable
7-2	Reserved	0	Reserved
1-0	PCS_ERR_MODE	0-3h	Error Injection Mode

17.4.4.58 PCS_STATUS Register

The PCS status register is described in the figure and table below.

Figure 17-70. PCS_STATUS Register



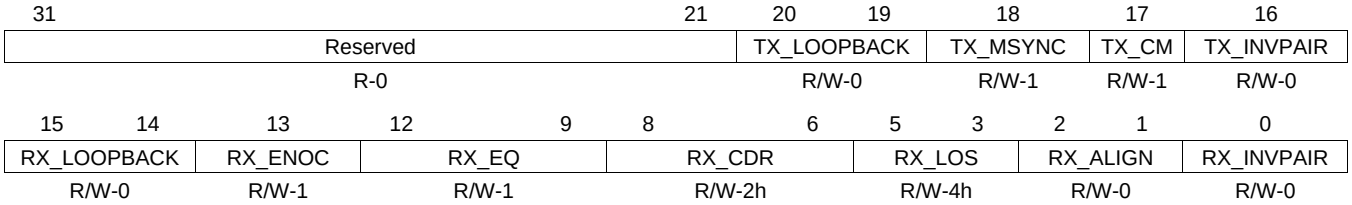
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-73. PCS_STATUS Register Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14-12	PCS_REV	0-7h	PCS RTL Revision
11-10	Reserved	0	Reserved
9-8	PCS_LN_EN	0-3h	PCS Lanes enabled status
7-6	Reserved	0	Reserved
5-4	PCS_TX_EN	0-3h	PCS Transmitters enabled status
3-2	Reserved	0	Reserved
1-0	PCS_RX_EN	0-3h	PCS Receivers enabled status

17.4.4.59 SERDES_CFG0 Register

Figure 17-71. SERDES_CFG0 Register



31	21	20	19	18	17	16
Reserved		TX_LOOPBACK		TX_MSYNC	TX_CM	TX_INVPAIR
R-0		R/W-0		R/W-1	R/W-1	R/W-0

15	14	13	12	9	8	6	5	3	2	1	0
RX_LOOPBACK		RX_ENOC	RX_EQ		RX_CDR		RX_LOS		RX_ALIGN		RX_INVPAIR
R/W-0		R/W-1	R/W-1		R/W-2h		R/W-4h		R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-74. SERDES_CFG0 Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved for future use.
20-19	TX_LOOPBACK	0-3h	Enable TX loopback. Set both bits high to enable. (CFGTX0[28:27])
18	TX_MSYNC	0	Master mode for synchronization (CFGTX0[22])
17	TX_CM	0	Enable common mode adjustment (CFGTX0[8])
16	TX_INVPAIR	0	Invert TX pair polarity (CFGTX0[7])
15-14	RX_LOOPBACK	0	Enable RX Loopback. Set both bits to high to enable loopback (CFGRX0[28:27])
13	RX_ENOC	0	Enable RX offset compensation (CFGRX0[23])
12-9	RX_EQ	0-Fh	Enable RX adaptive equalization (CFGRX0[22:19])
8-6	RX_CDR	0-7h	Enable RX clock data recovery (CFGRX0[18:16])
5-3	RX_LOS	0-7h	Enable RX loss of signal detection (CFGRX0[15:13])
2-1	RX_ALIGN	0-3h	Enable RX symbol alignment (CFGRX0[12:11])
0	RX_INVPAIR	0	Invert RX pair polarity (CFGRX0[7])

17.4.4.60 SERDES_CFG1 Register

Figure 17-72. SERDES_CFG1 Register

31	21	20	19	18	17	16
Reserved		TX_LOOPBACK		TX_MSYNC	TX_CM	TX_INVPAIR
R-0		R/W-0		R/W-0	R/W-1	R/W-0

15	14	13	12	9	8	6	5	3	2	1	0
RX_LOOPBACK		RX_ENOC	RX_EQ		RX_CDR		RX_LOS		RX_ALIGN		RX_INVPAIR
R/W-0		R/W-1	R/W-1		R/W-2h		R/W-4h		R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-75. SERDES_CFG1 Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved for future use.
20-19	TX_LOOPBACK	0-3h	Enable TX loopback. Set both bits high to enable. (CFGTX1[28:27])
18	TX_MSYNC	0	Master mode for synchronization (CFGTX1[22])
17	TX_CM	0	Enable common mode adjustment (CFGTX1[8])
16	TX_INVPAIR	0	Invert TX pair polarity (CFGTX1[7])
15-14	RX_LOOPBACK	0-3h	Enable RX Loopback. Set both bits to high to enable loopback (CFGRX1[28:27])
13	RX_ENOC	0	Enable RX offset compensation (CFGRX1[23])
12-9	RX_EQ	0-Fh	Enable RX adaptive equalization (CFGRX1[22:19])
8-6	RX_CDR	0-7h	Enable RX clock data recovery (CFGRX1[18:16])
5-3	RX_LOS	0-7h	Enable RX loss of signal detection (CFGRX1[15:13])
2-1	RX_ALIGN	0-3h	Enable RX symbol alignment (CFGRX1[12:11])
0	RX_INVPAIR	0	Invert RX pair polarity (CFGRX1[7])

17.4.5 Configuration Registers Common to Type 0 and Type 1 Headers

Table 17-76. Configuration Registers Common to Type 0 and Type 1 Headers

Offset	Acronym	Section
0h	VENDOR_DEVICE_ID	Section 17.4.5.1
4h	STATUS_COMMAND	Section 17.4.5.2
8h	CLASSCODE_REVID	Section 17.4.5.3

17.4.5.1 VENDOR_DEVICE_ID Register

The vendor and device identification register (VENDOR_DEVICE_ID) is described in the figure and table below.

Figure 17-73. VENDOR_DEVICE_ID Register

31	16
device_id	
R-8888h	

15	0
vendor_id	
R-104Ch	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-77. VENDOR_DEVICE_ID Register Field Descriptions

Bit	Field	Value	Description
31-16	device_id	0-FFFFh	PCIe Device ID. Writable from internal bus interface.
15-0	vendor_id	0-FFFFh	PCIe Vendor ID. Writable from internal bus interface.

17.4.5.2 STATUS_COMMAND Register

The status and command register (STATUS_COMMAND) is described in the figure and table below.

Figure 17-74. STATUS_COMMAND Register

31	30	29	28	27	26	25	24
Parity Error	Signaled System Error	Received Master Abort	Received Target Abort	Signaled Target Abort	Reserved		Data Parity Error
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R-0		R/W1C-0

23	21	20	19	18	16
Reserved		Capabilities List	Interrupt Status	Reserved	
R-0		R-1	R-0	R-0	

15	11	10	9	8
Reserved		INTx Disable	Reserved	SERR Enable
R-0		R/W-0	R-0	R/W-0

7	6	5	3	2	1	0
Reserved	Parity Error	Reserved		Bus Master	Memory Space	IO Space
R-0	R/W-0	R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-78. STATUS_COMMAND Register Field Descriptions

Bit	Field	Value	Description
31	Parity Error	0	Set if function receives poisoned TLP.
30	Signaled System Error	0	Set if function sends a ERR_FATAL or ERR_NONFATAL message and SERR enable bit is set to one.
29	Received Master Abort	0	Set when a Requester receives a Completion with Unsupported Request Completion Status.
28	Received Target Abort	0	Set when a Requester receives a Completion with Completer Abort Status.
27	Signaled Target Abort	0	Set when a function acting as a Completer terminates a request by issuing Completer Abort Completion Status to the Requester.
26-25	Reserved	0	Reserved
24	Data Parity Error	0	This bit is set by a Requester if the Parity Error Enable bit is set in its Command register and either the condition that the requester receives a poisoned Completion or the condition that the requester poisons a write request are true.
23-21	Reserved	0	Reserved
20	Capabilities List	1	For PCIe, this field must be set to 1.
19	Interrupt Status	0	Indicates that the function has received an interrupt.
18-11	Reserved	0	Reserved
10	INTx Disable	0	Setting this bit disables generation of INTx messages.
9	Reserved	0	Reserved
8	SERR Enable	0	When set, it enables System Error reporting to the Root Complex.
7	Reserved	0	Reserved
6	Parity Error	0	This bit controls whether or not the device responds to detected parity errors. If this bit is set, the PCIESS will respond normally to parity errors. If this bit is cleared, the PCIES will ignore detected parity errors.
5-3	Reserved	0	Reserved
2	Bus Master	0	Enables mastership of the bus.
1	Memory Space	0	This bit is set to enable the device to respond to memory accesses.
0	IO Space	0	This bit is set to enable the device to respond to I/O accesses. This functionality is not supported in PCIESS and there this bit is set to zero.

17.4.5.3 CLASSCODE_REVID Register

The class code and revision ID register (CLASSCODE_REVID) is described in the figure and table below.

Figure 17-75. CLASSCODE_REVID Register

31							8
Class Code							
R-0							
7							0
Revision ID							
R-1							

LEGEND: R = Read only; -*n* = value after reset

Table 17-79. CLASSCODE_REVID Register Field Descriptions

Bit	Field	Value	Description
31-8	Class Code	0-FF FFFFh	PCIe Class Code per PCIe Base Specifications Revision 2.0. Writable from internal bus interface.
7-0	Revision ID	0-FFh	Updated with each revision of hardware. Writable from internal bus interface.

17.4.6 Configuration Type 0 Registers

Table 17-80. Configuration Type 0 Registers

Offset	Acronym	Section
Ch	BIST_HEADER	Section 17.4.6.1
10h	BAR0	Section 17.4.6.2
14h	BAR1	Section 17.4.6.3
18h	BAR2	Section 17.4.6.4
1Ch	BAR3	Section 17.4.6.5
20h	BAR4	Section 17.4.6.6
24h	BAR5	Section 17.4.6.7
2Ch	SUBSYS_VNDR_ID	Section 17.4.6.8
30h	EXPNSN_ROM	Section 17.4.6.9
34h	CAP_PTR	Section 17.4.6.10
3Ch	INT_PIN	Section 17.4.6.11

17.4.6.1 BIST_HEADER Register

The BIST, Header Type, Latency Time and Cache Line Size register (BIST_HEADER) is described in the figure and table below.

Figure 17-76. BIST_HEADER Register

31	30	29 28	27 24	23	22 16
BIST Capable	Start BIST	Reserved	Completion Code	Multi-function Device	Header Type
R-0	R-0	R-0	R-0	R-0	R-0

15 8	7 0
Latency Timer	Cache Line Size
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-81. BIST_HEADER Register Field Descriptions

Bit	Field	Value	Description
31	BIST Capable	0	Returns a 1 for BIST capability and 0 otherwise. Not supported by PCIESS.
30	Start BIST	0	Write a 1 to start BIST. Not supported by PCIESS.
29-28	Reserved	0	Reserved
27-24	Completion Mode	0-Fh	Completion Code. Not supported by PCIESS.
23	Multi-function Device	0	Returns 1 if it is a multi function device. Writable from internal bus interface.
22-16	Header Type	0-7Fh	Configuration Header Format. It is set to 1 for RC and cleared to 0 for EP.
15-8	Latency Timer	0-FFh	Not applicable in PCIe
7-0	Cache Line Size	0-FFh	Not applicable in PCIe

17.4.6.2 BAR0 Register

The base address register 0 (BAR0) is described in the figure and table below.

Figure 17-77. BAR0 Register

31	8
Base Address	
R/W-0	

7	6	4	3	2	1	0
Base Address	Reserved		Prefetchable	Type		Memory Space
R/W-0	R-0		R-0	R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-82. BAR0 Register Field Descriptions

Bit	Field	Value	Description
31-7	Base Address	0-1FF FFFFh	Base Address
6-4	Reserved	0	Reserved
3	Prefetchable	0	For memory BARs, it indicates whether the region is prefetchable. For IO Bars, it is used as second LSB of the base address. Writable from internal bus interface.
2-1	Type	0-3h	Decode type. Writable from internal bus interface.
		0	32 bit decode
		1h	Reserved
		2h	64 bit decode
		3h	Reserved
0	Memory Space	0	Set to indicate Memory Space. Writable from internal bus interface.

17.4.6.3 BAR1 Register

The base address register 1 (BAR1) is described in the figure and table below.

Figure 17-78. BAR1 Register

31	8
Base Address	
R/W-0	

7	6	4	3	2	1	0
Base Address	Reserved		Prefetchable	Type		Memory Space
R/W-0	R-0		R-1	R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-83. BAR1 Register Field Descriptions

Bit	Field	Value	Description
31-7	Base Address	0-1FF FFFFh	Base Address
6-4	Reserved	0	Reserved
3	Prefetchable	0	For memory BARs, it indicates whether the region is prefetchable. For IO Bars, it is used as second LSB of the base address. Writable from internal bus interface.
2-1	Type	0-3h	Decode type. Writable from internal bus interface.
		0	32 bit decode
		1h	Reserved
		2h	64 bit decode
		3h	Reserved
0	Memory Space	0	Set to indicate Memory Space. Writable from internal bus interface.

17.4.6.3.1 BAR1 Register

The base address register 1 (BAR1) is described in the figure and table below.

Figure 17-79. BAR1 Register

31	0
Base Address	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-84. BAR1 Register Field Descriptions

Bit	Field	Value	Description
31-0	Base Address	0-FFFF FFFFh	Upper 32 bits of BAR0 address

17.4.6.4 BAR2 Register

The base address register 2 (BAR2) is described in the figure and table below.

Figure 17-80. BAR2 Register

31	8
Base Address	
R/W-0	

7	6	4	3	2	1	0
Base Address	Reserved		Prefetchable	Type		Memory Space
R/W-0	R-0		R-1	R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-85. BAR2 Register Field Descriptions

Bit	Field	Value	Description
31-7	Base Address	0-1FF FFFFh	Base Address
6-4	Reserved	0	Reserved
3	Prefetchable	0	For memory BARs, it indicates whether the region is prefetchable. For IO Bars, it is used as second LSB of the base address. Writable from internal bus interface.
2-1	Type	0-3h	Decode type. Writable from internal bus interface.
		0	32 bit decode
		1h	Reserved
		2h	64 bit decode
		3h	Reserved
0	Memory Space	0	Set to indicate Memory Space. Writable from internal bus interface.

17.4.6.5 BAR3 Register

The base address register 3 (BAR3) is described in the figure and table below.

Figure 17-81. BAR3 Register

31	8
Base Address	
R/W-0	

7	6	4	3	2	1	0
Base Address	Reserved		Prefetchable	Type		Memory Space
R/W-0	R-0		R-1	R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-86. BAR3 Register Field Descriptions

Bit	Field	Value	Description
31-7	Base Address	0-1FF FFFFh	Base Address
6-4	Reserved	0	Reserved
3	Prefetchable	0	For memory BARs, it indicates whether the region is prefetchable. For IO Bars, it is used as second LSB of the base address. Writable from internal bus interface.
2-1	Type	0-3h	Decode type. Writable from internal bus interface.
		0	32 bit decode
		1h	Reserved
		2h	64 bit decode
		3h	Reserved
0	Memory Space	0	Set to indicate Memory Space. Writable from internal bus interface.

17.4.6.5.1 BAR3 Register

The base address register 3 (BAR3) is described in the figure and table below.

Figure 17-82. BAR3 Register

31	0
Base Address	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-87. BAR3 Register Field Descriptions

Bit	Field	Value	Description
31-0	Base Address	0-FFFF FFFFh	Upper 32 bits of BAR2 address

17.4.6.6 BAR4 Register

The base address register 4 (BAR4) is described in the figure and table below.

Figure 17-83. BAR4 Register

31	8
Base Address	
R/W-0	

7	6	4	3	2	1	0
Base Address	Reserved		Prefetchable	Type		Memory Space
R/W-0	R-0		R-1	R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-88. BAR4 Register Field Descriptions

Bit	Field	Value	Description
31-7	Base Address	0-1FF FFFFh	Base Address
6-4	Reserved	0	Reserved
3	Prefetchable	0	For memory BARs, it indicates whether the region is prefetchable. For IO Bars, it is used as second LSB of the base address. Writable from internal bus interface.
2-1	Type	0-3h	Decode type. Writable from internal bus interface.
		0	32 bit decode
		1h	Reserved
		2h	64 bit decode
		3h	Reserved
0	Memory Space	0	Set to indicate Memory Space. Writable from internal bus interface.

17.4.6.7 BAR5 Register

The base address register 5 (BAR5) is described in the figure and table below.

Figure 17-84. BAR5 Register

31	8
Base Address	
R/W-0	

7	6	4	3	2	1	0
Base Address	Reserved		Prefetchable	Type		Memory Space
R/W-0	R-0		R-1	R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-89. BAR5 Register Field Descriptions

Bit	Field	Value	Description
31-7	Base Address	0-1FF FFFFh	Base Address
6-4	Reserved	0	Reserved
3	Prefetchable	0	For memory BARs, it indicates whether the region is prefetchable. For IO Bars, it is used as second LSB of the base address. Writable from internal bus interface.
2-1	Type	0-3h	Decode type. Writable from internal bus interface.
		0	32 bit decode
		1h	Reserved
		2h	64 bit decode
		3h	Reserved
0	Memory Space	0	Set to indicate Memory Space. Writable from internal bus interface.

17.4.6.7.1 BAR5 Register

The base address register 5 (BAR5) is described in the figure and table below.

Figure 17-85. BAR5 Register

31	0
Base Address	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

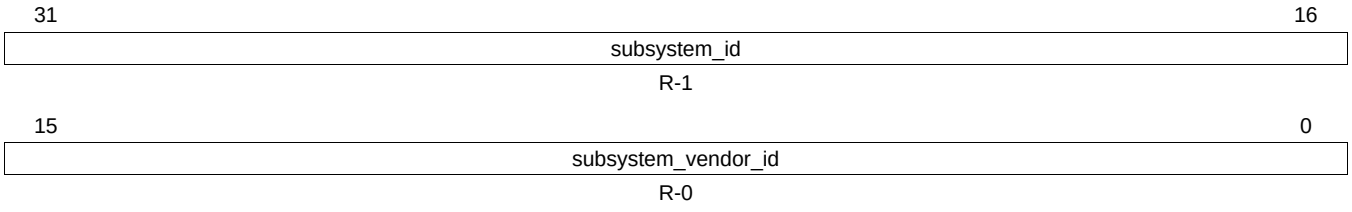
Table 17-90. BAR5 Register Field Descriptions

Bit	Field	Value	Description
31-0	Base Address	0-FFFF FFFFh	Upper 32 bits of BAR4 address

17.4.6.8 SUBSYS_VNDR_ID Register

The subsystem and subsystem vendor ID register (SUBSYS_VNDR_ID) is described in the figure and table below.

Figure 17-86. SUBSYS_VNDR_ID Register



LEGEND: R = Read only; -*n* = value after reset

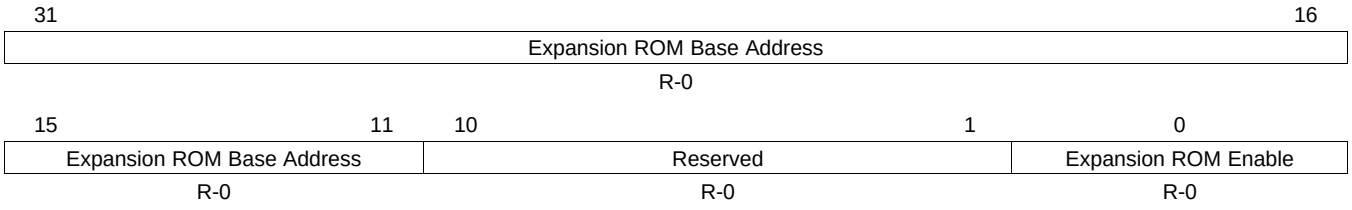
Table 17-91. SUBSYS_VNDR_ID Register Field Descriptions

Bit	Field	Value	Description
31-16	subsystem_id	0-FFFFh	PCIe Subsystem ID (TBD). Writable from internal bus interface.
15-0	subsystem_vendor_id	0-FFFFh	PCIe Subsystem Vendor ID (TBD). Writable from internal bus interface.

17.4.6.9 EXPNSN_ROM Register

The expansion ROM base address (EXPNSN_ROM) is described in the figure and table below.

Figure 17-87. EXPNSN_ROM Register



LEGEND: R = Read only; -*n* = value after reset

Table 17-92. EXPNSN_ROM Register Field Descriptions

Bit	Field	Value	Description
31-11	Expansion ROM Base Address	0-1F FFFFh	Address of Expansion ROM
10-1	Reserved	0	Reserved
0	Expansion ROM Enable	0	Expansion ROM Enable

17.4.6.10 CAP_PTR Register

The capabilities pointer register (CAP_PTR) is described in the figure and table below.

Figure 17-88. CAP_PTR Register

31	8
Reserved	
R-0	

7	0
cap_ptr	
R-40h	

LEGEND: R = Read only; -*n* = value after reset

Table 17-93. CAP_PTR Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	cap_ptr	0-FFh	First Capability Pointer. By default, it points to Power Management Capability structure. Writable from internal bus interface.

17.4.6.11 INT_PIN Register

Figure 17-89. INT_PIN Register

31	24	23	16
max_latency		min_grant	
R-0		R-0	

15	8	7	0
int_pin		int_line	
R-1		R/W-FFh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-94. INT_PIN Register Field Descriptions

Bit	Field	Value	Description
31-24	max_latency	0-FFh	Not applicable to PCI Express
23-16	min_grant	0-FFh	Not applicable to PCI Express
15-8	int_pin	0-FFh	Interrupt Pin. Valid values at 00h, 01h, 02h, 03h and 04h for unused legacy interrupt, INTA, INTB, INTC or INTD respectively. For single function configuration, the core only uses INTA. Writable from internal bus interface.
7-0	int_line	0-FFh	Interrupt Line

17.4.7 Configuration Type 1 Registers

Table 17-95. Configuration Type 1 Registers

Offset	Acronym	Section
Ch	BIST_HEADER	Section 17.4.7.1
10h	BAR0	Section 17.4.7.2
14h	BAR1	Section 17.4.7.3
18h	BUSNUM	Section 17.4.7.4
1Ch	SECSTAT	Section 17.4.7.5
20h	MEMSPACE	Section 17.4.7.6
24h	PREFETCH_MEM	Section 17.4.7.7
28h	PREFETCH_BASE	Section 17.4.7.8
2Ch	PREFETCH_LIMIT	Section 17.4.7.9
30h	IOSPACE	Section 17.4.7.10
34h	CAP_PTR	Section 17.4.7.11
38h	EXPNSN_ROM	Section 17.4.7.12
3Ch	BRIDGE_INT	Section 17.4.7.13

17.4.7.1 BIST_HEADER Register

The BIST, Header Type, Latency Time and Cache Line Size register (BIST_HEADER) is described in the figure and table below.

Figure 17-90. BIST_HEADER Register

31	30	29 28	27 24	23	22 16
BIST Capable	Start BIST	Reserved	Completion Code	Multi-function Device	Header Type
R-0	R-0	R-0	R-0	R-0	R-1

15 8	7 0
Latency Timer	Cache Line Size
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-96. BIST_HEADER Register Field Descriptions

Bit	Field	Value	Description
31	BIST Capable	0	Returns a 1 for BIST capability and 0 otherwise. Not supported by PCIESS.
30	Start BIST	0	Write a 1 to start BIST. Not supported by PCIESS.
29-28	Reserved	0	Reserved
27-24	Completion Code	0-Fh	Completion Code. Not supported by PCIESS.
23	Multi-function Device	0	Returns 1 if it is a multi function device. Writable from internal bus interface.
22-16	Header Type	0-7Fh	Configuration Header Format. It is set to 1 for RC and cleared to 0 for EP.
15-8	Latency Timer	0-FFh	Not applicable in PCIe.
7-0	Cache Line Size	0-FFh	Not applicable in PCIe.

17.4.7.2 BAR0 Register

The base address register 0 (64/32 bit mode) is described in the figure and table below.

Figure 17-91. BAR0 Register

31	8
Base Address	
R-0	

7	4	3	2	1	0
Base Address		Prefetchable	Type		Memory Space
R-0		R-0	R-0		R-0

LEGEND: R = Read only; -*n* = value after reset

Table 17-97. BAR0 Register Field Descriptions

Bit	Field	Value	Description
31-4	Base Address	0-FFF FFFFh	Base Address. Actual writable bits are determined by BAR0 Mask Register.
3	Prefetchable	0	Set to indicate prefetchable space. This field is writable from internal bus interface.
2-1	Type	0-3h	Decode type. Writable from internal bus interface.
		0	32 bit decode
		1h	Reserved
		2h	64 bit decode
		3h	Reserved
0	Memory Space	0	Set to indicate Memory Space. Writable from internal bus interface.

17.4.7.3 BAR1 Register

Figure 17-92. BAR1 Register

31	8
Base Address	
R-0	

7	4	3	2	1	0
Base Address		Prefetchable	Type		Memory Space
R-0		R-1	R-0		R-0

LEGEND: R = Read only; -*n* = value after reset

Table 17-98. BAR1 Register Field Descriptions

Bit	Field	Value	Description
31-4	Base Address	0-FFF FFFFh	Base Address. Actual writable bits are determined by BAR0 Mask Register.
3	Prefetchable	0	Set to indicate prefetchable space. This field is writable from internal bus interface.
2-1	Type	0-3h	Decode type. Writable from internal bus interface.
		0	32 bit decode
		1h	Reserved
		2h	64 bit decode
		3h	Reserved
0	Memory Space	0	Set to indicate Memory Space. Writable from internal bus interface.

17.4.7.3.1 BAR1 Register

The base address register 1(BAR1) is described in the figure and table below.

Figure 17-93. BAR1 Register

31	0
Base Address	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-99. BAR1 Register Field Descriptions

Bit	Field	Value	Description
31-0	Base Address	0-FFFF FFFFh	Base Address high 32 bits for BAR0. Actual writable bits are determined by BAR0 Mask Register.

17.4.7.4 BUSNUM Register

The latency timer and bus number register (BUSNUM) is described in the figure and table below.

Figure 17-94. BUSNUM Register

31	24	23	16
Secondary Latency Timer		Subordinate Bus Number	
R-0		R/W-0	

15	8	7	0
Secondary Bus Number		Primary Bus Number	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-100. BUSNUM Register Field Descriptions

Bit	Field	Value	Description
31-24	Secondary Latency Timer	0-FFh	Not applicable in PCI Express.
23-16	Subordinate Bus Number	0-FFh	Subordinate Bus Number. This is highest bus number on downstream interface.
15-8	Secondary Bus Number	0-FFh	Secondary Bus Number. It is typically 1h for RC.
7-0	Primary Bus Number	0-FFh	Primary Bus Number. It is zero for RC and nonzero for switch devices only.

17.4.7.5 SECSTAT Register

The secondary status and IO base/limit register (SECSTAT) is described in the figure and table below.

Figure 17-95. SECSTAT Register

31	30	29	28	27	26	25	24
DTCT_PERROR	RX_SYS_ERROR	RX_MST_ABORT	RX_TGT_ABORT	TX_TGT_ABORT	Reserved		MST_DPERR
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R-0		R/W1C-0

23	16
Reserved	
R-0	

15	12	11	9	8
IO Limit		Reserved		IO Addressing
R/W-0		R-0		R-0

7	4	3	1	0
IO Base		Reserved		IO Addressing
R/W-0		R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-101. SECSTAT Register Field Descriptions

Bit	Field	Value	Description
31	DTCT_PERROR	0	Detected Parity Error
30	RX_SYS_ERROR	0	Received System Error
29	RX_MST_ABORT	0	Received Master Abort
28	RX_TGT_ABORT	0	Received Target Abort
27	TX_TGT_ABORT	0	Signaled Target Abort
26-25	Reserved	0	Reserved
24	MST_DPERR	0	Master Data Parity Error
23-16	Reserved	0	Reserved
15-12	IO Limit	0-Fh	I/O Space Limit
11-9	Reserved	0	Reserved
8	IO Addressing	0	32 bit IO Space indication for IO Limit Register. Indicates 16 bit and 32 bit addressing for 0 and 1 respectively.
7-4	IO Base	0-Fh	IO Space Base
3-1	Reserved	0	Reserved
0	IO Addressing	0	Indicates 0 for 16 bit and 1 for 32 bit addressing for the IO Base Register. Writable from internal bus interface.

17.4.7.6 MEMSPACE Register

The memory limit and base register (MEMSPACE) is described in the figure and table below.

Figure 17-96. MEMSPACE Register

31	20	19	16
Memory Limit		Reserved	
R/W-0		R-0	

15	4	3	0
Memory Base		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-102. MEMSPACE Register Field Descriptions

Bit	Field	Value	Description
31-20	Memory Limit	0-FFFh	Memory Limit Address
19-16	Reserved	0	Reserved
15-4	Memory Base	0-FFFh	Memory Base Address
3-0	Reserved	0	Reserved

17.4.7.7 PREFETCH_MEM Register

The prefetchable memory limit and base register (PREFETCH_MEM) is described in the figure and table below.

Figure 17-97. PREFETCH_MEM Register

31	20	19	17	16
End Address		Reserved		Memory Addressing
R/W-0		R-0		R-0

15	4	3	1	0
Start Address		Reserved		Memory Addressing
R/W-0		R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-103. PREFETCH_MEM Register Field Descriptions

Bit	Field	Value	Description
31-20	End Address	0-FFFh	Upper 12 bits of 32 bit Prefetchable Memory End Address
19-17	Reserved	0	Reserved
16	Memory Addressing		Memory addressing. Writable from internal bus interface.
		0	32 bit memory addressing
		1	64 bit memory addressing
15-4	Start Address	0-FFFh	Upper 12 bits of 32 bit Prefetchable Memory Start Address
3-1	Reserved	0	Reserved
0	Memory Addressing		Memory addressing. Writable from internal bus interface.
		0	32 bit memory addressing
		1	64 bit memory addressing

17.4.7.8 PREFETCH_BASE Register

The prefetchable memory base upper 32 bits register (PREFETCH_BASE) is described in the figure and table below.

Figure 17-98. PREFETCH_BASE Register

31	0
Base Address	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-104. PREFETCH_BASE Field Descriptions

Bit	Field	Value	Description
31-0	Base Address	0-FFFF FFFFh	Upper 32 bits of base address of prefetchable memory space. Used when 64 bit addressing is enabled.

17.4.7.9 PREFETCH_LIMIT Register

The prefetchable limit upper 32 bits register (PREFETCH_LIMIT) is described in the figure and table below.

Figure 17-99. PREFETCH_LIMIT Register

31	0
Base Address	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-105. PREFETCH_LIMIT Register Field Descriptions

Bit	Field	Value	Description
31-0	Limit Address	0-FFFF FFFFh	Upper 32 bits of Limit Address of Prefetchable Memory Space. Used with 64 bit prefetchable memory addressing only.

17.4.7.10 IOSPACE Register

The IO base and limit upper 16 bits register (IOSPACE) is described in the figure and table below.

Figure 17-100. IOSPACE Register

31	16
IOBASE	
R/W-0	
15	0
IOLIMIT	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

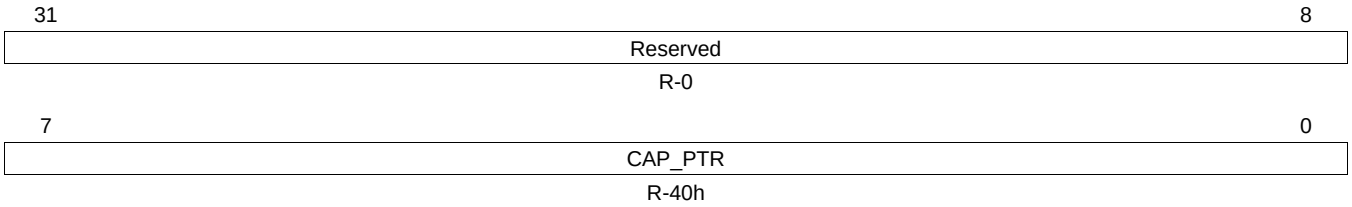
Table 17-106. IOSPACE Register Field Descriptions

Bit	Field	Value	Description
31-16	IOBASE	0-FFFFh	Upper 16 bits of IO Base
15-0	IOLIMIT	0-FFFFh	Upper 16 bits of IO Limit

17.4.7.11 CAP_PTR Register

The capabilities pointer register (CAP_PTR) is described in the figure and table below.

Figure 17-101. CAP_PTR Register



LEGEND: R = Read only; -*n* = value after reset

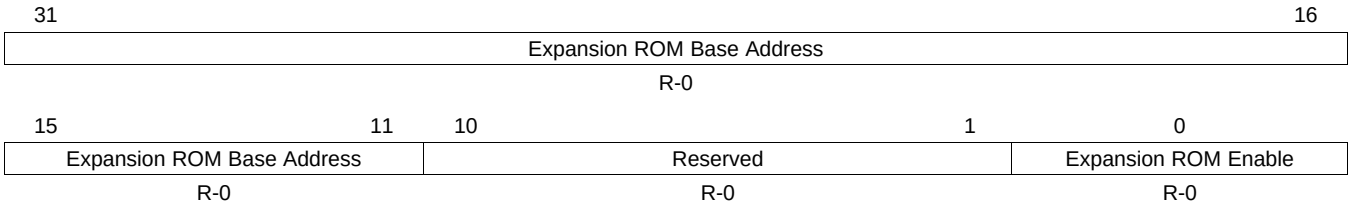
Table 17-107. CAP_PTR Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	CAP_PTR	0-FFh	First Capability Pointer. By default, it points to Power Management Capability structure. Writable from internal bus interface.

17.4.7.12 EXPNSN_ROM Register

The expansion ROM base address register (EXPNSN_ROM) is described in the figure and table below.

Figure 17-102. EXPNSN_ROM Register



LEGEND: R = Read only; -*n* = value after reset

Table 17-108. EXPNSN_ROM Register Field Descriptions

Bit	Field	Value	Description
31-11	Expansion ROM Base Address	0-1F FFFFh	Address of Expansion ROM
10-1	Reserved	0	Reserved
0	Expansion ROM Enable	0	Expansion ROM Enable

17.4.7.13 BRIDGE_INT Register

The bridge control register (BRIDGE_INT) is described in the figure and table below.

Figure 17-103. BRIDGE_INT Register

31	30	29	28	27	26	25	24
Reserved				SERREN_STATUS	TIMER_STATUS	SEC_TIMER	PRI_TIMER
R-0				R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
B2B_EN	SEC_BUS_RST	MST_ABORT_MODE	VGA_DECODE	VGA_EN	ISA_EN	SERR_EN	PERR_RESP_EN
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
INT_PIN							
R-1							
7	6	5	4	3	2	1	0
INT_LINE							
R/W-FFh							

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-109. BRIDGE_INT Register Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27	SERREN_STATUS	0	Discard Timer SERR Enable Status. Not Applicable to PCI Express.
26	TIMER_STATUS	0	Discard Timer Status. Not applicable to PCI Express.
25	SEC_TIMER	0	Secondary Discard Timer. Not applicable to PCI Express
24	PRI_TIMER	0	Primary Discard Timer. Not applicable to PCI Express.
23	B2B_EN	0	Fast Back to Back Transactions Enable. Not applicable to PCI Express.
22	SEC_BUS_RST	0	Secondary Bus Reset
21	MST_ABORT_MODE	0	Master Abort Mode. Not applicable to PCI Express. Always zero.
20	VGA_DECODE	0	VGA 16 bit Decode
19	VGA_EN	0	VGA Enable
18	ISA_EN	0	ISA Enable
17	SERR_EN	0	SERR Enable
16	PERR_RESP_EN	0	Parity Error Response Enable
15-8	INT_PIN	0-FFh	Interrupt Pin. It identifies the legacy interrupt message that the device uses. Valid values are 00h for legacy interrupt not used and 01h, 02h, 03h and 04h for INTA, INTB, INTC or INTD respectively. For single function configuration, the core only uses INTA. This register is writable through internal bus interface.
7-0	INT_LINE	0-FFh	Interrupt Line. Value is system software specified.

17.4.8 PCIe Capability Registers

Table 17-110. PCIe Capability Registers

Offset	Acronym	Section
0h	PCIE_CAP	Section 17.4.8.1
4h	DEVICE_CAP	Section 17.4.8.2
8h	DEV_STAT_CTRL	Section 17.4.8.3
Ch	LINK_CAP	Section 17.4.8.4
10h	LINK_STAT_CTRL	Section 17.4.8.5
14h	SLOT_CAP	Section 17.4.8.6
18h	SLOT_STAT_CTRL	Section 17.4.8.7
1Ch	ROOT_CTRL_CAP	Section 17.4.8.8
20h	ROOT_STATUS	Section 17.4.8.9
24h	DEV_CAP2	Section 17.4.8.10
28h	DEV_STAT_CTRL2	Section 17.4.8.11
30h	LINK_CTRL2	Section 17.4.8.12

17.4.8.1 PCIE_CAP Register

The PCI Express capabilities register (PCIE_CAP) is described in t he figure and table below.

Figure 17-104. PCIE_CAP Register

31 30	29 25	24	23 20	19 16
Reserved	INT_MSG	SLT_IMPL_N	DPORT_TYPE	PCIE_CAP
R-0	R-0	R-0	R-x	R-2h

15 8	7 0
NEXT_CAP	CAP_ID
R-0	R-10h

LEGEND: R = Read only; -*n* = value after reset

Table 17-111. PCIE_CAP Register Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved
29-25	INT_MSG	0-1Fh	Interrupt Message Number. Updated by hardware and writable through internal bus Interface.
24	SLT_IMPL_N	0	Slot Implemented. Writable from internal bus interface.
23-20	DPORT_TYPE	0-Fh	Device Port Type (4h for RC, 0 for EP)
19-16	PCIE_CAP	0-Fh	PCI Express Capability Version
15-8	NEXT_CAP	0-FFh	PCIe Next Capability Pointer containing the offset to next capability structure. Writable from internal bus interface.
7-0	CAP_ID	0-FFh	PCIe Capability ID

17.4.8.2 DEVICE_CAP Register

The device capabilities register (DEVICE_CAP) is described in the figure and table below.

Figure 17-105. DEVICE_CAP Register

31–28	27–26	25–18	17–16
Reserved	PWR_LIMIT_SCALE	PWR_LIMIT_VALUE	Reserved
R-0	R-0	R-0	R-0

15	14–12	11–9	8–6	5	4–3	2–0
ERR_RPT	Reserved	L1_LATENCY	L0_LATENCY	EXT_TAG_FLD	PHANTOM_FLD	MAX_PAYLD_SZ
R-1	R-0	R-x	R-x	R-0	R-0	R-1

LEGEND: R = Read only; -*n* = value after reset

Table 17-112. DEVICE_CAP Register Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-26	PWR_LIMIT_SCALE	0-3h	Captured Slot Power Limit Scale
25-18	PWR_LIMIT_VALUE	0-FFh	Captured Slow Power Limit Value
17-16	Reserved	0	Reserved
15	ERR_RPT	0	Role based Error Reporting. Writable from internal bus interface.
14-12	Reserved	0	Reserved
11-9	L1_LATENCY	0-7h	Endpoint L1 Acceptable Latency (0 in RC mode, 3h for Endpoint)
8-6	L0_LATENCY	0-7h	Endpoint L0 Acceptable Latency (0 in RC mode, 4h for Endpoint)
5	EXT_TAG_FLD	0	Extended Tag Field Supported. Writable from internal interface but should not be as the hardware is not capable.
4-3	PHANTOM_FLD	0-3h	Phantom Field Supported. Writable from internal bus interface.
2-0	MAX_PAYLD_SZ	0-7h	Maximum Payload size supported. Writable from internal bus interface.

17.4.8.3 DEV_STAT_CTRL Register

Figure 17-106. DEV_STAT_CTRL Register

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved		TPEND	AUX_PWR	UNSUP_RQ_DET	FATAL_ERR	NFATAL_ERR	CORR_ERR
R-0		R-0	R-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

15	14	13	12	11	10	9	8
Reserved	MAX_REQ_SZ			NO_SNOOP	AUX_PWR_PM_EN	PHANTOM_EN	XTAG_FIELD_EN
R-0	R/W-2h			R/W-1	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
MAX_PAYLOAD			RELAXED	UNSUP_REQ_REP	FATAL_ERR_REP	NFATAL_ERR_REP	CORR_ERR_REP
R/W-0			R/W-1	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-113. DEV_STAT_CTRL Register Field Descriptions

Bit	Field	Value	Description
31-22	Reserved	0	Reserved
21	TPEND	0	Transaction Pending
20	AUX_PWR	0	Auxiliary Power Detected
19	UNSUP_RQ_DET	0	Unsupported Request Detected
18	FATAL_ERR	0	Fatal Error Detected
17	NFATAL_ERR	0	Non-fatal Error Detected
16	CORR_ERR	0	Correctable Error Detected
15	Reserved	0	Reserved
14-12	MAX_REQ_SZ	0-7h	Maximum Read Request Size
11	NO_SNOOP	0	Enable no snoop
10	AUX_PWR_PM_EN	0	AUX Power PM Enable
9	PHANTOM_EN	0	Phantom Function Enable
8	XTAG_FIELD_EN	0	Extended Tag Field Enable
7-5	MAX_PAYLOAD	0-7h	Maximum Payload Size
4	RELAXED	0	Enable Relaxed Ordering
3	UNSUP_REQ_REP	0	Enable Unsupported Request Reporting
2	FATAL_ERR_REP	0	Fatal Error Reporting Enable
1	NFATAL_ERR_REP	0	Non-Fatal Error Reporting Enable
0	CORR_ERR_REP	0	Correctable Error Reporting Enable

17.4.8.4 LINK_CAP Register

Figure 17-107. LINK_CAP Register

31	30	29	28	27	26	25	24
PORT_NUM							
R-0							

23	22	21	20	19	18	17	16
Reserved		BW_NOTIFY_CAP	DLL_REP_CAP	DOWN_ERR_REP_CAP	CLK_PWR_MGMT	L1_EXIT_LAT	
R-0		R-0	R-0	R-0	R-0	R-6h	

15	14	13	12	11	10	9	8
L1_EXIT_LAT	LOS_EXIT_LAT			AS_LINK_PM		MAX_LINK_WIDTH	
R-6h	R-5h			R-1		R-0	

7	6	5	4	3	2	1	0
MAX_LINK_WIDTH				MAX_LINK_SPEED			
R-2h				R-2h			

LEGEND: R = Read only; -*n* = value after reset

Table 17-114. LINK_CAP Register Field Descriptions

Bit	Field	Value	Description
31-24	PORT_NUM	0-FFh	Port Number. Writable from internal bus interface.
23-22	Reserved	0	Reserved
21	BW_NOTIFY_CAP	0	Link Bandwidth Notification Capable. Always 1 for downstream and 0 for upstream.
20	DLL_REP_CAP	0	Data Link Layer Active Reporting Capable. Always 1 for downstream and 0 for upstream.
19	DOWN_ERR_REP_CAP	0	Surprise Down Error Reporting Capable. Not supported. Always zero.
18	CLK_PWR_MGMT	0	Clock Power Management. Zero for downstream ports. Writable from internal bus interface.
17-15	L1_EXIT_LAT	0-7h	L1 Exit Latency when common clock is used. Writable from internal bus interface.
14-12	LOS_EXIT_LAT	0-7h	L0s Exit Latency. Writable from internal bus interface.
11-10	AS_LINK_PM	0-3h	Active State Link PM Support. Writable from internal bus interface. By default, L0s is enabled and L1 is disabled.
9-4	MAX_LINK_WIDTH	0-3Fh	Maximum Link Width. Writable from internal bus interface.
3-0	MAX_LINK_SPEED	0-Fh	Maximum Link Speed. Writable from internal bus interface.

17.4.8.5 LINK_STAT_CTRL Register

Figure 17-108. LINK_STAT_CTRL Register

31	30	29	28	27	26	25	24
LINK_BW_STATUS	LINK_BW_MGMT_STATUS	DLL_ACTIVE	SLOT_CLK_CFG	LINK_TRAINING	UNDEF	NEGOTIATED_LINK_WD	
R/W1C-0	R/W1C-0	R-0	R-1	R-0	R-0	R-0	

23	20	19	16
NEGOTIATED_LINK_WD		LINK_SPEED	
R-1		R-1	

15	12	11	10	9	8
Reserved		LINK_BW_INIT_EN	LINK_BW_MGMT_INT_EN	HW_AUTO_WIDTH_DIS	CLK_PWR_MGMT_EN
R-0		R-0	R-0	R-0	R/W-0

7	6	5	4	3	2	1	0
EXT_SYNC	COMMON_CLK_CFG	RETRAIN_LINK	LINK_DISABLE	RCB	Reserved	ACTIVE_LINK_PM	
R/W-0	R/W-0	R/W-0	R/W-0	R-1	R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-115. LINK_STAT_CTRL Register Field Descriptions

Bit	Field	Value	Description
31	LINK_BW_STATUS	0	Link Autonomous Bandwidth Status. NA and Reserved for End Point.
30	LINK_BW_MGMT_STATUS	0	Link Bandwidth Management Status. NA and Reserved for End Point.
29	DLL_ACTIVE	0	Data Link Layer Active
28	SLOT_CLK_CFG	0	Slot Clock Configuration. Writable from internal bus interface.
27	LINK_TRAINING	0	Link Training. Not applicable to Root Complex.
26	UNDEF	0	Undefined for PCI Express
25-20	NEGOTIATED_LINK_WD	0-3Fh	Negotiated Link Width. Set automatically by hardware after link initialization.
19-16	LINK_SPEED	0-Fh	Link Speed. Set automatically by hardware after link initialization.
15-12	Reserved	0	Reserved
11	LINK_BW_INT_EN	0	Link Autonomous Bandwidth Interrupt Enable. Not applicable and is Reserved for End Point.
10	LINK_BW_MGMT_INT_EN	0	Link Bandwidth Management Interrupt Enable. Not applicable and is Reserved for End Point.
9	HW_AUTO_WIDTH_DIS	0	Hardware Autonomous Width Disable. Not supported and hardwired to zero.
8	CLK_PWR_MGMT_EN	0	Enable Clock Power Management
7	EXT_SYNC	0	Extended Synch
6	COMMON_CLK_CFG	0	Common Clock Configuration
5	RETRAIN_LINK	0	Retrain Link. Not applicable and Reserved for EP.
4	LINK_DISABLE	0	Link disable.
3	RCB	0	Read Completion Boundary. Writable via internal bus interface for Root Complex.
2	Reserve	0	Reserved
1-0	ACTIVE_LINK_PM	0-3h	Active State Link PM Control

17.4.8.6 SLOT_CAP Register

Figure 17-109. SLOT_CAP Register

31	30	29	28	27	26	25	24
SLOT_NUM							
R-0							

23	22	21	20	19	18	17	16
SLOT_NUM					CMD_COMP_SUPP	EML_PRESENT	PWR_LMT_SCALE
R-0					R-0	R-0	R-0

15	14	13	12	11	10	9	8
PWR_LMT_SCALE	PWR_LMT_VALUE						
R-0	R-0						

7	6	5	4	3	2	1	0
PWR_LMT_VALUE	HP_CAP	HP_SURPRISE	PWR_IND	ATTN_IND	MRL_SENSOR	PWR_CTL	ATTN_BUTTON
R-0	R-1	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 17-116. SLOT_CAP Register Field Descriptions

Bit	Field	Value	Description
31-19	SLOT_NUM	0-1FFF	Physical Slot Number. Writable from internal bus interface.
18	CMD_COMP_SUPP	0	No Command Complete Support. Writable from internal bus interface.
17	EML_PRESENT	0	Electromechanical Interlock Present. Writable from internal bus interface.
16-15	PWR_LMT_SCALE	0-3h	Slow Power Limit Scale. Writable from internal bus interface.
14-7	PWR_LMT_VALUE	0-FFh	Slow Power Limit Value. Writable from internal bus interface.
6	HP_CAP	0	Hot Plug Capable. Writable from internal bus interface.
5	HP_SURPRISE	0	Hot Plug Surprise. Writable from internal bus interface.
4	PWR_IND	0	Power Indicator Present. Writable from internal bus interface.
3	ATTN_IND	0	Attention Indicator Present. Writable from internal bus interface.
2	MRL_SENSOR	0	MRL Sensor Present. Writable from internal bus interface.
1	PWR_CTL	0	Power Controller Present. Writable from internal bus interface. If there is no power controller, software must ensure that system power is up before reading Presence Detect state.
0	ATTN_BUTTON	0	Attention Indicator Present. Writable from internal bus interface.

17.4.8.7 SLOT_STAT_CTRL Register

Figure 17-110. SLOT_STAT_CTRL Register

31						25	24
Reserved							DLL_STATE
R-0							R/W1C-0
23	22	21	20	19	18	17	16
EM_LOCK	PRESENCE_DET	MRL_STATE	CMD_COMLETE	PRESENCE_CHG	MRL_CHANGE	PWR_FAULT	ATTN_PRESSED
R-0	R-1	R-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0
15		13	12	11	10	9	8
Reserved			DLL_CHG_EN	EM_LOCK_CTL	PM_CTL	PM_IND_CTL	
R-0			R/W-0	R/W-0	R/W-0	R/W-3h	
7	6	5	4	3	2	1	0
ATTN_IND_CTL		HP_INT_EN	CMD_CMP_INT_EN	PRS_DET_CHG_EN	MRL_CHG_EN	PWR_FLT_DET_EN	ATTN_BUTT_EN
R/W-3h		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-117. SLOT_STAT_CTRL Register Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24	DLL_STATE	0	Data Link Layer State Changed
23	EM_LOCK	0	Electromechanical Lock Status
22	PRESENCE_DET	0	Presence Detect State
21	MRL_STATE	0	MRL Sensor State
20	CMD_COMLETE	0	Command Completed
19	PRESENCE_CHG	0	Presence Detect Changed
18	MRL_CHANGE	0	MRL Sensor Changed
17	PWR_FAULT	0	Power Fault Detected
16	ATTN_PRESSED	0	Attention Button Pressed
15-13	Reserved	0	Reserved
12	DLL_CHG_EN	0	Data Link Layer State Changed Enable
11	EM_LOCK_CTL	0	Electromechanical Interlock Control
10	PM_CTL	0	Power Controller Control
9-8	PM_IND_CTL	0-3h	Power Indicator Control
7-6	ATTN_IND_CTL	0-3h	Attention Indicator Control
5	HP_INT_EN	0	Hot Plug Interrupt Enable
4	CMD_CMP_INT_EN	0	Command Completed Interrupt Enable
3	PRS_DET_CHG_EN	0	Presence Detect Changed Enable
2	MRL_CHG_EN	0	MRL Sensor Changed Enable
1	PWR_FLT_DET_EN	0	Power Fault Detected Enable
0	ATTN_BUTT_EN	0	Attention Button Pressed Enable

17.4.8.8 ROOT_CTRL_CAP Register

Figure 17-111. ROOT_CTRL_CAP Register

31	17	16
Reserved		CRS_SW
R-0		R-0

15	8
Reserved	
R-0	

7	5	4	3	2	1	0
Reserved		CRS_SW_EN	PME_INT_EN	SERR_FATAL_ERR	SERR_NFATAL_ERR	SERR_EN
R-0		R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-118. ROOT_CTRL_CAP Register Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	CRS_SW	0	CRS Software Visibility. Not Supported. Hardwired to Zero.
15-5	Reserved	0	Reserved
4	CRS_SW_EN	0	CRS Software Visibility Enable. Not Supported and set to 0h.
3	PME_INT_EN	0	PME Interrupt Enable
2	SERR_FATAL_ERR	0	System Error on Fatal Error Enable
1	SERR_NFATAL_ERR	0	System Error on Non-fatal Error Enable
0	SERR_EN	0	System Error on Correctable Error Enable

17.4.8.9 ROOT_STATUS Register

Figure 17-112. ROOT_STATUS Register

31	18	17	16
Reserved		PME_PEND	PME_STATUS
R-0		R-0	R/W1C-0

15	0
PME_REQ_ID	
R-0	

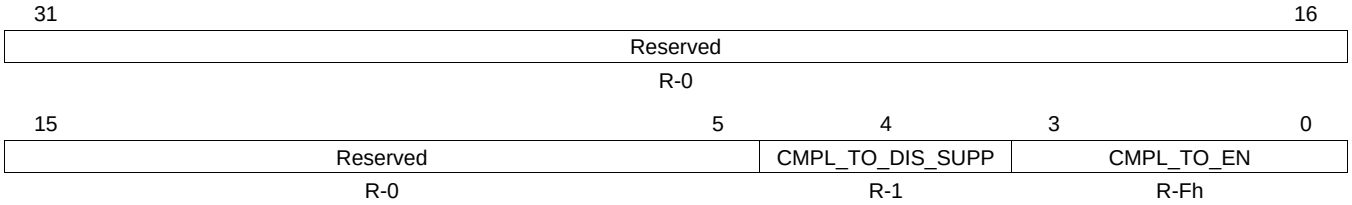
LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-119. ROOT_STATUS Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	PME_PEND	0	PME is pending.
16	PME_STATUS	0	Indicates that PME was asserted by the PME Requester.
15-0	PME_REQ_ID	0	ID of the last PME Requester.

17.4.8.10 DEV_CAP2 Register

Figure 17-113. DEV_CAP2 Register



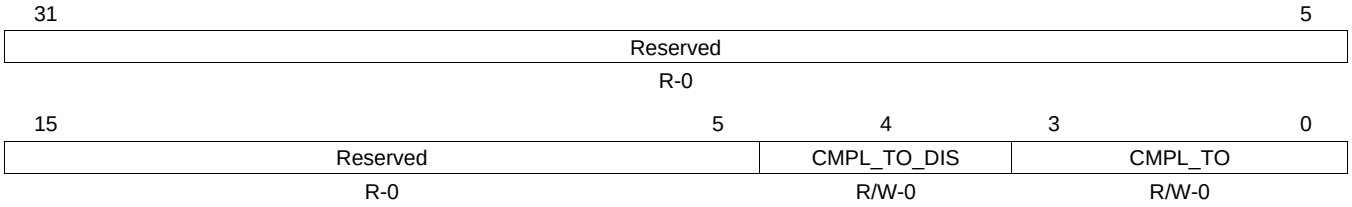
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-120. DEV_CAP2 Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	CMPL_TO_DIS_SUPP	0	Completion timeout disable supported
3-0	CMPL_TO_EN	0-Fh	Completion timeout ranges supported. Applicable to RC/EP.

17.4.8.11 DEV_STAT_CTRL2 Register

Figure 17-114. DEV_STAT_CTRL2 Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-121. DEV_STAT_CTRL2 Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	CMPL_TO_DIS	0	Completion timeout disabled
3-0	CMPL_TO	0-Fh	Completion timeout value

17.4.8.12 LINK_CTRL2 Register

Figure 17-115. LINK_CTRL2 Register

31	17	16
Reserved		DE_EMPH
R-0		R-1

15	13	12	11	10	9	8
Reserved		POLL_DEEMPH	CMPL_SOS	ENTR_MOD_COMPL	TX_MARGIN	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	

7	6	5	4	3	0
TX_MARGIN	SEL_DEEMPH	HW_AUTO_SPEED_DIS	ENTR_COMPL	TGT_SPEED	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-2h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-122. LINK_CTRL2 Register Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	DE_EMPH	0	Current De-emphasis level
15-13	Reserved	0	Reserved
12	POLL_DEEMPH	0	DE-emphasis level in polling-compliance state
11	CMPL_SOS	0	Compliance SOS
10	ENTR_MOD_COMPL	0	Enter modified compliance
9-7	TX_MARGIN	0-7h	Value of non-de-emphasized voltage level at transmitter pins
6	SEL_DEEMPH	0	Selectable De-emphasis (0 for 6 dB and 1 for 3.5 dB)
5	HW_AUTO_SPEED_DIS	0	Hardware autonomous speed disable
4	ENTR_COMPL	0	Enter compliance
3-0	TGT_SPEED	0-Fh	Gen-1 is 1h and Gen-2 is 2h.

17.4.9 PCIe Extended Capability Registers

Table 17-123. PCIe Extended Capability Registers

Offset	Acronym	Section
100h	PCIE_EXTCAP	Section 17.4.9.1
104h	PCIE_UNCERR	Section 17.4.9.2
108h	PCIE_UNCERR_MASK	Section 17.4.9.3
10Ch	PCIE_UNCERR_SVRTY	Section 17.4.9.4
110h	PCIE_CERR	Section 17.4.9.5
114h	PCIE_CERR_MASK	Section 17.4.9.6
118h	PCIE_ACCR	Section 17.4.9.7
11Ch	HDR_LOG0	Section 17.4.9.8
120h	HDR_LOG1	Section 17.4.9.9
124h	HDR_LOG2	Section 17.4.9.10
128h	HDR_LOG3	Section 17.4.9.11
12Ch	RC_ERR_CMD	Section 17.4.9.12
130h	RC_ERR_ST	Section 17.4.9.13
134h	ERR_SRC_ID	Section 17.4.9.14

17.4.9.1 PCIE_EXTCAP Register

Figure 17-116. PCIE_EXTCAP Register

31	20	19	16
NEXT_CAP		EXT_CAP_VER	
R-0		R-1	

15	0
EXT_CAP_ID	
R-1	

LEGEND: R = Read only; -*n* = value after reset

Table 17-124. PCIE_EXTCAP Register Field Descriptions

Bit	Field	Value	Description
31-20	NEXT_CAP	0	Next Capability Offset
19-16	EXT_CAP_VER	0-Fh	Extended Capability Version
15-0	EXT_CAP_ID	0-FFFFh	PCIe Extended Capability ID

17.4.9.2 PCIE_UNCERR Register

Figure 17-117. PCIE_UNCERR Register

31	24
Reserved	
R-0	

23	21	20	19	18	17	16
Reserved		UR_ERR_ST	ECRC_ERR_ST	MTLP_ERR_ST	RCVR_OF_ST	UCMP_ST
R-0		R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

15	14	13	12	11	8
CMPL_ABRT_ST	CMPL_TMOT_ST	FCP_ERR_ST	PSND_TLP_ST	Reserved	
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R-0	

7	6	5	4	3	0
Reserved		SRPS_DN_ST	DLP_ERR_ST	Reserved	
R-0		R-0	R/W1C-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-125. PCIE_UNCERR Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	UR_ERR_ST	0	Unsupported Request Error Status
19	ECRC_ERR_ST	0	ECRC Error Status
18	MTLP_ERR_ST	0	Malformed TLP Status
17	RCVR_OF_ST	0	Receiver Overflow Status
16	UCMP_ST	0	Unexpected Completion Status
15	CMPL_ABRT_ST	0	Completer Abort Status
14	CMPL_TMOT_ST	0	Completion Timeout Status
13	FCP_ERR_ST	0	Flow Control Protocol Error Status
12	PSND_TLP_ST	0	Poisoned TLP Status
11-6	Reserved	0	Reserved
5	SRPS_DN_ST	0	Surprise Down Error Status (not supported)
4	DLP_ERR_ST	0	Data Link Protocol Error Status
3-0	Reserved	0	Reserved

17.4.9.3 PCIE_UNCERR_MASK Register

Figure 17-118. PCIE_UNCERR_MASK Register

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved			UR_ERR_MSK	ECRC_ERR_MSK	MTLP_ERR_MSK	RCVR_OF_MSK	UCMP_MSK
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8
CMPL_ABRT_MSK	CMPL_TMOT_MSK	FCP_ERR_MSK	PSND_TLP_MSK	Reserved			
R/W-0	R/W-0	R/W-0	R/W-0	R-0			

7	6	5	4	3	2	1	0
Reserved		SRPS_DN_MSK	DLP_ERR_MSK	Reserved			
R-0		R-0	R/W-0	R-0			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-126. PCIE_UNCERR_MASK Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	UR_ERR_MSK	0	Unsupported Request Error Mask
19	ECRC_ERR_MSK	0	ECRC Error Mask
18	MTLP_ERR_MSK	0	Malformed TLP Mask
17	RCVR_OF_MSK	0	Receiver Overflow Mask
16	UCMP_MSK	0	Unexpected Completion Mask
15	CMPL_ABRT_MSK	0	Completer Abort Mask
14	CMPL_TMOT_MSK	0	Completion Timeout Mask
13	FCP_ERR_MSK	0	Flow Control Protocol Error Mask
12	PSND_TLP_MSK	0	Poisoned TLP Mask
11-6	Reserved	0	Reserved
5	SRPS_DN_MSK	0	Surprise Down Error Mask (not supported)
4	DLP_ERR_MSK	0	Data Link Protocol Error Mask
3-0	Reserved	0	Reserved

17.4.9.4 PCIE_UNCERR_SVRTY Register

Figure 17-119. PCIE_UNCERR_SVRTY Register

31	24
Reserved	
R-0	

23	21	20	19	18	17	16
Reserved		UR_ERR_SVRTY	ECRC_ERR_SVRTY	MTLP_ERR_SVRTY	RCVR_OF_SVRTY	UCMP_SVRTY
R-0		R/W-0	R/W-0	R/W-1	R/W-1	R/W-0

15	14	13	12	11	8
CMPL_ABRT_SVRTY	CMPL_TMOT_SVRTY	FCP_ERR_SVRTY	PSND_TLP_SVRTY	Reserved	
R/W-0	R/W-0	R/W-1	R/W-0	R-0	

7	6	5	4	3	0
Reserved		SRPS_DN_SVRTY	DLP_ERR_SVRTY	Reserved	
R-0		R-1	R/W-1	R-0	

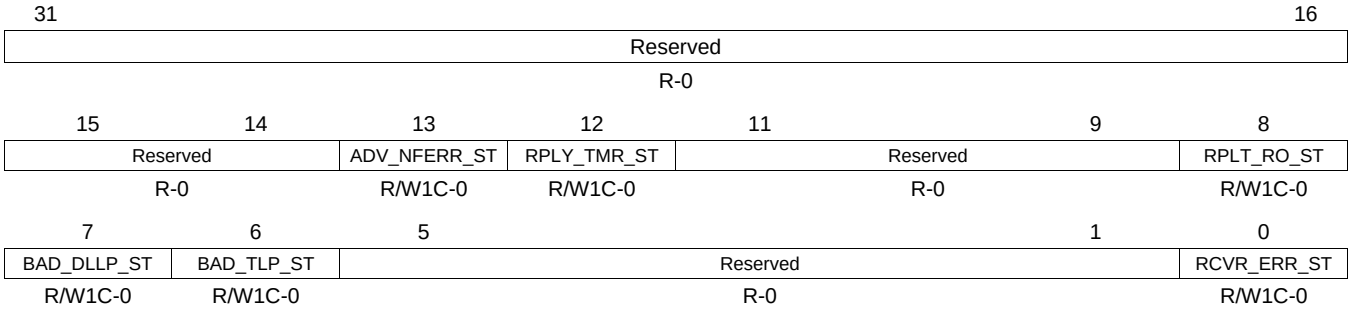
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-127. PCIE_UNCERR_SVRTY Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	UR_ERR_SVRTY	0	Unsupported Request Error Severity
19	ECRC_ERR_SVRTY	0	ECRC Error Severity
18	MTLP_ERR_SVRTY	1	Malformed TLP Severity
17	RCVR_OF_SVRTY	1	Receiver Overflow Severity
16	UCMP_SVRTY	0	Unexpected Completion Severity
15	CMPL_ABRT_SVRTY	0	Completer Abort Severity
14	CMPL_TMOT_SVRTY	0	Completion Timeout Severity
13	FCP_ERR_SVRTY	1	Flow Control Protocol Error Severity
12	PSND_TLP_SVRTY	0	Poisoned TLP Severity
11-6	Reserved	0	Reserved
5	SRPS_DN_SVRTY	1	Surprise Down Error Severity (not supported)
4	DLP_ERR_SVRTY	1	Data Link Protocol Error Severity
3-0	Reserved	0	Reserved

17.4.9.5 PCIE_CERR Register

Figure 17-120. PCIE_CERR Register



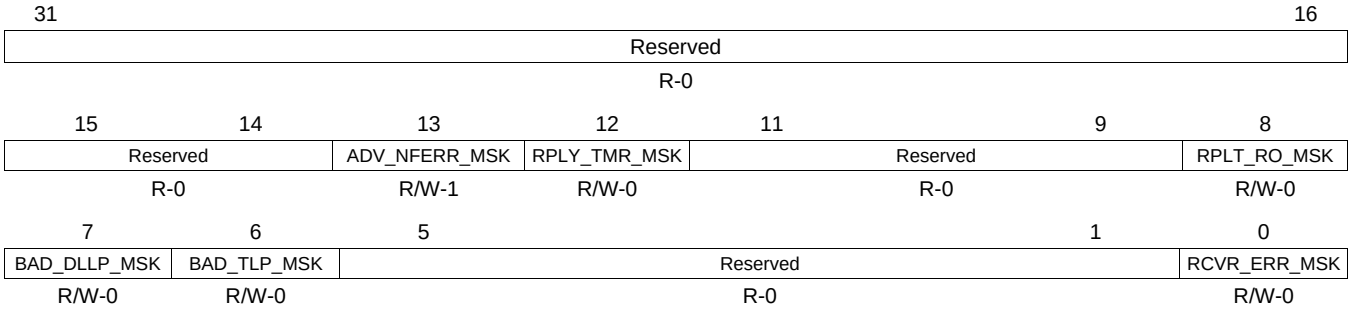
LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-128. PCIE_CERR Register Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13	ADV_NFERR_ST	0	Advisory Non-fatal Error Mask
12	RPLY_TMR_ST	0	Reply Timer Timeout Mask
11-9	Reserved	0	Reserved
8	RPLT_RO_ST	0	REPLAY_NUM Rollover Mask
7	BAD_DLLP_ST	0	Bad DLLP Mask
6	BAD_TLP_ST	0	Bad TLP Mask
5-1	Reserved	0	Reserved
0	RCVR_ERR_ST	0	Receiver Error Mask

17.4.9.6 PCIE_CERR_MASK Register

Figure 17-121. PCIE_CERR_MASK Register



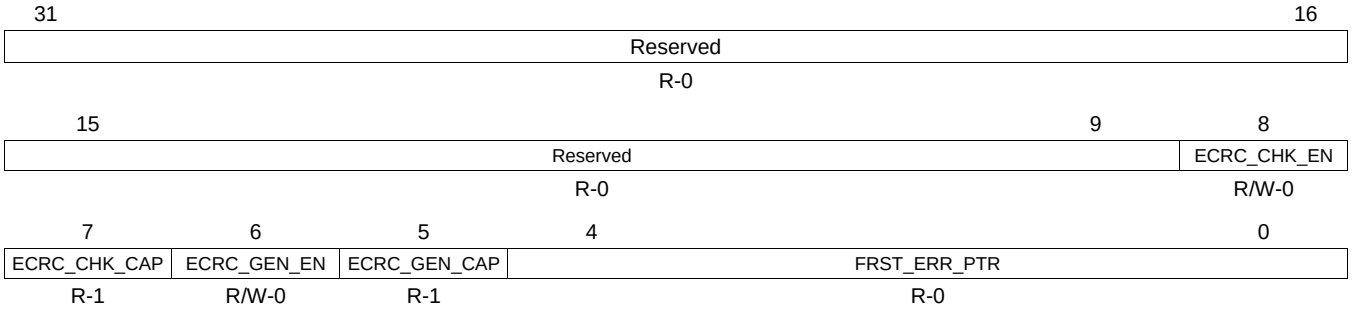
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-129. PCIE_CERR_MASK Register Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13	ADV_NFERR_MSK	1	Advisory Non-fatal Error Mask
12	RPLY_TMR_MSK	0	Reply Timer Timeout Mask
11-9	Reserved	0	Reserved
8	RPLT_RO_MSK	0	REPLAY_NUM Rollover Mask
7	BAD_DLLP_MSK	0	Bad DLLP Mask
6	BAD_TLP_MSK	0	Bad TLP Mask
5-1	Reserved	0	Reserved
0	RCVR_ERR_MSK	0	Receiver Error Mask

17.4.9.7 PCIE_ACCR Register

Figure 17-122. PCIE_ACCR Register



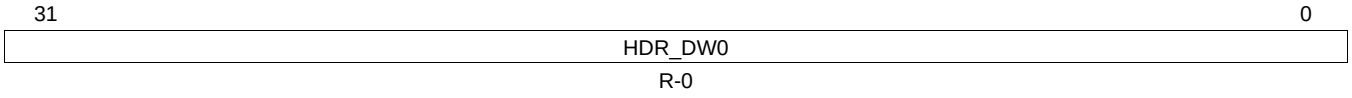
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-130. PCIE_ACCR Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	ECRC_CHK_EN	0	ECRC Check Enable
7	ECRC_CHK_CAP	1	ECRC Check Capable
6	ECRC_GEN_EN	0	ECRC Generation Enable
5	ECRC_GEN_CAP	1	ECRC Generation Capable
4-0	FRST_ERR_PTR	0	First Error Pointer

17.4.9.8 HDR_LOG0 Register

Figure 17-123. HDR_LOG0 Register



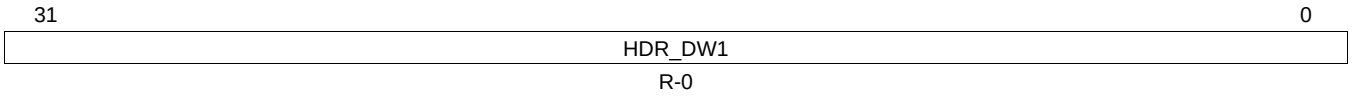
LEGEND: R = Read only; -*n* = value after reset

Table 17-131. HDR_LOG0 Register Field Descriptions

Bit	Field	Value	Description
31-0	HDR_DW0	0-FFFF FFFFh	First DWORD of Header for a detected error

17.4.9.9 HDR_LOG1 Register

Figure 17-124. HDR_LOG1 Register



LEGEND: R = Read only; -*n* = value after reset

Table 17-132. HDR_LOG1 Register Field Descriptions

Bit	Field	Value	Description
31-0	HDR_DW1	0-FFFF FFFFh	Second DWORD of Header for a detected error

17.4.9.10 HDR_LOG2 Register

Figure 17-125. HDR_LOG2 Register

31	0
HDR_DW2	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-133. HDR_LOG2 Register Field Descriptions

Bit	Field	Value	Description
31-0	HDR_DW2	0-FFFF FFFFh	Third DWORD of Header for a detected error

17.4.9.11 HDR_LOG3 Register

Figure 17-126. HDR_LOG3 Register

31	0
HDR_DW3	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-134. HDR_LOG3 Register Field Descriptions

Bit	Field	Value	Description
31-0	HDR_DW3	0-FFFF FFFFh	Fourth DWORD of Header for a detected error

17.4.9.12 RC_ERR_CMD Register

Figure 17-127. RC_ERR_CMD Register

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		FERR_RPT_EN	NFERR_RPT_EN	CERR_RPT_EN
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-135. RC_ERR_CMD Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	FERR_RPT_EN	0	Fatal Error Reporting Enable
1	NFERR_RPT_EN	0	Nonfatal Error Reporting Enable
0	CERR_RPT_EN	0	Correctable Error Reporting Enable

17.4.9.13 RC_ERR_ST Register

Figure 17-128. RC_ERR_ST Register

31	27	26	16
AER_INT_MSG		Reserved	
R-0		R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved	FERR_RCV	NFERR	UNCOR_FATAL	MULT_FNF	ERR_FNF	MULT_COR	CORR_ERR
R-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-136. RC_ERR_ST Register Field Descriptions

Bit	Field	Value	Description
31-27	AER_INT_MSG	0-1Fh	AER Interrupt Message Number. Writable through internal bus interface
26-7	Reserved	0	Reserved
6	FERR_RCV	0	Fatal Error Messages Received
5	NFERR	0	0h Non-Fatal Error Messages Received
4	UNCOR_FATAL	0	First Uncorrectable Fatal
3	MULT_FNF	0	Multiple ERR_FATAL/NONFATAL Received
2	ERR_FNF	0	ERR_FATAL/NONFATAL Received
1	MULT_COR	0	Multiple ERR_COR Received
0	CORR_ERR	0	ERR_COR Received

17.4.9.14 ERR_SRC_ID Register

Figure 17-129. ERR_SRC_ID Register

31	16
FNF_SRC_ID	
R-0	

15	0
CORR_SRC_ID	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-137. ERR_SRC_ID Register Field Descriptions

Bit	Field	Value	Description
31-16	FNF_SRC_ID	0-FFFFh	Error Fatal or Non-Fatal source identification
15-0	CORR_SRC_ID	0-FFFFh	Error Correctable source identification

17.4.10 Message Signaled Interrupts Registers

Table 17-138. Message Signaled Interrupts Registers

Offset	Acronym	Section
0h	MSI_CAP	Section 17.4.10.1
4h	MSI_LOW32	Section 17.4.10.2
8h	MSI_UP32	Section 17.4.10.3
Ch	MSI_DATA	Section 17.4.10.4

17.4.10.1 MSI_CAP Register

Figure 17-130. MSI_CAP Register

31	24	23	22	20	19	17	16
Reserved		64BIT_EN	MULT_MSG_EN		MULT_MSG_CAP		MSI_EN
R-0		R-1	R/W-0		R-0		R/W-0

15	8	7	0
NEXT_CAP		CAP_ID	
R-70h		R-5h	

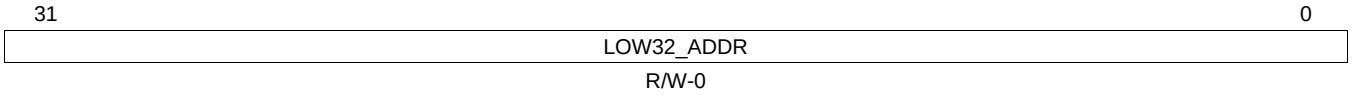
LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 17-139. MSI_CAP Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23	64BIT_EN	0	64-Bit address enabled. Writable from internal bus interface.
22-20	MULT_MSG_EN	0-7h	Multiple Message Enabled. Indicates that multiple message mode is enabled by software. Number of messages enabled must not be greater than Multiple Message Capable value.
19-17	MULT_MSG_CAP	0-7h	Multiple Message Capable. Writable from internal bus interface.
16	MSI_EN	0	MSI Enabled. When set, INTx must be disabled.
15-8	NEXT_CAP	0-FFh	Next Capability Pointer containing the offset to next capability structure. Writable from internal bus interface.
7-0	CAP_ID	0-FFh	MSI Capability ID

17.4.10.2 MSI_LOW32 Register

Figure 17-131. MSI_LOW32 Register



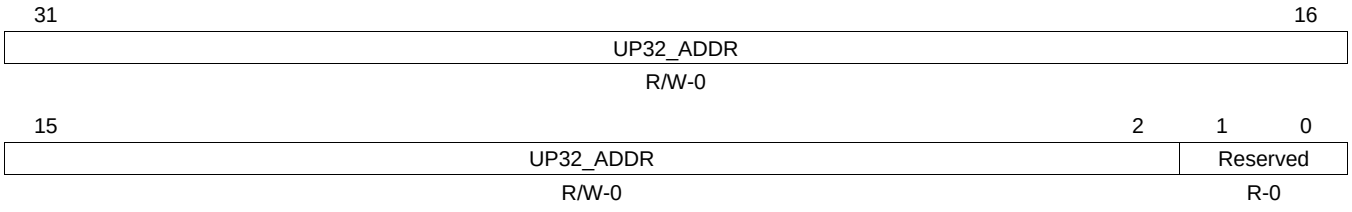
LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-140. MSI_LOW32 Register Field Descriptions

Bit	Field	Value	Description
31-0	LOW32_ADDR	0-FFFF FFFFh	Lower 32 bit address

17.4.10.3 MSI_UP32 Register

Figure 17-132. MSI_UP32 Register



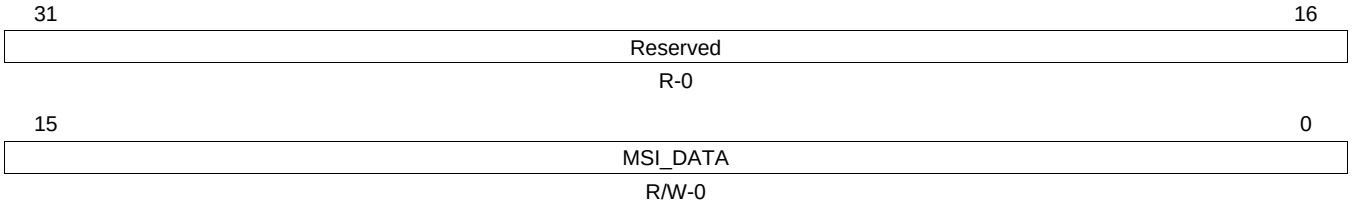
LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-141. MSI_UP32 Register Field Descriptions

Bit	Field	Value	Description
31-2	UP32_ADDR	0-3FFF FFFFh	Upper 32 bit address
1-0	Reserved	0	Reserved

17.4.10.4 MSI_DATA Register

Figure 17-133. MSI_DATA Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-142. MSI_DATA Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	MSI_DATA	0-FFFFh	MSI Data

17.4.11 Power Management Capability Registers

Table 17-143. Power Management Capability Registers

Offset	Acronym	Section
0h	PMCAP	Section 17.4.11.1
4h	PM_CTL_STAT	Section 17.4.11.2

17.4.11.1 PMCAP Register

Figure 17-134. PMCAP Register

31	30	29	28	27	26	25	24
PME_SUPP_N					D2_SUPP_N	D1_SUPP_N	AUX_CURR_N
R-0					R-0	R-0	R-0

23	22	21	20	19	18	17	16
AUX_CURR_N		DSI_N	Reserved	PME_CLK	PME_SPEC_VER		
R-0		R-0	R-0	R-0	R-3h		

15	14	13	12	11	10	9	8
PM_NEXT_PTR							
R-50h							

7	6	5	4	3	2	1	0
PM_CAP_ID							
R-1							

LEGEND: R = Read only; -*n* = value after reset

Table 17-144. PMCAP Register Field Descriptions

Bit	Field	Value	Description
31-27	PME_SUPP_N	0-1Fh	PME Support. Writable from internal bus interface.
26	D2_SUPP_N	0	D2 Support. Writable from internal bus interface.
25	D1_SUPP_N	0	D1 Support. Writable from internal bus interface.
24-22	AUX_CURR_N	0-7h	Auxiliary Current. Writable from internal bus interface.
21	DSI_N	0	Device Specific Initialization. Writable from internal bus interface.
20	Reserved	0	Reserved
19	PME_CLK	0	PME Clock. Hardwired to Zero.
18-16	PME_SPEC_VER	0-7h	Power Management Specification Version. Writable from internal bus interface.
15-8	PM_NEXT_PTR	0-FFh	Next Capability Pointer. Writable from internal bus interface.
7-0	PM_CAP_ID	0-FFh	Power Management Capability ID.

17.4.11.2 PM_CTL_STAT Register

Figure 17-135. PM_CTL_STAT Register

31	24
DATA_REG	
R-0	

23	22	21	16
CLK_CTRL_EN	B2_B3_SUPPORT	Reserved	
R-0	R-0	R-0	

15	14	13	12	9	8
PME_STATUS	DATA_SCALE		DATA_SELECT		PME_EN
R/W1C-0	R-0		R-0		R/W-0

7	4	3	2	1	0
Reserved		NO_SOFT_RST	Reserved	PWR_STATE	
R-0		R-0	R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-145. PM_CTL-STAT Register Field Descriptions

Bit	Field	Value	Description
31-24	DATA_REG	0-FFh	Data register for additional information. Not supported.
23	CLK_CTRL_EN	0	Bus Power/Clock Control Enable. Hardwired to zero.
22	BW_B3_SUPPORT	0	B2 and B3 support. Hardwired to zero.
21-16	Reserved	0	Reserved
15	PME_STATUS	0	PME Status. Indicates if a previously enabled PME event occurred or not.
14-13	DATA_SCALE	0-3h	Data Scale. Not supported.
12-9	DATA_SELECT	0-Fh	Data select. Not supported.
8	PME_EN	0	PME Enable. Value of 1 indicates device is enabled to generate PME. Writable from internal bus interface.
7-4	Reserved	0	Reserved
3	NO_SOFT_RST	0	No soft reset. It is set to disable reset during a transition from D3 to D0. Writable from internal bus interface.
2	Reserved	0	Reserved
1-0	PWR_STATE	0-3h	Power State. Controls the device power state. Writes are ignored if the state is not supported. Writable from internal bus interface.
		0	D0 power state
		1h	D1 power state
		2h	D2 power state
		3h	D3 power state

17.4.12 Port Logic Registers

Table 17-146. Port Logic Registers

Offset	Acronym	Section
700h	PL_ACKTIMER	Section 17.4.12.1
704h	PL_OMSG	Section 17.4.12.2
708h	PL_FORCE_LINK	Section 17.4.12.3
70Ch	ACK_FREQ	Section 17.4.12.4
710h	PL_LINK_CTRL	Section 17.4.12.5
714h	LANE_SKEW	Section 17.4.12.6
718h	SYM_NUM	Section 17.4.12.7
71Ch	SYMTIMER_FLTMASK	Section 17.4.12.8
720h	FLT_MASK2	Section 17.4.12.9
728h	DEBUG0	Section 17.4.12.10
72Ch	DEBUG1	Section 17.4.12.11
80Ch	PL_GEN2	Section 17.4.12.12

17.4.12.1 PL_ACKTIMER Register

Figure 17-136. PL_ACKTIMER Register

31	16
RPLY_LIMT	
R/W1S-C0h	

15	0
RND_TRP_LMT	
R/W1S-40h	

LEGEND: R/W = Read/Write; W1S = Write 1 to set; -*n* = value after reset

Table 17-147. PL_ACKTIMER Register Field Descriptions

Bit	Field	Value	Description
31-16	RPLY_LIMT	0-FFFFh	Replay Time Limit
15-0	RND_TRP_LMT	0-FFFFh	Round Trip Latency Time Limit

17.4.12.2 PL_OMSG Register

Figure 17-137. PL_OMSG Register

31	0
OMSG	
R/W-FFFF FFFFh	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-148. PL_OMSG Register Field Descriptions

Bit	Field	Value	Description
31-0	OMSG	0-FFFF FFFFh	Other Message Register. It can be used to send a specific PCI Express message in which case this register is programmed with the payload and bit 0 of Port Link Control Register is set to transmit the message.

17.4.12.3 PL_FORCE_LINK Register

Figure 17-138. PL_FORCE_LINK Register

31	24	23	22	21	16
LPE_CNT		Reserved		LNK_STATE	
R/W-7h		R-0		R/W-0	

15	14	8	7	0
FORCE_LINK	Reserved		LINK_NUM	
W1S-0	R-0		R/W-4h	

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set; -*n* = value after reset

Table 17-149. PL_FORCE_LINK Register Field Descriptions

Bit	Field	Value	Description
31-24	LPE_CNT	0-FFh	Low Power Entrance Count
23-22	Reserved	0	Reserved
21-16	LNK_STATE	0-3Fh	Link State
15	FORCE_LINK	0	Force Link
14-8	Reserved	0	Reserved
7-0	LINK_NUM	0-FFh	Link Number

17.4.12.4 ACK_FREQ Register

Figure 17-139. ACK_FREQ Register

31	30	29	27	26	24	23	16
Rsvd	ASPM_L1	L1_ENTRY_LATENCY		L0S_ENTRY_LATENCY		COMM_NFTS	
R-0	R/W-0	R/W-3h		R/W-3h		R/W-Fh	

15	8	7	0
NFTS		ACK_FREQ	
R/W-64h		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-150. ACK_FREQ Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30	ASPM_L1	0	Set to allow entering ASPM L1 even when link partner did not to L0s. When cleared, the ASPM L1 state is entered only after idle period during which both RX and TX are in L0s.
29-27	L1_ENTRY_LATENCY	0-7h	L1 Entrance Latency. The latency is set to $2^{L1\_ENTRY\_LATENCY}$ microseconds with the max being 64 microseconds.
26-24	L0S_ENTRY_LATENCY	0-7h	L0s Entrance Latency. The latency is set to L0S_ENTRY_LATENCY + 1 microseconds. Maximum is 7 microseconds.
23-16	COMM_NFTS	0-FFh	Number of Fast Training Sequences when common clock is used and when transitioning from L0s to L0.
15-8	NFTS	0-FFh	Number of Fast Training Sequences to be transmitted when transitioning from L0s to L0. Value of zero is not supported.
7-0	ACK_FREQ	0-FFh	Ack Frequency. Default is to wait until 255 Ack DLLPs are pending before it is sent.

17.4.12.5 PL_LINK_CTRL Register

Figure 17-140. PL_LINK_CTRL Register

31	22	21	16
Reserved		LNK_MODE	
R-0		R/W-3h	

15	12	11	8
Reserved		LNK_RATE	
R-0		R/W-1	

7	6	5	4	3	2	1	0
FLNK_MODE	Reserved	DLL_EN	Reserved	RST_ASRT	LPBK_EN	SCRM_DIS	OMSG_REQ
R/W-0	R-0	R/W-1	R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-151. PL_LINK_CTRL Register Field Descriptions

Bit	Field	Value	Description
31-22	Reserved	0	Reserved
21-16	LNK_MODE	0-3Fh	Link Mode Enable.
		0	Reserved
		1h	×1
		2h	Reserved
		3h	×2
		4h-6h	Reserved
		7h	×4
		8h-Eh	Reserved
		Fh	×8
		10h-1Eh	Reserved
		1Fh	×16
		20h-3Eh	Reserved
		3Fh	×32
15-12	Reserved	0	Reserved
11-8	LNK_RATE	0-Fh	Default Link Rate. For 2.5, it is 1h. This register does not affect any functionality.
7	FLNK_MODE	0	Fast Link Mode
6	Reserved	0	Reserved
5	DLL_EN	1	DLL Link Enable
4	Reserved	0	Reserved
3	RST_ASRT	0	Reset Assert
2	LPBK_EN	0	Loopback Enable
1	SCRM_DIS	0	Scramble Disable
0	OMSG_REQ	0	Other Message Request

17.4.12.6 LANE_SKEW Register

Figure 17-141. LANE_SKEW Register

31	30	26	25	24
L2L_DESKEW	Reserved		ACK_DISABLE	FC_DISABLE
R/W-0	R-0		R/W-0	R/W-0

23	0
LANE_SKEW	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-152. LANE_SKEW Register Field Descriptions

Bit	Field	Value	Description
31	L2L_DESKEW	0	Disable Lane to Lane deskew
30-26	Reserved	0	Reserved
25	ACK_DISABLE	0	Disable Ack and Nak DLLP transmission.
24	FC_DISABLE	0	Flow control disable. Set to disable transmission of Flow Control DLLPs.
23-0	LANE_SKEW	0-FF FFFFh	Insert Lane Skew for Transmit. The value is in units of one symbol time. Thus a value 02h will force a skew of two symbol times for that lane. Max allowed is five symbol times. This 24 bit field is used for programming skew for eight lanes with three bits per lane.

17.4.12.7 SYM_NUM Register

Figure 17-142. SYM_NUM Register

31	29	28	24	23	19	18	16
MAX_FUNC		FCWATCH_TIMER		ACK_LATENCY_TIMER		REPLAY_TIMER	
R/W-0		R/W-0		R/W-0		R/W-4h	

15	14	13	11	10	8	7	4	3	0
REPLAY_TIMER		Reserved		SKP_COUNT		NUM_TS2_SYMBOLS		TS_COUNT	
R/W-4h		R-0		R/W-3h		R/W-Ah		R/W-Ah	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-153. SYM_NUM Register Field Descriptions

Bit	Field	Value	Description
31-29	MAX_FUNC	0-7h	Configuration requests at function numbers above this value will result in UR response.
28-24	FCWATCH_TIMER	0-1Fh	Timer Modifier for Flow Control Watchdog Timer. Increases the timer value for Flow control watchdog timer in increments of 16 clock cycles.
23-19	ACK_LATENCY_TIMER	0-1Fh	Timer Modifier for Ack/Nak latency timer in increments of 64 clock periods.
18-14	REPLAY_TIMER	0-1Fh	Timer for replaying TLPs in increments of 64 clock cycles.
13-11	Reserved	0	Reserved
10-8	SKP_COUNT	0-7h	Number of SKP symbols
7-4	NUM_TS2_SYMBOLS	0-Fh	Number of TS2 symbols. This field does not affect any functionality.
3-0	TS_COUNT	0-Fh	Number of TS symbols. Set the number of TS identifier symbols that are sent in TS1 and TS2 ordered sets.

17.4.12.8 SYMTIMER_FLTMASK Register

Figure 17-143. SYMTIMER_FLTMASK Register

31	30	29	28	27	26	25	24
F1_CFG_DROP	F1_IO_DROP	F1_MSG_DROP	F1_CPL_ECRC_DROP	F1_ECRC_DROP	F1_CPL_LEN_TEST	F1_CPL_ATTR_TEST	F1_CPL_TC_TEST
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

23	22	21	20	19	18	17	16
F1_CPL_FUNC_TEST	F1_CPL_REQID_TEST	F1_CPL_TAGERR_TEST	F1_LOCKED_RD_AS_UR	F1_CFG1_RE_AS_US	F1_UR_OUT_OF_BAR	F1_UR_POISON	F1_UR_FUN_MISMATCH
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	11	10	0
FC_WDOG_DISABLE	Reserved		SKP_VALUE	
R/W-0	R-0		R/W-500h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-154. SYMTIMER_FLTMASK Register Field Descriptions

Bit	Field	Value	Description
31	F1_CFG_DROP	0	Set to allow CFG TLPs on RC
30	F1_IO_DROP	0	Set to allow IO TLPs on RC
29	F1_MSG_DROP	0	Set to allow MSG TLPs on RC
28	F1_CPL_ECRC_DROP	0	Set to allow Completion TLPs with ECRC to pass up
27	F1_ECRC_DROP	0	Set to allow TLPs with ECRC to pass up
26	F1_CPL_LEN_TEST	0	Set to mask length match for received completion TLPs
25	F1_CPL_ATTR_TEST	0	Set to mask attribute match on received completion TLPs
24	F1_CPL_TC_TEST	0	Set to mask traffic class match on received completion TLPs
23	F1_CPL_FUNC_TEST	0	Set to mask function match for received completion TLPs
22	F1_CPL_REQID_TEST	0	Set to mask request ID match for received completion TLPs
21	F1_CPL_TAGERR_TEST	0	Set to mask tag error rules for received completion TLPs
20	F1_LOCKED_RD_AS_UR	0	Set to treat locked read TLPs as supported for EP, UR for RC.
19	F1_CFG1_RE_AS_US	0	Set to treat type 1 CFG TLPs as supported for EP and UR for RC
18	F1_UR_OUT_OF_BAR	0	Set to treat out-of-BAR TLPs as supported requests
17	F1_UR_POISON	0	Set to treat poisoned TLPs as supported requests
16	F1_UR_FUN_MISMATCH	0	Set to treat mismatched TLPs as supported
15	FC_WDOG_DISABLE	0	Disable FC Watchdog Timer
14-11	Reserved	0	Reserved
10-0	SKP_VALUE	0-7FFh	Number of symbol times to wait between transmitting SKP ordered sets. For example, for a setting of 1536 decimal, the wait will be for 1537 symbol times.

17.4.12.9 FLT_MASK2 Register

Figure 17-144. FLT_MASK2 Register

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		FLUSH_REQ	DLLP_ABORT	VMSG1_DROP	VMSG0_DROP
R-0		R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-155. FLT_MASK2 Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	FLUSH_REQ	0	Set to enable the filter to handle flush request
2	DLLP_ABORT	0	Set to disable DLLP Abort for unexpected CPL
1	VMSG1_DROP	0	Set to disable dropping of Vendor MSG Type 1. When cleared, Vendor MSG Type 1 will be passed to internal bus interface
0	VMSG0_DROP	0	Set to disable dropping of Vendor MSG Type 0 with UR reporting. When cleared, Vendor MSG Type 0 will be passed to internal bus interface

17.4.12.10 DEBUG0 Register

Figure 17-145. DEBUG0 Register

31	28	27	26	25	24
TS_LINK_CTRL		TS_LANE_K237	TS_LINK_K237	RCVD_IDLE0	RCVD_IDLE1
R-0		R-0	R-0	R-0	R-0

23	8
PIPE_TXDATA	
R-0	

7	6	5	4	0
PIPE_TXDATAK		TXB_SKIP_TX	LTSSM_STATE	
R-0		R-0	R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-156. DEBUG0 Register Field Descriptions

Bit	Field	Value	Description
31-28	TS_LINK_CTRL	0-Fh	Link control bits advertised by link partner
27	TX_LANE_K237	0	Currently receiving k237 (PAD) in place of lane number
26	TS_LINK_K237	0	Currently receiving k237 (PAD) in place of link number
25	RCVD_IDLE0	0	Receiver is receiving logical idle
24	RCVD_IDLE1	0	2nd symbol is also idle (16bit PHY interface only)
23-8	PIPE_TXDATA	0-FFFFh	PIPE Transmit data. Reset value is zero but changes at every clock after that.
7-6	PIPE_TXDATAK	0-3h	PIPE transmit K indication
5	TXB_SKIP_TX	0	A skip ordered set has been transmitted
4-0	LTSSM_STATE	0-1Fh	LTSSM current state. A read value of 11h indicates the up status of the Link.

17.4.12.11 DEBUG1 Register

Figure 17-146. DEBUG1 Register

31	30	29	28	27	26 24
SCRAMBLER_ DISABLE	LINK_DISABLE	LINK_IN_ TRAINING	RCVR_REVRS_POL_EN	TRAINING_RST_N	Reserved
R-0	R-0	R-0	R-0	R-1	R-0

23	22	21	20	19	18 16
Rsvd	PIPE_TXDETECTRX_LB	PIPE_TXELECIDLE	PIPE_TXCOMPLIANCE	APP_INIT_RST	Reserved
R-0	R-0	R-1	R-0	R-0	R-0

15 8
RMLH_TS_LINK_NUM
R-0

7 5	4	3	2	1	0
Reserved	XMLH_LINK_UP	RMLH_INSKIP_RCV	RMLH_TS1_RCVD	RMLH_TS2_RCVD	RMLH_RCVD_LANE _REV
R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 17-157. DEBUG1 Register Field Descriptions

Bit	Field	Value	Description
31	SCRAMBLER_DISABLE	0	Scrambling disabled for the link
30	LINK_DISABLE	0	LTSSM in DISABLE state. Link inoperable
29	LINK_IN_TRAINING	0	LTSSM performing link training
28	RCVR_REVRS_POL_EN	0	LTSSM testing for polarity reversal
27	TRAINING_RST_N	1	LTSSM-negotiated link reset
26-23	Reserved	0	Reserved
22	PIPE_TXDETECTRX_LB	0	PIPE receiver detect/loopback request
21	PIPE_TXELECIDLE	1	PIPE transmit electrical idle request
20	PIPE_TXCOMPLIANCE	0	PIPE transmit compliance request
19	APP_INIT_RST	0	Application request to initiate training reset
18-16	Reserved	0	Reserved
15-8	RMLH_TS_LINK_NUM	0-FFh	Link number advertised/confirmed by link partner
7-5	Reserved	0	Reserved
4	XMLH_LINK_UP	0	LTSSM reports PHY link up
3	RMLH_INSKIP_RCV	0	Receiver reports skip reception
2	RMLH_TS1_RCVD	0	TS1 training sequence received (pulse)
1	RMLH_TS2_RCVD	0	TS2 training sequence received (pulse)
0	RMLH_RCVD_LANE_REV	0	Receiver detected lane reversal

17.4.12.12 PL_GEN2 Register

Figure 17-147. PL_GEN2 Register

31	21	20	19	18	17	16
Reserved		DEEMPH	CFG_TX_CMPL	CFG_TX_SWING	DIR_SPD	LN_EN
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	8	7	0
LN_EN		NUM_FTS	
R/W-2h		R/W-Fh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-158. PL_GEN2 Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	DEEMPH	0	Set Deemphasis level for upstream ports
19	CFG_TX_CMPL	0	Configure TX Compliance Receive Bit
18	CFG_TX_SWING	0	Configure PHY TX Swing
17	DIR_SPD	0	Directed Speed Change
16-8	LN_EN	0-1FFh	Lane Enable. It is 1 for ×1, 2 for ×2, and so on.
7-0	NUM_FTS	0-FFh	Number of fast training sequences.

Power, Reset, and Clock Management (PRCM) Module

Topic **Page**

18.1 Introduction

The device power-management architecture ensures maximum performance and operation time for user satisfaction (audio/video support) while offering versatile power-management techniques for maximum design flexibility, depending on application requirements. This introduction contains the following information:

- Power-management architecture building blocks for the device
- State-of-the-art power-management techniques supported by the power-management architecture of the device

18.1.1 Device Power-Management Architecture Building Blocks

To provide a versatile architecture supporting multiple power-management techniques, the power-management framework is built with three levels of resource management: clock, power, and voltage management.

These management levels are enforced by defining the managed entities or building blocks of the power-management architecture, called the clock, power, and voltage domains. A domain is a group of modules or subsections of the device that share a common entity (for example, common clock source, common voltage source, or common power switch). The group forming the domain is managed by a policy manager. For example, a clock for a clock domain is managed by a dedicated clock manager within the power, reset, and clock management (PRCM) module. The clock manager considers the joint clocking constraints of all the modules belonging to that clock domain (and, hence, receiving that clock).

18.1.1.1 Clock Management

The PRCM module manages the gating (that is, switching off) and enabling of the clocks to the device modules. The clocks are managed based on the requirement constraints of the associated modules. The following sections identify the module clock characteristics, management policy, clock domains, and clock domain management.

Each module within the device has specific clock input characteristic requirements. Based on the characteristics of the clocks delivered to the modules, the clocks are divided into two categories: interface clocks and functional clocks.

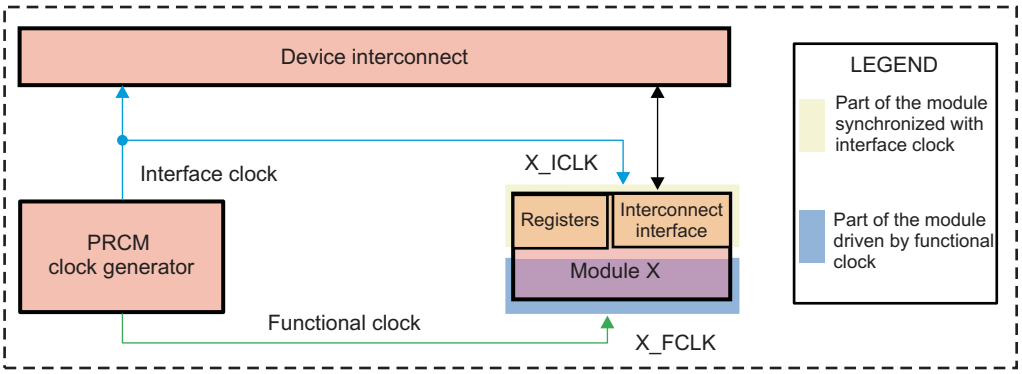
The interface clocks have the following characteristics:

- They ensure proper communication between any module/subsystem and the interconnect.
- In most cases, they supply the system interconnect interface and registers of the module.
- A typical module has one interface clock, but modules with multiple interface clocks may also exist (that is, when connected to multiple interconnect buses).
- Interface clock management is done at the device level.
- From the standpoint of the PRCM module, an interface clock is identified by an _ICLK suffix.

Functional clocks have the following characteristics:

- They supply the functional part of a module or subsystem.
- A module can have one or more functional clocks. Some functional clocks are mandatory, while others are optional. A module needs its mandatory clock(s) to be operational. The optional clocks are used for specific features and can be shut down without stopping the module activity.
- From the standpoint of the PRCM module, a functional clock is distributed directly to the related modules through a dedicated clock tree. It is identified with an _FCLK suffix.

Figure 18-1. Functional and Interface Clocks



18.1.2 Module-Level Clock Management

Each module in the device may also have specific clock requirements. Certain module clocks must be active when operating in specific modes, or may be gated otherwise. Globally, the activation and gating of the module clocks are managed by the PRCM module. Hence, the PRCM module must be aware of when to activate and when to gate the module clocks. The PRCM module differentiates the clock-management behavior for device modules based on whether the module can initiate transactions on the device interconnect (called master module) or cannot initiate transactions and only responds to the transactions initiated by the master (called slave module). Thus, two hardware-based power-management protocols are used:

- Master standby protocol: Clock-management protocol between the PRCM and master modules
- Slave idle protocol: Clock-management protocol between the PRCM and slave modules

18.1.2.1 Master Standby Protocol

This protocol is used to indicate that a master module must initiate a transaction on the device interconnect and requests specific (functional and interface) clocks for the purpose. The PRCM module ensures that the required clocks are active when the master module requests the PRCM module to enable them. This is called a module wake-up transition and the module is said to be functional after this transition completes. Similarly, when the master module no longer requires the clocks, it informs the PRCM module, which can then gate the clocks to the module. The master module is then said to be in standby mode. Although the protocol is completely hardware-controlled, software must configure the clock-management behavior for the module. This is done by setting the module register bit field <Module>_SYSCONFIG. MIDLEMODE or >Module>_SYSCONFIG. STANDBYMODE (see Table 18-1). The behavior, identified by standby mode values, must be configured.

The standby status of a master module is indicated by the CM_<Power_domain>_<Module>_CLKCTRL[x].STBYST bit in the PRCM module. Table 18-2 describes the master module standby status.

Table 18-1. Master Module Standby-Mode Settings

Standby Mode Value	Selected Mode	Description
0	Force-standby	The module unconditionally asserts the standby request to the PRCM module, regardless of its internal operations. The PRCM module may gate the functional and interface clocks to the module. This mode must be used carefully because it does not prevent the loss of data at the time the clocks are gated.
1h	No-standby	The module never asserts the standby request to the PRCM module. This mode is safe from a module point of view because it ensures that the clocks remain active. However, it is not efficient from a power-saving perspective because it never allows the output clocks of the PRCM module to be gated.
2h	Smart-standby	The module asserts the standby request based on its internal activity status.The standby signal is asserted only when all ongoing transactions are complete and the module is idled. The PRCM module can then gate the clocks to the module.
3h	Smart-standby wakeup-capable mode	The module asserts the standby request based on its internal activity status. The standby signal is asserted only when all ongoing transactions are complete and the module is idle. The PRCM module can then gate the clocks to the module. The module may generate (master-related) wake-up events when in STANDBY state. The mode is relevant only if the appropriate module mwakeup output is implemented.

Table 18-2. Master Module Standby Status

STBYST Bit Value	Description
0	The module is functional.
1	The module is in standby mode.

18.1.2.2 Slave Idle Protocol

This hardware protocol allows the PRCM module to control the state of a slave module. The PRCM module informs the slave module, through assertion of an idle request, when its clocks (interface and functional) can be gated. The slave can then acknowledge the request from the PRCM module and the PRCM module is then allowed to gate the clocks to the module. A slave module is said to be in IDLE state when its clocks are gated by the PRCM module. Similarly, an idled slave module may need to be wakened because of a service request from a master module or as a result of an event (called a wake-up event; for example, interrupt or DMA request) received by the slave module. In this situation the PRCM module enables the clocks to the module and then deasserts the idle request to signal the module to wake up. Although the protocol is completely hardware-controlled, software must configure the clock-management behavior for the slave module. This is done by setting the module register bit field <Module>_SYSCONFIG. SIDLEMODE or <Module>_SYSCONFIG. IDLEMODE (see Table 18-3). The behavior, listed in the Idle Mode Value column, must be configured by software.

Table 18-3. Module Idle Mode Settings

Idle Mode Value	Selected Mode	Description
0	Force-idle	The module unconditionally acknowledges the idle request from the PRCM module, regardless of its internal operations. This mode must be used carefully because it does not prevent the loss of data at the time the clock is switched off.
1h	No-idle	The module never acknowledges any idle request from the PRCM module. This mode is safe from a module point of view because it ensures that the clocks remain active. However, it is not efficient from a power-saving perspective because it does not allow the PRCM module output clock to be shut off, and thus the power domain to be set to a lower power state.
2h	Smart-idle	The module acknowledges the idle request basing its decision on its internal activity. Namely, the acknowledge signal is asserted only when all pending transactions, interrupts, or direct memory access (DMA) requests are processed. This is the best approach to efficient system power management.

Table 18-3. Module Idle Mode Settings (continued)

Idle Mode Value	Selected Mode	Description
3h	Smart-idle wakeup-capable mode	The module acknowledges the idle request basing its decision on its internal wakeup-capable mode activity. Namely, the acknowledge signal is asserted only when all pending transactions, interrupts, or DMA requests are processed. This is the best approach to efficient system power management. The module may generate (IRQ- or DMA-request-related) wake-up events when in IDLE state. The mode is relevant only if the appropriate module swakeup output(s) is implemented.

The idle status of a slave module is indicated by the CM_<Powerdomain>_<Module>_CLKCTRL[x] IDLEST bit field in the PRCM module. Table 18-4 lists the possible idle status for a slave module.

Table 18-4. Slave Module Idle Status

IDLEST Bit	Idle Status	Description
0	Functional	The module is fully functional. The interface and functional clocks are active.
1h	In transition	The module is performing a wake-up or a sleep transition.
2h	Interface idle	The module interface clock is idled. The module may remain functional if using a separate functional clock.
3h	Full idle	The module is fully idle. The interface and functional clocks are gated.

For the idle protocol management on the PRCM module side, the behavior of the PRCM module is configured in the CM_<Powerdomain>_<Module>_CLKCTRL[x] MODULEMODE bit field. Based on the configured behavior, the PRCM module asserts the idle request to the module unconditionally (that is, immediately when the software requests). Table 18-5 describes the configurable behavior of MODULEMODE.

Table 18-5. Slave Module Mode Settings in PRCM

MODULEMODE Bit	Selected Mode	Description
0	Disabled	The PRCM module unconditionally asserts the module idle request. This request applies to the gating of the functional and interface clocks to the module. If acknowledged by the module, the PRCM module can gate all clocks to the module (that is, the module is completely disabled).
1h	Reserved	Reserved
2h	Enabled	This mode applies to a module when the PRCM module manages its interface and functional clocks. The functional clock to the module remains active unconditionally, while the PRCM module automatically asserts/deasserts the module idle request based on the clock-domain transitions. If acknowledged by the module, the PRCM module can gate only the interface clock to the module.
3h	Reserved	Reserved

In addition to the IDLE and STANDBY protocol, PRCM offers also the possibility to manage optional clocks, through a direct software control: “OptFclken” bit from programming register.

Table 18-6. Module Clock Enabling Condition

Clock Enabling Condition	Condition:	
	AND	OR
Clock associated with STANDBY protocol	Clock domain is ready	MStandby is de-asserted Mwakeup is asserted

Table 18-6. Module Clock Enabling Condition (continued)

Clock Enabling Condition	Condition: AND	Condition: OR
Clock associated with IDLE protocol, as interface clock	Clock domain is ready	Idle status = FUNCT Idle status = TRANS SWakeup is asserted
Clock associated with IDLE protocol, as functional clock	Clock domain is ready	Idle status = FUNCT Idle status = TRANS Idle status = IDLE SWakeup is asserted
Optional clock	Clock domain is ready	OptFclken=Enabled (1)

18.1.3 Clock Domain

A clock domain is a group of modules fed by clock signals controlled by the same clock manager in the PRCM module (see Figure 18-2). By gating the clocks in a clock domain, the clocks to all the modules belonging to that clock domain can be cut to lower their active power consumption (that is, the device is on and the clocks to the modules are dynamically switched to ACTIVE or INACTIVE (GATED) states). Thus, a clock domain allows control of the dynamic power consumption of the device. The device is partitioned into multiple clock domains, and each clock domain is controlled by an associated clock manager within the PRCM module. This allows the PRCM module to individually activate and gate each clock domain of the device.

Figure 18-2. Generic Clock Domain

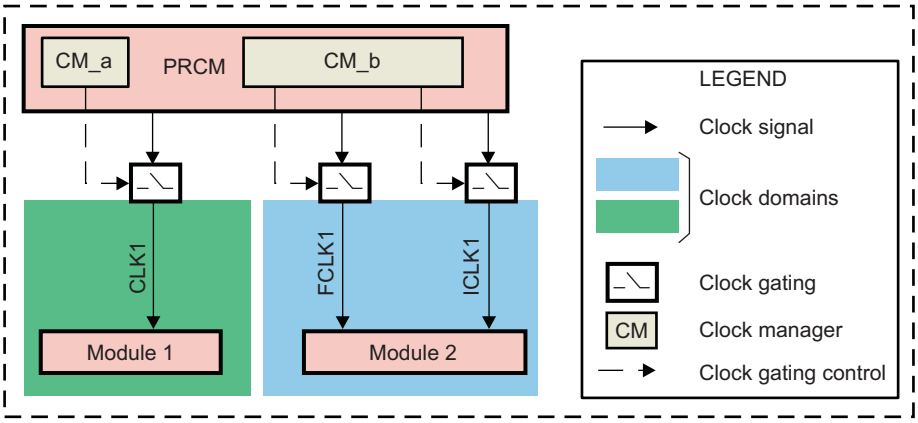


Figure 18-2 is an example of two clock managers: CM_a and CM_b. Each clock manager manages a clock domain. The clock domain of CM_b is composed of two clocks: a functional clock (FCLK2) and an interface clock (ICLK1), while the clock domain of CM_a consists of a clock (CLK1) that is used by the module as a functional and interface clock. The clocks to Module 2 can be gated independently of the clock to Module 1, thus ensuring power savings when Module 2 is not in use. The PRCM module lets software check the status of the clock domain functional clocks. The CM_<Clock_domain>_CLKSTCTRL[x] CLKACTIVITY_<FCLK/Clock name_FCLK> bit in the PRCM module identifies the state of the functional clock(s) within the clock domain. Table 18-7 shows the possible states of the functional clock.

Table 18-7. Clock Domain Functional Clock States

CLKACTIVITY Bit	Status	Description
0	Gated	The functional clock of the clock domain is inactive
1	Active	The functional clock of the clock domain is running

18.1.3.1 Clock Domain-Level Clock Management

The domain clock manager can automatically (that is, based on hardware conditions) and jointly manage the interface clocks within the clock domain. The functional clocks within the clock domain are managed through software settings. A clock domain can switch between three possible states: ACTIVE, IDLE_TRANSITION, and INACTIVE. Figure 18-3 shows the sleep and wake-up transitions of the clock domain between ACTIVE and INACTIVE states.

Figure 18-3. Clock Domain State Transitions

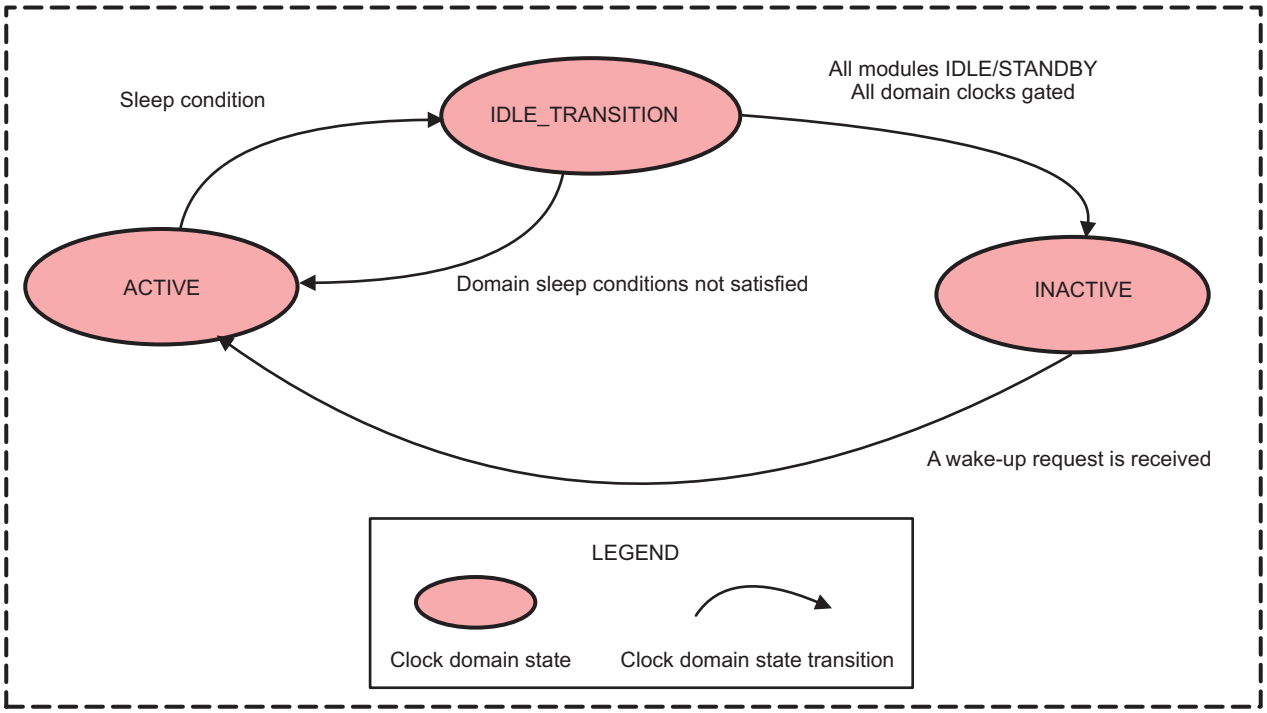


Table 18-8. Clock Domain States

State	Description
ACTIVE	Every nondisabled slave module (that is, those whose MODULEMODE value is not set to disabled) is put out of IDLE state. · All interface clocks to the nondisabled slave modules in the clock domain are provided. All functional and interface clocks to the active master modules (that is, not in STANDBY) in the clock domain are provided.· All enabled optional clocks to the modules in the clock domain are provided.
IDLE_TRANSITION	This is a transitory state.· Every master module in the clock domain is in STANDBY state · Every idle request to all the slave modules in the clock domain is asserted.· The functional clocks to the slave module in enabled state (that is, those whose MODULEMODE values are set to enabled) remain active.· All enabled optional clocks to the modules in the clock domain are provided.
INACTIVE	All clocks within the clock domain are gated.· Every slave module in the clock domain (that is, those whose MODULEMODE is set to disabled or auto) is in IDLE state and set to disabled or auto mode.· Every slave module in the clock domain (that is, those whose MODULEMODE is set to disabled or auto) is in IDLE state and set to disabled or auto mode.· Every optional functional clock in the clock domain is gated.

Each clock domain transition behavior is managed by an associated register bit field in the CM_<Clock domain>_CLKSTCTRL[x] CLKTRCTRL PRCM module. Table 18-9 describes the clock transition mode settings in the clock domain.

Table 18-9. Clock Transition Mode Settings

CLKTRCTRL Bit	Selected Mode	Description
0	Reserved	Reserved
1h	SW_SLEEP	A software-forced sleep transition. The transition is initiated when the associated hardware conditions are satisfied
2h	SW_WKUP	A software-forced clock domain wake-up transition is initiated
3h	Reserved	Reserved

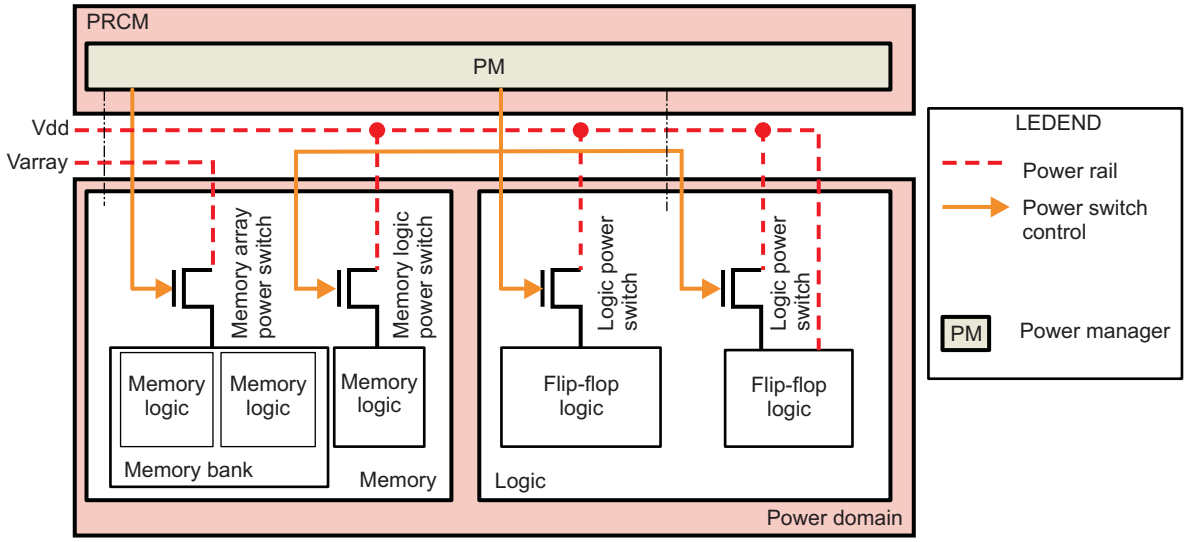
18.1.4 Power Management

The PRCM module manages the switching on and off of the power supply to the device modules. To minimize device power consumption, the power to the modules can be switched off when they are not in use. Independent power control of sections of the device allows the PRCM module to turn on and off specific sections of the device without affecting the others.

18.1.4.1 Power Domain

A power domain is a section (that is, a group of modules) of the device with an independent and dedicated power manager (see Figure 18-4). A power domain can be turned on and off without affecting the other parts of the device.

Figure 18-4. Power Domain Block Diagram



To minimize device power consumption, the modules are grouped into power domains. A power domain can be split into a logic area (Table 18-10) and a memory area (Table 18-11).

Table 18-10. States of a Logic Area in a Power Domain

State	Description
On	Logic is fully powered
Off	Logic power switches are off. All the logic (DFF) is lost.

Table 18-11. States of a Memory Area in a Power Domain

State	Description
On	The memory array is powered and fully functional.
Off	The memory array is powered down.

18.1.4.2 Power Domain Management

The power manager associated with each power domain is assigned the task of managing the domain power transitions. It ensures that all hardware conditions are satisfied before it can initiate a power domain transition from a source to a target power state.

Table 18-12. Power Domain Control and Status Registers

Register/Bit Field	Type	Description
PM_<Power domain>_PWRSTCTRL[1:0] POWERSTATE	Control	Selects the target power state of the power domain. It is OFF or ON.
PM_<Power domain>_PWRSTST[1:0] POWERSTATEST	Status	Identifies the current state of the power domain. It is OFF or ON.
PM_<Power domain>_PWRSTST[2] LOGICSTATEST	Status	Identifies the current state of the logic area in the power domain. It is OFF or ON.
PM_<Power domain>_PWRSTST[5:4] MEMSTATEST	Status	Identifies the current state of the memory area in the power domain. It is OFF or ON.

18.1.4.3 Power-Management Techniques

The following section describes the state-of-the-art power-management techniques supported by the device.

18.1.4.3.1 Adaptive Voltage Scaling

Adaptive voltage scaling (AVS) is a power-management technique based in Smart Reflex that is used for automatic control of the operating voltages of the device to reduce active power consumption. With Smart Reflex, power-supply voltage is adapted to silicon performance, either statically (based on performance points predefined in the manufacturing process of a given device) or dynamically (based on the temperature-induced real-time performance of the device). A comparison of these predefined performance points to the real-time on-chip measured performance determines whether to raise or lower the power-supply voltage. AVS achieves the optimal performance/power trade-off for all devices across the technology process spectrum and across temperature variation. The device voltage is automatically adapted to maintain performance of the device.

18.2 Power Reset Clock Management Overview

18.2.1 Introduction

The PRCM is structured using the architectural concepts presented in the 5000x Power Management Framework. This framework provides:

A set of modular, re-usable FSM blocks to be assembled into the full clock and power management mechanism. A register set and associated programming model. Functional sub-block definitions for clock management, power management, system clock source generation, and master clock generation.

The device supports an enhanced power management scheme based on:

- 4 functional power domains:

Generic domains:

- Active
- AlwaysOn
- Default
- HDVICP2-0
- HDVICP2-1
- HDVICP2-2
- SGX

The PRCM provides the following functional features:

- Device power-up sequence control
- Device sleep / wake-up sequence control
- Centralized reset generation and management
- Centralized clock generation and management

The PRCM modules implement these general functional interfaces:

- OCP configuration ports
- Power switch control signals
- Device control signals
- Clocks control signals
- Resets signals
- A set of power management protocol signals for each module to control and monitor standby, idle and wake-up modes (CM and PRM)
- Emulation signals

18.2.2 Interfaces Description

This section lists and shortly describes the different interfaces that allow PRCM to communicate with other modules or external devices.

18.2.2.1 OCP Interfaces

The PRCM has 1 target OCP interfaces, compliant with respect to the OCP/IP2 standard. The OCP port, for the PRCM module is used to control power, reset and wake-up Management.

18.2.2.1.1 OCP Slave Interfaces

PRCM implements a 32-bit OCP target interface compliant to the OCP/IP2.0 standard. PRCM is specified to operate at “clkin (27 Mhz)” clock rate.

18.2.3 Power Control Interface

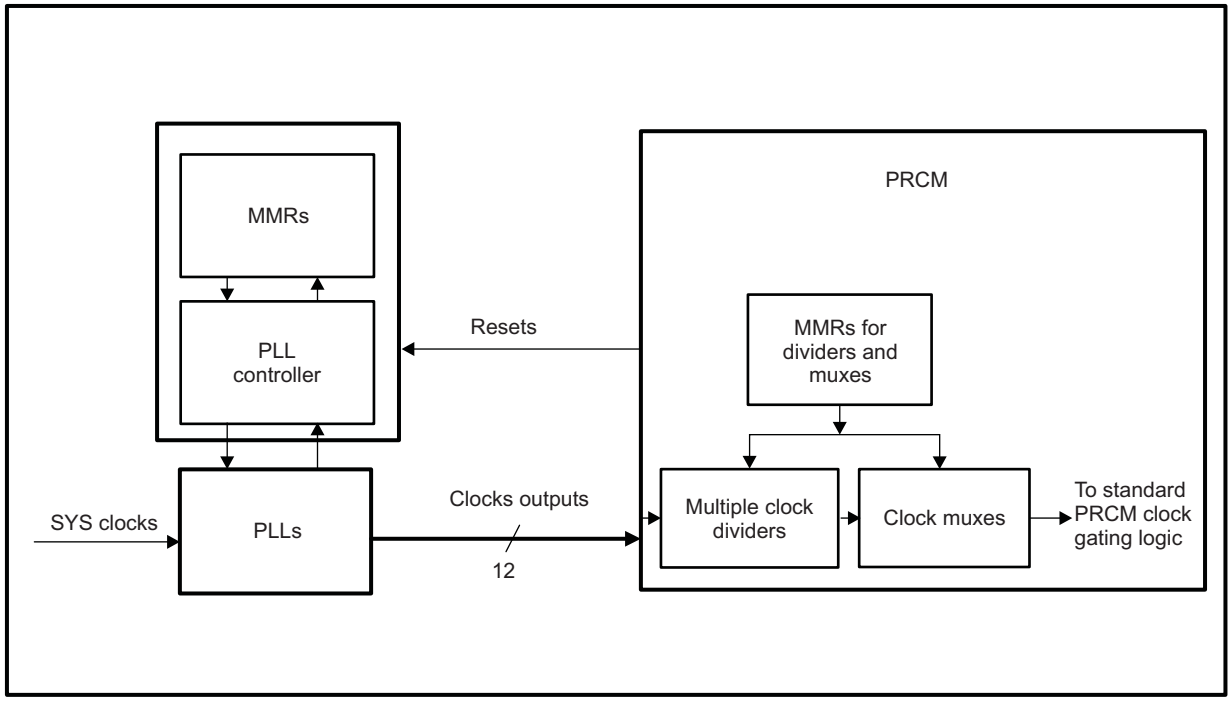
The Device does has power domain switches over the device, this interface provides PRCM control over power domain switches and receives responses from the power domains which indicate the switch status. It also controls the isolation signals. The control for power domain switches will be latched in PRCM Status Registers.

18.2.4 FAPLL interface

All Flying adder PLL control signals will be generated outside PRCM. Following diagram shows the overview of Flying adder PLLs with PRCM.

Figure 18-5 shows the proposed architecture of Device Flying-adder PLLs along with PRCM.

Figure 18-5. Device Flying-Adder PLLs



Following are the key points The PLLs (4 nos.) and PLL controllers are outside PRCM and all the controls required by PLLs will be generated by PLL controller. PLL controllers will be configured/controlled by chip level MMRs. All the resets required by these MMRs, PLL controllers will be generated by PRCM.

18.2.5 Device Control Interface

This interface provides PRM management of several device-level features which are not specific to any single power domain. This PRM interface controls signals to/from the device for global control:

- Device type coding
- I/Os isolation control

18.2.6 Clocks Interface

This interface gathers all clock inputs and outputs managed by PRCM modules.

18.2.7 Resets Interface

This interface gathers all resets inputs and outputs managed by PRCM module.

18.2.8 Modules Power Management Control Interface

Modules or sub-systems in the device are split over 2 categories:

Initiator: an initiator is a module able to generate traffic on the device interconnects (typically: processors, MMU, eDMA...).

Target: a target is a module that cannot generate traffic on the device interconnects, but that can generate interrupts or DMA request to the system (typically: peripherals). PRCM handles a power management handshake protocol with each module or sub-system. This protocol allows performing proper clock and power transition taking into account each module activity or state.

18.2.8.1 Initiator Modules Interface

PRCM module handle all initiator modules power management interfaces: MStandby signal MWait signal.

18.2.8.2 Targets Modules Interface

PRCM module handle all target modules power management interfaces: SIdleReq signal SIdleAck signal FCLKEN signal PRCM module handles all target modules wake-up: SWakeup signal.

18.3 Device Modules and Power-Management Attributes List

18.3.1 Active Power Domain Modules Attribute

Table 18-13. Active Power Domain Modules Attribute

Module Instance	Clock Domain Name	Module PM Ports
C674x DSP	GEM_GCLKIN	Master Slave
HD_DSS	HDSS_L3_GCLK	Master Slave

18.3.2 AlwaysOn Power Domain Modules Attribute

Table 18-14. AlwaysOn Power Domain Modules Attribute

Module Instance	Clock Domain Name	Module PM Ports
CONTRL	ALWON_L3_SLOW_GCLK	Slave
10/100/1000 EMAC0	ETHERNET_GCLK	Master Slave
10/100/1000 EMAC1	ETHERNET_GCLK	Master Slave
GPIO0	ALWON_L3_SLOW_GCLK	Slave
GPIO1	ALWON_L3_SLOW_GCLK	Slave
GPMC	ALWON_L3_SLOW_GCLK	Slave
I2C0	ALWON_L3_SLOW_GCLK	Slave
I2C1	ALWON_L3_SLOW_GCLK	Slave
L3	SYSCLK4_GCLK	Slave
L4_HS	SYSCLK5_GCLK	Slave
L4_STD	SYSCLK6_GCLK	Slave
MAILBOX	ALWON_L3_SLOW_GCLK	Slave
MCASP0	ALWON_L3_SLOW_GCLK	Slave
MCASP1	ALWON_L3_SLOW_GCLK	Slave
MCASP2	ALWON_L3_SLOW_GCLK	Slave

Table 18-14. AlwaysOn Power Domain Modules Attribute (continued)

Module Instance	Clock Domain Name	Module PM Ports
MCBSP	ALWON_L3_SLOW_GCLK	Slave
MMU	MMU_GCLK	Slave
MMU_CFG	MMU_CFG_GCLK	Slave
MPU	MPU_GCLK	Master Slave
OCMC_RAM0	OCMC0_GCLK	Slave
OCMC_RAM1	OCMC1_GCLK	Slave
RTC	OCMC0_GCLK	Slave
SDIO	ALWON_L3_SLOW_GCLK	Slave
SMARTREFLEX0	ALWON_L3_SLOW_GCLK	Slave
SMARTREFLEX1	ALWON_L3_SLOW_GCLK	Slave
SPI	ALWON_L3_SLOW_GCLK	Slave
SPINBOX	ALWON_L3_SLOW_GCLK	Slave
TIMER1	ALWON_L3_SLOW_GCLK	Slave
TIMER2	ALWON_L3_SLOW_GCLK	Slave
TIMER3	ALWON_L3_SLOW_GCLK	Slave
TIMER4	ALWON_L3_SLOW_GCLK	Slave
TIMER5	ALWON_L3_SLOW_GCLK	Slave
TIMER6	ALWON_L3_SLOW_GCLK	Slave
TIMER7	ALWON_L3_SLOW_GCLK	Slave
TPCC	L3_FAST_ALWON_GCLK	Master Slave
TPTC0	L3_FAST_ALWON_GCLK	Master Slave
TPTC1	L3_FAST_ALWON_GCLK	Master Slave
TPTC2	L3_FAST_ALWON_GCLK	Master Slave
TPTC3	L3_FAST_ALWON_GCLK	Master Slave
UART0	ALWON_L3_SLOW_GCLK	Slave
UART1	ALWON_L3_SLOW_GCLK	Slave
UART2	ALWON_L3_SLOW_GCLK	Slave
WDTIMER	RTC_GCLK	Slave

18.3.3 Default Power Domain Modules Attribute

Table 18-15. Default Power Domain Modules Attribute

Module Instance	Clock Domain Name	Module PM Ports
DMM	L3_FAST_DEFAULT_GCLK	Slave
EMIF4_0	L3_FAST_DEFAULT_GCLK	Slave
EMIF4_1	L3_FAST_DEFAULT_GCLK	Slave
EMIF_FW	L3_FAST_DEFAULT_GCLK	Slave
PCI	PCI_GCLK	Master Slave
SATA	L3_MED_DEFAULT_GCLK	Master Slave

Table 18-15. Default Power Domain Modules Attribute (continued)

Module Instance	Clock Domain Name	Module PM Ports
USB	USB_GCLK	Master Slave

18.3.4 HDVICP2-0 Power Domain Modules Attribute

Table 18-16. HDVICP2-0 Power Domain Modules Attribute

Module Instance	Clock Domain Name	Module PM Ports
HDVICP2-0	HDVICP0_GCLK	Master Slave
SL2_0	HDVICP0_GCLK	Slave

18.3.5 HDVICP2-1 Power Domain Modules Attribute

Table 18-17. HDVICP2-1 Power Domain Modules Attribute

Module Instance	Clock Domain Name	Module PM Ports
HDVICP2-1	HDVICP1_GCLK	Master Slave
SL2_1	HDVICP1_GCLK	Slave

18.3.6 HDVICP2-2 Power Domain Modules Attribute

Table 18-18. HDVICP2-2 Power Domain Modules Attribute

Module Instance	Clock Domain Name	Module PM Ports
HDVICP2-2	HDVICP2_GCLK	Master Slave
SL2_2	HDVICP2_GCLK	Slave

18.3.7 SGX Power Domain Modules Attribute

Table 18-19. SGX Power Domain Modules Attribute

Module Instance	Clock Domain Name	Module PM Ports
SGX	SGX_GCLK	Master Slave

18.4 Clock Management

PRCM provides a centralized control for the generation, distribution and gating of most clocks in the device. PRCM gathers externally clocks and device internally generated clocks (FAPLL clocks) for distribution to the other modules in the device. PRCM module manages system clock generation.

18.4.1 Terminology

The PRCM produces two types of clock: interface and functional clocks.

Interface clocks: these clocks primarily provide clocking for the system interconnect modules and the portions of device's functional modules which interface to the system interconnect modules. In most cases, the interface clock supplies the functional module's system interconnect interface and registers. For some modules, interface clock is also used as functional clock. In this specification, interface clocks are represented by blue lines.

Functional clock: this clock supplies the functional part of a module or a sub-system. In some cases, a module or a subsystem may require several functional clocks: 1 or several main functional clock(s), 1 or several optional clock(s). A module needs its main clock(s) to be operational. Optional clocks are used for specific features and can be shutdown without stopping the module.

18.4.2 External Clock Sources to PRCM

The Device requires the following clocks: The 32 kHz frequency is used for low frequency operation. It supplies the clock to the RTC. The MAIN PLL CLOCKS are the source clocks for the C674x DSP, Cortex-A8, HDVICP2 and all the interconnect clocks. The DDR PLL CLOCKS are the source clocks for DDR. The VIDEO PLL CLOCKS are the source clocks for HD-DSS. The AUDIO PLL CLOCKS are the source clocks for McASPs, McBSPs. The Device delivers four clock signals to external devices:

Output clock SYSCLK_OUT can deliver clock outputs of each PLL. CM_CLKOUT_CTRL Register controls the Clock selection and division ratio.

Table 18-20. External Clock Sources to PR

Clock name	Source	Destination	Description
Main_pll_clock1	Main PLL	PRCM	Clock input from Main PLL (Flying adder Synthesizer 1)
Main_pll_clock2	Main PLL	PRCM	Clock input from Main PLL (Flying adder Synthesizer 2)
Main_pll_clock3	Main PLL	PRCM	Clock input from Main PLL (Flying adder Synthesizer 3)
Main_pll_clock4	Main PLL	PRCM	Clock input from Main PLL (Flying adder Synthesizer 4)
DDR_pll_clock1	DDR PLL	PRCM	Clock input from DDR PLL (Flying adder Synthesizer 1)
DDR_pll_clock2	DDR PLL	PRCM	Clock input from DDR PLL (Flying adder Synthesizer 2)
DDR_pll_clock3	DDR PLL	PRCM	Clock input from DDR PLL (Flying adder Synthesizer 3)
Video_pll_clock1	Video PLL	PRCM	Clock input from Video PLL (Flying adder Synthesizer 1)
Video_pll_clock2	Video PLL	PRCM	Clock input from Video PLL (Flying adder Synthesizer 2)
Video_pll_clock3	Video PLL	PRCM	Clock input from Video PLL (Flying adder Synthesizer 3)
Audio_pll_clock1	Audio PLL	PRCM	Clock input from Audio PLL (Flying adder Synthesizer 1)
Audio_pll_clock2	Audio PLL	PRCM	Clock input from Audio PLL (Flying adder Synthesizer 2)
Audio_pll_clock3	Audio PLL	PRCM	Clock input from Audio PLL (Flying adder Synthesizer 3)
Audio_pll_clock4	Audio PLL	PRCM	Clock input from Audio PLL (Flying adder Synthesizer 4)
Audio_pll_clock5	Audio PLL	PRCM	Clock input from Audio PLL (Flying adder Synthesizer 5)
32 KHz	External	PRCM	32 kHz clock
CLKIN	External	PRCM	27 MHz clock
TCLKIN	External	PRCM	Timer clock
PCI_CLK	External	PRCM	Clock input from Main PLL (Flying adder Synthesizer 5)
SYSCLK_OUT	PRCM	External	Observation clock

18.4.3 Internal Clock Sources

Table 18-21. Internal Clock Sources

Clock Name	Frequency	Destination
SYSCLK1	~1 GHz	To C674x DSP
SYSCLK2	~1.2 GHz	To Cortex-A8
SYSCLK3	~600 MHz	To HDVICP2
SYSCLK4	~500 MHz	Interconnect Clock, Clock for HD DSS, TPTCs, TPCC, DMM
SYSCLK5	~250 MHz	Interconnect Clock, SGX530, USB SS, 10/100/1000 EMAC, SATA, PCIe, OCMC RAM
SYSCLK6	~125 MHz	Interconnect Clock, UART, I2C, SPI, SDIO, TIMER, GPIO, McASP, McBSP, GPMC
SYSCLK7	125 MHz (maximum)	Reserved
SYSCLK8	800 MHz	DDR clock
SYSCLK9	16 MHz	CEC clock, VTP
SYSCLK10	48 MHz	SPI. I2C, SDIO, and UART functional clock
SYSCLK11	216 MHz	Reserved
SYSCLK13	165 MHz (maximum)	HDVPSS
SYSCLK14	27 MHz	Reserved
SYSCLK15	165 MHz (maximum)	HDVPSS
SYSCLK16	27 MHz	Reserved
SYSCLK17	54 MHz	HDVPSS
SYSCLK18	32 KHz	RTC
SYSCLK19	62.5 MHz	Reserved
MCASP0_CLK		McASP0 AUX Clock
MCASP1_CLK		McASP1 AUX Clock
MCASP2_CLK		McASP2 AUX Clock
MCBSP_CLK		McBSP AUX Clock
TIMER1_CLK		Timer clock for Timer 1
TIMER2_CLK		Timer clock for Timer 2
TIMER3_CLK		Timer clock for Timer 3
TIMER4_CLK		Timer clock for Timer 4
TIMER5_CLK		Timer clock for Timer 5
TIMER6_CLK		Timer clock for Timer 6
TIMER7_CLK		Timer clock for Timer 7

18.4.4 Clock Generation

18.4.4.1 Main FAPLL Interface to PRCM

Figure 18-6 shows the interface between main FAPLL and PRCM.

CAUTION

"Glitch-free" muxes have an output which is free from any glitches. It requires the previously selected clock and the newly selected clock to both be freerunning for switch-over to occur. Switchover could take 1-2 clock cycles of previously selected and newly selected clock to complete. If newly selected clock is idle, a switchover never occurs (previously selected clock continues to pass through the mux). Do not switch to a non-existent clock.

Figure 18-6. Main FAPLL Interface to PRCM

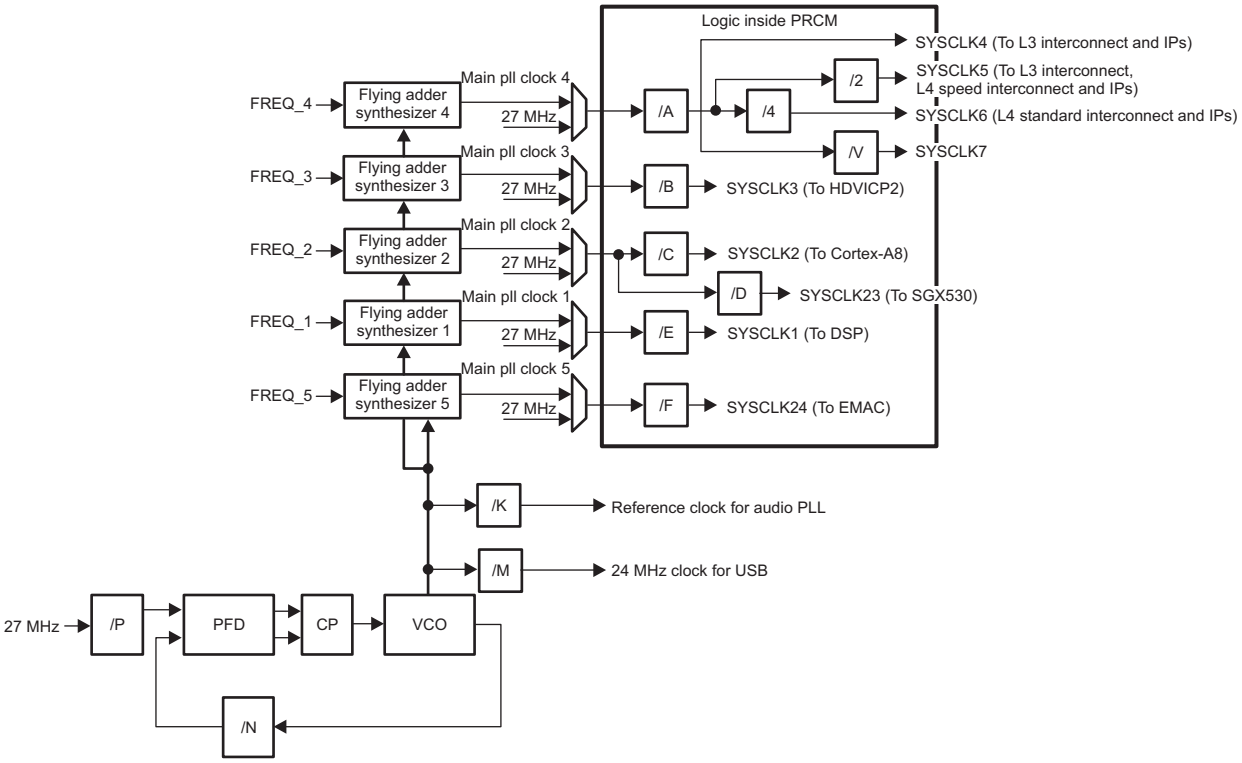


Table 18-22. MAIN PLL Dividers

Control Bit Filed	Divider	Supported Divide Ratios	Default Value
CM_SYSCLK4_CLKSEL.[CLKSEL]	A	1/1, 1/2	1/1
CM_SYSCLK3_CLKSEL.[CLKSEL]	B	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
CM_SYSCLK2_CLKSEL.[CLKSEL]	C	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
CM_SYSCLK23_CLKSEL.[CLKSEL]	D	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/3
CM_SYSCLK1_CLKSEL.[CLKSEL]	E	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
CM_SYSCLK7_CLKSEL.[CLKSEL]	V	1/5, 1/6, 1/8, 1/16	1/5
CM_SYSCLK24_CLKSEL.[CLKSEL]	F	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
Main PLL Divider 7 Register (MAINPLL_DIV7)	K	MAINPLL_DIV7 (see Section 1.16.1.2.14for details)	1/4

Table 18-22. MAIN PLL Dividers (continued)

Control Bit Filed	Divider	Supported Divide Ratios	Default Value
Main PLL Divider 6 Register (MAINPLL_DIV6)	M	MAINPLL_DIV6 (see Section 1.16.1.2.13 for details)	1/72

18.4.4.2 DDR FAPLL interface to PRCM

Figure 18-7 shows the interface between DDR FAPLL and PRCM.

Figure 18-7. DDR FAPLL Interface to PRCM

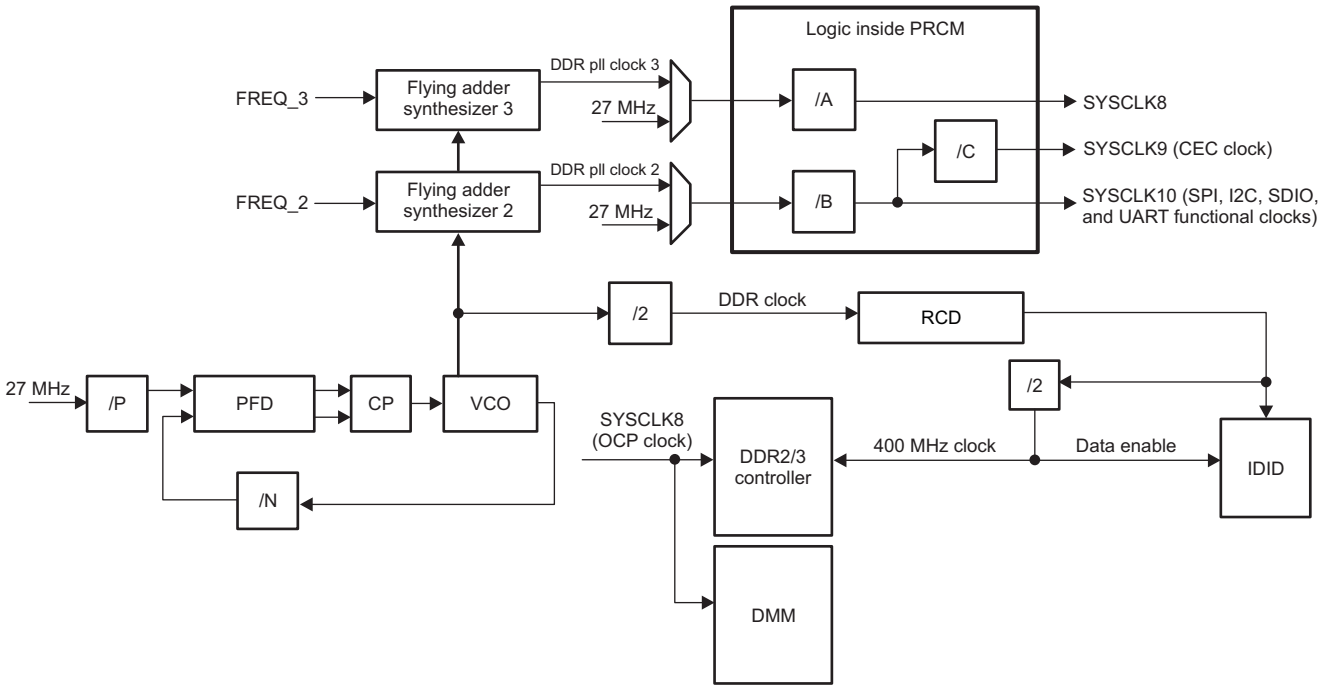


Table 18-23. DDR PLL Dividers

Control Bit Filed	Divider	Supported Divide Ratios	Default Value
-	A	1/1	1/1
CM_SYSCLK10_CLKSEL[CLKSEL]	B	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
-	C	1/3	1/3

18.4.4.3 Video FAPLL Interface to PRCM

Figure 18-8 shows the interface between video FAPLL and PRCM.

Figure 18-8. Video FAPLL Interface to PRCM

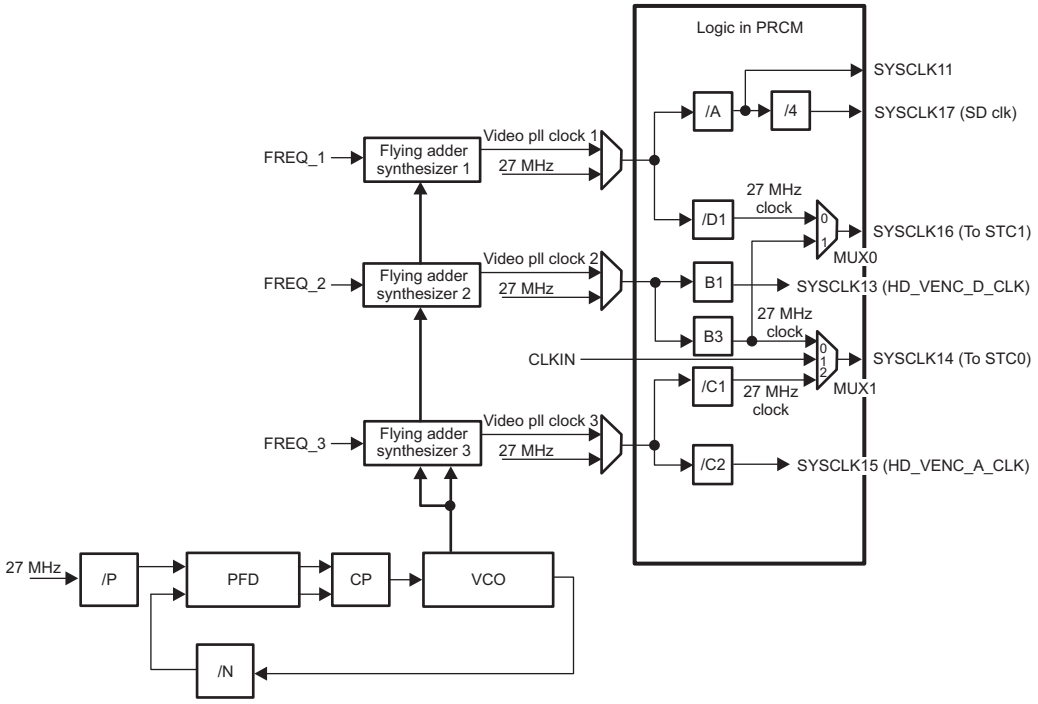


Table 18-24. Video PLL Dividers

Control Bit Filed	Divider	Supported Divide Ratios	Default Value
CM_SYSCLK11_CLKSEL[CLKSEL]	A	1/1, 1/2	1/1
CM_VPD1_CLKSEL [CLKSEL]	D1	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/8
CM_SYSCLK13_CLKSEL [CLKSEL]	B1	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/8
CM_VPB3_CLKSEL [CLKSEL]	B3	1/1,1/2, 1/22	1/22
CM_VPC1_CLKSEL [CLKSEL]	C1	1/1,1/2, 1/22	1/22
CM_SYSCLK15_CLKSEL [CLKSEL]	C2	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/4
CM_SYSCLK16_CLKSEL[CLKSEL]	MUX0	0,1	0
CM_SYSCLK14_CLKSEL[CLKSEL]	MUX1	0,1,2	0

18.4.4.4 Audio FAPLL Interface to PRCM

Figure 18-9 shows the interface between Audio FAPLL and PRCM.

Figure 18-9. Audio FAPLL Interface to PRCM

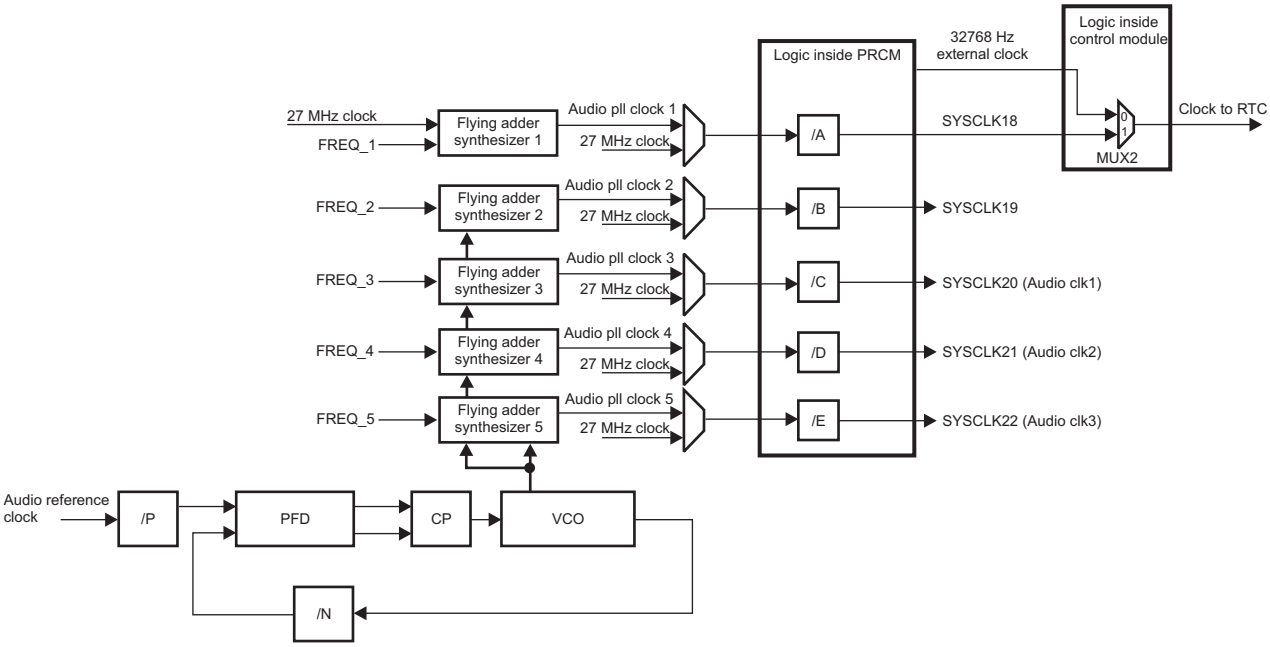


Table 18-25. Audio PLL Dividers

Control Bit Filed	Divider	Supported Divide Ratios	Default Value
CM_APA_CLKSEL[CLKSEL]	A	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
CM_SYSCLK19_CLKSEL[CLKSEL]	B	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
CM_SYSCLK20_CLKSEL[CLKSEL]	C	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
CM_SYSCLK21_CLKSEL[CLKSEL]	D	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
CM_SYSCLK22_CLKSEL[CLKSEL]	E	1/1,1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8	1/1
CM_SYSCLK18_CLKSEL[CLKSEL]	MUX2	0,1	0

Figure 18-10. Timer Functional Clock Mux

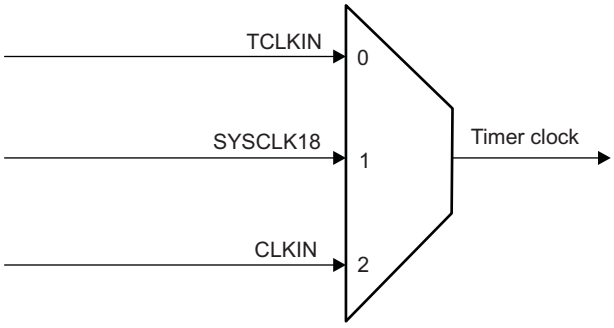


Table 18-26. Timer Functional Mux Select

Control Bit Filed	Mux Select	Default Value
CM_TIMER1_CLKSEL[CLKSEL]	0,1,2	1
CM_TIMER2_CLKSEL[CLKSEL]	0,1,2	1
CM_TIMER3_CLKSEL[CLKSEL]	0,1,2	1
CM_TIMER4_CLKSEL[CLKSEL]	0,1,2	1
CM_TIMER5_CLKSEL[CLKSEL]	0,1,2	1
CM_TIMER6_CLKSEL[CLKSEL]	0,1,2	1

Figure 18-11 shows McASP and McBSP clock select options. For more information on Clocking and FAPLL, refer to Chip Level resources.

Figure 18-11. McASP and McBSP Clock Connections

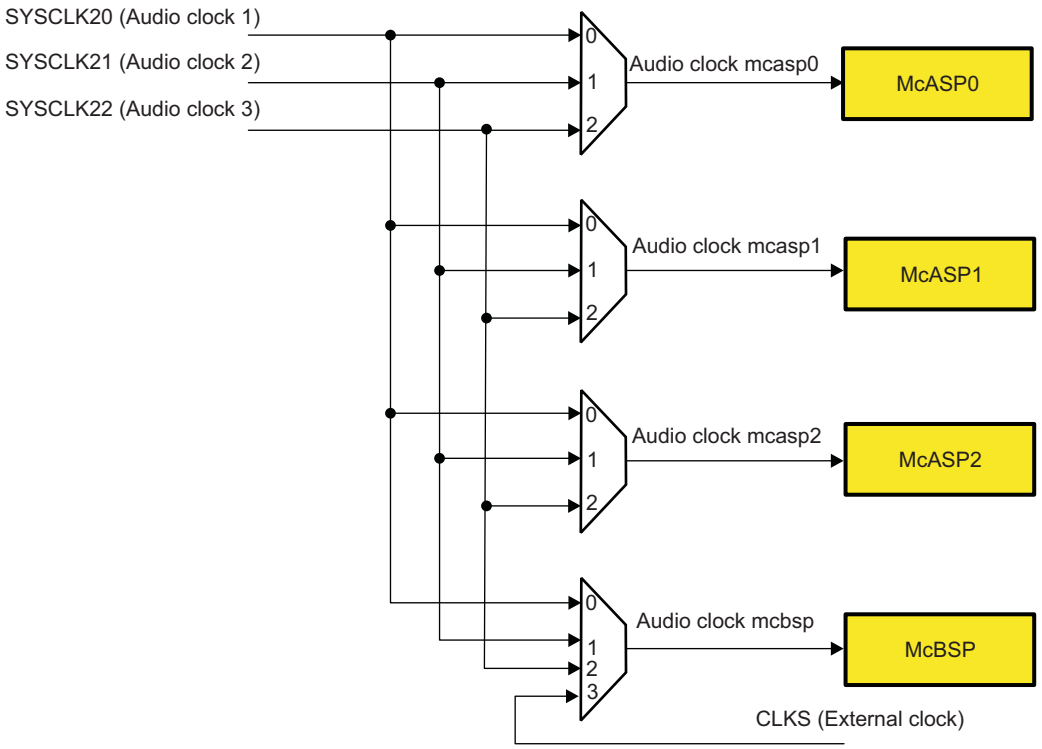


Table 18-27. McASP and McBSP Clock Mux Select

Control Bit Filed	Mux Select	Default Value
CM_AUDIOCLK_MCASP0_CLKSEL [CLKSEL]	0,1,2	1
CM_AUDIOCLK_MCASP1_CLKSEL [CLKSEL]	0,1,2	1
CM_AUDIOCLK_MCASP2_CLKSEL [CLKSEL]	0,1,2	1
CM_AUDIOCLK_MCBSP_CLKSEL [CLKSEL]	0,1,2,3	1

18.5 Reset Management

18.5.1 Overview

The PRCM manages the resets to all power domains inside device. And, it manages generation of a single reset output signal through device pin, SYS_RESWARM_RST, for external use. The PRCM has no knowledge of or control over resets generated locally within a module, for example, via the OCP configuration register bit IPName_SYSCONFIG.SoftReset.

All PRM reset outputs are asynchronously asserted. These outputs are active-low except for the DPLL resets. Deassertion is synchronous to the clock which runs a counter used to stall, or delay, reset de-assertion upon source deactivation. This clock will be SYS_CLK used by all the reset managers. All modules receiving a PRCM generated reset are expected to treat the reset as asynchronous and implement local re-synchronization upon de-activation as needed.

One or more Reset Managers are required per power domain. Independent management of multiple reset domains is required to meet the reset sequencing requirements of all modules in the power domain.

18.5.2 Reset Concepts and Definitions

The PRCM collects many sources of reset. Here below is a list of qualifiers of the source of reset:

- Cold reset: it affects all the logic in a given entity
- Warm reset: it is a partial reset which doesn't affect all the logic in a given entity
- Global reset: it affects the entire device
- Local reset: it affects part of the device (1 power domain for example)
- Software reset: it is initiated by software
- Hardware reset: it is hardware driven

Each reset source is specified as being a cold or warm type. Cold types are synonymous with power-on-reset (POR) types. Such sources are applied globally within each receiving entity (sub-system, module, macro-cell) upon assertion. Cold reset events include: device power-up, power-domain power-up, and E-Fuse programming failures.

Warm reset types are not necessarily applied globally within each receiving entity. A module may use a warm reset to reset a subset of its logic. This is often done to speed-up reset recovery time, that is, the time to transition to a safe operating state, compared to the time required upon receipt of a cold reset. Warm reset events include: software initiated per power-domain, watch-dog time-out, externally triggered, and emulation initiated.

Reset sources, warm or cold types, intended for device-wide effect are classified as global sources. Reset sources intended for regional effect are classified as local sources.

Each Reset Manager provides two reset outputs. One is a cold reset generated from the group of global and local cold reset sources it receives. The other is a warm+cold reset generated from the combined groups of, global and local, cold and warm reset sources it receives.

The Reset Manager asserts one, or both, of its reset outputs asynchronously upon reset source assertion. Reset deassertion is extended beyond the time the source gets de-asserted. The reset manager will then extend the active period of the reset outputs beyond the release of the reset source, according to the PRCM's internal constraints and device's constraints. Some reset durations can be software-configured. Most (but not all) reset sources are logged by PRCM's reset status registers. The same reset output can generally be activated by several reset sources and the same reset source can generally activate several reset outputs. All the reset signals output of the PRCM are active low. Several conventions are used in this document for signal and port names. They include:

- "_RST" in a signal or port name is used to denote reset signal.
- "_PWRN_RST" in a signal or port name is used to denote a cold reset source

Table 18-28. Reset Sources Overview

	Sys pwron	Icepick por	Global Cold SW	Sys in Warm	Icepick Warm	Global SW Warm	MPU WD
ACT_DOM_RST	X	X	X	X	X	X	X
ALW_DOM_RST	X	X	X	X	X		X
DEF_DOM_RST	X	X	X	X	X	X	X
DPLL_AUDIO_RST	X	X	X	X			
DPLL_DDR_RST	X	X	X	X			
DPLL_MAIN_RST	X	X	X	X			
DPLL_VIDEO_RST	X	X	X	X			
DSS_M3_PWRN_RST	X	X	X	X			
DSS_M3_RST1	X	X	X	X	X	X	X
DSS_M3_RST2	X	X	X	X	X	X	X
DSS_M3_RST3	X	X	X	X	X	X	X
EMU_EARLY_PWRN_RST	X			X			
EMU_PWRN_RST	X	X	X	X			
EMU_RST	X	X	X	X	X	X	X
GEM_GRST	X	X	X	X	X	X	X
GEM_LRST	X	X	X	X	X	X	X
GEM_POR	X	X	X	X			
IVAHD0_PWRN_RST	X	X	X	X			
IVAHD0_RST	X	X	X	X	X	X	X
IVAHD0_SEQ1_RST	X	X	X	X	X	X	X
IVAHD0_SEQ2_RST	X	X	X	X	X	X	X
IVAHD1_PWRN_RST	X	X	X	X			
IVAHD1_RST	X	X	X	X	X	X	X
IVAHD1_SEQ1_RST	X	X	X	X	X	X	X
IVAHD1_SEQ2_RST	X	X	X	X	X	X	X
IVAHD2_PWRN_RST							
IVAHD2_RST	X	X	X	X	X	X	X
IVAHD2_SEQ1_RST	X	X	X	X	X	X	X
IVAHD2_SEQ2_RST	X	X	X	X	X	X	X
MPU_AO_RST	X	X	X	X			
MPU_PWRN_RST	X	X	X	X			
MPU_RST	X	X	X	X	X	X	X
PCI_LRST	X	X	X	X	X	X	X
SGX_RST	X	X	X	X	X	X	X
USB1_RST	X	X	X	X	X	X	X
USB2_RST	X	X	X	X	X	X	X
USB_POR	X	X	X	X			

18.5.3 Global Power-On (Cold) Reset

The PRCM is required internally to generate the global power-on reset used by the domain Reset Managers from three cold reset sources: SYS_PWRN_RST, ICEPICK_POR_RST, GLOBAL_COLD_SW_RST.

Source SYS_PWRN_RST is received over a dedicated device input pin, typically driven by the same power IC that supplies device power and 27MHz clock. This is the normal system POR.

The internal emulation module, ICE-Pick, generates ICEPICK_POR_RST. This reset is used in emulation mode only.The PRCM is required to provide an output port, ACT_LIKE_SECURE, which is asynchronously set to a logic high upon ICEPICK_POR_RST assertion. This state is maintained until a normal system POR occurs.

Source GLOBAL_COLD_SW_RST is a PRM internally generated one. Activation is triggered upon setting PRM memory-mapped register bit, PRM_RSTCTRL.GLOBAL_COLD_SW_RST. This bit is self-clearing, it is automatically cleared by the hardware.

All the reset driven by the PRCM are asserted when any one of global power on reset sources above are active. The PRCM must stall the release of the global power-on reset until after de-assertion of all cold reset sources and until after receiving indication that the voltage domains, have all ramped to their operating levels and until after receiving an indication that the system clock is running and is stable. The PRCM implements a counter after all of these indications have occurred to provide a delay before releasing internal global power on reset signals. This requires use of a PM FW Reset Manager. The "always-on" 27 MHz clock must run the RM's stall-period timer. The max count is provided by PRM register RM_RSTTIME.RSTIME1 bit-field. During device power-up, this mechanism enforces the system requirement of a running 27 MHz clock.

18.5.3.1 Power-On Reset Sequence

The following sequence describes the main chronological steps during the power-on reset sequence of the device.

- The system settings are done (all the voltages are ramped-up, 27 MHz clock is stable, power domains are ramped up) and now SYS_PWRN_RST is de-asserted and the PRCM proceeds with the device power-up sequence.
- Following the de-assertion of this power on reset signal, the PRCM starts the power on reset sequence.
- SYS_WARM_OUT_RESET will be de-asserted by PRCM.
- The DPLLs are released from reset.
- The MPU is released from reset provided the MPU clock is running.
- Once MPU_RST_DONE is asserted by MPU, all other domain resets are released provided the domain clocks are running.
- All the software controlled resets, will either remain asserted/De-asserted as per the default MMR bit.

18.5.4 Global Warm Reset

The PRCM is required to generate the global reset used by the domain Reset Managers from the following warm reset input sources: SYS_WARM_IN_RST, MPU_WD_RST, GLOBAL_SW_WARM_RST, and ICEPICK_RST.

Source SYS_WARM_IN_RST is received over the input path from a dedicated device bidirectional pin. The source is typically hardware generated via a reset push-button switch. Source MPU_WDT_RST is generated from internal watchdog timer module. Activation is triggered by a timeout event. Source GLOBAL_WARM_SW_RST is PRCM internally generated. Activation is triggered upon setting PRM memory-mapped register bit. This bit is self-clearing, that is, it is automatically cleared by the hardware.

18.5.4.1 Global Warm Reset Sequence

The following sequence describes the main chronological steps during a global warm reset sequence: Assumptions are: Device V_{DD} is regulated at to the required voltage The system is running: Resets are released All the PLLs are locked.

- Upon assertion of any global warm reset source: The GLOBAL_WARM_RSTACTST signal is asserted High by the PRCM to indicate to the EMIF module that global warm reset event has occurred. Following the assertion of the status signal, EMIF starts putting its OCP interface into the idle state. The PRCM delays global warm reset to the device for minimum 16 L3 clock cycles until EMI puts its OCP interface properly in idle state.

  In parallel to this, the PRCM also asserts the EMIF_IDLEREQ driven to EMIF module so that EMIF can drain all its outstanding requests and starts putting the external SDRAM memory into the self-refresh mode. The EMIF asserts its IDLEACK high once memory is properly put in self-refresh mode.
- The device reset manager resets part of the device by asserting the global warm reset The external warm reset is asserted (SYS_NRES_WARM_OUT port) but gets de-asserted after approx 30 CLKIN cycles irrespective of SYS_WARMIN_RST state.
  - All the domain warm resets are asserted
  - The DPLL resets are not asserted
  - The system clock, CLKIN, is running at the system clock frequency
  - The registers sensitive to a warm reset are synchronously reset (PRCM registers sets)
  - The PRCM cuts all the clocks not requested by the registers reset value setting
- The global warm reset is released. The global warm reset is extended after release of the warm reset source until all the following conditions are met:

  Device reset manager counter overflowed (setup by the register PRM_RSTTIME.RSTTIME2) Voltages are stable
- The external warm reset is de-asserted.
- The MPU clock is running.
- The MPU domain is released from reset. The MPU re-boots.

NOTE: The C674x DSP, HDVICP2-0, HDVICP2-1, HDVICP2-2, are held under reset after global warm reset by assertion of software source of reset. Other domains such are held under reset after global warm reset until the MPU software enables their respective interface clock.

18.5.5 MPU Subsystem POR Sequence

The power on reset must be applied when MPU SS is first powered up. Following is the detailed description of MPU power on reset sequence.

During the initial power up sequence, the PRCM it asynchronously releases the DPLL_MAIN_RST signal for the main PLL and. Once the main PLL reset is released, it starts the PLL initialization. The PLL is configured in bypass mode and it provides the bypass output CLK to all the modules inside MPU SS.

The PRCM releases the reset to INTC module inside MPUSS, it asynchronously releases the MPU_AO_RST signal to MPUSS.

Before the MPU power on reset is de-asserted, the PRCM must ensure that the MPU CLK is running so that MPU can be reset properly

The PRCM releases MPU_PWRN_RST and it waits until MPUSS power on Reset done, MPU_PWRN_RSTDONE signal is asserted HIGH by the MPUSS.

Following the de-assertion of MPU_PWRN_RST signal, the MPU SS de-asserts the nPORESET signal to the Cortex-A8 CPU and after sufficient number of MPU CLK cycles MPU SS asserts the MPU_PWRN_RSTDONE signal high to indicate to the PRCM that synchronous power on reset sequence is completed inside MPU SS.

Once the PRCM sees that this MPU_PWRN_RSTDONE signal is asserted HIGH, PRCM de-asserts the MPU_RST signal and then the MPU starts booting.

18.5.6 MPU Subsystem Warm Sequence

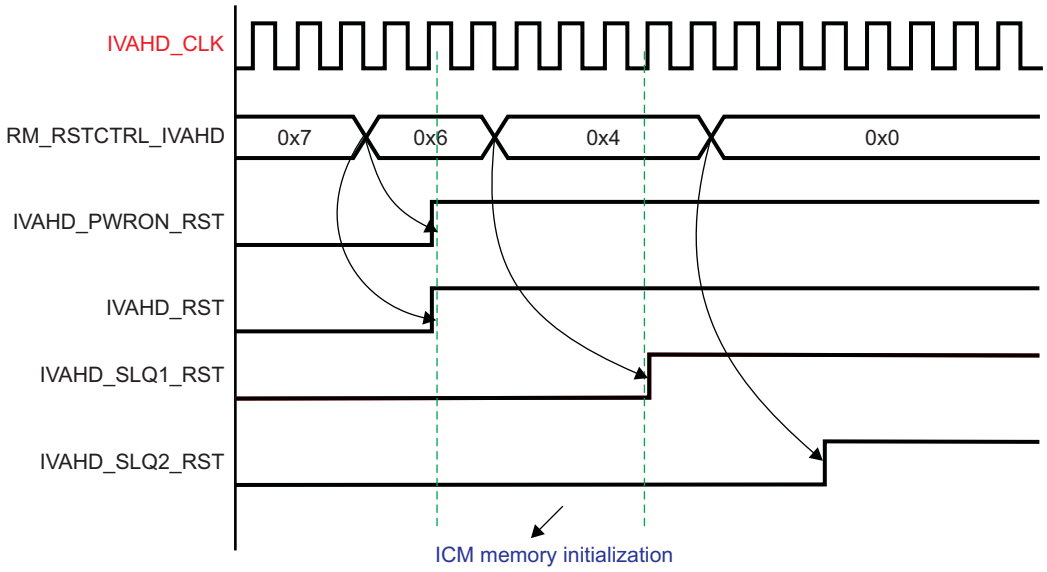
During the warm reset of MPU, the PRCM asserts the MPU_RST signal only and power on reset signal is not asserted.

The MPU_RST is kept asserted for minimum 32 MPU clock cycles, that is, SYS_CLK in this case since during warm the reset of MPU.

18.5.7 HDVICP2 Power-On Reset Sequence

Figure 18-12 illustrates the POR sequence of the HDVICP2.

Figure 18-12. POR Sequence of the HDVICP2



POR to HDVICP2 is applied when PD_IVAHD is powered up.

Whenever the HDVICP2 is enabled by software control, the following sequence happens:

- The PRCM module provides the functional clock IVAHDi_CLK to the HDVICP2
- The POR to HDVICP2 is de-asserted

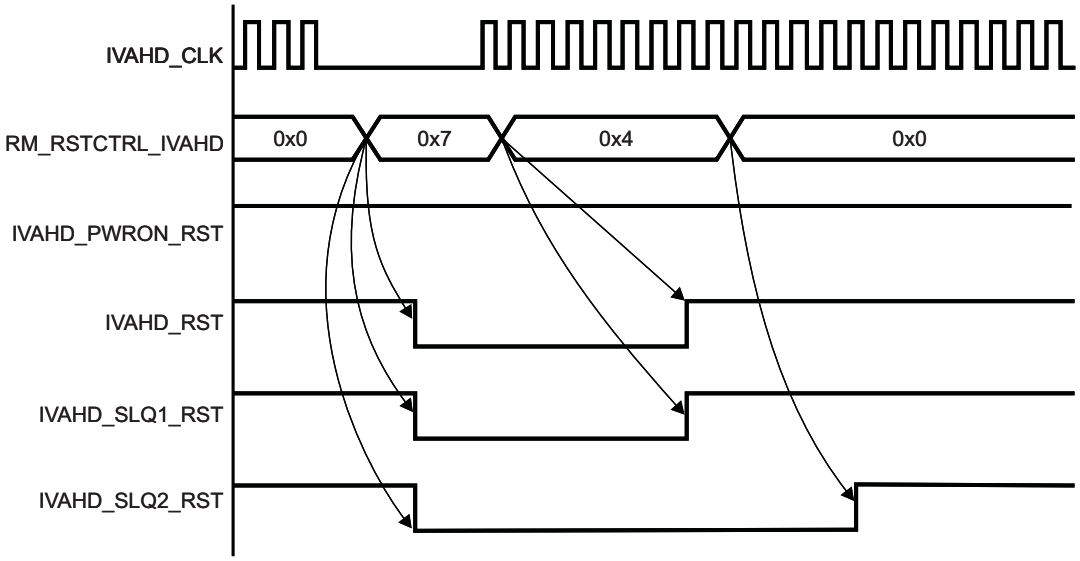
After POR de-assertion, the sequence is:

- Software clears the RM_IVAHDi_RSTCTRL[2] RST3 bit. This causes the PRCM module to release the IVAHDi_PWRN_RST reset, which is used inside HDVICP2 primarily to reset the emulation logic and the IVAHDi_RST reset, which is used to reset all logic inside HDVICP2. Then, software can download data into the TCM memory while keeping the sequencer CPUs under reset.
- Software clears the RM_IVAHDi_RSTCTRL[0] RST1 bit. This releases the IVAHDi_SEQ1_RST reset to the Sequencer1 CPU.
- Software can clear the RM_IVAHDi_RSTCTRL[1] RST2 bit. This releases the IVAHDi_SEQ2_RST reset to the Sequencer2 CPU.

18.5.8 HDVICP2 Software Warm Reset Sequence

Figure 18-13 illustrates the software warm reset sequence of the HDVICP2.

Figure 18-13. Software Warm Reset Sequence of the HDVICP2



Before asserting the software reset to the HDVICP2, the MPU software must ensure that:

- The HDVICP2 sequencer CPUs are in IDLE state (CM_IVAHDi_IVAHD_CLKCTRL[17:16] IDLEST).
- The HDVICP2 is in STANDBY state CM_IVAHDi_IVAHD_CLKCTRL[[18] STBYST).
- The functional clock to theHDVICP2 has been gated by the PRCM module (CM_IVAHDi_CLKSTCTRL[8] CLKACTIVITY_IVAHD_CLK).

The software reset sequence occurs when the MPU software:

1. Sets the RM_IVAHDi_RSTCTRL[2] RST3, RM_IVAHDi_RSTCTRL[1] RST2, and RM_IVAHDi_RSTCTRL[0] RST1 bits. This causes the PRCM module to assert the IVAHDi_RST, IVAHDi_SEQ1_RST, and IVAHDi_SEQ2_RST resets to the HDVICP2. IVAHDi_PWRN_RST remains deasserted.
2. Enables the functional clock to the HDVICP2.
3. clears the RM_IVAHDi_RSTCTRL[2] RST3 and RM_IVAHDi_RSTCTRL[0] RST1 bits. This causes the PRCM module to release the IVAHDi_RST and IVAHDi_SEQ1_RST resets to the HDVICP2.
4. clears the RM_IVAHDi_RSTCTRL[1] RST2 bit. This releases the IVAHDi_SEQ2_RST reset to the Sequencer2 CPU.

18.5.9 C674x DSP Power-On Reset Sequence

The assumptions about POR de-assertion are:

- The MPU software sets the CM_ACTIVE_GEM_CLKCTRL [1:0] MODULEMODE bit field to enabled.
- The MPU software sets the CM_ACTIVE_GEM_CLKSTCTRL [1:0] CLKTRCTRL bit field to SW_WKUP.
- The PRCM is providing the clocks to C674x DSP.

The POR sequence is:

- The PRCM module releases GEM_POR when the reset manager counter (PRM_RSTTIME[14:10] RSTTIME2) reaches its limit.
- The MPU software clears the RM_ACTIVE_RSTCTRL[1] GEM_SW_RST bit in the PRCM module register to release the DSP, MMU, and CACHE interface from reset.
- Once the reset manager counter (PRM_RSTTIME[14:10] RSTTIME2) expires, the PRCM module releases the GEM_GRST signal.
- On deassertion of the GEM_GRST signal, the C674x subsystem starts the initialization sequence. During the initialization sequence all the internal registers inside the C674x subsystem are properly reset and the reset for the DSP, MMU, and CACHE interface completes.
- The MPU software must configure the MMU once the MMU is out of reset. After the MMU is configured, the MPU software clears the RM_ACTIVE_RSTCTRL[0] GEM_LRST bit in the PRCM module register.

The PRCM module releases GEM_LRST, which causes the DSP to start booting.

18.5.10 C674x DSP Warm Reset Sequence

The assumptions about WARM reset de-assertion are:

- The DSP is in IDLE state

The sequence is:

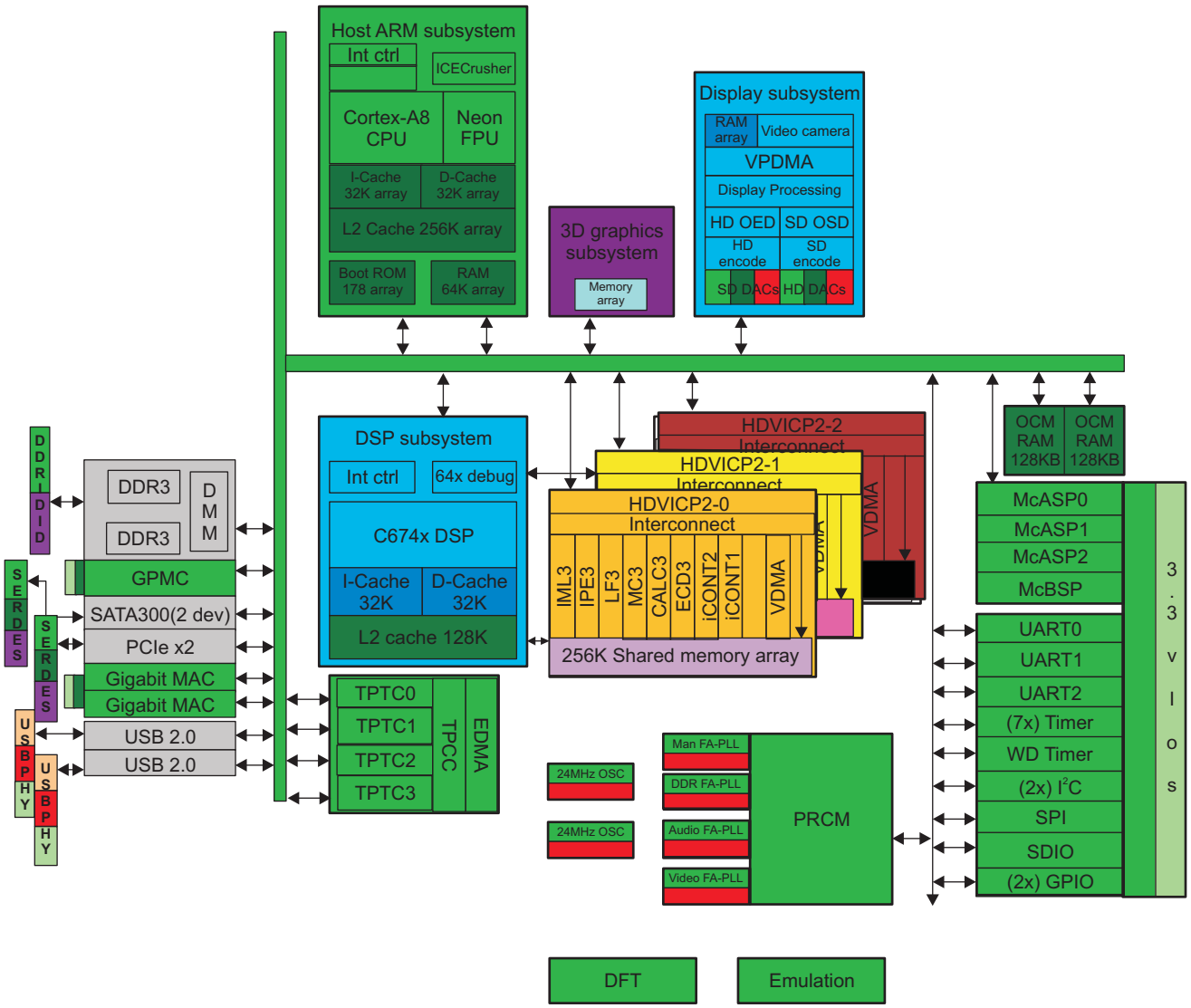
1. The MPU software sets the RM_ACTIVE_RSTCTRL[1] GEM_SW_RST and RM_ACTIVE_RSTCTRL[0] GEM_LRST bits.
2. The PRCM module asserts the GEM_GRST and GEM_LRST reset signals.GEM_POR remains deasserted in this case.
3. The MPU software re-enables the C674x DSP Clocks and clears the RM_ACTIVE_RSTCTRL[1] GEM_SW_RST bit in the PRCM module register to reset the DSP, MMU, and CACHE interface. The DSP subsystem starts the partial initialization sequence for the warm reset.
4. The MPU software clears the RM_ACTIVE_RSTCTRL[0] GEM_LRST bit in the PRCM module register.
5. The PRCM module releases GEM_LRST, which causes the DSP to start booting.

18.6 Power Management

18.6.1 Overview

The PRCM is responsible for managing power domain state transitions through generation of power-down controls to the following cells (for example, isolation latches, domain power switches), and other analog or mixed-signal cells (for example, memory, DPLLs, oscillator) in the device. See Figure 18-14.

Figure 18-14. Voltage and Power Domains



	1V AVS	1V	1.8V	3.3V	1.5V	0.9V
Always-on						
Default						
Active						
HDVICP2-0						
HDVICP2-1						
HDVICP2-2						
SGX						

18.6.2 Power Domains Management

PRCM contains a set of memory-mapped PM-type registers for those functional power domains. These registers allow software to configure define the states of the physical domains under each functional domain state. All domain transitions are fully software controllable. Always ON power domain is never transitioned from the ON power state.

18.6.3 Power Domain Transition Control

The Device has multiple power domains. All these power domains will be controlled by on-chip power switches.

18.6.3.1 Power-Down Sequence

Power-down sequence defined here assumes that IP with STANDBY interface is already in smart-standby mode and STANDBY is already asserted by the IP.

1. Software will request PRCM to put all modules in the specific power domain in “Disable” by programming “Module control Register: Disable” inside PRCM. CM_<Power domain>_<module>_CLKCTRL[x] MODULEMODE = 0 (DISABLED).
2. PRCM starts the power management handshake (IdleReq/IdleAck) with IPs.
3. FCLKEN will be driven LOW by PRCM. PRCM will gate-off all the clock to the functional clock domains.(ICLK and FCLK).
4. Software will request PRCM to put all Interface clock domains in the specific power domain in “force sleep” by programming “Functional clock Domain control register: force sleep” inside PRCM. CM_<Clock domain>_CLKSTCTRL[x] CLKTRCTRL = 1 (SW_SLEEP).
5. Software will request PRCM to put a specific power domain in OFF state by programming MMR (PWRSTCTRL: OFF) inside PRCM. PM_<Power domain>_PWRSTCTRL[POWERSTATE] = 0 (OFF).
6. Specific PSCON inside PRCM will assert the control signals for enabling Isolation cells.
7. PRCM asserts reset of the domain.
8. PSCON then will assert control signal to switch-off the actual power supply.
9. On Die switch will control the actual power supply to the Domain and the acknowledgement is fed back to PRCM.

18.6.3.2 Power-Up Sequence

1. One of the Always on domain IPs send a wake-up interrupt to the Cortex-A8. [All IPs that can generate wake-up are always enabled].
2. Software will request PRCM to put all Interface clock domains in the specific power domain in “force wakeup” by programming “Functional clock Domain control register: force wakeup” inside PRCM. CM_<Clock domain>_CLKSTCTRL[x] .CLKTRCTRL = 2h (SW_WKUP).
3. This will enable power as well as interface clocks.
4. Specific PSCON inside PRCM will assert the control signals to switch-on the actual power supply.
5. This On Die power switch will switch ON the power supply to the domain.
6. Once the power supply is switched ON, acknowledgement is fed back to PSCON.
7. PRCM de-asserts reset of the domain.
8. PRCM will then turn-off isolation cells.
9. Software will request PRCM to put all modules in the specific power domain in “Enable” by programming “Module control Register: Enable” inside PRCM. CM_<Power domain>_<module>_CLKCTRL[x] MODULEMODE = 2h (ENABLED).
10. PRCM will de-assert the “IdleReq” to the modules

NOTE:

- If only modules need to be disabled, then only power-down sequence steps 1, 2, 3 will be performed. (PRCM will clock gate clocks to the module, if all modules sharing that clock are disabled).
- Programming “PWRSTCTRL” can be skipped in both the sequnces. In such a scenario PWRSTCTRL:OFF should be in default state. Whenever Functional clock Domain control register: force sleep is programmed, it will automatically go to OFF state. Whenever Functional clock Domain control registers: force wakeup is programmed, it will automatically go to ON state.
- PWRSTCTRL: ON should be programmed only if there is any requirement of not to switch off the power but gate all the clocks.

18.7 PRCM Registers

Table 18-29 shows the base address offset for the PRCM module. For the base address of these registers, see Table 1-12.

Table 18-29. PRCM Modules

Base Address Offset	Module Name	Section
000h	PRM_DEVICE	Section 18.7.1
100h	CM_DEVICE	Section 18.7.2
200h	OCP_SOCKET_PRM	Section 18.7.3
300h	CM_DPLL	Section 18.7.4
400h	CM_ACTIVE	Section 18.7.5
500h	CM_DEFAULT	Section 18.7.6
600h	CM_IVAHD0	Section 18.7.7
700h	CM_IVAHD1	Section 18.7.8
800h	CM_IVAHD2	Section 18.7.9
900h	CM_SGX	Section 18.7.10
A00h	PRM_ACTIVE	Section 18.7.11
B00h	PRM_DEFAULT	Section 18.7.12
C00h	PRM_IVAHD0	Section 18.7.13
D00h	PRM_IVAHD1	Section 18.7.14
E00h	PRM_IVAHD2	Section 18.7.15
F00h	PRM_SGX	Section 18.7.16
1400h	CM_ALWON	Section 18.7.17

18.7.1 PRM_DEVICE

Table 18-30 lists the PRMDEVICE registers.

Table 18-30. PRM_DEVICE Registers

Offset Address	Acronym	Section
A0h	PRM_RSTCTRL	Section 18.7.1.1
A4h	PRM_RSTTIME	Section 18.7.1.2
A8h	PRM_RSTST	Section 18.7.1.3

18.7.1.1 Reset Control (PRM_RSTCTRL) Register

The reset control (PRM_RSTCTRL) register is shown and described in the figure and table below.

Figure 18-15. Reset Control (PRM_RSTCTRL) Register

31	2	1	0
Reserved		RST_GLOBAL_COLD_SW	RST_GLOBAL_WARM_SW
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-31. Reset Control (PRM_RSTCTRL) Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	RST_GLOBAL_COLD_SW		Global COLD software reset control. This bit is reset only upon a global cold source of reset.
		0	Global COLD software reset is cleared.
		1	Asserts a global COLD software reset. The software must ensure the SDRAM is properly put in sef-refresh mode before applying this reset.
0	RST_GLOBAL_WARM_SW		Global WARM software reset control. This bit is reset upon any global source of reset (warm and cold).
		0	Global warm software reset is cleared.
		1	Asserts a global warm software reset.

18.7.1.2 PRM_RSTTIME Register

The PRM_RSTTIME register is shown and described in the figure and table below.

Figure 18-16. PRM_RSTTIME Register

31	13	12	8	7	0
Reserved		RSTTIME2		RSTTIME1	
		R/W-10h		R/W-6h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-32. PRM_RSTTIME Register Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved
12-8	RSTTIME2	0-1Fh	(Power domain) reset duration 2 (number of RM.SYSCLK clock cycles)
7-0	RSTTIME1	0-FFh	(Global) reset duration 1 (number of SYS_CLK clock cycles)

18.7.1.3 PRM_RSTST Register

The PRM_RSTST register logs the global reset sources. Each bit is set upon release of the domain reset signal. Must be cleared by software. This register is shown and described in the figure and table below.

Figure 18-17. PRM_RSTST Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved						ICEPICK_RST	Reserved
R-0						R/W-0	R-0

7	6	5	4	3	2	1	0
Reserved		EXTERNAL_WARM_RST	Reserved				
R-0		R/W-0	R-0				

3	2	1	0
MPU_WDT_RST	Reserved	GLOBAL_WARM_SW_RST	GLOBAL_COLD_RST
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-33. PRM_RSTST Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	ICEPICK_RST		IcePick reset event. This is a source of global warm reset initiated by the emulation.
		0	No ICEPICK reset
		1	IcePick reset has occurred
8-6	Reserved	0	Reserved
5	EXTERNAL_WARM_RST		External warm reset event
		0	No global warm reset
		1	Global external warm reset has occurred
4	Reserved	0	Reserved
3	MPU_WDT_RST		MPU Watchdog timer reset event. This is a source of global WARM reset.
		0	No MPU watchdog reset
		1	MPU watchdog reset has occurred
2	Reserved	0	Reserved
1	GLOBAL_WARM_SW_RST		Global warm software reset event
		0	No global warm software reset
		1	Global warm software reset has occurred
0	GLOBAL_COLD_RST		Power-on (cold) reset event
		0	No power-on reset
		1	Power-on reset has occurred

18.7.2 CM_DEVICE

18.7.2.1 CM_CLKOUT_CTRL Register

The CM_CLKOUT_CTRL register provides the control over SYS_CCCLKOUT output. It is shown and described in the figure and table below.

Figure 18-18. CM_CLKOUT_CTRL Register

31 8	7	6	5 3	2 0
Reserved	CLKOUT2EN	Reserved	CLKOUT2DIV	CLKOUT2SOURCE
	R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-34. CM_CLKOUT_CTRL Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	CLKOUT2EN		This bit controls the external clock activity
		0	SYS_CLKOUT2 is disabled
		1	SYS_CLKOUT2 is enabled
6	Reserved	0	Reserved
5-3	CLKOUT2DIV		This field controls the external clock divison factor
		0	SYS_CLKOUT2/1
		1h	SYS_CLKOUT2/2
		2h	SYS_CLKOUT2/3
		3h	SYS_CLKOUT2/4
		4h	SYS_CLKOUT2/5
		5h	SYS_CLKOUT2/6
		6h	SYS_CLKOUT2/7
		7h	SYS_CLKOUT2/8
2-0	CLKOUT2SOURCE		This field selects the external output clock source
		0	Source clock is MAIN_PLL_CLK5
		1h	Source clock is DDR_PLL_CLK1
		2h	Source clock is VIDEO_PLL_CLK1
		3h	Source clock is AUDIO_PLL_CLK1

18.7.3 OCP_SOCKET_PRM Device

18.7.3.1 REVISION_PRM Register

The REVISION_PRM register contains the IP revision code for the PRM. It is shown and described in the figure and table below.

Figure 18-19. REVISION_PRM Register

31	8	7	0
Reserved		REV	
R-0		R-10h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-35. REVISION_PRM Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	REV	0-FFh	IP revision [7:4] Major revision [3:0] Minor revision

18.7.4 CM_DPLL Device

Table 18-36 lists the registers for the CM_DPLL device.

Table 18-36. CM_DPLL Device Registers

Offset Address	Acronym	Section
300h	CM_SYSCLK1_CLKSEL	Section 18.7.4.1
304h	CM_SYSCLK2_CLKSEL	Section 18.7.4.2
308h	CM_SYSCLK3_CLKSEL	Section 18.7.4.3
30Ch	CM_SYSCLK4_CLKSEL	Section 18.7.4.4
310h	CM_SYSCLK5_CLKSEL	Section 18.7.4.5
314h	CM_SYSCLK6_CLKSEL	Section 18.7.4.6
318h	CM_SYSCLK7_CLKSEL	Section 18.7.4.7
324h	CM_SYSCLK10_CLKSEL	Section 18.7.4.8
32Ch	CM_SYSCLK11_CLKSEL	Section 18.7.4.9
334h	CM_SYSCLK13_CLKSEL	Section 18.7.4.10
338h	CM_SYSCLK15_CLKSEL	Section 18.7.4.11
340h	CM_VPB3_CLKSEL	Section 18.7.4.12
344h	CM_VPC1_CLKSEL	Section 18.7.4.13
348h	CM_VPD1_CLKSEL	Section 18.7.4.14
34Ch	CM_SYSCLK19_CLKSEL	Section 18.7.4.15
350h	CM_SYSCLK20_CLKSEL	Section 18.7.4.16
354h	CM_SYSCLK21_CLKSEL	Section 18.7.4.17
358h	CM_SYSCLK22_CLKSEL	Section 18.7.4.18
370h	CM_SYSCLK14_CLKSEL	Section 18.7.4.19
374h	CM_SYSCLK16_CLKSEL	Section 18.7.4.20
378h	CM_SYSCLK18_CLKSEL	Section 18.7.4.21
37Ch	CM_AUDIOCLK_MCASP0_CLKSEL	Section 18.7.4.22
380h	CM_AUDIOCLK_MCASP1_CLKSEL	Section 18.7.4.23
384h	CM_AUDIOCLK_MCASP2_CLKSEL	Section 18.7.4.24
388h	CM_AUDIOCLK_MCBSP_CLKSEL	Section 18.7.4.25
390h	CM_TIMER1_CLKSEL	Section 18.7.4.26
394h	CM_TIMER2_CLKSEL	Section 18.7.4.27
398h	CM_TIMER3_CLKSEL	Section 18.7.4.28
39Ch	CM_TIMER4_CLKSEL	Section 18.7.4.29
3A0h	CM_TIMER5_CLKSEL	Section 18.7.4.30
3A4h	CM_TIMER6_CLKSEL	Section 18.7.4.31
3A8h	CM_TIMER7_CLKSEL	Section 18.7.4.32
3B0h	CM_SYSCLK23_CLKSEL	Section 18.7.4.33
3B4h	CM_SYSCLK24_CLKSEL	Section 18.7.4.34

18.7.4.1 CM_SYSCLK1_CLKSEL Register

The M_SYSCLK1_CLKSEL register selects the divider value for SYSCLK1. It is shown and described in the figure and table below.

Figure 18-20. CM_SYSCLK1_CLKSEL Register

31	3	2	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-37. CM_SYSCLK1_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.4.2 CM_SYSCLK2_CLKSEL Register

The CM_SYSCLK2_CLKSEL register selects the divider value for SYSCLK2. It is shown and described in the figure and table below.

Figure 18-21. CM_SYSCLK2_CLKSEL Register

31	3	2	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-38. CM_SYSCLK2_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.4.3 CM_SYSCLK3_CLKSEL Register

The CM_SYSCLK3_CLKSEL register selects the divider value for SYSCLK3. It is shown and described in the figure and table below.

Figure 18-22. CM_SYSCLK3_CLKSEL Register

31	3	2	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-39. CM_SYSCLK3_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.4.4 CM_SYSCLK4_CLKSEL Register

The CM_SYSCLK4_CLKSEL register selects the divider value for SYSCLK4. It is shown and described in the figure and table below.

Figure 18-23. CM_SYSCLK4_CLKSEL Register

31	1	0
Reserved		CLKSEL
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-40. CM_SYSCLK4_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1	Select SYS_CLK divided by 2

18.7.4.5 CM_SYSCLK5_CLKSEL Register

The CM_SYSCLK5_CLKSEL register selects the divider value for SYSCLK5. It is shown and described in the figure and table below.

Figure 18-24. CM_SYSCLK5_CLKSEL Register

31	1	0
Reserved		CLKSEL
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-41. CM_SYSCLK5_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1	Select SYS_CLK divided by 2

18.7.4.6 CM_SYSCLK6_CLKSEL Register

The SCM_SYSCLK6_CLKSEL register selects the divider value for SYSCLK6. It is shown and described in the figure and table below.

Figure 18-25. CM_SYSCLK6_CLKSEL Register

31	1	0
Reserved		CLKSEL
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-42. CM_SYSCLK6_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 2
		1	Select SYS_CLK divided by 4

18.7.4.7 CM_SYSCLK7_CLKSEL Register

The CM_SYSCLK7_CLKSEL register selects the divider value for SYSCLK7. It is shown and described in the figure and table below.

Figure 18-26. CM_SYSCLK7_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-43. CM_SYSCLK7_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 5
		1h	Select SYS_CLK divided by 6
		2h	Select SYS_CLK divided by 8
		3h	Select SYS_CLK divided by 16

18.7.4.8 CM_SYSCLK10_CLKSEL Register

The CM_SYSCLK10_CLKSEL register selects the divider value for SYSCLK10. It is shown and described in the figure and table below.

Figure 18-27. CM_SYSCLK10_CLKSEL Register

31	3	2	0
Reserved		CLKSEL	
R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-44. CM_SYSCLK10_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.4.9 CM_SYSCLK11_CLKSEL Register

The CM_SYSCLK11_CLKSEL register selects the divider value for SYSCLK11. It is shown and described in the figure and table below.

Figure 18-28. CM_SYSCLK11_CLKSEL Register

31	1	0
Reserved		CLKSEL
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-45. CM_SYSCLK11_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1	Select SYS_CLK divided by 2

18.7.4.10 CM_SYSCLK13_CLKSEL Register

The SCM_SYSCLK13_CLKSEL register selects the divider value for SYSCLK13. It is shown and described in the figure and table below.

Figure 18-29. CM_SYSCLK13_CLKSEL Register

31	3	2	0
Reserved		CLKSEL	
R-0		R/W-7h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-46. CM_SYSCLK13_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.4.11 CM_SYSCLK15_CLKSEL Register

The CM_SYSCLK15_CLKSEL register selects the divider value for SYSCLK13. It is shown and described in the figure and table below.

Figure 18-30. CM_SYSCLK15_CLKSEL Register

31	3	2	0
Reserved		CLKSEL	
R-0		R/W-3h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-47. CM_SYSCLK15_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.4.12 CM_VPB3_CLKSEL Register

The CM_VPB3_CLKSEL register selects the divider value for video PLL B3 divider. It is shown and described in the figure and table below.

Figure 18-31. CM_VPB3_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-2h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-48. CM_VPB3_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 22
		3h	Reserved

18.7.4.13 CM_VPC1_CLKSEL Register

The CM_VPC1_CLKSEL register selects the divider value for video PLL B1 divider. It is shown and described in the figure and table below.

Figure 18-32. CM_VPC1_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-2h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-49. CM_VPC1_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 22
		3h	Reserved

18.7.4.14 CM_VPD1_CLKSEL Register

The CM_VPD1_CLKSEL register selects the divider value for video PLL D1 divider. It is shown and described in the figure and table below.

Figure 18-33. CM_VPD1_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-7h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-50. CM_VPD1_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.4.15 CM_SYSCLK19_CLKSEL Register

The CM_SYSCLK19_CLKSEL register selects the divider value for SYSCLK19. It is shown and described in the figure and table below.

Figure 18-34. CM_SYSCLK19_CLKSEL Register

31	3	2	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-51. CM_SYSCLK19_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.4.16 CM_SYSCLK20_CLKSEL Register

The CM_SYSCLK20_CLKSEL register selects the divider value for SYSCLK20. It is shown and described in the figure and table below.

Figure 18-35. CM_SYSCLK20_CLKSEL Register

31	3	2	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-52. CM_SYSCLK20_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.4.17 CM_SYSCLK21_CLKSEL Register

The CM_SYSCLK21_CLKSEL Register selects the divider value for SYSCLK21. It is shown and described in the figure and table below.

Figure 18-36. CM_SYSCLK21_CLKSEL Register

31	3	2	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-53. CM_SYSCLK21_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.4.18 CM_SYSCLK22_CLKSEL Register

The CM_SYSCLK22_CLKSEL register selects the divider value for SYSCLK22.

Figure 18-37. CM_SYSCLK22_CLKSEL Register

31	3	2	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-54. CM_SYSCLK22_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.4.19 CM_SYSCLK14_CLKSEL Register

The CM_SYSCLK14_CLKSEL register selects the Mux select line for SYSCLK14 clock. It is shown and described in the figure and table below.

Figure 18-38. CM_SYSCLK14_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-55. CM_SYSCLK14_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the Mux select line for SYSCLK14 [warm reset insensitive]
		0	Select SYSCLK14 to be B3 divider output
		1h	Select SYSCLK14 to be CLKIN
		2h	Select SYSCLK14 to be C1 divider output
		3h	Reserved

18.7.4.20 CM_SYSCLK16_CLKSEL Register

The CM_SYSCLK16_CLKSEL register selects the Mux select line for SYSCLK16 clock. It is shown and described in the figure and table below.

Figure 18-39. CM_SYSCLK16_CLKSEL Register

31	1	0
Reserved		CLKSEL
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-56. CM_SYSCLK16_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	CLKSEL		Selects the Mux select line for SYSCLK16 [warm reset insensitive]
		0	Select SYSCLK16 to be D1 divider output
		1	Select SYSCLK16 to be divider B3 output

18.7.4.21 CM_SYSCLK18_CLKSEL Register

Selects the Mux select line for SYSCLK18 clock. It is shown and described in the figure and table below

Figure 18-40. CM_SYSCLK18_CLKSEL Register

31	1	0
Reserved		CLKSEL
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-57. CM_SYSCLK18_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	CLKSEL		Selects the Mux select line for SYSCLK18 [warm reset insensitive]
		0	Select 32 KHz Clock
		1	Select Audio PLL generated 32 KHz Clock

18.7.4.22 CM_AUDIOCLK_MCASP0_CLKSEL Register

The CM_AUDIOCLK_MCASP0_CLKSEL register selects the Mux select line for McASP0 audio clock. It is shown and described in the figure and table below.

Figure 18-41. CM_AUDIOCLK_MCASP0_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-58. CM_AUDIOCLK_MCASP0_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the Mux select line for McASP0 audio clock [warm reset insensitive]
		0	Select McASP0 audio clock to be SYSCLK20
		1h	Select McASP0 audio clock to be SYSCLK21
		2h	Select McASP0 audio clock to be SYSCLK22
		3h	Reserved

18.7.4.23 CM_AUDIOCLK_MCASP1_CLKSEL Register

The CM_AUDIOCLK_MCASP1_CLKSEL register selects the Mux select line for McASP1 audio clock. It is shown and described in the figure and table below.

Figure 18-42. CM_AUDIOCLK_MCASP1_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-59. CM_AUDIOCLK_MCASP1_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the Mux select line for McASP0 audio clock [warm reset insensitive]
		0	Select McASP1 audio clock to be SYSCLK20
		1h	Select McASP1 audio clock to be SYSCLK21
		2h	Select McASP1 audio clock to be SYSCLK22
		3h	Reserved

18.7.4.24 CM_AUDIOCLK_MCASP2_CLKSEL Register

The CM_AUDIOCLK_MCASP2_CLKSEL register selects the Mux select line for McASP2 audio clock. It is shown and described in the figure and table below.

Figure 18-43. CM_AUDIOCLK_MCASP2_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-60. CM_AUDIOCLK_MCASP2_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the Mux select line for McASP0 audio clock [warm reset insensitive]
		0	Select McASP2 audio clock to be SYSCLK20
		1h	Select McASP2 audio clock to be SYSCLK21
		2h	Select McASP2 audio clock to be SYSCLK22
		3h	Reserved

18.7.4.25 CM_AUDIOCLK_MCBSP_CLKSEL Register

The CM_AUDIOCLK_MCBSP_CLKSEL register selects the Mux select line for McBSP audio clock. It is shown and described in the figure and table below.

Figure 18-44. CM_AUDIOCLK_MCBSP_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-61. CM_AUDIOCLK_MCBSP_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the Mux select line for McASP0 audio clock [warm reset insensitive]
		0	Select McASP2 audio clock to be SYSCLK20
		1h	Select McASP2 audio clock to be SYSCLK21
		2h	Select McASP2 audio clock to be SYSCLK22
		3h	Reserved

18.7.4.26 CM_TIMER1_CLKSEL Register

The CM_TIMER1_CLKSEL register selects the Mux select line for TIMER1 clock. It is shown and described in the figure and table below

Figure 18-45. CM_TIMER1_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-62. CM_TIMER1_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the Mux select line for TIMER1 clock [warm reset insensitive]
		0	Select TIMER clock to be TCLKIN
		1h	Select TIMER clock to be external 32 KHz clock.(After MUX)
		2h	Select TIMER clock to be CLKIN
		3h	Reserved

18.7.4.27 CM_TIMER2_CLKSEL Register

The CM_TIMER2_CLKSEL register selects the Mux select line for TIMER2 clock. It is shown and described in the figure and table below.

Figure 18-46. CM_TIMER2_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-63. CM_TIMER2_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the Mux select line for TIMER2 clock [warm reset insensitive]
		0	Select TIMER clock to be TCLKIN
		1h	Select TIMER clock to be external 32 KHz clock.(After MUX)
		2h	Select TIMER clock to be CLKIN
		3h	Reserved

18.7.4.28 CM_TIMER3_CLKSEL Register

The CM_TIMER3_CLKSEL register selects the Mux select line for TIMER3 clock. It is shown and described in the figure and table below

Figure 18-47. CM_TIMER3_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-64. CM_TIMER3_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the Mux select line for TIMER3 clock [warm reset insensitive]
		0	Select TIMER clock to be TCLKIN
		1h	Select TIMER clock to be external 32 KHz clock.(After MUX)
		2h	Select TIMER clock to be CLKIN
		3h	Reserved

18.7.4.29 CM_TIMER4_CLKSEL Register

The CM_TIMER4_CLKSEL register selects the Mux select line for TIMER4 clock. It is shown and described in the figure and table below

Figure 18-48. CM_TIMER4_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-65. CM_TIMER4_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the Mux select line for TIMER4 clock [warm reset insensitive]
		0	Select TIMER clock to be TCLKIN
		1h	Select TIMER clock to be external 32 KHz clock.(After MUX)
		2h	Select TIMER clock to be CLKIN
		3h	Reserved

18.7.4.30 CM_TIMER5_CLKSEL Register

The CM_TIMER5_CLKSEL register selects the Mux select line for TIMER5 clock. It is shown and described in the figure and table below

Figure 18-49. CM_TIMER5_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-66. CM_TIMER5_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the Mux select line for TIMER5 clock [warm reset insensitive]
		0	Select TIMER clock to be TCLKIN
		1h	Select TIMER clock to be external 32 KHz clock.(After MUX)
		2h	Select TIMER clock to be CLKIN
		3h	Reserved

18.7.4.31 CM_TIMER6_CLKSEL Register

The CM_TIMER6_CLKSEL register selects the Mux select line for TIMER6 clock. It is shown and described in the figure and table below

Figure 18-50. CM_TIMER6_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-67. CM_TIMER6_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the Mux select line for TIMER6 clock [warm reset insensitive]
		0	Select TIMER clock to be TCLKIN
		1h	Select TIMER clock to be external 32 KHz clock.(After MUX)
		2h	Select TIMER clock to be CLKIN
		3h	Reserved

18.7.4.32 CM_TIMER7_CLKSEL Register

The CM_TIMER7_CLKSEL register selects the Mux select line for TIMER7 clock. It is shown and described in the figure and table below

Figure 18-51. CM_TIMER7_CLKSEL Register

31	2	1	0
Reserved		CLKSEL	
R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-68. CM_TIMER7_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKSEL		Selects the Mux select line for TIMER7 clock [warm reset insensitive]
		0	Select TIMER clock to be TCLKIN
		1h	Select TIMER clock to be external 32 KHz clock.(After MUX)
		2h	Select TIMER clock to be CLKIN
		3h	Reserved

18.7.4.33 CM_SYSCLK23_CLKSEL Register

The CM_SYSCLK23_CLKSEL register selects the divider value for SYSCLK23. It is shown and described in the figure and table below.

Figure 18-52. CM_SYSCLK23_CLKSEL Register

31	3	2	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-69. CM_SYSCLK23_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.4.34 CM_SYSCLK24_CLKSEL Register

The CM_SYSCLK24_CLKSEL register selects the divider value for SYSCLK24. It is shown and described in the figure and table below.

Figure 18-53. CM_SYSCLK24_CLKSEL Register

31	3	2	0
Reserved		CLKSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-70. CM_SYSCLK24_CLKSEL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	CLKSEL		Selects the divider value [warm reset insensitive]
		0	Select SYS_CLK divided by 1
		1h	Select SYS_CLK divided by 2
		2h	Select SYS_CLK divided by 3
		3h	Select SYS_CLK divided by 4
		4h	Select SYS_CLK divided by 5
		5h	Select SYS_CLK divided by 6
		6h	Select SYS_CLK divided by 7
		7h	Select SYS_CLK divided by 8

18.7.5 CM_ACTIVE Device

Table 18-71 lists the registers for the CM_ACTIVE device.

Table 18-71. CM_ACTIVE Device Registers

Offset Address	Acronym	Section
400h	CM_GEM_CLKSTCTRL	Section 18.7.5.1
404h	CM_HDDSS_CLKSTCTRL	Section 18.7.5.2
420h	CM_ACTIVE_GEM_CLKCTRL	Section 18.7.5.3
424h	CM_ACTIVE_HDDSS_CLKCTRL	Section 18.7.5.4

18.7.5.1 CM_GEM_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also holds one status bit per clock input of the domain.

Figure 18-54. CM_GEM_CLKSTCTRL Register

31	16
Reserved	
R-0	

15	11	10	9	8
Reserved		CLKACTIVITY_GEM_TRCCLK	CLKACTIVITY_GEM_VBUSPCLK	CLKACTIVITY_GEM_GCLK
R-0		R-0	R-0	R-0

7	2	1	0
Reserved		CLKTRCTRL	
R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 18-72. CM_GEM_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	CLKACTIVITY_GEM_TRCCLK		This field indicates the state of the GEM_TRCCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
9	CLKACTIVITY_GEM_VBUSPCLK		This field indicates the state of the GEM_VBUSP clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
8	CLKACTIVITY_GEM_GCLK		This field indicates the state of the GEM_GICLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved

Table 18-72. CM_GEM_CLKSTCTRL Register Field Descriptions (continued)

Bit	Field	Value	Description
1-0	CLKTRCTRL		Controls the clock state transition of the C674x DSP clock domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.5.2 CM_HDDSS_CLKSTCTRL Register

The CM_HDDSS_CLKSTCTRL register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also holds one status bit per clock input of the domain.

Figure 18-55. CM_HDDSS_CLKSTCTRL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
CLKACTIVITY_HD_DSS_L3_EN_GCLK	CLKACTIVITY_PRC_GCLK	CLKACTIVITY_HD_DSS_L4_GCLK	CLKACTIVITY_HD_DSS_L4_GCLK	Reserved	CLKACTIVITY_SD_GCLK	CLKACTIVITY_HD_VENC_A_GCLK	CLKACTIVITY_HD_VENC_D_GCLK
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
Reserved						CLKTRCTRL	
R-0						R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-73. CM_HDDSS_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	CLKACTIVITY_HD_DSS_L3_EN_GCLK		This field indicates the state of the L3_EN_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
14	CLKACTIVITY_PRC_GCLK		This field indicates the state of the PRC_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
13	CLKACTIVITY_HD_DSS_L4_GCLK		This field indicates the state of the HD_DSS_L4_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
12	CLKACTIVITY_HD_DSS_L4_GCLK		This field indicates the state of the HD_DSS_L3_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
11	Reserved	0	Reserved
10	CLKACTIVITY_SD_GCLK		This field indicates the state of the SD_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
9	CLKACTIVITY_HD_VENC_A_GCLK		This field indicates the state of the HD_VENC_A_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active

Table 18-73. CM_HDDSS_CLKSTCTRL Register Field Descriptions (continued)

Bit	Field	Value	Description
8	CLKACTIVITY_HD_VENC_D_GCLK		This field indicates the state of the HD_VENC_D_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the HD-DSS clock domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.5.3 CM_ACTIVE_GEM_CLKCTRL Register

The CM_ACTIVE_GEM_CLKCTRL register manages the C674x DSP clocks. It is shown and described in the figure and table below.

Figure 18-56. CM_ACTIVE_GEM_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-74. CM_ACTIVE_GEM_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.5.4 CM_ACTIVE_HDDSS_CLKCTRL Register

The CM_ACTIVE_HDDSS_CLKCTRL register manages the HD_DSS clocks. It is shown and described in the figure and table below.

Figure 18-57. CM_ACTIVE_HDDSS_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-75. CM_ACTIVE_HDDSS_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.6 CM_DEFAULT Device

Table 18-76 lists the registers for the CM_DEFAULT device.

Table 18-76. CM_DEFAULT Device Registers

Offset Address	Acronym	Section
504h	CM_DEFAULT_L3_MED_CLKSTCTRL	Section 18.7.6.1
508h	CM_DEFAULT_L3_FAST_CLKSTCTRL	Section 18.7.6.2
510h	CM_DEFAULT_PCI_CLKSTCTRL	Section 18.7.6.3
514h	CM_DEFAULT_L3_SLOW_CLKSTCTRL	Section 18.7.6.4
518h	CM_DEFAULT_CLKSTCTRL	Section 18.7.6.5
520h	CM_DEFAULT_EMIF_0_CLKCTRL	Section 18.7.6.6
524h	CM_DEFAULT_EMIF_1_CLKCTRL	Section 18.7.6.7
528h	CM_DEFAULT_DMM_CLKCTRL	Section 18.7.6.8
52Ch	CM_DEFAULT_FW_CLKCTRL	Section 18.7.6.9
558h	CM_DEFAULT_USB_CLKCTRL	Section 18.7.6.10
560h	CM_DEFAULT_SATA_CLKCTRL	Section 18.7.6.11
574h	CM_DEFAULT_CLKCTRL	Section 18.7.6.12
578h	CM_DEFAULT_PCI_CLKCTRL	Section 18.7.6.13

18.7.6.1 CM_DEFAULT_L3_MED_CLKSTCTRL Register

The CM_DEFAULT_L3_MED_CLKSTCTRL register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also holds one status bit per clock input of the domain.

Figure 18-58. CM_DEFAULT_L3_MED_CLKSTCTRL Register

31	16
Reserved	
R-0	

15	9	8	7	2	1	0
Reserved		CLKACTIVITY_L3_MED_GCLK	Reserved		CLKTRCTRL	
R-0		R-0	R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-77. CM_DEFAULT_L3_MED_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_L3_MED_GCLK		This field indicates the state of the L3_MED_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL	0-3h	Controls the clock state transition of the L3_MED clock domain in DEFAULT power domain.

18.7.6.2 CM_DEFAULT_L3_FAST_CLKSTCTRL Register

The CM_DEFAULT_L3_FAST_CLKSTCTRL register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also holds one status bit per clock input of the domain.

Figure 18-59. CM_DEFAULT_L3_FAST_CLKSTCTRL Register

31	16
Reserved	
R-0	

15	10	9	8	7	2	1	0
Reserved		CLKACTIVITY_DDR_GCLK	CLKACTIVITY_L3_FAST_GCLK	Reserved		CLKTRCTRL	
R-0		R-0	R-0	R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-78. CM_DEFAULT_L3_FAST_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	CLKACTIVITY_DDR_GCLK		This field indicates the state of the DDR_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
8	CLKACTIVITY_L3_FAST_GCLK		This field indicates the state of the L3_FAST_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the L3_FAST clock domain in DEFAULT power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.6.3 CM_DEFAULT_PCI_CLKSTCTRL Register

The CM_DEFAULT_PCI_CLKSTCTRL register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also holds one status bit per clock input of the domain.

Figure 18-60. CM_DEFAULT_PCI_CLKSTCTRL Register

31 9	8	7 2	1 0
Reserved	CLKACTIVITY_PCI_GCLK	Reserved	CLKTRCTRL
R-0	R-0	R-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 18-79. CM_DEFAULT_PCI_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_PCI_GCLK		This field indicates the state of the PCI_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the PCI clock domain in DEFAULT power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.6.4 CM_DEFAULT_L3_SLOW_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also holds one status bit per clock input of the domain.

Figure 18-61. CM_DEFAULT_L3_SLOW_CLKSTCTRL Register

31 9	8	7 2	1 0
Reserved	CLKACTIVITY_USB_GCLK	Reserved	CLKTRCTRL
R-0	R-0	R-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

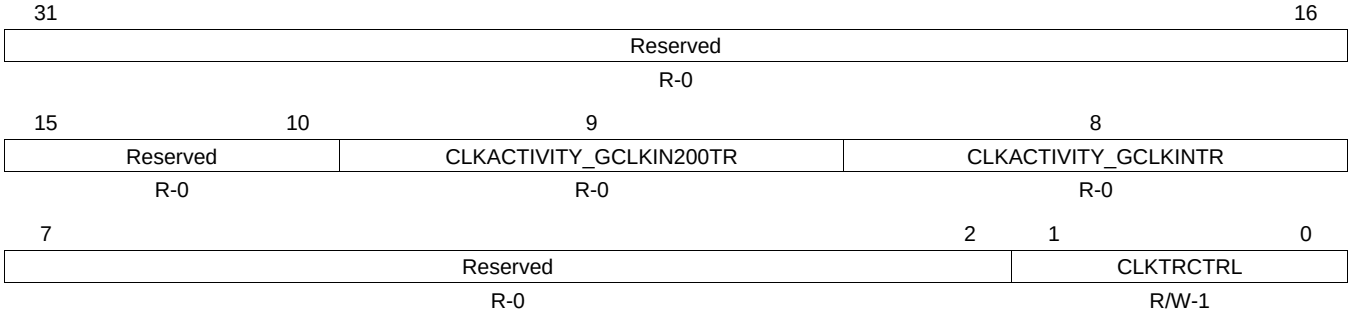
Table 18-80. CM_DEFAULT_L3_SLOW_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_USB_GCLK		This field indicates the state of the L3_SLOW_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the L3_SLOW clock domain in DEFAULT power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.6.5 CM_DEFAULT_CLKSTCTRL Register

The CM_DEFAULT_CLKSTCTRL register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also holds one status bit per clock input of the domain.

Figure 18-62. CM_DEFAULT_CLKSTCTRL Register



LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 18-81. CM_DEFAULT_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	CLKACTIVITY_GCLKIN200TR		This field indicates the state of the CLKIN200TR clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
8	CLKACTIVITY_GCLKINTR		This field indicates the state of the CLKINTR clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the clock domain in DEFAULT power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.6.6 CM_DEFAULT_EMIF_0_CLKCTRL Register

The CM_DEFAULT_EMIF_0_CLKCTRL register manages the EMIF_0 clocks. It is shown and described in the figure and table below.

Figure 18-63. CM_DEFAULT_EMIF_0_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-82. CM_DEFAULT_EMIF_0_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.6.7 CM_DEFAULT_EMIF_1_CLKCTRL Register

The CM_DEFAULT_EMIF_1_CLKCTRL register manages the EMIF_1 clocks. It is shown and described in the figure and table below.

Figure 18-64. CM_DEFAULT_EMIF_1_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-83. CM_DEFAULT_EMIF_1_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.6.8 CM_DEFAULT_DMM_CLKCTRL Register

Th CM_DEFAULT_DMM_CLKCTRL register manages the DMM clocks. It is shown and described in the figure and table below.

Figure 18-65. CM_DEFAULT_DMM_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-84. CM_DEFAULT_DMM_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.6.9 CM_DEFAULT_FW_CLKCTRL Register

The CM_DEFAULT_FW_CLKCTRL register manages the EMIF FW clocks. It is shown and described in the figure and table below.

Figure 18-66. CM_DEFAULT_FW_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-85. CM_DEFAULT_FW_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.6.10 CM_DEFAULT_USB_CLKCTRL Register

The CM_DEFAULT_USB_CLKCTRL register manages the USB clocks. It is shown and described in the figure and table below.

Figure 18-67. CM_DEFAULT_USB_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-86. CM_DEFAULT_USB_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.6.11 CM_DEFAULT_SATA_CLKCTRL Register

The CM_DEFAULT_SATA_CLKCTRL register manages the SATA clocks. It is shown and described in the figure and table below.

Figure 18-68. CM_DEFAULT_SATA_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-87. CM_DEFAULT_SATA_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.6.12 CM_DEFAULT_CLKCTRL Register

The CM_DEFAULT_CLKCTRL register manages the clocks. It is shown and described in the figure and table below.

Figure 18-69. CM_DEFAULT_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-88. CM_DEFAULT_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.6.13 CM_DEFAULT_PCI_CLKCTRL Register

The CM_DEFAULT_PCI_CLKCTRL register manages the PCI clocks. It is shown and described in the figure and table below.

Figure 18-70. CM_DEFAULT_PCI_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-89. CM_DEFAULT_PCI_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.7 CM_IVAHD0 Device

Table 18-90 lists the registers for the CM_IVAHD0 device.

Table 18-90. CM_IVAHD0 Device Registers

Offset Address	Acronym	Section
600h	CM_IVAHD0_CLKSTCTRL	Section 18.7.7.1
620h	CM_IVAHD0_IVAHD_CLKCTRL	Section 18.7.7.2
624h	CM_IVAHD0_SL2_CLKCTRL	Section 18.7.7.3

18.7.7.1 CM_IVAHD0_CLKSTCTRL Register

The CM_IVAHD0_CLKSTCTRL register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also holds one status bit per clock input of the domain.

Figure 18-71. CM_IVAHD0_CLKSTCTRL Register

31	16
Reserved	
R-0	

15	9	8	7	2	1	0
Reserved		CLKACTIVITY_IVAHD0_GCLK	Reserved		CLKTRCTRL	
R-0		R-0	R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-91. CM_IVAHD0_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_IVAHD0_GCLK		This field indicates the state of the IVAHD0_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the HDVICP2-0 clock domain in HDVICP2-0 power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.7.2 CM_IVAHD0_IVAHD_CLKCTRL Register

The CM_IVAHD0_IVAHD_CLKCTRL register manages the HDVICP2-0 clocks. It is shown and described in the figure and table below.

Figure 18-72. CM_IVAHD0_IVAHD_CLKCTRL Register

31	19	18	17	16
Reserved		STBYST	IDLEST	
R-0		R-1	R-3h	

15	2	1	0
Reserved		MODULEMODE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-92. CM_IVAHD0_IVAHD_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.7.3 CM_IVAHD0_SL2_CLKCTRL Register

The M_IVAHD0_SL2_CLKCTRL register manages the SL2 clocks. It is shown and described in the figure and table below.

Figure 18-73. CM_IVAHD0_SL2_CLKCTRL Register

31	18	17	16
Reserved		IDLEST	
R-0		R-3h	

15	2	1	0
Reserved		MODULEMODE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-93. CM_IVAHD0_SL2_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.8 CM_IVAHD1 Device

Table 18-94 lists the registers for the CM_IVAHD1 device.

Table 18-94. CM_IVAHD1 Device Registers

Offset Address	Acronym	Section
700h	CM_IVAHD1_CLKSTCTRL	Section 18.7.8.1
720h	CM_IVAHD1_IVAHD_CLKCTRL	Section 18.7.8.2
724h	CM_IVAHD1_SL2_CLKCTRL	Section 18.7.8.3

18.7.8.1 CM_IVAHD1_CLKSTCTRL Register

The CM_IVAHD1_CLKSTCTRL register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also holds one status bit per clock input of the domain.

Figure 18-74. CM_IVAHD1_CLKSTCTRL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CLKACTIVITY_IVAHD1_GCLK	Reserved						CLKTRCTRL	
R-0							R-0	R-0						R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-95. CM_IVAHD1_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_IVAHD1_GCLK		This field indicates the state of the HDVICP2-1_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the HDVICP2-1 clock domain in IVAHD1 power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.8.2 CM_IVAHD1_IVAHD_CLKCTRL Register

The CM_IVAHD1_IVAHD_CLKCTRL register manages the HDVICP2-1 clocks. It is shown and described in the following figure and table.

Figure 18-75. CM_IVAHD1_IVAHD_CLKCTRL Register

31	19	18	17	16
Reserved		STBYST	IDLEST	
R-0		R-1	R-3h	

15	2	1	0
Reserved		MODULEMODE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-96. CM_IVAHD1_IVAHD_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.8.3 CM_IVAHD1_SL2_CLKCTRL Register

The CM_IVAHD1_SL2_CLKCTRL register manages the SL2 clocks. It is shown and described in the figure and table below

Figure 18-76. CM_IVAHD1_SL2_CLKCTRL Register

31	18	17	16
Reserved		IDLEST	
R-0		R-3h	

15	2	1	0
Reserved		MODULEMODE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-97. CM_IVAHD0_SL2_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.9 CM_IVAHD2 Device

Table 18-98 lists the registers for the CM_IVAHD2 device.

Table 18-98. CM_IVAHD2 Device Registers

Offset Address	Acronym	Section
800h	CM_IVAHD2_CLKSTCTRL	Section 18.7.9.1
820h	CM_IVAHD2_IVAHD_CLKCTRL	Section 18.7.9.2
824h	CM_IVAHD2_SL2_CLKCTRL	Section 18.7.9.3

18.7.9.1 CM_IVAHD2_CLKSTCTRL Register

The CM_IVAHD2_CLKSTCTRL register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also holds one status bit per clock input of the domain.

Figure 18-77. CM_IVAHD2_CLKSTCTRL Register

31	16
Reserved	
R-0	

15	9	8	7	2	1	0
Reserved		CLKACTIVITY_IVAHD2_GCLK	Reserved		CLKTRCTRL	
R-0		R-0	R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-99. CM_IVAHD2_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_IVAHD2_GCLK		This field indicates the state of the IVAHD2_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the HDVICP2-2 clock domain in HDVICP2-2 power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.9.2 CM_IVAHD2_IVAHD_CLKCTRL Register

The CM_IVAHD2_IVAHD_CLKCTRL register manages the HDVICP2-2 clocks.

Figure 18-78. CM_IVAHD2_IVAHD_CLKCTRL Register

31	19	18	17	16
Reserved		STBYST	IDLEST	
R-0		R-1	R-3h	

15	2	1	0
Reserved		MODULEMODE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-100. CM_IVAHD2_IVAHD_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.9.3 CM_IVAHD2_SL2_CLKCTRL Register

The CM_IVAHD2_SL2_CLKCTRL register manages the SL2 clocks. It is shown and described in the figure and table below.

Figure 18-79. CM_IVAHD2_SL2_CLKCTRL Register

31	18	17	16
Reserved		IDLEST	
R-0		R-3h	

15	2	1	0
Reserved		MODULEMODE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-101. CM_IVAHD2_SL2_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.10 CM_SGX Device

Table 18-102 lists the registers for the CM_SGX device.

Table 18-102. CM_SGX Device Registers

Offset Address	Acronym	Section
900h	CM_SGX_CLKSTCTRL	Section 18.7.10.1
920h	CM_SGX_SGX_CLKCTRL	Section 18.7.10.2

18.7.10.1 CM_SGX_CLKSTCTRL Register

The CM_SGX_CLKSTCTRL register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also holds one status bit per clock input of the domain.

Figure 18-80. CM_SGX_CLKSTCTRL Register

31 9	8	7 2	1 0
Reserved	CLKACTIVITY_SGX_GCLK	Reserved	CLKTRCTRL
R-0	R-0	R-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-103. CM_SGX_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_SGX_GCLK		This field indicates the state of the SGX_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the SGX clock domain in SGX power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.10.2 CM_SGX_SGX_CLKCTRL Register

The CM_SGX_SGX_CLKCTRL register manages the SGX clocks. It is shown and described in the figure and table below.

Figure 18-81. CM_SGX_SGX_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-104. CM_SGX_SGX_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST	0-1	Module standby status.
17-16	IDLEST		Module idle status
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock.
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.11 PRM_ACTIVE Device

Table 18-105 lists the registers for the PRM_ACTIVE device.

Table 18-105. PRM_ACTIVE Device Registers

Offset Address	Acronym	Section
A00h	PM_ACTIVE_PWRSTCTRL	Section 18.7.11.1
A04h	PM_ACTIVE_PWRSTST	Section 18.7.11.2
A10h	RM_ACTIVE_RSTCTRL	Section 18.7.11.3
A14h	RM_ACTIVE_RSTST	Section 18.7.11.4

18.7.11.1 PM_ACTIVE_PWRSTCTRL Register

This register controls the ACTIVE power state to reach upon a domain sleep transition [warm reset insensitive]. It is shown and described in the figure and table below.

Figure 18-82. PM_ACTIVE_PWRSTCTRL Register

31	18	17	16
Reserved		ACTIVE_MEM_ONSTATE	
R-0		R-3h	

15	2	1	0
Reserved		POWERSTATE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-106. PM_ACTIVE_PWRSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	ACTIVE_MEM_ONSTATE		Active domain memory state when domain is ON
		0	Reserved
		1h	Reserved
		2h	Reserved
		3h	Memory bank is on when the domain is ON
15-2	Reserved	0	Reserved
1-0	POWERSTATE		Power state control
		0	OFF State [warm reset insensitive]
		1h	Reserved
		2h	Reserved
		3h	ON State [warm reset insensitive]

18.7.11.2 PM_ACTIVE_PWRSTST Register

The PM_ACTIVE_PWRSTST register provides a status on the current ACTIVE power domain state. [warm reset insensitive]. It is shown and described in the figure and table below.

Figure 18-83. PM_ACTIVE_PWRSTST Register

31	21	20	19	3	2	1	0
Reserved		INTRANSITION	Reserved		LOGICSTATEST	POWERSTATEST	
R-0		R-0	R-0		R-1	R-3h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-107. PM_ACTIVE_PWRSTST Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	INTRANSITION		Domain transition status
		0	No on-going transition on power domain
		1	Power domain transition is in progress
19-3	Reserved	0	Reserved
2	LOGICSTATEST		Logic state status
		0	Logic in domain is OFF
		1	Logic in domain is ON
1-0	POWERSTATEST		Current power state status
		0	OFF State
		1h	Reserved
		2h	Reserved
		3h	ON State

18.7.11.3 RM_ACTIVE_RSTCTRL Register

The RM_ACTIVE_RSTCTRL register controls the release of the ACTIVE domain resets. It is shown and described in the figure and table below.

Figure 18-84. RM_ACTIVE_RSTCTRL Register

31	2	1	0
Reserved		GEM_SW_RST	GEM_LRST
R-0		R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-108. RM_ACTIVE_RSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	GEM_SW_RST		ACTIVE domain C674x DSP warm reset control
		0	Reset is cleared for the active domain C674x DSP warm reset
		1	Reset is asserted for the active domain C674x DSP warm reset
0	GEM_LRST		ACTIVE domain C674x DSP local reset control
		0	Reset is cleared for the active domain C674x DSP local reset
		1	Reset is asserted for the active domain C674x DSP local reset

18.7.11.4 RM_ACTIVE_RSTST Register

The RM_ACTIVE_RSTST register logs the different reset sources of the ACTIVE domain. Each bit is set upon release of the domain reset signal. Must be cleared by software. [warm reset insensitive]. It is shown and described in the figure and table below.

Figure 18-85. RM_ACTIVE_RSTST Register

31	2	1	0
Reserved		GEM_GRST	GEM_LRST
R-0		R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-109. RM_ACTIVE_RSTST Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	GEM_GRST		C674x DSP warm reset
		0	No software reset occurred
		1	C674x DSP warm reset has been asserted
0	GEM_LRST		C674x DSP local software reset
		0	No software reset occurred
		1	C674x DSP local reset has been asserted

18.7.12 PRM_DEFAULT Device

Table 18-110 lists the registers for the PRM_DEFAULT device.

Table 18-110. PRM_DEFAULT Device Registers

Offset Address	Acronym	Section
B00h	PM_DEFAULT_PWRSTCTRL	Section 18.7.12.1
B04h	PM_DEFAULT_PWRSTST	Section 18.7.12.2
B10h	RM_DEFAULT_RSTCTRL	Section 18.7.12.3
B14h	RM_DEFAULT_RSTST	Section 18.7.12.4

18.7.12.1 PM_DEFAULT_PWRSTCTRL Register

The PM_DEFAULT_PWRSTCTRL register controls the DEFAULT power state to reach upon a domain sleep transition [warm reset insensitive]. It is shown and described in the figure and table below.

Figure 18-86. PM_DEFAULT_PWRSTCTRL Register

31 18	17 16	15 2	1 0
Reserved	DEFAULT_MEM_ONSTATE	Reserved	POWERSTATE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 18-111. PM_DEFAULT_PWRSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	DEFAULT_MEM_ONSTATE		Default power domain memory state when domain is ON
		0	Reserved
		1h	Reserved
		2h	Reserved
		3h	Memory bank is on when the domain is ON
15-2	Reserved	0	Reserved
1-0	POWERSTATE		Power state control
		0	OFF State [warm reset insensitive]
		1h	Reserved
		2h	Reserved
		3h	ON State [warm reset insensitive]

18.7.12.2 PM_DEFAULT_PWRSTST Register

The PM_DEFAULT_PWRSTST register provides a status on the current DEFAULT power domain state. [warm reset insensitive]. It is shown and described in the figure and table below.

Figure 18-87. PM_DEFAULT_PWRSTST Register

31	21	20	19	3	2	1	0
Reserved		INTRANSITION	Reserved		LOGICSTATEST	POWERSTATEST	
R-0		R-0	R-0		R-1	R-3h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-112. PM_DEFAULT_PWRSTST Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	INTRANSITION		Domain transition status
		0	No on-going transition on power domain
		1	Power domain transition is in progress
19-3	Reserved	0	Reserved
2	LOGICSTATEST		Logic state status
		0	Logic in domain is OFF
		1	Logic in domain is ON
1-0	POWERSTATEST		Current power state status
		0	OFF State
		1h	Reserved
		2h	Reserved
		3h	ON State

18.7.12.3 RM_DEFAULT_RSTCTRL Register

This register controls the release of the DEFAULT subsystem resets. It is shown and described in the figure and table below.

Figure 18-88. RM_DEFAULT_RSTCTRL Register

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
PCI_LRST	Reserved	Reserved	RST3	M3_RST2	M3_RST1	Reserved	
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-3h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 18-113. RM_DEFAULT_RSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	PCI_LRST		ACTIVE domain PCI local reset control
		0	Reset is cleared for PCIe
		1	Reset is asserted for PCIe
6-5	Reserved	3h	Reserved. Always write the default value for future device compatibility.
4	RST3		Logic and MMU reset control
		0	Reset is cleared for the logic and MMU
		1	Reset is asserted for the logic and MMU
3	M3_RST2		Second M3 reset control
		0	Reset is cleared for the ALWON sequencer CPU2
		1	Reset is asserted for the ALWON sequencer CPU2
2	M3_RST1		First M3 reset control
		0	Reset is cleared for the ALWON sequencer CPU1
		1	Reset is asserted for the ALWON sequencer CPU1
1-0	Reserved	3h	Reserved. Always write the default value for future device compatibility.

18.7.12.4 RM_DEFAULT_RSTST Register

The RM_DEFAULT_RSTST register logs the different reset sources of the DEFAULT domain. Each bit is set upon release of the domain reset signal. Must be cleared by software. [warm reset insensitive].

Figure 18-89. RM_DEFAULT_RSTST Register

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
PCI_LRST	Reserved	Reserved	RST3	M3_RST2	M3_RST1	Reserved	
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 18-114. RM_DEFAULT_RSTST Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	PCI_LRST		PCI local software reset
		0	No software reset occurred
		1	PCI has been reset upon software reset
6-5	Reserved	0	Reserved. Always write the default value for future device compatibility.
4	RST3		Logic and MMU software reset
		0	No software reset occurred
		1	Logic and MMU has been reset upon software reset
3	M3_RST2		Second M3 software reset
		0	No software reset occurred
		1	M3_2 has been reset upon software reset
2	M3_RST1		First M3 software reset
		0	No software reset occurred
		1	M3_1 has been reset upon software reset
1-0	Reserved	0	Reserved. Always write the default value for future device compatibility.

18.7.13 PRM_IVAHD0 Device

Table 18-115 lists the registers for the PRM_IVAHD0 device.

Table 18-115. PRM_IVAHD0 Device Registers

Offset Address	Acronym	Section
C00h	PM_IVAHD0_PWRSTCTRL	Section 18.7.13.1
C04h	PM_IVAHD0_PWRSTST	Section 18.7.13.2
C10h	RM_IVAHD0_RSTCTRL	Section 18.7.13.3
C14h	RM_IVAHD0_RSTST	Section 18.7.13.4

18.7.13.1 PM_IVAHD0_PWRSTCTRL Register

This register controls the HDVICP2-0 power state to reach upon a domain sleep transition [warm reset insensitive]. It is shown and described in the figure and table below.

Figure 18-90. PM_IVAHD0_PWRSTCTRL Register

31	18	17	16
Reserved		IVA0_MEM_ONSTATE	
R-0		R-3h	

15	2	1	0
Reserved		POWERSTATE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-116. PM_IVAHD0_PWRSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IVA0_MEM_ONSTATE		HDVICP2-0 memory state when domain is ON.
		0	Reserved
		1h	Reserved
		2h	Reserved
		3h	Memory bank is on when the domain is ON.
15-2	Reserved	0	Reserved
1-0	POWERSTATE		Power state control
		0	OFF State [warm reset insensitive]
		1h	Reserved
		2h	Reserved
		3h	ON State [warm reset insensitive]

18.7.13.2 PM_IVAHD0_PWRSTST Register

This register provides a status on the current HDVICP2-0 power domain state [warm reset insensitive]. It is shown and described in the figure and table below.

Figure 18-91. PM_IVAHD0_PWRSTST Register

31	21	20	19	16
Reserved		INTRANSITION	Reserved	
R-0		R-0	R-0	

15	6	5	4	3	2	1	0
Reserved		IVA0_MEM_STATEST		Rsvd	LOGICSTATEST	POWERSTATE	
R-0		R-1		R-0	R-1	R-3h	

LEGEND: R = Read only; -*n* = value after reset

Table 18-117. PM_IVAHD0_PWRSTST Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	INTRANSITION		Domain transition status
		0	No on-going transition on power domain.
		1	Power domain transition is in progress.
19-6	Reserved	0	Reserved
5-4	IVA0_MEM_STATEST		HDVICP2-0 memory state status
		0	Memory is off.
		1h	Reserved
		2h	Reserved
		3h	Memory is on.
3	Reserved	0	Reserved
2	LOGICSTATEST		Logic state status
		0	Logic in domain is off.
		1h	Reserved
		2h	Reserved
		3h	Logic in domain is on.
1-0	POWERSTATEST		Current Power State Status
		0	Off state [warm reset insensitive]
		1h	Reserved
		2h	Reserved
		3h	On state [warm reset insensitive]

18.7.13.3 RM_IVAHD0_RSTCTRL Register

This register controls the release of the HDVICP2-0 resets. It is shown and described in the figure and table below.

Figure 18-92. RM_IVAHD0_RSTCTRL Register

31	3	2	1	0
Reserved		IVA0_RST3	IVA0_RST2	IVA0_RST1
R-0		R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-118. RM_IVAHD0_RSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	IVA0_RST3		HDVICP2-0 logic and SL2 reset control
		0	Reset is cleared for the HDVICP2-0 logic and SL2
		1	Reset is asserted for HDVICP2-0 logic and SL2
1	IVA0_RST2		HDVICP2-0 sequencer2 reset control
		0	Reset is cleared for the HDVICP2-0 logic and SL2
		1	Reset is asserted for HDVICP2-0 logic and SL2
0	IVA0_RST1		HDVICP2-0 sequencer1 reset control
		0	Reset is cleared for the HDVICP2-0 logic and SL2
		1	Reset is asserted for HDVICP2-0 logic and SL2

18.7.13.4 RM_IVAHD0_RSTST Register

This register logs the different reset sources of the HDVICP2-0 domain. Each bit is set upon release of the domain reset signal. Must be cleared by software. [warm reset insensitive] It is shown and described in the figure and table below.

Figure 18-93. RM_IVAHD0_RSTST Register

31	3	2	1	0
Reserved		IVA0_RST3	IVA0_RST2	IVA0_RST1
R-0		R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 18-119. RM_IVAHD0_RSTST Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	IVA0_RST3		HDVICP2-0 logic and SL2 software reset
		0	No software reset occurred
		1	HDVICP2-0 logic and SL2 has been reset upon software reset
1	IVA0_RST2		HDVICP2-0 Sequencer2 CPU software reset
		0	No software reset occurred
		1	HDVICP2-0 logic and SL2 has been reset upon software reset
0	IVA0_RST1		HDVICP2-0 Sequencer1 CPU software reset
		0	No software reset occurred
		1	HDVICP2-0 logic and SL2 has been reset upon software reset

18.7.14 PRM_IVAHD1 Device

Table 18-120 lists the registers for the PRM_IVAHD1 device.

Table 18-120. PRM_IVAHD1 Device Registers

Offset Address	Acronym	Section
D00h	PM_IVAHD1_PWRSTCTRL	Section 18.7.14.1
D04h	PM_IVAHD1_PWRSTST	Section 18.7.14.2
D10h	RM_IVAHD1_RSTCTRL	Section 18.7.14.3
D14h	RM_IVAHD1_RSTST	Section 18.7.14.4

18.7.14.1 PM_IVAHD1_PWRSTCTRL Register

This register controls the HDVICP2-1 power state to reach upon a domain sleep transition [warm reset insensitive].

Figure 18-94. PM_IVAHD1_PWRSTCTRL Register

31	18	17	16
Reserved		IVA1_MEM_ONSTATE	
R-0		R-3h	

15	2	1	0
Reserved		POWERSTATE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-121. PM_IVAHD1_PWRSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IVA1_MEM_ONSTATE		HDVICP2-1 memory state when domain is ON.
		0	Reserved
		1h	Reserved
		2h	Reserved
		3h	Memory bank is on when the domain is ON.
15-2	Reserved	0	Reserved
1-0	POWERSTATE		Power state control
		0	OFF State [warm reset insensitive]
		1h	Reserved
		2h	Reserved
		3h	ON State [warm reset insensitive]

18.7.14.2 PM_IVAHD1_PWRSTST Register

This register provides a status on the current HDVICP2-1 power domain state. [warm reset insensitive].

Figure 18-95. PM_IVAHD1_PWRSTST Register

31	21	20	19	16
Reserved		INTRANSITION	Reserved	
R-0		R-0	R-0	

15	3	2	1	0
Reserved		LOGICSTATEST	POWERSTATE	
R-0		R-1	R-3h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-122. PM_IVAHD1_PWRSTST Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	INTRANSITION		Domain transition status
		0	No on-going transition on power domain
		1	Power domain transition is in progress
19-3	Reserved	0	Reserved
2	LOGICSTATEST		Logic state status
		0	OFF State
		1h	Reserved
		2h	Reserved
		3h	ON State
1-0	POWERSTATEST		Current Power State Status
		0	OFF State
		1h	Reserved
		2h	Reserved
		3h	ON State

18.7.14.3 RM_IVAHD1_RSTCTRL Register

This register controls the release of the HDVICP2-1 resets.

Figure 18-96. RM_IVAHD1_RSTCTRL Register

31	3	2	1	0
Reserved		IVA1_RST3	IVA1_RST2	IVA1_RST1
R-0		R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-123. RM_IVAHD1_RSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	IVA1_RST3		HDVICP2-1 logic and SL2 reset control
		0	Reset is cleared for the HDVICP2-1 logic and SL2
		1	Reset is asserted for HDVICP2-1 logic and SL2
1	IVA1_RST2		HDVICP2-1 sequencer2 reset control
		0	Reset is cleared for the HDVICP2-1 logic and SL2
		1	Reset is asserted for HDVICP2-1 logic and SL2
0	IVA1_RST1		IVAHD1 sequencer1 reset control
		0	Reset is cleared for the HDVICP2-1 logic and SL2
		1	Reset is asserted for HDVICP2-1 logic and SL2

18.7.14.4 RM_IVAHD1_RSTST Register

This register logs the different reset sources of the HDVICP2-1 domain. Each bit is set upon release of the domain reset signal. Must be cleared by software. [warm reset insensitive]

Figure 18-97. RM_IVAHD1_RSTST Register

31	3	2	1	0
Reserved		IVA1_RST3	IVA1_RST2	IVA1_RST1
R-0		R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 18-124. RM_IVAHD1_RSTST Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	IVA1_RST3		HDVICP2-1 logic and SL2 software reset
		0	No software reset occurred
		1	HDVICP2-1 logic and SL2 has been reset upon software reset
1	IVA1_RST2		HDVICP2-1 Sequencer2 CPU software reset
		0	No software reset occurred
		1	HDVICP2-1 logic and SL2 has been reset upon software reset
0	IVA1_RST1		HDVICP2-1 Sequencer2 CPU software reset
		0	No software reset occurred
		1	HDVICP2-1 logic and SL2 has been reset upon software reset

18.7.15 PRM_IVAHD2 Device

Table 18-125 lists the registers for the PRM_IVAHD2 device.

Table 18-125. PRM_IVAHD2 Device Registers

Offset Address	Acronym	Section
E00h	PM_IVAHD2_PWRSTCTRL	Section 18.7.15.1
E04h	PM_IVAHD2_PWRSTST	Section 18.7.15.2
E10h	RM_IVAHD2_RSTCTRL	Section 18.7.15.3
E14h	RM_IVAHD2_RSTST	Section 18.7.15.4

18.7.15.1 PM_IVAHD2_PWRSTCTRL Register

This register controls the HDVICP2-2 power state to reach upon a domain sleep transition [warm reset insensitive].

Figure 18-98. PM_IVAHD2_PWRSTCTRL Register

31	18	17	16
Reserved		IVA2_MEM_ONSTATE	
R-0		R-3h	

15	2	1	0
Reserved		POWERSTATE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-126. PM_IVAHD2_PWRSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IVA2_MEM_ONSTATE		HDVICP2-2 memory state when domain is ON
		0	Reserved
		1h	Reserved
		2h	Reserved
		3h	Power domain transition is in progress
15-2	Reserved	0	Reserved
1-0	POWERSTATE		Power state control
		0	OFF State [warm reset insensitive]
		1h	Reserved
		2h	Reserved
		3h	ON State [warm reset insensitive]

18.7.15.2 PM_IVAHD2_PWRSTST Register

This register provides a status on the current HDVICP2-2 power domain state. [warm reset insensitive]

Figure 18-99. PM_IVAHD2_PWRSTST Register

31	21	20	19	16
Reserved		INTRANSITION	Reserved	
R-0		R-0	R-0	

15	3	2	1	0
Reserved		LOGICSTATEST	POWERSTATE	
R-0		R-1	R-3h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-127. PM_IVAHD2_PWRSTST Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	INTRANSITION		Domain transition status
		0	No on-going transition on power domain
		1	Power domain transition is in progress
19-3	Reserved	0	Reserved
2	LOGICSTATEST		Logic state status
		0	Logic in domain is OFF
		1	Logic in domain is ON
1-0	POWERSTATEST		Current Power State Status
		0	OFF State
		1	Reserved

18.7.15.3 RM_IVAHD2_RSTCTRL Register

This register controls the release of the HDVICP2-2 resets.

Figure 18-100. RM_IVAHD2_RSTCTRL Register

31	3	2	1	0
Reserved		IVA2_RST3	IVA2_RST2	IVA2_RST1
R-0		R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-128. RM_IVAHD2_RSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	IVA2_RST3		HDVICP2-2 logic and SL2 reset control
		0	Reset is cleared for the HDVICP2-2 logic and SL2
		1	Reset is asserted for HDVICP2-2 logic and SL2
1	IVA2_RST2		HDVICP2-2 sequencer2 reset control
		0	Reset is cleared for the HDVICP2-2 logic and SL2
		1	Reset is asserted for HDVICP2-2 logic and SL2
0	IVA2_RST1		HDVICP2-2 sequencer1 reset control
		0	Reset is cleared for the HDVICP2-2 logic and SL2
		1	Reset is asserted for HDVICP2-2 logic and SL2

18.7.15.4 RM_IVAHD2_RSTST Register

This register logs the different reset sources of the HDVICP2-2 domain. Each bit is set upon release of the domain reset signal. Must be cleared by software. [warm reset insensitive].

Figure 18-101. RM_IVAHD2_RSTST Register

31	3	2	1	0
Reserved		IVA2_RST3	IVA2_RST2	IVA2_RST1
R-0		R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 18-129. RM_IVAHD2_RSTST Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	IVA2_RST3		HDVICP2-2 logic and SL2 software reset
		0	No software reset occurred
		1	HDVICP2-2 logic and SL2 has been reset upon software reset
1	IVA2_RST2		HDVICP2-2 sequencer2 CPU software reset
		0	No software reset occurred
		1	HDVICP2-2 logic and SL2 has been reset upon software reset
0	IVA2_RST1		HDVICP2-2 sequencer1 CPU software reset
		0	No software reset occurred
		1	HDVICP2-2 logic and SL2 has been reset upon software reset

18.7.16 PRM_SGX Device

Table 18-130 lists the registers for the PRM_SGX device.

Table 18-130. PRM_SGX Device Registers

Offset Address	Acronym	Section
F00h	PM_SGX_PWRSTCTRL	Section 18.7.16.1
F04h	RM_SGX_RSTCTRL	Section 18.7.16.2
F10h	PM_SGX_PWRSTST	Section 18.7.16.3
F14h	RM_SGX_RSTST	Section 18.7.16.4

18.7.16.1 PM_SGX_PWRSTCTRL Register

This register controls the SGX power state to reach upon a domain sleep transition [warm reset insensitive].

Figure 18-102. PM_SGX_PWRSTCTRL Register

31 18	17 16	15 2	1 0
Reserved	SGX_MEM_ONSTATE	Reserved	POWERSTATE
R-0	R-3h	R-0	R/W-3h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-131. PM_SGX_PWRSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	SGX_MEM_ONSTATE		SGX memory state when domain is ON.
		0	Reserved
		1h	Reserved
		2h	Reserved
		3h	Memory bank is on when the domain is ON.
15-2	Reserved	0	Reserved
1-0	POWERSTATE		Power state control
		0	OFF State [warm reset insensitive]
		1h	Reserved
		2h	Reserved
		3h	ON State [warm reset insensitive]

18.7.16.2 RM_SGX_RSTCTRL Register

This register controls the release of the SGX Domain resets.

Figure 18-103. RM_SGX_RSTCTRL Register

31 1	0
Reserved	SGX_RST
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-132. RM_SGX_RSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	SGX_RST		SGX domain local reset control
		0	Reset is cleared for the SGX Domain
		1	Reset is asserted for the SGX Domain

18.7.16.3 PM_SGX_PWRSTST Register

This register provides a status on the current SGX power domain state. [warm reset insensitive]

Figure 18-104. PM_SGX_PWRSTST Register

31	21	20	19	3	2	1	0
Reserved		INTRANSITION	Reserved		LOGICSTATEST	POWERSTATEST	
R-0		R-0	R-0		R-1	R-3h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-133. PM_SGX_PWRSTST Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	INTRANSITION		Domain transition status
		0	No on-going transition on power domain
		1	Power domain transition is in progress
19-3	Reserved	0	Reserved
2	LOGICSTATEST		Logic state status
		0	OFF State
		1h	Reserved
		2h	Reserved
		3h	ON State
1-0	POWERSTATEST		Current Power State Status
		0	OFF State
		1h	Reserved
		2h	Reserved
		3h	ON State

18.7.16.4 RM_SGX_RSTST Register

This register logs the different reset sources of the SGX domain. Each bit is set upon release of the domain reset signal. Must be cleared by software. [warm reset insensitive].

Figure 18-105. RM_SGX_RSTST Register

31	1	0
Reserved		SGX_RST
R-0		R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-134. RM_SGX_RSTST Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	SGX_RST		SGX Domain Logic Reset
		0	Reset is cleared for the SGX Domain
		1	Reset is asserted for the SGX Domain

18.7.17 CM_ALWON Device

Table 18-135 lists the registers for the CM_ALWON device.

Table 18-135. CM_ALWON Device Registers

Offset Address	Acronym	Section
1400h	CM_ALWON_L3_SLOW_CLKSTCTRL	Section 18.7.17.1
1404h	CM_ETHERNET_CLKSTCTRL	Section 18.7.17.2
1408h	CM_ALWON_L3_MED_CLKSTCTRL	Section 18.7.17.3
140Ch	CM_MMU_CLKSTCTRL	Section 18.7.17.4
1410h	CM_MMUCFG_CLKSTCTRL	Section 18.7.17.5
1414h	CM_ALWON_OCMC_0_CLKSTCTRL	Section 18.7.17.6
1418h	CM_ALWON_OCMC_1_CLKSTCTRL	Section 18.7.17.7
141Ch	CM_ALWON_MPU_CLKSTCTRL	Section 18.7.17.8
1420h	CM_ALWON_SYSCLK4_CLKSTCTRL	Section 18.7.17.9
1424h	CM_ALWON_SYSCLK5_CLKSTCTRL	Section 18.7.17.10
1428h	CM_ALWON_SYSCLK6_CLKSTCTRL	Section 18.7.17.11
142Ch	CM_ALWON_RTC_CLKSTCTRL	Section 18.7.17.12
1430h	CM_ALWON_L3_FAST_CLKSTCTRL	Section 18.7.17.13
1540h	CM_ALWON_MCASP0_CLKCTRL	Section 18.7.17.14
1544h	CM_ALWON_MCASP1_CLKCTRL	Section 18.7.17.15
1548h	CM_ALWON_MCASP2_CLKCTRL	Section 18.7.17.16
154Ch	CM_ALWON_MCBSP_CLKCTRL	Section 18.7.17.17
1550h	CM_ALWON_UART_0_CLKCTRL	Section 18.7.17.18
1554h	CM_ALWON_UART_1_CLKCTRL	Section 18.7.17.19
1558h	CM_ALWON_UART_2_CLKCTRL	Section 18.7.17.20
155Ch	CM_ALWON_GPIO_0_CLKCTRL	Section 18.7.17.21
1560h	CM_ALWON_GPIO_1_CLKCTRL	Section 18.7.17.22
1564h	CM_ALWON_I2C_0_CLKCTRL	Section 18.7.17.23
1568h	CM_ALWON_I2C_1_CLKCTRL	Section 18.7.17.24
1570h	CM_ALWON_TIMER_1_CLKCTRL	Section 18.7.17.25
1574h	CM_ALWON_TIMER_2_CLKCTRL	Section 18.7.17.26
1578h	CM_ALWON_TIMER_3_CLKCTRL	Section 18.7.17.27
157Ch	CM_ALWON_TIMER_4_CLKCTRL	Section 18.7.17.28
1580h	CM_ALWON_TIMER_5_CLKCTRL	Section 18.7.17.29
1584h	CM_ALWON_TIMER_6_CLKCTRL	Section 18.7.17.30
1588h	CM_ALWON_TIMER_7_CLKCTRL	Section 18.7.17.31
158Ch	CM_ALWON_WDTIMER_CLKCTRL	Section 18.7.17.32
1590h	CM_ALWON_SPI_CLKCTRL	Section 18.7.17.33
1594h	CM_ALWON_MAILBOX_CLKCTRL	Section 18.7.17.34
1598h	CM_ALWON_SPINBOX_CLKCTRL	Section 18.7.17.35
159Ch	CM_ALWON_MMUDATA_CLKCTRL	Section 18.7.17.36
15A8h	CM_ALWON_MMUCFG_CLKCTRL	Section 18.7.17.37
15B0h	CM_ALWON_SDIO_CLKCTRL	Section 18.7.17.38
15B4h	CM_ALWON_OCMC_0_CLKCTRL	Section 18.7.17.39
15B8h	CM_ALWON_OCMC_1_CLKCTRL	Section 18.7.17.40
15C4h	CM_ALWON_CONTRL_CLKCTRL	Section 18.7.17.41
15D0h	CM_ALWON_GPMC_CLKCTRL	Section 18.7.17.42
15D4h	CM_ALWON_ETHERNET_0_CLKCTRL	Section 18.7.17.43
15D8h	CM_ALWON_ETHERNET_1_CLKCTRL	Section 18.7.17.44
15DCh	CM_ALWON_MPU_CLKCTRL	Section 18.7.17.45

Table 18-135. CM_ALWON Device Registers (continued)

Offset Address	Acronym	Section
15E4h	CM_ALWON_L3_CLKCTRL	Section 18.7.17.46
15E8h	CM_ALWON_L4HS_CLKCTRL	Section 18.7.17.47
15ECh	CM_ALWON_L4LS_CLKCTRL	Section 18.7.17.48
15F0h	CM_ALWON_RTC_CLKCTRL	Section 18.7.17.49
15F4h	CM_ALWON_TPCC_CLKCTRL	Section 18.7.17.50
15F8h	CM_ALWON_TPTC0_CLKCTRL	Section 18.7.17.51
15FCh	CM_ALWON_TPTC1_CLKCTRL	Section 18.7.17.52
1600h	CM_ALWON_TPTC2_CLKCTRL	Section 18.7.17.53
1604h	CM_ALWON_TPTC3_CLKCTRL	Section 18.7.17.54
1608h	CM_ALWON_SR_0_CLKCTRL	Section 18.7.17.55
160Ch	CM_ALWON_SR_1_CLKCTRL	Section 18.7.17.56

18.7.17.1 CM_ALWON_L3_SLOW_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-106. 32-bit, 2 Rows

31	27	26	25	24
Reserved		CLKACTIVITY_TIMER7_GCLK	CLKACTIVITY_TIMER6_GCLK	CLKACTIVITY_TIMER5_GCLK
R-0		R-0	R-0	R-0
23		22	21	20
CLKACTIVITY_TIMER4_GCLK		CLKACTIVITY_TIMER3_GCLK	CLKACTIVITY_TIMER2_GCLK	CLKACTIVITY_TIMER1_GCLK
R-0		R-0	R-0	R-0
19		18	17	16
Reserved			CLKACTIVITY_SPI_GSYSCLK	CLKACTIVITY_I2C_GSYSCLK
R-0			R-0	R-0
15		14	13	12
CLKACTIVITY_GPIO_1_GDBCLK		CLKACTIVITY_GPIO_0_GDBCLK	CLKACTIVITY_UART_GFCLK	CLKACTIVITY_MCBSP_AUX_GCLK
R-0		R-0	R-0	R-0
11		10	9	8
CLKACTIVITY_MCASP2_AUX_GCLK		CLKACTIVITY_MCASP1_AUX_GCLK	CLKACTIVITY_MCASP0_AUX_GCLK	CLKACTIVITY_L3_SLOW_GCLK
R-0		R-0	R-0	R-0
7			2	1 0
Reserved				CLKTRCTRL
R-0				R/W-2h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-136. CM_ALWON_L3_SLOW_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26	CLKACTIVITY_TIMER7_GCLK		This field indicates the state of the TIMER7 CLKTIMER clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
25	CLKACTIVITY_TIMER6_GCLK		This field indicates the state of the TIMER6 CLKTIMER clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
24	CLKACTIVITY_TIMER5_GCLK		This field indicates the state of the TIMER5 CLKTIMER clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
23	CLKACTIVITY_TIMER4_GCLK		This field indicates the state of the TIMER4 CLKTIMER clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
22	CLKACTIVITY_TIMER3_GCLK		This field indicates the state of the TIMER3 CLKTIMER clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
21	CLKACTIVITY_TIMER2_GCLK		This field indicates the state of the TIMER2 CLKTIMER clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
20	CLKACTIVITY_TIMER1_GCLK		This field indicates the state of the TIMER1 CLKTIMER clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
19-18	Reserved	0	Reserved
17	CLKACTIVITY_SPI_GSYSCLK		This field indicates the state of the SPI_GSYSCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
16	CLKACTIVITY_I2C_GSYSCLK		This field indicates the state of the I2C_GSYSCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
15	CLKACTIVITY_GPIO_1_GDBCLK		This field indicates the state of the GPIO_GDBCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
14	CLKACTIVITY_GPIO_0_GDBCLK		This field indicates the state of the GPIO_GDBCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
13	CLKACTIVITY_UART_GFCLK		This field indicates the state of the UART_GFCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active

Table 18-136. CM_ALWON_L3_SLOW_CLKSTCTRL Register Field Descriptions (continued)

Bit	Field	Value	Description
12	CLKACTIVITY_MCBSP_AUX_GCLK		This field indicates the state of the MCBSP_AUX_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
11	CLKACTIVITY_MCASP2_AUX_GCLK		This field indicates the state of the MCASP2_AUX_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
10	CLKACTIVITY_MCASP1_AUX_GCLK		This field indicates the state of the MCASP1_AUX_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
9	CLKACTIVITY_MCASP0_AUX_GCLK		This field indicates the state of the MCASP0_AUX_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
8	CLKACTIVITY_L3_SLOW_GCLK		This field indicates the state of the L3_SLOW_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the L3_SLOW clock domain in Always ON power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.17.2 CM_ETHERNET_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-107. CM_ETHERNET_CLKSTCTRL Register

31 10	9	8	7 2	1 0
Reserved	CLKACTIVITY_RFT_GCLK	CLKACTIVITY_ETHERNET_GCLK	Reserved	CLKTRCTRL
R-0	R-0	R-0	R-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-137. CM_ETHERNET_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	CLKACTIVITY_RFT_GCLK		This field indicates the state of the CPGMAC_RFT_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
8	CLKACTIVITY_ETHERNET_GCLK		This field indicates the state of the ETHERNET_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the ETHERNET clock domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.17.3 CM_ALWON_L3_MED_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-108. CM_ALWON_L3_MED_CLKSTCTRL Register

31	2	1	0
Reserved		CLKTRCTRL	
R-0		R/W-2h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-138. CM_ALWON_L3_MED_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the L3 Medium clock domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.17.4 CM_MMU_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-109. CM_MMU_CLKSTCTRL Register

31	9	8	7	2	1	0
Reserved		CLKACTIVITY_MMU_GCLK	Reserved		CLKTRCTRL	
R-0		R-0	R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-139. CM_MMU_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_MMU_GCLK		This field indicates the state of the MMU_GICLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the MMU clock domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.17.5 CM_MMUCFG_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-110. CM_MMUCFG_CLKSTCTRL Register

31 9	8	7 2	1 0
Reserved	CLKACTIVITY_MMU_CFG_GCLK	Reserved	CLKTRCTRL
R-0	R-0	R-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-140. CM_MMUCFG_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_MMU_CFG_GCLK		This field indicates the state of the MMU_CFG_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the MMU CFG clock domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.17.6 CM_ALWON_OCMC_0_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-111. CM_ALWON_OCMC_0_CLKSTCTRL Register

31	9	8	7	2	1	0
Reserved		CLKACTIVITY_OCMC_0_GCLK	Reserved		CLKTRCTRL	
R-0		R-0	R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-141. CM_ALWON_OCMC_0_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_OCMC_0_GCLK		This field indicates the state of the OCMC_0_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the OCMC clock domain in DEFAULT power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.17.7 CM_ALWON_OCMC_1_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-112. CM_ALWON_OCMC_1_CLKSTCTRL Register

31	9	8	7	2	1	0
Reserved		CLKACTIVITY_OCMC_1_GCLK	Reserved		CLKTRCTRL	
R-0		R-0	R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 18-142. CM_ALWON_OCMC_1_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_OCMC_1_GCLK		This field indicates the state of the OCMC_1_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the OCMC_2 clock domain in DEFAULT power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.17.8 CM_ALWON_MPU_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-113. CM_ALWON_MPU_CLKSTCTRL Register

31	9	8	7	2	1	0
Reserved		CLKACTIVITY_MMU_GCLK	Reserved		CLKTRCTRL	
R-0		R-0	R-0		R-2h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-143. CM_ALWON_MPU_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_MPU_GCLK		This field indicates the state of the MPU_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the MPU clock domain in Always ON power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.17.9 CM_ALWON_SYSCLK4_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-114. CM_ALWON_SYSCLK4_CLKSTCTRL Register

31	12
Reserved	
R-0	

11	10	9	8
CLKACTIVITY_L3_F_EN_GCLK	CLKACTIVITY_L3_S_GCLK	CLKACTIVITY_L3_M_GCLK	CLKACTIVITY_SYSCLK4_GCLK
R-0	R-0	R-0	R-0

7	2	1	0
Reserved		CLKTRCTRL	
R-0		R-2h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 18-144. CM_ALWON_SYSCLK4_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11	CLKACTIVITY_L3_F_EN_GCLK		This field indicates the state of the L3_F_EN_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
10	CLKACTIVITY_L3_S_GCLK		This field indicates the state of the L3_S_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
9	CLKACTIVITY_L3_M_GCLK		This field indicates the state of the L3_M_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
8	CLKACTIVITY_SYSCLK4_GCLK		This field indicates the state of the SYSCLK4_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the SYSCLK4 clock domain in Always ON power domain.
		0	Reserved
		1h	Reserved
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.17.10 CM_ALWON_SYSCLK5_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-115. CM_ALWON_SYSCLK5_CLKSTCTRL Register

31	9	8	7	2	1	0
Reserved		CLKACTIVITY_SYSCLK5_GCLK	Reserved		CLKTRCTRL	
R-0		R-0	R-0		R-2h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-145. CM_ALWON_SYSCLK5_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_SYSCLK5_GCLK		This field indicates the state of the SYSCLK5_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the SYSCLK5 clock domain in Always ON power domain.
		0	Reserved
		1h	Reserved
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.17.11 CM_ALWON_SYSCLK6_CLKSTCTRL Register

This register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-116. CM_ALWON_SYSCLK6_CLKSTCTRL Register

31	9	8	7	2	1	0
Reserved		CLKACTIVITY_SYSCLK6_GCLK	Reserved		CLKTRCTRL	
R-0		R-0	R-0		R-2h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-146. CM_ALWON_SYSCLK6_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_SYSCLK6_GCLK		This field indicates the state of the SYSCLK6_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the SYSCLK6 clock domain in Always ON power domain.
		0	Reserved
		1h	Reserved
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.17.12 CM_ALWON_RTC_CLKSTCTRL Register

The CM_ALWON_RTC_CLKSTCTRL register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-117. CM_ALWON_RTC_CLKSTCTRL Register

31	9	8	7	2	1	0
Reserved		CLKACTIVITY_RTC_GCLK	Reserved		CLKTRCTRL	
R-0		R-0	R-0		R-2h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 18-147. CM_ALWON_RTC_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_RTC_GCLK		This field indicates the state of the RTC_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the RTC clock domain in Always ON power domain.
		0	Reserved
		1h	Reserved
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

18.7.17.13 CM_ALWON_L3_FAST_CLKSTCTRL Register

The CM_ALWON_L3_FAST_CLKSTCTRL register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

Figure 18-118. CM_ALWON_L3_FAST_CLKSTCTRL Register

31	10	9	8	2	1	0
Reserved		CLKACTIVITY_FAST_GCLK	Reserved		CLKTRCTRL	
R-0		R-0	R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-148. CM_ALWON_L3_FAST_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	CLKACTIVITY_FAST_GCLK		This field indicates the state of the L3 Fast clock for TPTC and TPCC in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
8-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the RTC clock domain in Always ON power domain.
		0	NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	HW_AUTO: Automatic transition is enabled. Sleep and wakeup transition are based upon hardware conditions.

18.7.17.14 CM_ALWON_MCASP0_CLKCTRL Register

The CM_ALWON_MCASP0_CLKCTRL register manages the MCASP_0 clocks. It is shown and described in the figure and table below.

Figure 18-119. CM_ALWON_MCASP0_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-149. CM_ALWON_MCASP0_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.15 CM_ALWON_MCASP1_CLKCTRL Register

The CM_ALWON_MCASP1_CLKCTRL register manages the MCASP1 clocks. It is shown and described in the figure and table below.

Figure 18-120. CM_ALWON_MCASP1_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-150. CM_ALWON_MCASP1_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.16 CM_ALWON_MCASP2_CLKCTRL Register

The CM_ALWON_MCASP2_CLKCTRL register manages the MCASP2 clocks. It is shown and described in the figure and table below.

Figure 18-121. CM_ALWON_MCASP2_CLKCTRL Register

31	18	17	16	15	2	1	0
Reserved		IDLEST		Reserved		MODULEMODE	
R-0		R-3h		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-151. CM_ALWON_MCASP2_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.17 CM_ALWON_MCBSP_CLKCTRL Register

The CM_ALWON_MCBSP_CLKCTRL register manages the MCBSP clocks. It is shown and described in the figure and table below.

Figure 18-122. CM_ALWON_MCBSP_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-152. CM_ALWON_MCBSP_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.18 CM_ALWON_UART_0_CLKCTRL Register

Th CM_ALWON_UART_0_CLKCTRL register manages the UART_0 clocks. It is shown and described in the figure and table below.

Figure 18-123. CM_ALWON_UART_0_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-153. CM_ALWON_UART_0_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.19 CM_ALWON_UART_1_CLKCTRL Register

The CM_ALWON_UART_1_CLKCTRL register manages the UART_1 clocks. It is shown and described in the figure and table below.

Figure 18-124. CM_ALWON_UART_1_CLKCTRL Register

31	18	17	16	15	2	1	0
Reserved		IDLEST		Reserved		MODULEMODE	
R-0		R-3h		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-154. CM_ALWON_UART_1_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.20 CM_ALWON_UART_2_CLKCTRL Register

The CM_ALWON_UART_2_CLKCTRL register manages the UART_2 clocks. It is shown and described in the figure and table below.

Figure 18-125. CM_ALWON_UART_2_CLKCTRL Register

31	18	17	16	15	2	1	0
Reserved		IDLEST		Reserved		MODULEMODE	
R-0		R-3h		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-155. CM_ALWON_UART_2_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.21 CM_ALWON_GPIO_0_CLKCTRL Register

The CM_ALWON_GPIO_0_CLKCTRL register manages the GPIO_0 clocks. It is shown and described in the figure and table below.

Figure 18-126. CM_ALWON_GPIO_0_CLKCTRL Register

31 18	17 16	15 9	8	7 2	1 0
Reserved	IDLEST	Reserved	OPTFCLKEN_DBCLK	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-156. CM_ALWON_GPIO_0_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-9	Reserved	0	Reserved
8	OPTFCLKEN_DBCLK		Optional functional clock control.
		0	Optional functional clock is disabled
		1	Optional functional clock is enabled
7-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.22 CM_ALWON_GPIO_1_CLKCTRL Register

Th e CM_ALWON_GPIO_1_CLKCTRL register manages the GPIO_1 clocks. It is shown and described in the figure and table below.

Figure 18-127. CM_ALWON_GPIO_1_CLKCTRL Register

31 18	17 16	15 9	8	7 2	1 0
Reserved	IDLEST	Reserved	OPTFCLKEN_DBCLK	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-157. CM_ALWON_GPIO_1_CLKCTRL Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-9	Reserved	0	Reserved
8	OPTFCLKEN_DBCLK		Optional functional clock control.
		0	Optional functional clock is disabled
		1	Optional functional clock is enabled
7-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.23 CM_ALWON_I2C_0_CLKCTRL Register

The CM_ALWON_I2C_0_CLKCTRL register manages the I2C_0 clocks. It is shown and described in the figure and table below.

Figure 18-128. CM_ALWON_I2C_0_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-158. CM_ALWON_I2C_0_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.24 CM_ALWON_I2C_1_CLKCTRL Register

The CM_ALWON_I2C_1_CLKCTRL register manages the I2C_1 clocks. It is shown and described in the figure and table below.

Figure 18-129. CM_ALWON_I2C_1_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-159. CM_ALWON_I2C_1_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.25 CM_ALWON_TIMER_1_CLKCTRL Register

The CM_ALWON_TIMER_1_CLKCTRL register manages the TIMER_1 clocks. It is shown and described in the figure and table below.

Figure 18-130. CM_ALWON_TIMER_1_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-160. CM_ALWON_TIMER_1_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.26 CM_ALWON_TIMER_2_CLKCTRL Register

The CM_ALWON_TIMER_2_CLKCTRL register manages the TIMER_2 clocks. It is shown and described in the figure and table below.

Figure 18-131. CM_ALWON_TIMER_2_CLKCTRL Register

31	18	17	16	15	2	1	0
Reserved		IDLEST		Reserved		MODULEMODE	
R-0		R-3h		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-161. CM_ALWON_TIMER_2_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.27 CM_ALWON_TIMER_3_CLKCTRL Register

This register manages the TIMER_3 clocks. It is shown and described in the figure and table below.

Figure 18-132. CM_ALWON_TIMER_3_CLKCTRL Register

31	18	17	16	15	2	1	0
Reserved		IDLEST		Reserved		MODULEMODE	
R-0		R-3h		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 18-162. CM_ALWON_TIMER_3_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.28 CM_ALWON_TIMER_4_CLKCTRL Register

The CM_ALWON_TIMER_4_CLKCTRL register manages the TIMER_4 clocks. It is shown and described in the figure and table below.

Figure 18-133. CM_ALWON_TIMER_4_CLKCTRL Register

31	18	17	16	15	2	1	0
Reserved		IDLEST		Reserved		MODULEMODE	
R-0		R-3h		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-163. CM_ALWON_TIMER_4_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.29 CM_ALWON_TIMER_5_CLKCTRL Register

The CM_ALWON_TIMER_5_CLKCTRL register manages the TIMER_5 clocks. It is shown and described in the figure and table below.

Figure 18-134. CM_ALWON_TIMER_5_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-164. CM_ALWON_TIMER_5_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.30 CM_ALWON_TIMER_6_CLKCTRL Register

The CM_ALWON_TIMER_6_CLKCTRL register manages the TIMER_6 clocks. It is shown and described in the figure and table below.

Figure 18-135. CM_ALWON_TIMER_6_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-165. CM_ALWON_TIMER_6_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.31 CM_ALWON_TIMER_7_CLKCTRL Register

The CM_ALWON_TIMER_7_CLKCTRL register manages the TIMER_7 clocks. It is shown and described in the figure and table below.

Figure 18-136. CM_ALWON_TIMER_7_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-166. CM_ALWON_TIMER_7_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.32 CM_ALWON_WDTIMER_CLKCTRL Register

The CM_ALWON_WDTIMER_CLKCTRL register manages the WDTIMER clocks. It is shown and described in the figure and table below.

Figure 18-137. CM_ALWON_WDTIMER_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-2h	R-0	R-2h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-167. CM_ALWON_WDTIMER_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.33 CM_ALWON_SPI_CLKCTRL Register

The CM_ALWON_SPI_CLKCTRL register manages the SPI clocks. It is shown and described in the figure and table below.

Figure 18-138. CM_ALWON_SPI_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-168. CM_ALWON_SPI_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.34 CM_ALWON_MAILBOX_CLKCTRL Register

The CM_ALWON_MAILBOX_CLKCTRL register manages the MAILBOX clocks. It is shown and described in the figure and table below.

Figure 18-139. CM_ALWON_MAILBOX_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-169. CM_ALWON_MAILBOX_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.35 CM_ALWON_SPINBOX_CLKCTRL Register

The CM_ALWON_SPINBOX_CLKCTRL register manages the SPINBOX clocks. It is shown and described in the figure and table below.

Figure 18-140. CM_ALWON_SPINBOX_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-170. CM_ALWON_SPINBOX_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.36 CM_ALWON_MMUDATA_CLKCTRL Register

The CM_ALWON_MMUDATA_CLKCTRL register manages the MMU data clocks. It is shown and described in the figure and table below.

Figure 18-141. CM_ALWON_MMUDATA_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-171. CM_ALWON_MMUDATA_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.37 CM_ALWON_MMUCFG_CLKCTRL Register

The CM_ALWON_MMUCFG_CLKCTRL register manages the MMU config clocks. It is shown and described in the figure and table below.

Figure 18-142. CM_ALWON_MMUCFG_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-172. CM_ALWON_MMUCFG_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.38 CM_ALWON_SDIO_CLKCTRL Register

The CM_ALWON_SDIO_CLKCTRL register manages the SDIO clocks. It is shown and described in the figure and table below.

Figure 18-143. CM_ALWON_SDIO_CLKCTRL Register

31	18	17	16	15	2	1	0
Reserved		IDLEST		Reserved		MODULEMODE	
R-0		R-3h		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-173. CM_ALWON_SDIO_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.39 CM_ALWON_OCMC_0_CLKCTRL Register

The CM_ALWON_OCMC_0_CLKCTRL register manages the OCMC_0 clocks. It is shown and described in the figure and table below.

Figure 18-144. CM_ALWON_OCMC_0_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-174. CM_ALWON_OCMC_0_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.40 CM_ALWON_OCMC_1_CLKCTRL Register

The CM_ALWON_OCMC_1_CLKCTRL register manages the OCMC_1 clocks. It is shown and described in the figure and table below.

Figure 18-145. CM_ALWON_OCMC_1_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-175. CM_ALWON_OCMC_1_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.41 CM_ALWON_CONTRL_CLKCTRL Register

TheiCM_ALWON_CONTRL_CLKCTRL register manages the CONTRL clocks. It is shown and described in the figure and table below.

Figure 18-146. CM_ALWON_CONTRL_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-176. CM_ALWON_CONTRL_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.42 CM_ALWON_GPMC_CLKCTRL Register

The CM_ALWON_GPMC_CLKCTRL register manages the GPMC clocks. It is shown and described in the figure and table below.

Figure 18-147. CM_ALWON_GPMC_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-177. CM_ALWON_GPMC_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.43 CM_ALWON_ETHERNET_0_CLKCTRL Register

The CM_ALWON_ETHERNET_0_CLKCTRL register manages the ETHERNET_0 clocks. It is shown and described in the figure and table below.

Figure 18-148. CM_ALWON_ETHERNET_0_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-178. CM_ALWON_ETHERNET_0_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status.
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.44 CM_ALWON_ETHERNET_1_CLKCTRL Register

The CM_ALWON_ETHERNET_1_CLKCTRL register manages the ETHERNET_1 clocks. It is shown and described in the figure and table below.

Figure 18-149. CM_ALWON_ETHERNET_1_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-179. CM_ALWON_ETHERNET_1_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status.
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.45 CM_ALWON_MPU_CLKCTRL Register

The CM_ALWON_MPU_CLKCTRL register manages the MPU clocks. It is shown and described in the figure and table below.

Figure 18-150. CM_ALWON_MPU_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-0	R-0	R-0	R-2h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-180. CM_ALWON_MPU_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status.
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.46 CM_ALWON_L3_CLKCTRL Register

The CM_ALWON_L4HS_CLKCTRL register manages the L3 clocks. It is shown and described in the figure and table below.

Figure 18-151. CM_ALWON_L3_CLKCTRL Register

31	18	17	16	15	2	1	0
Reserved		IDLEST		Reserved		MODULEMODE	
R-0		R-0		R-0		R-2h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-181. CM_ALWON_L3_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.47 CM_ALWON_L4HS_CLKCTRL Register

The CM_ALWON_L4HS_CLKCTRL register manages the L4HS clocks. It is shown and described in the figure and table below.

Figure 18-152. CM_ALWON_L4HS_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-0	R-0	R-2h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-182. CM_ALWON_L4HS_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.48 CM_ALWON_L4LS_CLKCTRL Register

The CM_ALWON_L4LS_CLKCTRL register manages the L4LS clocks. It is shown and described in the figure and table below.

Figure 18-153. CM_ALWON_L4LS_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-0	R-0	R-2h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-183. CM_ALWON_L4LS_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.49 CM_ALWON_RTC_CLKCTRL Register

The CM_ALWON_RTC_CLKCTRL register manages the RTC clocks. It is shown and described in the figure and table below.

Figure 18-154. CM_ALWON_RTC_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-0	R-0	R-2h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-184. CM_ALWON_RTC_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.50 CM_ALWON_TPCC_CLKCTRL Register

The CM_ALWON_TPCC_CLKCTRL register manages the TPCC clocks. It is shown and described in the figure and table below.

Figure 18-155. CM_ALWON_TPCC_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-185. CM_ALWON_TPCC_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.51 CM_ALWON_TPTC0_CLKCTRL Register

The CM_ALWON_TPTC0_CLKCTRL register manages the TPTC_0 clocks. It is shown and described in the figure and table below.

Figure 18-156. CM_ALWON_TPTC0_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-186. CM_ALWON_TPTC0_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status.
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.52 CM_ALWON_TPTC1_CLKCTRL Register

The CM_ALWON_TPTC1_CLKCTRL register manages the TPTC_1 clocks. It is shown and described in the figure and table below.

Figure 18-157. CM_ALWON_TPTC1_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-187. CM_ALWON_TPTC1_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status.
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.53 CM_ALWON_TPTC2_CLKCTRL Register

The CM_ALWON_TPTC2_CLKCTRL register manages the TPTC_2 clocks. It is shown and described in the figure and table below.

Figure 18-158. CM_ALWON_TPTC2_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-188. CM_ALWON_TPTC2_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status.
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.54 CM_ALWON_TPTC3_CLKCTRL Register

The CM_ALWON_TPTC3_CLKCTRL register manages the TPTC_3 clocks. It is shown and described in the figure and table below.

Figure 18-159. CM_ALWON_TPTC3_CLKCTRL Register

31 19	18	17 16	15 2	1 0
Reserved	STBYST	IDLEST	Reserved	MODULEMODE
R-0	R-1	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-189. CM_ALWON_TPTC3_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	STBYST		Module standby status.
		0	Module is functional (not in standby)
		1	Module is in standby
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.55 CM_ALWON_SR_0_CLKCTRL Register

The CM_ALWON_SR_0_CLKCTRL register manages the Smart reflex 0 clocks. It is shown and described in the figure and table below.

Figure 18-160. CM_ALWON_SR_0_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-190. CM_ALWON_SR_0_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

18.7.17.56 CM_ALWON_SR_1_CLKCTRL Register

The CM_ALWON_SR_1_CLKCTRL register manages the Smart reflex 1 clocks. It is shown and described in the figure and table below.

Figure 18-161. CM_ALWON_SR_1_CLKCTRL Register

31 18	17 16	15 2	1 0
Reserved	IDLEST	Reserved	MODULEMODE
R-0	R-3h	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-191. CM_ALWON_SR_1_CLKCTRL Register Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	IDLEST		Module idle status.
		0	Module is fully functional, including OCP
		1h	Module is performing transition: wakeup, or sleep, or sleep abortion
		2h	Module is in Idle mode (only OCP part). It is functional if using separate functional clock
		3h	Module is disabled and cannot be accessed
15-2	Reserved	0	Reserved
1-0	MODULEMODE		Control the way mandatory clocks are managed.
		0	Module is disabled by software. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).
		1h	Reserved
		2h	Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen.
		3h	Reserved

Real-Time Clock (RTC)

This chapter provides a functional presentation of real-time clock (RTC).

Topic **Page**

19.1 Introduction

19.1.1 Overview

The real-time clock is a precise timer which can generate interrupts on intervals specified by the user. Interrupts can occur every second, minute, hour, or day. The clock itself can track the passage of real time for durations of several years, provided it has a sufficient power source the whole time.

The basic purpose for the RTC is to keep time of day. The other equally important purpose of RTC is for Digital Rights management. Some degree of tamper proofing is needed to ensure that simply stopping, resetting, or corrupting the RTC does not go unnoticed so that if this occurs, the application can re-acquire the time of day from a trusted source. The final purpose of RTC is to wake the rest of chip up from a power down state.

Alarms are available to interrupt the CPU at a particular time, or at periodic time intervals, such as once per minute or once per day. In addition, the RTC can interrupt the CPU every time the calendar and time registers are updated, or at programmable periodic intervals.

19.1.2 Features

The real-time clock (RTC) provides the following features:

- 100-year calendar (xx00 to xx99)
- Counts seconds, minutes, hours, day of the week, date, month, and year with leap year compensation
- Binary-coded-decimal (BCD) representation of time, calendar, and alarm
- 12-hour clock mode (with AM and PM) or 24-hour clock mode
- Alarm interrupt
- Periodic interrupt
- Single interrupt to the CPU

19.1.3 Functional Block Diagram

Figure 19-1 shows the RTC module block diagram. Figure 19-2 shows a functional block diagram of the RTC.

Figure 19-1. RTC Block Diagram

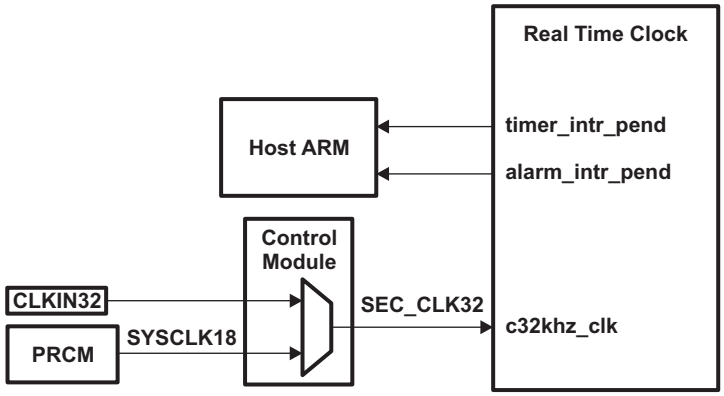
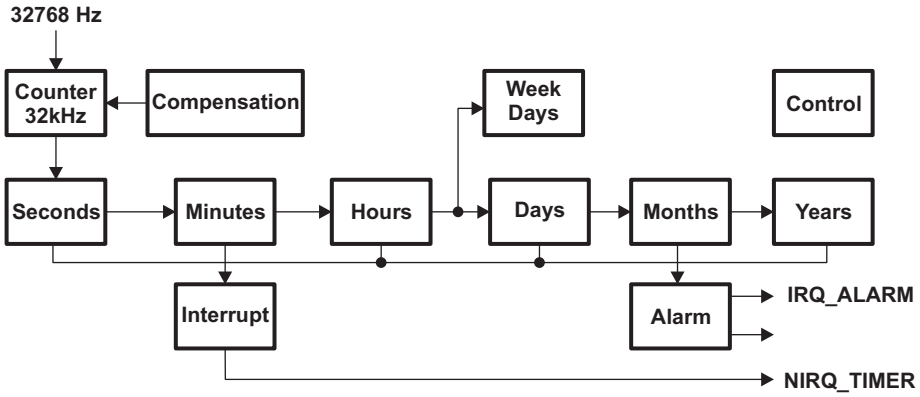


Figure 19-2. RTC Functional Block Diagram



19.2 Architecture

This section defines the module interrupt capabilities and requirements.

19.2.1 Clock Source

The clock reference for the RTC is an internal 32.768-kHz clock signal (SYSCLK18) or an optional external clock source of the same frequency input on the CLKIN32 pin, as selected by the device Control Module. For more information about the optional CLKIN32 pin connection, see your device-specific data manual. If the CLKIN32 pin is not used, this pin should be held low.

The RTC_DISABLE bit in the control register (RTC_CTRL_REG) can be set to save power; however, the RTC_DISABLE bit should not be cleared once it has been set. If the application requires the RTC module to stop and continue, the STOP_RTC bit in RTC_CTRL_REG should be used instead.

19.2.2 Signal Descriptions

Table 19-1 lists the signals and their descriptions for the RTC.

Table 19-1. RTC Signals

Signal	I/O	Description
32.768 kHz Functional Clock	I	RTC clock input: SYSCLK18 or optional CLKIN32 pin

19.2.3 Interrupt Support

19.2.3.1 CPU Interrupts

The RTC generates two interrupt outputs:

- timer_intr is a timer interrupt.
- alarm_intr is an alarm interrupt.

NOTE: Both interrupt outputs support high-level and high-pulse.

19.2.3.2 Interrupt Description

19.2.3.2.1 Timer Interrupt (timer_intr)

The timer interrupt can be generated periodically: every second, every minute, every hour, or every day (see RTC_INTERRUPTS_REG[1:0] for a description of how to set this up). The IT_TIMER bit of RTC_INTERRUPTS_REG enables this interrupt. The timer interrupt is active-low.

The RTC_STATUS_REG[5:2] bits are only updated at each new interrupt and occur according to Table 19-2. For example, bit 2 (SEC) will always be set when one second has passed. It will also be set when one minute has passed since the completion of one minute also marks the completion of one second (from 59 seconds to 60 seconds). The same holds true for hours and days: each of them will also correspond to the passing of a second.

Conversely, bit 5 (DAY) will always be set when a day has passed. It might also be set when an hour, minute, or second has passed. However, this only occurs when the elapsed hour, minute, or second corresponds to the start of a new day.

Table 19-2. Interrupt Trigger Events

RTC_STATUS_REG bit	One day has passed	One hour has passed	One minute has passed	One second has passed
[5] (DAY)	1	0/1[(1)]	0/1[(1)]	0/1[(1)]
[4] (HOUR)	1	1	0/1[(1)]	0/1[(1)]
[3] (MIN)	1	1	1	0/1[(1)]
[2] (SEC)	1	1	1	1

(1) This event is only triggered when the elapsed time unit (for example, Day) corresponds to the passage of another unit (for example, Seconds). For example, when the clock ticks from 00:23:59:59 (days : hours : minutes : seconds) to 01:00:00:00.

19.2.3.2.2 Alarm Interrupt (alarm_intr)

The alarm interrupt can be generated when the time set into the TC ALARM registers is exactly the same as in the TC registers. This interrupt is then generated, if the IT_ALARM bit in the interrupt register (RTC_INTERRUPTS_REG) is set. This interrupt is low-level sensitive. The RTC_STATUS_REG[6] bit indicates that IRQ_ALARM_CHIP has occurred. This interrupt is disabled by writing a 1 into the RTC_STATUS_REG[6] bit.

To set up an alarm:

- Modify the ALARM_SECONDS_REG, ALARM_MINUTES_REG, ALARM_HOURS_REG, ALARM_DAYS_REG, ALARM_MONTHS_REG, and ALARM_YEARS_REG registers to the exact time you want an alarm to generate.
- Set the IT_ALARM bit in RTC_INTERRUPTS_REG to enable the alarm interrupt.

19.2.4 Programming/Usage Guide

19.2.4.1 Time/Calendar Data Format

The time and calendar data in the RTC is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. Although most of the time/calendar registers have 4 bits assigned to each BCD digit, some of the register fields are shorter since the range of valid numbers may be limited. For example, only 3 bits are required to represent the day of the week (WEEK_REG) since only BCD numbers 1 through 7 are required. The following time and calendar registers are supported (BCD Format):

Note that the ALARM registers which share the names above also share the same BCD formatting.

- SECOND - Second Count (00-59)
- MINUTE - Minute Count (00-59)
- HOUR - Hour Count (12HR: 01-12; 24HR: 00-23)
- DAY - Day of the Month Count (01-31)
- WEEK - Day of the Week (0-6: SUN = 0)
- MONTH - Month Count (01-12; JAN = 1)
- YEAR - Year Count (00-99)

19.2.4.2 Register Access

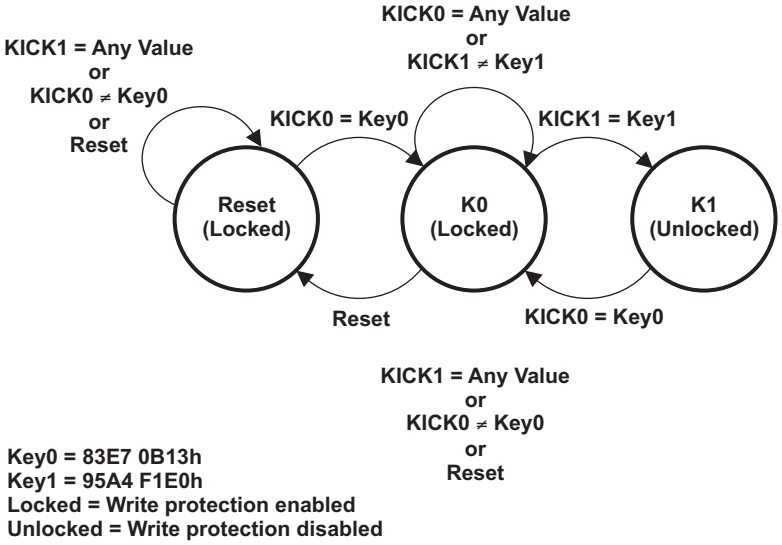
The three register types are as follows and each has its own access constraints:

- TC registers and TC alarm registers
- General registers
- Compensation registers

19.2.4.3 MMR Spurious Write Protection

The module also contains a kicker mechanism (Figure 19-3) to prevent any spurious writes from changing the register values. This mechanism requires two MMR writes to the Kick0 and Kick1 registers with exact data values before the kicker lock mechanism is released. Once released, the MMRs are writeable. The Kick0 data is 83E7 0B13h; the Kick1 data is 95A4 F1E0h. Note that it remains in an unlocked state until a reset or invalid data pattern is written to one of the Kick0 or Kick1 registers.

Figure 19-3. Kick Register State Machine Diagram



19.2.4.4 Reading the Timer/Calendar (TC) Registers

The TC registers have a read-show register. The reading of the SECONDS register will update all of the TC registers. For example, the Year will only get updated on a reading of the SECONDS register. The time/calendar registers are updated every second as the time changes. During a read of the SECONDS register, the RTC copies the current values of the time/date registers into shadow read registers. This isolation assures that the CPU can capture all the time/date values at the moment of the SECONDS read request and not be subject to changing register values from time updates.

If desired, the RTC also provides a one-time-triggered minute-rounding feature to round the MINUTE:SECOND registers to the nearest minute (with zero seconds). This feature is enabled by setting the ROUND_30S bit in the control register (RTC_CTRL_REG); the RTC automatically rounds the time values to the nearest minute upon the next read of the SECONDS register.

NOTE: Software should always read the SECONDS register first. However, the software does not have to poll any status bit to determine when to read the TC registers. Table 19-3 defines the TC set that gets shadowed.

Table 19-3. RTC Register Names and Values

Time Unit	Range	Remarks
Year	00 to 99	
Month	01 to 12	
Day	01 to 31	Months 1, 3, 5, 7, 8, 10, 12
	01 to 30	Months 4, 6, 9, 11
	01 to 29	Month 2 (leap year)
	01 to 28	Month 2 (common year)
Week	00 to 06	Day of week
Hour	00 to 23	24 hour mode
	01 to 12	AM/PM mode
Minute	00 to 59	
Seconds	00 to 59	

19.2.4.4.1 Rounding Seconds

Time can be rounded to the closest minute, by setting the ROUND_30S bit of the control register (RTC_CTRL_REG). When this bit is set, TC values are set to the closest minute value at the next second. The ROUND_30S bit is automatically cleared when rounding time is performed.

Example:

- If current time is 10H59M45S, round operation will change time to 11H00M00S.
- If current time is 10H59M29S, round operation will change time to 10H59M00S.

19.2.4.5 Modifying the TC Registers

To write correct data from/to the TC and TC alarm registers and read the TC alarm registers, the ARM must first read the BUSY bit of the status register (RTC_STATUS_REG) until BUSY is equal to zero. Once the BUSY flag is zero, there is a 15 µs access period in which the ARM can program the TC and TC alarm registers. Once the 15 µs access period passes, the BUSY flag has to be read again from the STATUS register as described previously. If the ARM accesses the TC registers outside of the access period, then the access is not assured.

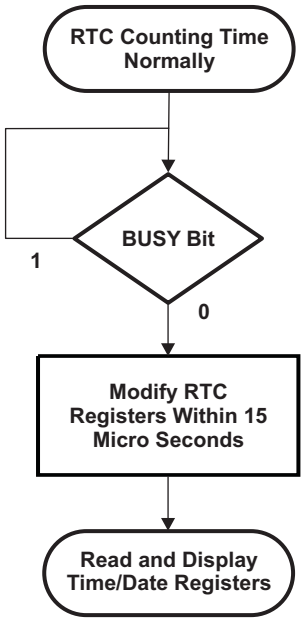
The ARM can access the RTC_STATUS_REG and the RTC_CTRL_REG registers at any time, with the exception of the RTC_CTRL_REG[5] bit, which can only be changed when the RTC is stopped. The ARM can stop the RTC by clearing the STOP_RTC bit of RTC_CTRL_REG. After clearing this bit, the RUN bit in RTC_STATUS_REG needs to be checked to verify the RTC has stopped. Once this is confirmed, the TC values can be updated. After the values have been updated, the RTC can be re-started by resetting the STOP_RTC bit.

NOTE: After writing to a TC register, you must wait 4 clock cycles before reading the value from the register. If this wait time is not observed and the TC register is accessed, then old data will be read from the register.

CAUTION

In order to remove any possibility of interrupting the register's read process, thus introducing a potential risk of violating the authorized 15 µs access period, it is recommended that you disable all incoming interrupts during the register read process.

Figure 19-4. Flow Control for Updating RTC Registers



19.2.4.5.1 General Registers

The ARM can access the RTC_STATUS_REG and the RTC_CTRL_REG registers at any time (except the RTC_CTRL_REG[5] bit, which must be changed only when the RTC is stopped). For the RTC_INTERRUPTS_REG, the ARM should respect the available access period to prevent spurious interrupt.

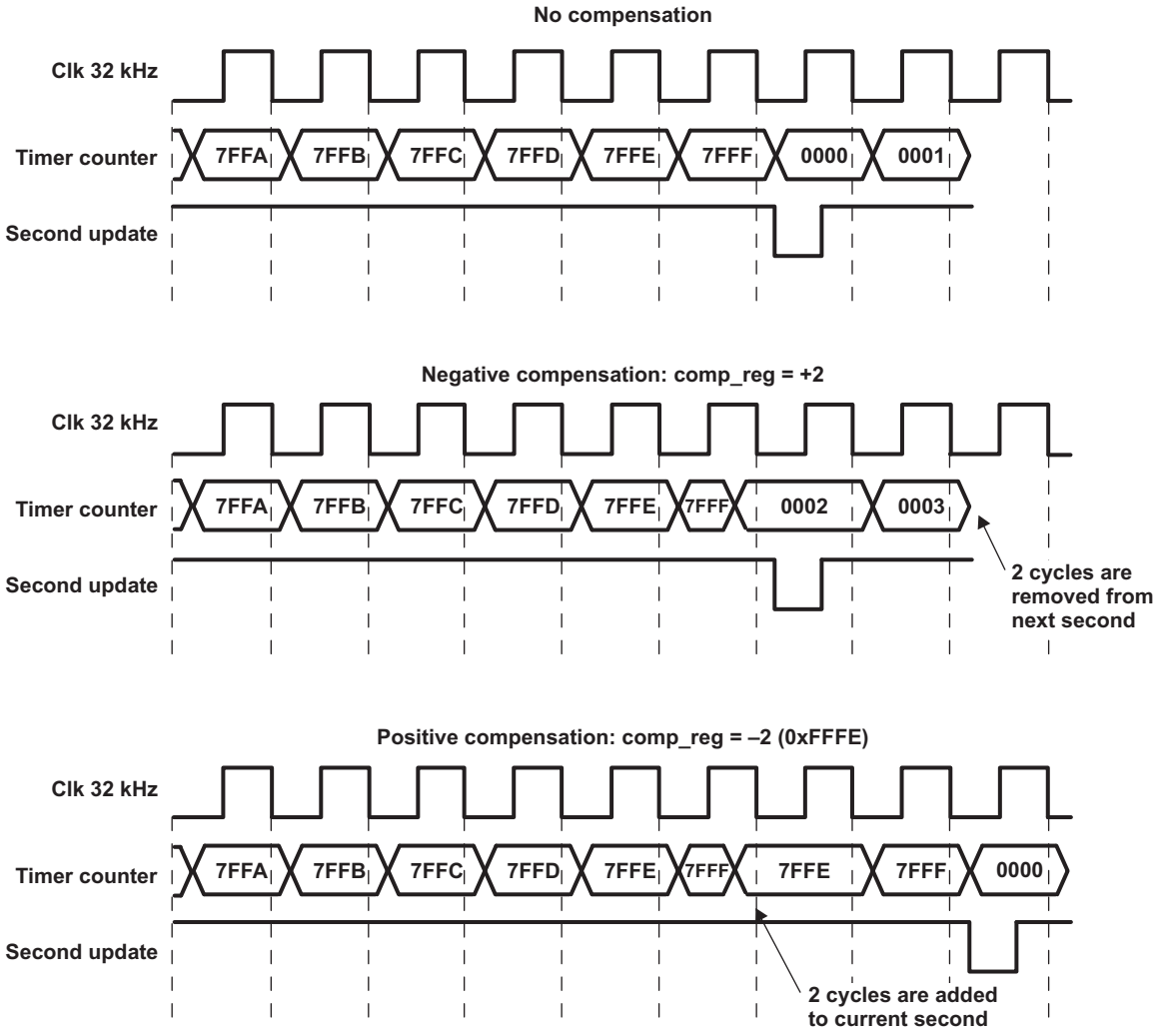
The RTC_DISABLE bit of RTC_CTRL_REG must only be used to completely disable the RTC function. When this bit is set, the 32 kHz clock is gated, and the RTC is frozen. From this point, resetting this bit to zero can lead to unexpected behavior. In order to save power, this bit should only be used if the RTC function is unwanted in the application.

19.2.4.6 Crystal Compensation

To compensate for any inaccuracy of the 32 kHz oscillator, the ARM can perform a calibration of the oscillator frequency, calculate the drift compensation versus one-hour period, and load the compensation registers with the drift compensation value. Auto compensation is enabled by the AUTO_COMP bit in the RTC_CTRL_REG register. If the RTC_COMP_REG value is positive, compensation occurs after the second change event. COMP_REG cycles are removed from the next second. If the RTC_COMP_REG value is negative, compensation occurs before the second change event. RTC_COMP_REG cycles are added to the current second. This enables compensation with a 1 32-kHz period accuracy each hour. The waveform below summarizes positive and negative compensation effect.

Access to the RTC_COMP_MSB_REG and the RTC_COMP_LSB_REG registers must respect the available access period. These registers should not be updated during compensation (first second of each hour), but it is alright to update them during the second preceding a compensation event. For example, the ARM could load the compensation value into these registers after each hour event, during an available access period.

Figure 19-5. Compensation Illustration



19.2.5 Scratch Registers

The RTC provides three general-purpose registers (RTC_SCRATCHx_REG) that can be used to store 32-bit words -- these registers have no functional purpose for the RTC. Software using the RTC may find the RTC_SCRATCHx_REG registers to be useful in indicating RTC states. For example, the RTC_SCRATCHx_REG registers may be used to indicate write-protection lock status or unintentional power downs. To indicate write-protection, the software should write a unique value to one of the RTC_SCRATCHx_REG registers when write-protection is disabled and another unique value when write-protection is enabled again. In this way, the lock-status of the registers can be determined quickly by reading the RTC_SCRATCHx_REG register. To indicate unintentional power downs, the software should write a unique value to one of the RTC_SCRATCHx_REG registers when RTC is configured and enabled. If the RTC is unintentionally powered down, the value written to the RTC_SCRATCHx_REG register is cleared.

19.2.6 Power Management

The RTC supports the power idle protocol. It has two SWakeup ports: one for the alarm event and one for a timer event.

When the RTC is in IDLE mode, the clock is turned off and the 32 kHz clock remains on. The time and calendar continue to count in IDLE mode. When the RTC is placed back in FUNCTIONAL mode, the TC registers can be read.

The Alarm SWakeup event can be used to wakeup the RTC when it is in IDLE state. In order to do so, the alarm needs to be set and enabled before RTC enters the IDLE state. Once this is done, the SWakeup will occur when the alarm event triggers.

NOTE: Since SWakeup is not periodic, using it to wake up the RTC when in IDLE state is not recommended, use Alarm SWakeup instead.

19.2.7 Reset Considerations

When the device is initially powered on, the RTC may issue spurious interrupt signals to the CPU. To avoid issues, a software reset should be performed on the RTC module before the CPU interrupt controller is initialized.

19.3 RTC Registers

Table 19-4 lists the registers for the RTC. For the base address of these registers, see Table 1-12.

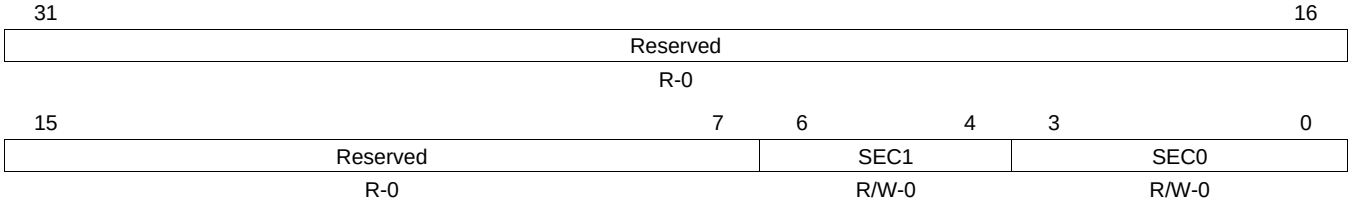
Table 19-4. Real-Time Clock (RTC) Registers

Address Offset	Acronym	Register Description	Section
0h	SECONDS_REG	Seconds Register	Section 19.3.1
4h	MINUTES_REG	Minutes Register	Section 19.3.2
8h	HOURS_REG	Hours Register	Section 19.3.3
Ch	DAYS_REG	Day of the Month Register	Section 19.3.4
10h	MONTHS_REG	Month Register	Section 19.3.5
14h	YEARS_REG	Year Register	Section 19.3.6
18h	WEEK_REG	Day of the Week Register	Section 19.3.7
20h	ALARM_SECONDS_REG	Alarm Seconds Register	Section 19.3.8
24h	ALARM_MINUTES_REG	Alarm Minutes Register	Section 19.3.9
28h	ALARM_HOURS_REG	Alarm Hours Register	Section 19.3.10
2Ch	ALARM_DAYS_REG	Alarm Day of the Month Register	Section 19.3.11
30h	ALARM_MONTHS_REG	Alarm Months Register	Section 19.3.12
34h	ALARM_YEARS_REG	Alarm Years Register	Section 19.3.13
40h	RTC_CTRL_REG	Control Register	Section 19.3.14
44h	RTC_STATUS_REG	Status Register	Section 19.3.15
48h	RTC_INTERRUPTS_REG	Interrupt Enable Register	Section 19.3.16
4Ch	RTC_COMP_LSB_REG	Compensation (LSB) Register	Section 19.3.17
50h	RTC_COMP_MSB_REG	Compensation (MSB) Register	Section 19.3.18
54h	RTC_OSC_REG	Oscillator Register	Section 19.3.19
60h	RTC_SCRATCH0_REG	Scratch 0 Register (General-Purpose)	Section 19.3.20
64h	RTC_SCRATCH1_REG	Scratch 1 Register (General-Purpose)	Section 19.3.20
68h	RTC_SCRATCH2_REG	Scratch 2 Register (General-Purpose)	Section 19.3.20
6Ch	KICK0R	Kick 0 Register (Write Protect)	Section 19.3.21
70h	KICK1R	Kick 1 Register (Write Protect)	Section 19.3.21
74h	RTC_REVISION	Revision Register	Section 19.3.22
78h	RTC_SYSCONFIG	System Configuration Register	Section 19.3.23
7Ch	RTC_IRQWAKEEN_0	Wakeup Enable Register	Section 19.3.24

19.3.1 Seconds Register (SECONDS_REG)

The SECONDS_REG is used to program the required seconds value of the current time. Seconds are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. If the seconds value is 45, then the value of SEC0 is 5 and value of SEC1 is 4.

Figure 19-6. Seconds Register (SECONDS_REG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

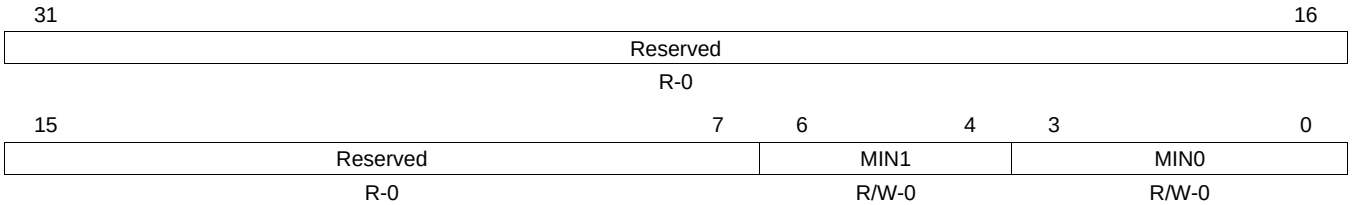
Table 19-5. Seconds Register (SECONDS_REG) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved.
6-4	SEC1	0-5h	2nd digit of seconds, Range is 0 to 5
3-0	SEC0	0-9h	1st digit of seconds, Range is 0 to 9

19.3.2 Minutes Register (MINUTES_REG)

The MINUTES_REG is used to program the minutes value of the current time. Minutes are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. If the minutes value is 32, then the value of MIN0 is 2 and value of MIN1 is 3.

Figure 19-7. Minutes Register (MINUTES_REG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-6. Minutes Register (MINUTES_REG) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6-4	MIN1	0-5h	2nd digit of minutes, Range is 0 to 5
3-0	MIN0	0-9h	1st digit of minutes, Range is 0 to 9

19.3.3 Hours Register (HOURS_REG)

The HOURS_REG is used to program the hours value of the current time. Hours are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. In 24Hr time mode if you want to set the hour as 18, then HOUR0 is set as 8 and HOUR1 is set as 1.

Figure 19-8. Hours Register (HOURS_REG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PM_nAM	Rsvd	HOUR1		HOUR0			
R-0								R/W-0	R-0	R/W-0		R/W-0			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-7. Hours Register (HOURS_REG) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	PM_nAM		Only used in PM_AM mode (otherwise 0)
		0	AM
		1	PM
6	Reserved	0	Reserved
5-4	HOUR1	0-2h	2nd digit of hours, Range is 0 to 2
3-0	HOUR0	0-9h	1st digit of hours, Range is 0 to 9

19.3.4 Day of the Month Register (DAYS_REG)

The DAYS_REG is used to program the day of the month value of the current date. Days are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. If the day value of the date is 28, DAY0 is set as 8 and DAY1 is set as 2.

Figure 19-9. Days of the Month Register (DAYS_REG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										DAY1		DAY0			
R-0										R/W-0		R/W-1			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-8. Day of the Month (DAYS_REG) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5-4	DAY1	0-3h	2nd digit of days, Range is 0 to 3
3-0	DAY0	0-9h	1st digit of days, Range is 0 to 9

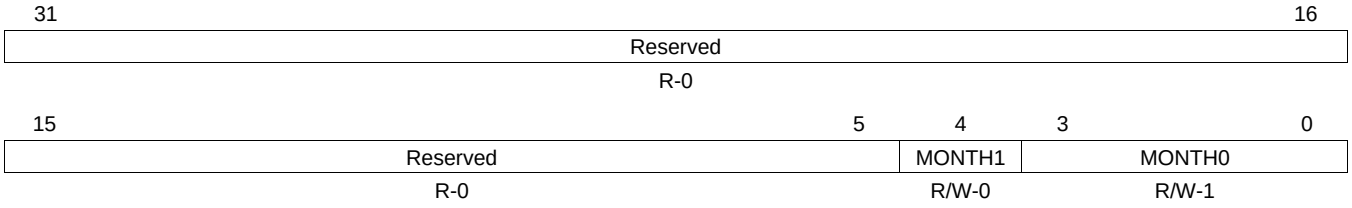
19.3.5 Month Register (MONTHS_REG)

The MONTHS_REG is used to set the month in the year value of the current date. Months are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

NOTE: Usual notation is taken for month value:

- 01 => January
- 02 => February
- ...
- 12 => December

Figure 19-10. Month Register (MONTHS_REG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

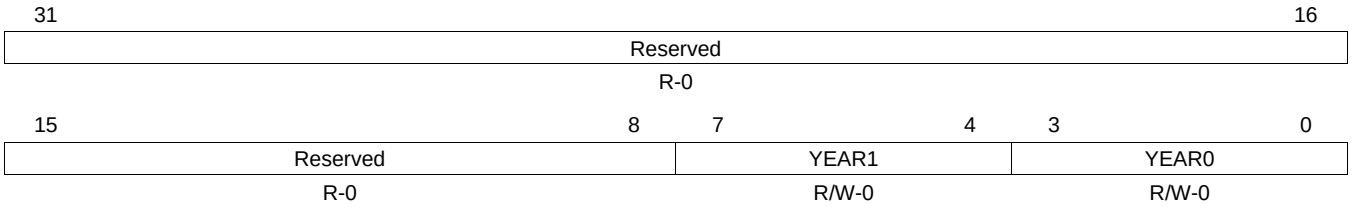
Table 19-9. Month Register (MONTHS_REG) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	MONTH1	0-1h	2nd digit of months, Range is 0 to 1
3-0	MONTH0	0-9h	1st digit of months, Range is 0 to 9

19.3.6 Year Register (YEARS_REG)

The YEARS_REG is used to program the year value of the current date. The year value is represented by only the last 2 digits and is stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The year 1979 is programmed as 79 with YEAR0 set as 9 and YEAR1 set as 7.

Figure 19-11. Year Register (YEARS_REG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-10. Year Register (YEARS_REG) Field Descriptions

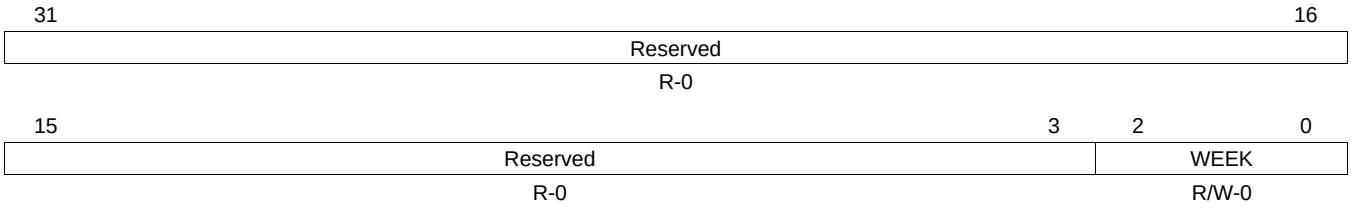
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-4	YEAR1	0-9h	2nd digit of years, Range is 0 to 9
3-0	YEAR0	0-9h	1st digit of years, Range is 0 to 9

19.3.7 Day of the Week Register (WEEKS_REG)

The WEEKS_REG is used to program the day of the week value of the current date. The day of the week is stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

NOTE: Sunday is treated as 0, Monday 1, and ending at Saturday with 6.

Figure 19-12. Day of the Week Register (WEEKS_REG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

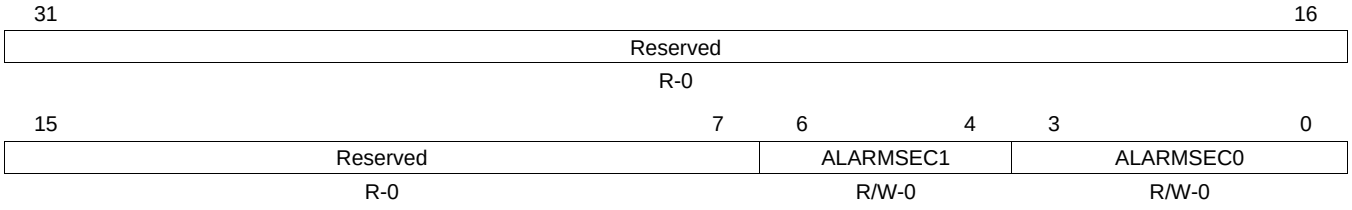
Table 19-11. Day of the Week (WEEKS_REG) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	WEEK	0-6h	1st digit of days in a week, Range from 0 (Sunday) to 6 (Saturday)

19.3.8 Alarm Seconds Register (ALARM_SECONDS_REG)

The ALARM_SECONDS_REG is used to program the second value for the alarm interrupt. Seconds are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

Figure 19-13. Alarm Second Register (ALARM_SECONDS_REG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

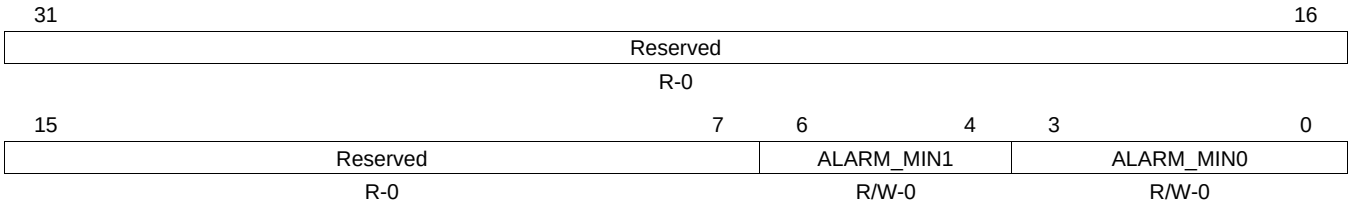
Table 19-12. Alarm Second Register (ALARM_SECONDS_REG) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6-4	ALARMSEC1	0-5h	2nd digit of seconds, Range is 0 to 5
3-0	ALARMSEC0	0-9h	1st digit of seconds, Range is 0 to 9

19.3.9 Alarm Minutes Register (ALARM_MINUTES_REG)

The ALARM_MINUTES_REG is used to program the minute value for the alarm interrupt. Minutes are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

Figure 19-14. Alarm Minute Register (ALARM_MINUTES_REG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

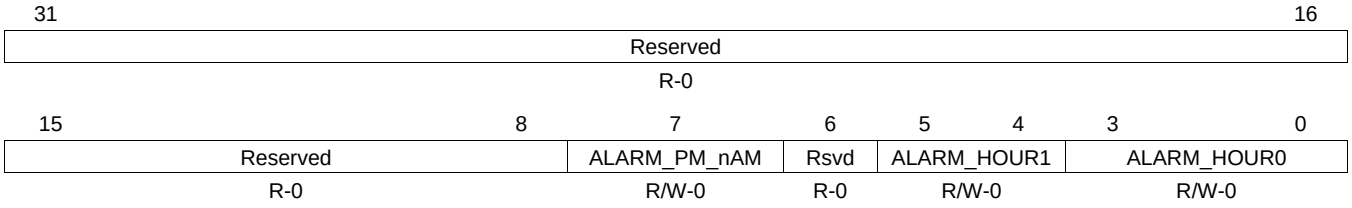
Table 19-13. Alarm Minute Register (ALARM_MINUTES_REG) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6-4	ALARM_MIN1	0-5h	2nd digit of minutes, Range is 0 to 5
3-0	ALARM_MIN0	0-9h	1st digit of minutes, Range is 0 to 9

19.3.10 Alarm Hours Register (ALARM_HOURS_REG)

The ALARM_HOURS_REG is used to program the hour value for the alarm interrupt. Hours are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

Figure 19-15. Alarm Hour Register (ALARM_HOURS_REG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

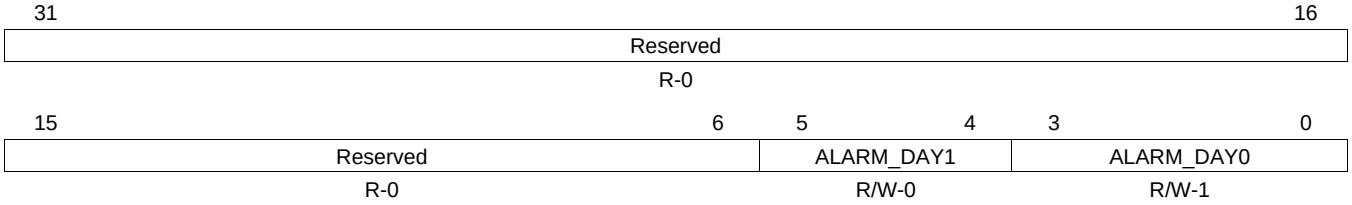
Table 19-14. Alarm Hour Register (ALARM_HOURS_REG) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7	ALARM_PM_nAM		Only used in PM_AM mode (otherwise 0)
		0	AM
		1	PM
6	Reserved	0	Reserved.
5-4	ALARM_HOUR1	0-2h	2nd digit of hours, Range is 0 to 2
3-0	ALARM_HOUR0	0-9h	1st digit of hours, Range is 0 to 9

19.3.11 Alarm Day of the Month Register (ALARM_DAYS_REG)

The ALARM_DAYS_REG is used to program the day of the month value for the alarm interrupt. Days are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

Figure 19-16. Alarm Day of the Month (ALARM_DAYS_REG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

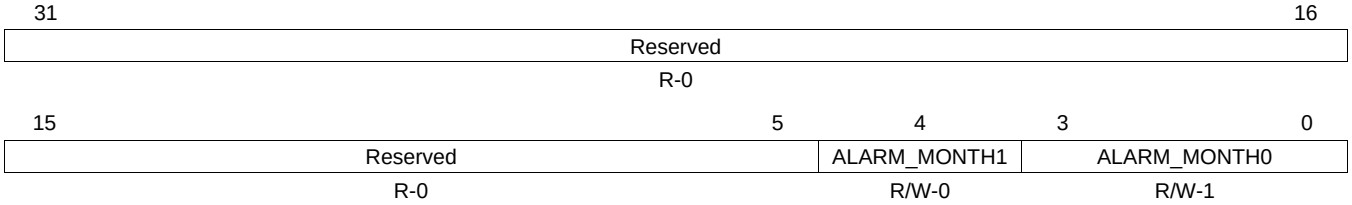
Table 19-15. Alarm Day of the Month Register (ALARM_DAYS_REG) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5-4	ALARM_DAY1	0-3h	2nd digit for days, Range from 0 to 3
3-0	ALARM_DAY0	0-9h	1st digit for days, Range from 0 to 9

19.3.12 Alarm Months Register (ALARM_MONTHS_REG)

The ALARM_MONTHS_REG is used to program the month in the year value for the alarm interrupt. The month is stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

Figure 19-17. Alarm Month Register (ALARM_MONTHS_REG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

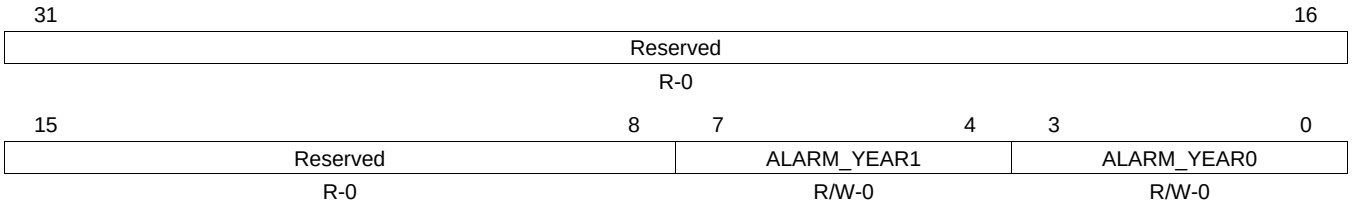
Table 19-16. Alarm Month Register (ALARM_MONTHS_REG) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	ALARM_MONTH1	0-1h	2nd digit of months, Range from 0 to 1
3-0	ALARM_MONTH0	0-9h	1st digit of months, Range from 0 to 9

19.3.13 Alarm Years Register (ALARM_YEARS_REG)

The ALARM_YEARS_REG is used to program the year for the alarm interrupt. Only the last two digits are used to represent the year and is stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

Figure 19-18. Alarm Year Register (ALARM_YEARS_REG)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-17. Alarm Year Register (ALARM_YEARS_REG) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-4	ALARM_YEAR1	0-9h	2nd digit of years, Range from 0 to 9
3-0	ALARM_YEAR0	0-9h	1st digit of years, Range from 0 to 9

19.3.14 Control Register (RTC_CTRL_REG)

The RTC_CTRL_REG contains the controls to enable/disable the RTC, set the 12/24 hour time mode, to enable the 30 second rounding feature, and to STOP/START the RTC.

Figure 19-19. Control Register (RTC_CTRL_REG)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Rsvd	RTC_DISABLE	SET_32_COUNTER	TEST_MODE	MODE_12_24	AUTO_COMP	ROUND_30S	STOP_RTC
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-18. Control Register (RTC_CTRL_REG) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved.
6	RTC_DISABLE		Disable RTC module and gate 32-kHz reference clock.
		0	RTC enable
		1	RTC disable (no 32 kHz clock)
5	SET_32_COUNTER		Set the 32-kHz counter with the value stored in the compensation registers when the SET_32_COUNTER bit is set.
		0	No action.
		1	Set the 32Khz counter with compensation registers value
4	TEST_MODE		Test mode.
		0	Functional mode
		1	Test mode (Auto compensation is enabled when the 32-kHz counter reaches its end)
3	MODE_12_24		Enable 12-hour mode for HOURS and ALARMHOURS registers.
		0	24-hr mode
		1	12-hour mode
2	AUTO_COMP		Enable oscillator compensation mode.
		0	No auto compensation
		1	Auto compensation enabled
1	ROUND_30S		Enable one-time rounding to nearest minute on next time register read.
		0	No update
		1	Time is rounded to the nearest minute
0	STOP_RTC		Stop the RTC 32-kHz counter.
		0	RTC is frozen
		1	RTC is running

NOTE:

- The SET_32_COUNTER bit must only be used when the RTC is frozen.
- The RTC_DISABLE bit must only be used to completely disable the RTC function. When this bit is set, the 32 kHz clock is gated and the RTC is frozen. From this point, resetting this bit to zero can lead to unexpected behavior. This bit should only be used if the RTC function is unwanted in the application, in order to save power.
- MODE_12_24: It is possible to switch between the two modes at any time without disturbing the RTC; read or write is always performed with the current mode.
- Auto compensation is enabled by the AUTO_COMP bit. If the COMP_REG value is positive, compensation occurs after the second change event. COMP_REG cycles are removed from the next second. If the COMP_REG value is negative, compensation occurs before the second change event. COMP_REG cycles are added to the current second. This enables it to compensate with one 32-kHz period accuracy each hour.
- The ROUND_30S bit is a toggle bit; the ARM can only write 1 and the RTC clears it. If the ARM sets the ROUND_30S bit and then reads it, the ARM reads 1 until the round-to-the-closest-minute is performed at the next second.
- The ARM can stop the RTC by clearing the STOP_RTC bit (owing to internal resynchronization, the RUN bit of the status register (RTC_STATUS_REG) must be checked to ensure that the RTC is frozen), then update TC values, and re-start the RTC by resetting the STOP_RTC bit.

19.3.15 Status Register (RTC_STATUS_REG)

The RTC_STATUS_REG contains bits that signal the status of interrupts, events to the processor. Status for the alarm interrupt and timer events are notified by the register.

Figure 19-20. Status Register (RTC_STATUS_REG)

31							16
Reserved							
R-0							

15	7	6	5	4	3	2	1	0
Reserved		ALARM	1D_EVENT	1H_EVENT	1M_EVENT	1S_EVENT	RUN	BUSY
R-0		R/W-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 19-19. Status Register (RTC_STATUS_REG) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6	ALARM	0-1	Indicates that an alarm interrupt has been generated
5	1D_EVENT	0-1	One day has occurred
4	1H_EVENT	0-1	One hour has occurred
3	1M_EVENT	0-1	One minute has occurred
2	1S_EVENT	0-1	One second has occurred
1	RUN		RTC is frozen or is running.
		0	RTC is frozen
		1	RTC is running
0	BUSY		Status of RTC module.
		0	Updating event in more than 15 µs
		1	Updating event

NOTE:

- The alarm interrupt keeps its low level until the ARM writes 1 in the ALARM bit of the RTC_STATUS_REG register.
- ALARM: This bit will indicate the status of the alarm interrupt. Writing a 1 to the bit clears the interrupt.
- 1D_EVENT1: This bit will indicate if a day event has occurred. An interrupt will be generated to the processor based on the masking of the interrupt controller.
- 1H_EVENT1: This bit will indicate if an hour event has occurred. An interrupt will be generated to the processor based on the masking of the interrupt controller.
- 1M_EVENT1: This bit will indicate if a minute event has occurred. An interrupt will be generated to the processor based on the masking of the interrupt controller.
- 1S_EVENT1: This bit will indicate if a second event has occurred. An interrupt will be generated to the processor based on the masking of the interrupt controller.
- RUN: This bit will indicate if RTC is frozen or it is running. The RUN bit shows the real state of the RTC. Indeed, because the STOP_RTC signal is resynchronized on 32-kHz clock the action of this bit is delayed.
- BUSY: This bit will give the status of RTC module. The Time and alarm registers can be modified only when this bit is 0.
- The timer interrupt is a negative edge sensitive low-level pulse.

19.3.16 Interrupt Register (RTC_INTERRUPTS_REG)

The RTC_INTERRUPTS_REG is used to enable or disable the RTC from generating interrupts. The timer interrupt and alarm interrupt can be controlled using this register.

Figure 19-21. Interrupt Register (RTC_INTERRUPTS_REG)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		IT_ALARM	IT_TIMER	EVERY	
R-0		R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 19-20. Interrupt Register (RTC_INTERRUPTS_REG) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	IT_ALARM	0	Enable one interrupt when the alarm value is reached (TC ALARM registers) by the TC registers
2	IT_TIMER		Enable periodic interrupt.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
1-0	EVERY	0-3h	Interrupt period.
		0	Every second
		1h	Every minute
		2h	Every hour
		3h	Every day

NOTE:

- The ARM must respect the BUSY period to prevent spurious interrupt. To set a period timer interrupt, the respective period value must be set in the EVERY field. For example, to set a periodic timer interrupt for every hour, the EVERY field has to be set to 2h. Along with this, the IT_TIMER bit also has to be set for the periodic interrupt to be generated.
- The IT_ALARM bit has to be set to generate an alarm interrupt.

19.3.17 Compensation (LSB) Register (RTC_COMP_LSB_REG)

The RTC_COMP_LSB_REG is used to program the LSB value of the 32 kHz periods to be added to the 32 kHz counter every hour. This is used to compensate the oscillator drift. The RTC_COMP_LSB_REG works together with the compensation (MSB) register (RTC_COMP_MSB_REG). The AUTO_COMP bit in the control register (RTC_CTRL_REG) must be enabled for compensation to take place.

Figure 19-22. Compensation (LSB) Register (RTC_COMP_LSB_REG)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		RTC_COMP_LSB	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-21. Compensation (LSB) Register (RTC_COMP_LSB_REG) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RTC_COMP_LSB	0-FFh	Indicates number of 32-kHz periods to be added into the 32-kHz counter every hour

NOTE: This register must be written in two's complement. That means that to add one 32-kHz oscillator period every hour, the ARM must write FFFFh into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG. To remove one 32-kHz oscillator period every hour, the ARM must write 0001h into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG. The 7FFFh value is forbidden.

19.3.18 Compensation (MSB) Register (RTC_COMP_MSB_REG)

The RTC_COMP_MSB_REG is used to program the MSB value of the 32 kHz periods to be added to the 32 kHz counter every hour. This is used to compensate the oscillator drift. The RTC_COMP_MSB_REG works together with the compensation (LSB) register (RTC_COMP_LSB_REG) to set the hourly oscillator compensation value. The AUTO_COMP bit in the control register (RTC_CTRL_REG) must be enabled for compensation to take place.

Figure 19-23. Compensation (MSB) Register (RTC_COMP_MSB_REG)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		RTC_COMP_MSB	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-22. Compensation (MSB) Register (RTC_COMP_MSB_REG) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RTC_COMP_MSB	0-FFh	Indicates number of 32-kHz periods to be added into the 32-kHz counter every hour

NOTE: This register must be written in two's complement. That means that to add one 32-kHz oscillator period every hour, the ARM must write FFFFh into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG. To remove one 32-kHz oscillator period every hour, the ARM must write 0001h into RTC_COMP_MSB_REG and RTC_COMP_LSB_REG. The 7FFFh value is forbidden.

19.3.19 Oscillator Register (RTC_OSC_REG)

The RTC_OSC_REG is used to program the oscillator resistance value and the SWRESET bit can be used to reset the RTC. This is the only way to reset the RTC.

Figure 19-24. Oscillator Register (RTC_OSC_REG)

31	16
Reserved	
R-0	

15	6	5	4	3	0
Reserved		SWRESET	OSC32KPWRDNR	SWRESPROG	
R-0		R/W-0	R/W-0(H/W)	R/W-7h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-23. Oscillator Register (RTC_OSC_REG) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5	SWRESET	0-1	Software reset bit
4	OSC32KPWRDNR	0-1	Control of 32 kHz Oscillator powerdown
3-0	SWRESPROG	0-Fh	Value of the oscillator resistance

NOTE: The SWRESET bit is set to 1 to reset the RTC module. This bit is self-clearing, and is always read as 0. CPU must care of interrupt handling before RTC is reset. After setting the SWRESET bit, you must not access any of the RTC registers for three 32-kHz clock cycles.

19.3.20 Scratch Registers (RTC_SCRATCHx_REG)

The RTC_SCRATCH_REG is used to hold some required values for the RTC register.

Figure 19-25. Scratch Registers (RTC_SCRATCHx_REG)

31	0
RTCSCRATCHx	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 19-24. Scratch Registers (RTC_SCRATCHx_REG) Field Descriptions

Bit	Field	Value	Description
31-0	RTCSCRATCHx	0-FFFF FFFFh	Scratch registers, available to program

19.3.21 Kick Registers (KICK0R, KICK1R)

The kick registers (KICKnR) are used to enable and disable write protection on the RTC registers. Out of reset, the RTC registers are write-protected. To disable write protection, correct keys must be written to the KICKnR registers.

Kick0 Register (KICK0R)

The Kick0 register allows writing to unlock the kick0 data. To disable RTC register write protection, the value of 83E7 0B13h must be written to KICK0R, followed by the value of 95A4 F1E0h written to KICK1R. RTC register write protection is enabled when any value is written to KICK0R.

Figure 19-26. Kick0 Register (KICK0R)

31	0
KICK0	
W-0	

LEGEND: W = Write only; -*n* = value after reset

Table 19-25. Kick0 Register (KICK0R) Field Descriptions

Bit	Field	Value	Description
31-0	KICK0	0	Kick0 data

Kick1 Register (KICK1R)

The Kick1 register allows writing to unlock the kick1 data and the kicker mechanism to write to other MMRs. To disable RTC register write protection, the value of 83E7 0B13h must be written to KICK0R, followed by the value of 95A4 F1E0h written to KICK1R.

Figure 19-27. Kick1 Register (KICK1R)

31	0
KICK1	
W-0	

LEGEND: W = Write only; -*n* = value after reset

Table 19-26. Kick1 Register (KICK1R) Field Descriptions

Bit	Field	Value	Description
31-0	KICK1	0	Kick1 data

19.3.22 RTC Revision Register (RTC_REVISION)

Figure 19-28. RTC Revision Register (RTC_REVISION)

31	30	29	28	27	16
SCHEME		Reserved		FUNC	
R-1		R-0		R-EB0h	

15	11	10	8	7	6	5	0
R_RTL		X_MAJOR		CUSTOM		Y_MINOR	
R-1		R-1		R-0		R-4h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-27. RTC Revision Register (RTC_REVISION) Field Descriptions

Bit	Field	Value	Description
31-30	SCHEME	0-3h	Used to distinguish between old scheme and current scheme.
29-28	Reserved	0	Reserved
27-16	FUNC	0-FFFh	Function indicates a software compatible module family.
15-11	R_RTL	0-1Fh	RTL Version (R)
10-8	X_MAJOR	0-7h	Major Revision
7-6	CUSTOM	0-3h	Indicates a special version for a particular device.
5-0	Y_MINOR	0-3Fh	Minor Revision (Y)

19.3.23 System Configuration Register (RTC_SYSCONFIG)

Figure 19-29. System Configuration Register (RTC_SYSCONFIG)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		IDLEMODE	
R-0		R/W-2h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-28. System Configuration Register (RTC_SYSCONFIG) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	IDLEMODE	0-3h	Configuration of the local target state management mode, By definition target can handle read/write transaction as long as it is out of IDLE state.
		0	Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally, regardless of the IP module internal requirements; Backup mode, for debug only.
		1h	No-idle mode: local target never enters idle state, Backup mode, for debug only.
		2h	Smart-idle mode: local target's state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements, IP module shall not generate (IRQ- or DMA-request-related) wakeup events.
		3h	Smart-idle wakeup-capable mode: local target's idle state eventually follows (acknowledges the system's idle requests, depending on the IP module's internal requirements, IP module may generate (IRQ- or DMA-request-related) wakeup events when in idle state, Mode is only relevant if the appropriate IP module "swakeup" output(s) is (are) implemented.

19.3.24 Wakeup Enable Register (RTC_IRQWAKEEN_0)

Figure 19-30. Wakeup Enable Register (RTC_IRQWAKEEN_0)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		ALARM_WAKEEN	TIMER_WAKEEN
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-29. Wakeup Enable Register (RTC_IRQWAKEEN_0) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	ALARM_WAKEEN		Wakeup generation for event Alarm.
		0	Wakeup disabled
		1	Wakeup enabled
0	TIMER_WAKEEN		Wakeup generation for event Timer.
		0	Wakeup disabled
		1	Wakeup enabled

Serial ATA (SATA) Controller

This chapter describes the Serial ATA Controller (SATA) in the device.

Topic **Page**

20.1 Introduction

The serial ATA is the successor of the Parallel ATA/ATAPI controller that has served as the choice of the communication medium between a portable computer (PC) and a hard-disk drive for several decades. During these times, the PATA interface has gone through changes to sustain the demands of the newly emerging applications needs and eventually reached its technical limitation creating a throughput bound. The PATA controller reached a point where it required major changes to satisfy the upcoming applications requirements and this led to its successor, the birth of the SATA controller. The SATA controller was designed to use the same logical command structures that PATA uses but with a whole new physical characteristic. The SATA controller makes use of 2 pairs of high-speed differential conductors as opposed to the parallel 16 bit low-speed interface. The device has a built in SATA controller with a dual HBA port operating in AHCI mode and is used to interface to data storage devices at both 1.5 Gbits/second and 3.0 Gbits/second line speeds per HBA port. AHCI describes a system memory structure which contains a generic area for control and status, and a table of entries describing a command list where each command list entry contains information necessary to program an SATA device, and a pointer to a descriptor table for transferring data between system memory and the device.

20.1.1 Purpose of the Peripheral

The SATA controller addresses the drawback of the PATA interface architecture, throughput, and protocol perspectives. PATA required 40/80 wire parallel cable with a length requirement not exceeding 18 inches. With the latest/final PATA hardware design, PATA's maximum transfer rate saturated to a 133 Mbytes/second of transfer. In addition to newly added features, like hot swapping and native command queuing, the SATA controller abandoned the parallel physical interface and transitioned to a high-speed serial format using two differential pairs supporting up to 3 Mbits/second transfer rate, translating to a 300 Mbytes/second raw throughput per HBA port.

With the intent on handling roles of the PATA controller and addressing the future needs, SATA controller is architected with the option of operating in a Legacy mode; a mode that behaves similar to the PATA controller from the protocol/driver perspective, and a new AHCI mode that is different from the Legacy mode allowing it to overcome the drawbacks of PATA protocol as well; extending PATAs capabilities. The SATA controller that is supported by the device supports AHCI mode of operation only. That is, the AHCI controller supported has no support for Legacy mode of operation. AHCI is a PCI class device that acts as a data movement engine between system memory and Serial ATA devices. However, the AHCI controller is integrated within the core chipset which usually is a common attribute for embedded devices.

Communication between a device and software moves from the PATA task file registers access made via byte-wide accesses to a structure based, command Frame Information Structure (FIS), located in system memory that is fetched by the HBA. This reduces command setup time significantly, allowing for many more devices to be added to a single host controller port (up to 15 of them via the use of a hardware Port Multiplier). Software no longer communicates directly to a device via the task file. In other words, all data transfers between the device and system memory occur through the HBA acting as a bus master to system memory. Whether the transaction is of a DMA type or a PIO type (the use of the PIO command type is strongly discouraged and all transfers should be performed using DMA unless a transaction is only performed via PIO command, which in this case is still done via the AHCI master port not the CPU), the HBA fetches and stores data to memory, offloading the CPU. No data port exists for moving data in and out of the system memory similar to the PATA offerings. Software written for AHCI is not allowed to utilize any of the legacy mechanisms to program devices.

The SATA controller uses a less massive thinner flexible cable that can be up to 3 feet (1 meter) in length allowing for easier routing and better air ventilation inside a case. Its power budget is significantly reduced to 250 mV compared to the required 5V and/or 3.3V power of PATA.

Like its predecessor the SATA controller is most commonly used by PCs, portable devices, and embedded devices to interface a host processor with external data storage or CD/audio devices. It also has the support for hot swapping capability and the ability of interfacing to a Port Multiplier (PM) to increase the number of devices that can be attached to a single HBA port by as many as fifteen devices. In other words, with the use of two PMs, a total of 30 devices can be connected to the two available HBA ports. Note that in the case where a PM is used, the bandwidth offered by a single HBA port will be shared amongst the total number of devices attached to the Port Multiplier. The device has two HBA ports rendering aggregated 6Gbits/second throughput of raw data capability.

20.1.2 Features Supported

The main features of the SATA controller are:

- Support for Synopsys DWH Serial ATA 1.5Gbps and 3Gbps speeds core
- Supports AHCI Controller (specification version 1.1)
- Support two HBA Ports
- Integrated TI SERDES PHY
- Built in/integrated Rx and Tx data buffers
- Supports all SATA power management features
- Internal DMA engine per port
- Hardware-assisted native command queuing (NCQ) for up to 32 entries
- 32-bit addressing
- Supports port multiplier with command-based switching
- Activity LED support (one for each Port)

20.1.3 Features Not Supported

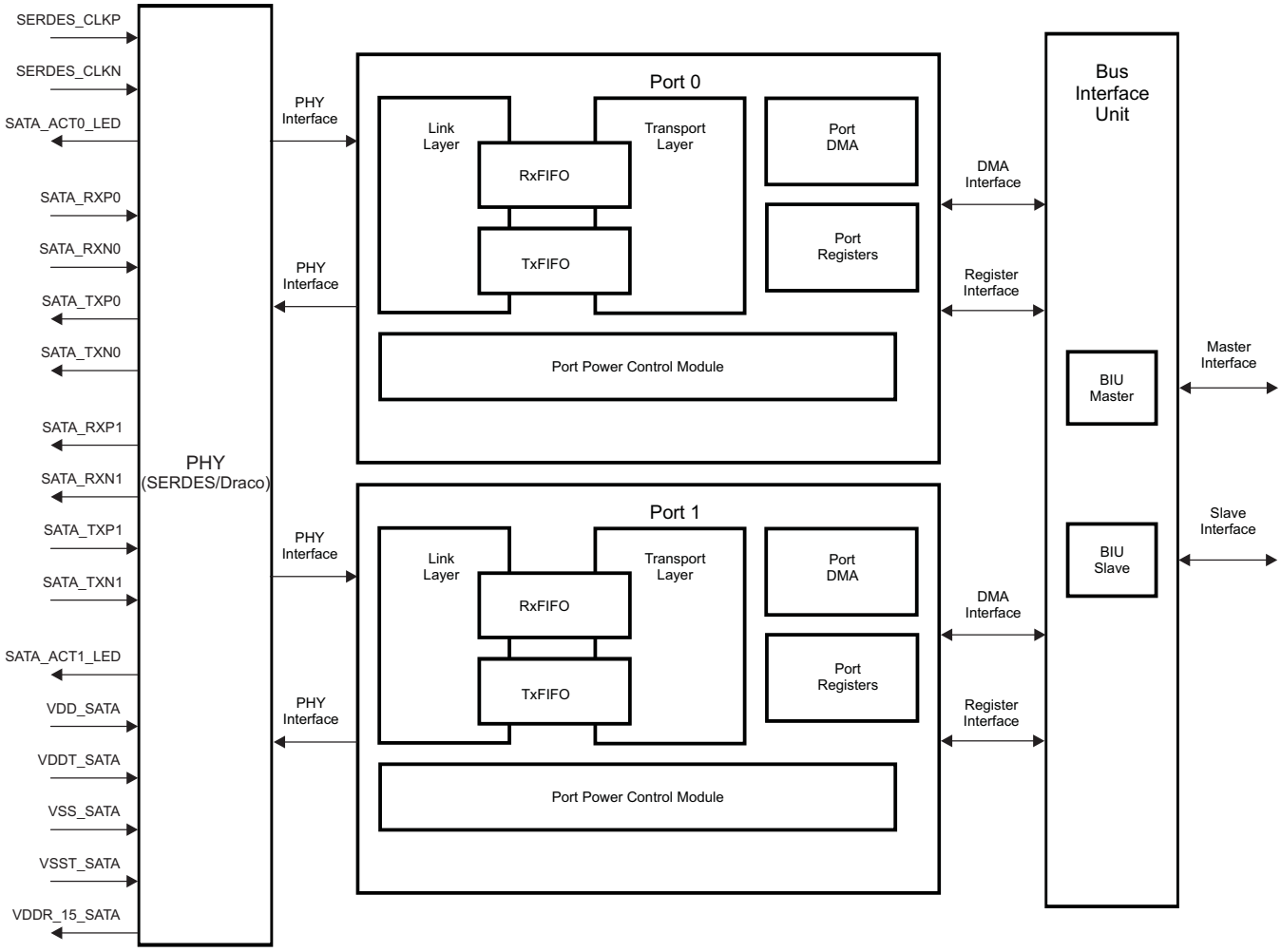
Features not supported in this SATA controller are:

- Legacy mode of operation
- Master/slave type of configuration
- Far-end analog loopback
- Message signaled interrupts
- 64-bit addressing
- Dedicated Mechanical Presence Switch Pin (GPIO usage is expected to supplement this task if necessary)
- Dedicated Cold Presence Detect Pin (GPIO usage is expected to supplement this task if necessary)

20.1.4 Functional Block Diagram

The SATASS is a fully contained Serial ATA host with built in DMA. It uses the AHCI standard for communication with a SATA device. It has no support for the Legacy mode of operation. Figure 20-1 shows a high level block diagram of the SATA Subsystem (core and the integrated TI PHY (SERDES)).

Figure 20-1. SATA Core Block Diagram



20.1.5 Industry Standard(s) Compliance

The SATA Subsystem complies with the following industry standards:

- SATA revision 2.6 Gold standard.
- AHCI revision 1.1 specification.

20.1.6 Non-Industry Standard(s) Compliance

The SATA Subsytem complies with the following proprietary standards:

- Synopsys Design Ware Core DWC AHCI Controller Version 1.30.
- TI SERDES PHY: wiz7c2xxn5x1px (Draco) Macro specification Revision 01.02.01.

20.1.7 Terminology Used in this Document

The following is a brief explanation of some terms used in this document:

AHCI — Advanced Host Controller Interface (necessary for implementing newly supported features like native command queuing). The device supports only this mode of operation and has no support for Legacy mode. It also reduces CPU/Software overhead when moving data in and out of the system memory. Note that system software is responsible to ensure that queued and non-queued commands are not mixed in the command list.

ATA/ATAPI— Advanced Technology (AT) attachment/ATA packet interface

CF— Compact Flash

Device— External SATA or ATAPI device attached to the SATA controller

DMA— DMA within the ATA controller, not the processor EDMA system

DWORD— DWORD is 32 bits of data.

Command List— Command List is a memory buffer for as much as 32 commands. Each command is entered in a command slot. This is a required feature when supporting command queuing.

Command Slot— A command slot is a subset of the command list. It is the memory buffer for a single command. A total of 32 command slots exist.

D2H— D2H is an acronym for “device to HBA” and is mostly used to indicate the direction of the transmission of FIS which is from device to the HBA (host).

FIS— FIS is an acronym for “Frame Information Structure”. A frame is made of a data payload with computed CRC and a start and end primitives.

H2D— H2D is an acronym for “HBA to device” and is mostly used to indicate the direction of the transmission of FIS which is from HBA (host) to device.

HBA — HBA is an acronym for “host bus adapter”. It is the SATA controller that implements the AHCI specification to communicate between system memory and Serial ATA devices.

Legacy Mode— This mode of operation makes the user access (application software or driver) to view the SATA controller in a similar fashion as a PATA controller. The device does not support Legacy mode of operation but AHCI mode of operation.

OOB — Out of Band. OOB signaling is used for during device detection and when recovering from power states.

PM — PM is an acronym for ‘Port Multiplier”. A Port Multiplier allows extending an HBA port connection capability to connect to multiple SATA Devices, a maximum of 15 Devices. Note that the operating bandwidth is shared amongst all the Devices when using this type of configuration.

PMP — Port Multiplier Port : A Port of a Port Multiplier (PM).

PORT— Refers to HBA Port mainly, within this document. However, the PORT is also be used to refer to a port of a PORT Multiplier.

PRD — PRD is an acronym for “physical region descriptor”. A PRD table is a data structure used by DMA engines that comply with the ATA/ATAPI Host Adapters standard. The PRD describes memory regions to be used as the source or destination of data during DMA transfers. A PRD table is often referred to as a scatter/gather list.

SATA Controller— Serial ATA controller, also HBA.

System Memory— The memory that is external from the SATA controller but is a memory that is accessible by the built in SATA controller DMA or the CPU.

20.2 Architecture

This section discusses the architecture of the Serial ATA Controller. The Serial ATA Controller supports Advanced Host Controller Interface (AHCI). The dual port SATA controller does not support the Legacy mode of operation. Since the controller complies with the AHCI standard (version 1.1) the details of its operation is captured within the AHCI version 1.1 specification. Please consult the specification for the general behavior of the SATA Core operation. The SATA controller supported within the device has two HBA ports.

20.2.1 Clock Control

The SATA controller makes use of one internal 250 MHz clock (SYSCLK5) for both functional and keep alive functions. During a Low Power down event, the keep alive clock is necessary to insure portion of the controller interface to remain up and running at all times keeping controller context in tact as well as clocking the h/w logic necessary used in detecting wake up events from power down modes. Access to the SATA controller and resources are implemented through, SYSCLK5 derived from the output of the main input 27MHz clock source. SYCLK5 to the SATA controller is gated by the PRCM (Power, Reset, and Clock Management) at the controller boundary and is required to be enabled prior to accessing the SATA controller. However, the keep alive clock is always ON, it can not be gated at the controller boundary, and can not be turned off so long as the device is in normal operation state.

A high quality low jitter external differential clock is required as a source clock input to the PHY and the frequency of the input clock should be between 60MHz and 375 MHz (depending upon the supported multiplier used). It is recommended to keep the input clock differential frequency selection to a frequency value that is desirable to both SATA and PCI Express peripherals and is highly recommended for the use of a 100 MHz input differential clock source since this input clock source frequency is expected by most, if not all, of the PCIe systems.

This input frequency requirement is dependent upon the supported PHY PLL multiplier value. A single PLL within the SERDES device exists that is shared amongst the two HBA Ports. The control for this PLL and PHY operation is done via P0PHYCR register. That is, the programming done for the clock setting using Port 0 PHY Control Register applies to both ports and the respective field within P1PHYCR is reserved. See P#PHYCR (# = 0 or 1) register field descriptions for more detail. The MPY field of the P0PHYCR register is programmed with the PLL multiplier value based on the input frequency clock. Note that the PHY PLL output frequency should be exactly 1.5GHz (for both 3 and 1.5 Gbits/second line rate) and its accuracy is very important to the operation of the SATA controller.

Table 20-1 shows the MPY bit field of P#PHYCR for supported PHY PLL multiplier values.

Table 20-1. MPY Bit Field of P0PHYCR

MPY Bit Field Value	Effect
0	4x
1h	5x
2h	6x
3h	8x
4h	8.25x
5h	10x
6h	12x
7h	12.5x
8h	15x
9h	16x
Ah	16.5x
Bh	20x
Ch	22x
Dh	25x
Eh	Reserved
Fh	Reserved

20.2.2 Signal Description

The device has bonded out the Data lines (two sets of differential data lines) and device activity per port indicators. Signals needed to detect device detection and attached device power indicators need to be implemented via GPIO if desired and no dedicated signals for these tasks exist. The power pins and logic necessary to power up a cold SATA device should be handled external to the device.

Table 20-2 summarizes the available SATA module signals.

Table 20-2. SATA Interface Signal Descriptions

Terminal Name	Direction from the HBA Perspective (In/Out)	Description
SATA_RXP0	Input	Receive Data Positive Differential Signal for Port 0
SATA_RXN0	Input	Receive Data Negative Differential Signal for Port 0
SATA_TXP0	Input	Transmit Data Positive Differential Signal for Port 0
SATA_TXN0	Input	Transmit Data Positive Differential Signal for Port 0
SATA_RXP1	Input	Receive Data Positive Differential Signal for Port 1
SATA_RXN1	Input	Receive Data Positive Differential Signal for Port 0
SATA_TXP1	Input	Transmit Data Positive Differential Signal for Port 1
SATA_TXN1	Input	Transmit Data Negative Differential Signal for Port 1
SERDES_CLKP	Input	PHY Reference Positive Differential Signal
SERDES_CLKN	Input	PHY Reference Negative Differential Signal
SATA_ACT0_LED	Output[1]	Device Activity Indicator for HBA Port 0
SATA_ACT1_LED	Output[2]	Device Activity Indicator for HBA Port 1
VDD_SATA		
VSS_SATA		
VDDT_SATA		
VSST_SATA		

(1) Multiplexed with GPIO 30.
(2) Multiplexed with GPIO 31.

20.2.3 DMA

Each HBA ports contain two DMA engines. One of the DMA engine is used to fetch command from the command list. The other DMA is used to move FISes in and out of system memory

The DMA used to move FISs in and out of system memory has a register, P#DMACR (one for each HBA port where # = 0 or 1), to control the burst transfer. This DMA is used to transfer all information between system memory and the attached SATA device, as well as configuration and status FISs.

The user can program the maximum burst size that will be issued on the system bus independently for both reads and writes. The DMA will issue transactions equaling the programmed size or smaller (in DWORD increments). This can be used to optimize burst size for over-all system throughput efficiency. Refer to Section 20.4.33 for details on legal values. Note that programming a burst size of greater than a transaction size (below), while is not invalid, is meaningless because the DMA maximizes out at the maximum transaction size.

The user can also program the transaction size for both receive and transmit (see Section 20.4.33). The transaction size is the minimum amount of data that the DMA will work on. For example, if there is an FIS coming from the device to the host, the DMA will not begin transferring data into system memory until there is at least RX_TRANSACTION_SIZE (RXTS) data in the receive FIFO. During transmit, the DMA will read data from system memory in TX_TRANSACTION_SIZE (TXTS) increments to put into the transmit FIFO. Note that transactions may be broken up into multiple bursts based on burst size, crossing of a 1k boundary, or end-of-frame.

20.2.4 Transport Layer

The transport layer handles all of the transport layer functions of the SATA protocol. During reception, it receives a FIS from the Link layer via the Rx FIFO, decodes the type, and routes it to the proper location via the port DMA. During transmission, it transfers a FIS constructed by the port DMA to the link layer via the Tx FIFO. It also passes link layer errors and checks for transport layer errors to pass up to the system.

20.2.5 FIFOs

The transport layer also contains the Tx and Rx FIFOs. These FIFOs are used as asynchronous data buffers between the serial domain and the bus clock domain. The size of these FIFOs affects the subsystems ability to buffer data before flow control must be asserted. It also affects the maximum programmable transaction and burst sizes that can be programmed into the port DMA. The Tx FIFO size is 64 DWORDS (256 bytes) deep while the Rx FIFO size is 128 DWORDS (512 bytes) deep.

20.2.6 Link Layer

The link layer maintains the link and supports all SATA link layer functionality including:

- Out-of-band (OOB) transmit signaling
- Frame negotiation and arbitration
- Envelope framing/de-framing
- CRC calculation (receive and transmit)
- 8b/10b encoding/decoding
- Flow control
- Frame acknowledgment and status
- Data width conversion
- Data scrambling/descrambling
- Primitive transmission
- Primitive detection and dropping
- Power management

20.2.7 PHY

The SATASS includes an integrated TI SERDES macro as a PHY. The PHY handles all of the serialization/de-serialization, symbol alignment, and Rx OOB signal detection.

20.2.8 Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package; therefore, some of the SATA controller peripheral signals share SERDES clock input pins (SERDES_CLKP/N) with the PCI Express peripheral and GPIO Module (GPIO30/31). Since the same SERDES is used by SATA and PCI Express peripherals, it is recommended the use of a 100 MHz differential input clock source that is ideal for both peripherals. Refer to the device-specific data manual to determine how pin multiplexing affects the SATA.

20.2.9 Power Management

The SATA controller can be placed in reduced power modes to conserve power during periods of no activity or no use. The main power management of the peripheral is controlled by the Power, Reset, and Clock Management (PRCM) unit. The PRCM acts as a master controller for power management of all of the peripherals on the processor. For detailed information on power management procedures using the PRCM, see the *Power, Reset, and Clock Management (PRCM) Module* chapter.

During times that the SATA peripheral is in use, the SATASS supports the industry standard power down modes (both Partial and Slumber low power modes) as provided within the SATA specification. These modes allow for power savings by powering down part of the SERDES PHY and by providing the ability to gate off the clocks to the link layer. The Port Power Control Module is used to enter and exit this power down modes (that is power down mode is controlled at Port level) which may have the normal functional clocks gated off.

NOTE: When SATA communication is in the idle state, that is, when no disk activity takes place, the communication remains active with both the host and the device sending a logical sync primitive and scrambled data at the speed negotiated continuously. For power sensitive applications, the power consumed during the disk inactivity stage might be undesirable and it might be a desired task to place the communication interface into an electrical idle (Partial or Slumber) state until data transfer activity is needed in order to conserve power.

20.2.10 Reset

The SATA controller reset is handled via PRCM. Please consult the PRCM document for details on how to use the PRCM module. Other types of resets (HBA Reset, Port Reset, and Software Reset) supported by the SATA controller are part of the AHCI specification. See the AHCI Standard Specification 1.1 for details on HBA Reset, Port Rest, and Software Reset.

20.2.11 Interfacing to Single and Multiple Devices

The SATA controller supports dual HBA ports. This means that each HBA port can be used to interface directly to a single SATA device or to multiple SATA Devices via a Port Multiplier. Note that on a multiple target setup, the available bandwidth will be shared amongst all the attached devices.

Note that Port Multipliers and Power Supplies for external SATA devices, when needed, should be furnished external to the device. No dedicated signals for controlling power and device detection are not bonded out and usage of GPIO pins can be used to supplement this task. However, the controller supports the capability to spin-up-attached devices independently.

20.2.11.1 Interfacing to a Single Device

If need to interface directly to a Single Device, there is no need of populating a Port Multiplier. The software needs to ensure that the PMP field within the Command Header and the PM_PORT field of the FIS be cleared at all times.

If the external SATA device is not Self Powered, it is the responsibility of the System Designer to populate and furnish the right power supply needed. Consult the SATA Specification for details.

20.2.11.2 Interfacing to Multiple Devices

Interfacing to multiple devices is pretty much identical to interfacing with a single device except the addition of a hardware Port Multiplier and initialization on software side. User software is required to perform the additional task of detecting and configuring the PM prior to accessing the attached devices. The user software is required to populate the Port Multiplier Port (PMP) field of the FIS in order to select one of the fifteen devices that could be connected and accessed by the HBA. The details on this are fully captured within the AHCI specification Version 1.1.

20.2.12 Initialization

Proper initialization of the HBA is required after power up to ensure proper operation of the SATA controller peripheral. The initialization process starts by performing a write to one-time write only registers (this initialization step is documented as Firmware initialization within the AHCI specification) where the values programmed depend on the features that the applications support. Note that the features that are enabled by the Firmware initialization are subsets of the features supported by the SATA subsystem. This Firmware initialization is similar to what a PC BIOS does and allows the user to enable/disable some features by using software.

The software can then continue with the normal initialization that is required by the software. The details and sequence of initialization is documented within the AHCI specification. In general, everything that is achieved by the Software Initialization has to do with configuring and furnishing resources needed by the AHCI controller and these tasks include PHY Initialization, allocating structures and memories for Command Slots, FIS, and Data Memories and concludes by enabling the receive FIS DMA. The software will then Spin-Up the Device and ensure that a proper Device Detection and Speed Negotiation has completed prior to enabling the Command DMA.

Note that DMA Configuration/Initialization should take place after Device Detection and Speed Negotiation. If done earlier, the default value is used (which is the recommended setting) since RESET removes the user programmed values. The only time it is advisable to change the DMA Configuration is when you need to prioritize System Resource access. This still has to be done after "PHY Ready" status is set (in other words, after the device detection and speed negotiation has completed). It also requires that the Command DMA is not running (P0CMD.ST = 0) when modifying the value of the DMA Configuration fields.

20.2.12.1 Initialization (Firmware and Software)

Software reads the HBA capabilities register (CAP), ports implemented register (PI), AHCI version register (VS), global parameter 1 register (GPARAM1R), global parameter 2 register (GPARAM2R), and the port parameter register (PPARAMR) to obtain information about the subsystem's capabilities. The software should then take the following steps to configure each port for operation:

1. Do all firmware capability writes.
2. Setup all appropriate structures in memory as per the AHCI specification.
3. Configure the PHY using the port PHY control register (P#PHYCR):
   (a) Set the MPY bit field for the PLL multiply factor (multiply value selected should target a 1.5-GHz frequency that is good enough for both GEN1 and GEN2 speeds).
   (b) Set LOS = 1 (enable loss of signal detection)
   (c) Set ENPLL = 1 (enable the PLL)
4. Set the port command list base address register (P#CLB).
5. Set the port FIS base address register (P#FB).
6. Set appropriate bits in the port command register (P#CMD).
7. Program the port serial ATA control register (P#SCTL).
8. Wait for Device Detection and Speed Negotiation to end.
9. Program the port DMA control register (P#DMACR).
10. Enable the appropriate interrupts.
11. Enable FIS reception in P#CMD.
12. Spin-up the device(s), if necessary.

20.2.12.2 Issuing a Command

Once the host and device are configured, perform the following steps to issue a command:

1. Create the appropriate FIS in system memory.
2. Create the PRD.
3. Queue the command to the command queue list (location specified by the port command list base address register (P#CLB).

For detailed information, see the AHCI Specification Version 1.1.

20.2.13 Interrupt Support

The AHCI controller supports both standard interrupt sourcing, where interrupts are generated when enabled events occur, or a different type of method of generating interrupts that minimize interrupt loading by either generating interrupts in a batch or periodically. The latter method is handled using Command Completion Coalescing method and the details is captured within the AHCI Specification Version 1.1.

20.2.13.1 Command Completion Coalescing

Command Completion Coalescing (CCC) is a feature designed to reduce the interrupt and command completion overhead in a heavily loaded system. The feature enables the number of interrupts taken per completion to be reduced significantly, while ensuring a minimum quality of service for command completions. When software specified number of commands have completed or a software specified timeout has expired, an interrupt is generated by hardware to allow software to process completed commands. The command completion coalescing ports register (CCC_PORTS) should be programmed by setting the corresponding bit of the Port that is to be included in command completion coalescing feature. Note that the device supports 2 HBA ports.

For a detailed explanation of the CCC initialization and usage, see the AHCI Specification 1.1 Section 11.6.

20.2.13.1.1 CCC Interrupt Based on Timer Expiration

When CCC is enabled and the desired method to receive an interrupt is based on a timer elapse condition, then the user needs to communicate a resolution for a 1ms time by programming the TIMER1MS register with the OCP cycle count derived from the OCP clock frequency sourced to the SATA controller.

As an example, if a user desires for interrupt to be generated every 15 ms, for OCP bus Clock frequency of 250 MHz, the 1-ms cycle count should be programmed with a value of 250 000, that is, 250 MHz/1000 = 250 000, and CCC_CTL.TV is programmed with a non-zero value (15 in this case). When CCC_CTL.EN is set to 1 (CCC is enabled), the CCC will periodically generate interrupt every 15 ms or every 15 × 250 000 = 3,750,000 OCP cycles.

NOTE: Make sure the EN bit in the command completion coalescing control register (CCC_CTL) is cleared to 0 (CCC is disabled, prior to programming this field).

20.2.13.1.2 CCC Interrupt Based on Completion Count

When CCC is enabled and the desired method to receive an interrupt is based on a completion count, that is, the CC bit in the command completion coalescing control register (CCC_CTL) is programmed with a non-zero value and the CCC interrupt is enabled (EN bit in CCC_CTL is set to 1), an interrupt is sourced from the SATA controller when the programmed desired number of interrupt is received.

NOTE: Make sure the EN bit in the command completion coalescing control register (CCC_CTL) is cleared to 0 (CCC is disabled, prior to programming this field).

20.2.13.2 Non CCC Interrupt Configuration

For a standard interrupt handling method where every event that is enabled generates an interrupt, is handled as follows. For more information, see the AHCI Specification.

After insuring that CCC is disabled, the EN bit in the command completion coalescing control register (CCC_CTL) is 0, in order for the SATA Core to source interrupts, the interrupt should be enabled at both global level (the IE bit in the global HBA control register (GHC) is set to 1) and port level by enabling the bit fields for the desired interrupt. An enable bit at a Port level controls corresponding interrupt dispatch to the processor interrupt handling resource. So long as the CPU interrupt handler is configured properly, the CPU receives the interrupt when the enabled event occurs.

20.2.14 EDMA Event Support

The SATA controller makes use of its own built-in DMA and has no need for the use of the processor EDMA.

20.3 Use Cases

The following sections include some sample program snippets that can be used as a guide for software development. The example demonstrates one of the ways of creating the necessary structures; properly aligned system memory resources, initialization, as well as performing basic DMA Read/Write transfer using a couple of the Command Slots.

Section 20.3.1 contains examples in relations to System Memory resource allocations, Structures, and Subroutines used by the Initialization and Read/Write Transfer functions. The remaining sections include examples of basic Initialization, DMA Write transfer, and DMA Read transfer examples using Port 0.

20.3.1 General Utilities: Structures and Subroutines Sample Program Uses

```
/* Allocating memory for Command List. The structure pointed to by this
   address range is 1K-bytes in length and must be 1K-byte aligned.
   Note that each command header occupies 32 bytes of memory and 32
   command headers require 1024 bytes of memory.
*/
#pragma DATA_SECTION(CmdLists,     OCMCRAM0);
#pragma DATA_ALIGN(CmdLists, 1024);
CmdListHeader CmdLists[32]={0};

/* Indicates the 32-bit physical address of the command table, which
   contains
       the command FIS,
       ATAPI Command,
       andPRD table.
   This address must be aligned to a 128-bytes of memory,
*/
#pragma DATA_SECTION(CmdTable,     OCMCRAM0);
#pragma DATA_ALIGN(CmdTable, 128);
CommandTable CmdTable[LISTLENGTH];

/* Indicates the 32-bit base physical address for received FISes. The
   structure pointed to by this address range is 256 bytes in length
   and must be 256-byte aligned.
*/
#pragma DATA_SECTION(RcvFis,       OCMCRAM0);
#pragma DATA_ALIGN(RcvFis, 256);
ReceiveFis RcvFis;

#pragma DATA_SECTION(prdTableDataBuff,     OCMCRAM0);
unsigned char prdTableDataBuff[LISTLENGTH][PRDLENGTH][DATABUFFERLEN];


#define NUMOFPORTS       (2)  // Supports Two HBA Ports. However it can support up to
                              // 15 additional Ports, per HBA Port, for a combined total of 30
Port Multiplier
                              // Ports (PMP) capable of attaching to 30 devices.
#define LISTLENGTH       (2)  // Max Command Header Per Port is 32

#define WRITE_CMD_SLOT   (0)  // Value used here should be <= LISTLENGTH-1
#define READ_CMD_SLOT    (1)  // Value used here should be <= LISTLENGTH-1

        // WARNING. PRDLENGTH can not be greater than 8 for this program.
        //  See Note captured by the area when memory has been reserved for within
        //  sata_utilities.c for Command Table "CmdTable" for more information.

#if 1
    #define _MAX_DATA_TRANSFER_     // Define this in project file when needed.
#endif
```

```
#ifndef _MAX_DATA_TRANSFER_             // 512 Bytes Data Size within 2 PRD Descriptors.
    #define PRDLENGTH     (2)       // Max PRD Length is 65535 per port.
    #define DATABUFFERLEN (256)     // DMA Data Buffer Length
#else // Max Data Size Transfer 8K Bytes within 2 PRD Descriptors
    #define PRDLENGTH     (2)       // Max PRD Length is 65535 per port.
    #define DATABUFFERLEN (2*4096)  // DMA Data Buffer Length
#endif

#if ((PRDLENGTH > 8) | (WRITE_CMD_SLOT > LISTLENGTH-1) | (READ_CMD_SLT > LISTLENGTH-1))
#error PRDLENGTH ENTRY ERROR - PROGRAM HARD CODED FOR MAX VALUE OF 8 - CMD SLOT ENTRY ERROR
#endif


#define DESIRED_SPEED         (GEN1)         // GOASFASTASDEVICE, GEN1, GEN2
#define DEVICE_LBA_ADDRESS    (0x00000002)   // Dev28bitLbaAddress = 28-Bit LBA Address
#define WAIT_500_MILLISECONDS (50)           // This should be set to 500 once the ONE_MS_VALUE is
                                                programmed correctly.
#define WAIT_1_MILLISECOND    (1)
#define ONE_MS_VALUE          (1)            // Number of CPU Cycles needed to generate a
                                                millisecond wait time.
#define DMA_BURST_LENGTH      (0x9)          // [0x0 - 0x9] Burst=2^(-1) i.e., 0x8=> 2^(9-1)=256
#define DMA_TRANSACTION_SIZE  (0x4)          // [0x0 - 0xA] TransSize=2^n i.e., 0xA=> 2^10=1024

////////////////////////////////////////////////////////////////////////////////////////
// Maximum of 32 commands slots per port exist where each command occupies 8 DWs (64 Bytes).
// The structure 'CmdListHeader' defines a single command header definition.
// The start of the first Command List &CmdListHeader[0] needs to be programmed onto P0CLB.
//
// Command List Base Address should be 1K Byte Aligned.

typedef struct {
    Uint32 CmdLen:5;   //bits[4:0]
    Uint32 Atapi:1;    //bit[5]
    Uint32 Write:1;    //bit[6]
    Uint32 Prefetch:1; //bit[7]
    Uint32 Reset:1;    //bit[8]
    Uint32 Bist:1;     //bit[9]
    Uint32 Rok:1;      //bit[10]
    Uint32 Rsv:1;      //bit[11]
    Uint32 Pmp:4;      //bits[15:12]
    Uint32 Prdtl:16;   //bits[31:16]
}CmdListHeaderW0;


typedef struct {
    Uint32 PrdByteCnt; //bits[31:0]
}CmdListHeaderW1;


typedef struct {
//      Uint32 CmdTableAddLowRsv:7; //bit[6:0]
//      Uint32 CmdTableAddLow:25;   //bits[31:7]
        Uint32 CmdTableAddLow;      //bits[31:7]
}CmdListHeaderW2;

typedef struct {
        Uint32 CmdTableAddHigh;     //bits[31:0]
}CmdListHeaderW3;

typedef struct {
    CmdListHeaderW0 DW0;
    CmdListHeaderW1 DW1;
    CmdListHeaderW2 DW2;
    CmdListHeaderW3 DW3;
    Uint32          DW4;
```

```
    Uint32          DW5;
    Uint32          DW6;
    Uint32          DW7;
} CmdListHeader;


typedef struct {
    Uint32 B0FisType:8;    //bits[7:0]
    Uint32 BYTE1:8;        //bits[15:8]
    Uint32 B2Cmd:8;        //bits[23:16]
    Uint32 B3Feature:8;    //bits[31:24]
}CmdFisWord0;

typedef struct {
    Uint32 B0LbaLow:8;     //bits[7:0]
    Uint32 B1LbaMid:8;     //bits[15:8]
    Uint32 B2LbaHigh:8;    //bits[23:16]
    Uint32 B3Device:8;     //bits[31:24]
}CmdFisWord1;

typedef struct {
    Uint32 B0LbaLowExp:8;  //bits[7:0]
    Uint32 B1LbaMidExp:8;  //bits[15:8]
    Uint32 B2LbaHighExp:8; //bits[23:16]
    Uint32 B3FeatureExp:8; //bits[31:24]
}CmdFisWord2;

typedef struct {
    Uint32 B0SecCnt:8;     //bits[7:0]
    Uint32 B1SecCntExp:8;  //bits[15:8]
    Uint32 B2Rsv:8;        //bits[23:16]
    Uint32 B3Control:8;    //bits[31:24]
}CmdFisWord3;


typedef struct {
    Uint32 DWResv;         //bits[31:0]
}CmdFisWord4;

typedef struct {
    CmdFisWord0 DW0;
    CmdFisWord1 DW1;
    CmdFisWord2 DW2;
    CmdFisWord3 DW3;
    CmdFisWord4 DW4;
    Uint32      DW5;
    Uint32      DW6;
    Uint32      DW7;
    Uint32      DW8;
    Uint32      DW9;
    Uint32      DW10;
    Uint32      DW11;
    Uint32      DW12;
    Uint32      DW13;
    Uint32      DW14;
    Uint32      DW15;
}CommandFIS;

//-----------Command FIS end ATAPI Command ------

// ATAPI Command Data Structure
typedef struct {
    Uint32 ATAPI[4];
}Atapi;

//-----------ATAPI Command end PRDT -------------
```

```
// Physical Region Descriptor Table Data Structure
typedef struct {
    Uint32 DbaLow;      //bits[31:0]
}DbaAddressLow;

typedef struct {
    Uint32 DbaHigh;     //bits[31:0]
}DbaAddressHigh;

typedef struct {
    Uint32 DW2Reserved; //bits[31:0]
}PrdtRsv;

typedef struct {
    Uint32 DataBC:22;   //bits[21:0]
}DataByteCnt;


typedef struct {
    DbaAddressLow  DW0;
    DbaAddressHigh DW1;
    PrdtRsv        DW2;
    DataByteCnt    DW3;
}PRDT;
//-----------PRDT end --------------------------

//-----------Command Table Data Structure -------
// Since Command Table has to be 128 bytes = 0x80 bytes aligned if supporting more
//  than a single Command Header, then need to make sure that the Array you are
//  creating for all associated Command Tables match the 128 bytes alignment.
//  In order to do so, the number of PRD Table length you are allocating should be
//  multiples of 8.
typedef struct {
    CommandFIS cfis;
    Atapi      atapi;
    Uint32     Rsv[12];
    PRDT       prdTable[16];   // Have forced this size to 8 in order to meet the minimum
                                   // required size for Command Table.
}CommandTable;

//-----------Command Table Data Structure end ---


////////////////////////////////////////////////////////////////////////////////////////
// Receive FIS requires the Receive FIS to be 256 byte aligned. P0FB should be programmed
// with this restriction.
//
// RECEIVE FIS Data Structure
// Members: DMA Setup FIS (DSFIS)
//          PIO Setup FIS (PSFIS)
//          D2H Register FIS (RFIS)
//          Set Device Bits FIS (SDBFIS)
//          Unknown FIS (UFIS)

//-----------DMA Setup FIS----------------------

typedef struct {
    Uint32 B0FisType:8; //bits[7:0]
    Uint32 BYTE1:8;     //bits[15:8]
    Uint32 B2Rsv:8;     //bits[23:16]
    Uint32 B3Rsv:8;     //bits[31:24]
}DsfisW0;

typedef struct {
    DsfisW0 DW0;
    Uint32 DW1DmaBuffLow;
```

```
    Uint32 DW2DmaBuffHigh;
    Uint32 DW3Rsv;
    Uint32 DW4DmaBuffOffset;
    Uint32 DW5DmaXfrCnt;
    Uint32 DW6Rsv;
}DMASetupFis;

//-----------DMA Setup FIS end PIO Setup FIS ----

typedef struct {
    Uint32 B0FisType:8;    //bits[7:0]
    Uint32 BYTE1:8;        //bits[15:8]
    Uint32 B2Status:8;     //bits[23:16]
    Uint32 B3Errror:8;     //bits[31:24]
}PioSetupDW0;


typedef struct {
    Uint32 B0LbaLow:8;     //bits[7:0]
    Uint32 B1LbaMid:8;     //bits[15:8]
    Uint32 B2LbaHigh:8;    //bits[23:16]
    Uint32 B3Device:8;     //bits[31:24]
}PioSetupDW1;


typedef struct {
    Uint32 B0LbaLowExp:8;  //bits[7:0]
    Uint32 B1LbaMidExp:8;  //bits[15:8]
    Uint32 B2LbaHighExp:8; //bits[23:16]
    Uint32 B3Rsv:8;        //bits[31:24]
}PioSetupDW2;


typedef struct {
    Uint32 B0SecCnt:8;     //bits[7:0]
    Uint32 B1SecCntExp:8;  //bits[15:8]
    Uint32 B2Rsv:8;        //bits[23:16]
    Uint32 B3Estatus:8;    //bits[31:24]
}PioSetupDW3;


typedef struct {
    Uint32 HW0XferCnt:16;  //bits[15:0]
    Uint32 HW1Rsv:16;      //bits[31:16]
}PioSetupDW4;


typedef struct {
    PioSetupDW0 DW0;
    PioSetupDW1 DW1;
    PioSetupDW2 DW2;
    PioSetupDW3 DW3;
    PioSetupDW4 DW4;
}PIOSetupFis;

//-----------PIO Setup FIS end D2H Reg FIS-------

typedef struct {
    Uint32 B0FisType:8;   //bits[7:0]
    Uint32 BYTE1:8;       //bits[15:8]
    Uint32 B2Status:8;    //bits[23:16]
    Uint32 B3Errror:8;    //bits[31:24]
}D2HRegDW0;


typedef struct {
```

```
    Uint32 B0LbaLow:8;    //bits[7:0]
    Uint32 B1LbaMid:8;    //bits[15:8]
    Uint32 B2LbaHigh:8;   //bits[23:16]
    Uint32 B3Device:8;    //bits[31:24]
}D2HRegDW1;


typedef struct {
    Uint32 B0LbaLowExp:8; //bits[7:0]
    Uint32 B1LbaMidExp:8; //bits[15:8]
    Uint32 B2LbaHighExp:8;//bits[23:16]
    Uint32 B3Rsv:8;       //bits[31:24]
}D2HRegDW2;


typedef struct {
    Uint32 B0SecCnt:8;    //bits[7:0]
    Uint32 B1SecCntExp:8; //bits[15:8]
    Uint32 HW1Rsv:16;     //bits[31:16]
}D2HRegDW3;


typedef struct {
    Uint32 W0Rsv;         //bits[31:0]
}D2HRegDW4;


typedef struct {
    D2HRegDW0 DW0;
    D2HRegDW1 DW1;
    D2HRegDW2 DW2;
    D2HRegDW3 DW3;
    D2HRegDW4 DW4;
}D2HRegFis;

//-----------D2H Reg FIS end Set Device Bits FIS-
// The Set Device Bit FIS definition does not contain the 2nd Word required
// for Native Command Queueing. This second word is the SACTVE register and
// the AHCI takes care of updating P0SACT register at its location.

typedef struct {
    Uint32 B0FisType:8;   //bits[7:0]
    Uint32 BYTE1:8;       //bits[15:8]
    Uint32 B2Status:8;    //bits[23:16]
    Uint32 B3Errror:8;    //bits[31:24]
}SetDevBitsDW0;

typedef struct {
    Uint32 W1Rsv;         //bits[31:0]
}SetDevBitsDW1;

typedef struct {
    SetDevBitsDW0 DW0;
    SetDevBitsDW1 DW1;
}SetDevBitsFis;

//-----------Set Device Bits FIS end Unkonwn FIS-

typedef struct {
    Uint32 UserDefined; //bits[31:0]
}UnknownDWx;


typedef struct {
    UnknownDWx DW[16]; // 16 Words (Max 64 Bytes allowed)
}UnknownFis;
```

```
//-----------Unkonw FIS end----------------------

//-----------Receive Register FIS Structure------

typedef struct {
    DMASetupFis   DSFIS;
    Uint32        Rsv1;
    PIOSetupFis   PSFIS;
    Uint32        Rsv2[3];
    D2HRegFis     RFIS;
    Uint32        Rsv3;
    SetDevBitsFis SDBFIS;
    UnknownFis    UFIS;
}ReceiveFis;

/


typedef struct {
    Uint8 cfisType;
    Uint8 cfisByte1;
    Uint8 cfisCmd;
    Uint8 cfisFeature;
    Uint8 cfisDw1SecNumLbaLow;
    Uint8 cfisDw1CylLowLbaMid;
    Uint8 cfisDw1CylHighLbahigh;
    Uint8 cfisDw1Dev;
    Uint8 cfisDw2SecNumLbaLowExp;
    Uint8 cfisDw2CylLowLbaMidExp;
    Uint8 cfisDw2CylHighLbahighExp;
    Uint8 cfisDw2FeatureExp;
    Uint8 cfisDw3SecCnt;
    Uint8 cfisDw3SecCntExp;
    Uint8 cfisDw3Ctrl;
}cmdFis;

typedef struct {
    Uint8 dsfisType;
    Uint8 dsfisByte1;
    Uint32 dsfisDw1DmaBuffLow;
    Uint32 dsfisDw2DmaBuffHigh;
    Uint32 dsfisDw4DmaBuffOffset;
    Uint32 dsfisDw5DmaXferCnt;
}dsFis;

typedef struct {
    Uint8 psfisType;
    Uint8 psfisByte1;
    Uint8 psfisStatus;
    Uint8 psfisError;
    Uint8 psfisDw1SecNumLbaLow;
    Uint8 psfisDw1CylLowLbaMid;
    Uint8 psfisDw1CylHighLbahigh;
    Uint8 psfisDw1Dev;
    Uint8 psfisDw3SecCnt;
    Uint8 psfisDw3Estatus;
    Uint16 psfisDw4XferCnt;
}piosFis;

typedef struct {
    Uint8 regfisType;
    Uint8 regfisByte1;
    Uint8 regfisStatus;
    Uint8 regfisError;
    Uint8 regfisDw1SecNumLbaLow;
```

```
    Uint8 regfisDw1CylLowLbaMid;
    Uint8 regfisDw1CylHighLbahigh;
    Uint8 regfisDw1Dev;
    Uint8 regfisDw3SecCnt;
}regFis;

typedef struct {
    Uint8 sdbfisType;
    Uint8 sdbfisByte1;
    Uint8 sdbfisStatus;
    Uint8 sdbfisError;
}sdbFis;

typedef struct {
    Uint32 ufisWord[16];
}uFis;

typedef struct {
    Uint32  capSMPS:1;
    Uint32  capSSS:1;
    Uint32  piPi:2;
    Uint32  p0cmdCpd:1;
    Uint32  p0cmdEsp:1;
    Uint32  p0cmdMpsp:1;
    Uint32  p0cmdHpcp:1;
    Uint32  rsv:24;
}FirmwareCtrlFeatures;


void initMemory(Uint32 *startAddress, Uint32 length, Uint32 seedWord, Uint32 modifyVal) {
    Uint32 I;
    *startAddress++ = seedWord;
    for (I=0; i<length-1; I++) {
        seedWord += modifyVal;
        *startAddress++ = seedWord;
    }
}

void clearCmdList(void) {
    //clear Host to Device (Command FIS) Space
    initMemory((Uint32*)CmdLists, (LISTLENGTH*(sizeof(CmdListHeader)/4)), 0, 0);
}

void clearCmdTables(void) {
    Uint16 cmdSlot;
    for (cmdSlot=0; cmdSlot<LISTLENGTH; cmdSlot++) {
        //Clear Command FIS and ATAPI Command Spaces for Command Header X; LISTLENGTH < X < 0
        initMemory((Uint32 *)&CmdTable[cmdSlot], (sizeof(CommandFIS)/4)+(sizeof(Atapi)/4), 0, 0);
        //Clear PRD Descriptor Locations for Command Header X; LISTLENGTH < X < 0
        initMemory((Uint32*)((Uint32)&CmdTable[cmdSlot]+0x80),
(sata_input_filePageSize*(sizeof(PRDT)/4)*sata_input_prdLength), 0, 0);
    }
}

void clearRcvFis() {
    //clear Receive DMA Setup FIS Space.
    initMemory((Uint32*)&RcvFis, (sizeof(DMASetupFis)/4), 0, 0);
    //clear Receive PIO Setup FIS Space.
    initMemory((Uint32*)((Uint32)&RcvFis+0x20), (sizeof(PIOSetupFis)/4), 0, 0);
    //clear Receive Device to Host (D2H) Register FIS Space.
    initMemory((Uint32*)((Uint32)&RcvFis+0x40), (sizeof(D2HRegFis)/4), 0, 0);
    //clear Set Device Bits FIS Space.
    initMemory((Uint32*)((Uint32)&RcvFis+0x58), (sizeof(SetDevBitsFis)/4), 0, 0);
    //clear Unknow FIS Space.
    initMemory((Uint32*)((Uint32)&RcvFis+0x60), (sizeof(UnknownFis)/4), 0, 0);
}
```

```
void clearDmaBuffers(void) {
    //Clear PRD Data Buffer Memory
    initMemory((Uint32 *)prdTableDataBuff, (LISTLENGTH*PRDLENGTH*DATABUFFERLEN/4), 0, 0);
}


void performFirmwareInit(void) {
/* Firmware Initialization*/
//  Make sure you perform a Single Write in one operation of all HwInit Fields
//  initialization defined within a single register
    sataRegs->CAP |= ((swCtrlFeatures.capSMPS << 28) |
                      (swCtrlFeatures.capSSS  << 27)
                      );
// Configure PI[31:0]
    sataRegs->PI  |=  (swCtrlFeatures.piPi  << 0);

// Configure P0CMD[ESP,CPD,MPSP,HPCP=21,20,19,18]
    sataRegs->P0CMD |= ((swCtrlFeatures.p0cmdEsp << 21) |
                        (swCtrlFeatures.p0cmdCpd << 20) |
                        (swCtrlFeatures.p0cmdMpsp << 19)|
                        (swCtrlFeatures.p0cmdHpcp << 18)
                       );

/* Software Initialization*/
    // Initialize PHY and DMA Parameters (Corresponding to Gen1/2i timing)
    sataRegs->P0PHYCR = 0x80182048;

    initBaseAddresses(); // Initialize Command List (P0CLB) and Receive FIS (P0FS)

    // Configure Line Speed.
    setSataSpeed(DESIRED_SPEED);  //GOASFASTASDEVICE=0,GEN1=1,GEN2=2

    enableRcvFis();    // Enable Receive DMA


// The below are general Semaphores/Flags used and want to make sure they are initialized.
// 28 Bit LBA Address of Device. 0xFFFFFFFF is used by test S/W to indicate that it is not
initialized
    Dev28bitLbaAddress = 0xFFFFFFFF; // S/W needs to initialize this variable prior to calling


    //_INT_DRIVEN_TEST_ is defined within the Project File
    #ifdef _INT_DRIVEN_TEST_
        intHandlingMethod = USE_INT_HANDLER;
    #else
        intHandlingMethod = USE_POLLING;
    #endif

}


char spinUpDeviceAndWaitForInitToComplete(void) {
    // Make sure that the HBA is in a Listen Mode prior to Spinning Up Device
    // Following Configuration is not allowed.
    //      [P0SCTL.DET, P0CMD.SUD] = [1,1] NOT Allowed.
    if((sataRegs->P0SCTL & AHCI_PxSCTL_PxSSTS_DET) != 0)
        sataRegs->P0SCTL &= ~(0xf << AHCI_PxSCTL_PxSSTS_DET_SHIFT);

    // Clear P0SERR.DIAG.X (RWC bit field) so that the P0TFD is updated by HBA.
    sataRegs->P0SERR |= 0x04000000;

    // Spin Up Device.
    sataRegs->P0CMD |= (1 << AHCI_PxCMD_SUD_SHIFT);
```

```
    // Wait for Device Detection or/and Speed Negotiation to take place and finish.
    while ((sataRegs->P0SSTS & AHCI_PxSCTL_PxSSTS_DET) !=0x3);

    // Device would send its status and and default Task file regs content (signature)
    // when finished with Power Up: Look for Device ready status.
    while ((sataRegs->P0TFD & AHCI_PxTFD_STS_BSY_DRQ_ERR) != 0);

    // Make sure that the expected Device signature is received.
    if (sataRegs->P0SIG != AHCI_P0SIG_SIG_ATA_DEV_GOOD_STAT) // LBAhigh:LBAmid:LBAlow:SECcnt
        return(1);                                           // =0x00000101

    return(0);

}

void initIntAndClearFlags(void) {
        // Make sure Interrupt is disabled (Disable at Port Level followed by Global Level).
        // Clear Interrupt at Port Level
        enableDisableInt(PORTint, DISABLE, 0xFFFFFFFF); // clearInt(int type, intState,
specificField)
                                                        // int type=GLOBALint or PORTint
                                                        // intState=DISABLE or ENABLE
                                                        // specificField=bit field to Enable or
Disable
                                                        // is used for the RWC feature for PORTint
       // Disable interrrupt at Global Level
       enableDisableInt(GLOBALint, DISABLE, 0);            // clearInt(intType, intState
fields2clr)
                                                        // int type=GLOBALint or PORTint
                                                        // intState=DISABLE or ENABLE
                                                        // fields2clr=dontcare for GLOBALint
                                                        // is used for the RWC feature for PORTint

    // Need to clear interrupts at Port Level followed by Global Level.
    // Ensure all pending Port Error and Status are cleared.
    clearIntOrErrorDiag(ERRORFIELDS, sataRegs->P0SERR);    // Clear P0SERR Register
    // Ensure all pending Port Interrupts and The Single Global Interrupt are cleared.
    clearIntOrErrorDiag(INTFIELDS, sataRegs->P0IS);      // Clear P0IS and IS Regs
}


void invokeHBAReset() {
    // HBA Reset will not affect the following Registers settings of PxFB and PxCLB
    // regs and HwInit fields of Port Registers are not affected.
    // To Do: Check if the Global Registers are affected. Spec mentions not affected.

    // Note: COMRESET OOB will not be sent to attached Device because this device supports
    //  Staggered Spinup capability and P0CMD.SUD is cleared to Zero when HBA Reset
    //  takes place. Software needs to invoke this if needed.

    // Most likely user want to ensure HBA comes up in its default operation state
    //  or has hung and is unable to idle the port when needing to perform an HBA
    //  reset. Regardless, there is no need to attempt to idle the HBA from
    //  running
    sataRegs->GHC |= (1 << AHCI_GHC_HR_SHIFT);

    // Max Spec time is 1 Second for Reset to complete.
    while((sataRegs->GHC & AHCI_GHC_HR) != 0) {
        waitForXms(WAIT_500_MILLISECONDS);
        waitForXms(WAIT_500_MILLISECONDS);
    }
}

char placeHbaInIdle(void) {
    // To Place HBA In IDLE, need to make sure both DMAs (Cmd List and Rcv FIS) are not running.
    // Order of Disabling the DMA is important.
```

```
// Ensure that the Cmd List DMA is not running
    //  If is running, clear ST and wait for 500ms. Then Check CR.
    if (sataRegs->P0CMD & AHCI_PxCMD_ST) {
        sataRegs->P0CMD &= ~(1 << AHCI_PxCMD_ST_SHIFT);
        waitForXms(WAIT_500_MILLISECONDS);
    }// Wait another 500 Milliseconds for CR to clear. This is twice more than required.
    if (sataRegs->P0CMD & AHCI_PxCMD_CR)
        waitForXms(WAIT_500_MILLISECONDS);

    // If P0CMD.CR is still set, HBA probably has hunged. No need to continue.
    // Need to perform HBA Reset.
    if (sataRegs->P0CMD & AHCI_PxCMD_CR)
        return(1);

// Ensure that the Receive FIS DMA is running.
    //  If is running, clear FRE and wait for 500ms. Then Check FR.
    if (sataRegs->P0CMD & AHCI_PxCMD_FRE) {
            sataRegs->P0CMD &= ~(1 << AHCI_PxCMD_FRE_SHIFT);
            waitForXms(WAIT_500_MILLISECONDS);
        }// Wait unti FR is Cleared.
    while (sataRegs->P0CMD & AHCI_PxCMD_FR)
        waitForXms(WAIT_500_MILLISECONDS);

    // If P0CMD.FRE is still set, HBA probably has hunged. No need to continue.
    // Need to perform HBA Reset.
    if (sataRegs->P0CMD & AHCI_PxCMD_FRE)
        return(1);

    return(0);
}


void associateSysMem2Hba(Uint16 cmdSlot) {
    associtateCmdSlotWithCmdTable(cmdSlot);    // Assign Sys Mem allocated for Cmd Table to Cmd
List Slot
    //associtatePrdsWithCmdTable(cmdSlot);     // Assign PRD info to Cmd Table
    associtatePrdsWithCmdTable(cmdSlot, sata_input_filePageSize);
}

void associtateCmdSlotWithCmdTable(Uint16 cmdSlot) {
    CmdLists[cmdSlot].DW2.CmdTableAddLow=((unsigned int)&CmdTable[cmdSlot] & 0xFFFFFF80);
    CmdLists[cmdSlot].DW3.CmdTableAddHigh=0x0;
}

void associtatePrdsWithCmdTable(Uint16 cmdSlot, Uint16 fSize) {
    Uint16 fileSIZE, prdLength;
    for (fileSIZE=0; fileSIZE<fSize; fileSIZE++) {
        for (prdLength=0; prdLength<sata_input_prdLength; prdLength++) {
            // Command Header 0 PRD Descriptors 0 & 1 are Initialized.
CmdTable[cmdSlot].prdTable[(sata_input_prdLength*fileSIZE)+prdLength].DW0.DbaLow=(unsigned
int)&prdTableDataBuff[cmdSlot][prdLength];
CmdTable[cmdSlot].prdTable[(sata_input_prdLength*fileSIZE)+prdLength].DW1.DbaHigh=0x0;
CmdTable[cmdSlot].prdTable[(sata_input_prdLength*fileSIZE)+prdLength].DW3.DataBC=sata_input_prd_da
taBuffLen-1;
        }
    }
}


void setSataSpeed(unsigned char iSpeed) {
    sataRegs->P0SCTL |= (iSpeed << AHCI_PxSCTL_PxSSTS_SPD_SHIFT);
    waitForXms(5); // This might not be necessary: wait a bit
}

char setupCfisEntriesForDataRdWr(CmdListHeader *CmdListNum, dataXferDir readOrWrite, xferProtocol
```

```
xferType) {
    // ********************************************************************************
    //$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ **** Initializing the Command Header *******
    // Other part of the Command List Structure, with the exception of Word 0 for the
    //  Command Slots[0] and [1] are already initialized when invoking sata_init_and_spin_up()
    //  function via associateMem2HBA() function.
    // Configure Word 0 of Command List
    CmdListNum->DW0.CmdLen=5;   // This is the length of H2D FIS. This might need changing
                                //   based on the Command issued to Device. Need to Check.
    CmdListNum->DW0.Atapi=0;    // Command is destined to HDD Like Device.
    CmdListNum->DW0.Prefetch=1; // Doesn't hurt prefetching so do it.
                                // WARNING: Do Not Prefetch if using:
                                //          => Command Queuing
                                //          => Port Multiplier
    CmdListNum->DW0.Reset=0;    // This is normally set to Zero unless a Soft Reset is required.
    CmdListNum-
>DW0.Bist=0;     // This is for entering test mode and should be cleared for normal operation.
    CmdListNum-
>DW0.Rok=0;      // For Normal operation require to Clear this bit so P0TFD and P0CI are modified
by HBA as appropriate.
                                //  Rok should be set for S/W Reset Command.
    CmdListNum-
>DW0.Pmp=0x0;    // Used only if an external Port Multiplier is attached and selects the Port of
the Port Multiplier.
    //$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
    // The above DW0 fields usually would not change for Normal operation.

    if (readOrWrite == DATA_DIR_WR) // The Write setting here is based on the Data FIS direction.
        CmdListNum->DW0.Write=1;    //  Write=1/0=>Write/Read;
    else if (readOrWrite == DATA_DIR_RD) // The Write setting here is based on the Data FIS
direction.
        CmdListNum->DW0.Write=0;    //  Write=1/0=>Write/Read;
    else return(1);

//    CmdListNum-
>DW0.Prdtl=sata_input_prdLength;    // Need to update this when using DMA for Data transfer.
    CmdListNum-
>DW0.Prdtl=sata_input_filePageSize*sata_input_prdLength;    // Need to update this when using DMA
for Data transfer.

    // ********************************************************************************
    //$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ **** Initializing the Command FIS (H2D FIS) ******

    // Cmd FIS is made of a 20 Bytes size FIS.
    // FIS Fieds are Initialized into the alloted data buffers.
    // Strucutre type 'cmdFis' holds 8 of the 20 bytes of the FIS. These
    //  seven elements of the cfis are good enough for majority of command
    //  building tasks.
    // If need to access other cfis members, need to access the cfis member
    //  directly. Example of how to access the FIS Type, cfis Byte0.
    //     CmdTable[n].cfis.DW0.B0FisType=0x27;
    // FIS Type is hard coded within buildCmdFis function and Reserved Locations
    //  are also cleared to Zeros, so no need of iniitalizing this bytes is necessary here.

    // Make sure the Device 28 Bit LBA Address is initialize prior to calling this function.
    if (Dev28bitLbaAddress == 0xFFFFFFFF)
        return(1);

        myCmdFis.cfisByte1          = CMDFIS_BYTE1_C_IS_CMD_UPDATE;   // Bit7 of Byte1 is set for
Command Wr and Cleared for Control Wr
       //myCmdFis.cfisByte1         = CMDFIS_BYTE1_C_IS_CTRL_UPDATE;  // Bit7 of Byte1 is Cleared
for Command Control Wr.

       if (readOrWrite == DATA_DIR_WR) {
           if (xferType == DMA_PROTOCOL) {
               myCmdFis.cfisCmd     = ATA_CMD_WRITE_DMA;     // Uses 28-Bit Addressing.
```

```
                //myCmdFis.cfisCmd   = ATA_CMD_WRITE_DMA_EXT; // Uses 48-Bit Addressing.
            } else {
              myCmdFis.cfisCmd       = ATA_CMD_WRITE_SECTOR;  // PIO Write Command: Uses 28-
Bit Addressing
          }
    }
       else {
             if (xferType == DMA_PROTOCOL) {
                myCmdFis.cfisCmd    = ATA_CMD_READ_DMA;     // Uses 28-Bit Addressing.
                //myCmdFis.cfisCmd  = ATA_CMD_READ_DMA_EXT; // Uses 48-Bit Addressing.
            } else {
              if (PioCmd == ATA_CMD_IDENTIFY_DEVICE)
                  myCmdFis.cfisCmd  = ATA_CMD_IDENTIFY_DEVICE; // PIO Read Command: Uses 28-
Bit Addressing.
               else
                  myCmdFis.cfisCmd  = ATA_CMD_READ_SECTOR;     // PIO Read Command: Uses 28-
Bit Addressing.
            }
          }

    myCmdFis.cfisFeature           = 0x00;
    myCmdFis.cfisDw1SecNumLbaLow   = (Uint8) Dev28bitLbaAddress;
    myCmdFis.cfisDw1CylLowLbaMid   = (Uint8)(Dev28bitLbaAddress>>8);
    myCmdFis.cfisDw1CylHighLbahigh = (Uint8)(Dev28bitLbaAddress>>16);
    myCmdFis.cfisDw1Dev            = ( DEVICE_REG_USE_LBA_ADDRESSING |
                                     (Uint8)(Dev28bitLbaAddress>>24)
                                    );
//    myCmdFis.cfisDw3SecCnt         = (sata_input_prdLength*sata_input_prd_dataBuffLen)/512;
    myCmdFis.cfisDw3SecCnt         =
(sata_input_filePageSize*sata_input_prd_dataBuffLen*sata_input_prdLength)/512;
    myCmdFis.cfisDw3Ctrl           = 0x00;

    // The below require to be initialized at least once and can be ignored especially if
    //  not using 48-Bit Addressing. If use 48-Bit Addressing, then require constant
    //  maintenance prior to invoking a command.
    myCmdFis.cfisDw2SecNumLbaLowExp=0x00;
    myCmdFis.cfisDw2CylLowLbaMidExp=0x00;
    myCmdFis.cfisDw2CylHighLbahighExp=0x00;
    myCmdFis.cfisDw2FeatureExp=0x00;
    myCmdFis.cfisDw3SecCntExp=0x00;

    // Invalidate for future use.
    Dev28bitLbaAddress = 0xFFFFFFFF;

    return(0);
}

void buildCmdFis(CommandTable *CmdSlotNum) {
//    +----------+----------+--------------+-----------+
// DW0|  FEATURE | COMMAND  | c r r r port |FISTYPE 27h|
//    +----------+----------+--------------+-----------+
// DW1|  DEVICE  | LBA HIGH |  LBA MID     |  LBA LOW  |
//    +----------+----------+--------------+-----------+
// DW2|FETURESexp|LBAHIGHexp|  LBAMIDexp   | LBALOWexp |
//    +----------+----------+--------------+-----------+
// DW3| CONTROL  | RESERVED | SEC CNTexp   |  SEC CNT  |
//    +----------+----------+--------------+-----------+
// DW4| RESERVED | RESERVED |  RESERVED    | RESERVED  |
//    +----------+----------+--------------+-----------+
    CmdSlotNum->cfis.DW0.B0FisType=0x27;
    CmdSlotNum-
>cfis.DW0.BYTE1=myCmdFis.cfisByte1;      //Make Sure the 'C' bit field is correctly set or
cleared.
    CmdSlotNum->cfis.DW0.B2Cmd=myCmdFis.cfisCmd;
    CmdSlotNum->cfis.DW0.B3Feature=myCmdFis.cfisFeature;
```

```
    CmdSlotNum->cfis.DW1.B0LbaLow=myCmdFis.cfisDw1SecNumLbaLow;
    CmdSlotNum->cfis.DW1.B1LbaMid=myCmdFis.cfisDw1CylLowLbaMid;
    CmdSlotNum->cfis.DW1.B2LbaHigh=myCmdFis.cfisDw1CylHighLbahigh;
    CmdSlotNum->cfis.DW1.B3Device=myCmdFis.cfisDw1Dev;  //Make Sure 48-Bit or 28-
Bit Addressing is indicated here.


    CmdSlotNum->cfis.DW2.B0LbaLowExp=myCmdFis.cfisDw2SecNumLbaLowExp;    //0x0;
    CmdSlotNum->cfis.DW2.B1LbaMidExp=myCmdFis.cfisDw2CylLowLbaMidExp;    //0x0;
    CmdSlotNum->cfis.DW2.B2LbaHighExp=myCmdFis.cfisDw2CylHighLbahighExp; //0x0;
    CmdSlotNum->cfis.DW2.B3FeatureExp=myCmdFis.cfisDw2FeatureExp;        //0x0;

    CmdSlotNum->cfis.DW3.B0SecCnt=myCmdFis.cfisDw3SecCnt;
    CmdSlotNum->cfis.DW3.B1SecCntExp=myCmdFis.cfisDw3SecCntExp;          //0x0;
    CmdSlotNum->cfis.DW3.B2Rsv=0x0;
    CmdSlotNum->cfis.DW3.B3Control=myCmdFis.cfisDw3Ctrl;

    CmdSlotNum->cfis.DW4.DWResv=0x0;
}

char startCmdListProcessing(void) {
    // Make sure that a device is present and HBA has established communications.
    while ((sataRegs->P0SSTS & AHCI_PxSCTL_PxSSTS_DET) !=0x3);

    // Clear P0SERR.DIAG.X (RWC bit field) so that the P0TFD is updated by HBA.
    // Make sure it is cleared.
    sataRegs->P0SERR |= 0x04000000;

    // Make sure the Command List is not Running.

    if (sataRegs->P0CMD & AHCI_PxCMD_CR)
        return(1);
    //  Task file regs and look for Device ready status.
    while ((sataRegs->P0TFD & AHCI_PxTFD_STS_BSY_DRQ_ERR) != 0);

    // Make sure the the Receive FIS DMA is running.
    if ((sataRegs->P0CMD & (AHCI_PxCMD_FRE | AHCI_PxCMD_FRE)) !=
                            (AHCI_PxCMD_FRE | AHCI_PxCMD_FRE))
        return(1);

    // Enable the Cmd List DMA Engine.
    sataRegs->P0CMD |= AHCI_PxCMD_ST;

    // Wait here a bit until the Command List DMA Engine has started to run
    while ((sataRegs->P0CMD & AHCI_PxCMD_CR) == 0)
        waitForXms(1);

    return(0);
}

char submitCmd(Uint8 commandType, Uint8 commandSlot) {
    // Make sure both the Command List and Receive FIS DMAs are eanbled and running prior to
    //  submiting command
    Uint16 I;
    if ((sataRegs->P0CMD & (AHCI_PxCMD_FRE | AHCI_PxCMD_FRE | AHCI_PxCMD_CR | AHCI_PxCMD_ST)) !=
                            (AHCI_PxCMD_FRE | AHCI_PxCMD_FRE | AHCI_PxCMD_CR | AHCI_PxCMD_ST))
        return(1);

    switch (commandType) {
            case NON_QUEUED_CMD:
                sataRegs->P0CI   |= (0x1 << commandSlot);
                break;

            case QUEUED_CMD:
                sataRegs->P0SACT |= (0x1 << commandSlot);
                sataRegs->P0CI   |= (0x1 << commandSlot);
```

```
                    break;

                default:
                    break;
        }
        return(0);
    }

    // Basic Interrupt Handler Function.
    void sataIsr(void) {
        intIsrCnt++;    // Count interrupt.
        intIsrFlag=1;

        // Ensure all pending Port Interrupts and The Single Global Interrupt are cleared.
        clearIntOrErrorDiag(INTFIELDS, sataRegs->P0IS); // Clear P0IS and IS Regs

    }
```

20.3.2 Example on Initialization and Spinning Up Device

```
//char hetero_doTest(void) {
char sata_setup() {

        progStatus1='F';
        progStatus2='F';


        // Firmware HwInit Fields Configuration values.
        //  Need to configure this prior to calling sata_init_and_spin_up();
        swCtrlFeatures.capSMPS=1;   // Input Pin exist for external activity detection presence.
        swCtrlFeatures.capSSS=1;    // Always set to 1 in order to avoid spin up when HBA is
powered.
        swCtrlFeatures.piPi=3;      // Supports two HBA Ports (0 and 1). Corresponding bit fields
of Pl
                                       need to be set.
        swCtrlFeatures.p0cmdEsp=0;  // The state of this bit is based on the support for eSATA.
                                       CAP.SXS setting is the Logical OR of all Ports PxCMD.ESP.
                                       If any of the PxCMD.ESP is set, the CAP.SXS will be set
too.
        swCtrlFeatures.p0cmdHpcp=0; // Since ESP is mutually exclusive with HPCP (as mentioned in
spec)
                                       then HPCP should be set to 0.


        if(chceckSysMemorySize())
           for(;;);                 // If program stays here, need to fix alignment issue.

        // Clear all allocated System Memory
        clearCmdList();                // Clear Cmd List allocated within Sys Mem
        clearCmdTables();           // Clear Cmd Tables allocated within Sys Mem
        clearRcvFis();                // Clear Receive FIS allocated within Sys Mem
        clearDmaBuffers();          // Clear all DMA Buffers

        // Make sure that both DMAs (Cmd List and Rcv FIS) are not running.
        if (placeHbaInIdle())
            invokeHBAReset();       // If unable to shut one or both DMAs, Perform HBA Reset.

        performFirmwareInit();

        //Start the Disk Drive (spin it up)
        if(spinUpDeviceAndWaitForInitToComplete())
           while(1);               // Stay here if device signature does not match.

        //DMA Settings get affected by RESET
        cfgDmaSetting();

        initIntAndClearFlags();    // Disable CCC, Initialize 1ms Time, Clear Int and Flags


        if(intHandlingMethod == USE_INT_HANDLER) { //  USE_POLLING or USE_INT_HANDLER
           // Setup Interrupt Handler
           intIsrFlag=0;
           sata_intc_setup();                     //  Configure Interrupt Handler


           enableDisableInt(PORTint, ENABLE, 0xFFC000FF); // enableDisableInt(int type, intState,
specificField)
                                                          // int type=GLOBALint or PORTint
                                                          // intState=DISABLE or ENABLE
           enableDisableInt(GLOBALint, ENABLE, 0);        // enableDisableInt(int type, intState,
specificField)
                                                          // int type=GLOBALint or PORTint
                                                          // intState=DISABLE or ENABLE
                                                          // specificField = Don't Care for
GLOBALint
```

```
            debugInt=0;
        } else
            enableDisableInt(PORTint, ENABLE, 0xFFC000FF); // enableDisableInt(int type, intState,
specificField)
                                                           // int type=GLOBALint or PORTint
                                                           // intState=DISABLE or ENABLE
        // Initialize Golden Data
        initMemory((Uint32 *)&prdTableDataBuff[0],
(sata_input_prdLength*sata_input_prd_dataBuffLen/4), 0x12345678, 0x01010101);
        // Invalidate Read Data Buffer
        initMemory((Uint32 *)&prdTableDataBuff[1],
(sata_input_prdLength*sata_input_prd_dataBuffLen/4), 0xDEADDEAD, 0x00000000);

        return(0);
}
```

20.3.3 Example of DMA Write Transfer

```
void performDmaWrite(void) {
// WRITE DMA
/*
  When performing the Non-Queued Command "Write DMA" the following captures the FISes that are
   communicated between Host (HBA) and Device (SpeedBridge or SATA Drive).

  HBA ==> Device    H2D FIS (Command FIS that has the "Write DMA" Command within H2D.Command
field).
  Device ==> HBA    DMA Activate FIS (Device notifies HBA that it's ready to receive data).
  HBA ==> Device    Data FIS (Actual Data Requested and pointed by the DMA/PRD Contents within
Command Table).
  Device ==> HBA    D2H FIS (Completion Status from Device)


  If need to capture Interrupt (just have only the Flag set) at:
    Global Level (IS Register), need to enable P0IE.Interrupt field.
    Port Level (No Enable bit needs to be set. P0IS will be set if D2H FIS DW0.Byte1.bit6 is set).

  If need to generate Interrupt at:
    Global Level (enable interrupt by setting GHC.IE bit field and P0IE.xxx should be set to
enable particular interrupt). Port Level interrupt handling is a subset of Global Level.
    Port Level (So long as Global Level interrupt is not enabled, GHC.IE is set, Interrupt will
not be sent to CPU just from Port Level only).
*/
// Write to Disk

    cmdSlot2Use=0;
    associateSysMem2Hba(cmdSlot2Use);    // Associate Sys Memory to CmdList and PRDs to Cmd Table

    //Initialize PRD Data Buffer Memory
//    initMemory((Uint32 *)&prdTableDataBuff[cmdSlot2Use],
(sata_input_prdLength*DATABUFFERLEN/4), 0x12345678, 0x01010101);

    // Configure Device 28 bit LBA Address (Start Address for Rd/Wr Transfer);
    Dev28bitLbaAddress              = sata_input_startAddress; // 28-Bit LBA Address

    // If problem exist when setuping CmdList, stay here. If not continue
    setupCfisEntriesForDataRdWr(&CmdLists[cmdSlot2Use], DATA_DIR_WR, DMA_PROTOCOL); // Usage:
setupCfisEntriesForDataRdWr(CmdListHeader *, DataDir, xferProtocol)
                                                                    // DataDir = DATA_DIR_RD or
DATA_DIR_WR, xferProtocol = PIO/DMA_PROTOCOL
    // Write cfis Data
    buildCmdFis(&CmdTable[cmdSlot2Use]);

    //getCmdFis(&CmdTable[0]);

    startCmdListProcessing(); // Stay here if unable to start processing command list.

    submitCmd (NON_QUEUED_CMD, cmdSlot2Use);    // Usage: CommandType (QUEUED(NON_QUEUED)_CMD,
Command Slot);

    if(intHandlingMethod == USE_INT_HANDLER) { //     USE_POLLING or USE_INT_HANDLER
        while(intIsrFlag==0);
        intIsrFlag=0;
    }
    else
        while(sataRegs-
>IS == 0);                           // Stay here until an interrupt is received.


    // If P0IS.DPS is set, A PRD with the I bit set has transferred all of its data.
    // This bitfield might not bit set for Non-Queued DMA. Applicable for Queued DMA
    //while(getRegStatus((Uint32*)&sataRegs-
>P0IS, AHCI_P0IS_DPS) == 0); // getRegStatus(ptr2int,field Mask)
```

```
    while(getRegStatus((Uint32*)&sataRegs-
>P0CI, (1<<cmdSlot2Use)) == (1<<cmdSlot2Use));    // Wait Until P0CI[ChHeader] to be cleared by
HBA

    // Clears both P0IS and IS register. Only Clears RWC bit fields. So, it is OK for this task.
    clearIntOrErrorDiag(INTFIELDS, sataRegs->P0IS);      // Clear P0IS Register
}
```

20.3.4 Example of DMA Read Transfer

```
void performDmaRead() {
// READ DMA
/*
  When performing the Non-Queued Command "Read DMA" the following captures the FISes that are
   communicated between Host (HBA) and Device (SpeedBridge or SATA Drive).

  HBA ==> Device    H2D FIS (Command FIS that has the "Read DMA" Command within H2D.Command
field).
  Device ==> HBA    Data FIS (Actual Data Requested and pointed by the DMA/PRD Contents within
Command Table).
  Device ==> HBA    D2H FIS (Completion Status from Device)

  If need to capture Interrupt (just have only the Flag set) at:
    Global Level (IS Register), need to enable P0IE.Interrupt field.
    Port Level (No Enable bit needs to be set. P0IS will be set if D2H FIS DW0.Byte1.bit6 is set).

  If need to generate Interrupt at:
    Global Level (enable interrupt by setting GHC.IE bit field and P0IE.xxx should be set to
enable particular interrupt). Port Level interrupt handling is a subset of Global Level.
    Port Level (So long as Global Level interrupt is not enabled, GHC.IE is set, Interrupt will
not be sent to CPU just from Port Level only).
*/
// Read from Disk
    cmdSlot2Use=1;
    associateSysMem2Hba(cmdSlot2Use);    // Associate Sys Memory to CmdList and PRDs to Cmd Table

    //Initialize PRD Data Buffer Memory
//    initMemory((Uint32 *)&prdTableDataBuff[cmdSlot2Use],
(sata_input_prdLength*DATABUFFERLEN/4), 0xDEADBEEF, 0x00000000);

    // Configure Device 28 bit LBA Address (Start Address for Rd/Wr Transfer);
    Dev28bitLbaAddress              = sata_input_startAddress; // 28-Bit LBA Address

    setupCfisEntriesForDataRdWr(&CmdLists[cmdSlot2Use], DATA_DIR_RD, DMA_PROTOCOL);    // Usage:
setupCfisEntriesForDataRdWr(CmdListHeader *, DataDir, xferProtocol)
                                                                    // DataDir = DATA_DIR_RD or
DATA_DIR_WR, xferProtocol = PIO/DMA_PROTOCOL
    // Write cfis Data
    buildCmdFis(&CmdTable[cmdSlot2Use]);

    //getCmdFis(&CmdTable[cmdSlot2Use]);

    startCmdListProcessing(); // Stay here if unable to start processing command list.


    submitCmd (NON_QUEUED_CMD, cmdSlot2Use); // Usage: CommandType (QUEUED(NON_QUEUED)_CMD,
Command Slot);

    if(intHandlingMethod == USE_INT_HANDLER) { //     USE_POLLING or USE_INT_HANDLER
        while(intIsrFlag==0);
        intIsrFlag=0;
    }
    else
        while(sataRegs-
>IS == 0);                            // Stay here until an interrupt is received.


    // If P0IS.DPS is set, A PRD with the I bit set has transferred all of its data.
    // This bitfield might not bit set for Non-Queued DMA. Applicable for Queued DMA
    //while(getRegStatus((Uint32*)&sataRegs-
>P0IS, AHCI_P0IS_DPS) == 0); // getRegStatus(ptr2int,field Mask)

    while(getRegStatus((Uint32*)&sataRegs-
>P0CI, (1<<cmdSlot2Use)) == (1<<cmdSlot2Use));    // Wait Until P0CI[ChHeader] to be cleared by
HBA
```

```
    // Clears both P0IS and IS register. Only Clears RWC bit fields. So, it is OK for this task.
    clearIntOrErrorDiag(INTFIELDS, sataRegs->P0IS);      // Clear P0IS Register

}
```

20.4 SATA Registers

Due to the support of a standard controller, that is compliant with the AHCI 1.1 specifications, the memory-map of the controller and the register descriptions matches the memory-map documented within the standard. For features that are not part of the standard or are implementer specific, (for example, DMA burst control or built-in self test) the reserved locations are used to document the necessary registers required by the non-standard features.

The sub-system core contains register space for global host programming and space for port programming (the device supports two Host Ports and confirms to the AHCI specification calls for the support of multiple Host ports and the register space partitioning comes from this perspective). All registers that start below offset address 100h are global and meant to apply to the HBA, the registers settings entered at the global region applies to both HBA ports. The port control registers reside at offset 100h and above. Dedicated registers that pertains to the specific port reside within this space. There are as many register banks as there are ports (for the device two banks of registers are available for use).

The subsystem core contains register space for global host programming and space for port programming (the AHCI specification calls for the support of multiple Host ports and the register space partitioning comes from this perspective). All registers that start below offset address 100h are global and meant to apply to the entire HBA, the registers settings entered at the global region applies to both HBA ports. The Port 0 control registers reside at offset 100h and above while the Port 1 registers reside at offset 180h and above. Dedicated registers that pertains to the specific port reside within this space. There are as many register banks as there are ports (for this device, two banks of registers are available).

Table 20-3 lists the registers of the SATA. For the base address of these registers, see Table 1-13.

Table 20-3. SATA Controller Registers

Address Offset	Acronym	Register Description	Section
0h	CAP[1]	HBA Capabilities Register	Section 20.4.1
4h	GHC	Global HBA Control Register	Section 20.4.2
8h	IS	Interrupt Status Register	Section 20.4.3
Ch	PI	Ports Implemented Register	Section 20.4.4
10h	VS	AHCI Version Register	Section 20.4.5
14h	CCC_CTL	Command Completion Coalescing Control Register	Section 20.4.6
18h	CCC_PORTS	Command Completion Coalescing Ports Register	Section 20.4.7
A0h	BISTAFR	BIST Active FIS Register	Section 20.4.8
A4h	BISTCR	BIST Control Register	Section 20.4.9
A8h	BISTFCTR	BIST FIS Count Register	Section 20.4.10
ACh	BISTSR	BIST Status Register	Section 20.4.11
B0h	BISTDECR	BIST DWORD Error Count Register	Section 20.4.12
E0h	TIMER1MS[1]	BIST DWORD Error Count Register	Section 20.4.13
E8h	GPARAM1R	Global Parameter 1 Register	Section 20.4.14
ECh	GPARAM2R	Global Parameter 2 Register	Section 20.4.15
F0h	PPARAMR	Port Parameter Register	Section 20.4.16
F4h	TESTR	Test Register	Section 20.4.17
F8h	VERSIONR	Version Register	Section 20.4.18
FCh	IDR	ID Register	Section 20.4.19
100h	P0CLB	Port 0 Command List Base Address Register	Section 20.4.20
108h	P0FB	Port 0 FIS Base Address Register	Section 20.4.21

(1) This register requires a one-time initialization after power-up. The register is written once after a hard reset by firmware, and then remains as read-only thereafter. The register is not affected by software reset.

Table 20-3. SATA Controller Registers (continued)

Address Offset	Acronym	Register Description	Section
110h	P0IS	Port 0 Interrupt Status Register	Section 20.4.22
114h	P0IE	Port 0 Interrupt Enable Register	Section 20.4.23
118h	P0CMD	Port 0 Command Register	Section 20.4.24
120h	P0TFD	Port 0 Task File Data Register	Section 20.4.25
124h	P0SIG	Port 0 Signature Register	Section 20.4.26
128h	P0SSTS	Port 0 Serial ATA Status Register	Section 20.4.27
12Ch	P0SCTL	Port 0 Serial ATA Control Register	Section 20.4.28
130h	P0SERR	Port 0 Serial ATA Error Register	Section 20.4.29
134h	P0SACT	Port 0 Serial ATA Active Register	Section 20.4.30
138h	P0CI	Port 0 Command Issue Register	Section 20.4.31
13Ch	P0SNTF	Port 0 Serial ATA Notification Register	Section 20.4.32
170h	P0DMACR	Port 0 DMA Control Register	Section 20.4.33
178h	P0PHYCR	Port 0 PHY Control Register	Section 20.4.34
17Ch	P0PHYSR	Port 0 PHY Status Register	Section 20.4.35
180h	P1CLB	Port 1 Command List Base Address Register	Section 20.4.20
188h	P1FB	Port 1 FIS Base Address Register	Section 20.4.21
190h	P1IS	Port 1 Interrupt Status Register	Section 20.4.22
194h	P1IE	Port 1 Interrupt Enable Register	Section 20.4.23
198h	P1CMD	Port 1 Command Register	Section 20.4.24
1A0h	P1TFD	Port 1 Task File Data Register	Section 20.4.25
1A4h	P1SIG	Port 1 Signature Register	Section 20.4.26
1A8h	P1SSTS	Port 1 Serial ATA Status Register	Section 20.4.27
1ACh	P1SCTL	Port 1 Serial ATA Control Register	Section 20.4.28
1B0h	P1SERR	Port 1 Serial ATA Error Register	Section 20.4.29
1B4h	P1SACT	Port 1 Serial ATA Active Register	Section 20.4.30
1B8h	P1CI	Port 1 Command Issue Register	Section 20.4.31
1BCh	P1SNTF	Port 1 Serial ATA Notification Register	Section 20.4.32
1F0h	P1DMACR	Port 1 DMA Control Register	Section 20.4.33
1F8h	P1PHYCR	Port 1 PHY Control Register	Section 20.4.34
1FCh	P1PHYSR	Port 1 PHY Status Register	Section 20.4.35
1100h	IDLE	Idle Register	Section 20.4.36
1104h	PHYCFGR2	PHY Configuration Register 2	Section 20.4.37

20.4.1 HBA Capabilities Register (CAP)

The HBA capabilities register (CAP) indicates basic capabilities of the DWC SATA AHCI to the driver software.

The CAP register is shown in Figure 20-2 and described in Table 20-4.

Figure 20-2. HBA Capabilities Register (CAP)

31	30	29	28	27	26	25	24	23	20	19	18	17	16
S64A	SNCQ	SSNTF	SMPS	SSS	SALP	SAL	SCLO	ISS		SNZO	SAM	SPM	Rsvd
R-0	R-1	R-1	W/RO-0	W/RO-0	R-1	R-1	R-1	R-2h		R-0	R-1	R-1	R-0

15	14	13	12	8	7	6	5	4	0
PMD	SSC	PSC	NCS		CCCS	EMS	SXS	NP	
R-1	R-1	R-1	R-1Fh		R-1	R-0	R-0	R-1	

LEGEND: R/W = Read/Write; R = Read only; W/RO: written once after hard reset by firmware, then remain as read-only; *-n* = value after reset

Table 20-4. HBA Capabilities Register (CAP) Field Descriptions

Bit	Field	Value	Description
31	S64A	0	Indicates Support for 64-Bit Addressing. The SATASS only supports 32-bit addressing so this bit is always 0.
30	SNCQ	1	Supports Native Command Queuing. SATASS supports SATA native command queuing by handling DMA Setup FIS natively.
29	SSNTF	1	Supports SNotification Register. SATASS supports P0SNTF (SNotification) register and its associated functionality.
28	SMPS	0	Supports Mechanical Presence Switch. This bit is set by firmware when the platform supports a mechanical presence switch for hot plug operation. This should be written with Zero value. Can use GPIO to supplement this task.
27	SSS	0	Supports Staggered Spin-Up. Only writable once after power up. Set this bit during Firmware Initialization prior to attempting to spin up device (P0CMD[SUD]).
26	SALP	1	Supports Aggressive Link Power Management. SATASS supports auto-generating (Port-initiated) Link Layer requests to the PARTIAL or SLUMBER power management states when there are no commands to process.
25	SAL	1	Supports Activity LED.
24	SCLO	1	Supports Command List Override. Supports the P0CMD.CLO bit functionality for Port Multiplier devices enumeration.
23-20	ISS	2h	Interface Speed Support. SATASS Supports 1.5 and 3 Gbps.
19	SNZO	0	Supports Non-Zero DMA Offsets. Not supported.
18	SAM	1	Supports AHCI Mode Only. SATASS supports AHCI mode only and does not support legacy, task-file based register interface.
17	SPM	1	Supports Port Multiplier. SATASS supports command-based switching Port Multiplier on any of its Ports.
16	Reserved	0	Reserved.
15	PMD	1	PIO Multiple DRQ Block. SATASS supports multiple DRQ block data transfers for the PIO command protocol.
14	SSC	1	Slumber State Capable. SATASS supports transitions to the interface SLUMBER power management state.
13	PSC	1	Partial State Capable. SATASS supports transitions to the interface PARTIAL power management state.
12-8	NCS	1Fh	Number of Command Slots. SATASS supports 32 command slots per Port.
7	CCCS	1	Command Completion Coalescing Supported. SATASS supports command completion coalescing.
6	EMS	0	Enclosure Management Supported. Enclosure Management is not supported.
5	SXS	1	Supports External SATA.

Table 20-4. HBA Capabilities Register (CAP) Field Descriptions (continued)

Bit	Field	Value	Description
4-0	NP		Number of Ports. 0's based value indicating the number of Ports supported by the SATSS.
		0	1 Port
		1	2 Ports
		2h - Fh	Reserved

20.4.2 Global HBA Control Register (GHC)

The global HBA control register (GHC) provides various global functions for the SATASS.

The GHC register is shown in Figure 20-3 and described in Table 20-5.

Figure 20-3. Global HBA Control Register (GHC)

31	30	16
AE	Reserved	
R-1	R-0	

15	2	1	0
Reserved		IE	HR
R-0		R/W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 20-5. Global HBA Control Register (GHC) Field Descriptions

Bit	Field	Value	Description
31	AE	1	AHCI Enable. This bit is always set since SATASS supports only AHCI mode as indicated by the SAM bit in the HBA capabilities register (CAP) = 1.
30-2	Reserved	0	Reserved.
1	IE		Interrupt Enable. This global bit enables interrupts from the SATASS. This field is reset on Global reset (GHC.HR = 1).
		0	All interrupt sources from all the Ports are disabled (masked).
		1	Interrupts are enabled and any SATASS interrupt event causes interrupt output assertion.
0	HR	0	HBA Reset. When set by the software, this bit causes an internal Global reset of the SATASS. All state machines that relate to data transfers and queuing return to an idle state, and all the Ports are reinitialized by sending COMRESET if staggered spin-up is not supported. If staggered spin-up is supported, then it is the responsibility of the software to spin-up each Port after this reset has completed. The SATASS clears this bit when the reset action is done. A software write of 0 has no effect.

20.4.3 Interrupt Status Register (IS)

The interrupt status register (IS) indicates which port inside of the subsystem has a pending interrupt.

The IS register is shown in Figure 20-4 and described in Table 20-6.

Figure 20-4. Interrupt Status Register (IS)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		IPS[*n*]	
R-0		R/W1C-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 20-6. Interrupt Status Register (IS) Field Descriptions

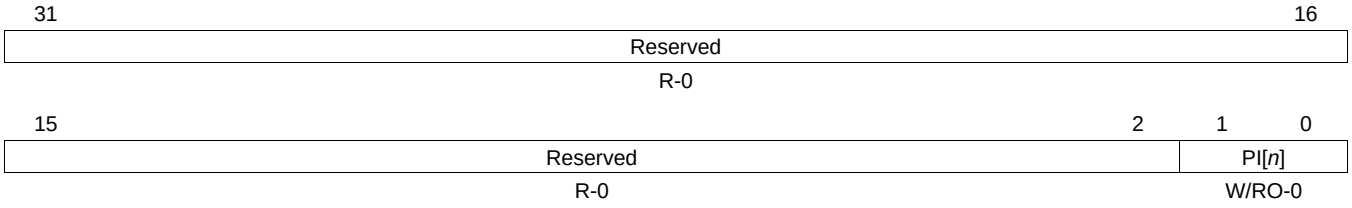
Bit	Field	Value	Description
31-2	Reserved	0	Reserved.
1-0	IPS[*n*]	0-1	Interrupt Pending Status (IPS[0] is for Port 0 and IPS[1] is for Port[1]).. If a bit *n* is set to 1 If set, the corresponding Port has an interrupt pending. Software can use this information to determine which Ports require service after an interrupt. The bits of this field are set by the Ports that have interrupt events pending in the port interrupt status register (P#IS) bits and enabled by the port interrupt enable register (P#IE) bits. Set bits are cleared by the software writing 1 to all bits to clear.

20.4.4 Ports Implemented Register (PI)

The ports implemented register (PI) indicates which ports are exposed by the SATASS and are available for software to use. It is loaded by the BIOS. For example, the SATASS supports 2 Ports as indicated in the CAP.NP, only Port 1 could be available, while Port 0 is unavailable. During Firmware Initialization the software needs to indicate that Port 0 is not available by setting only PI[bit1] only. This means the Port Enabled is a subset of Port available. Note that a minimum of one port needs to be enabled.

The PI register is shown in Figure 20-5 and described in Table 20-7.

Figure 20-5. Ports Implemented Register (PI)



LEGEND: R/W = Read/Write; R = Read only; W/RO: written once after hard reset by firmware, then remain as read-only; -*n* = value after reset

Table 20-7. Ports Implemented Register (PI) Field Descriptions

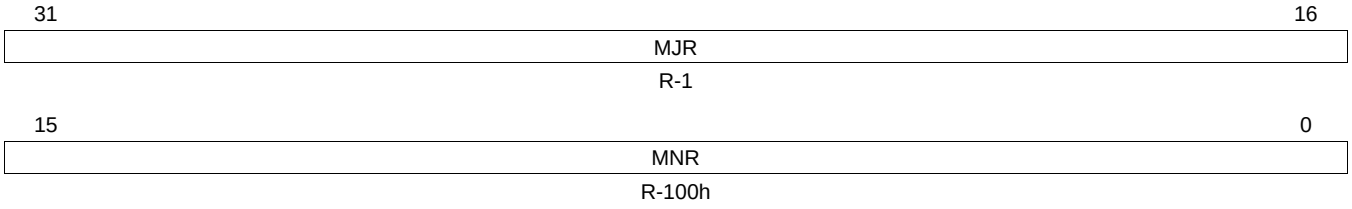
Bit	Field	Value	Description
31-2	Reserved	0	Reserved.
1-0	PI[*n*]	0-1	Ports Implemented. This register is bit significant. If a bit *n* is set to 1, the corresponding Port is available for software to use. If a bit *n* is cleared to 0, the Port is not available for software to use. The maximum number of bits that can be set is 2 for a module with two HBA ports. **Note:** The contents of this register are relevant to the command completion coalescing ports register (CCC_PORTS).

20.4.5 AHCI Version Register (VS)

The AHCI version register (VS) indicates the version of the Synopsis AHCI core embedded in the SATASS.

The VS register is shown in Figure 20-6 and described in Table 20-8.

Figure 20-6. AHCI Version Register (VS)



LEGEND: R = Read only; -*n* = value after reset

Table 20-8. AHCI Version Register (VS) Field Descriptions

Bit	Field	Value	Description
31-16	MJR	1	Major Revision Number.
15-0	MNR	100h	Minor Revision Number.

20.4.6 Command Completion Coalescing Control Register (CCC_CTL)

The command completion coalescing control register (CCC_CTL) is used to configure the command completion coalescing (CCC) feature for the SATASS core. It is reset on Global reset.

The CCC_CTL register is shown in Figure 20-7 and described in Table 20-9.

Figure 20-7. Command Completion Coalescing Control Register (CCC_CTL)

31															16
TV															
R/W-1															
15							8	7				3	2	1	0
CC								INT					Reserved		EN
R/W-1								R-1					R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 20-9. Command Completion Coalescing Control Register (CCC_CTL) Field Descriptions

Bit	Field	Value	Description
31-16	TV	0-FFFFh	Time-out value. This bit field specifies the CCC time-out value in 1 ms intervals. Software loads this value prior to enabling CCC. This bit field is: • Read/Write (R/W) when EN = 0 • Read only (R) when EN = 1 A time-out value of 0 is reserved and should not be used.
15-8	CC	0-FFh	Command Completions. This bit field specifies the number of command completions that are necessary to cause a CCC interrupt. Software loads this value prior to enabling CCC. This bit field is: • Read/Write (R/W) when EN = 0 • Read only (R) when EN = 1 A value of 0 disables CCC interrupts being generated based on the number of commands completed, that is, CCC interrupts are only generated based on the timer in this case.
7-3	INT	0-1Fh	Interrupt. This bit field specifies the interrupt used by the CCC feature, using the number of ports configured for the core. For a single Port instantiation, INT should be programmed to 1. When a CCC interrupt occurs, the IS.IPS[INT] bit is set to 1.
2-1	Reserved	0	Reserved.
0	EN		CCC feature enable. When EN = 1, software can not change the bit fields: TV and CC.
		0	CCC feature is disabled and no CCC interrupts are generated.
		1	CCC feature is enabled and CCC interrupts may be generated based on the time-out or command completion conditions. **Note:** When EN = 1, software can not change the fields : CCC_CTL.TV and CCC_CTL.CC.

20.4.7 Command Completion Coalescing Ports Register (CCC_PORTS)

The command completion coalescing ports register (CCC_PORTS) specifies the Ports that are coalesced as part of the command completion coalescing (CCC) feature when CCC_CTL.EN = 1. It is reset on Global reset.

The CCC_PORTS register is shown in Figure 20-8 and described in Table 20-10.

Figure 20-8. Command Completion Coalescing Ports Register (CCC_PORTS)

31	2	1	0
Reserved		PRT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 20-10. Command Completion Coalescing Ports Register (CCC_PORTS) Field Description

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	PRT		This field is bit significant. Each bit corresponds to a particular Port, where bit 0 corresponds to Port 0 and bit 1 corresponds to Port 1. Bits set to 1 must also have the corresponding bit set to 1 in the ports implemented register (PI).
		0	The corresponding Port is not part of the CCC feature.
		1	The corresponding Port is part of the CCC feature.

20.4.8 BIST Active FIS Register (BISTAFR)

The BIST active FIS register (BISTAFR) is shown in Figure 20-9 and described in Table 20-11.

Figure 20-9. BIST Active FIS Register (BISTAFR)

31	16
Reserved	
R-0	

15	8	7	0
NCP		PD	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 20-11. BIST Active FIS Register (BISTAFR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-8	NCP	0-FFh	Non-Compliant Pattern. Least significant byte of the received BIST Activate FIS second DWORD (bits [7:0]). This value defines the required pattern for the far-end transmit only modes (PD=80h or A0h). If none of these values are decoded, the simultaneous switching pattern is transmitted by default.
		0-49h	Reserved
		4Ah	High frequency test pattern (HFTP)
		4Bh-77h	Reserved
		78h	Mid frequency test pattern (MFTP)
		79h-7Dh	Reserved
		7Eh	Low frequency test pattern (LFTP)
		7Fh	Simultaneous switching outputs pattern (SSOP)
		80h-8Ah	Reserved
		8Bh	Lone Bit pattern (LBP)
		8Ch-AAh	Reserved
		ABh	Low frequency spectral component pattern (LFSCP)
		ACh-B4h	Reserved
		B5h	High transition density pattern (HTDP)
		B6h-F0h	Reserved
		F1h	Low transition density pattern (LTDP)
		F2h-FFh	Reserved
7-0	PD	0-FFh	Pattern Definition. Indicates the pattern definition field of the received BIST Activate FIS (bits [23:16]) of the first DWORD. It is used to put the SATASS in one of the following BIST modes. All reserved values should not be used by the device; otherwise, the FIS is negatively acknowledged with R_ERRp. For far-end transmit only modes, the NCP bit field contains the required data pattern.
		0-7h	Reserved
		8h	Far-end analog (if PHY supports this mode)
		9h-Fh	Reserved
		10h	Far-end retimed
		11h-7Fh	Reserved
		80h	Far-end transmit only
		81h-9Fh	Reserved
		A0h	Far-end transmit only with scrambler bypassed
		A1h-FFh	Reserved

20.4.9 BIST Control Register (BISTCR)

The BIST control register (BISTCR) is shown in Figure 20-10 and described in Table 20-12.

Figure 20-10. BIST Control Register (BISTCR)

31							24
Reserved							
R-0							
23				19	18	17	16
Reserved					TXO	CNTCLR	NEALB
R-0					W-0	W-0	R/W-0
15				11	10		8
Reserved					LLC		
R-0					R/W-7h		
7	6	5	4	3			0
Reserved	ERREN	FLIP	PV	PATTERN			
R-0	R/W-0	R/W-0	R/W-0	R/W-0			

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 20-12. BIST Control Register (BISTCR) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved.
18	TXO	0	Transmit Only. This bit is used to initiate transmission of one of the non-compliant patterns defined by the BISTCR.PATTERN value when the device is disconnected.
17	CNTCLR	0	Counter Clear. This bit clears BIST error count registers. Writing a 1 clears BISTFCTR, BISTSR, and BISTDECR registers.
16	NEALB		Near-end Analog Loopback. Places the Port PHY into near-end analog loopback mode.
		0	Near-end analog loopback is not requested.
		1	Initiates a near-end analog loopback request. BISTCR.CP field contains the appropriate pattern. This mode should be initiated either in the PARTIAL or SLUMBER power mode, or with the device disconnected from the Port PHY (Link NOCOMM state). BIST Activate FIS is not sent to the device in this mode.
15-11	Reserved	0	Reserved.
10-8	LLC	0-7h	Link Layer Control. This bit field controls the Port Link Layer functions: scrambler, descrambler, and repeat primitive drop (RPD). In normal mode, the functions scrambler, descrambler, or RPD are changed only during Port reset (P0SCTL.DET = 1). Note the different meanings for normal and BIST modes of operation:
			Bit 10 (RPD):
		0	Repeat primitive drop function is disabled in normal mode, enabled in BIST mode.
		1	Repeat primitive drop function is enabled in normal mode, disabled in BIST mode.
			Bit 9 (DESCRAM):
		0	Descrambler is disabled in normal mode, enabled in BIST mode.
		1	Descrambler is enabled in normal mode, disabled in BIST mode.
			Bit 8 (SCRAM):
		0	Scrambler is disabled in normal mode, enabled in BIST mode.
		1	Scrambler is enabled in normal mode, disabled in BIST mode.
			The SCRAM bit is cleared (enabled) by the Port when the Port enters a responder far-end transmit BIST mode with scrambling enabled (BISTAFR.PD = 80h).
7	Reserved	0	Reserved.
6	ERREN		Error Enable. Used to allow or filter (disable) PHY internal errors outside the FIS boundary to set corresponding port serial ATA error register (P0SERR) bits.
		0	Filter errors outside the FIS, allow errors inside the FIS.
		1	Allow errors outside or inside the FIS.
5	FLIP	0-1	Flip Disparity. Enables changing disparity of the current test pattern to the opposite every time its state is changed by software.

Table 20-12. BIST Control Register (BISTCR) Field Descriptions (continued)

Bit	Field	Value	Description
4	PV		Pattern Version. Selects either short or long version of the SSOP, HTDP, LTDP, LFSCP, and COMP patterns.
		0	Short pattern version
		1	Long pattern version
3-0	PATTERN	0-Fh	Defines one of the following SATA compliant patterns for far-end retimed/ far-end analog/near-end analog initiator modes, or non-compliant patterns for transmit-only responder mode when initiated by software writing to the BISTCR.TXO bit.
		0	Simultaneous switching outputs pattern (SSOP)
		1h	High-transition density pattern
		2h	Low-transition density pattern
		3h	Low-frequency spectral component pattern (LFSCP)
		4h	Composite pattern (COMP)
		5h	Lone bit pattern (LBP)
		6h	Mid-frequency test pattern (MFTP)
		7h	High-frequency test pattern (HFTP)
		8h	Low-frequency test pattern (LFTP)
		9h-Fh	Reserved.

20.4.10 BIST FIS Count Register (BISTFCTR)

The BIST FIS count register (BISTFCTR) contains the received BIST FIS count in the loopback initiator far-end retimed, far-end analog, and near-end analog modes. It is updated each time a new BIST FIS is received. It is reset by Global reset, Port reset (COMRESET) or by setting the BISTCR.CNTCLR bit. This register does not roll over and freezes when the FFFF FFFFh value is reached. It takes approximately 65 hours of continuous BIST operation to reach this value.

The BISTFCTR register is shown in Figure 20-11 and described in Table 20-13.

Figure 20-11. BIST FIS Count Register (BISTFCTR)

31	0
BISTFCTR	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 20-13. BIST FIS Count Register (BISTFCTR) Field Description

Bit	Field	Value	Description
31-0	BISTFCTR	0-FFFF FFFFh	Received BIST FIS Count.

20.4.11 BIST Status Register (BISTSR)

The BIST status register (BISTSR) contains errors detected in the received BIST FIS in the loopback initiator far-end retimed, far-end analog, and near-end analog modes. It is not meaningful in responder mode. It is updated each time a new BIST FIS is received. It is reset by Global reset, Port reset (COMRESET) or by setting the BISTCR.CNTCLR bit.

The BISTSR register is shown in Figure 20-12 and described in Table 20-14.

Figure 20-12. BIST Status Register (BISTSR)

31	24	23	16
Reserved		BRSTERR	
R-0		R-0	

15	0
FRAMERR	
W-0	

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 20-14. BIST Status Register (BISTSR) Field Description

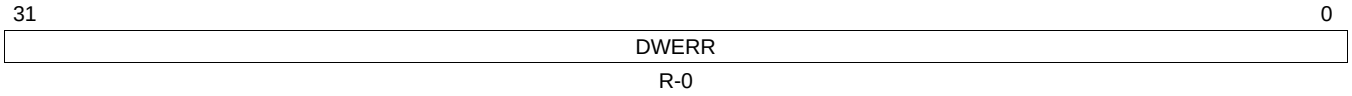
Bit	Field	Value	Description
31-24	Reserved	0	Reserved.
23-16	BRSTERR	0-FFh	Burst Error. This bit field contains the burst error count. It is accumulated each time a burst error condition is detected: DWORD error is detected in the received frame and 1.5 seconds (27,000 frames) passed since the previous burst error was detected. The BRSTERR value does not roll over and freezes at FFh. This bit field is updated if BIST_MODE = DWORD.
15-0	FRAMERR	0-FFFFh	Frame Error. This bit field contains the frame error count. It is accumulated (new value is added to the old value) each time a new BIST frame with a CRC error is received. The FRAMERR value does not roll over and freezes at FFFFh.

20.4.12 BIST DWORD Error Count Register (BISTDECR)

The BIST DWORD error count register (BISTDECR) contains the number of DWORD errors detected in the received BIST frame in the loopback initiator far-end retimed, far-end analog, and near-end analog modes. It is updated each time a new BIST frame is received. It is reset by Global reset, Port reset (COMRESET) or by setting the BISTCR.CNTCLR bit. This register is updated only when the parameter BIST_MODE=DWORD.

The BISTDECR register is shown in Figure 20-13 and described in Table 20-15.

Figure 20-13. BIST DWORD Error Count Register (BISTDECR)



LEGEND: R = Read only; -*n* = value after reset

Table 20-15. BIST DWORD Error Count Register (BISTDECR) Field Description

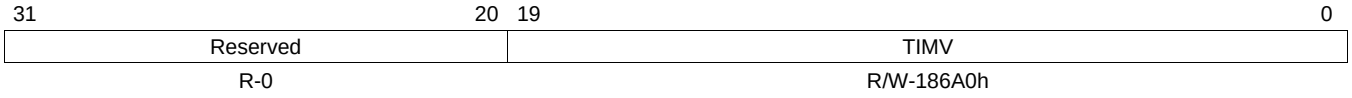
Bit	Field	Value	Description
31-0	DWERR	0-FFFF FFFFh	DWORD Error Count. This bit field contains the DWORD error count. It is accumulated (new value is added to the old value) each time a new BIST frame is received. The DWERR value does not roll over and freezes if it exceeds FFFF F000h.

20.4.13 BIST DWORD Error Count Register (TIMER1MS)

The BIST DWORD error count register (TIMER1MS) is used to generate 1ms tick for the command completion coalescing (CCC) logic based on the OCP clock frequency sourced to the SATA controller. Software must initialize this register with the required value after power up before using the CCC feature. This register is reset to 100,000d (corresponding to a VBUS Clock frequency of 100 MHz hclk) on power up and is not affected by Global reset.

The TIMER1MS register is shown in Figure 20-14 and described in Table 20-16.

Figure 20-14. BIST DWORD Error Count Register (TIMER1MS)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 20-16. BIST DWORD Error Count Register (TIMER1MS) Field Description

Bit	Field	Value	Description
31-20	Reserved	0	Reserved.
19-0	TIMV	0-F FFFFh	1ms Timer Value. This bit field contains the cycle count of the SATA VBUS Clock Cycle generating 1ms tick. This bit field is: • Read/Write (R/W) when CCC_CTL.EN = 0 • Read only (R) when CCC_CTL.EN = 1

20.4.14 Global Parameter 1 Register (GPARAM1R)

The global parameter 1 register (GPARAM1R) is a read-only register that contains encoded information about the configuration of the embedded Synopsis AHCI SATA Core.

The GPARAM1R register is shown in Figure 20-15 and described in Table 20-17.

Figure 20-15. Global Parameter 1 Register (GPARAM1R)

31	30	29	28	27	26	24
ALIGN_M	RX_BUFFER	PHY_DATA		PHY_RST	PHY_CTRL	
R-1	R-1	R-0		R-0	R-1Ah	

23	21	20	15
PHY_CTRL		PHY_STAT	
R-1Ah		R-2h	

14	13	12	11	10	9	8
LATCH_M	BIST_M	PHY_TYPE	Reserved	RETURN_ERR	AHB_ENDIAN	
R-0	R-0	R-0	R-0	R-1	R-2h	

7	6	5	3	2	0
S_HADDR	M_HADDR	S_HDATA		M_HDATA	
R-0	R-0	R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 20-17. Global Parameter 1 Register (GPARAM1R) Field Descriptions

Bit	Field	Value	Description
31	ALIGN_M	1	Rx Data Alignment. Data is always aligned.
30	RX_BUFFER	1	Rx Data Buffer. Core includes an Rx Data Buffer.
29-28	PHY_DATA	0	PHY Data Width. Indicates width = 0 (8-bits).
27	PHY_RST	0	PHY Reset Mode. Indicates that the PHY reset output is active-low.
26-21	PHY_CTRL	1Ah	PHY Control Width. Indicates that there are 32-bits of PHY control.
20-15	PHY_STAT	2h	PHY Status Width. Indicates that there are 32-bits of PHY status.
14	LATCH_M	0	Latch Mode. Indicates that the subsystem does not include latches.
13	BIST_M	0	BIST Loopback Checking Depth. Checks errors per FIS not DWORD.
12	PHY_TYPE	0	PHY Interface Type. Indicates a non-Synopsis PHY.
11	Reserved	0	Reserved.
10	RETURN_ERR	1	AHB Error Response. Indicates AHB Errors are returned.
9-8	AHB_ENDIAN	2h	Bus Endianness. Indicates that endianness may be configured by input pin.
7	S_HADDR	0	Slave address bus width. Indicates 32-bit wide address bus.
6	M_HADDR	0	Master address bus width. Indicates 32-bit wide address bus.
5-3	S_HDATA	0	Slave Data Bus Width. Indicates 32-bit data bus.
2-0	M_HDATA	0	Master Data Bus Width. Indicates 32-bit data bus.

20.4.15 Global Parameter 2 Register (GPARAM2R)

The global parameter 2 register (GPARAM2R) is a read-only register that contains encoded information about the configuration of the embedded Synopsis AHCI SATA Core.

The GPARAM2R register is shown in Figure 20-16 and described in Table 20-18.

Figure 20-16. Global Parameter 2 Register (GPARAM2R)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DEV_CP	DEV_MP	ENCODE_M	RXOOB_CLK_M	RXOOB_M	TX_OOB_M	RXOOB_CLK								
R-0	R-1	R-1	R-1	R-0	R-0	R-1	R-96h								

LEGEND: R = Read only; -*n* = value after reset

Table 20-18. Global Parameter 2 Register (GPARAM2R) Field Descriptions

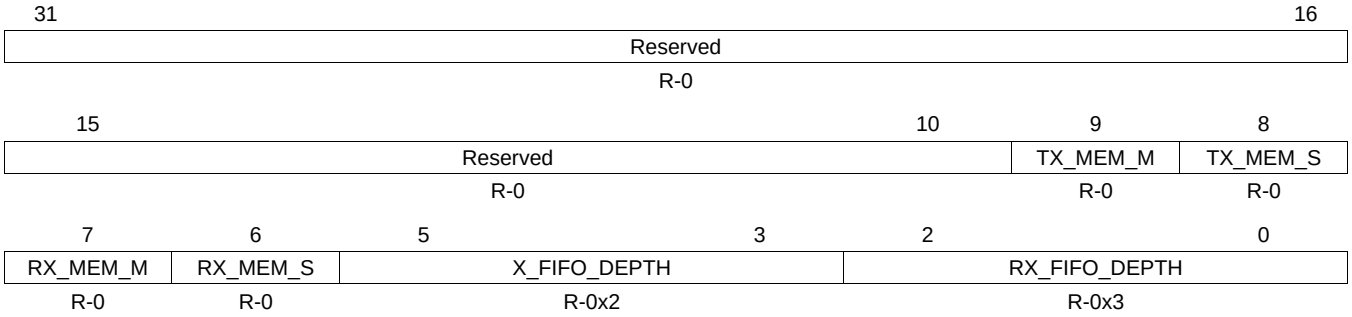
Bit	Field	Value	Description
31-15	Reserved	0	Reserved.
14	DEV_CP	0	Cold Presence Detect. CPD is supported in SATASS.
13	DEV_MP	0	Mechanical Presence Switch. MP is supported in SATASS.
12	ENCODE_M	1	8b/10b Encoding/Decoding. Indicates 8-bit/10-bit encoding and decoding are done in the Synopsis core.
11	RXOOB_CLK_M	0	Rx OOB Clock Mode. Indicates that the Rx OOB Functions are on the Rx Clock.
10	RX_OOB_M	1	Rx OOB Mode. Indicates that Rx Out-of-Band signals are detected in the PHY.
9	TX_OOB_M	1	Tx OOB Mode. Indicates that Tx Out-of-Band signaling is done by the Synopsis Core.
8-0	RXOOB_CLK	96h	Rx OOB Clock Frequency. Reflects the hexadecimal value of the RXOOB_CLK_FREQ parameter.

20.4.16 Port Parameter Register (PPARAMR)

The port parameter register (PPARAMR) is a read-only register that contains encoded information about the selected Port configuration parameters settings. The Port is selected by the TESTR.PSEL field. Note, there are two HBA ports on this sub-system and the configuration setting applies to both ports.

The PPARAMR register is shown in Figure 20-17 and described in Table 20-19.

Figure 20-17. Port Parameter Register (PPARAMR)



LEGEND: R = Read only; -*n* = value after reset

Table 20-19. Port Parameter Register (PPARAMR) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved.
9	TX_MEM_M	0	Tx FIFO Memory Read Port Type. Indicates that the Tx FIFO memory is asynchronous read.
8	TX_MEM_S	0	Tx FIFO Memory Type. Indicates that the Tx FIFO memory is outside of the core (but still inside the subsystem wrapper).
7	RX_MEM_M	0	Rx FIFO Memory Read Port Type. Indicates that the Rx FIFO memory is asynchronous read.
6	RX_MEM_S	0	Rx FIFO Memory Type. Indicates that the Rx FIFO memory is outside of the core (but still inside the subsystem wrapper).
5-3	TX_FIFO_DEPTH	2h	Tx FIFO Depth. Indicates that the depth of the Tx FIFO is 64 DWORDS.
2-0	RX_FIFO_DEPTH	3h	Rx FIFO Depth. Indicates that the depth of the Rx FIFO is 128 DWORDS.

20.4.17 Test Register (TESTR)

The test register (TESTR) is used to put the SATASS slave interface into a test mode and to select a Port for BIST operation.

The TESTR register is shown in Figure 20-18 and described in Table 20-20.

Figure 20-18. Test Register (TESTR)

31	19	18	16
Reserved		PSEL	
R-0		R/W-0	

15	0
Reserved	TEST_IF
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 20-20. Test Register (TESTR) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved.
18-16	PSEL	0	Port Select. Selects the port for BIST operation. Note that there are two ports in this subsystem, so 0 and 1 are valid values.
15-1	Reserved	0	Reserved.
0	TEST_IF		Test Interface. Places the DWC SATA AHCI slave interface into the test mode.
		0	Normal mode: the read back value of some registers is a function of the DWC SATA AHCI state and does not match the value written.
		1	Test mode: the read back value of the registers matches the value written. Normal operation is disabled. The following registers are accessed in this mode: • GHC register: IE bit • BISTAFR register: NCP and PD bits become read/write • BISTCR register: LLC, ERREN, FLIP, PV, PATTERN • BISTFCTR, BISTSR, BISTDECR registers become read/write • P#CLB (# = 0 or 1)/CLBU, P#FB (# = 0 or 1)/FBU registers • P#IS (# = 0 or 1) register: RW1C and UFS bits become read/write • P#IE (# = 0 or 1) register • P#CMD (# = 0 or 1) register: ASP, ALPE, DLAE, ATAPI, PMA bits • P#TFD (# = 0 or 1), P#SIG (# = 0 or 1) registers become read/write • P#SCTL (# = 0 or 1) register • P#SERR (# = 0 or 1) register: R/W1C bits become read/write bits • P#SACT (# = 0 or 1), P#CI (# = 0 or 1), P#SNTF (# = 0 or 1) registers become read/write • P#DMACR (# = 0 or 1) register • P#PHYCR (# = 0 or 1) register • P#PHYSR (# = 0 or 1) register becomes read/write Notes: Where # = Port 0 or Port 1 above, the PSEL field will select the Port Number under test. Interrupt is asserted if any of the IS register bits is set after setting the corresponding P0IS and P0IE registers and GHC.IE = 1. CAP.SMPS/SSS, PI, P0CMD.ESP/CPD/MPSP/ HPCP register bits are W/RO (written once after power-on reset, then remain as read-only) type and can not be used in test mode. Global DWC SATA AHCI reset must be issued (GHC.HR = 1) after TEST_IF bit is cleared following the test mode operation.

20.4.18 Version Register (VERSIONR)

The version register (VERSIONR) contains the 32-bit SATASS version.

The VERSIONR register is shown in Figure 20-19 and described in Table 20-21.

Figure 20-19. Version Register (VERSIONR)

LEGEND: R = Read only; -*n* = value after reset

Table 20-21. Version Register (VERSIONR) Field Description

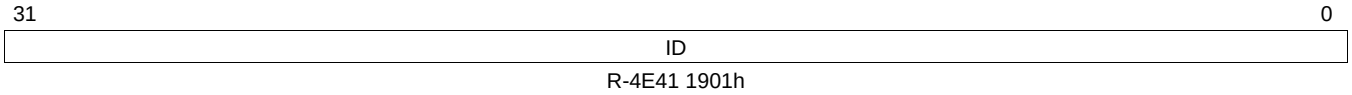
Bit	Field	Value	Description
31-0	VERSION	3134 302Ah	32-bit version of SATASS.

20.4.19 ID Register (IDR)

The ID register (IDR) contains the 32-bit SATASS ID.

The IDR register is shown in Figure 20-20 and described in Table 20-22.

Figure 20-20. ID Register (IDR)



LEGEND: R = Read only; -*n* = value after reset

Table 20-22. ID Register (IDR) Field Description

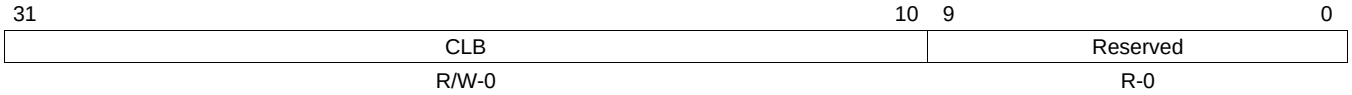
Bit	Field	Value	Description
31-0	ID	4E41 1901h	32-bit ID of SATASS.

20.4.20 Port Command List Base Address Register (P#CLB) (# = 0 or 1)

The port command list base address register (P#CLB) contains the 32-bit base address of the command list.

The P#CLB register is shown in Figure 20-21 and described in Table 20-23.

Figure 20-21. Port Command List Base Address Register (P#CLB)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 20-23. Port Command List Base Address Register (P#CLB) Field Description

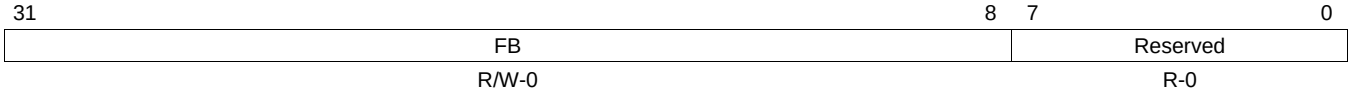
Bit	Field	Value	Description
31-10	CLB	0-3F FFFFh	Command List Base Address. Indicates the 32-bit base physical address for the command list for this port. This base is used when fetching commands to execute. The structure pointed to by this address range is 1Kbyte in length. This address must be 1Kbyte-aligned as indicated by bits [9:0] being read-only.
9-0	Reserved	0	Reserved.

20.4.21 Port FIS Base Address Register (P#FB) (# = 0 or 1)

The port FIS base address register (P#FB) contains the 32-bit base address of the destination for incoming received FISes.

The P#FB register is shown in Figure 20-22 and described in Table 20-24.

Figure 20-22. Port FIS Base Address Register (P#FB)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 20-24. Port FIS Base Address Register (P#FB) Field Description

Bit	Field	Value	Description
31-8	FB	0-FF FFFFh	FIS Base Address. Indicates the 32-bit base physical address for received FISes. The structure pointed to by this address range is 256 bytes in length. This address must be 256 byte-aligned as indicated by bits [7:0] being read-only.
7-0	Reserved	0	Reserved.

20.4.22 Port Interrupt Status Register (P#IS) (# = 0 or 1)

The port interrupt status register (P#IS) is used to generate SATASS interrupts when any of the bits are set. Bits in this register are set by some internal conditions, and cleared by software writing ones in the positions it wants to clear. This register is reset on Global reset.

The P#IS register is shown in Figure 20-23 and described in Table 20-25.

Figure 20-23. Port Interrupt Status Register (P#IS)

31	30	29	28	27	26	25	24	23	22	21	16
Rsvd	TFES	HBFS	HBDS	IFS	INFS	Rsvd	OFS	IPMS	PRCS	Reserved	
R-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R-0	R/W1C-0	R/W1C-0	R-0	R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		DMPS	PCS	DPS	UFS	SDBS	DSS	PSS	DHRS
R-0		R/W1C-0	R-0	R/W1C-0	R-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 20-25. Port Interrupt Status Register (P#IS) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved. Since no dedicated pins for CPD detection/control exist for the device, this bit is reserved. If need to support this feature, use of GPIO is recommended.
30	TFES	0-1	Task File Error Status. This bit is set whenever the P0TFD register is updated by the device and the error bit (P0TFD.STS[0]) is set.
29	HBFS	0-1	Host Bus Fatal Error Status. This bit is set when SATASS bus master detects an ERROR response from the slave.
28	HBDS	0-1	Host Bus Data Error Status. This bit is always cleared to 0.
27	IFS	0-1	Interface Fatal Error Status. This bit is set if any of the following conditions is detected: • SYNC escape is received from the device during H2D Register or Data FIS transmission. • One or more of the following errors are detected during Data FIS transfer: Protocol (P0SERR.ERR_P), CRC (P0SERR.DIAG_C), Handshake (P0SERR.DIAG_H), PHY Not Ready (P0SERR.ERR_C). • Unknown FIS is received with good CRC, but the length exceeds 64 bytes. • PRD table byte count is zero. Port DMA transitions to a fatal state until software clears P0CMD.ST bit or resets the interface by way of Port or Global reset.
26	INFS	0-1	Interface Non-fatal Error Status. This bit is set if any of the following conditions is detected: • One or more of the following errors are detected during non-data FIS transfer: Protocol (P0SERR.ERR_P), CRC (P0SERR.DIAG_C), Handshake (P0SERR.DIAG_H), PHY Not Ready (P0SERR.ERR_C). • Command list underflow during read operation (DMA read) when software builds command table that has more total bytes than the transaction given to the device. In both cases, Port operation continues normally. If error is detected during non-data FIS transmission, this FIS is retransmitted continuously until it succeeds or software times out and resets the interface.
25	Reserved	0	Reserved.
24	OFS	0-1	Overflow Status. This bit is set if command list overflow is detected during read or write operation when software builds a command table that has fewer total bytes than the transaction given to the device. Port DMA transitions to a fatal state until software clears P0CMD.ST bit or resets the interface by way of Port or Global reset.
23	IPMS	0-1	Incorrect Port Multiplier Status. Indicates that the HBA received a FIS from a device whose Port Multiplier field did not match what was expected. This bit may be set during enumeration of devices on a Port Multiplier due to the normal Port Multiplier enumeration process. The software should only use the IPMS bit after enumeration is complete on the Port Multiplier.
22	PRCS	0-1	PHY Ready Change Status. When set to 1, indicates the internal PHY Ready signal changed state. This bit reflects the state of the P0SERR.DIAG_N bit. To clear this bit, the software must clear the P0SERR.DIAG_N bit to 0.
21-8	Reserved	0	Reserved.

Table 20-25. Port Interrupt Status Register (P#IS) Field Descriptions (continued)

Bit	Field	Value	Description
7	DMPS	0-1	Device Mechanical Presence Status. This bit is set when the state of SATA_MP_SWITCH input pin changes as a result of a mechanical switch attached to this port being opened or closed. This bit is only valid if both CAP.SMPS and P0CMD.MPSP are set to 1.
6	PCS		Port Connect Change Status. This bit reflects the state of the P0SERR.DIAG_X bit. This bit is only cleared when P0SERR.DIAG_X is cleared.
		0	No change in Current Connect Status
		1	Change in Current Connect Status
5	DPS	0-1	Descriptor Processed. A PRD with the I bit set has transferred all of its data. Note: This is an opportunistic interrupt and should not be used to definitely indicate the end of a transfer. Two PRD interrupts could happen close in time together such that the second interrupt is missed when the first PRD interrupt is being cleared.
4	UFS	0-1	Unknown FIS Interrupt. When set to 1, indicates that an unknown FIS was received and has been copied into system memory. This bit is cleared to 0 by software clearing the P0SERR.DIAG_F bit to 0. Note: The UFS bit does not directly reflect the P0SERR.DIAG_F bit. P0SERR.DIAG_F bit is set immediately when an unknown FIS is detected, whereas the UFS bit is set when that FIS is posted to memory. Software should wait to act on an unknown FIS until the UFS bit is set to 1 or the two bits may become out of sync.
3	SDBS	0-1	Set Device Bits Interrupt. A Set Device Bits FIS has been received with the I bit set and has been copied into system memory.
2	DSS	0-1	DMA Setup FIS Interrupt. A DMA Setup FIS has been received with the I bit set and has been copied into system memory.
1	PSS	0-1	PIO Setup FIS Interrupt. A PIO Setup FIS has been received with the I bit set, it has been copied into system memory, and the data related to that FIS has been transferred.
0	DHRS	0-1	Device to Host Register FIS Interrupt. A D2H Register FIS has been received with the I bit set, and has been copied into system memory.

20.4.23 Port Interrupt Enable Register (P#IE) (# = 0 or 1)

The port interrupt enable register (P#IE) enables and disables the reporting of the corresponding interrupt to system software. When a bit is set (1), and the corresponding interrupt condition is active, then the SATASS intrq output is asserted. Interrupt sources that are disabled (0) are still reflected in the status registers. This register is symmetrical with the P#IS register. This register is reset on Global reset.

The P#IE register is shown in Figure 20-24 and described in Table 20-26.

Figure 20-24. Port Interrupt Enable Register (P#IE)

31	30	29	28	27	26	25	24	23	22	21	16
Rsvd	TFEE	HBFE	HBDE	IFE	INFE	Rsvd	OFE	IPME	PRCE	Reserved	
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		DMPE	PCE	DPE	UFE	SDBE	DSE	PSE	DHRE
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 20-26. Port Interrupt Enable Register (P#IE) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved. No CPD capability exist since CPD related pins are not bonded out for the device. If need this feature, use of GPIO is recommended.
30	TFEE	0-1	Task File Error Enable. When set to 1, GHC.IE = 1, and P0IS.HBFS = 1, the intrq output is asserted.
29	HBFE	0-1	Host Bus Fatal Error Enable. When set to 1, GHC.IE = 1, and P0IS.HBFS = 1, the interq output is asserted.
28	HBDE	0-1	Host Bus Data Error Enable. When set to 1, GHC.IE = 1, and P0IS.HBDS = 1, the intrq output is asserted.
27	IFE	0-1	Interface Fatal Error Enable. When set to 1, GHC.IE = 1, and P0IS.IFS = 1, the intrq output is asserted.
26	INFE	0-1	Interface Non-fatal Error Enable. When set to 1, GHC.IE = 1, and P0IS.INFS = 1, the intrq output is asserted.
25	Reserved	0	Reserved.
24	OFE	0-1	Overflow Enable. When set to 1, GHC.IE = 1, and P0IS.OFS = 1, the intrq output is asserted.
23	IPME	0-1	Incorrect Port Multiplier Enable. When set to 1, GHC.IE = 1, and P0IS.IPMS = 1, the intrq output is asserted.
22	PRCE	0-1	PHY Ready Change Enable. When set to 1, GHC.IE = 1, and P0IS.PRCS = 1, the intrq output is asserted.
21-8	Reserved	0	Reserved.
7	DMPE	0-1	Device Mechanical Presence Enable. When set to 1, GHC.IE = 1, and P0IS.DMPS = 1, the intrq output is asserted.
6	PCE	0-1	Port Change Interrupt Enable. When set to 1, GHC.IE = 1, and P0IS.PCS = 1, the intrq output is asserted.
5	DPE	0-1	Descriptor Processed Interrupt Enable. When set to 1, GHC.IE = 1 and P0IS.DPS = 1, the intrq output is asserted.
4	UFE	0-1	Unknown FIS Interrupt Enable. When set to 1, GHC.IE = 1, and P0IS.UFS = 1, the intrq output is asserted.
3	SDBE	0-1	Set Device Bits FIS Interrupt Enable. When set to 1, GHC.IE = 1, and P0IS.SDBS = 1, the intrq output is asserted.
2	DSE	0-1	DMA Setup FIS Interrupt Enable. When set to 1, GHC.IE = 1, and P0IS.PSS = 1, the intrq output is asserted.
1	PSE	0-1	PIO Setup FIS Interrupt Enable. When set to 1, GHC.IE = 1, and P0IS.PSS = 1, the intrq output is asserted.
0	DHRE	0-1	Device Host Register FIS Interrupt Enable. When set to 1, GHC.IE = 1, and P0IS.DHRS = 1, the intrq output is asserted.

20.4.24 Port Command Register (P#CMD) (# = 0 or 1)

The port command register (P#CMD) contains bits controlling various Port functions. All read/write bits are reset on Global reset.

The P#CMD register is shown in Figure 20-25 and described in Table 20-27.

Figure 20-25. Port Command Register (P#CMD)

31	28	27	26	25	24	23	22	21	20	19	18	17	16
ICC		ASP	ALPE	DLAE	ATAPI	Reserved		ESP	Rsvd	MPSP	HPCP	PMA	CPD
R/W-0		R/W-0	R/W-0	R/W-0	R/W-0	R-0		W/RO-0	W/RO-0	W/RO-0	W/RO-0	R/W-0	R-0

15	14	13	12	8	7	5	4	3	2	1	0
CR	FR	MPSS	CCS		Reserved		FRE	CLO	POD	SUD	ST
R-0	R-0	R-0	R-0		R-0		R/W-0	W-0	R/W-0	W/RO-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write; W/RO: written once after hard reset by firmware, then remain as read-only; *-n* = value after reset

Table 20-27. Port Command Register (P#CMD) Field Descriptions

Bit	Field	Value	Description
31-28	ICC	0-Fh	Interface Communication Control. Controls power management states of the interface. If the Link layer is currently in the L_IDLE state, writes to this bit field causes the Port to initiate a transition to the interface power management state requested. If the Link layer is not currently in the L_IDLE state, writes to this bit field have no effect. When system software writes a nonreserved value other than No-Op (0), the Port performs the action and updates this bit field back to Idle (0). If software writes to this bit field to change the state to a state the link is already in (that is, interface is in the active state and a request is made to go to the active state), the Port takes no action and returns this bit field to Idle. If the interface is in a low power state and software wants to transition to a different low power state, software must first bring the link to active and then initiate the transition to the desired low power state.
		0	No-Op/ Idle: When software reads this value, it indicates the Port is ready to accept a new interface control command, although the transition to the previously selected state may not yet have occurred.
		1h	Active: Causes the Port to request a transition of the interface into the active state.
		2h	Partial: Causes the Port to request a transition of the interface to the Partial state. The SATA device may reject the request and the interface remains in its current state.
		3h-5h	Reserved
		6h	Slumber: Causes the Port to request a transition of the interface to the Slumber state. The SATA device may reject the request and the interface remains in its current state.
		7h-Fh	Reserved
27	ASP		Aggressive Slumber/Partial.
		0	When cleared to 0 and P0CMD.ALPE = 1, the Port aggressively enters the PARTIAL state when it clears the P0CI register, and the P0SACT register is cleared when it clears the P0SACT register and P0CI is cleared.
		1	When set to 1 and P0CMD.ALPE = 1, the Port aggressively enters the SLUBMER state when it clears the P0CI register and the P0SACT register is cleared, or when it clears the P0SACT register and P0CI is cleared.
26	ALPE		Aggressive Link Power Management Enable.
		0	Aggressive power management state transition is disabled.
		1	Port aggressively enters a lower link power state (PARTIAL or SLUMBER) based on the setting of the P0CMD.ASP bit.
25	DLAE	0-1	Drive LED on ATAPI Enable. When set to 1, P0CMD.ATAPI = 1, and commands are active, the Port asserts P0_act_led output.
24	ATAPI	0-1	Device is ATAPI. When set to 1, the connected device is an ATAPI device. This bit is used by the Port to control whether or not to assert P0_act_led output when commands are active.
23-22	Reserved	0	Reserved.
21	ESP		External SATA Port. The ESP bit is mutually exclusive with the P0CMD.HPCP bit.
		0	Ports signal only connector is not externally accessible.
		1	Ports signal only connector is externally accessible. When set to 1, CAP.SXS is also set to 1.

Table 20-27. Port Command Register (P#CMD) Field Descriptions (continued)

Bit	Field	Value	Description
20	Reserved	0	Reserved. Since no CPD pins are bonded out for the device, this bit should be written with a zero value. If need to implement CPD functionality, GPIO usage is recommended.
19	MPSP		Mechanical Presence Switch Attached to Port.
		0	Platform does not support a mechanical presence switch on this port.
		1	Platform supports a mechanical presence switch attached to this port. When this bit is set to 1, P0CMD.HPCP should also be set to 1.
18	HPCP		Hot Plug.
		0	Ports signal and power connectors are not externally accessible.
		1	Ports signal and power connectors are externally accessible via a joint signal-power connector for blindmate device hot plug.
17	PMA		Port Multiplier Attached.
			Note: Software is responsible for detecting whether a Port Multiplier is present. There is no auto detection.
		0	When cleared to 0 by software, indicates that a Port Multiplier is not attached to this Port.
		1	When set to 1 by software, indicates that a Port Multiplier is attached to this Port.
16	CPD	0	Since no CPD pins are bonded out for the device, this bit is always zero. if need to implement CPD usage of GPIO is recommended.
15	CR	0-1	Command List Running. When this bit is set to 1, the command list DMA engine for this Port is running. See Synopsys spec for more detail.
14	FR	0-1	FIS Receive Running. When set to 1, the FIS Receive DMA engine for this port is running. Refer to the AHCI specification section 10.3.2 for details on when this bit is set and cleared by the Port.
13	MPSS		Mechanical Presence Switch State. Reports the state of a mechanical presence switch attached to this port as indicated by the P0_mp_switch input state (assuming CAP.SMPS = 1 and P0CMD.MPSP = 1). If CAP.SMPS = 0, then this bit is cleared to 0. Software should only use this bit if both CAP.SMPS and P0CMD.MPSP are set to 1.
			Note: The reset of this bit may change based on the SATA_MP_SWITCH state at reset.
		0	Switch is closed.
		1	Switch is open.
12-8	CCS	0-1Fh	Current Command Slot. This field is valid when P0CMD.ST is set to 1 and is set to the command slot value of the command that is currently being issued by the Port. When P0CMD.ST transitions from 1 to 0, this field is reset to 0x00. After P0CMD.ST transitions from 0 to 1, the highest priority slot to issue from next is command slot 0. After the first command has been issued, the highest priority slot to issue from next is P0CMD.CCS+1. For example, after the Port has issued its first command, if CCS=0x00 and P0CI is set to 0x3, the next command that will be issued is from command slot 1.
7-5	Reserved	0	Reserved.
4	FRE	0-1	FIS Receive Enable. When set to 1, the Port may post received FISes into the FIS receive area pointed to by P#FB. When cleared, received FISes are not accepted by the Port, except for the first D2H register FIS after the initialization sequence, and no FISes are posted to the FIS receive area.
			System software must not set this but until P#FB has been programmed with a valid pointer to the FIS receive area, and if software wishes to move the base, this bit must first be cleared, and software must wait for the P0CMD.FR bit to be cleared. Refer to the Synopsis spec for important restrictions on changing P0CMD.FRE.
3	CLO	1	Command List Override. Setting this bit to 1 causes P0TFD.STS.BSY and P0TFD.STS.DRQ to be cleared to 0. This allows a software reset to be transmitted to the device regardless of whether the BSY and DRQ bits are still set in the P0TFD.STS register. This bit is cleared to 0 when P0TFD.STS.BSY and P0TFD.STS.DRQ have been cleared to 0. A write to this register with a value of 0 has no effect.
			This bit should only be set to 1 immediately prior to setting P0CMD.ST bit to 1 from a previous value of 0. Setting this bit to 1 at any other time is not supported and results in indeterminate behavior.
2	POD	0-1	Power On Device. This bit is RW if cold presence detection is supported on this port as indicated by P0CMD.CPD=1. This bit is RO 1 if cold presence detection is not supported and P0CMD.CPD=0. Note: Since no CPD control/status related pins are bonded out on the device, this bit is read only. Usage of GPIO is recommended if need to supplement CPD feature.
1	SUD	0-1	Spin-Up Device. This bit is read/write if staggered spin-up is supported as indicated by the CAP.SSS = 1. This bit is read-only 1 if staggered spin-up is not supported and CAP.SSS = 0. On an edge detect from 0 to 1, the Port starts a COMRESET initialization sequence to the device. Clearing this bit causes no action on the interface.
			Note: the SUD bit is read-only 0 on power-up until CAP.SSS bit is written with the required value.

Table 20-27. Port Command Register (P#CMD) Field Descriptions (continued)

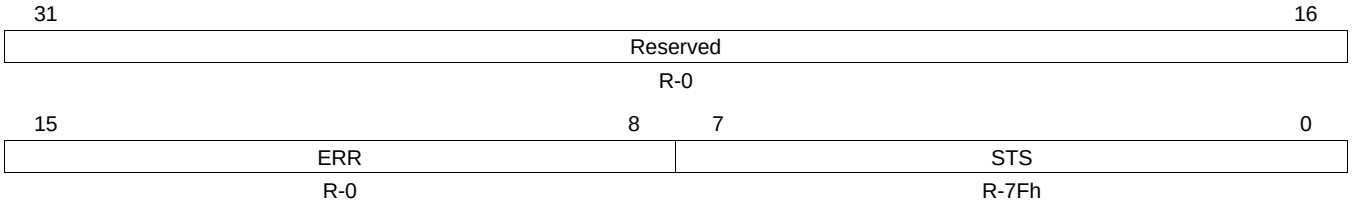
Bit	Field	Value	Description
0	ST	0-1	Start. When set to 1, the Port processes the command list. When cleared, the Port does not process the command list. Whenever this bit is changed from a 0 to a 1, the Port starts processing the command list at entry 0. Whenever this bit is changed from a 1 to a 0, the P0CI register is cleared by the Port upon transition into an idle state. Refer to the AHCI specification for important restrictions on when this bit can be set to 1.

20.4.25 Port Task File Data Register (P#TFD) (# = 0 or 1)

The port task file data register (P#TFD) contains Error and Status registers updated every time a new Register FIS.

The P#TFD register is shown in Figure 20-26 and described in Table 20-28.

Figure 20-26. Port Task File Data Register (P#TFD)



LEGEND: R = Read only; -*n* = value after reset

Table 20-28. Port Task File Data Register (P#TFD) Field Descriptions

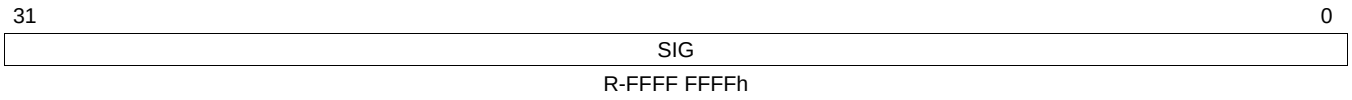
Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-8	ERR	0-FFh	Error. Contains the latest copy of the task file error register.
7-0	STS	0-FFh	Status. Contains the latest copy of the task file status register.
			Note: the HBA updates the entire 8-bit field not just the bits listed.
		0	ERR – Indicates an error during the transfer
		1h-2h	Reserved
		3h	DRQ – Indicates a data transfer is requested
		4h-6h	Reserved
		7h	BSY – Indicates the interface is busy
		8h-FFh	Reserved

20.4.26 Port Signature Register (P#SIG) (# = 0 or 1)

The port signature register (P#SIG) contains the Device Signature information.

The P#SIG register is shown in Figure 20-27 and described in Table 20-29.

Figure 20-27. Port Signature Register (P#SIG)



LEGEND: R = Read only; -*n* = value after reset

Table 20-29. Port Signature Register (P#SIG) Field Description

Bit	Field	Value	Description
31-0	SIG	0-FFFF FFFFh	Signature. Contains the signature received from a device on the first D2H Register FIS. This register is updated once after a reset sequence. Reset on Global or Port reset. The bit order as is:
			Bits 31-24: LBA High (Cylinder High) Register
			Bits 23-16: LBA Mid (Cylinder Low) Register
			Bits 15-8: LBA Low (Sector Number) Register
			Bits 7-0: Sector Count Register

20.4.27 Port Serial ATA Status Register (P#SSTS) (# = 0 or 1)

The port serial ATA status register (P#SSTS) conveys the current state of the interface and host. The Port updates it continuously and asynchronously. When the Port transmits a COMRESET to the device, this register is updated to its reset values (Global reset, Port reset, or COMINIT from the device).

The P#SSTS register is shown in Figure 20-28 and described in Table 20-30.

Figure 20-28. Port Serial ATA Status Register (P#SSTS)

31	16
Reserved	
R-0	

15	12	11	8	7	4	3	0
Reserved		IPM		SPD		DET	
R-0		R-0		R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 20-30. Port Serial ATA Status Register (P#SSTS) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved.
11-8	IPM	0-Fh	Interface Power Management. Indicates the current interface state.
		0	Device is not present or communication is not established.
		1h	Interface in active state
		2h	Interface in Partial power management state.
		3h-5h	Reserved
		6h	Interface in Slumber power management state.
		7h-Fh	Reserved
7-4	SPD	0-Fh	Current Interface Speed. Indicates the negotiated interface communication speed.
		0	Device is not present or communication is not established.
		1h	Generation 1 (1.5 Gbps) negotiated
		2h	Generation 2 (3 Gbps) negotiated
		3h-Fh	Reserved
3-0	DET	0-Fh	Device Detection. Indicates the interface device detection and PHY state.
		0	No device detected and PHY communication is not established.
		1h	Device presence detected but PHY communication is not established (COMINIT is detected).
		2h	Reserved
		3h	Device presence detected and PHY communication is established (PHY Ready is detected).
		4h	PHY in offline mode as a result of the interface being disabled or running in BIST loopback mode.
		5h-Fh	Reserved

20.4.28 Port Serial ATA Control Register (P#SCTL) (# = 0 or 1)

The port serial ATA control register (P#SCTL) is used by software to control SATA interface capabilities. Writes to this register result in an action being taken by the Port PHY interface. Reads from the register return the last value written to it. Reset on Global reset.

These bits are static and should not be changed frequently due to the clock crossing between the Transport and Link Layers. Software must wait for at least seven periods of the slower clock (clk_asic or OCP clock) before changing this register.

The P#SCTL register is shown in Figure 20-29 and described in Table 20-31.

Figure 20-29. Port Serial ATA Control Register (P#SCTL)

31	16
Reserved	
R-0	

15	12	11	8	7	4	3	0
Reserved		IPM		SPD		DET	
R-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 20-31. Port Serial ATA Control Register (P#SCTL) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved.
11-8	IPM	0-Fh	Interface Power Management Transitions Allowed. Indicates which power states the Port PHY interface is allowed to transition to. If an interface power management state is disabled, the Port does not initiate that state and any request from the device to enter that state is rejected via PMNAKp.
		0	No interface power management state restrictions.
		1h	Transitions to the Partial state are disabled.
		2h	Transitions to the Slumber state are disabled.
		3h	Transitions to both Partial and Slumber states are disabled.
		4h-Fh	Reserved
7-4	SPD	0-Fh	Speed Allowed. Indicates the highest allowable speed of the Port PHY interface. Note: When host software must change this bit field value, the host must also reset the Port (DET = 1) at the same time to ensure proper speed negotiation.
		0	No speed negotiation restrictions.
		1h	Limit speed negotiation to Generation 1 (1.5 Gbps) communication rate.
		2h	Limit speed negotiation to Generation 2 (3 Gbps) communication rate.
		3h-Fh	Reserved
3-0	DET	0-Fh	Device Detection Initialization. Controls the Ports device detection and interface initialization. Note: This bit field may only be modified when P0CMD.ST = 0; changing this bit field while P0CMD.ST = 1 results in undefined behavior. When P0CMD.ST is set to 1, this bit field should have a value of 0.
		0	No device detection or initialization action is requested.
		1h	Perform interface initialization sequence to establish communication. This results in the interface being reset and communication re-initialized.
		2h-3h	Reserved
		4h	Disable the Serial ATA interface and put the Port PHY in offline mode.
		5h-Fh	Reserved

20.4.29 Port Serial ATA Error Register (P#SERR) (# = 0 or 1)

The port serial ATA error register (P#SERR) represents all the detected interface errors accumulated since the last time it was cleared. The set bits in the P#SERR register indicate that the corresponding error condition became true one or more times since the last time the bit was cleared. The set bits in this register are explicitly cleared by a write operation to the register, Global reset, or Port reset (COMRESET). The value written to clear the set error bits should have ones encoded in the bit positions corresponding to the bits that are to be cleared. All bits in the following table have a reset value of 0.

The P#SERR register is shown in Figure 20-30 and described in Table 20-32.

Figure 20-30. Port Serial ATA Error Register (P#SERR)

31					27	26	25	24
Reserved						DIAG_X	DIAG_F	DIAG_T
R-0						R/W1C-0	R/W1C-0	R/W1C-0
23	22	21	20	19	18	17	16	
DIAG_S	DIAG_H	DIAG_C	DIAG_D	DIAG_B	DIAG_W	DIAG_I	DIAG_N	
R/W1C-0	R/W1C-0	R/W1C-0	R-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	
15			12	11	10	9	8	
Reserved				ERR_E	ERR_P	ERR_C	ERR_T	
R-0				R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	
7					2	1	0	
Reserved						ERR_M	ERR_I	
R-0						R/W1C-0	R/W1C-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 20-32. Port Serial ATA Error Register (P#SERR) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved.
26	DIAG_X	0-1	Exchanged. This bit is set to 1 when PHY COMINIT signal is detected. This bit is reflected in the P0IS.PCS bit.
25	DIAG_F	0-1	Unknown FIS Type. Indicates that one or more FISes were received by the Transport layer with good CRC, but had a type field that was not recognized/known and the length was less than or equal to 64 bytes. Note: If the Unknown FIS length exceeds 64 bytes, the DIAG_F bit is not set and the DIAG_T bit is set instead.
24	DIAG_T	0-1	Transport State Transition Error. Indicates that a Transport Layer protocol violation was detected.
23	DIAG_S	0-1	Link Sequence Error. Indicates that one or more Link state machine error conditions was encountered. One of the conditions that cause this bit to be set is the device doing SYNC escape during FIS transmission.
22	DIAG_H	0-1	Handshake Error. Indicates that one or more R-ERRp was received in response to frame transmission. Such errors may be the result of a CRC error detected by the device, a disparity or 8-bit/10-bit decoding error, or other error condition leading to a negative handshake on a transmitted frame.
21	DIAG_C	0-1	CRC Error. Indicates that one ore more CRC errors were detected by the Link layer during FIS reception.
20	DIAG_D	0	Disparity Error. This bit is always 0 since it is not used by the AHCI specification.
19	DIAG_B	0-1	10B to 8B Decode Error. Indicates errors were detected by 10b8b decoder. Note: This bit is set only when an error is detected on the received FIS data word. This bit is not set when an error is detected on the primitive, regardless of whether it is inside or outside the FIS.
18	DIAG_W	0-1	Comm Wake. This bit is set when the PHY COMWAKE signal is detected.
17	DIAG_I	0-1	PHY Internal Error. This bit is set when the PHY detects some internal error. Note: The TI PHY does not support any errors so this bit will never be set.
16	DIAG_N	0-1	PHY Ready Change. Indicates that the PHY Ready signal changed state. This bit is reflected in the P0IS.PRCS bit.
15-12	Reserved	0	Reserved.

Table 20-32. Port Serial ATA Error Register (P#SERR) Field Descriptions (continued)

Bit	Field	Value	Description
11	ERR_E	0-1	Internal Error. This bit is set to 1 when one or more AHB bus ERROR responses are detected on the master or the slave interfaces.
10	ERR_P	0-1	Protocol Error. This bit is set to 1 when any of the following conditions are detected: • Transport state transition error (DIAG_T) • Link sequence error (DIAG_S) • RxFIFO overflow • Link bad end error (WTRM instead of EOF is received)
9	ERR_C	0-1	Non-recovered Persistent Communication Error. This bit is set to 1 when the PHY Ready signal is negated due to the loss of communication with the device or problems with the interface, but not after transition from active to Partial or Slumber power management state.
8	ERR_T	0-1	Non-recovered Transient Data Integrity Error. This bit is set if any of the following P0SERR register bits is set during Data FIS transfer: • ERR_P (Protocol) • DIAG_C (CRC) • DIAG_H (Handshake) • ERR_C (PHY Ready negation)
7-2	Reserved	0	Reserved.
1	ERR_M	0-1	Recovered Communication Error. This bit is set to 1 when the PHY Ready condition is detected after interface initialization, but not after transition from partial or Slumber power management state to active state.
0	ERR_I	0-1	Recovered Data Integrity. This bit is set if any of the following P0SERR register bits is set during non-Data FIS transfer: • ERR_P (Protocol) • DIAG_C (CRC) • DIAG_H (Handshake) • ERR_C (PHY Ready negation)

20.4.30 Port Serial ATA Active Register (P#SACT) (# = 0 or 1)

The port serial ATA active register (P#SACT) is used to indicate what command slots have commands in them.

The P#SACT register is shown in Figure 20-31 and described in Table 20-33.

Figure 20-31. Port Serial ATA Active Register (P#SACT)

31	0
DS[*n*]	
R/W1S-0	

LEGEND: R/W = Read/Write; W1S = Write 1 to set; -*n* = value after reset

Table 20-33. Port Serial ATA Active Register (P#SACT) Field Description

Bit	Field	Value	Description
31-0	DS[*n*]	0-1	This field is bit significant. Each bit *n* corresponds to the TAG and command slot of a native queued command, where bit 0 corresponds to TAG 0 and command slot 0. This field is set by software prior to issuing a native queued command for a particular command slot. Prior to writing P0CI[TAG] to 1, software will set DS[TAG] to 1 to indicate that a command with that TAG is outstanding. The device clears bits in this field by sending a Set Device Bits FIS to the Port. The Port clears bits in this field that are set to 1 in the SActive field of the Set Device Bits FIS. The Port only clears bits that correspond to native queued commands that have completed successfully. Software should only write to this bit field when P0CMD.ST bit is set to 1. This bit field is cleared when P0CMD.ST is written from a 1 to a 0 by software. This bit field is not cleared by a Port reset (COMRESET) or a software reset.

20.4.31 Port Command Issue Register (P#CI) (# = 0 or 1)

The port command issue register (P#CI) is used to indicate what that a command has been constructed and may be carried out.

The P#CI register is shown in Figure 20-32 and described in Table 20-34.

Figure 20-32. Port Command Issue Register (P#CI)

31	0
CI[*n*]	
R/W1S-0	

LEGEND: R/W = Read/Write; W1S = Write 1 to set; -*n* = value after reset

Table 20-34. Port Command Issue Register (P#CI) Field Description

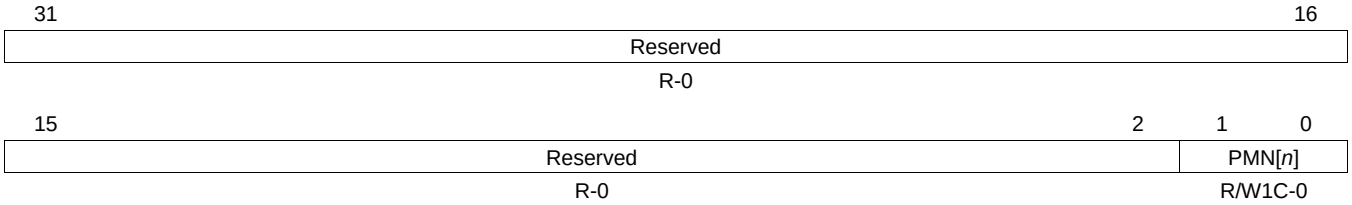
Bit	Field	Value	Description
31-0	CI[*n*]	0-1	Command Issued. This field is bit significant. Each bit *n* corresponds to a command slot, where bit 0 corresponds to command slot 0. This bit field is set by software to indicate to the Port that a command has been built in system memory for a command slot and may be sent to the device. When the Port receives a FIS that clears the BSY, DRQ, and ERR bits for the command, it clears the corresponding bit *n* in this register for that command slot. Bits in this field can only be set to 1 by software when P0CMD.ST is set to 1.

20.4.32 Port Serial ATA Notification Register (P#SNTF) (# = 0 or 1)

The port serial ATA notification register (P#SNTF) is used to determine if asynchronous notification events have occurred for directly connected devices and devices connected to a Port Multiplier.

The P#SNTF register is shown in Figure 20-33 and described in Table 20-35.

Figure 20-33. Port Serial ATA Notification Register (P#SNTF)



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 20-35. Port Serial ATA Notification Register (P#SNTF) Field Description

Bit	Field	Value	Description
31-2	Reserved	0	Reserved.
1-0	PMN[*n*]	0-1	PM Notify. Indicates whether a particular device with the corresponding PM Port number issued a Set Device Bits FIS to the SATASS Port with the Notification bit set. Individual bits are cleared by software writing 1s to the corresponding bit *n* positions. This bit field is reset on Global reset, but it is not reset by Port reset (COMRESET) or software reset. • PM Port 0 sets bit 0 • PM Port 1 sets bit 1 Note that two different types of ports exist. HBA Port (the device has the support for 2 HBA Ports) and Port Multiplier Port (PMP). A Port of a Port Multiplier (PMP) connects a single device to a PM. As many as 15 devices can be connected to a single HBA Port.

20.4.33 Port DMA Control Register (P#DMACR) (# = 0 or 1)

The port DMA control register (P#DMACR) contains bits for controlling the Port DMA engine. The software can change the fields of this register only when P0CMD.ST=0. Power-up (system reset), Global reset, or Port reset (COMRESET) reset this register to the default value.

The P#DMACR register is shown in Figure 20-34 and described in Table 20-36.

Figure 20-34. Port DMA Control Register (P#DMACR)

31	16
Reserved	
R-0	

15 12	11 8	7 4	3 0
RXABL	TXABL	RXTS	TXTS
R-0	R/W-0	R/W-4h	R/W-4h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 20-36. Port DMA Control Register (P#DMACR) Field Description

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-12	RXABL	0-Fh	Receive Burst Limit. Allows software to limit the OCP master write burst size.
			Note: SATASS master breaks the burst at 1Kbyte address boundaries regardless of the RXABL value.
		0	Limit OCP burst size to 256 DWORDS
		1h	Limit OCP Burst size to 1 DWORD
		2h	Limit OCP Burst size to 2 DWORDs
		3h	Limit OCP Burst size to 4 DWORDs
		4h	Limit OCP Burst size to 8 DWORDs
		5h	Limit OCP Burst size to 16 DWORDs
		6h	Limit OCP Burst size to 32 DWORDs
		7h	Limit OCP Burst size to 64 DWORDs
		9h-Fh	Limit OCP Burst size to 256 DWORDS
			Note: The maximum sized burst that will be issued on the OCP is 64 bytes; for this reason higher programmed values than 0x7 will still result with 64 bytes burst.
			Note: This field is read-write when P0CMD.ST=0 and read-only when P0CMD.ST=1.
11-8	TXABL	0-Fh	Transmit Burst Limit. This field allows software to limit the OCP master read burst size. This bit field is read/write when P0CMD.ST = 0 and read-only when P0CMD.ST = 1.
			Note: SATASS master breaks the burst at 1Kbyte address boundaries regardless of the TXABL value.
		0	Limit OCP burst size to 256 DWORDS
		1h	Limit OCP Burst size to 1 DWORD
		2h	Limit OCP Burst size to 2 DWORDs
		3h	Limit OCP Burst size to 4 DWORDs
		4h	Limit OCP Burst size to 8 DWORDs
		5h	Limit OCP Burst size to 16 DWORDs
		6h	Limit OCP Burst size to 32 DWORDs
		7h	Limit OCP Burst size to 64 DWORDs
		9h-Fh	Limit OCP Burst size to 256 DWORDs
			Note: The maximum sized burst that will be issued on the OCP is 64 bytes; for this reason higher programmed values than 0x7 will still result with 64 bytes burst.

Table 20-36. Port DMA Control Register (P#DMACR) Field Description (continued)

Bit	Field	Value	Description
7-4	RXTS	0-Fh	Receive Transaction Size (RX_TRANSACTION_SIZE). This field defines the Port DMA transaction size in DWORDs for receive (system bus write, device read) operation. This bit field is read/write when P0CMD.ST = 0 and read-only when P0CMD.ST = 1. The maximum value of this bit field is determined by the Rx FIFO Depth (P0_RXFIFO_DEPTH). If software attempts to write a value exceeding this maximum value, the maximum value would be set instead.
		0	1 DWORD
		1h	2 DWORDs
		2h	4 DWORDs
		3h	8 DWORDs
		4h	16 DWORDS
		5h	32 DWORDs
		6h	64 DWORDs
		7h-Fh	Reserved
3-0	TXTS	0-Fh	Transmit Transaction Size (TX_TRANSACTION_SIZE). This field defines the DMA transaction size in DWORDs for transmit (system bus read, device write) operation. This bit field is read/write when P0CMD.ST = 0 and read-only when P0CMD.ST = 1. The maximum value of this bit field is determined by the Tx FIFO Depth (P0_TXFIFO_DEPTH). If software attempts to write a value exceeding this maximum value, the maximum value would be set instead.
		0	1 DWORD
		1h	2 DWORDs
		2h	4 DWORDs
		3h	8 DWORDs
		4h	16 DWORDS
		5h	32 DWORDs
		6h-Fh	Reserved

20.4.34 Port PHY Control Register (P#PHYCR) (# = 0 or 1)

The port PHY control register (P#PHYCR) is used to control the PHY. These ports are configured for the WIZ6C2B2N5W0M (Maverick B2) macro. Refer to that spec for further details.

Note: This description is only valid for configuration GS60. See the corresponding P0PHYSR Register description for each configuration.

The P#PHYCR register is shown in Figure 20-35 and described in Table 20-37.

Figure 20-35. Port PHY Control Register (P#PHYCR)

31							27	26						24
TXDE								TXSWING						
R/W-0								R/W-0						
23	22	21	20	19										16
TXSWING	TXCM	TXINVPAIR	RXENOC	RXEQ										
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0										
15			13	12	11			10			9			8
RXCDR				RXLOS	LOOPBACK						RXINVPAIR			CLKBYP
R/W-0				R/W-0	R/W-0						R/W-0			R/W-0
7	6		5	4									1	0
CLKBYP	LB			MPY										ENPLL
R/W-0	R/W-0			R/W-0										R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 20-37. Port PHY Control Register (P#PHYCR) Field Descriptions

Bit	Field	Value	Description	
31-27	TXDE	0-Fh	Transmitter De-Emphasis. Selects 1 of 16 output de-emphasis settings from 0 to 71.42%	
			Amplitude Reduction	
			%	**dB**
		0	0	0
		1	4.76	-0.42
		2h	9.52	-0.87
		3h	14.28	-1.34
		4h	19.04	-1.83
		5h	23.8	-2.36
		6h	28.56	-2.92
		7h	33.32	-3.52
		8h	38.08	-4.16
		9h	42.85	-4.86
		Ah	47.61	-5.61
		Bh	52.38	-6.44
		Ch	57.14	-7.35
		Dh	61.9	-8.38
		Eh	66.66	-9.54
		Fh	71.42	-10.87

Table 20-37. Port PHY Control Register (P#PHYCR) Field Descriptions (continued)

Bit	Field	Value	Description
26-23	TXSWING		Transmitter Output Swing. See SERDES specification for complete description of values.
			DC-Coupled Amplitude / **AC-Coupled Amplitude**
		0	100 / tbd
		1h	tbd / tbd
		2h	tbd / tbd
		3h	tbd / tbd
		4h	tbd / tbd
		5h	tbd / tbd
		6h	tbd / tbd
		7h	tbd / tbd
		8h	tbd / tbd
		9h	tbd / tbd
		Ah	tbd / tbd
		Bh	tbd / tbd
		Ch	tbd / tbd
		Dh	tbd / tbd
		Eh	tbd / tbd
		Fh	1000 / tbd
22	TXCM		Transmitter Common Mode.
		0	Normal Common Mode
		1	Raised common mode.
21	TXINVPAIR	0	Transmitter Invert Polarity. Inverts the polarity of TXP# and TXN#
20	RXENOC		Receiver Offset Compensation.
		0	Compensation Disabled.
		1	Compensation Enabled.
19-16	RXEQ	0-Fh	Receiver Equalizer Configuration.
			Low Frequency Gain / **Zero Frequency**
		0	Maximum / —
		1h	Adaptive
		2h-7h	Reserved
		8h	Adaptive / 365 MHz
		9h	Adaptive / 275 MHz
		Ah	Adaptive / 195 MHz
		Bh	Adaptive / 140 MHz
		Ch	Adaptive / 105 MHz
		Dh	Adaptive / 75 MHz
		Eh	Adaptive / 55 MHz
		Fh	Adaptive / 50 MHz

Table 20-37. Port PHY Control Register (P#PHYCR) Field Descriptions (continued)

Bit	Field	Value	Description
15-13	RXCDR	0-7h	Receiver Clock/data Recovery. Configures the clock/data recovery algorithm.
		0	First order, threshold of 1. Phase offset tracking up to +/-488ppm. Suitable for use in asynchronous systems with low frequency offset.
		1h	First order, threshold of 17. Phase offset tracking up to +/-325ppm. Suitable for use in synchronous systems. Offers superior rejection of random jitter, but is less responsive to systematic variation such as sinusoidal jitter.
		2h	Second order, high precision, threshold of 1. Highest precision frequency offset matching but relatively poor response to changes in frequency offset, and long lock time. Suitable for use in systems with fixed frequency offset.
		3h	Second order, high precision, threshold of 1. Highest precision frequency offset matching but relatively poor response to changes in frequency offset, and long lock time. Suitable for use in systems with fixed frequency offset.
		4h	Second order, high precision, threshold of 1. Highest precision frequency offset matching but relatively poor response to changes in frequency offset, and long lock time. Suitable for use in systems with fixed frequency offset.
		5h	Second order, high precision, threshold of 1. Highest precision frequency offset matching but relatively poor response to changes in frequency offset, and long lock time. Suitable for use in systems with fixed frequency offset.
		6h	Reserved.
		7h	Reserved.
12	RXLOS		Loss of Signal Detection.
			Note: This must be enabled by software before initialization is started on the device.
			Note: Refer to PHY CFGR2 Register for what value is actually programmed into the SERDES. In most cases, only the default will have to be used.
		0	Disabled.
		1	Enabled.
11-10	LOOPBACK		Transmitter and Receiver Loopback Selection.
		0	Transmitter – Disabled
			Receiver – Disabled
		1h	Transmitter – Reserved
			Receiver – Reserved
		2h	Transmitter – Loopback, TX Driver Disabled. The loopback path covers all the stages of the transmitter except the TX output itself. A differential current is passed to the receiver. The magnitude of this current is dependent on SWING. The transmitter driver itself is disabled.
			Receiver – Reserved (N/A)
		3h	Transmitter – Loopback, TX Driver Enabled. As above, but the transmit driver operates normally.
			Receiver – Loopback. The differential current from the transmitter is converted to a voltage and applied to the receiver input if LOS is disabled, or to the Loss of Signal detector if LOS is enabled.
9	RXINVPAIR	0	Receiver Invert Polarity. Inverts the polarity of RXP# and RXN#.
8-7	CLKBYP		Clock Bypass. Facilitates bypassing of the PLL with either REFCLKP/N or TESTCLKT, and bypassing of the recovered receiver clock with TESTCLKR.
			Note: This field applies to Port 0 only, P0PHYCR. This bit is reserved for Port 1, P1PHYCR.
		0	No bypass. Macro operates normally from the PLL.
		1h	Reserved.
		2h	Functional Bypass. The macro operates functionally at low speed using TESTCLKT and TESTCLKR.
		3h	REFCLK Observe. The PLL is bypassed by REFCLKP/N. This diagnostic capability is designed to facilitate observation of the reference clock from macros containing transmitters

Table 20-37. Port PHY Control Register (P#PHYCR) Field Descriptions (continued)

Bit	Field	Value	Description
6-5	LB		Loopback. Enables Loopback.
			Note: This field applies to Port 0 only, P0PHYCR. This bit is reserved for Port 1, P1PHYCR.
		0	Disabled.
		1h	Reserved.
		2h	Inner loopback, CML driver disabled. The loopback path is fully digital and excludes the transmit CML driver and receive sense amps. The transmit CML driver is disabled.
		3h	Inner loopback, CML driver enabled. As above, but the CML driver operates normally.
4-1	MPY	0-Fh	PLL Multiply. Select PLL multiply factors between 5 and 20.
			The MPY factor along with the SERDES_CLKP/N frequency is what determines the required data rate. Full Rate corresponds to 3Gbps Line Rate (Gen2) and Half Rate corresponds to 1.5Gbps Line Rate (Gen1).
			Example: If SERDES_CLKP/N is300MHz, then MPY=4'b0001 (5x).
			Note: This field should only be written before or at the same time while enabling the ENPLL field. You should not change MPY field after the PLL is enabled.
			Note: This field applies to Port 0 only, P0PHYCR. This field is reserved for Port 1, P1PHYCR.
		0	4x
		1h	5x
		2h	6x
		3h	8x
		4h	8.25x
		5h	10x
		6h	12x
		7h	12.5x
		8h	15x
		9h	16x
		Ah	16.5x
		Bh	20x
		Ch	22x
		Dh	25x
		Eh	Reserved.
		Fh	Reserved.
0	ENPLL		Enable PHY PLL. This bit must be written to 1 to enable the PHY's PLL and use the link.
			Note: This must be enabled by software before initialization is started on the device.
			Note: This bit should not be enabled unless the MPY field has been previously set or is being set in the same write. Changing the MPY field while the PLL is enabled disrupts the link clocks and may disrupt the link, requiring a port reset to fix.
			Note: This field applies to Port 0 only, P0PHYCR. This bit is reserved for Port 1, P1PHYCR.
		0	Disabled.
		1	Enabled.

20.4.35 Port PHY Status Register (P#PHYSR) (# = 0 or 1)

The port PHY status register (P#PHYSR) is used to reflect the PHY status. Note: In multi-port configurations, each of these registers will read identically, P0PHYSR and P1PHYSR have the same value.

The P#PHYSR register is shown in Figure 20-36 and described in Table 20-38.

Figure 20-36. Port PHY Status Register (P#PHYSR)

31	16
Reserved	
R-0	

15	1	0
Reserved	SIGDET	LOCK
R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 20-38. Port PHY Status Register (P#PHYSR) Field Description

Bit	Field	Value	Description
31-2	Reserved	0	Reserved.
1	SIGDET	0-1	Signal Detect. Indicates that Port # has a signal present.
0	LOCK	0-1	PLL Lock. Indicates that the PLL has locked.

20.4.36 Idle Register (IDLE)

The Idle Register (IDLE) is used to control the idle and standby modes.

The Idle Register (IDLE) is shown in Figure 20-37 and described in Table 20-39.

Figure 20-37. Idle Register (IDLE)

31	18	17	16
Reserved		OVERRIDE1	OVERRIDE0
R-0		R/W-0	R/W-0

15	4	3	2	1	0
Reserved		STANDBYMODE		IDLEMODE	
R-0		R/W-0		R/W-0	

LEGEND: R = Read only; -*n* = value after reset

Table 20-39. Idle Register (IDLE) Field Description

Bit	Field	Value	Description
31-18	Reserved	0	Reserved.
17	OVERRIDE1		Override for Clock Stopping. Normally the functional clock can only be stopped if the link is put into partial or slumber power mode. However, if there is no device attached (such as in a removable media situation) or the device has not been started the user may wish to stop the functional clocks but not be able to enter a low power state. In this case, software can set the OVERRIDE bit to 1, removing the requirement for a low power state. **WARNING:** If there is a device attached, the OVERRIDE bit is used, and the functional clock is stopped when the link is not in a low power state it will ruin the link and cause undetermined behavior. A port reset or full SATASS reset may be required to recover.
		0	Normal.
		1	Override.
16	OVERRIDE0		Override for Clock Stopping. Normally the functional clock can only be stopped if the link is put into partial or slumber power mode. However, if there is no device attached (such as in a removable media situation) or the device has not been started the user may wish to stop the functional clocks but not be able to enter a low power state. In this case, software can set the OVERRIDE bit to 1, removing the requirement for a low power state. **WARNING:** If there is a device attached, the OVERRIDE bit is used, and the functional clock is stopped when the link is not in a low power state it will ruin the link and cause undetermined behavior. A port reset or full SATASS reset may be required to recover.
		0	Normal.
		1	Override.
15-4	Reserved	0	Reserved.
3-2	STANDBYMODE		This gives direct control to the standby_mode control of the standby generic.
1-0	IDLEMODE		This gives direct control to the idle_mode control of the idle generic. **Note:** Only force idle mode is supported.

20.4.37 PHY Configuration Register 2 (PHYCFGR2)

The Idle Register (IDLE) is used to control the idle and standby modes.

The Idle Register (IDLE) is shown in Figure 20-38 and described in Table 20-40.

Figure 20-38. PHY Configuration Register 2 (PHYCFGR2)

31	16
Reserved	
R-0	

15	10	9	8	7	6	5	3	2	0
Reserved		P1ALIGN		P0ALIGN		P1LOS		P0LOS	
R-0		R/W-1		R/W-1		R/W-4		R/W-4	

LEGEND: R = Read only; -*n* = value after reset

Table 20-40. PHY Configuration Register 2 (PHYCFGR2) Field Description

Bit	Field	Value	Description
31-10	Reserved	0	Reserved.
9-8	P1ALIGN		Port 1 ALIGN –this register is used to directly control the ALIGN (comma alignment enable) on the SERDES. Comma alignment defaults to 0x1 (ON) and in most cases will never have to be changed. However, if a system is experiencing a large amount of incorrect commas detected, software may use this register to disable comma alignment after a link has been negotiated.
		0	Alignment Disabled.
		1h	Comma Alignment Enabled.
		2h	Alignment Jog.
		3h	Reserved.
7-6	P0ALIGN		Port 0 ALIGN – this register is used to directly control the ALIGN (comma alignment enable) on the SERDES. Comma alignment defaults to 0x1 (ON) and in most cases will never have to be changed. However, if a system is experiencing a large amount of incorrect commas detected, software may use this register to disable comma alignment after a link has been negotiated.
		0	Alignment Disabled.
		1h	Comma Alignment Enabled.
		2h	Alignment Jog.
		3h	Reserved.
5-3	P1LOS		Port 1 LOS –This register is used to directly control the Loss-of-Signal configuration for the SERDES macro on port 1. When the LOS detection is enabled in the **Error! Reference source not found.** based on the LOS bit, this is the value that will be fed directly into the SERDES. Usually, only the default will need to be used. However, if silicon variations cause the LOS detection threshold to need to be adjusted, this register can be used to do so.
2-0	P0LOS		Port 0 LOS –This register is used to directly control the Loss-of-Signal configuration for the SERDES macro on port 1. When the LOS detection is enabled in the **Error! Reference source not found.** based on the LOS bit, this is the value that will be fed directly into the SERDES. Usually, only the default will need to be used. However, if silicon variations cause the LOS detection threshold to need to be adjusted, this register can be used to do so.

Timers

This chapter describes the operation of the software-programmable timer.

Topic **Page**

21.1 Introduction

21.1.1 Overview

The timer module contains a free running upward counter with auto reload capability on overflow. The timer counter can be read and written in real-time (while counting). The timer module includes compare logic to allow an interrupt event on a programmable counter matching value.

A dedicated output signal can be pulsed or toggled on overflow and a matched event. This output can be used as a timing stamp trigger signal or PWM (pulse-width modulation) signal source. An additional dedicated output signal (PORGPOCFG) can be used for general-purpose IO, directly driven by bit 14 of the Timer Control Register (TCLR). Based on the programmable input signal transition type, a dedicated signal can be used to trigger an automatic timer counter capture event and generate an interrupt. A programmable clock divider (prescaler) allows reduction of the timer input clock frequency. All internal timer interrupt sources are merged in one module interrupt line and one wake-up line. Each internal interrupt source can be independently enabled or disabled.

This module is controllable through the OCP peripheral bus.

As two clock domains are managed inside this module, resynchronization is done by special logic between the OCP clock domain and the Timer clock domain. At reset, synchronization logic allows utilization of all ratios between the OCP clock and the Timer clock. A drawback of this mode is that the full-resynchronization path is used with access latency performance impact in terms of OCP clock cycles. In order to improve module access latency, and under restricted conditions on clock ratios, write-posted mode can be used by setting the POSTED bit of the Timer Synchronous Interface Control Register (TSICR). Under posted mode, the OCP write command is granted before the write process is complete in the timer clock domain. This mode allows software to do concurrent writes on Dual Mode timer registers and to observe write process completion (synchronization) at the software level by reading independent write posted status bits in the Write Posted Status Register (TWPS).

21.1.2 Features

The timer consists of the following features:

- Counter timer with compare and capture modes
- Auto-reload mode
- Start-stop mode
- Programmable divider clock source
- 16-32 bit addressing
- “On the fly” read/write registers
- Interrupts generated on overflow, compare and capture
- Interrupt enable
- Wake-up enable
- Write posted mode
- Dedicated input trigger for capture mode and dedicated output trigger/PWM signal
- Dedicated output signal for general purpose use PORGPOCFG
- OCP interface compatible

The Timer resolution and interrupt period are dependent on the selected input clock and clock prescale value. Example resolutions for common clock values are shown in Table 21-1.

Table 21-1. Timer Resolution and Maximum Range

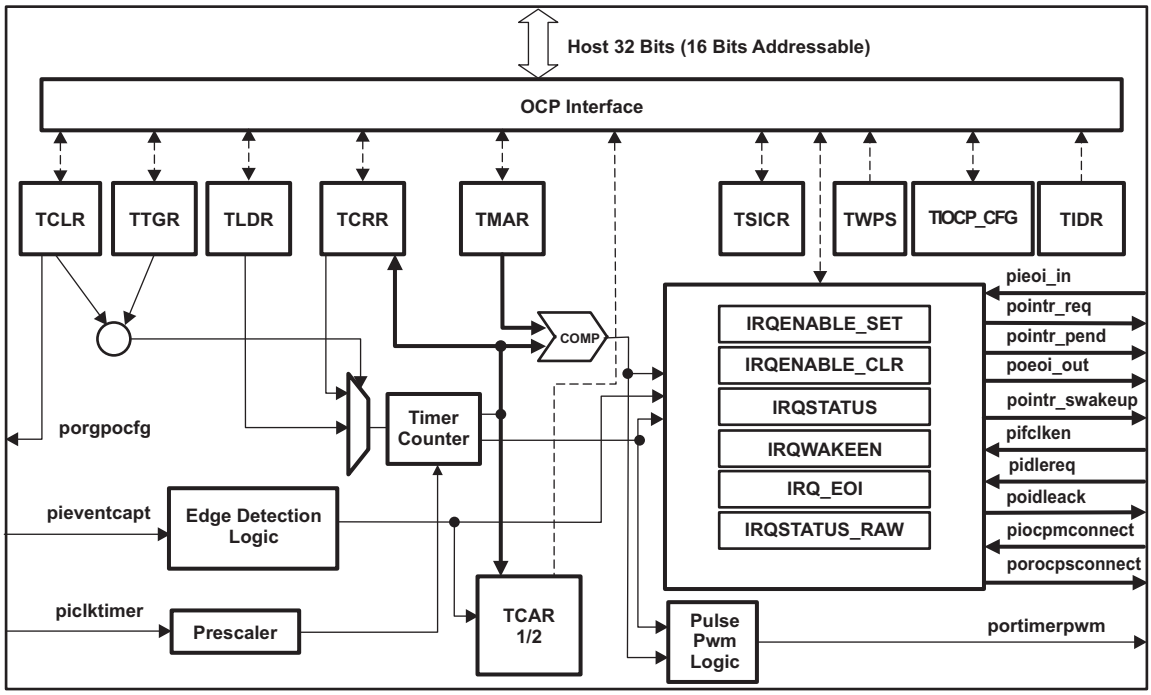
Clock	Prescaler	Resolution	Interrupt Period Range[1]
32 KHz	1 (min)	31.25 us	62.5 us to ~37h 17m
	256 (max)	8 ms	16 ms to ~397d 16h 22m
27 MHz	1 (min)	~37 ns	~74 ns to ~159s
	256 (max)	~9.48 us	~18.9 us to ~11h 18m
38.4 MHz	1 (min)	~26 ns	~52 ns to ~112 s
	256 (max)	~6.7 us	~13.3 us to ~7h 57m

(1) Minimum calculation adheres to recommendation of 2 clock cycle minimum between interrupts.

21.1.3 Functional Block Diagram

Figure 21-1 shows a block diagram of the timer.

Figure 21-1. Timer Block Diagram

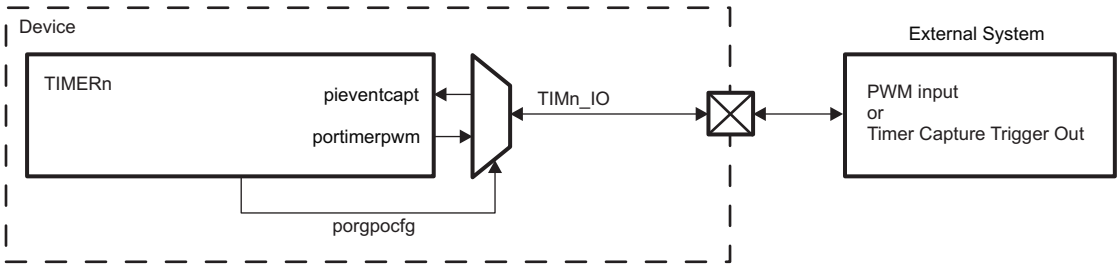


21.1.4 GP Timer External System Interface

The general-purpose (GP) timer can send or receive stimulus to/from the external (off-chip) system. In the device, however, only certain timers are configured to output a PWM pulse or receive an external event signal used as a trigger to capture the current timer count. See the device-specific data manual details on which timers have IO capabilities.

Figure 21-2 shows the external system interface for the GP timers.

Figure 21-2. GP Timers External System Interface



NOTE: For the timers with IO capabilities, the PORGPOCFG output signal selects the timer PWM output or the timer capture input signal to the TIMn_IO pad at the top level. When TCLR[GPO_CFG] = 1, TIMn_IO functions as a capture input. When TCLR[GPO_CFG] = 0, TIMn_IO functions as a PWM output.

21.2 Architecture

21.2.1 Functional Description

The general-purpose timer is an upward counter. It supports 3 functional modes:

- Timer mode
- Capture mode
- Compare mode

By default, after core reset, the capture and compare modes are disabled.

21.2.1.1 Timer Mode Functionality

The timer is an upward counter that can be started and stopped at any time through the ST bit of the Timer Control Register (TCLR). The Timer Counter Register (TCRR) can be loaded when stopped or on the fly (while counting). TCRR can be loaded directly by a TCRR write access with the new timer value. TCRR can also be loaded with the value held in the Timer Load Register (TLDR), due to a write access to the Timer Trigger Register (TTGR). TCRR is loaded regardless of the TTGR written value. The TCRR value can be read when stopped or captured on-the-fly by a TCRR read access. The timer is stopped and the counter value is cleared to “0” when the module’s reset is asserted. The timer is maintained in a stopped state after reset is released. When the timer is stopped, TCRR does not increment. The counter can be restarted from the frozen value unless TCRR has been reloaded with a new value.

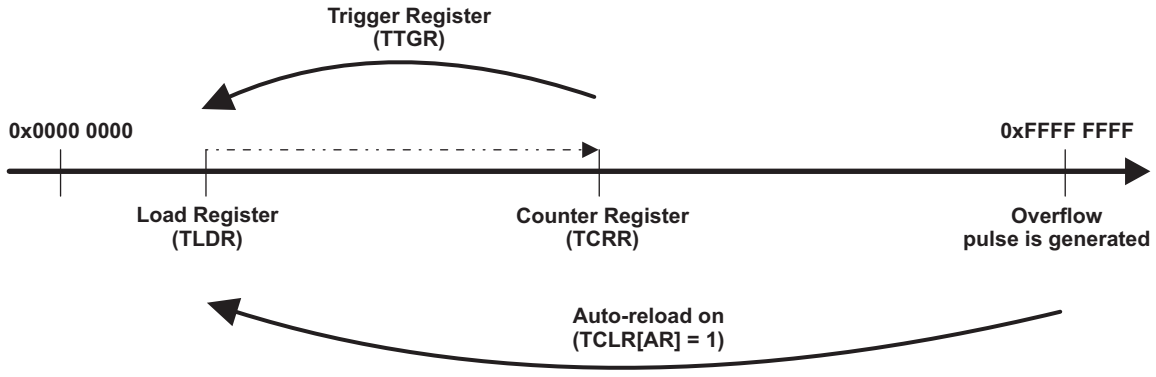
When the one-shot mode is enabled (TCLR[AR] = 0), the counter is stopped after a counting overflow (counter value remains at zero).

When the auto-reload mode is enabled (TCLR[AR] = 1), TCRR is reloaded with the Timer Load Register (TLDR) value after a counting overflow.

It is not recommended to put the overflow value (FFFF FFFFh) in TLDR because it can lead to undesired results.

An interrupt can be issued on overflow, if the overflow interrupt enable bit is set in the Timer IRQENABLE Set Register (IRQENABLE_SET[OVF_EN_FLAG] = 1). In addition, a dedicated output pin can be used to generate one positive pulse (piclktimer duration) or to invert the current value (toggle mode) when an overflow occurs. Refer to Section 21.2.1.5 for more details related to the dedicated pulse-width modulation (PWM) output pin.

Figure 21-3. TCRR Timing Value



21.2.1.2 Capture Mode Functionality

The timer value in TCRR can be captured and saved in TCAR1 or TCAR2 function of the mode selected in TCLR through the field CAPT_MODE when a transition is detected on the module input pin (PIEVENTCAPT). The edge detection circuitry monitors transitions on the input pin (PIEVENTCAPT).

Rising transition, falling transition or both can be selected in TCLR (TCM bit) to trig the timer counter capture. The module sets the IRQSTATUS (TCAR_IT_FLAG bit) when an active transition is detected and at the same time the counter value TCRR is stored in one of the timer capture registers TCAR1 or TCAR2 as follows:

- If TCLR's CAPT_MODE field is 0 then, on the first enabled capture event, the value of the counter register is saved in TCAR1 register and all the next events are ignored (no update on TCAR1 and no interrupt triggering) until the detection logic is reset or the interrupt status register is cleared on TCAR's position writing a 1 in it.
- If TCLR's CAPT_MODE field is 1 then, on the first enabled captured event, the counter value is saved in TCAR1 register and, on the second enabled capture event, the value of the counter register is saved in TCAR2 register. All the other events are ignored (no update on TCAR1/2 and no interrupt triggering) until the detection logic is reset or the interrupt status register is cleared on TCAR's position writing a 1 in it. This mechanism is useful for period calculation of a clock if that clock is connected to the PIEVENTCAPT input pin.

The edge detection logic is reset (a new capture is enabled) when the active capture interrupt is served - TCAR_IT_FLAG bit of IRQSTATUS (previously 1) is cleared. The timer functional clock (input to prescaler) is used to sample the input pin (PIEVENTCAPT). Input negative or positive pulse can be detected when pulse time is above functional clock period. An interrupt can be issued on transition detection if the capture interrupt enable bit is set in the Timer Interrupt Enable Register IRQENABLE_SET (TCAR_IT_FLAG bit).

In Figure 21-4, the TCM value is 01 and CAPT_MODE is 0 - only rising edge of the PIEVENTCAPT will trigger a capture in TCAR and only TCAR1 will update.

In Figure 21-5, the TCM value is 01 and CAPT_MODE is 1 - only rising edge of the PIEVENTCAPT will trigger a capture in TCAR1 on first enabled event and TCAR2 will update on the second enabled event.

Figure 21-4. Capture Wave Example for CAPT_MODE = 0

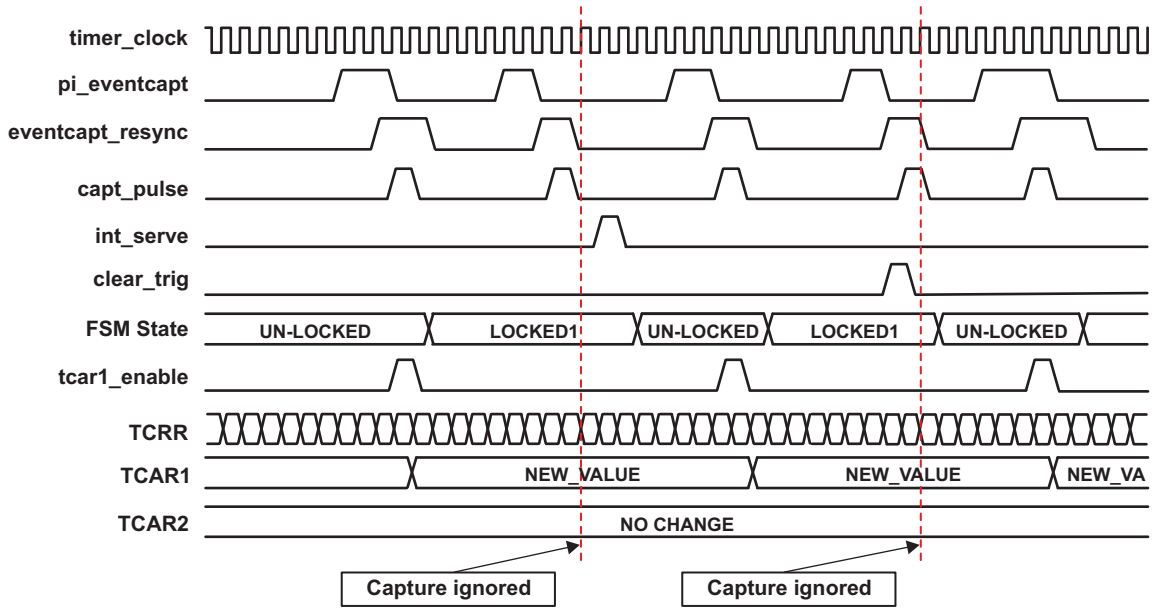
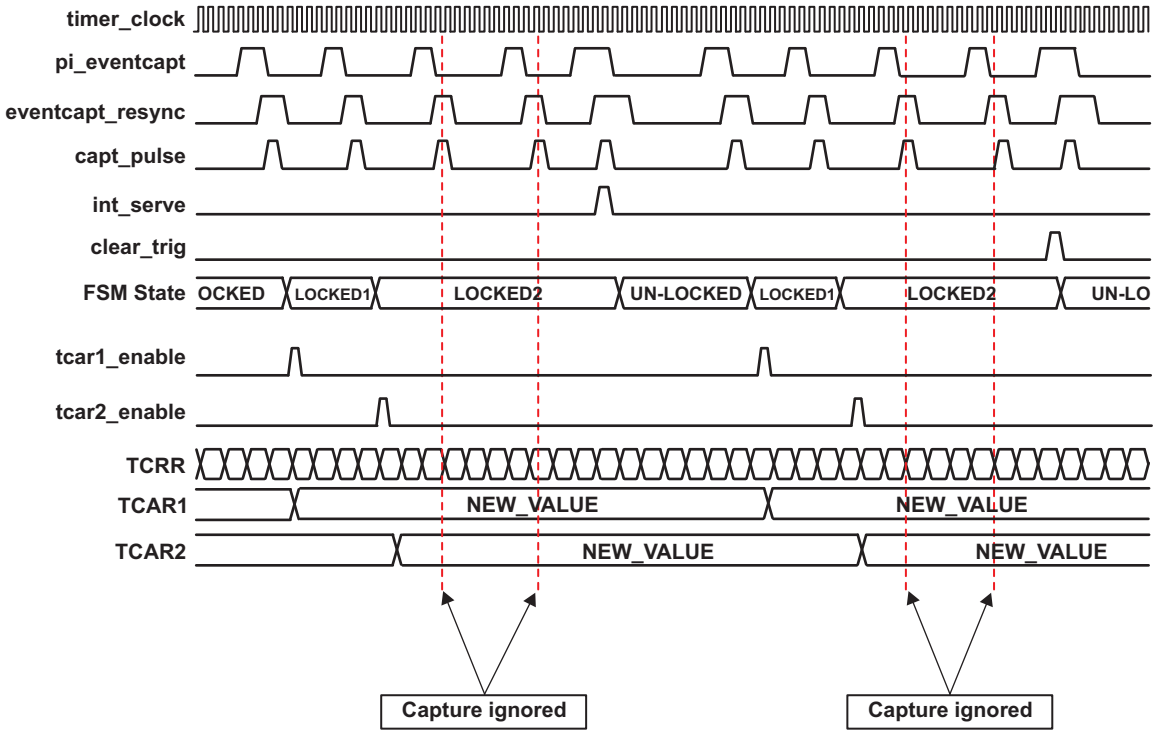


Figure 21-5. Capture Wave Example for CAPT_MODE = 1



21.2.1.3 Compare Mode Functionality

When the Compare Enable bit of the Timer Control Register (TCLR[CE]) is set to 1, the timer value in the Timer Counter Register (TCRR) is permanently compared to the value held in the Timer Match Register (TMAR). The TMAR value can be loaded at any time (timer counting or stopped). When the TCRR and the TMAR values match, an interrupt can be issued if the match interrupt enable bit is set in the Timer IRQENABLE Set Register (IRQENABLE_SET[MAT_EN_FLAG] = 1). Software should write a compare value in the TMAR register before setting the TCLR[CE] bit to avoid any unwanted interrupt due to a reset value matching effect.

A dedicated output pin can be used to generate one positive pulse (piclktimer duration) or to invert the current value (toggle mode) when an overflow and a match occur. Refer to Section 21.2.1.5 for more details related to the dedicated pulse-width modulation (PWM) output pin.

21.2.1.4 Prescaler Functionality

A prescaler counter can be used to divide the timer counter input clock frequency. The prescaler is enabled when the PRE bit of the Timer Control Register (TCLR[PRE]) is set. The 2^n division ratio value (PTV) can be configured in the TCLR register. The prescaler counter is reset when the timer counter is stopped or reloaded on-the-fly. Refer to Section 21.2.1.6 for more details on the prescaler clock ratio values.

Table 21-2. Prescaler Functionality

Contexts	Prescaler Counter	Timer Counter
Overflow (when Auto-reload on)	Reset	TLDR
TCRR Write	Reset	TCRR
TTGR Write	Reset	TLDR
Stop	Reset	Frozen

21.2.1.5 Pulse-Width Modulation

The timer can be configured to provide a programmable pulse-width modulation (PWM) output on a dedicated output pin (PORTIMERPWM). The PORTIMERPWM output pin can be configured to toggle on a specified event. The TRG field of the Timer Control Register (TCLR[TRG]) determines the trigger event when the PORTIMERPWM pin toggles. Either overflow or the combination of both overflow and match can be used to toggle the PORTIMERPWM pin.

When both overflow and match are configured to trigger the PWM pin, the matched event will be ignored from the moment the mode was set-up until the first overflow event occurs (see Figure 21-7).

The SCPWM bit in the TCLR register can be programmed to set or clear the PORTIMERPWM output signal only while the counter is stopped or the triggering is disabled. This allows fixing a deterministic state of the output pin while modulation is stopped. The modulation is synchronously stopped when the TRG bit is cleared and overflow occurs.

In the following timing diagrams, the internal overflow pulse is set each time an overflow occurs in the Timer Counter Register (TCRR), and the internal match pulse is set when the counter reaches the Timer Match Register (TMAR) value. Depending on the configured value of the TRG and PT bits of the TCLR register, the timer generates one pulse or inverts the current value on the output pin (PORTIMERPWM).

The Timer Load Register (TLDR) and Timer Match Register (TMAR) must keep values smaller than the overflow value (FFFF FFFFh) with at least 2 units. In case the PWM trigger events are both overflow and match, the difference between the values kept in the TLDR and TMAR registers must be at least 2 units. When match event is used, the CE bit of the TCLR register must be set.

In Figure 21-6, TCLR[SCPWM] is cleared to 0. In Figure 21-7, TCLR[SCPWM] is set to 1.

Figure 21-6. Timing Diagram of Pulse-Width Modulation with SCPWM = 0

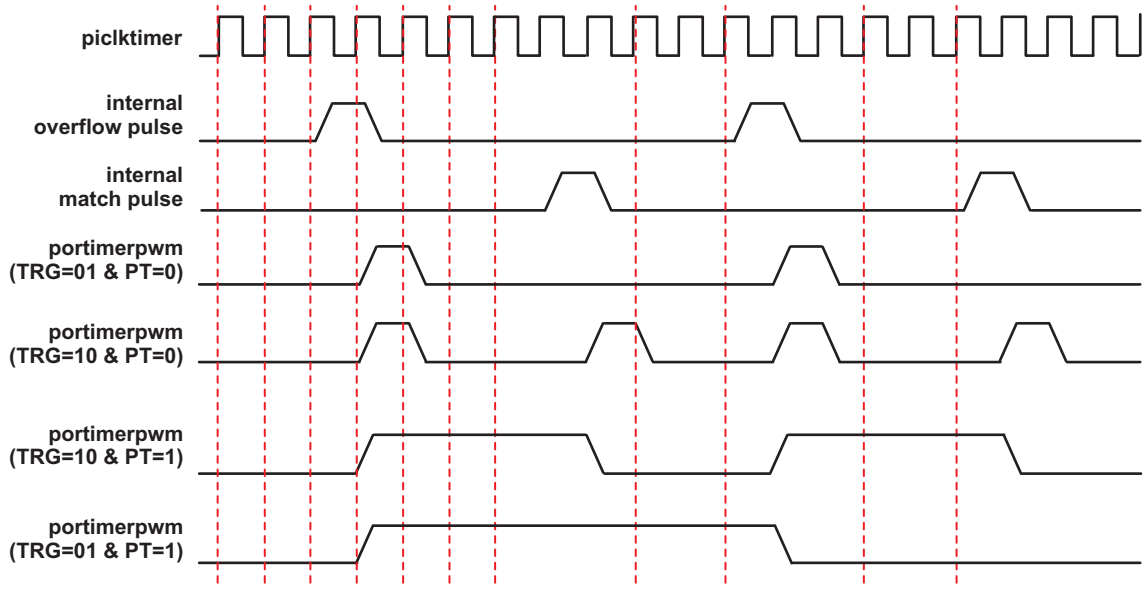
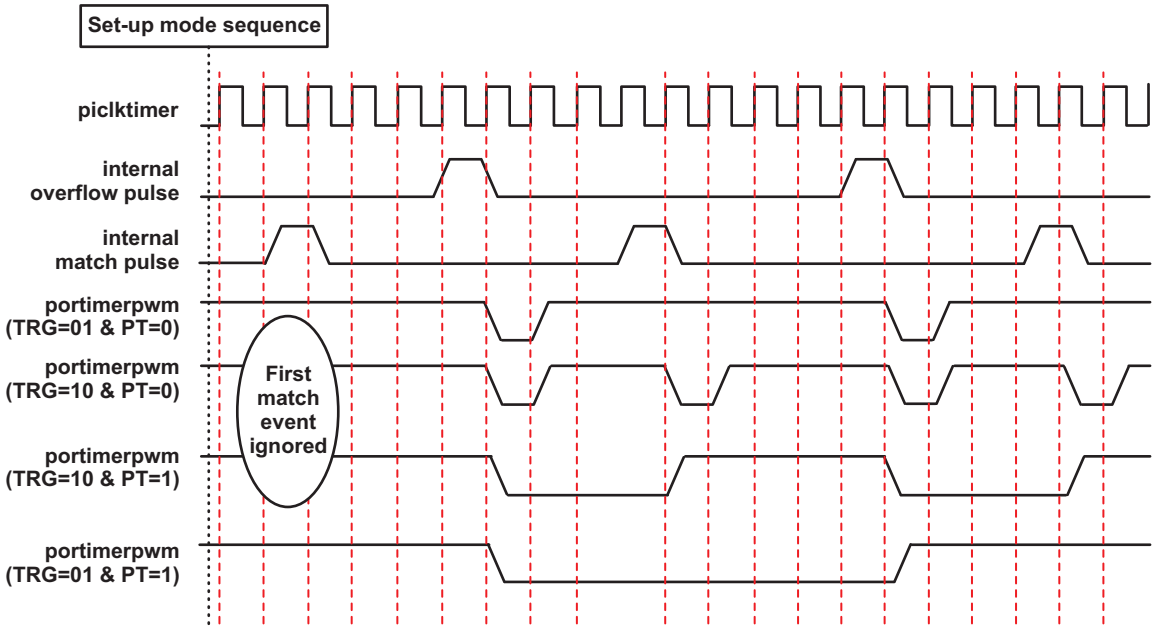


Figure 21-7. Timing Diagram of Pulse-Width Modulation with SCPWM = 1



21.2.1.6 Timer Counting Rate

The timer counter is composed of a prescaler stage and a timer counter. The prescaler stage is clocked with the timer input clock and acts as a clock divider for the timer counter stage. The ratio can be managed by accessing the ratio definition field of the control register (PTV and PRE of TCLR). See Table 21-3.

The timer rate is defined by:

- The value of the prescaler fields (PRE and PTV of TCLR register)
- The value loaded into the Timer Load Register (TLDR).

Table 21-3. Prescaler Clock Ratios Value

PRE	PTV	Divisor (PS)
0	X	1
1	0	2
1	1	4
1	2	8
1	3	16
1	4	32
1	5	64
1	6	128
1	7	256

The timer rate equation is as follows:

(FFFF FFFFh – TLDR + 1) × timer Clock period × Clock Divider (PS)

With timer Clock period = 1/ timer Clock frequency and PS = $2^{(PTV + 1)}$.

As an example, if we consider a timer clock input of 32 kHz, with a PRE field equal to 0, the timer output period is:

Table 21-4. Value and Corresponding Interrupt Period

TLDR	Interrupt period
0000 0000h	~37 h
FFFF 0000h	~2 s
FFFF FFF0h	500 µs
FFFF FFFEh	62.5 µs

21.2.1.7 Dual Mode Timer Under Emulation

During emulation mode (when PINSUSPENDN signal is active), the timer can/cannot continue running according to the value of the EmuFree bit of the timer OCP configuration register (TIOCP_CFG).

If EmuFree is 1, timer execution is not stopped and, regardless of the value of PINSUSPENDN signal, and the interrupt assertion is still generated when overflow is reached.

If EmuFree is 0, counters (prescaler/timer) are frozen and an increment start occurs again as soon as PINSUSPENDN becomes inactive. The asynchronous input pin is internally synchronized on 2 TIMER clock rising edges.

21.2.2 Accessing Registers

All registers are 32-bit wide, accessible via OCP interface with 16-bit or 32-bit OCP access (Read/Write). The 32-bit registers write update in 16 bits access must be LSB16 first and the second write access must be MSB16. For the write operation, the module allows skipping the MSB access if the user does not need to update the 16 MSB bits of the register, but only for the OCP registers (TIDR, TIOCP_CFG, IRQ_EOI, IRQSTATUS_RAW, IRQSTATUS, IRQENABLE_SET, IRQENABLE_CLR, IRQWAKEEN and TSICR). The write operation on any functional register (TCLR, TCRR, TLDR, TTGR and TMAR) must be complete (the MSB must be written even if the MSB data is not used).

21.2.2.1 Programming the Timer Registers

The TLDR, TCRR, TCLR, TIOCP_CFG, IRQ_EOI, IRQSTATUS, IRQENABLE_SET, IRQENABLE_CLR, IRQWAKEEN, TTGR, TSICR and TMAR registers write is done synchronously with OCP clock, by the host, using the OCP bus protocol.

21.2.2.2 Reading the Timer Registers

The counter register (TCRR) is a 32-bit “atomic datum” and 16-bit capture is done on the 16-bit LSB first to allow atomic LSB16 + MSB16 capture. Atomic capture is also performed for the TCAR1 and TCAR2 registers as they may change due to internal processes. DSP 16 bit accesses can be interleaved with MCU 32 bit accesses.

21.2.2.3 OCP Error Generation

The timer module responds with error indication in the following cases:

Error on write transactions

- Assert the PORSRESP = ERR signal in the same cycle as PORSCMDACCEPTED.
- Use the ERR code for PORSRESP during the response phase.

Error on read transactions

- Assert the PORSRESP = ERR signal in the same cycle as PORSCMDACCEPTED.
- Use the ERR code for PORSRESP during the response phase. PORSDATA in this case is not valid.

Table 21-5. OCP Error Reporting

Error Type	Response: SRESP = ERR
Unsupported PIOCPMCMD command	Yes
Address error: Read or write to a non-existing internal address	No
Read to write-only registers and write to read-only registers	No
Unaligned address (PIOCPMADDR ≠ 00) on read/write transaction	Yes
Unsupported PIOCPMBYTEEN on read/write transaction	Yes

NOTE: Byte enable “0000” is a supported byte enable.

21.2.3 Posted Mode Selection

A choice between the two synchronization modes will be made taking into account the frequency ratio and the stall periods that can be supported by the system, without impacting the global performance.

The posted mode selection applies only to functional registers that require synchronization on/from timer clock domain. For write operation the registers affected by this posted/non-posted selection are: TCLR, TLDR, TCRR, TTGR and TMAR. For read operation the register affected by this posted/non-posted selection are: TCRR, TCAR1 and TCAR2.

The OCP clock domain synchronous registers TIDR, TIOCP_CFG, TISTAT, IRQ_EOI,IRQSTATUS,IRQSTATUS_RAW, IRQENABLE_SET, IRQENABLE_CLR, IRQWAKEEN, TWPS and TSICR are not affected by the posted/non-posted mode selection; the write/read operation is effective and acknowledged (command accepted) after one OCP clock cycle from the command assertion.

The configuration posted/non-posted is made by setting the PIFREQRATIO when the module is integrated. The PIFREQRATIO signal should be tied to '1' when the freq (timer) < freq (OCP)/4 frequency, and tied to 0 when it is the opposite frequency ratio. The PIFREQRATIO represent the reset value of the TSICR (POSTED bit). The configuration can be changed (overwritten) by software, writing the TSICR (POSTED bit) register.

The following cases are possible:

- Posted Mode can be used when the functional frequency range is: freq (timer) < freq (OCP)/4.
- Non-Posted Mode can be used regardless of the frequency range. Recommended frequency is: freq (timer) >= freq (OCP)/4.

NOTE: The Non-Posted Mode can be also used when freq (timer) < freq (OCP)/4, but it is recommended to use the Posted Mode. Using Non-Posted Mode will delay the command accept and the transaction latency will be as described in the below chapters. Posted mode offers an OCP interface latency improvement and can be used only if the frequencies respect the freq (timer) < freq (OCP)/4 formula.

21.2.4 Write Registers Access

21.2.4.1 Write Posted

This mode can be used only if the functional frequency range is freq (timer) < freq (OCP)/4.

This mode is used if TSICR (POSTED bit) is set to 1 in the timer control register.

This mode uses a posted-write scheme to update any internal register. The write transaction is immediately acknowledged on the OCP interface, although the effective write operation will occur later, due to a resynchronisation in the timer clock domain. This has the advantage of not stalling either the interconnect system, or the CPU that requested the write transaction. For each register, a status bit is provided, that is set if there is a pending write access to this register.

In this mode, it is mandatory that the CPU checks the status bit prior to any write access. In case a write is attempted to a register with a previous access pending, the previous access is discarded without notice (this can lead to unexpected results also).

There is one status bit per register, accessible in the Timer Write Posted Status Register. When the timer module operates in this mode, there is an automatic sampling of the current timer counter value, in an OCP-synchronized capture register. Consequently, any read access to the timer counter register does not add any re-synchronization latency; the current value is always available.

A register read following a write posted register (on the same register) is not insured to read the previous write value if the write posted process is not completed. Software synchronization should be used to avoid non-coherent read.

The drawback of this automatic update mechanism is that it assumes a given relationship between the OCP interface frequency and the timer functional frequency.

This posted period is defined as the interval between the posted write access request and the reset of the posted bit in TWPS register, and can be quantified:

T (reset posted max.) = 3 OCP clock + 2.5 TIMER clock

The time when the write accomplishes is:

T (write accomplish) = 1 OCP clock + 2.5 TIMER clock

21.2.4.2 Write Non-Posted

This mode is functional regardless of the ratio between the OCP interface frequency and the functional clock frequency. Recommended functional frequency range is freq (timer) >= freq (OCP)/4.

This mode is used if TSICR (POSTED bit) is cleared to 0 in the timer control register.

This mode uses a non posted-write scheme to update any internal register. That means the write transaction will not be acknowledged on the OCP interface, until the effective write operation occurs, after the resynchronisation in the timer clock domain. The drawback is that both the interconnect system and the CPU are stalled during this period.

- The latency of the interrupt serving is increased, as the interconnect system and CPU are stalled.
- An interconnect logic, including time-out logic to detect erroneous transactions, can generate an unwanted system abort event.

The stall period is defined as the interval between the non-posted write access request and the rise of the command accept signal and can be quantified:

T (stall max.) = 3 OCP clock + 2.5 TIMER clock

The time when the write accomplishes is:

T (write accomplish) = 1 OCP clock + 2.5 TIMER clock

A register read following a write to the same register is always coherent.

21.2.5 Read Registers Access

21.2.5.1 Read Posted

This mode can be used only if the functional frequency range is freq (timer) < freq (OCP)/4.

This mode is used if TSICR (POSTED bit) is set to 1 in the timer control register.

This mode uses a posted-read scheme, for reading any internal register. The read transaction is immediately acknowledged on the OCP interface, and the value to be read has been previously resynchronised. This has the advantage of not stalling either the interconnect system, or the CPU that requested the read transaction.

21.2.5.2 Read Non-Posted

This mode is functional whatever the ratio between the OCP interface frequency and the functional clock frequency. Recommended functional frequency range is freq (timer) >= freq (OCP)/4.

This mode is used if TSICR (POSTED bit) is cleared to 0 in the timer control register.

This mode uses a non posted-read scheme, for reading any internal register. The read transaction will not be acknowledged on the OCP interface, until the effective read operation occurs, after the resynchronisation in the timer clock domain. The drawback is that both the interconnect system and the CPU are stalled during this period.

- The latency of the interrupt serving is increased, as the interconnect system and the CPU are stalled.
- An interconnect system including time-out logic to detect erroneous transactions can generate an unwanted system abort event.

This mode only applies only to three registers: TCRR, TCAR1 and TCAR2, which need resynchronisation from functional to OCP clock domains.

The stall period is defined as the interval between the non-posted read access request and the rise of the command accept signal and can be quantified:

T (stall max.) = 3 OCP clock + 2.5 TIMER clock

The time when the value is sampled is:

T (read sample) = 1 OCP clock + 2.5 TIMER clock

21.3 Timer Registers

The timer registers are listed in Table 21-6. All registers are:

- 32-bit register accessible in 16-bit mode
- Little-endian addressing

Table 21-6. Timer Registers

Offset	Acronym	Register Name	Section
00h	TIDR	Identification Register	Section 21.3.1
10h	TIOCP_CFG	Timer OCP Configuration Register	Section 21.3.2
20h	IRQ_EOI	Timer IRQ End-Of-Interrupt Register	Section 21.3.3
24h	IRQSTATUS_RAW	Timer IRQSTATUS Raw Register	Section 21.3.4
28h	IRQSTATUS	Timer IRQSTATUS Register	Section 21.3.5
2Ch	IRQENABLE_SET	Timer IRQENABLE Set Register	Section 21.3.6
30h	IRQENABLE_CLR	Timer IRQENABLE Clear Register	Section 21.3.7
34h	IRQWAKEEN	Timer IRQ Wakeup Enable Register	Section 21.3.8
38h	TCLR	Timer Control Register	Section 21.3.9
3Ch	TCRR	Timer Counter Register	Section 21.3.10
40h	TLDR	Timer Load Register	Section 21.3.11
44h	TTGR	Timer Trigger Register	Section 21.3.12
48h	TWPS	Timer Write Posted Status Register	Section 21.3.13
4Ch	TMAR	Timer Match Register	Section 21.3.14
50h	TCAR1	Timer Capture Register	Section 21.3.15
54h	TSICR	Timer Synchronous Interface Control Register	Section 21.3.16
58h	TCAR2	Timer Capture Register	Section 21.3.17

21.3.1 TIDR Register

This read only register contains the revision number of the module. A write to this register has no effect. This register is used by software to track features, bugs, and compatibility.

Figure 21-8. TIDR Register

31	30	29	28	27	16
SCHEME		Reserved		FUNC	
R-1		R-0		R-FFFh	

15	11	10	8	7	6	5	0
R_RTL		X_MAJOR		CUSTOM		Y_MINOR	
R-2h		R-3h		R-0		R-1	

LEGEND: R = Read only; -*n* = value after reset

Table 21-7. TIDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	SCHEME	R	1	Used to distinguish between old scheme and current.
29-28	Reserved	R	0	Reads return 0
27-16	FUNC	R	FFFh	Function indicates a software compatible module family.
15-11	R_RTL	R	2h	RTL Version (R).
10-8	X_MAJOR	R	3h	Major Revision (X).
7-6	CUSTOM	R	0	Indicates a special version for a particular device.
5-0	Y_MINOR	R	1	Minor Revision (Y).

21.3.2 TIOCP_CFG Register

This register allows controlling various parameters of the OCP interface.

Figure 21-9. TIOCP_CFG Register

31	4	3	2	1	0
Reserved		IDLEMODE		EMUFREE	SOFTRESET
R-0		R/W-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 21-8. TIOCP_CFG Register Field Descriptions

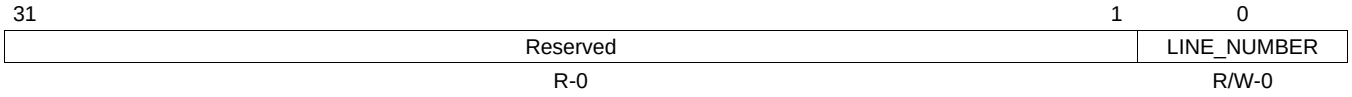
Bit	Field	Type	Reset	Description
31-4	Reserved	R	0	Reserved
3-2	IDLEMODE	R/W	0	Power management, req/ack control 0 = Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally, i.e. regardless of the IP module's internal requirements. Backup mode, for debug only. 1h = No-idle mode: local target never enters idle state. Backup mode, for debug only. 2h = Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module shall not generate (IRQ- or DMA-request-related) wakeup events. 3h = Smart-idle wakeup-capable mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module may generate (IRQ- or DMA-request-related) wakeup events when in idle state. Mode is only relevant if the appropriate IP module "swakeup" output(s) is (are) implemented.
1	EMUFREE	R/W	0	Emulation mode 0 = The timer is frozen in emulation mode (PINSUSPENDN signal active). 1 = The timer runs free, regardless of PINSUSPENDN value.
0	SOFTRESET	R/W	0	Software reset Read 0 = Reset done, no pending action Write 0 = No action Read 1 = Reset ongoing Write 1 = Initiate software reset

21.3.3 IRQ_EOI Register

Software End-Of-Interrupt: Allows the generation of further pulses on the interrupt line, if a new interrupt event is pending, when using the pulsed output.

Unused when using the level interrupt line (depending on module integration).

Figure 21-10. IRQ_EOI Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-9. IRQ_EOI Register Field Descriptions

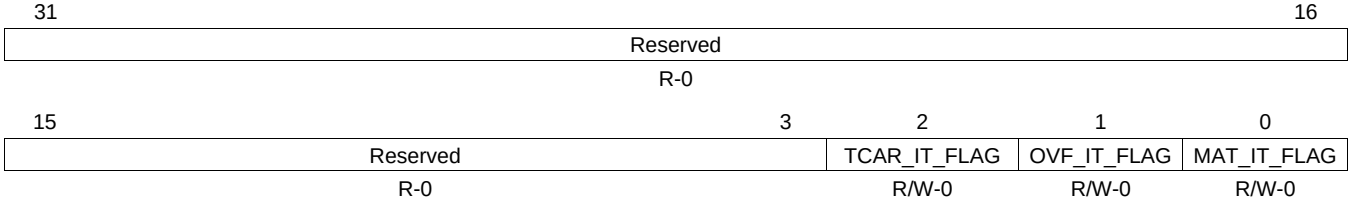
Bit	Field	Type	Reset	Description
31-1	Reserved	R	0	Reserved
0	LINE_NUMBER	R/W	0	Write the number of the interrupt line to apply a SW EOI to it. Note that there is only a single line (that is, number 0) Read 0 = Always returns 0 Write 0 = SW EOI on interrupt line Write 1 = No action

21.3.4 IRQSTATUS_RAW Register

Component interrupt request status.

Check the corresponding secondary status register. Raw status is set even if event is not enabled. Write 1 to set the (raw) status, mostly for debug.

Figure 21-11. IRQSTATUS_RAW Register



LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 21-10. IRQSTATUS_RAW Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	Reserved	R	0	Reserved
2	TCAR_IT_FLAG	R/W	0	IRQ status for Capture Read 0 = No event pending Write 0 = No action Read 1 = IRQ event pending Write 1 = Trigger IRQ event by software
1	OVF_IT_FLAG	R/W	0	IRQ status for Overflow Read 0 = No event pending Write 0 = No action Read 1 = IRQ event pending Write 1 = Trigger IRQ event by software
0	MAT_IT_FLAG	R/W	0	IRQ status for Match Read 0 = No event pending Write 0 = No action Read 1 = IRQ event pending Write 1 = Trigger IRQ event by software

21.3.5 IRQSTATUS Register

Component interrupt request status.

Check the corresponding secondary status register. Enabled status is not set unless event is enabled. Write 1 to clear the status after interrupt has been serviced (raw status gets cleared, that is, even if not enabled).

Figure 21-12. IRQSTATUS Register

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		TCAR_IT_FLAG	OVF_IT_FLAG	MAT_IT_FLAG
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-11. IRQSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	Reserved	R	0	Reserved
2	TCAR_IT_FLAG	R/W	0	IRQ status for Capture Read 0 = No event pending Write 0 = No action Read 1 = IRQ event pending Write 1 = Clear pending event, if any
1	OVF_IT_FLAG	R/W	0	IRQ status for Overflow Read 0 = No event pending Write 0 = No action Read 1 = IRQ event pending Write 1 = Clear pending event, if any
0	MAT_IT_FLAG	R/W	0	IRQ status for Match Read 0 = No event pending Write 0 = No action Read 1 = IRQ event pending Write 1 = Clear pending event, if any

21.3.6 IRQENABLE_SET Register

Component interrupt request enable.

Write 1 to set (enable interrupt).

Readout equal to corresponding _CLR register.

Figure 21-13. IRQENABLE_SET Register

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		TCAR_EN_FLAG	OVF_EN_FLAG	MAT_EN_FLAG
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-12. IRQENABLE_SET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	Reserved	R	0	Reserved
2	TCAR_EN_FLAG	R/W	0	IRQ enable for Compare Read 0 = IRQ event is disabled Write 0 = No action Read 1 = IRQ event is enabled Write 1 = Set IRQ enable
1	OVF_EN_FLAG	R/W	0	IRQ enable for Overflow Read 0 = IRQ event is disabled Write 0 = No action Read 1 = IRQ event is enabled Write 1 = Set IRQ enable
0	MAT_EN_FLAG	R/W	0	IRQ enable for Match Read 0 = IRQ event is disabled Write 0 = No action Read 1 = IRQ event is enabled Write 1 = Set IRQ enable

21.3.7 IRQENABLE_CLR Register

Component interrupt request enable.

Write 1 to clear (disable interrupt).

Readout equal to corresponding _SET register.

Figure 21-14. IRQENABLE_CLR Register

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		TCAR_EN_FLAG	OVF_EN_FLAG	MAT_EN_FLAG
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-13. IRQENABLE_CLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	Reserved	R	0	Reserved
2	TCAR_EN_FLAG	R/W	0	IRQ enable for Compare Read 0 = IRQ event is disabled Write 0 = No action Read 1 = IRQ event is enabled Write 1 = Clear IRQ enable
1	OVF_EN_FLAG	R/W	0	IRQ enable for Overflow Read 0 = IRQ event is disabled Write 0 = No action Read 1 = IRQ event is enabled Write 1 = Clear IRQ enable
0	MAT_EN_FLAG	R/W	0	IRQ enable for Match Read 0 = IRQ event is disabled Write 0 = No action Read 1 = IRQ event is enabled Write 1 = Clear IRQ enable

21.3.8 IRQWAKEEN Register

Wakeup-enabled events taking place when module is idle will generate an asynchronous wakeup.

Figure 21-15. IRQWAKEEN Register

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		TCAR_WUP_FLAG	OVF_WUP_FLAG	MAT_WUP_FLAG
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-14. IRQWAKEEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	Reserved	R	0	Reserved
2	TCAR_WUP_ENA	R/W	0	Wakeup generation for Compare 0 = Wakeup disabled 1 = Wakeup enabled
1	OVF_WUP_ENA	R/W	0	Wakeup generation for Overflow 0 = Wakeup disabled 1 = Wakeup enabled
0	MAT_WUP_ENA	R/W	0	Wakeup generation for Match 0 = Wakeup disabled 1 = Wakeup enabled

21.3.9 TCLR Register

When the TCM field passed from (00) to any other combination then the TCAR_IT_FLAG and the edge detection logic are cleared.

The ST bit of TCLR register may be updated from the OCP interface or reset due to one-shot overflow. The OCP interface update has the priority.

Figure 21-16. TCLR Register

31	16
Reserved	
R-0	

15	14	13	12	11	10	9	8
Reserved	GPO_CFG	CAPT_MODE	PT	TRG		TCM	
R-0	R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	

7	6	5	4	2	1	0
SCPWM	CE	PRE	PTV		AR	ST
R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-15. TCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-15	Reserved	R	0	Reserved
14	GPO_CFG	R/W	0	General purpose output—this register directly drives the PORGPOCFG output pin 0 = PORGPOCFG drives 0 1 = PORGPOCFG drives 1 Note: For specific use of GPO_CFG bit, see Section 21.1.4.
13	CAPT_MODE	R/W	0	Capture mode. 0 = Single capture 1 = Capture on second event
12	PT	R/W	0	Pulse or toggle mode on PORTIMERPWM output pin 0 = Pulse 1 = Toggle
11-10	TRG	R/W	0	Trigger output mode on PORTIMERPWM output pin 0 = No trigger 1h = Trigger on overflow 2h = Trigger on overflow and match 3h = Reserved
9-8	TCM	R/W	0	Transition Capture Mode on PIEVENTCAPT input pin 0 = No capture 1h = Capture on low to high transition 2h = Capture on high to low transition 3h = Capture on both edge transition
7	SCPWM	R/W	0	This bit should be set or cleared while the timer is stopped or the trigger is off 0 = Clear the PORTIMERPWM output pin and select positive pulse for pulse mode 1 = Set the PORTIMERPWM output pin and select negative pulse for pulse mode
6	CE	R/W	0	0 = Compare mode is disabled 1 = Compare mode is enabled
5	PRE	R/W	0	Prescaler enable 0 = The TIMER clock input pin clocks the counter 1 = The divided input pin clocks the counter
4-2	PTV	R/W	0	Pre-scale clock Timer value. See Table 21-3 for more details.
1	AR	R/W	0	0 = One shot timer 1 = Auto-reload timer
0	ST	R/W	0	In the case of one-shot mode selected (AR = 0), this bit is automatically reset by internal logic when the counter is overflowed. 0 = Stop timer: Only the counter is frozen 1 = Start timer

21.3.10 TCRR Register

The TCRR register is a 32-bit register, 16-bit addressable. The MCU can perform a 32-bit access or two 16-bit accesses to access the register while the DSP performs 2 consecutive 16-bit transactions. Note that since the OCP clock is completely asynchronous with the timer clock, some synchronization is done in order to make sure that the TCRR value is not read while it is being incremented.

In 16-bit mode the following sequence must be followed to read the TCRR register properly:

First, perform an OCP Read Transaction to Read the lower 16-bit of the TCRR register (offset = 3Ch). When the TCRR is read and synchronized, the lower 16-bit LSBs are driven onto the output OCP data bus and the upper 16-bit MSBs of the TCRR register are stored in a temporary register.

Second, perform an OCP Read Transaction to read the upper 16-bit of the TCRR register (offset = 3Eh). During this Read, the value of the upper 16-bit MSBs that has been temporary register is forwarded onto the output OCP data bus.

So, to read the value of TCRR correctly, the first OCP read access has to be to the lower 16-bit (offset = 3Ch), followed by OCP read access to the upper 16-bit (offset = 3Eh).

As the TCRR is updated using more sources (shadow_in_tcrr, incremented value of tcrr, TLDR and “0”), a priority order will be defined:

The first priority is the OCP update.

The second is the reload way (triggered through TTGR reg. or following an auto-reload overflow).

The third is the one-shot overflow reset to 0.

The last is the incremented value.

Figure 21-17. TCRR Register

31	0
TIMER_COUNTER	
R/W-0	

LEGEND: R = Read only; -*n* = value after reset

Table 21-16. TCRR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TIMER_COUNTER	R/W	0	Value of TIMER counter

21.3.11 TLDR Register

LOAD_VALUE must be different than the timer overflow value (FFFF FFFFh).

Figure 21-18. TLDR Register

31	0
LOAD_VALUE	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 21-17. TLDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LOAD_VALUE	R/W	0	Timer counter value loaded on overflow in auto-reload mode or on TTGR write access

21.3.12 TTGR Register

The read value of this register is always FFFF FFFFh.

Figure 21-19. TTGR Register

31	0
TTGR_VALUE	
R/W-FFFF FFFFh	

LEGEND: R = Read only; -*n* = value after reset

Table 21-18. TTGR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TTGR_VALUE	R/W	FFFF FFFFh	Any write to the TTGR register will load the value from TLDR into TCRR and clear the prescaler counter. This will occur regardless of the AR field value in the TCLR register. A read access of TTGR will always return FFFF FFFFh.

21.3.13 TWPS Register

In posted mode the software must read the pending write status bits (Timer Write Posted Status register bits [4:0]) to insure that following write access will not be discarded due to on going write synchronization process. These bits are automatically cleared by internal logic when the write to the corresponding register is acknowledged.

Figure 21-20. TWPS Register

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		W_PEND_TMAR	W_PEND_TTGR	W_PEND_TLDR	W_PEND_TCRR	W_PEND_TCLR
R-0		R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 21-19. TWPS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	Reserved	R	0	Reserved
4	W_PEND_TMAR	R	0	When equal to 1, a write is pending to the TMAR register
3	W_PEND_TTGR	R	0	When equal to 1, a write is pending to the TTGR register
2	W_PEND_TLDR	R	0	When equal to 1, a write is pending to the TLDR register
1	W_PEND_TCRR	R	0	When equal to 1, a write is pending to the TCRR register
0	W_PEND_TCLR	R	0	When equal to 1, a write is pending to the TCLR register

21.3.14 TMAR Register

The TMAR register stores the value to be compared against the counter's current value using the timer's compare logic.

Figure 21-21. TMAR Register

31	0
COMPARE_VALUE	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 21-20. TMAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	COMPARE_VALUE	R/W	0	Value to be compared to the timer counter

21.3.15 TCAR1 Register

When the appropriate (rising, falling or both) transition is detected in the edge detection logic the current counter value is stored to the TCAR1 register. Note that since the OCP clock is completely asynchronous with the timer clock, some synchronization is done in order to make sure that the TCAR1 value is not read while it is being updated due to some capture event.

In 16-bit mode the following sequence must be followed to read the TCAR1 register properly:

First, perform an OCP Read Transaction to Read the lower 16-bits of the TCAR1 register.

Second, perform an OCP Read Transaction to read the upper 16-bits of the TCAR1 register.

Figure 21-22. TCAR1 Register

31	0
CAPTURED_VALUE	
R-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 21-21. TCAR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CAPTURED_VALUE	R	0	Timer counter value captured on an external event trigger

21.3.16 TSICR Register

Access to this register is not stalled even if the timer is in non-posted mode configuration. To abort any wrong behavior, software can permanently reset the functional part of the module. Also in case of a wrong hardware PIFREQRATIO tied the POSTED field can be reprogrammed on the fly, so deadlock situation cannot happen.

Reset value of POSTED depends on hardware integration module at design time. Software must read POSTED field to get the hardware module configuration.

Figure 21-23. TSICR Register

31	3	2	1	0
Reserved		POSTED	SFT	Reserved
R-0		R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-22. TSICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	Reserved	R	0	Reserved
2	POSTED	R/W	0	PIFREQRATIO 0 = Posted mode inactive: will delay the command accept output signal. 1 = Posted mode active (clocks ratio needs to fit freq (timer) less than freq (OCP)/4 frequency requirement)
1	SFT	R/W	0	This bit resets all the function parts of the module. During reads it always returns 0. 0 = Software reset is disabled 1 = Software reset is enabled
0	Reserved	R	0	Reserved

21.3.17 TCAR2 Register

When the appropriate (rising, falling or both) transition is detected in the edge detection logic and the capture on second event is activated from the control register (TCLR), the current counter value is stored to the TCAR2 register. Note that since the OCP clock is completely asynchronous with the timer clock, some synchronization is done in order to make sure that the TCAR2 value is not read while it is being updated due to some capture event.

In 16-bit mode the following sequence must be followed to read the TCAR2 register properly:

First, perform an OCP Read Transaction to Read the lower 16-bits of the TCAR2 register.

Second, perform an OCP Read Transaction to read the upper 16-bits of the TCAR2 register.

Figure 21-24. TCAR2 Register

31	0
CAPTURED_VALUE	
R-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 21-23. TCAR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CAPTURED_VALUE	R	0	Timer counter value captured on an external event trigger

Watchdog Timer

This chapter describes the watchdog timer.

Topic **Page**

22.1 Introduction

22.1.1 Overview

The watchdog timer is an upward counter capable of generating a pulse on the reset pin and an interrupt to the device system modules following an overflow condition. The watchdog timer serves resets to the PRCM module and serves watchdog interrupts to the host subsystem. The reset of the PRCM module causes warm reset of the device.

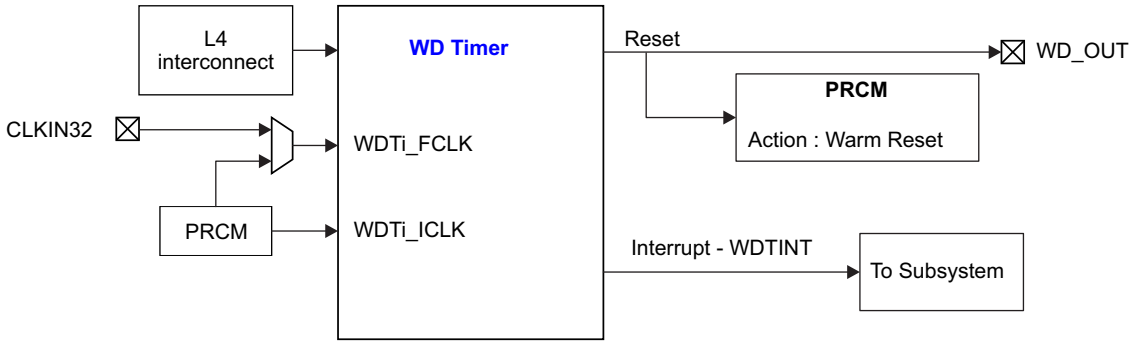
The watchdog timer can be accessed, loaded, and cleared by registers through the L4 interface. The watchdog timer has the 32-kHz clock for its timer clock input.

The watchdog timer connects to a single target agent port on the L4 interconnect. The default state of the watchdog timer is enabled and not running.

22.1.2 Functional Block Diagram

Figure 22-1 shows a block diagram of the watchdog timer.

Figure 22-1. Watchdog Timer Block Diagram



22.1.3 Features

The main features of the watchdog timer controllers are:

- L4 slave interface support:
  - 32-bit data bus width
  - 32-/16-bit access supported
  - 8-bit access not supported
  - 11-bit address bus width
  - Burst mode not supported
  - Write nonposted transaction mode only
- Free-running 32-bit upward counter
- Programmable divider clock source (2^n where n = 0-7)
- On-the-fly read/write register (while counting)
- Subset programming model of the timer
- The watchdog timers are reset either on power-on or after a warm reset before they start counting.
- Reset or interrupt actions when a timer overflow condition occurs
- The watchdog timer generates a reset or an interrupt in its hardware integration.

22.1.4 Watchdog Timer Environment

The watchdog timers are accessible through the L4 interface.

22.2 Architecture

22.2.1 Power Management

There are two clock domains in the watchdog timers:

- Functional clock domain: WDTi_FCLK is a 32 kHz watchdog timer functional clock. It is used to clock the watchdog timer internal logic.
- Interface clock domain: WDTi_ICLK is a 125 MHz watchdog timer interface clock. It is used to synchronize the watchdog timer L4 port to the L4 interconnect. All accesses from the interconnect are synchronous to WDTi_ICLK.

In this device, the watchdog timer clocks are always On. The clocks cannot be turned off, if the watchdog timer is not being used.

22.2.2 Interrupts

Table 22-1 list the event flags, and their masks, that cause module interrupts.

Table 22-1. Watchdog Timer Events

Event Flag	Event Mask	Mapping	Comments
WDT_WIRQSTAT[0] EVENT_OVF	WDT_WIRQENSET/WDT_WIRQENCLR[0] OVF_IT_ENA	WDTINT	Watchdog timer overflow
WDT_WIRQSTAT[1] EVENT_DLY	WDT_WIRQENSET/WDT_WIRQENCLR[1] DLY_IT_ENA	WDTINT	Watchdog delay value reached

22.2.3 General Watchdog Timer Operation

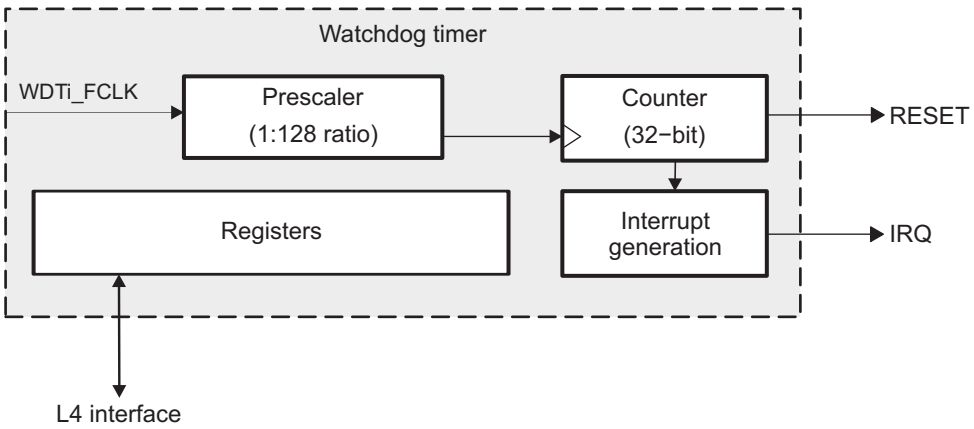
The watchdog timers are based on an upward 32-bit counter coupled with a prescaler. The counter overflow is signaled through two independent signals: a simple reset signal and an interrupt signal, both active low. Figure 22-2 is a functional block diagram of the watchdog timer.

The interrupt generation mechanism is controlled through the WDT_WIRQENSET/WDT_WIRQENCLR and WDT_WIRQSTAT registers.

The prescaler ratio can be set from 1 to 128 by accessing the WDT_WCLR[4:2] PTV bit field and the WDT_WCLR[5] PRE bit of the watchdog control register (WDT_WCLR).

The current timer value can be accessed on-the-fly by reading the watchdog timer counter register (WDT_WCRR), can be modified by accessing the watchdog timer load register (WDT_WLDR) (no on-the-fly update), or can be reloaded by following a specific reload sequence on the watchdog timer trigger register (WDT_WTGR). A start/stop sequence applied to the watchdog timer start/stop register (WDT_WSPR) can start and stop the watchdog timers.

Figure 22-2. 32-Bit Watchdog Timer Functional Block Diagram



22.2.4 Reset Context

The watchdog timers are enabled after reset. Table 22-2 lists the default reset values of the two watchdog timer load registers (the WDT_WLDR) and prescaler ratios (the WDT_WCLR[4:2] PTV bit field). To get these values, software must read the corresponding WDT_WCLR[4:2] PTV bit field and the 32-bit register to retrieve the static configuration of the module.

Table 22-2. Count and Prescaler Default Reset Values

Timer	WDT_WLDR Reset Value	PTV Reset Value
WDT	FFFF FFBEh	0

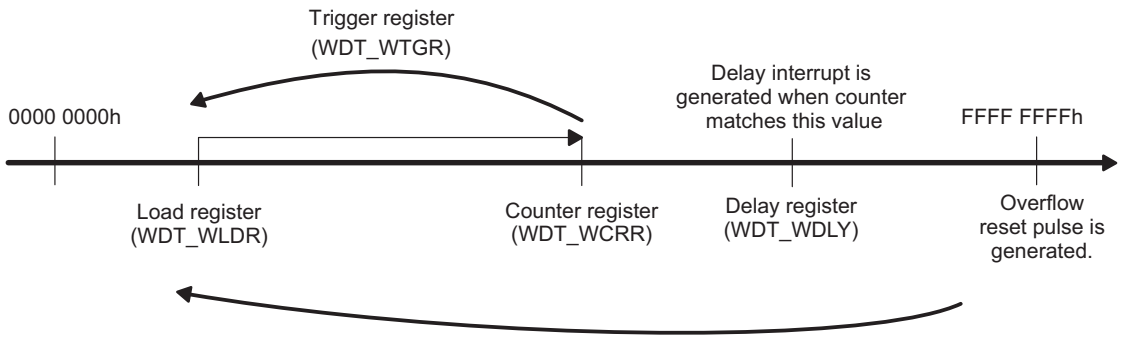
22.2.5 Overflow/Reset Generation

When the watchdog timer counter register (WDT_WCRR) overflows, an active-low reset pulse is generated to the PRCM module. This RESET pulse causes the PRCM module to generate global WARM reset of the device. It is also driven out of the device through the WD_OUT pin. This pulse is one prescaled timer clock cycle wide and occurs at the same time as the timer counter overflow.

After reset generation, the counter is automatically reloaded with the value stored in the watchdog load register (WDT_WLDR) and the prescaler is reset (the prescaler ratio remains unchanged). When the reset pulse output is generated, the timer counter begins incrementing again.

Figure 22-3 shows a general functional view of the watchdog timers.

Figure 22-3. Watchdog Timers General Functional View



22.2.6 Prescaler Value/Timer Reset Frequency

Each watchdog timer is composed of a prescaler stage and a timer counter.

The timer rate is defined by the following values:

- Value of the prescaler fields (the WDT_WCLR[5] PRE bit and the WDT_WCLR[4:2] PTV bit field)
- Value loaded into the timer load register (WDT_WLDR)

The prescaler stage is clocked with the timer clock and acts as a clock divider for the timer counter stage. The ratio is managed by accessing the ratio definition field (the WDT_WCLR[4:2] PTV bit field) and is enabled with the WDT_WCLR[5] PRE bit.

Table 22-3 lists the prescaler clock ratio values.

Table 22-3. Prescaler Clock Ratio Values

WDT_WCLR[5] PRE	WDT_WCLR[4:2] PTV	Clock Divider (PS)
0	X	1
1	0	1
1	1	2
1	2	4
1	3	8
1	4	16
1	5	32
1	6	64
1	7	128

Thus the watchdog timer overflow rate is expressed as:

OVF_Rate = (FFFF FFFFh – WDT_WLDR + 1) × (wd-functional clock period) × PS

where wd-functional clock period = 1/(wd-functional clock frequency) and PS = $2^{(PTV)}$

CAUTION

Internal resynchronization causes some latency in any software write to WDT_WSPR before WDT_WSPR is updated with the programmed value:

1.5 × functional clock cycles ≤ write_WDT_WSPR_latency ≤ 2.5 × functional clock cycles

Remember to consider this latency whenever the watchdog timer must be started or stopped.

For example, for a timer clock input of 32 kHz with a prescaler ratio value of 1 (clock divided by 2) and WDT_WCLR[5] PRE = 1 (clock divider enabled), the reset period is as listed in Table 22-4.

Table 22-4. Reset Period Examples

WDT_WLDR Value	Reset Period
0000 0000h	74 h 56 min
FFFF 0000h	4 s
FFFF FFF0h	1 ms
FFFF FFFFh	62.5 us

CAUTION

- Ensure that the reloaded value allows the correct operation of the application. When a watchdog timer is enabled, software must periodically trigger a reload before the counter overflows. Hence, the value of the WDT_WLDR[31:0] bit field must be chosen according to the ongoing activity preceding the watchdog reload.
- Due to design reasons, WDT_WLDR[31:0] = FFFF FFFFh is a special case, although such a value of WDT_WLDR is meaningless. When WDT_WLDR is programmed with the overflow value, a triggering event generates a reset/interrupt one functional clock cycle later, even if the watchdog timer is stopped.

Table 22-5 lists the default reset periods for the watchdog timers.

Table 22-5. Default Watchdog Timer Reset Periods

Watchdog Timers	Clock Source	Default Reset Period
WDT	32 kHz	2 s

22.2.7 Triggering a Timer Reload

To reload the timer counter and reset the prescaler before reaching overflow, a reload command is executed by accessing the watchdog timer trigger register (WDT_WTGR) using a specific reload sequence.

The specific reload sequence is performed whenever the written value on the WDT_WTGR register differs from its previous value. In this case, reload is executed in the same way as an overflow autoreload, but without the generation of a reset pulse.

The timer counter is loaded with the value of the watchdog timer load register (the WDT_WLDR[31:0] TIMER_LOAD bit field), and the prescaler is reset.

22.2.8 Start/Stop Sequence for Watchdog Timers (Using the WDT_WSPR Register)

To start and stop a watchdog timer, access must be made through the start/stop register (WDT_WSPR) using a specific sequence.

To disable the timer, follow this sequence:

1. Write XXXX AAAAh in WDT_WSPR.
2. Write XXXX 5555h in WDT_WSPR.

To enable the timer, follow this sequence:

1. Write XXXX BBBBh in WDT_WSPR.
2. Write XXXX 4444h in WDT_WSPR.

All other write sequences on the WDT_WSPR register have no effect on the start/stop feature of the module.

22.2.9 Modifying Timer Count/Load Values and Prescaler Setting

To modify the timer counter value (the WDT_WCRR register), prescaler ratio (the WDT_WCLR[4:2] PTV bit field), delay configuration value (the WDT_WDLY[31:0] DLY_VALUE bit field), or the load value (the WDT_WLDR[31:0] TIMER_LOAD bit field), the watchdog timer must be disabled by using the start/stop sequence (the WDT_WSPR register).

After a write access, the load register value and prescaler ratio registers are updated immediately, but new values are considered only after the next consecutive counter overflow or after a new trigger command (the WDT_WTGR register).

22.2.10 Watchdog Counter Register Access Restriction (WDT_WCRR Register)

A 32-bit shadow register is implemented to read a coherent value of the WDT_WCRR register because the WDT_WCRR register is directly related to the timer counter value and is updated on the timer clock (WDT_FCLK). The shadow register is updated by a 16-bit LSB read command.

NOTE: Although the L4 clock (WDT_ICLK) is completely asynchronous with the timer clock (WDT_FCLK), some synchronization is performed to ensure that the value of the WDT_WCRR register is not read while it is being incremented.

When 32-bit read access is performed, the shadow register is not updated. Read access is performed directly from the accessed register.

To ensure that a coherent value is read inside WDT_WCRR, the first read access is to the lower 16 bits (offset = 8h), followed by read access to the upper 16 bits (offset = Ah).

22.2.11 Watchdog Timer Interrupt Generation

When an interrupt source occurs, the interrupt status bit (the WDT_WIRQSTAT[0] EVENT_OVF or WDT_WIRQSTAT[1] EVENT_DLY bit) is set to 1. The output interrupt line (WDTi_IRQ) is asserted (active low) when status (the EVENT_xxx bit) and enable (the xxx_IT_ENA bit) flags are set to 1; the order is not relevant. Writing 1 to the enable bit (the status is already set at 1) also triggers the interrupt in the normal order (enable first, status next). The pending interrupt event is cleared when the set status bit is overwritten by a value of 1 by a write command in the WDT_WIRQSTAT register. Reading the WDT_WIRQSTAT register and writing the value back allows a fast interrupt acknowledge process.

The watchdog timer issues an overflow interrupt if this interrupt is enabled in the watchdog interrupt enable register (WDT_WIRQENSET[0] OVF_IT_ENA = 1). When the overflow occurs, the interrupt status bit (the WDT_WIRQSTAT[0] EVENT_OVF bit) is set to 1. The output interrupt line (WDT_IRQ) is asserted (active low) when status (EVENT_OVF) and enable (OVF_IT_ENA) flags are set to 1; the order is not relevant. This interrupt can be disabled by setting the WDT_WIRQENCLR[0] OVF_IT_ENA bit to 1.

The watchdog can issue the delay interrupt if this interrupt is enabled in the interrupt enable register (WDT_WIRQENSET[1] DLY_IT_ENA = 1). When the counter is running and the counter value matches the value stored in the delay configuration register (WDT_WDLY), the corresponding interrupt status bit is set in the watchdog status register (WDT_WIRQSTAT) and the output interrupt line is asserted (active low) when the flag (EVENT_DLY) and enable (DLY_IT_ENA) bits are 1 in the WDT_WIRQSTAT and WDT_WIRQENSET registers, respectively; the order (normally enable, then flag), is not relevant. This interrupt can be disabled by setting the WDT_WIRQENCLR[1] DLY_IT_ENA bit to 1.

NOTE: Writing 0 to the WDT_WIRQSTAT[0] EVENT_OVF bit or the WDT_WIRQSTAT[1] EVENT_DLY bit has no effect.

The two clock domains are resynchronized because the interrupt event is generated on the functional clock domain (WDTi_FCLK) during the updating of the interrupt status register (WDT_WIRQSTAT).

The WDT_WDLY register is used to specify the value of the delay configuration register. The delay time to interrupt is the difference between the reload value stored in the counter load register (WDT_WLDR) and the programmed value in this register (WDT_WDLY).

Use the following formula to estimate the delay time:

Delay time period = (WDT_WDLY – WDT_WLDR + 1) × Timer clock period × Clock divider

Where:

- Timer clock period = 1/(Timer clock frequency)
- Clock divider = 2PTV

If the counter value (WDT_WCRR) reaches the programmed value (WDT_WDLY), the status bit (EVENT_DLY) gets set in the interrupt status register (WDT_WIRQSTAT), and an interrupt occurs if the corresponding enable bit is set in the interrupt enable register (WDT_WIRQENSET).

CAUTION

If the reload event occurs (after a triggering sequence or after a reset sequence) before reaching the programmed value (WDT_WDLY[31:0] WDLY_VALUE), no interrupt is generated.

Also, no interrupt is generated if the value programmed in the delay configuration register (WDT_WDLY) is less than the value stored in the counter load register (WDT_WLDR).

22.2.12 Watchdog Timers Under Emulation

During emulation mode, the watchdog timer can/cannot continue running, according to the value of the WDT_WDSC[5] EMUFREE bit of the system configuration register (WDT_WDSC).

- When EMUFREE is 1, watchdog timer execution is not stopped and a reset pulse is still generated when overflow is reached.
- When EMUFREE is 0, the counters (prescaler/timer) are frozen and incrementation restarts after exiting from emulation mode.

22.2.13 Accessing Watchdog Timer Registers

Posted/nonposted selection applies only to functional registers that require synchronization on/from the timer functional clock domain (WDTi_FCLK). For write/read operation, the following registers are affected:

- WDT_WCLR
- WDT_WCRR
- WDT_WLDR
- WDT_WTGR
- WDT_WDLY
- WDT_WSPR

The timer interface clock domain synchronous registers are not affected by the posted/nonposted selection; the write/read operation is effective and acknowledged (command accepted) after one WDT_ICLK cycle from the command assertion. The timer interface clock domain synchronous registers are:

- WDT_WIDR
- WDT_WDSC
- WDT_WDST
- WDT_WIRQSTATRAW
- WDT_WIRQSTAT
- WDT_WIRQENSET
- WDT_WIRQENCLR
- WDT_WWPS

NOTE: Accesses to WDT2 and WDT3 are posted.

22.3 Low-Level Programming Model

This section covers the low-level hardware programming sequences for configuration and use of the module.

22.3.1 Global Initialization

22.3.1.1 Surrounding Modules Global Initialization

This section identifies the requirements for initializing the surrounding modules when the watchdog timer is to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration and environment of the watchdog timer (see Table 22-6).

Table 22-6. Global Initialization of Surrounding Modules

Surrounding Modules	Comments
PRCM	The module interface and functional clocks must be enabled.
Control module	Module-specific pad multiplexing must be set in the control module.
MPU INTC	The MPU INTC configuration must be performed to enable the interrupts from the watchdog timer.

22.3.1.2 Main Sequence – Watchdog Timer Module Global Initialization

Table 22-7 lists the steps for initializing the watchdog timer module when the module is to be used for the first time.

Table 22-7. Watchdog Timer Module Global Initialization

Step	Register/Bit Field/Programming Model	Value
Execute software reset.	WDT_WDSC[1] SOFTRESET	1
Wait until reset release?	WDT_WDSC[1] SOFTRESET	0
Enable delay interrupt.	WDT_WIRQENSET[1] ENABLE_DLY	1
Enable overflow interrupt.	WDT_WIRQENSET[0] ENABLE_OVF	1

22.3.2 Operational Mode Configuration

22.3.2.1 Main Sequence – Watchdog Timer Basic Configuration

Table 22-8 lists the steps for the basic configuration of the watchdog timer.

Table 22-8. Watchdog Timer Basic Configuration

Step	Register/Bit Field/Programming Model	Value
Disable the watchdog timer.	See Section 22.3.2.2.	
Set prescaler value.	WDT_WCLR[4:2] PTV	xxx
Enable prescaler.	WDT_WCLR[5] PRE	1
Load delay configuration value.	WDT_WDLY	xxx
Load timer counter value.	WDT_WCRR	xxx
Enable the watchdog timer.	See Section 22.3.2.3.	

22.3.2.2 Subsequence – Disable the Watchdog Timer

Table 22-9 lists the steps to disable the watchdog timer.

Table 22-9. Disable the Watchdog Timer

Step	Register/Bit Field/Programming Model	Value
Write disable sequence Data1.	WDT_WSPR	XXXX AAAAh
Write disable sequence Data2.	WDT_WSPR	XXXX 5555h

22.3.2.3 Subsequence – Enable the Watchdog Timer

Table 22-10 lists the steps to enable the watchdog timer.

Table 22-10. Enable the Watchdog Timer

Step	Register/Bit Field/Programming Model	Value
Write enable sequence Data1.	WDT_WSPR	XXXX BBBBh
Write enable sequence Data2.	WDT_WSPR	XXXX 4444h

22.4 Watchdog Timer Registers

The watchdog timer registers are listed in Table 22-11. For the base address of these registers, see Table 1-12.

CAUTION

The watchdog timers registers are limited to 32-bit and 16-bit data accesses; 8-bit access is not allowed and can corrupt register content.

NOTE:

- The WDT_WISR and WDT_WIRQSTATRAW registers have the same functionality. The WDT_WISR register is used for software backward compatibility.
- The WDT_WIER and WDT_WIRQENSET/WDT_WIRQENCLR registers have the same functionality. The WDT_WIER register is used for software backward compatibility.
- The WDT_WIRQSTATRAW and WDT_WIRQSTAT registers give the same information when read. The WDT_WIRQSTATRAW register is used for debug.

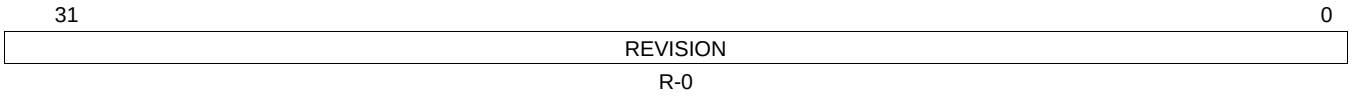
Table 22-11. Watchdog Timer Registers

Offset	Acronym	Register Name	Section
0h	WDT_WIDR	WDT_WIDR	Section 22.4.1
10h	WDT_WDSC	WDT_WDSC	Section 22.4.2
14h	WDT_WDST	WDT_WDST	Section 22.4.3
18h	WDT_WISR	WDT_WISR	Section 22.4.4
1Ch	WDT_WIER	WDT_WIER	Section 22.4.5
24h	WDT_WCLR	WDT_WCLR	Section 22.4.6
28h	WDT_WCRR	WDT_WCRR	Section 22.4.7
2Ch	WDT_WLDR	WDT_WLDR	Section 22.4.8
30h	WDT_WTGR	WDT_WTGR	Section 22.4.9
34h	WDT_WWPS	WDT_WWPS	Section 22.4.10
44h	WDT_WDLY	WDT_WDLY	Section 22.4.11
48h	WDT_WSPR	WDT_WSPR	Section 22.4.12
54h	WDT_WIRQSTATRAW	WDT_WIRQSTATRAW	Section 22.4.13
58h	WDT_WIRQSTAT	WDT_WIRQSTAT	Section 22.4.14
5Ch	WDT_WIRQENSET	WDT_WIRQENSET	Section 22.4.15
60h	WDT_WIRQENCLR	WDT_WIRQENCLR	Section 22.4.16

22.4.1 WDT_WIDR Register

The WDT_WIDR register is shown and described in the figure and table below.

Figure 22-4. WDT_WIDR Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-12. WDT_WIDR Register Field Descriptions

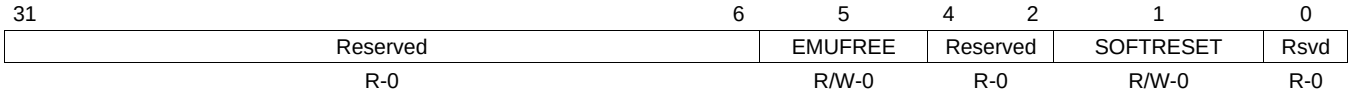
Bit	Field	Type	Reset	Description
31-0	REVISION	R	0	IP Revision

22.4.2 WDT_WDSC Register

The WDT_WDSC register is shown and described in the figure and table below.

This register controls the various parameters of the L4 interface.

Figure 22-5. WDT_WDSC Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-13. WDT_WDSC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	Reserved	R	0	Write 0s for future compatibility. Reads return 0.
5	EMUFREE	R/W	0	Emulation mode 0 = Timer counter frozen in emulation 1 = Timer counter free-running in emulation
4-2	Reserved	R	0	Write 0s for future compatibility. Reads return 0.
1	SOFTRESET	R/W	0	Software reset. (Optional) Read 0 = Reset done, no pending action Write 0 = No action Write 1 = Initiate software reset. Read 1 = Reset (software or other) ongoing
0	Reserved	R	0	Write 0s for future compatibility. Reads return 0.

22.4.3 WDT_WDST Register

The WDT_WDST register provides status information about the module.

It is shown and described in the figure and table below.

Figure 22-6. WDT_WDST Register

31	1	0
Reserved		RESETDONE
R-0		R-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-14. WDT_WDST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0	Reads return 0.
0	RESETDONE	R	1	Internal module reset monitoring Read 0 = Internal module reset is ongoing. Read 1 = Reset completed

22.4.4 WDT_WISR Register

This register shows which interrupt events are pending inside the module.

It is shown and described in the figure and table below.

Figure 22-7. WDT_WISR Register

31	2	1	0
Reserved		DLY_IT_FLAG	OVF_IT_FLAG
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-15. WDT_WISR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0	Reads return 0.
1	DLY_IT_FLAG	R/W	0	Pending delay interrupt status. Read 0 = No delay interrupt pending Write 0 = Status unchanged Write 1 = Status bit cleared Read 1 = Delay interrupt pending
0	OVF_IT_FLAG	R/W	0	Pending overflow interrupt status. Read 0 = No overflow interrupt pending Write 0 = Status unchanged Write 1 = Status bit cleared Read 1 = Overflow interrupt pending

22.4.5 WDT_WIER Register

The WDT_WIERs register controls (enable/disable) the interrupt events.

It is shown and described in the figure and table below.

Figure 22-8. WDT_WIER Register

31	2	1	0
Reserved		DLY_IT_ENA	OVF_IT_ENA
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-16. WDT_WIER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0	Reads return 0.
1	DLY_IT_ENA	R/W	0	Delay interrupt enable/disable 0 = Disable delay interrupt. 1 = Enable delay interrupt.
0	OVF_IT_ENA	R/W	0	Overflow interrupt enable/disable 0 = Disable overflow interrupt. 1 = Enable overflow interrupt.

22.4.6 WDT_WCLR Register

The WDT_WCLR register controls the prescaler stage of the counter.

It is shown and descirbed in the figure and table below.

Figure 22-9. WDT_WCLR Register

31	6	5	4	2	0
Reserved		PRE	PTV		Rsvd
R-0		R/W-1	R/W-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-17. WDT_WCLR Register Field Descriptions

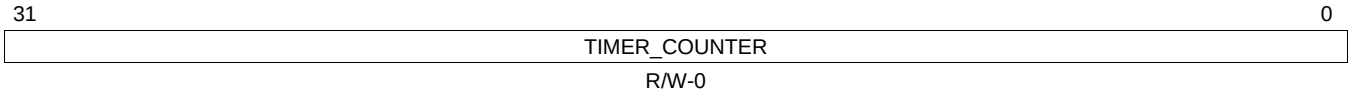
Bit	Field	Type	Reset	Description
31-6	Reserved	R	0	Reads return 0.
5	PRE	R/W	1	Prescaler enable/disable configuration 0 = Prescaler disabled 1 = Prescaler enabled
4-2	PTV	R/W	0	Prescaler value The timer counter is prescaled with the value: 2**PTV. Example: PTV = 3 then counter increases value if started after 8 functional clock periods. On reset, it is loaded from PI_PTV_RESET_VALUE input port.
1-0	Reserved	R	0	Write 0s for future compatibility. Reads return 0.

22.4.7 WDT_WCRR Register

The WDT_WCRR register holds the value of the internal counter.

It is shown and described in the figure and table below.

Figure 22-10. WDT_WCRR Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-18. WDT_WCRR Register Field Descriptions

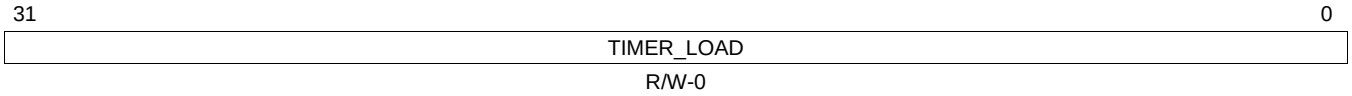
Bit	Field	Type	Reset	Description
31-0	TIMER_COUNTER	R/W	0	Value of the timer counter register

22.4.8 WDT_WLDR Register

The WDT_WLDR register holds the timer load value.

It is shown and described in the figure and table below.

Figure 22-11. WDT_WLDR Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

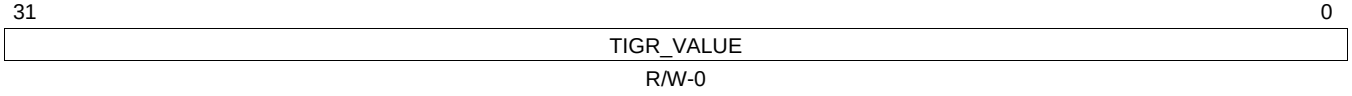
Table 22-19. WDT_WLDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TIMER_LOAD	R/W	0	Value of the timer load register

22.4.9 WDT_WTGR Register

Writing a different value than the one already written in this register does a watchdog counter reload.

Figure 22-12. WDT_WTGR Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-20. WDT_WTGR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TTGR_VALUE	R/W	0	Value of the trigger register

22.4.10 WDT_WWPS Register

This register contains the write posting bits for all writeable functional registers.

Figure 22-13. WDT_WWPS Register

31	16
Reserved	
R-0	

15	6	5	4	3	2	1	0
Reserved		W_PEND_WDLY	W_PEND_WSPR	W_PEND_WTGR	W_PEND_WLDR	W_PEND_WCRR	W_PEND_WCLR
R-0		R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

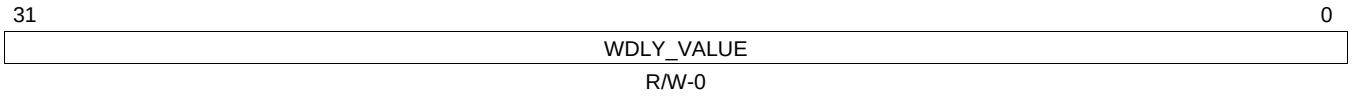
Table 22-21. WDT_WWPS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	Reserved	R	0	Write 0s for future compatibility. Reads return 0.
5	W_PEND_WDLY	R	0	Write pending for register WDLY Read 0 = No register write pending Read 1 = Register write pending
4	W_PEND_WSPR	R	0	Write pending for register WSPR Read 0 = No register write pending Read 1 = Register write pending
3	W_PEND_WTGR	R	0	Write pending for register WTGR Read 0 = No register write pending Read 1 = Register write pending
2	W_PEND_WLDR	R	0	Write pending for register WLDR Read 0 = No register write pending Read 1 = Register write pending
1	W_PEND_WCRR	R	0	Write pending for register WCRR Read 0 = No register write pending Read 1 = Register write pending
0	W_PEND_WCLR	R	0	Write pending for register WCLR Read 0 = No register write pending Read 1 = Register write pending

22.4.11 WDT_WDLY Register

ThE WDT_WDLY register holds the delay value that controls the internal pre-overflow event detection.

Figure 22-14. WDT_WDLY



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

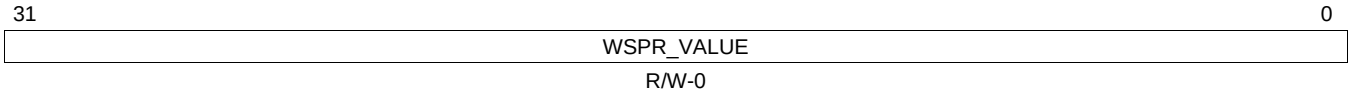
Table 22-22. WDT_WDLY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDLY_VALUE	R/W	0	Value of the delay register

22.4.12 WDT_WSPR Register

The WDT_WSPR register holds the start-stop value that controls the internal start-stop FSM.

Figure 22-15. WDT_WSPR Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-23. WDT_WSPR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WSPR_VALUE	R/W	0	Value of the start-stop register

22.4.13 WDT_WIRQSTATRAW Register

IRQ unmasked status, status set per-event raw interrupt status vector, line 0. Raw status is set even if event is not enabled. Write 1 to set the (raw) status, mostly for debug.

Figure 22-16. WDT_WIRQSTATRAW Register

31	2	1	0
Reserved		EVENT_DLY	EVENT_OVF
R-0		R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-24. WDT_WIRQSTATRAW Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0	Write 0s for future compatibility. Reads return 0.
1	EVENT_DLY	R/W1S	0	Settable raw status for delay event Read 0 = No event pending Write 0 = No action Write 1 = Set event (debug) Read 1 = Event pending
0	EVENT_OVF	R/W1S	0	Settable raw status for overflow event Read 0 = No event pending Write 0 = No action Write 1 = Set event (debug) Read 1 = Event pending

22.4.14 WDT_WIRQSTAT Register

IRQ masked status, status clear per-event enabled interrupt status vector, line 0. Enabled status is not set unless event is enabled. Write 1 to clear the status after interrupt has been serviced (raw status gets cleared, that is, even if not enabled).

Figure 22-17. WDT_WIRQSTAT Register

31	2	1	0
Reserved		EVENT_DLY	EVENT_OVF
R-0		R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-25. WDT_WIRQSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0	Write 0s for future compatibility. Reads return 0.
1	EVENT_DLY	R/W1C	0	Clearable, enabled status for delay event Read 0 = No (enabled) event pending Write 0 = No action Write 1 = Clear (raw) event Read 1 = Event pending
0	EVENT_OVF	R/W1C	0	Clearable, enabled status for overflow event Read 0 = No (enabled) event pending Write 0 = No action Write 1 = Clear (raw) event Read 1 = Event pending

22.4.15 WDT_WIRQENSET Register

IRQ enable set per-event interrupt enable bit vector, line 0. Write 1 to set (enable interrupt). Readout equal to corresponding _CLR register.

Figure 22-18. WDT_WIRQENSET Register

31	2	1	0
Reserved		ENABLE_DLY	ENABLE_OVF
R-0		R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-26. WDT_WIRQENSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0	Write 0s for future compatibility. Reads return 0.
1	ENABLE_DLY	R/W1S	0	Enable for delay event Read 0 = Interrupt disabled (masked) Write 0 = No action Write 1 = Enable interrupt. Read 1 = Interrupt enabled
0	ENABLE_OVF	R/W1S	0	Enable for overflow event Read 0 = Interrupt disabled (masked) Write 0 = No action Write 1 = Enable interrupt. Read 1 = Interrupt enabled

22.4.16 WDT_WIRQENCLR Register

IRQ enable clear per-event interrupt enable bit vector, line 0. Write 1 to clear (disable interrupt). Readout equal to corresponding _SET register.

Figure 22-19. WDT_WIRQENCLR Register

31	2	1	0
Reserved		ENABLE_DLY	ENABLE_OVF
R-0		R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-27. WDT_WIRQENCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0	Write 0s for future compatibility. Reads return 0.
1	ENABLE_DLY	R/W1C	0	Enable for delay event Read 0 = Interrupt disabled (masked) Write 0 = No action Write 1 = Disable interrupt. Read 1 = Interrupt enabled
0	ENABLE_OVF	R/W1C	0	Enable for overflow event Read 0 = Interrupt disabled (masked) Write 0 = No action Write 1 = Disable interrupt. Read 1 = Interrupt enabled

UART/IrDA/CIR Module

This chapter describes the function, operation, and configuration of the universal asynchronous receiver/transmitter (UART)/infrared data association (IrDA)/consumer infrared (CIR) module.

Topic **Page**

23.1 Introduction

23.1.1 Main Features

The main features of each of the UARTs are:

- Selectable UART/IrDA/CIR modes
- 16C750 compatibility
- Dual 64 entry FIFOs for received and transmitted data payload
- Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation
- Programmable sleep mode
- Complete status reporting capabilities in both normal and sleep mode
- Frequency prescaler values from 0 to 16383 to generate the appropriate baud rates
- Internal system clock reference for baud setting (In this document, a 48 MHz reference clock is considered. For more information on reference clock support, see your device-specific data manual)
- Two DMA requests, 1 interrupt request to the system
- Break character detection and generation
- Configurable data format
  - Data bit: 5, 6, 7, or 8 bits
  - Parity bit: Even, odd, none
  - Stop-bit: 1, 1.5, 2 bit(s)
- Flow control: Hardware (RTS/CTS) or software (XON/XOFF)

The UART/IrDA module can operate in six different modes:

- UART 16× mode
- UART 16× mode with autobauding (greater than or equal to 1200 bits/s and less than or equal to 115.2 Kbits/s)
- UART 13× mode
- IrDA SIR mode (less than or equal to 115.2 Kbits/s)
- IrDA MIR mode (0.576 and 1.152 Mbits/s)
- IrDA FIR mode (4 Mbits/s)
- CIR mode (programmable modulation rates specific to remote control applications)

23.1.2 UART/Modem Functions

The UART/Modem has the following functions:

- Baud-rate from 300 bits/s up to 3.6864 Mbits/s (with a 48 MHz reference clock)
- Auto-baud between 1200 bits/s and 115.2 Kbits/s
- Software/Hardware flow control
  - Programmable Xon/Xoff characters
  - Programmable auto-RTS and auto-CTS
- Programmable serial interface characteristics
  - 5, 6, 7, or 8 bit characters
  - Even, odd, mark (always = 1), space (always = 0) or no parity (non parity bit frame) bit generation and detection
  - 1, 1.5, or 2 stop bit generation
- False start bit detection
- Line break generation and detection
- Fully prioritized interrupt system controls
- Internal test and loopback capabilities
- Modem control functions ($\overline{\text{CTS}}$, $\overline{\text{RTS}}$, $\overline{\text{DSR}}$, $\overline{\text{DTR}}$, RIN, and $\overline{\text{DCD}}$)

23.1.3 IrDA Functions

The IrDA has the following functions:

- Slow infrared (SIR 115.2 KBAUD), medium infrared (MIR 0.576 MBAUD) and fast infrared (FIR 4.0 MBAUD) operations (very fast infrared (VFIR) is not supported)
- Framing error, cyclic redundancy check (CRC) error, illegal symbol (FIR), abort pattern (SIR, MIR) detection
- 8-entry status FIFO (with selectable trigger levels) available to monitor frame length and frame errors

23.1.4 CIR Features

The CIR mode uses a variable pulse-width modulation (PWM) technique (based on multiples of a programmable *t* period) to encompass the various formats of infrared encoding for remote-control applications. The CIR logic transmits data packets based on a user-definable frame structure and packet content.

The CIR includes the following key features to provide CIR support for remote control applications:

- Consumer Infrared remote control mode with programmable data encoding
- Transmit and receive modes supported
- Free data format (supports any remote-control private standards)
- Selectable bit rate
- Configurable carrier frequency
- 1/2, 5/12, 1/3, or 1/4 carrier duty cycle

23.2 Architecture

23.2.1 UART Signal Descriptions

Table 23-1 describes the UART interface.

Table 23-1. UART Signal Description

Signal	I/O	Description	Reset
RX	I	Serial data input for all modes.	Unknown
TX	O	Serial data output in UART modes. In other modes, this pin is set to the reset value (inactive state).	1
IRTX	O	Serial data output in IrDA modes (SIR, MIR and FIR). In other modes, this pin is set to the reset value (inactive state).	0
RCTX	O	Serial data output in CIR mode. In other modes, this pin is set to the reset value (inactive state).	0
$\overline{CTS}$	I	Clear to send. Active-low modem status signal. Reading bit 4 of the modem status register checks the condition of $\overline{CTS}$. Reading bit 0 of that register checks a change of state of $\overline{CTS}$ since the last read of the modem status register. $\overline{CTS}$ is used in auto-CTS mode to control the transmitter.	Unknown
$\overline{RTS}$	O	Request to send. When active (low), the module is ready to receive data. Setting modem control register bit 1 activates $\overline{RTS}$. It becomes inactive as a result of a module reset, loop back mode or by clearing the MCR[1]. In auto-RTS mode, it becomes inactive as a result of the receiver threshold logic.	1
SD	O	SD mode is used to configure the transceivers. The SD pin out is an inverted value of ACREG[6].	1
$\overline{DSR}$	I	Data set ready: Active-low modem status signal. Reading bit 5 of the modem status register checks the condition of $\overline{DSR}$. Reading bit 1 of that register checks a change of state of $\overline{DSR}$ since the last read of the modem status register.	Unknown
$\overline{DTR}$	O	Data terminal ready: When active (low), this signal informs the modem that the module is ready to communicate. It is activated by setting bit 0 of the modem control register.	1
$\overline{DCD}$	I	Data carrier detect: Active-low modem status signal. The condition of $\overline{DCD}$ can be checked by reading bit 7 of the modem status register and any change in its state can be detected by reading bit 3 of that register.	Unknown
RIN	I	Ring indicator: Active-low modem status signal. The condition of RIN can be checked by reading bit 6 of the modem status register and any change in its state can be detected by reading bit 2 of that register.	Unknown

NOTE: Not all UART instances on a specific device support the full pinout shown here. Refer to the device data manual for details

23.2.2 UART/IrDA/CIR Mode Selection

To select a mode, set the MODESELECT field in MDR1[2:0] (see Table 23-2).

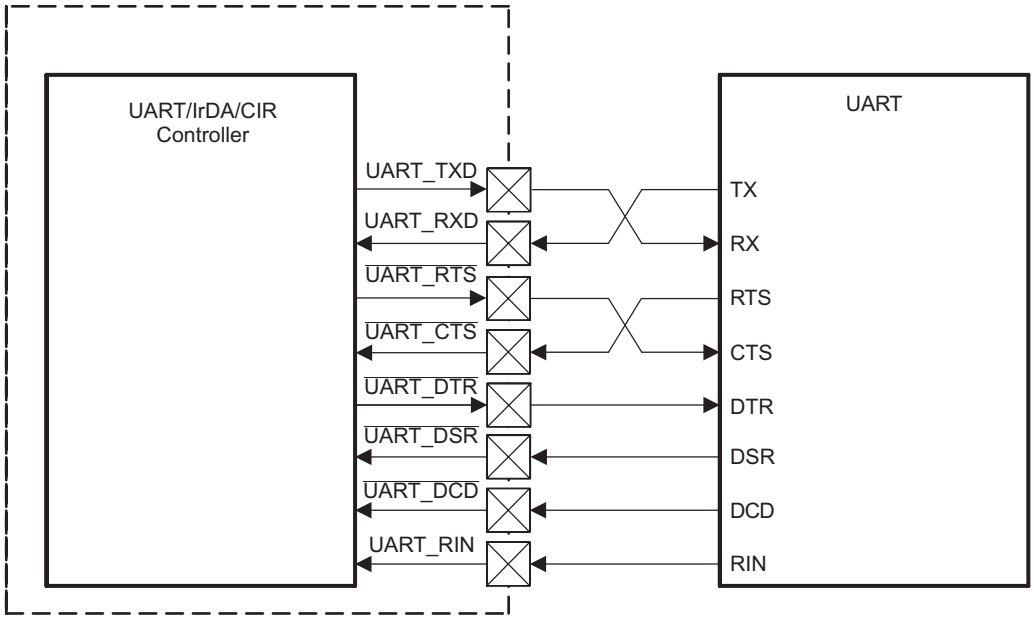
Table 23-2. UART Mode Selection

MDR1[2:0] Value	Mode
0h	UART 16x mode
1h	SIR mode
2h	UART 16x auto-baud
3h	UART 13x mode
4h	MIR mode
5h	FIR mode
6h	CIR mode
7h	Disable (default state)

23.2.3 UART Mode

This section describes how the UART functions in UART wired mode. Figure 23-1 shows how to connect the UART operating in UART mode to another UART on an external device.

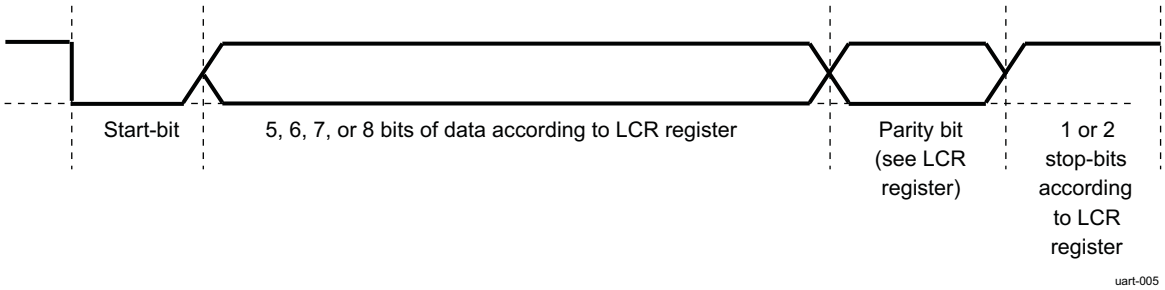
Figure 23-1. UART to UART Connection With Full Handshaking



The UART uses a wired interface for serial communication with a remote device. It can use hardware or software flow control to manage transmission/reception. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using the $\overline{\text{RTS}}$ output and $\overline{\text{CTS}}$ input signals. Software flow control automatically controls data flow by using programmable XON/XOFF characters.

The UART modem module is enhanced with an autobauding functionality, which in control mode allows to automatically set the speed, the number of bit per character, and the parity selected. Figure 23-2 shows the UART frame data format. Each frame consists of a start bit, 5 to 8 data bits (user configurable), an optional parity bit (user configurable), and 1 or 2 stop bits (user configurable).

Figure 23-2. UART Frame Data Format



23.2.3.1 Hardware Flow Control

Hardware flow control is composed of auto-CTS and auto-RTS. Auto-CTS and auto-RTS can be enabled/disabled independently by programming EFR[7:6]. With auto-CTS, $\overline{CTS}$ must be active before the module can transmit data.

Auto-RTS only activates the $\overline{RTS}$ output when there is enough room in the RX FIFO to receive data and de-activates the $\overline{RTS}$ output when the RX FIFO is sufficiently full. The HALT and RESTORE trigger levels in the TCR determine the levels at which $\overline{RTS}$ is activated/de-activated.

If both auto-CTS and auto-RTS are enabled, data transmission does not occur unless the receiver FIFO has empty space. Thus, overrun errors are eliminated during hardware flow control. If not enabled, overrun errors occur if the transmit data rate exceeds the receive FIFO latency.

23.2.3.1.1 Auto-RTS

Auto-RTS data flow control originates in the receiver block (see functional block diagram). The receiver FIFO trigger levels used in auto-RTS are stored in the TCR. $\overline{RTS}$ is active if the RX FIFO level is below the HALT trigger level in TCR[3:0]. When the receiver FIFO HALT trigger level is reached, $\overline{RTS}$ is de-asserted. The sending device (for example, another UART) may send an additional byte after the trigger level is reached because it may not recognize the de-assertion of $\overline{RTS}$ until it has begun sending the additional byte. $\overline{RTS}$ is automatically reasserted once the receiver FIFO reaches the RESUME trigger level programmed via TCR[7:4]. This reassertion requests the sending device to resume transmission.

23.2.3.1.2 Auto-CTS

The transmitter circuitry checks $\overline{CTS}$ before sending the next data byte. When $\overline{CTS}$ is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, $\overline{CTS}$ must be de-asserted before the middle of the last stop bit that is currently being sent. The auto-CTS function reduces interrupts to the host system. When auto-CTS flow control is enabled, the $\overline{CTS}$ state changes need not trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result.

23.2.3.2 Software Flow Control

Software flow control is enabled through the enhanced feature register (EFR) and the modem control register (MCR). Different combinations of software flow control can be enabled by setting different combinations of EFR[3:0]. There are two other enhanced features relating to software flow control:

- XON any function (MCR[5]): operation will resume after receiving any character after recognizing the XOFF character. The XON-any character is written into the RX FIFO even if it is a software flow character.
- Special character (EFR[5]): Incoming data is compared to XOFF2. Detection of the special character sets the XOFF interrupt (IIR[4]) but does not halt transmission. The XOFF interrupt is cleared by a read of the IIR. The special character is transferred to the RX FIFO.

23.2.3.2.1 Receive (RX)

When software flow control operation is enabled, the UART compares incoming data with XOFF1/2 programmed characters (in certain cases XOFF1 and XOFF2 must be received sequentially). When the correct XOFF characters are received, transmission is halted after completing transmission of the current character. XOFF detection also sets IIR[4] (if enabled via IER[5]) and causes the interrupt line to go low. To resume transmission a XON1/2 character must be received (in certain cases XON1 and XON2 must be received sequentially). When the correct XON characters are received IIR[4] is cleared and the XOFF interrupt disappears.

NOTE: If a parity, framing or break error occurs while receiving a software flow control character, this character will be treated as normal data and will be written to the RX FIFO.

When XON-any and special character detect are disabled and software flow control is enabled no valid XON or XOFF characters are written to the RX FIFO. For example, EFR[1:0] = 10, if XON1 and XOFF1 characters are received they do not get written to the RX FIFO. In the case where pairs of software flow characters are programmed to be received sequentially (EFR[1:0] = 11) the software flow characters are not written to the RX FIFO if they are received sequentially. However, received XON1/XOFF1 characters must be written to the RX FIFO if the subsequent character is not XON2/XOFF2.

23.2.3.2.2 Transmit (TX)

- XOFF1: Two characters are transmitted when the RX FIFO has passed the programmed trigger level TCR[3:0].
- XON1: Two characters are transmitted when the RX FIFO reaches the trigger level programmed via TCR[7:4].

NOTE: If after an XOFF character has been sent, software flow control is disabled, the module transmits XON characters automatically to enable normal transmission to proceed.

The transmission of XOFF/XON (s) follows the exact same protocol as transmission of an ordinary byte from the TX FIFO. This means that even if the word length is set to be 5, 6 or 7 characters then the 5, 6 or 7 least significant bits of XOFF1,2/XON1,2 are transmitted. Note that the transmission of 5, 6, or 7 bits of a character is seldom done but this functionality is included to maintain compatibility with earlier designs. It is assumed that software flow control and hardware flow control will never be enabled simultaneously.

23.2.3.3 Break and Time-Out Conditions

23.2.3.3.1 Time-Out Counter

An RX idle condition is detected when the receiver line, RX, has been high for a time equivalent to 4X programmed word length+12 bits. The receiver line is sampled midway through each bit. For sleep mode the counter is reset when there is activity on the RX line. For the timeout interrupt, the counter only counts when there is data in the RX FIFO and the count is reset when there is activity on the RX line or when the RHR is read.

23.2.3.3.2 Break Condition

When a break condition occurs, the TX line is pulled low. A break condition is activated by setting LCR[6]. Be aware that the break condition is not aligned on word stream; that is, a break condition can occur in the middle of a character. The only way to send a break condition on a full character, is:

- Reset transmit FIFO (if enabled)
- Wait for transmit shift register becomes empty (LSR[6] = 1)
- Take a guard time according to stop bit definition
- Set LCR[6] to 1

A break condition is asserted as long as LCR[6] is set to 1. The above functionality (timeout counter and break condition) only applies to the UART Modem operation and does not extend to the IrDA/CIR modes of operation.

NOTE: A break condition is asserted irrespective of CTS state when auto-CTS is programmed.

23.2.3.4 UART Configuration Example

This section proposes outlines the programming stages to operate one UART module with FIFO, interrupt and no DMA capabilities. This is a three-step procedure that ensures quick start of these modules (obviously it does not cover every UART module feature). The first stage covers SW reset of the module (interrupts, status and controls); the second stage deals with FIFO configuration and enable. Finally, last stage deals with the baud rate data and stop configuration. The procedure below is programming language agnostic.

23.2.3.4.1 UART Software Reset

Goal: To clear IER and MCR registers: remove UART breaks (LCR[6] = 0) and put module in reset (MDR1[2:0] = 7h).

Procedure: To write into both the IER and MCR register, EFR[4] must first be set to 1. To be able to access the EFR register, BFh must first be written to LCR register. Hence LCR = BFh; first write to the LCR register EFR[4] = 1. When LCR = BFh, enable the enhanced feature register LCR[7] = 0. Here, access to IER and MCR is allowed. IER = 0; disable interrupt MCR = 0. Force control signals inactive LCR[6] = 0. Here, UART breaks are removed MDR1 = 7h. Here, UART is in reset or disabled. Alternatively, the SYSC[1] can be set to 1 to start a hardware reset from the generic synchronous reset module. The reset progress can be monitored via the SYSS[0]. Once complete the above sequence should ensure the UART is in the equivalent disabled mode with reference to MDR1[2:0].

23.2.3.4.2 UART FIFO Configuration

Goal: To set trigger level for halt/restore (TCR register), set trigger level for transmit/receive (TLR register), and configure FIFO (FCR register).

Procedure: To write into both the TLR and TCR registers, EFR[4] must be set to 1 and MCR[6] to 1. To write into FCR, EFR[4] must be set to 1. Note that EFR[4] = 1 has already been done in the previous section, hence a write to MCR[6] is necessary. MCR[6] = 1; set TCR, TLR, and FCR to the desired value. Here, accesses to TCR, TLR, and FCR must be disabled to avoid any further undesired writes to these registers. Hence, LCR = BFh, provides access to EFR, EFR[4] = 0; LCR[7] = 0; MCR[6] = 0.

23.2.3.4.3 Baud Rate Data and Stop Configuration

Goal: To configure UART data and stop (LCR register). baud rate (DLH and DLL registers), and enable UART operation. In case interrupt capability is added, configuration must be added right before UART enable.

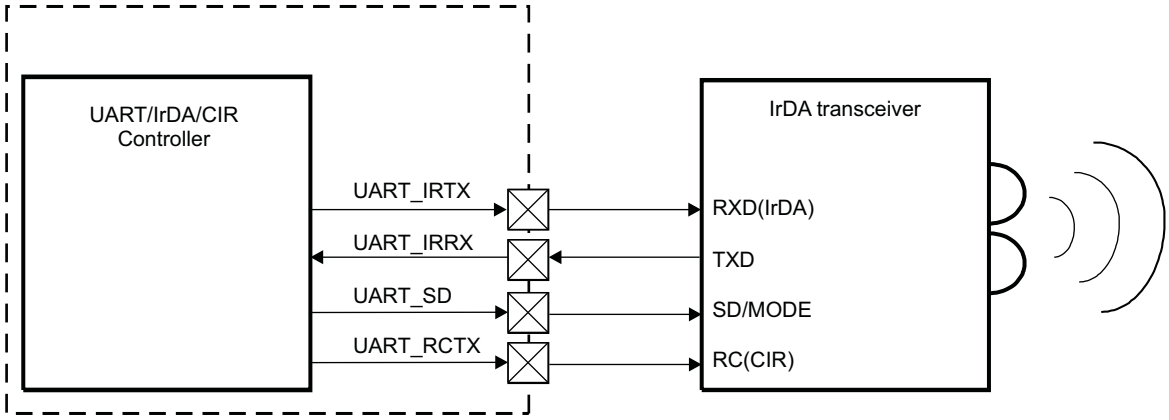
Procedure: Set LCR to desired value; LCR[7] to 1, gives access to DLH and DLL registers. Set DLH and DLL; LCR[7] = 0, removes access to DLH and DLL registers. Set IER to desired value. Set interrupts MDR1[2:0] = 0; enables UART without autobauding.

The UART module is operational.

23.2.4 IrDA Mode

This section describes the IrDA connection with an external device. Figure 23-3 shows how UART3 can be connected to an external infrared transceiver in the IrDA mode.

Figure 23-3. UART IrDA to External IR Device



The module performs serial-to-parallel conversion on received data characters and parallel-to-serial conversion on transmitted data characters by the processor. The complete status of each channel of the module and each received character/frame can be read at any time during functional operation via the line status register (LSR).

The module can be placed in an alternate mode (FIFO mode) relieving the processor of excessive software overhead by buffering received/transmitted characters. Both the receiver and transmitter FIFOs can store up to 64 bytes of data (plus three additional bits of error status per byte for the receiver FIFO) and have selectable trigger levels. Both interrupts and DMA are available to control the data-flow between the LH and the module.

23.2.4.1 SIR Mode

In slow infrared (SIR) mode, data transfer takes place between the LH and peripheral devices at speeds of up to 115200 bauds speed. A SIR transmit frame starts with start flags (either a single C0h, multiple C0h, or a single C0h preceded by a number of FFh flags), followed by frame data, CRC-16 and ends with a stop flag (C1h). The bit format for a single word uses a single start bit, eight data bits and one stop bit and is unaffected by the use and settings of the LCR register.

Note that BLR[6] is used to select whether C0h or FFh start patterns is to be used, when multiple start flags are required. The SIR transmit state machine attaches start flags, CRC-16 and stop flags. It checks the outgoing data to establish if data transparency is required. SIR transparency is carried out if the outgoing data, between the start and stop flags, contains C0h, C1h, or 7Dh. If one of these is about to be transmitted, then the SIR state machine sends an escape character (7Dh) first, then inverts the fifth bit of the real data to be sent, and sends this data immediately after the 7Dh character.

The SIR receive state machine recovers the receive clock, removes the start flags, removes any transparency from the incoming data and determines frame boundary with reception of the stop flag. It also checks for errors such as: frame abort (0x7D character followed immediately by a C1h stop flag, without transparency), CRC error and frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to find out possible errors of the received frame.

NOTE: Data can be transferred both ways by the module, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. See the UART3.ACREG[5] DIS_IR_RX bit description. This applies to all three modes: SIR, MIR, and FIR.

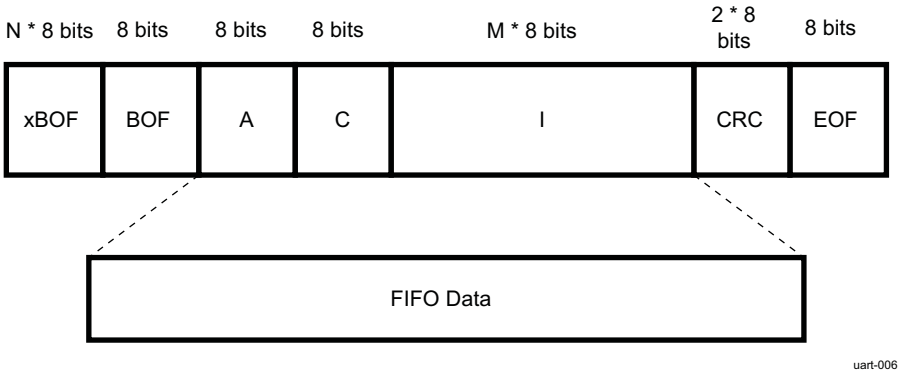
The infrared output in SIR mode can either be 1.6 µs or 3/16 encoding, selected by the PULSETYPE bit of the auxiliary control register (ACREG[7]). In 1.6 µs encoding, the infrared pulse width is 1.6 µs and in 3/16 encoding the infrared pulse width is 3/16 of a bit duration (1/baud-rate). The transmitting device must send at least two start flags at the start of each frame for back-to-back frames.

NOTE: Reception supports variable-length stop-bits.

23.2.4.1.1 Frame Format

Figure 23-4 shows the IrDA SIR frame format.

Figure 23-4. IrDA SIR Frame Format



The CRC is applied on the address (A), control (C), and information (I) bytes.

NOTE: The two words of CRC are written to the FIFO in reception.

23.2.4.1.2 Asynchronous Transparency

Before transmitting a byte, the UART IrDA controller examines each byte of the payload and the CRC field (between BOF and EOF). For each byte equal to C0h (BOF), C1h (EOF), or 7Dh (control escape), the controller performs certain tasks:

- In transmission:
  - Inserts a control escape (CE) byte preceding the byte
  - Complements bit 5 of the byte (that is, exclusive ORs the byte with 20h)

  The byte sent for the CRC computation is the initial byte written in the TX FIFO (before the XOR with 20h).
- In reception:

  For the A, C, I, CRC field:
  - Compares the byte with the CE byte; if they are not equal, sends the byte to the CRC detector and stores it in the RX FIFO
  - If the byte is equal to the CE byte, discards the CE byte
  - Complements bit 5 of the byte following the CE
  - Sends the complemented byte to the CRC detector and stores it in the RX FIFO

23.2.4.1.3 Abort Sequence

The transmitter may decide to prematurely close a frame. The transmitter aborts by sending following sequence: 7DC1h. The abort pattern closes the frame without a CRC field or an ending flag. It is possible to abort a transmission frame by programming the ABORTEN bit of the auxiliary control register (ACREG[1]). When this bit is set to 1, 7Dh and C1h are transmitted and the frame is not terminated with CRC or stop flags. The receiver treats a frame as an aborted frame when a 7Dh character followed immediately by a C1h character have been received without transparency.

23.2.4.1.4 Pulse Shaping

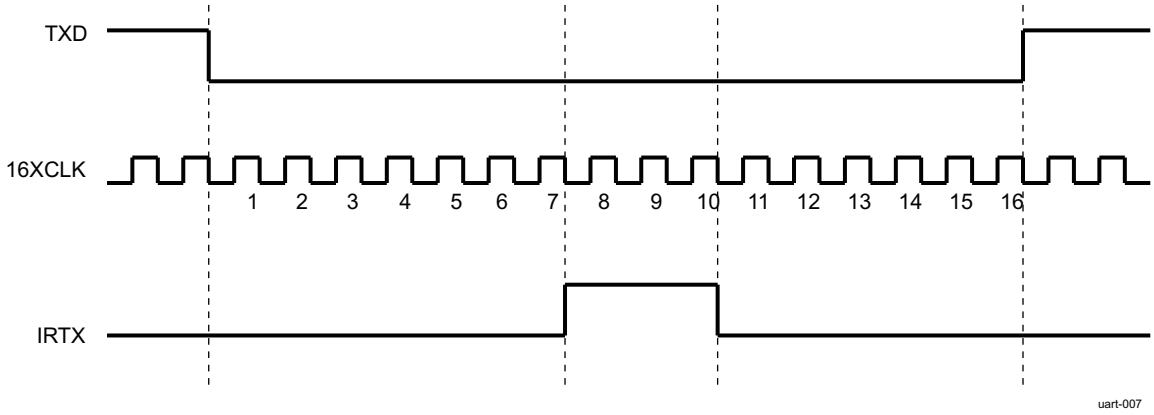
The SIR mode supports both the 3/16th and the 1.6-µs pulse duration methods. The PULSETYPE bit of the auxiliary control register (ACREG[7]) selects the pulse width method in transmit mode.

23.2.4.1.5 Encoder

Serial data from transmit state-machine is encoded to transmit data to the optoelectronics. While the serial data input to the (TXD) is high, the output (IRTX) is always low, and the counter used to form a pulse on IRTX is continuously cleared. After TXD resets to 0, IRTX rises on the falling edge of the 7th 16XCLK. On the falling edge of the 10th 16XCLK pulse, IRTX falls, creating a 3-clock-wide pulse. While TXD stays low, a pulse is transmitted during the 7th to the 10th clock of each 16-clock bit cycle.

Figure 23-5 shows the IrDA SIR encoding mechanism.

Figure 23-5. IrDA SIR Encoding Mechanism

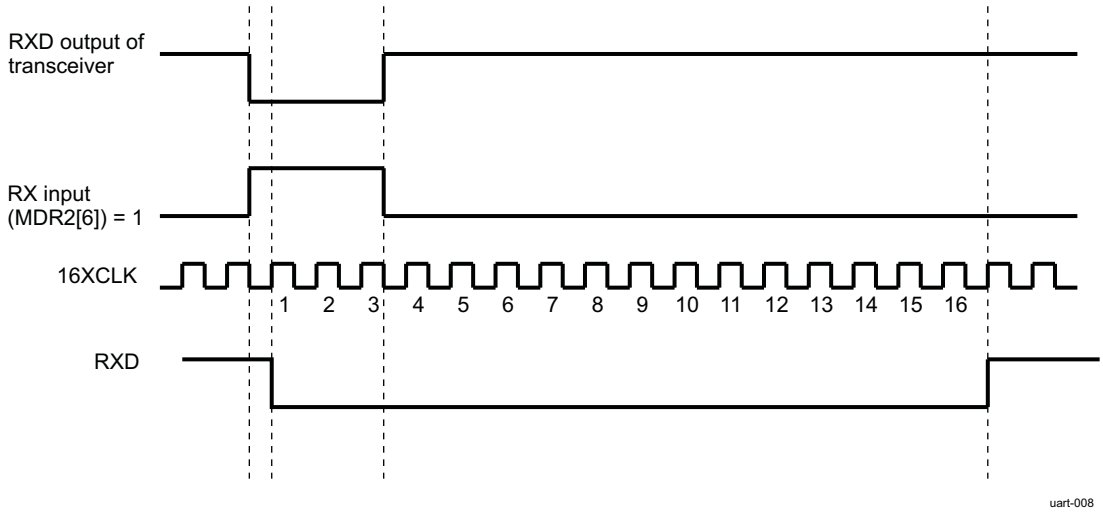


23.2.4.1.6 Decoder

After reset, RXD is high and the 4-bit counter is cleared. When a rising edge is detected on RX, RXD falls on the next rising edge of 16XCLK with sufficient setup time. RXD stays low for 16 cycles (16XCLK) and then returns to high as required by the IrDA specification. As long as no pulses (rising edges) are detected on the RX, RXD remains high.

Figure 23-6 shows the IrDA SIR decoding mechanism.

Figure 23-6. IrDA SIR Decoding Mechanism



The operation of the RX input can be disabled with DISIRRX bit of the auxiliary control register (ACREG[5]). Furthermore, the MDR2[6] can be used to invert the signal from the transceiver (RX output) pin to the IRRX logic inside the UART.

23.2.4.1.7 IR Address Checking

In all IR modes, if address checking is enabled, only frames intended for the device are written to the RX FIFO. This is to avoid receiving frames not meant for this device in a multi-point infrared environment. It is possible to program two frame addresses that the UART IrDA receives with XON1/ADDR1 and XON2/ADDR2 registers.

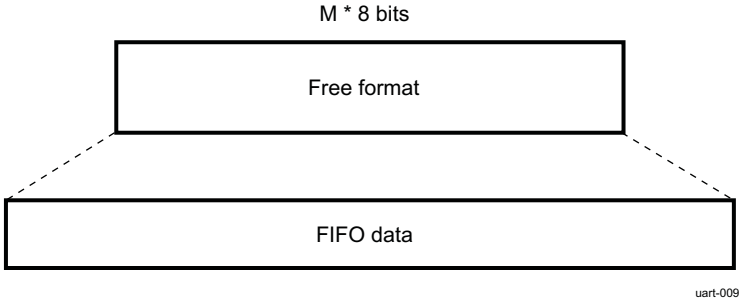
Selecting address1 checking is done by setting EFR[0] to 1; selecting address2 checking is done by setting EFR[1] to 1. Clearing EFR[1:0] to 0 disables all address checking operations. If both bits are set, then the incoming frame is checked for both private and public addresses. If address checking is disabled, then all received frames are written into the reception FIFO.

23.2.4.1.8 SIR Free Format Mode

To allow complete software flexibility in the transmission and reception of Infrared data packets, the SIR Free Format mode is a sub function of the existing SIR mode such that all frames going to and from the FIFO buffers are untouched with respect to appending and removing control characters and CRC values. In transmission phase, it uses the IRTX pin as in SIR mode.

This mode corresponds to a UART mode with a pulse modulation of 3/16 of baud-rate pulse width. For example, a normal SIR packet has BOF control and CRC error checking data appended (transmitting) or removed (receiving) from the data going to and from the FIFOs. In SIR Free Format mode only the data termed the FIFO DATA area as shown in Figure 23-7 (and in Figure 23-4) would be transmitted and received.

Figure 23-7. SIR Free Format Mode



In this mode, the entire FIFO data packet is to be constructed (encoded and decoded) by the LH software.

The SIR Free Format mode is selected by setting the module in UART mode (MDR1[2:0] = 000) and the MDR2[3] register bit to one to allow the pulse shaping.

As the bit format is to remain the same, some UART mode configuration register need to be set at specific value:

- LCR[1:0] = 11 (8 data bits)
- LCR[2] = 0 (1 stop bit)
- LCR[3] = 0 (no parity)
- ACREG[7] = 0 (3/16 of baud-rate pulse width)

The features defined through MDR2[6] and ACREG[5] are also supported.

NOTE: All other configuration registers need to be at the reset value.

The UART mode interrupts are used for the SIR Free Format mode but many of them are not relevant (XOFF, RTS, CTS, Modem status register, …….).

23.2.4.1.9 Programming Models

Example 23-1. Receive IrDA

Receive IrDA frame with parity forced to 1, baud-rate = 112.5Kbs, FIFOs disabled.

- MDR1 = 01h
  - MDR1[2:0] = 001b: SIR mode
- LCR = A8h
  - LCR[7] = 1: access to write DLL and DLH
  - LCR[5:3] = 101: set parity type to forced 1 (default: no parity).
  - optional: LCR[2] = 1: set number of stop bits to 2 (default: 1)
  - optional: LCR[1:0] = 11b: set word length to 8 bits (default: 5)
- DLL = 1Ah: 115.2 Kbs
- DLH = 0h: 115.2 Kbs
- LCR = 28h: LCR[7] = 0 disables access to DLL and DLH and gives access to MCR, FCR, IER, BLR, EBLR, RHR
- optional: IER = 1: enable RHR interrupt

Example 23-2. Transmit IrDA

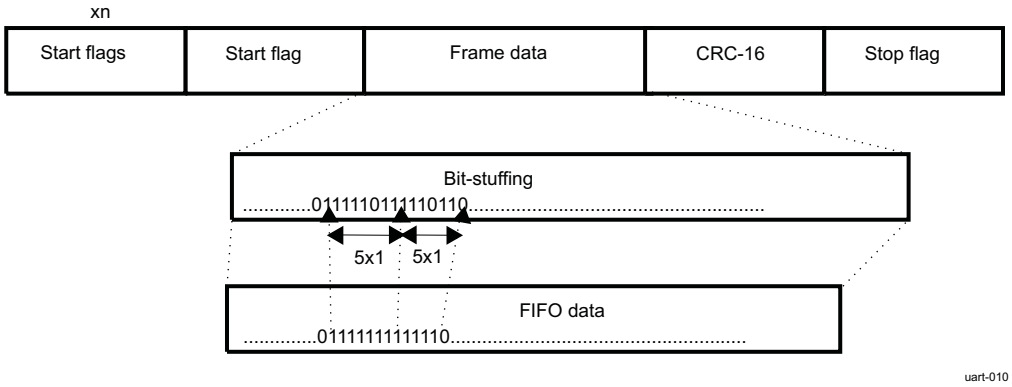
Transmit IrDA 6bytes frame with no parity, baud-rate = 112.5Kbs, FIFOs disabled, 3/16 encoding.

- MDR1 = 41h
  - MDR1[2:0] = 001b: SIR mode
  - MDR1[6] = 1: SIP is generated automatic after each transmission
- LCR = BFh: gives you access to EFR
- EFR = 10h: enhanced functions write enable
- LCR = 86h
  - LCR[7] = 1: access to write DLL and DLH
  - LCR[2] = 1: set number of stop bits to 2 (default: 1)
  - LCR[1:0] = 10b: set word length to 7 bits (default: 5)
- DLL = 1Ah: 115.2 Kbs
- DLH = 0h: 115.2 Kbs
- LCR = 06h: LCR[7] = 0 disables access to DLL and DLH and gives access to MCR, FCR, IER, BLR, EBLR, THR
- MCR =01h: force $\overline{\text{DTR}}$ output to active (low)
- optional: IER = 02h: enable THR interrupt
- TXFLL = 06h: transmit frame length is 6 bytes
- EBLR = 08: transmit 7 starts of frame
- optional: set ACREG[7] = 1: selects SIR pulse width to be 1.6 µs (default 3/16 bit period).
- THR = desired data to be transmitted

23.2.4.2 MIR Mode

In medium infrared (MIR) mode, data transfer takes place between LH and peripheral devices at 0.576 or 1.152 Mbits/s speed. A MIR transmit frame starts with start flags (at least two), followed by a frame data, CRC-16 and ends with a stop flag (see Figure 23-8).

Figure 23-8. MIR Transmit Frame Format



On transmit, the MIR state machine attaches start flags, CRC-16, and stop flags. It also looks for 5 consecutive 1s in the frame data and automatically inserts 0 after 5 consecutive 1s (this is called bit stuffing).

On receive, the MIR receive state machine recovers the receive clock, removes the start flags, de-stuffs the incoming data and determines frame boundary with reception of the stop flag. It also checks for errors such as: frame abort, CRC error or frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to find out possible errors of received frame.

Data can be transferred both ways by the module but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. Refer to the DISIRRX bit of the auxiliary control register (ACREG[5]) for a description of the logical operation.

NOTE: This applies to all three modes SIR, MIR, and FIR.

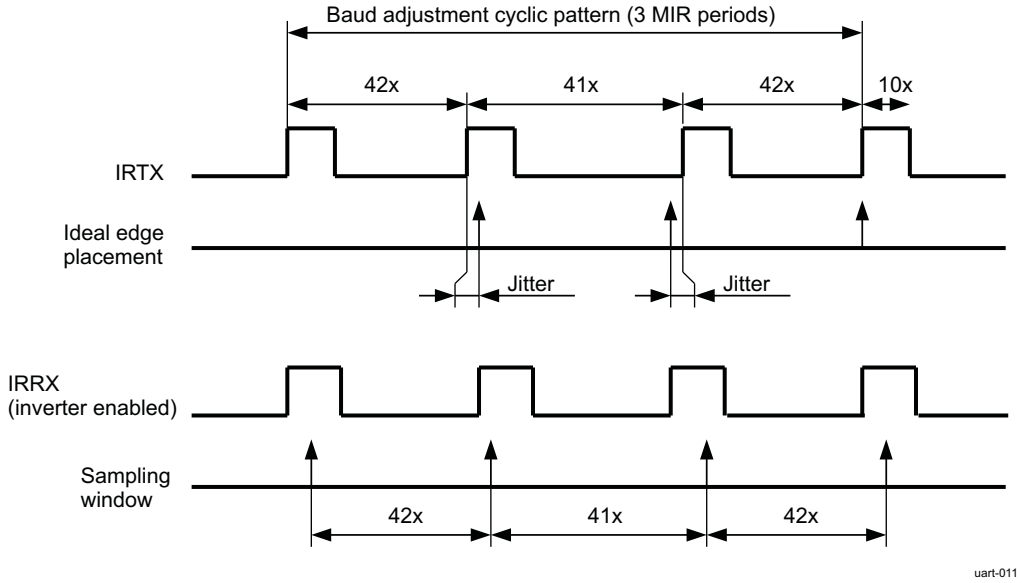
23.2.4.2.1 MIR Encoder/Decoder

To meet the MIR baud rate tolerance of +/–0.1 percent with a 48-MHz clock input, a 42-41-42 encoding/decoding adjustment is performed. The reference start point is the first start flag, and the 42-41-42 cyclic pattern is repeated until the stop flag is sent or detected.

The jitter created this way is within MIR tolerances. The pulse width is not exactly 1/4, but it is within the tolerances defined by the IrDA specifications.

Figure 23-9 shows the MIR baud rate adjustment mechanism.

Figure 23-9. MIR Baud Rate Adjustment Mechanism



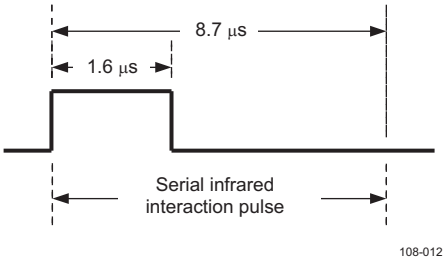
23.2.4.2.2 SIP Generation

In the MIR and FIR operation modes, the transmitter must send a serial infrared interaction pulse (SIP) at least once every 500 ms. The SIP informs slow devices (operating in SIR mode) that the medium is currently occupied.

When the SIPMODE bit of the mode definition register 1 (MDR1[6]) equals 1, the TX state machine always sends one SIP at the end of a transmission frame. But when MDR1[6] = 0, the transmission of the SIP depends on the SENDSIP bit of the auxiliary control register (ACREG[3]). The system (LH) can set ACREG[3] at least once every 500 ms. The advantage of this approach over the default approach is that the TX state machine does not need to send the SIP at the end of each frame which may reduce the overhead required.

Figure 23-10 shows the SIP.

Figure 23-10. Serial Infrared Interaction Pulse (SIP)



23.2.4.2.3 Programming Models

Example 23-3. Receive IrDA

Receive IrDA frame with no parity, baud-rate = 1.152 Mbs, FIFOs disabled.

- MDR1 = 04h
  - MDR1[2:0] = 100b: MIR mode
- LCR = 86h
  - LCR[7] = 1: access to write DLL and DLH
  - LCR[2] = 1: set number of stop bits to 2 (default: 1)
  - LCR[1:0] = 10b: set word length to 7 bits (default: 5)
- DLL = 01h: 1.152 Mbs
- DLH = 0h: 1.152 Mbs
- LCR = 06h: LCR[7] = 0 disables access to DLL and DLH and gives access to MCR, FCR, IER, BLR, EBLR, RHR
- MCR =03h: forces $\overline{\text{DTR}}$ and $\overline{\text{RTS}}$ output to active (low).
- optional: IER = 1: enable RHR interrupt

Example 23-4. Transmit IrDA

Transmit IrDA 60 bytes frame with no parity, baud-rate = 1.152 Mbs, FIFOs disabled.

- MDR1 = 04h
  - MDR1[2:0] = 100b: MIR mode
- LCR = 82h
  - LCR[7] = 1: access to write DLL and DLH
  - LCR[1:0] = 10b: set word length to 7 bits (default: 5)
  - optional: LCR[2] = 1: set number of stop bits to 2 (default: 1)
- DLL = 01h: 1.152 Mbs
- DLH = 0h: 1.152 Mbs
- LCR = 02h: LCR[7] = 0 disables access to DLL and DLH and gives access to MCR, FCR, IER, BLR, EBLR, THR
- MCR =01h: force $\overline{\text{DTR}}$ output to active (low)
- optional: IER = 02h: enable THR interrupt
- TXFLL = 3Ch: transmit frame length is 60 bytes
- optional: EBLR = 08: transmit 8 additional starts of frame (MIR mode requires 2 starts anyway)
- optional: set ACREG[3] = 1: SIP sends at the end of transmission
- THR = desired data to be transmitted

23.2.4.3 FIR Mode

In fast infrared mode (FIR), data transfer takes place between LH and peripheral devices at 4 Mbits/s speed. A FIR transmit frame starts with preamble, followed by a start flag, frame data, CRC-32 and ends with a stop flag.

Table 23-3 shows the FIR transmit frame format.

Table 23-3. FIR Transmit Frame Format

Preamble (16x)	Start flag	Frame data	CRC-32	Stop flag

On transmit, the FIR transmit state machine attaches the preamble, start flag, CRC-32, and stop flag. It also encodes the transmit data into 4PPM format. It also generates the serial infrared interaction pulse (SIP).

On receive, the FIR receive state machine recovers the receive clock, removes start flag, decodes the 4PPM incoming data and determines frame boundary with reception of stop flag. It also checks for errors such as: illegal symbol, CRC error and frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to find out possible errors of the received frame.

Data can be transferred both ways by the module but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. Refer to the DISIRRX bit of the auxiliary control register (ACREG[5]) for a description of the logical operation.

NOTE: This applies to all three modes SIR, MIR, and FIR.

23.2.5 CIR Mode

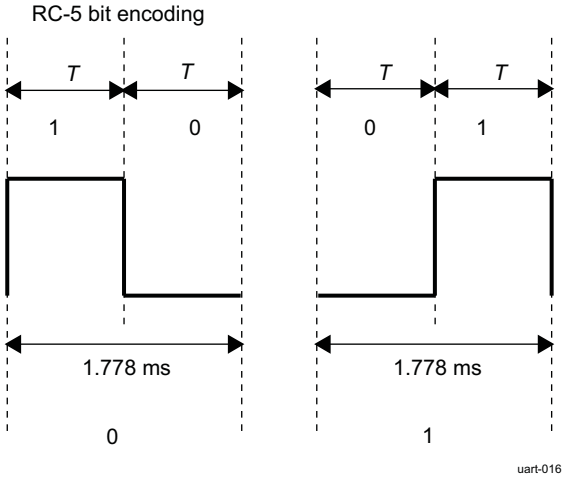
In consumer infrared (CIR) mode, the infrared operation is designed to function as a programmable (universal) remote control. By setting the MDR1 register, the UART can be set to CIR mode in the same way as the other IrDA modes are set using the MDR1 register. The CIR mode uses a variable pulse width modulation technique (based on multiples of a programmable T period) to encompass the various formats of infrared encoding for remote control applications. The CIR logic is to transmit and receive data packets according to the user definable frame structure and packet content.

23.2.5.1 Consumer IR Encoding

There are two distinct methods of encoding for remote control applications. The first uses time extended bit forms, that is, a variable pulse distance (or duration) whereby the difference between a logic 1 and logic 0 is the length of the pulse width; and the second is the use of a bi-phase where the encoding of the logic 0 and logic 1 is in the change of signal level from 1 -> 0 or 0 -> 1, respectively. Japanese manufacturers tend to favor the use of pulse duration encoding whereas European manufacturers favor the use of bi-phase encoding.

The CIR mode is designed to use a completely flexible free format encoding where a digit ‘1’ from the TX/RX FIFO is to be transmitted/received as a modulated pulse with duration T. Equally, a ‘0’ is to be transmitted/received as a blank duration T. The protocol of the data is to be constructed and deciphered by the host CPU. For example, the RC-5 protocol (Figure 23-11) using Manchester encoding can be emulated as using a “01” pair for one and “10” pair for a zero.

Figure 23-11. RC-5 Bit Encoding



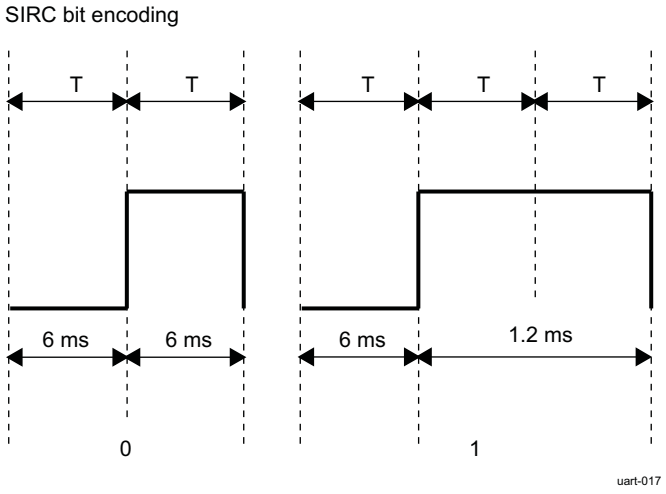
Since the CIR mode logic does not impose a fixed format for infrared packets of data, the CPU software is at liberty to define the format through the use of simple data structures that will then be modulated into an industry standard such as RC-5 or SIRC to name a few. To send a sequence of “0101” in RC-5, the host software must write an eight bit binary character of “10011001” to the data TX FIFO of the UART.

For SIRC, the modulation length (multiples of T) is the method to distinguish between a “1” or a “0”. The following SIRC digits show the difference in encoding between this and RC-5 for example.

NOTE: The pulse width is extended for “1” digits.

Figure 23-12 shows SIRC bit encoding.

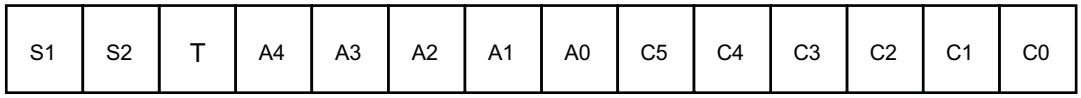
Figure 23-12. SIRC Bit Encoding



To construct comprehensive packets that constitute remote control commands, the host software must combine a number of 8-bit data characters in a sequence that follows one of the universally accepted formats.

Figure 23-13 shows a standard RC-5 frame as detected by the UART3 in CIR mode (the SIRC format follows this). Each field in RC-5 can be considered as two T pulses (digital bits) from the TX and RX FIFOs.

Figure 23-13. RC-5 Standard Packet Format



Where:

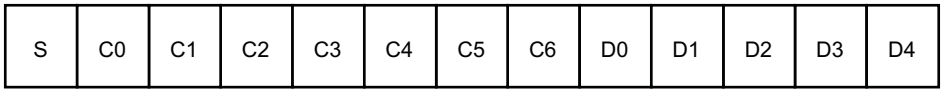
S1, S2:	Start bits (always 1)
T:	Toggle bit
A4..A0:	Address (or system) bits
C5..C0:	Command bits

The toggle bit T changes each time a new command is transmitted to allow detection of pressing the same key twice (or effectively receiving the same data from the host consecutively). Since a code is being sent as long as the CPU transmits characters to the UART for transmission, a brief delay in the transmission of the same command would be detected by the use of the toggle bit. The address bits define the machine or device that the Infrared transmission is intended for and the command defines the operation.

To accommodate an extended RC-5 format, the S2 bit is replaced by an additional command bit (C6) that allows the command range to increase to 7 bits. This format is known as the extended RC-5 format.

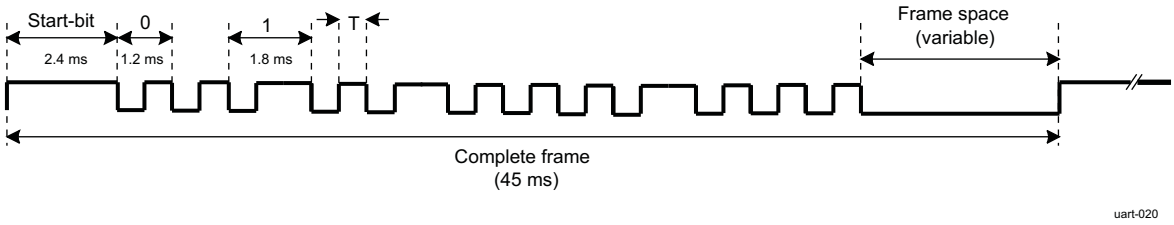
The SIRC encoding uses the duration of modulation for mark and space; hence the duration of data bits inside the standard frame length will vary depending upon the logic 1 content. Figure 23-14 shows the packet format and bit encoding. As Figure 23-14 shows, 1 start bit of 2 ms and control codes are followed by data that constitute the entire frame.

Figure 23-14. SIRC Packet Format



NOTE: The encoding must take a standard duration, but the contents of the data can vary. This implies that the control software for emitting and receiving data packets must exercise a scheme of interpacket delay, where the emission of successive packets can be done only after a real-time delay has expired.

Figure 23-15. SIRC Bit Transmission Example



This document does not describe all encoding methods and techniques; the preceding information shows the considerations required to employ different encoding methods for different industry-standard protocols. See industry-standard documentation for specific methods of encoding and protocol usage.

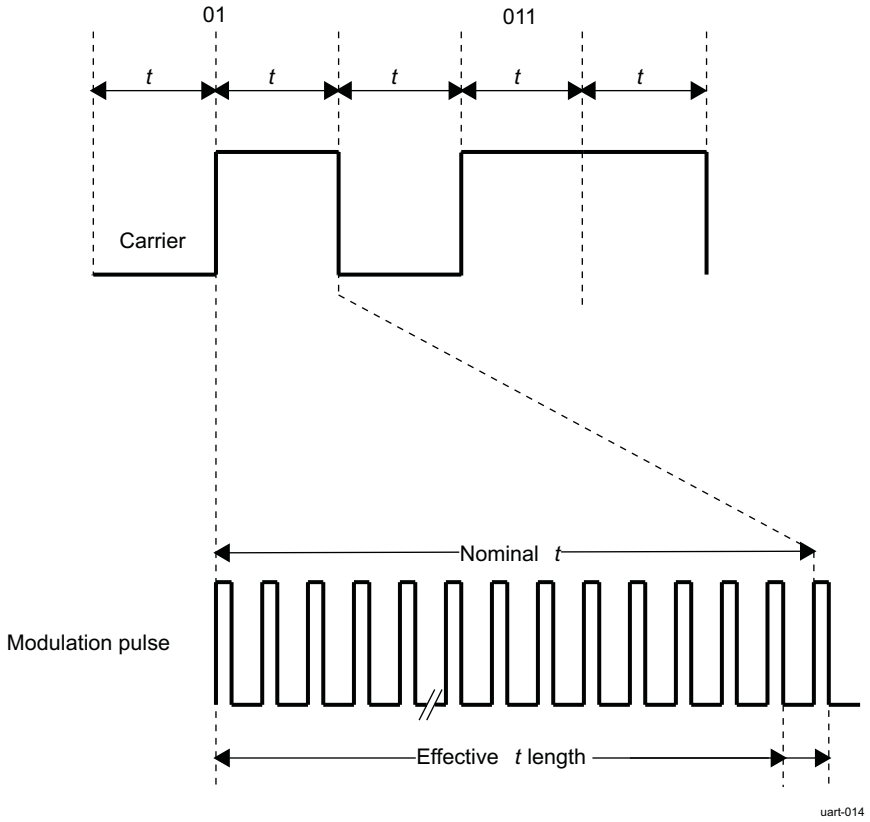
23.2.5.2 CIR Mode Operation

In CIR mode, the infrared operation functions as a programmable (universal) remote control. CIR mode uses a variable PWM technique (based on multiples of a programmable t period) to encompass the various formats of infrared encoding for remote-control applications. The CIR logic transmits data packets based on user-defined frame structure and packet content.

23.2.5.3 Carrier Modulation

Looking closely at the actual modulation pulses of the infrared data stream, each modulated pulse that constitutes a digit is a train of on/off pulses (see Figure 23-16).

Figure 23-16. CIR Pulse Modulation



A minimum of 4 modulation pulses per bit is required by the module. Based on the requested modulation frequency, the CFPS register must be set with the correct dividing value to provide the more accurate pulse frequency:

Dividing value = (FCLK/12)/MODfreq

Where:

FCLK = System clock frequency (48 MHz)
12 = real value of BAUD multiple
MODfreq = Effective frequency of the modulation (MHz)

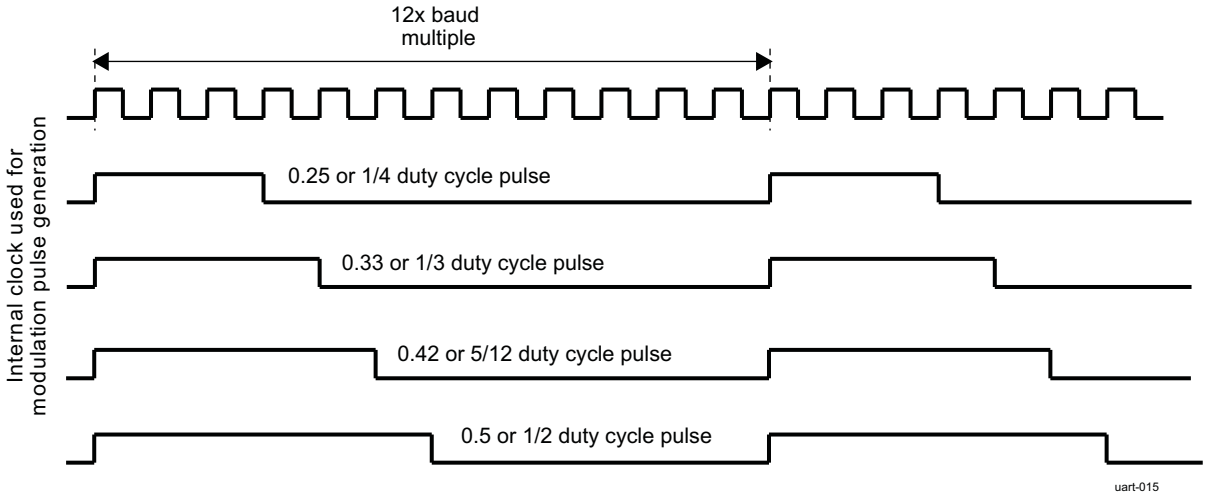
For example, for a targeted modulation frequency of 36 kHz, the CFPS value must be set to 111 in decimal, which provides a modulation frequency of 36.04 kHz.

NOTE: The CFPS register starts with a reset value of 105 (decimal), which translates to a frequency of 38.1 kHz.

The duty cycle of these pulses is user-defined by the CIRPULSEMODE field in the mode definition register 2 (MDR2[5:4]).

Figure 23-17 shows the CIR modulation duty cycles.

Figure 23-17. CIR Modulation Duty Cycle



The transmission logic ensures that all pulses are transmitted completely (that is, no cutoff during transmission). Furthermore, while transmitting continuous bytes back-to-back, no delay is inserted between 2 transmitted bytes.

NOTE: The CIR RX demodulation can be bypassed by setting the MDR3[0] register bit. This bit will not affect the transmission modulation.

23.2.5.4 Frequency Divider Values

As previously explained, the data transferred is a succession of pulses with a T period. Depending on the standards used, the T period is defined through the DLL and DLH registers, which defined the value to divide the functional clock (48 MHz):

Dividing value = (FCLK/16)/Tfreq

Where:

FCLK =	System clock frequency (48 MHz)
16 =	real value of BAUD multiple
Tfreq =	Effective frequency of the T pulse (MHz)

23.2.6 FIFO Management

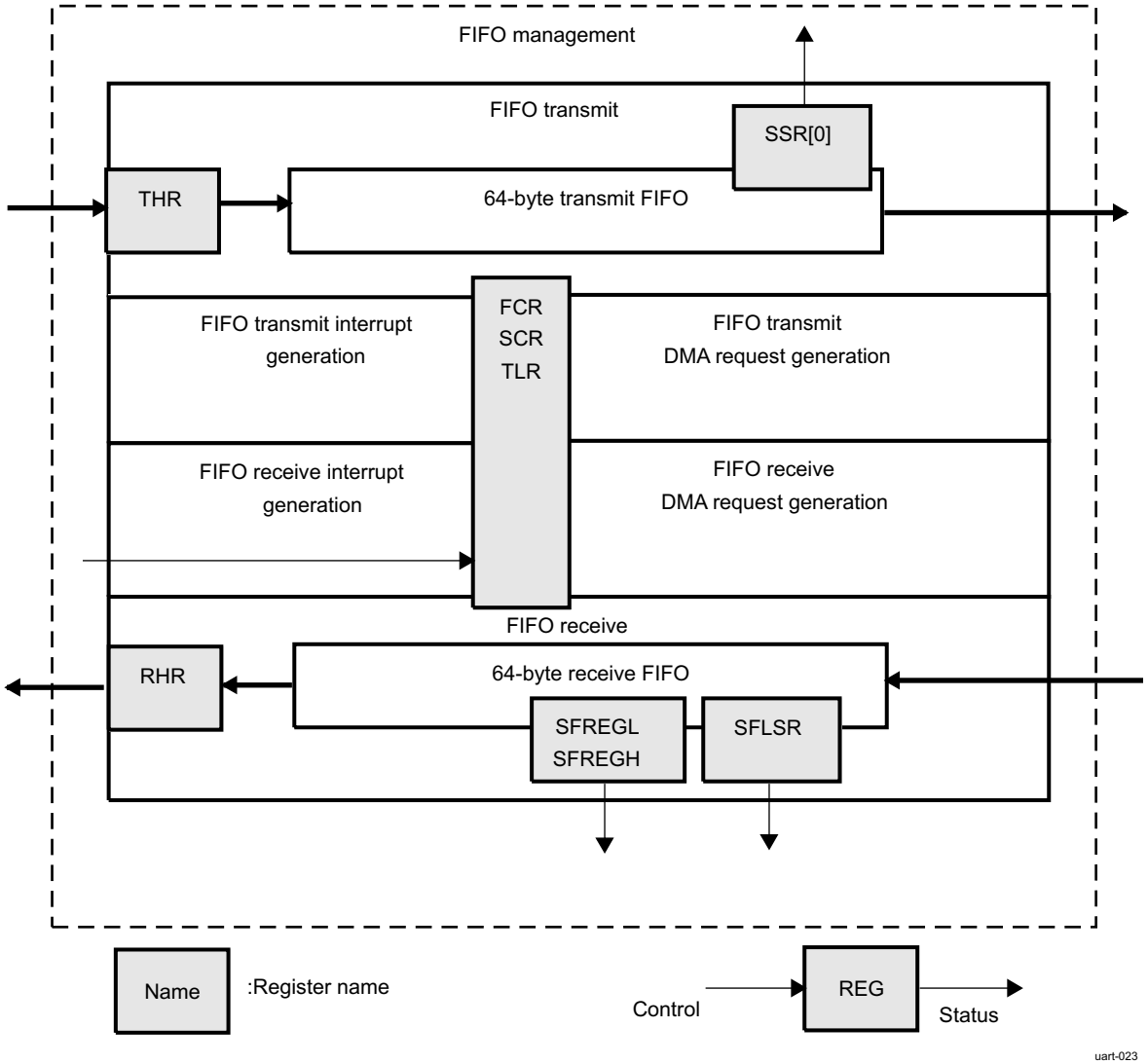
The FIFO is accessed by reading/writing the RHR and THR registers. Parameters are controlled using the FIFO control register (FCR) and supplementary control register (SCR). Reading the TXFIFOFULL bit in SSR[0] as 1 means the FIFO is full.

The TLR register controls the FIFO trigger level, which enables the DMA and interrupt generation. After reset, both transmitter and receiver FIFOs are disabled; so, in effect, the trigger level is the default value of 1 byte. Figure 23-18 shows the FIFO management registers.

NOTE: Data in the RHR register is not overwritten when an overflow occurs.

The SFLSR, SFREGL, and SFREGH status registers are used in IrDA mode only.

Figure 23-18. FIFO Management

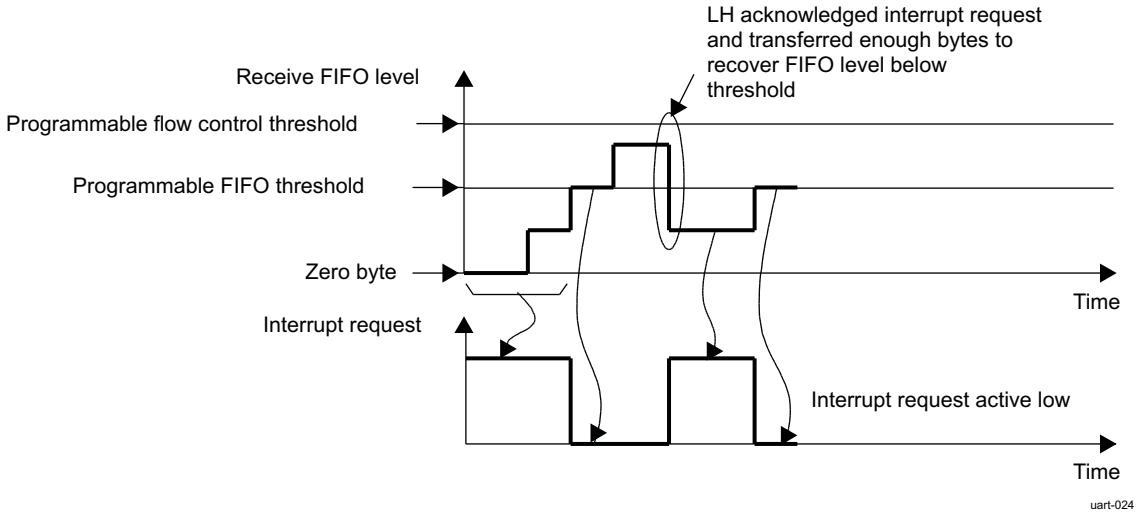


23.2.6.1 FIFO Interrupt Mode

In the FIFO interrupt mode (FIFO_EN bit in the FIFO control register (FCR[0]) is set to 1 and relevant interrupts are enabled via the interrupt enable register (IER)), an interrupt signal informs the processor of the status of the receiver and transmitter. These interrupts are raised when the receive/transmit FIFO threshold (TLR[7:4] and TLR[3:0] fields or the FCR[7:6] and FCR[5:4] fields, respectively) are reached; the interrupt signals instruct the Local Host to transfer data to the destination (from the UART module in receive mode and/or from any source to the UART FIFO in transmit mode). Note also that in the case of the UART flow control being enabled along with the interrupt capabilities, the user must ensure that the UART flow control FIFO threshold (TCR[3:0]) is greater than or equal to the receive FIFO threshold.

Figure 23-19 shows the generation of the receive FIFO interrupt request.

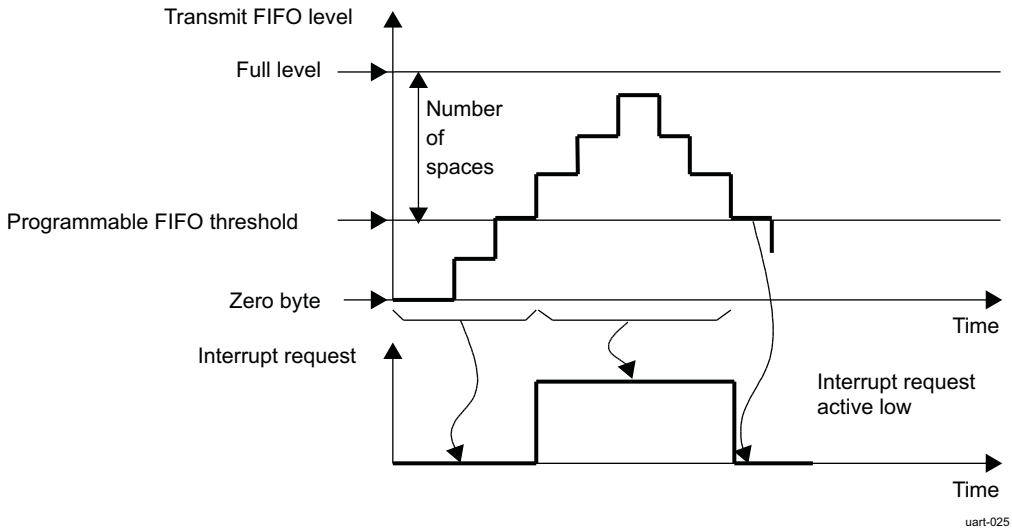
Figure 23-19. Receive FIFO Interrupt Request Generation



In receive mode, no interrupt is generated until the receive FIFO reaches its threshold. Once low, the interrupt can be de-asserted only when the Local Host has handled enough bytes to make the FIFO level below threshold. The flow control threshold is set at a higher value than the FIFO threshold.

Figure 23-20 shows the generation of the transmit FIFO interrupt request.

Figure 23-20. Transmit FIFO Interrupt Request Generation



In transmit mode, an interrupt request is automatically asserted when the TX FIFO is empty. This request is de-asserted when the TX FIFO crosses the threshold level. The interrupt line is de-asserted until a sufficient number of elements is transmitted to go below the TX FIFO threshold.

23.2.6.2 FIFO Polled Mode Operation

In FIFO polled mode (FCR[0] = 0, relevant interrupts disabled via interrupt enable register (IER)) the status of the receiver and transmitter can then be checked by polling the line status register (LSR). This mode is an alternative to the FIFO interrupt mode of operation where the status of the receiver and transmitter is automatically known by means of interrupts sent to the LH.

23.2.6.3 FIFO DMA Mode Operation

23.2.6.3.1 DMA Signaling

There are four modes of DMA operation, DMA mode 0/1/2/3. They can be selected as follows:

- When SCR[0] = 0: setting FCR[3] to 0 enables DMA mode 0, setting FCR[3] to 1 enables DMA mode 1.
- When SCR[0] = 1: SCR[2:1] determines DMA mode 0 to 3, according to supplementary control register (SCR).

So for instance:

- if no DMA operation is desired: set SCR[0] to 1 and SCR[2:1] to 00 (FCR[3] is discarded)
- if DMA mode 1 is desired: either set SCR[0] to 0 and FCR[3] to 1 or set SCR[0] to 1 and SCR[2:1] to 01 (FCR[3] is discarded)

If the FIFOs are disabled (FCR[0] = 0), DMA occurs in single character transfers.

NOTE: Note that when DMA mode 0 has been programmed, the signals associated with DMA operation are not active.

23.2.6.3.2 DMA Transfers (DMA Mode 1, 2, or 3)

Figure 23-21, Figure 23-22, Figure 23-23, and Figure 23-24 show the supported DMA operations.

In receive mode, a DMA request is generated as soon as the receive FIFO reaches its threshold level defined in the trigger level register (TLR). This request is de-asserted when the number of bytes defined by the threshold level has been read by the system DMA.

In transmit mode, a DMA request is automatically asserted when the transmit FIFO is empty. This request is de-asserted when the number of bytes defined by the number of spaces in the trigger level register (TLR) has been written by the system DMA. If an insufficient number of characters are written, then the DMA request will remain active.

The DMA request is again asserted if the FIFO can receive the number of bytes defined by the TLR register.

There are a number of ways the threshold can be programmed. The example in Figure 23-22 is of a DMA transfer that operates with a space setting of 56, which could arise from the use of the auto settings in the FCR[5:4] or the use of the TLR[3:0] concatenated with the FCR[5:4]. The setting of 56 spaces in the UART/IrDA/CIR module should correlate with settings of the system DMA so that the buffer does not overflow (program the DMA request size of the Local Host controller to be equal to the number of spaces value in the UART/IrDA/CIR module).

Figure 23-21. Receive FIFO DMA Request Generation (32 Characters)

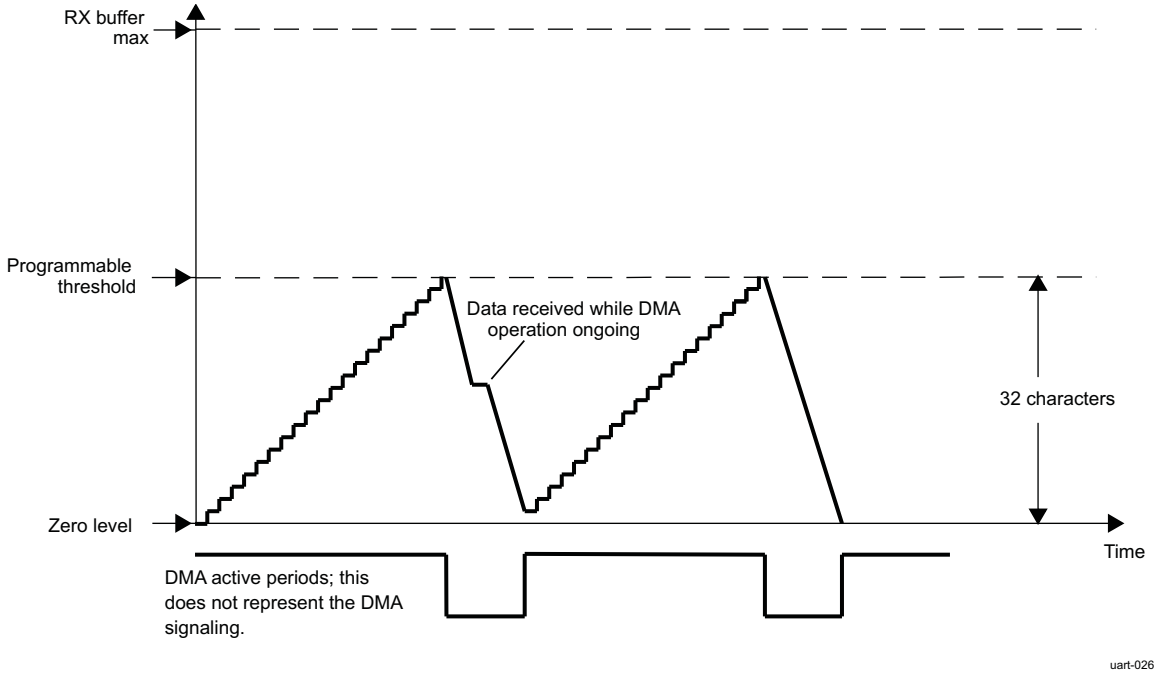


Figure 23-22. Transmit FIFO DMA Request Generation (56 Spaces)

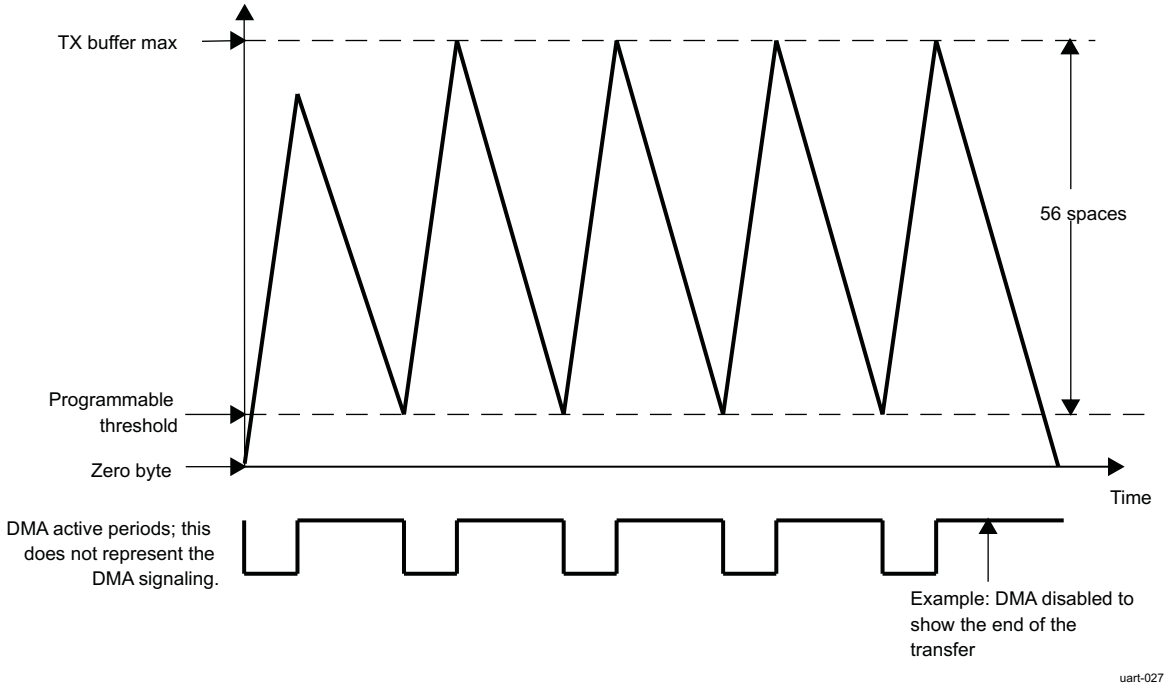
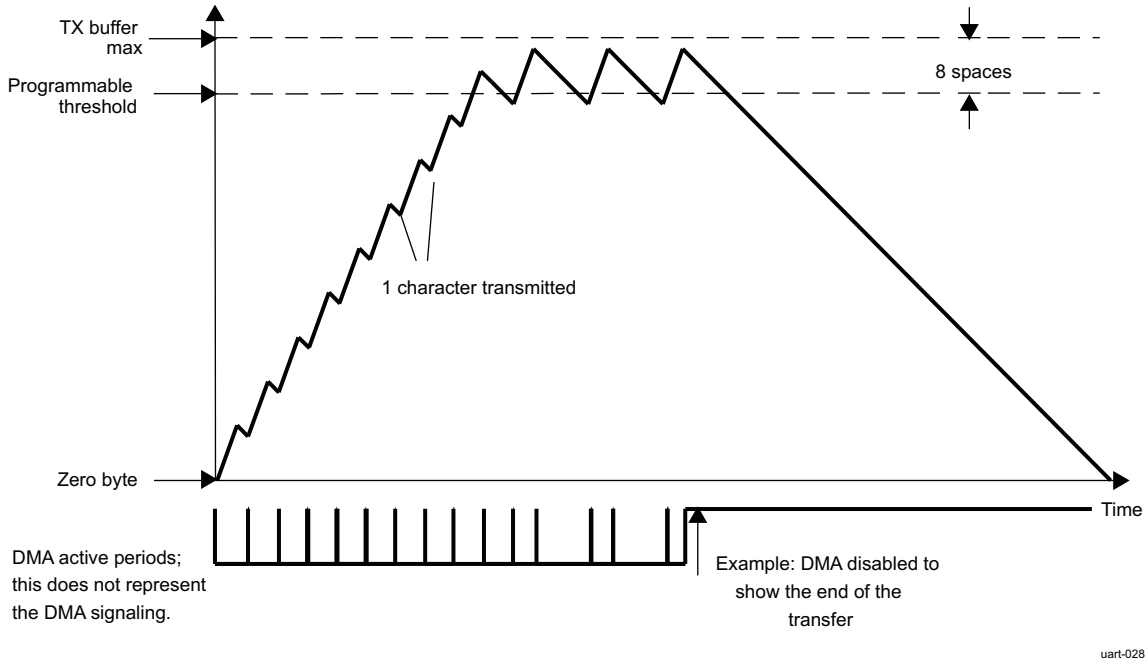


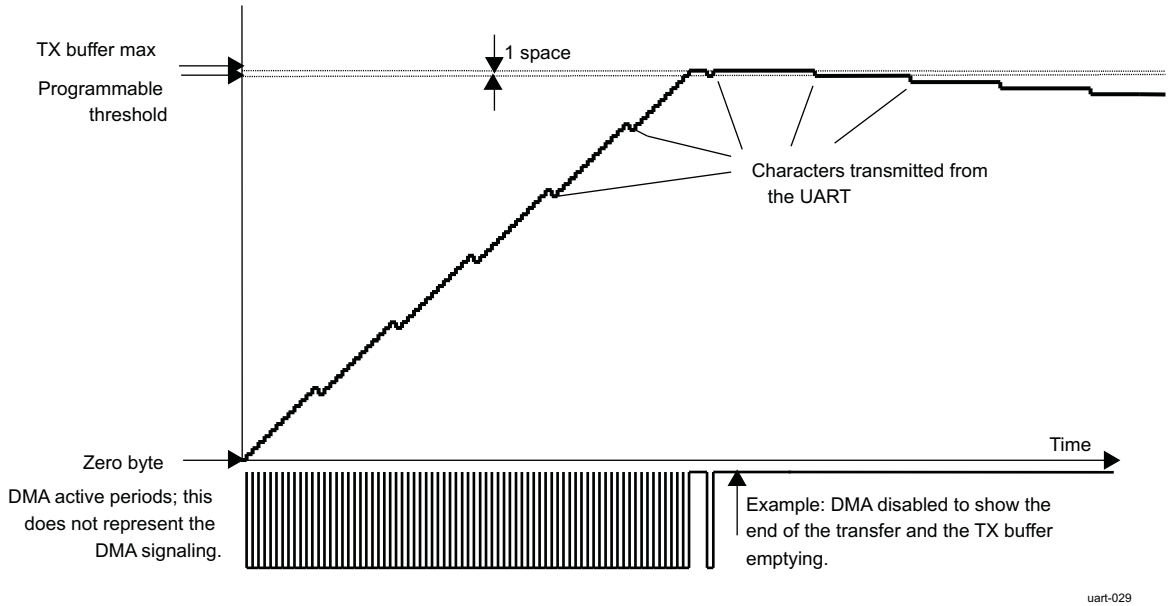
Figure 23-23 shows an example with 8 spaces to show the buffer level crossing the space threshold. Again, the Local Host DMA controller settings must correspond to that of the UART/IrDA/CIR module.

Figure 23-23. Transmit FIFO DMA Request Generation (8 Spaces)



The final example illustrates the setting of one space which uses the DMA for each transfer of one character to the transmit buffer (see Figure 23-24). The buffer is filled at a faster rate than the BAUD rate transmits data to the TX pin. Eventually, the buffer is completely full and the DMA operations stop transferring data to the transmit buffer. There are two occasions where the buffer holds the maximum amount of data words, shortly after this, the DMA is disabled to illustrate the slower transmission of the data words to the TX pin. Eventually, the buffer will be emptied at the rate specified by the BAUD Rate Settings of the DLL and DLH registers. Again, the DMA settings should correspond to the system's Local Host DMA controller settings in order to ensure the correct operation of this logic.

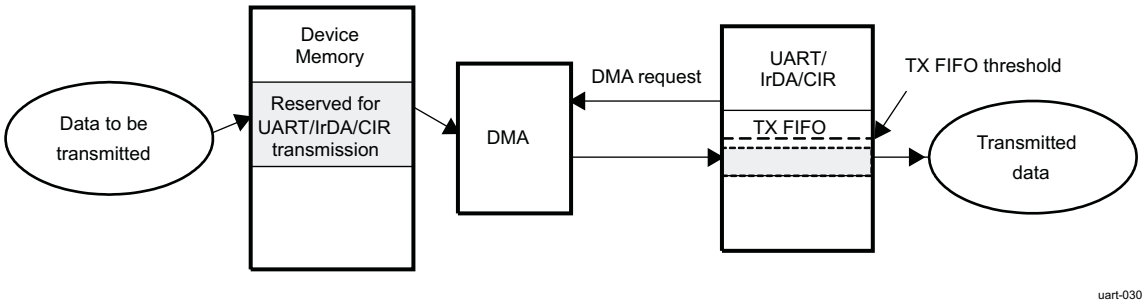
Figure 23-24. Transmit FIFO DMA Request Generation (1 Space)



23.2.6.3.3 DMA Transmission

Figure 23-25 shows the DMA transmission process.

Figure 23-25. Transmission Process



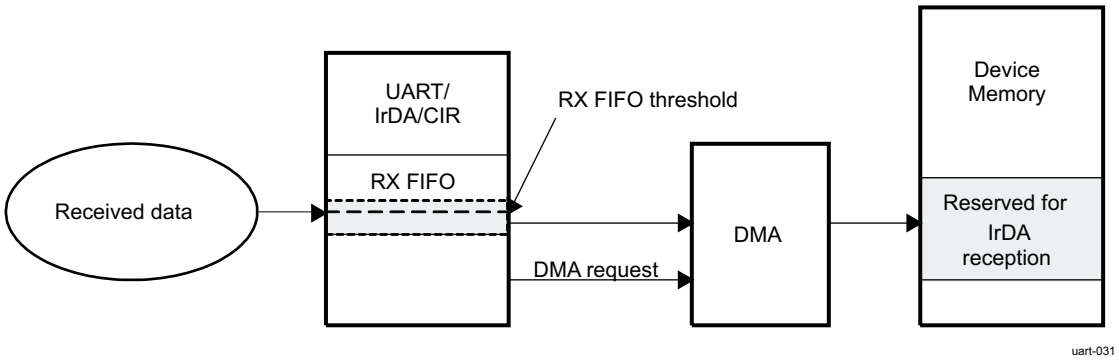
- Data to be transmitted are put in the memory reserved for UART/IrDA/CIR transmission by the DMA:
  - Until the TX FIFO trigger level is not reached, a DMA request is generated.
  - An element (1 byte) is transferred from the SDRAM to the TX FIFO at each DMA request (DMA element synchronization).
- Data in the TX FIFO are automatically transmitted.
- The end of the transmission is signaled by the UARTi.THR empty (TX FIFO empty).

NOTE: In IrDA mode, the transmission is not ended immediately after the TX FIFO empties, at which point there are still the last data byte, the CRC field, and the stop flag to be transmitted; thus, the end of transmission is a few milliseconds after the UARTi.THR register empties.

23.2.6.3.4 DMA Reception

Figure 23-26 shows the DMA reception process.

Figure 23-26. Reception Process



1. Enable the reception.
2. Received data are put in the RX FIFO.
3. Data are transferred from the RX FIFO to the device memory by the DMA.
   - At each received byte, the RX FIFO trigger level (one character) is reached and a DMA request is generated.
   - An element (1 byte) is transferred from the RX FIFO to the SDRAM at each DMA request (DMA element synchronization).
4. The end of the reception is signaled by the EOF interrupt.

23.2.6.4 IrDA Status FIFO

In IrDA modes, a status FIFO is used to record the received frame status. When a complete frame is received, the length of the frame and the error bits associated with the frame are written into the status FIFO. The frame length and error status can be read by reading SFREGL/H and SFLSR. Reading the SFLSR causes the read pointer to be incremented. The status FIFO is eight entries deep and therefore can hold the status of eight frames. The LH uses the frame-length information to locate the frame-boundary in the received frame data. The LH can screen bad frames using the error-status information and later request the sender to resend only the bad frames. This status FIFO can be used very effectively in DMA as the LH need not be interrupted every time a frame is received but only whenever the programmed status FIFO trigger level is reached.

23.2.6.5 Frame Closing

There are two ways by which a transmission-frame can be properly terminated:

- Frame-length method: Frame-length method is selected when MDR1[7] = 0. The LH writes the frame-length value to TXFLH and TXFLL registers. The device automatically attaches end flags to the frame once the number of bytes transmitted becomes equal to the frame-length value.
- Set-EOT bit Method: Set-EOT bit method is selected when MDR1[7] = 1. The LH writes 1 to ACREG[0] (EOT bit) just before it writes the last byte to the TX FIFO. When the LH writes the last byte to the TX FIFO, the device internally sets the tag bit for that particular character in the TX FIFO. As the TX state machine reads data from the TX FIFO it uses this tag-bit information to attach end flags and properly terminate the frame.

23.2.6.6 Store and Controlled Transmission (SCT)

In SCT the LH first starts writing data into the TX FIFO. Then, after it writes a part of a frame (for a bigger frame) or a whole frame (a small frame, that is, supervisory frame), it writes a 1 to ACREG[2] (deferred TX start) to start transmission. SCT is enabled when MDR1[5] = 1. This method of transmission is different from the normal mode, where transmission of data starts immediately after data is written to the TX FIFO. SCT is useful to send short frames without TX underrun.

23.2.6.7 Underrun During Transmission

Underrun in transmission occurs when the TX FIFO becomes empty before the end of the frame is transmitted. When underrun occurs, the device closes the frame with end-flags but attaches an incorrect CRC value. The receiving device detects a CRC error and discards the frame; it can then ask for a re-transmission. Underrun also causes an internal flag to be set which disables further transmission. Before the next frame can be transmitted the system (LH) must:

- Reset the TX FIFO
- Read the RESUME register. This clears the internal flag.

This functionality can be disabled with ACREG[4], compensated by the extension of the stop bit in transmission in case the TX FIFO is empty.

23.2.6.8 Overrun During Receive

Overrun occurs during receive if the RX state machine tries to write data into the RX FIFO when it is already full. When overrun occurs, the device interrupts the LH with IIR[3] and discards the remaining portion of the frame. Overrun also causes an internal flag to be set, which disables further reception. Before the next frame can be received the system (LH) must:

- Reset the RX FIFO
- Read the RESUME register. This clears the internal flag.

23.2.7 Interrupts

The UART/IrDA/CIR module generates interrupts. All interrupts can be enabled/disabled by writing to the appropriate bit in the interrupt enable register (IER). The interrupt status of the device can be checked at any time by reading the interrupt identification register (IIR). The UART, IrDA and CIR modes have different interrupts in the UART/IrDA/CIR module and therefore different IER and IIR mappings according to the selected mode.

23.2.7.1 Trigger Levels

The UART provides programmable trigger levels for both receiver and transmitter DMA and Interrupt generation. After reset, both transmitter and receiver FIFOs are disabled; so, in effect, the trigger level is the default value of one byte. The programmable trigger levels are an enhanced feature available via the trigger level register (TLR).

23.2.7.2 UART Mode Interrupts

In UART modes, there are seven possible interrupts. These interrupts are prioritized to six different levels. When an interrupt is generated, the interrupt identification register (IIR) indicates that an interrupt is pending by bringing IIR[0] to 0 and provides the type of interrupt through IIR[5-1]. Table 23-4 summarizes the interrupt control functions.

It is important to note that for the receiver line status interrupt, RX_FIFO_STS bit (LSR[7]) generates the interrupt. For the XOFF interrupt, if a XOFF flow character detection caused the interrupt, the interrupt is cleared by a XON flow character detection. If special character detection caused the interrupt, the interrupt is cleared by a read of the IIR.

Table 23-4. UART Mode Interrupts

IIR[5:0]	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Method
00 0001	None	None	None	None
00 0110	1	Receiver line status	OE, FE, PE, or BI errors occur in characters in the RX FIFO.	FE, PE, BI: Read RHR register. OE: Read LSR register.
00 1100	2	RX time-out	Stale data in RX FIFO	Read RHR register.
00 0100	2	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read RHR register until interrupt condition disappears.
00 0010	3	THR interrupt	TFE (THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR register until interrupt condition disappears.
00 0000	4	Modem status	See the MSR register.	Read MSR register.
01 0000	5	XOFF interrupt/special character interrupt	Receive XOFF characters/special character	Receive XON character(s), if XOFF interrupt. Read IIR register, if special character interrupt.
10 0000	6	CTS, RTS, DSR	$\overline{\text{RTS}}$ pin, $\overline{\text{CTS}}$ pin, or $\overline{\text{DSR}}$ pin changes state from active (low) to inactive (high).	Read IIR register.

23.2.7.3 IrDA Mode Interrupts

In IrDA modes, there are eight possible interrupts. The interrupt line is activated when any of the eight interrupts is generated (there is no priority). For IIR[5], interrupt source 1 is used with interrupt reset method 1 and interrupt source 2 is used with interrupt reset method 2. Table 23-5 summarizes the interrupt control functions.

Table 23-5. IrDA Mode Interrupts

IIR Bit	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read RHR register until interrupt condition disappears.
1	THR interrupt	TFE (THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR register until interrupt condition disappears.
2	Last byte in RX FIFO	Last byte of frame in RX FIFO is available to be read at the RHR port.	Read RHR register.
3	RX overrun	Write to RHR register when RX FIFO full.	Read RESUME register.
4	Status FIFO interrupt	Status FIFO triggers level reached.	Read SFREGL/H, SFLSR. Status FIFO read pointer is incremented only when reading SFLSR.
5	TX status (indicated by MDR2[0] (IRTX_UNDERRUN)	1. THR empty before EOF sent. Last bit of transmission of the IrDA frame occurred, but with an underrun error. OR 2. Transmission of the last bit of the IrDA frame completed successfully.	1. Read RESUME register. OR 2. Read IIR register.
6	Receiver line status interrupt	CRC, ABORT, or frame-length error is written into STATUS FIFO.	Read STATUS FIFO (read until empty - maximum of eight reads required).
7	Received EOF	Received end-of-frame.	Read IIR register.

23.2.7.4 CIR Mode Interrupts

The CIR mode uses a subset of the existing IrDA mode interrupts. Table 23-6 summarizes the interrupt modes that are to be maintained. In CIR mode, IIR bit 5 has a single purpose of indicating that the last bit of infrared data has been passed to the IR TX pin.

Table 23-6. CIR Mode Interrupts

IIR Bit	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read RHR register until interrupt condition disappears.
1	THR interrupt	TFE (THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR register until interrupt condition disappears.
2	RXSTOPIT interrupt	Receive stop interrupt (depending on value set in the BOF length register (EBLR))	Read IIR register.
3	RXOEIT interrupt	Write to RHR register when RX FIFO full.	Read RESUME register.
4	N/A for CIR mode		
5	TX status	Transmission of the last bit of the frame is completed successfully.	Read IIR register.
6-7	N/A for CIR mode		

23.2.7.5 Wake-Up Interrupt

Wake-up interrupt is a special interrupt, which is not designed in the same way than previous ones. This interrupt is enabled when the RX_CTS_DSR_WAKE_UP_ENABLE bit of the supplementary control register (SCR[4]) is set to one. The IIR register is not modified when it occurs, SSR[1] must be checked to detect a wake-up-event. When wake-up interrupt occurs, the only way to clear it, is to reset SCR[4] to 0. Wake-up can also occur if the WER[7] TX_WAKEUP_EN is set to 1 and one of the followings occurred:

- THR interrupt occurred if it is enabled (omitted if TX DMA request is enabled)
- TX DMA request occurred if it is enabled
- TX_STATUS_IT occurred if it is enabled (only IrDA and CIR modes). Can not be used with THR interrupt

23.2.8 Sleep Modes

23.2.8.1 When in UART Mode

In UART modes, sleep mode is enabled by writing a 1 to IER[4] (when EFR[4] = 1). Sleep mode is entered when:

- The serial data input line, RX is idle
- The TX FIFO and TX shift register are empty
- The RX FIFO is empty
- There are no interrupts pending except THR interrupts

It should be noted that sleep mode is a good way to lower power consumption of UART but this state can be achieved only when the UART is set in modem mode. Therefore, even if the UART has no functional key role, it must be initialized in a functional mode to take advantage of sleep mode. In sleep mode the module clock and baud rate clock are stopped internally. Because most registers are clocked using these clocks the power consumption is greatly reduced. The module wakes up when any change is detected on the RX line, if data is written to the TX FIFO, when there is any change in the state of the modem input pins. Note that an interrupt can be generated on a wake up event by setting SCR[4] to 1. See interrupt section to know how to manage it.

NOTE: Writing to the divisor latches, DLL and DLH, to set the baud clock, BCLK, must not be done during sleep mode. Therefore, it is advisable to disable sleep mode using IER[4] before writing to DLL or DLH.

23.2.8.2 When in IR-IrDA and CIR Modes

In IrDA/CIR modes, sleep mode is enabled by writing a 1 to MDR1[3]. Sleep mode is entered when:

- The serial data input line, RX is idle
- The TX FIFO and TX shift register are empty
- The RX FIFO is empty
- There are no interrupts pending except THR interrupts

The module wakes up when any change is detected on the RX line or if data is written to the TX FIFO.

23.2.9 Idle Modes

Sleep and AutoIdle modes are embedded power-saving features. At the system level, power reduction techniques can be applied by shutting down certain internal clock and power domains of the device.

- The UART supports an idle req - idle ack handshaking protocol. This protocol is used at system level to shut down clocks of the UART in a clean and controlled manner and to switch the UART from the interrupt generation mode to a wakeup generation mode for unmasked events (Refer to SYSC[2] and WER).
- In the wakeup generation mode, interrupt request generation and DMA request generation are disabled.

23.2.10 Programmable Baud Rate Generator

The UART/rRDA/CIR module contains a programmable baud generator and a set of fixed dividers that takes the 48 MHz clock input and divides it down to the expected baud-rate. The baud rate generator and associated controls is shown in Figure 23-27.

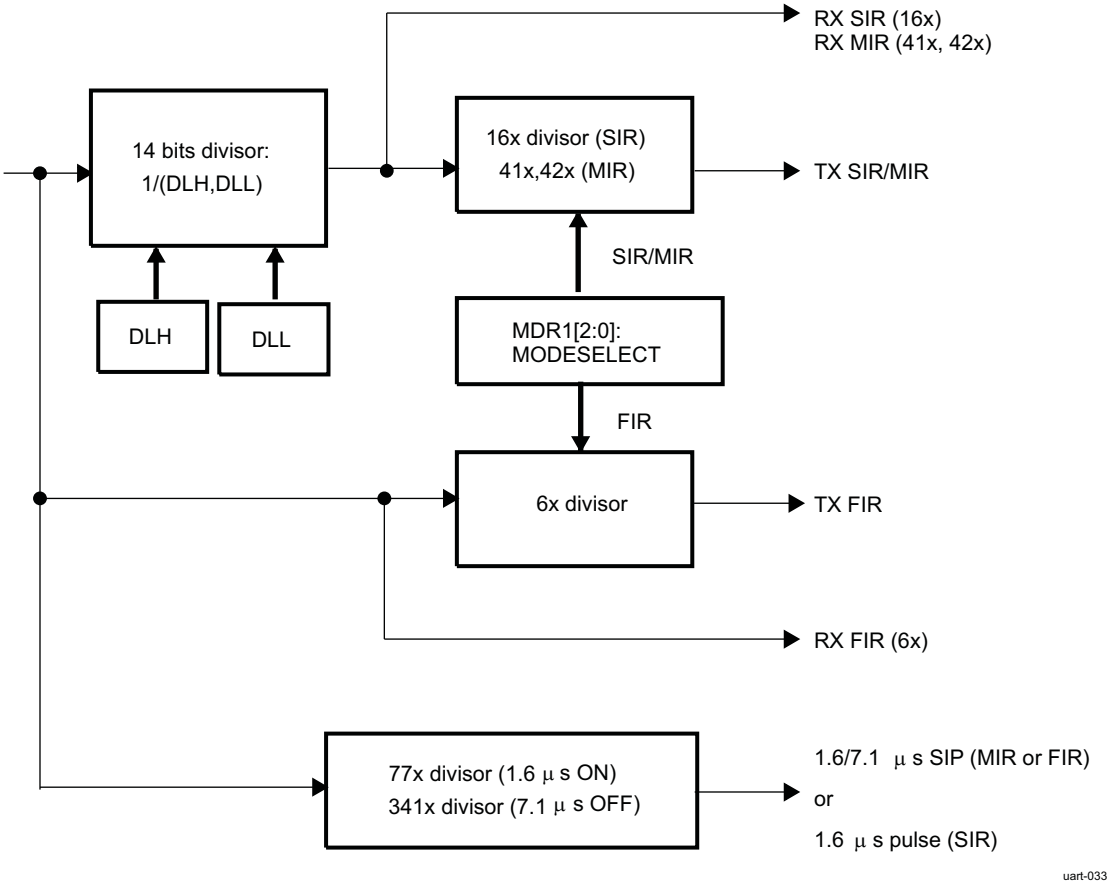
CAUTION

It is mandatory that the MODESELECT bit in MDR1[2:0] = 7h (disable) before initializing or modifying clock parameter controls (DLH and DLL). Failure to observe this rule can result in unpredictable module behavior.

Choosing the appropriate divisor value:

- UART 16× mode: Divisor value = Operating Frequency/(16× baud rate)
- UART 13× mode: Divisor value = Operating Frequency/(13× baud rate)
- SIR mode: Divisor value = Operating Frequency/(16× baud-rate)
- MIR mode: Divisor value = Operating Frequency/(41×/42× baud-rate)
- FIR mode: Divisor value = none

Figure 23-27. BAUD Rate Generator



23.2.10.1 UART Baud Rates (48 MHz Clock)

Table 23-7 describes the UART baud rate settings.

Table 23-7. UART Baud Rate Settings (48-MHz Clock)

Baud Rate	Baud Multiple	DLH, DLL (Decimal)	DLH, DLL (Hex)	Actual Baud Rate	Error (%)
0.3 Kbps	16x	10000	27h, 10h	0.3 Kbps	0
0.6 Kbps	16x	5000	13h, 88h	0.6 Kbps	0
1.2 Kbps	16x	2500	09h, C4h	1.2 Kbps	0
2.4 Kbps	16x	1250	04h, E2h	2.4 Kbps	0
4.8 Kbps	16x	625	02h, 71h	4.8 Kbps	0
9.6 Kbps	16x	312	01h, 38h	9.6153 Kbps	+0.16
14.4 Kbps	16x	208	00h, D0h	14.423 Kbps	+0.16
19.2 Kbps	16x	156	00h, 9Ch	19.231 Kbps	+0.16
28.8 Kbps	16x	104	00h, 68h	28.846 Kbps	+0.16
38.4 Kbps	16x	78	00h, 4Eh	38.462 Kbps	+0.16
57.6 Kbps	16x	52	00h, 34h	57.692 Kbps	+0.16
115.2 Kbps	16x	26	00h, 1Ah	115.38 Kbps	+0.16
230.4 Kbps	16x	13	00h, 0Dh	230.77 Kbps	+0.16
460.8 Kbps	13x	8	00h, 08h	461.54 Kbps	+0.16
921.6 Kbps	13x	4	00h, 04h	923.08 Kbps	+0.16
1.843 Mbps	13x	2	00h, 02h	1.846 Mbps	+0.16
3.6884 Mbps	13×	1	00h, 01h	3.6923 Mbps	+0.16
3.0 Mbps	16x	1	00h, 01h	3.0 Mbps	0

23.2.10.2 IrDA Baud Rates (48 MHz Clock)

Table 23-8 lists the IrDA baud rate settings.

Table 23-8. IrDA Baud Rates Settings

Baud Rate	IR Mode	Baud Multiple	Encoding	DLH, DLL (Decimal)	Actual Baud Rate	Error (%)(1)	Source Jitter (%)	Pulse Duration
2.4 Kbps	SIR	16×	3/16	1250	2.4 Kbps	0	0	78.1 µs
9.6 Kbps	SIR	16x	3/16	312	9.6153 Kbps	+0.16	0	19.5 µs
19.2 Kbps	SIR	16x	3/16	156	19.231 Kbps	+0.16	0	9.75 µs
38.4 Kbps	SIR	16x	3/16	78	38.462 Kbps	+0.16	0	4.87 µs
57.6 Kbps	SIR	16x	3/16	52	57.692 Kbps	+0.16	0	3.25 µs
115.2 Kbps	SIR	16x	3/16	26	115.38 Kbps	+0.16	0	1.62 µs
0.576 Mbps	MIR	41x/42x	1/4	2	0.5756 Mbps(2)	0	-2.0375	416 ns
1.152 Mbps	MIR	41x/42x	1/4	1	1.1511 Mbps(2)	0	-2.0375	208 ns
4 Mbps	FIR	6x	4PPM	-	4 Mbps	0	0	125 ns

(1) Baud rate error and source jitter table values do not include 48 MHz reference clock error and jitter
(2) Average value

23.2.10.3 Autobauding Modes

In autobauding mode, the UART extracts transfer characteristics (speed, length, and parity) from an AT command. These characteristics are used to receive data after an “at” (AT) and to send data.

The valid AT commands follow:

AT	DATA	<CR>
at	DATA	<CR>
A/		
a/		

A line break during the acquisition of the sequence AT is not recognized, and echo functionality is not implemented in hardware. A/ and a/ are not used to extract characteristics, but they must be recognized because of their special meaning. Either A/ or a/ is used to instruct the software to repeat the last received AT command; therefore, an a/ always follows an AT, and transfer characteristics are not expected to change between an AT and an a/.

When a valid AT is received, AT and all subsequent data, including the final <CR> (0Dh), are saved to RX FIFO. The autobaud state-machine waits for the next valid AT command. If an a/ (A/) is received, the a/ (A/) is saved into RX FIFO and the state-machine waits for the next valid AT command.

On the first successful detection of the baud rate, the UART activates an interrupt to signify that the AT (upper or lower case) sequence is detected. The UASR register reflects the correct settings for the baud rate detected. The interrupt activity continues in this way each time a subsequent character is received. Therefore, it is recommended that the software enables the RHR interrupt when using the autobaud mode.

The following settings are detected in autobaud mode with a module clock of 48 MHz:

- Speed: 115.2k baud, 57.6k baud, 38.4k baud, 28.8k baud, 19.2k baud, 14.4k baud, 9.6k baud, 4.8k baud, 2.4k baud, or 1.2k baud
- Length: 7 or 8 bits
- Parity: Odd, even, or space

NOTE: The combination of 7-bit character + space parity is not supported.

The method used to identify the speed is:

- Detect the transition 1->0 on the received data. This happens as soon as a stop to start bit transition occurs. The transition is valid after a majority vote on 3 sampling period.
- Sample the start bit duration with 115 200 × 16 Hz clock frequency as long as there is no rising edge. A transition 0->1 is considered as valid after a majority vote on 3 sampling period.
- Compare the sampled value with a table. If the sampled value is outside a valid range an error is reported (no speed identified). And the hardware goes back to the first state (1).
- Else store the first data bit in the received register (for serial to parallel conversion) and go to frame format identification.

The next received bits are sampled according to the programmed baud rate. However, after seven bits reception, the speed identification must be restarted since we may receive several “a” or “A” character before a valid “t” or “T” character.

Autobauding mode is selected when the MODESELECT field in MDR1[2:0] is set to 2h. In the UART autobauding mode, DLL, DLH, and LCR[5:0] settings are not used; instead, UASR is updated with the configuration detected by the autobauding logic.

23.3 UART Registers

Each register is selected using a combination of address and, for some, LCR register bit settings as shown in Table 23-9. For the base address of these registers, see Table 1-12. The following registers are accessible by the local host (LH) at address = module base address + address offset. The module base address is the module start address. Note that register address offsets depends upon the module address alignment at the system top level.

Table 23-9. UART Registers

Address Offset	Registers					
	LCR[7] = 0		LCR[7:0] ≠ BFh		LCR[7:0] = BFh	
	Read	Write	Read	Write	Read	Write
0h	RHR	THR	DLL	DLL	DLL	DLL
4h	IER(1)	IER(1)	DLH	DLH	DLH	DLH
8h	IIR	FCR(2)	IIR	FCR(2)	EFR	EFR
Ch	LCR	LCR	LCR	LCR	LCR	LCR
10h	MCR(2)	MCR(2)	MCR(2)	MCR(2)	XON1/ADDR1	XON1/ADDR1
14h	LSR	-	LSR	-	XON2/ADDR2	XON2/ADDR2
18h	MSR/TCR(3)	TCR(3)	MSR/TCR(3)	TCR(3)	XOFF1/TCR(3)	XOFF1/TCR(3)
1Ch	SPR/TLR(3)	SPR/TLR(3)	SPR/TLR(3)	SPR/TLR(3)	XOFF2/TLR(3)	XOFF2/TLR(3)
20h	MDR1	MDR1	MDR1	MDR1	MDR1	MDR1
24h	MDR2	MDR2	MDR2	MDR2	MDR2	MDR2
28h	SFLSR	TXFLL	SFLSR	TXFLL	SFLSR	TXFLL
2Ch	RESUME	TXFLH	RESUME	TXFLH	RESUME	TXFLH
30h	SFREGL	RXFLL	SFREGL	RXFLL	SFREGL	RXFLL
34h	SFREGH	RXFLH	SFREGH	RXFLH	SFREGH	RXFLH
38h	BLR	BLR	UASR	-	UASR	-
3Ch	ACREG	ACREG	-	-	-	-
40h	SCR	SCR	SCR	SCR	SCR	SCR
44h	SSR	SSR[2]	SSR	SSR[2]	SSR	SSR[2]
48h	EBLR	EBLR	-	-	-	-
50h	MVR	-	MVR	-	MVR	-
54h	SYSC	SYSC	SYSC	SYSC	SYSC	SYSC
58h	SYSS	-	SYSS	-	SYSS	-
5Ch	WER	WER	WER	WER	WER	WER
60h	CFPS	CFPS	CFPS	CFPS	CFPS	CFPS
80h	MDR3	MDR3	MDR3	MDR3	MDR3	MDR3

(1) In UART modes, IER[7:4] can only be written when EFR[4] = 1; in IrDA/CIR modes, EFR[4] has no impact on the access to IER[7:4].
(2) MCR[7:5] and FCR[5:4] can only be written when EFR[4] = 1.
(3) Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.

23.3.1 Receiver Holding Register (RHR)

The receiver section consists of the receiver holding register and the receiver shift register. The RHR is actually a 64-byte FIFO. The receiver shift register receives serial data from RX input. The data is converted to parallel data and moved to the RHR. If the FIFO is disabled, location zero of the FIFO is used to store the single data character.

NOTE: If an overflow occurs, the data in the RHR is not overwritten.

The receiver holding register (RHR) register is shown in Figure 23-28 and described in Table 23-10.

Figure 23-28. Receiver Holding Register (RHR)

15	8	7	0
Reserved		RHR	
R-0		R-unknown	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-10. Receiver Holding Register (RHR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	RHR	0-FFh	Receive holding register.

23.3.2 Transmit Holding Register (THR)

The transmitter section consists of the transmit holding register and the transmit shift register. The transmit holding register is a 64-byte FIFO. The MPU writes data to the THR. The data is placed in the transmit shift register where it is shifted out serially on the TX output. If the FIFO is disabled, location zero of the FIFO is used to store the data.

The transmit holding register (THR) is shown in Figure 23-29 and described in Table 23-11.

Figure 23-29. Transmit Holding Register (THR)

15	8	7	0
Reserved		THR	
R-0		W-unknown	

LEGEND: R/W = Read/Write; W = Write only; -*n* = value after reset

Table 23-11. Transmit Holding Register (THR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	THR	0-FFh	Transmit holding register.

23.3.3 Interrupt Enable Register (IER) - UART Mode

The interrupt enable register (IER) can be programmed to enable/disable any interrupt. There are seven types of interrupt in this mode: receiver error, RHR interrupt, THR interrupt, XOFF received and $\overline{CTS}$/$\overline{RTS}$ change of state from low to high. Each interrupt can be enabled/disabled individually. There is also a sleep mode enable bit. The UART interrupt enable register (IER) is shown in Figure 23-30 and described in Table 23-12.

Figure 23-30. UART Interrupt Enable Register (IER)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
CTSIT	RTSIT	XOFFIT	SLEEPMODE	MODEMSTSIT	LINESTSIT	THRIT	RHRIT
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-12. UART Interrupt Enable Register (IER) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	CTSIT		Can be written only when EFR[4] = 1.
		0	Disables the $\overline{CTS}$ interrupt.
		1	Enables the $\overline{CTS}$ interrupt.
6	RTSIT		Can be written only when EFR[4] = 1.
		0	Disables the $\overline{RTS}$ interrupt.
		1	Enables the $\overline{RTS}$ interrupt.
5	XOFFIT		Can be written only when EFR[4] = 1.
		0	Disables the XOFF interrupt.
		1	Enables the XOFF interrupt.
4	SLEEPMODE		Can be only written when EFR[4] = 1.
		0	Disables sleep mode.
		1	Enables sleep mode (stop baud rate clock when the module is inactive).
3	MODEMSTSIT	0	Disables the modem status register interrupt.
		1	Enables the modem status register interrupt
2	LINESTSIT	0	Disables the receiver line status interrupt.
		1	Enables the receiver line status interrupt.
1	THRIT	0	Disables the THR interrupt.
		1	Enables the THR interrupt.
0	RHRIT	0	Disables the RHR interrupt and time out interrupt.
		1	Enables the RHR interrupt and time out interrupt.

23.3.4 Interrupt Enable Register (IER) - IrDA Mode

The IrDA interrupt enable register (IER) can be programmed to enable/disable any interrupt. There are 8 types of interrupt in these modes, received EOF, LSR interrupt, TX status, status FIFO interrupt, RX overrun, last byte in RX FIFO, THR interrupt, and RHR interrupt. Each interrupt can be enabled/disabled individually. The IrDA interrupt enable register (IER) is shown in Figure 23-31 and described in Table 23-13.

NOTE: The TXSTATUSIT interrupt reflects two possible conditions. The MDR2[0] bit should be read to determine the status in the event of this interrupt.

Figure 23-31. IrDA Interrupt Enable Register (IER)

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
EOFIT	LINESTSIT	TXSTATUSIT	STSFIFOTRIGIT	RXOVERRUNIT	LASTRXBYTEIT	THRIT	RHRIT
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 23-13. IrDA Interrupt Enable Register (IER) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	EOFIT	0	Disables the received EOF interrupt.
		1	Enables the received EOF interrupt.
6	LINESTSIT	0	Disables the receiver line status interrupt.
		1	Enables the receiver line status interrupt.
5	TXSTATUSIT	0	Disables the TX status interrupt.
		1	Enables the TX status interrupt.
4	STSFIFOTRIGIT	0	Disables status FIFO trigger level interrupt.
		1	Enables status FIFO trigger level interrupt.
3	RXOVERRUNIT	0	Disables the RX overrun interrupt.
		1	Enables the RX overrun interrupt.
2	LASTRXBYTEIT	0	Disables the last byte of frame in RX FIFO interrupt.
		1	Enables the last byte of frame in RX FIFO interrupt.
1	THRIT	0	Disables the THR interrupt.
		1	Enables the THR interrupt.
0	RHRIT	0	Disables the RHR interrupt.
		1	Enables the RHR interrupt.

23.3.5 Interrupt Enable Register (IER) - CIR Mode

The CIR interrupt enable register (IER) can be programmed to enable/disable any interrupt. There are 5 types of interrupt in these modes, TX status, RX overrun, RX stop interrupt, THR interrupt, and RHR interrupt. Each interrupt can be enabled/disabled individually. The CIR interrupt enable register (IER) is shown in Figure 23-32 and described in Table 23-14.

NOTE: In CIR mode, the TXSTATUSIT bit has only one meaning corresponding to the case MDR2[0] = 0.

The RXSTOPIT interrupt is generated based on the value set in the BOF Length register (EBLR).

Figure 23-32. CIR Interrupt Enable Register (IER)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved		TXSTATUSIT	Reserved	RXOVERRUNIT	RXSTOPIT	THRIT	RHRIT
R-0		R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 23-14. CIR Interrupt Enable Register (IER) Field Descriptions

Bit	Field	Value	Description
15-6	Reserved	0	Reserved.
5	TXSTATUSIT	0	Disables the TX status interrupt.
		1	Enables the TX status interrupt.
4	Reserved	0	Reserved.
3	RXOVERRUNIT	0	Disables the RX overrun interrupt.
		1	Enables the RX overrun interrupt.
2	RXSTOPIT	0	Disables the RX stop interrupt.
		1	Disables the RX stop interrupt.
1	THRIT	0	Disables the THR interrupt.
		1	Enables the THR interrupt.
0	RHRIT	0	Disables the RHR interrupt.
		1	Enables the RHR interrupt.

23.3.6 Interrupt Identification Register (IIR) - UART Mode

The UART interrupt identification register (IIR) is a read-only register that provides the source of the interrupt. The UART interrupt identification register (IIR) is shown in Figure 23-33 and described in Table 23-15.

NOTE: An interrupt source can be flagged only if enabled in the IER register.

Figure 23-33. UART Interrupt Identification Register (IIR)

15	8
Reserved	
R-0	

7	6	5	1	0
FCR_MIRROR		IT_TYPE		IT_PENDING
R-0		R-0		R-1

LEGEND: R = Read only; -*n* = value after reset

Table 23-15. UART Interrupt Identification Register (IIR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-6	FCR_MIRROR	0-3h	Mirror the contents of FCR[0] on both bits.
5-1	IT_TYPE	0-1Fh	Seven possible interrupts in UART mode; other combinations never occur:
		0	Modem interrupt. Priority = 4.
		1h	THR interrupt. Priority = 3.
		2h	RHR interrupt. Priority = 2.
		3h	Receiver line status error. Priority = 1.
		4h-5h	Reserved
		6h	Rx timeout. Priority = 2.
		7h	Reserved
		8h	Xoff/special character. Priority = 5.
		9h-Fh	Reserved
		10h	$\overline{CTS}$, $\overline{RTS}$, $\overline{DSR}$ change state from active (low) to inactive (high). Priority = 6.
		11h-1Fh	Reserved
0	IT_PENDING		Interrupt pending.
		0	An interrupt is pending.
		1	No interrupt is pending.

23.3.7 Interrupt Identification Register (IIR) - IrDA Mode

The IrDA interrupt identification register (IIR) is a read-only register that provides the source of the interrupt. The IrDA interrupt identification register (IIR) is shown in Figure 23-34 and described in Table 23-16.

NOTE: An interrupt source can be flagged only if enabled in the IER register.

Figure 23-34. IrDA Interrupt Identification Register (IIR)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
EOF_IT	LINE_STS_IT	TX_STATUS_IT	STS_FIFO_IT	RX_OE_IT	RX_FIFO_LAST_BYTE_IT	THR_IT	RHR_IT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-16. IrDA Interrupt Identification Register (IIR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	EOF_IT	0	Received EOF interrupt inactive.
		1	Received EOF interrupt active.
6	LINE_STS_IT	0	Receiver line status interrupt inactive.
		1	Receiver line status interrupt active.
5	TX_STATUS_IT	0	TX status interrupt inactive.
		1	TX status interrupt active.
4	STS_FIFO_IT	0	Status FIFO trigger level interrupt inactive.
		1	Status FIFO trigger level interrupt active.
3	RX_OE_IT	0	RX overrun interrupt inactive.
		1	RX overrun interrupt active.
2	RX_FIFO_LAST_BYTE_IT	0	Last byte of frame in RX FIFO interrupt inactive.
		1	Last byte of frame in RX FIFO interrupt active.
1	THR_IT	0	THR interrupt inactive.
		1	THR interrupt active.
0	RHR_IT	0	RHR interrupt inactive.
		1	RHR interrupt active.

23.3.8 Interrupt Identification Register (IIR) - CIR Mode

The CIR interrupt identification register (IIR) is a read-only register that provides the source of the interrupt. The CIR interrupt identification register (IIR) is shown in Figure 23-35 and described in Table 23-17.

NOTE: An interrupt source can be flagged only if enabled in the IER register.

Figure 23-35. CIR Interrupt Identification Register (IIR)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved		TXSTATUSIT	Reserved	RXOEIT	RXSTOPIT	THRIT	RHRIT
R-0		R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-17. CIR Interrupt Identification Register (IIR) Field Descriptions

Bit	Field	Value	Description
15-6	Reserved	0	Reserved.
5	TXSTATUSIT	0	TX status interrupt inactive
		1	TX status interrupt active
4	Reserved	0	Reserved.
3	RXOEIT	0	RX overrun interrupt inactive
		1	RX overrun interrupt active
2	RXSTOPIT	0	Receive stop interrupt is inactive
		1	Receive stop interrupt is active
1	THRIT	0	THR interrupt inactive
		1	THR interrupt active
0	RHRIT	0	RHR interrupt inactive
		1	RHR interrupt active

23.3.9 FIFO Control Register (FCR)

The FIFO Control Register (FCR) is shown in Figure 23-36 and described in Table 23-18.

Figure 23-36. FIFO Control Register (FCR)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
RX_FIFO_TRIG		TX_FIFO_TRIG		DMA_MODE	TX_FIFO_CLEAR	RX_FIFO_CLEAR	FIFO_EN
W-0		W-0		W-0	W-0	W-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 23-18. FIFO Control Register (FCR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-6	RX_FIFO_TRIG	0-3h	Sets the trigger level for the RX FIFO: If SCR[7] = 0 and TLR[7:4] ≠ 0000, RX_FIFO_TRIG is not considered. If SCR[7] = 1, RX_FIFO_TRIG is 2 LSB of the trigger level (1 to 63 on 6 bits) with the granularity 1. If SCR[7] = 0 and TLR[7:4] = 0000:
		0	8 characters
		1h	16 characters
		2h	56 characters
		3h	60 characters
5-4	TX_FIFO_TRIG	0-3h	Can be written only if EFR[4] = 1. Sets the trigger level for the TX FIFO: If SCR[6] = 0 and TLR[3:0] ≠ 0000, TX_FIFO_TRIG is not considered. If SCR[6] = 1, TX_FIFO_TRIG is 2 LSB of the trigger level (1 to 63 on 6 bits) with a granularity of 1. If SCR[6] = 0 and TLR[3:0] = 0000:
		0	8 characters
		1h	16 characters
		2h	32 characters
		3h	56 characters
3	DMA_MODE		Can be changed only when the baud clock is not running (DLL and DLH cleared to 0). If SCR[0] = 0, this register is considered.
		0	DMA_MODE 0 (No DMA).
		1	DMA_MODE 1 (UART_NDMA_REQ[0] in TX, UART_NDMA_REQ[1] in RX).
2	TX_FIFO_CLEAR	0	No change.
		1	Clears the transmit FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.
1	RX_FIFO_CLEAR	0	No change.
		1	Clears the receive FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.
0	FIFO_EN		Can be changed only when the baud clock is not running (DLL and DLH cleared to 0).
		0	Disables the transmit and receive FIFOs. The transmit and receive holding registers are 1-byte FIFOs.
		1	Enables the transmit and receive FIFOs. The transmit and receive holding registers are 64-byte FIFOs.

23.3.10 Line Control Register (LCR)

The line control register (LCR) is shown in Figure 23-37 and described in Table 23-19.

NOTE: As soon as LCR[6] is set to 1, the TX line is forced to 0 and remains in this state as long as LCR[6] = 1.

Figure 23-37. Line Control Register (LCR)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
DIV_EN	BREAK_EN	PARITY_TYPE2	PARITY_TYPE1	PARITY_EN	NB_STOP	CHAR_LENGTH	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-19. Line Control Register (LCR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	DIV_EN		Divisor latch enable.
		0	Normal operating condition.
		1	Divisor latch enable. Allows access to DLL and DLH.
6	BREAK_EN		Break control bit.
			Note: When LCR[6] is set to 1, the TX line is forced to 0 and remains in this state as long as LCR[6] = 1.
		0	Normal operating condition.
		1	Forces the transmitter output to go low to alert the communication terminal.
5	PARITY_TYPE2		If LCR[3] = 1:
		0	If LCR[5] = 0, LCR[4] selects the forced parity format.
		1	If LCR[5] = 1 and LCR[4] = 0, the parity bit is forced to 1 in the transmitted and received data.
		1	If LCR[5] = 1 and LCR[4] = 1, the parity bit is forced to 0 in the transmitted and received data.
4	PARITY_TYPE1		If LCR[3] = 1:
		0	Odd parity is generated.
		1	Even parity is generated.
3	PARITY_EN		Parity bit.
		0	No parity.
		1	A parity bit is generated during transmission, and the receiver checks for received parity.
2	NB_STOP		Specifies the number of stop bits.
		0	1 stop bit (word length = 5, 6, 7, 8).
		1	1.5 stop bits (word length = 5) or 2 stop bits (word length = 6, 7, 8).
1-0	CHAR_LENGTH	0-3h	Specifies the word length to be transmitted or received.
		0	5 bits
		1h	6 bits
		2h	7 bits
		3h	8 bit

23.3.11 Modem Control Register (MCR)

Bits 3-0 control the interface with the modem, data set, or peripheral device that is emulating the modem. The modem control register (MCR) is shown in Figure 23-38 and described in Table 23-20.

Figure 23-38. Modem Control Register (MCR)

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
Reserved	TCRTLR	XONEN	LOOPBACKEN	CDSTSCH	RISTSCH	RTS	DTR
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 23-20. Modem Control Register (MCR) Field Descriptions

Bit	Field	Value	Description
15-7	Reserved	0	Reserved.
6	TCRTLR		Can be written only when EFR[4] = 1.
		0	No action.
		1	Enables access to the TCR and TLR registers.
5	XONEN		Can be written only when EFR[4] = 1.
		0	Disable XON any function.
		1	Enable XON any function.
4	LOOPBACKEN		Loopback mode enable.
		0	Normal operating mode.
		1	Enable local loopback mode (internal). In this mode, the MCR[3:0] signals are looped back into MSR[7:4]. The transmit output is looped back to the receive input internally.
3	CDSTSCH	0	In loopback mode, forces $\overline{\text{DCD}}$ input high and IRQ outputs to INACTIVE state.
		1	In loopback mode, forces $\overline{\text{DCD}}$ input low and IRQ outputs to INACTIVE state.
2	RISTSCH	0	In loopback mode, forces $\overline{\text{RI}}$ input inactive (high).
		1	In loopback mode, forces $\overline{\text{RI}}$ input active (low).
1	RTS		In loopback mode, controls MSR[4]. If auto-RTS is enabled, the $\overline{\text{RTS}}$ output is controlled by hardware flow control.
		0	Force $\overline{\text{RTS}}$ output to inactive (high).
		1	Force $\overline{\text{RTS}}$ output to active (low).
0	DTR	0	Force $\overline{\text{DTR}}$ output (used in loopback mode) to inactive (high).
		1	Force $\overline{\text{DTR}}$ output (used in loopback mode) to active (low).

23.3.12 Line Status Register (LSR) - UART Mode

When the UART line status register (LSR) is read, LSR[4:2] reflect the error bits (BI, FE, PE) of the character at the top of the RX FIFO (next character to be read). Therefore, reading the LSR and then reading the RHR identifies errors in a character. Reading RHR updates BI, FE, and PE. LSR [7] is set when there is an error anywhere in the RX FIFO and is cleared only when there are no more errors remaining in the RX FIFO. The UART line status register (LSR) is shown in Figure 23-39 and described in Table 23-21.

NOTE: Reading the LSR does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the RHR. Reading LSR clears OE if set.

Figure 23-39. UART Line Status Register (LSR)

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
RXFIFOSTS	TXSRE	TXFIFOE	RXBI	RXFE	RXPE	RXOE	RXFIFOE
R-0	R-1	R-1	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-21. UART Line Status Register (LSR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	RXFIFOSTS	0	Normal operation.
		1	At least one parity error, framing error, or break indication in the RX FIFO. Bit 7 is cleared when no errors are present in the RX FIFO.
6	TXSRE	0	Transmitter hold (TX FIFO) and shift registers are not empty.
		1	Transmitter hold (TX FIFO) and shift registers are empty.
5	TXFIFOE	0	Transmit hold register (TX FIFO) is not empty.
		1	Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.
4	RXBI	0	No break condition.
		1	A break was detected while the data being read from the RX FIFO was being received (RX input was low for one character + 1 bit time frame).
3	RXFE	0	No framing error in data being read from RX FIFO.
		1	Framing error occurred in data being read from RX FIFO (received data did not have a valid stop bit).
2	RXPE	0	No parity error in data being read from RX FIFO.
		1	Parity error in data being read from RX FIFO.
1	RXOE	0	No overrun error.
		1	Overrun error occurred. Set when the character held in the receive shift register is not transferred to the RX FIFO. This case occurs only when receive FIFO is full.
0	RXFIFOE	0	No data in the receive FIFO.
		1	At least one data character in the RX FIFO.

23.3.13 Line Status Register (LSR) - IrDA Mode

When the IrDA line status register (LSR) is read, LSR[4:2] reflect the error bits (FL, CRC, ABORT) of the frame at the top of the status FIFO (next frame status to be read). The IrDA line status register (LSR) is shown in Figure 23-40 and described in Table 23-22.

Figure 23-40. IrDA Line Status Register (LSR)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
THR_EMPTY	STS_FIFO_FULL	RX_LAST_BYTE	FRAME_TOO_LONG	ABORT	CRC	STS_FIFO_E	RX_FIFO_E
R-1	R-0	R-0	R-0	R-0	R-0	R-1	R-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-22. IrDA Line Status Register (LSR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	THR_EMPTY	0	Transmit holding register (TX FIFO) is not empty.
		1	Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.
6	STS_FIFO_FULL	0	Status FIFO is not full.
		1	Status FIFO is full.
5	RX_LAST_BYTE	0	The RX FIFO (RHR) does not contain the last byte of the frame to be read.
		1	The RX FIFO (RHR) contains the last byte of the frame to be read. This bit is set to 1 only when the last byte of a frame is available to be read. It is used to determine the frame boundary. It is cleared on a single read of the LSR register.
4	FRAME_TOO_LONG	0	No frame-too-long error in frame.
		1	Frame-too-long error in the frame at the top of the status FIFO (next character to be read). This bit is set to 1 when a frame exceeding the maximum length (set by RXFLH and RXFLL registers) is received. When this error is detected, current frame reception is terminated. Reception is stopped until the next START flag is detected.
3	ABORT	0	No abort pattern error in frame.
		1	Abort pattern received. SIR and MIR: abort pattern. FIR: Illegal symbol.
2	CRC	0	No CRC error in frame.
		1	CRC error in the frame at the top of the status FIFO (next character to be read).
1	STS_FIFO_E	0	Status FIFO is not empty.
		1	Status FIFO is empty.
0	RX_FIFO_E	0	At least one data character in the RX FIFO.
		1	No data in the receive FIFO.

23.3.14 Line Status Register (LSR) - CIR Mode

The CIR line status register (LSR) is shown in Figure 23-41 and described in Table 23-23.

Figure 23-41. CIR Line Status Register (LSR)

15					8
Reserved					
R-0					
7	6	5	4	1	0
THREMPTY	Reserved	RXSTOP	Reserved		RXFIFOE
R-1	R-0	R-0	R-0		R-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-23. CIR Line Status Register (LSR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	THREMPTY	0	Transmit holding register (TX FIFO) is not empty.
		1	Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.
6	Reserved	0	Reserved.
5	RXSTOP		The RXSTOP is generated based on the value set in the BOF Length register (EBLR).
		0	Reception is on going or waiting for a new frame.
		1	Reception is completed. It is cleared on a single read of the LSR register.
4-1	Reserved	0	Reserved
0	RXFIFOE	0	At least one data character in the RX FIFO.
		1	No data in the receive FIFO.

23.3.15 Modem Status Register (MSR)

The modem status register (MSR) provides information about the current state of the control lines from the modem, data set, or peripheral device to the Local Host. It also indicates when a control input from the modem changes state. The modem status register (MSR) is shown in Figure 23-42 and described in Table 23-24.

Figure 23-42. Modem Status Register (MSR)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
NCD_STS	NRI_STS	NDSR_STS	NCTS_STS	DCD_STS	RI_STS	DSR_STS	CTS_STS
R-unknown	R-unknown	R-unknown	R-unknown	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-24. Modem Status Register (MSR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	NCD_STS		This bit is the complement of the $\overline{DCD}$ input. In loopback mode, it is equivalent to MCR[3].
6	NRI_STS		This bit is the complement of the $\overline{RI}$ input. In loopback mode, it is equivalent to MCR[2].
5	NDSR_STS		This bit is the complement of the $\overline{DSR}$ input. In loopback mode, it is equivalent to MCR[0].
4	NCTS_STS		This bit is the complement of the $\overline{CTS}$ input. In loopback mode, it is equivalent to MCR[1].
3	DCD_STS	0	No change.
		1	Indicates that $\overline{DCD}$ input (or MCR[3] in loopback mode) has changed. Cleared on a read.
2	RI_STS	0	No change.
		1	Indicates that $\overline{RI}$ input (or MCR[2] in loopback mode) changed state from low to high. Cleared on a read.
1	DSR_STS	0	No change.
		1	Indicates that $\overline{DSR}$ input (or MCR[0] in loopback mode) changed state. Cleared on a read.
0	CTS_STS	0	No change.
		1	Indicates that $\overline{CTS}$ input (or MCR[1] in loopback mode) changed state. Cleared on a read.

23.3.16 Transmission Control Register (TCR)

The transmission control register (TCR) stores the receive FIFO threshold levels to start/stop transmission during hardware flow control. The transmission control register (TCR) is shown in Figure 23-43 and described in Table 23-25.

NOTE:

- Trigger levels from 0-60 bytes are available with a granularity of 4.
- Trigger level = 4 × [4-bit register value]
- You must ensure that TCR[3:0] > TCR[7:4], whenever auto-RTS or software flow control is enabled to avoid a misoperation of the device. In FIFO interrupt mode with flow control, you have to also ensure that the trigger level to HALT transmission is greater or equal to receive FIFO trigger level (either TLR[7:4] or FCR[7:6]); otherwise, FIFO operation stalls.
- In FIFO DMA mode with flow control, this concept does not exist because the DMA request is sent each time a byte is received.

Figure 23-43. Transmission Control Register (TCR)

15	8	7	4	3	0
Reserved		RXFIFOTRIGSTART		RXFIFOTRIGHALT	
R-0		R/W-0		R/W-Fh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-25. Transmission Control Register (TCR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-4	RXFIFOTRIGSTART	0-Fh	RX FIFO trigger level to RESTORE transmission (0 to 60).
3-0	RXFIFOTRIGHALT	0-Fh	RX FIFO trigger level to HALT transmission (0 to 60).

23.3.17 Scratchpad Register (SPR)

The scratchpad register (SPR) is a read/write register that does not control the module. It is a scratchpad register used to hold temporary data. The scratchpad register (SPR) is shown in Figure 23-44 and described in Table 23-26.

Figure 23-44. Scratchpad Register (SPR)

15	8	7	0
Reserved		SPR_WORD	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-26. Scratchpad Register (SPR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	SPR_WORD	0-FFh	Scratchpad register.

23.3.18 Trigger Level Register (TLR)

This register stores the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation. The trigger level register (TLR) is shown in Figure 23-45 and described in Table 23-27.

Figure 23-45. Trigger Level Register (TLR)

15 8	7 4	3 0
Reserved	RX_FIFO_TRIG_ DMA	TX_FIFO_TRIG_ DMA
R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-27. Trigger Level Register (TLR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-4	RX_FIFO_TRIG_ DMA	0-Fh	Receive FIFO trigger level. See Table 23-28.
3-0	TX_FIFO_TRIG_ DMA	0-Fh	Transmit FIFO trigger level. See Table 23-29.

Table 23-28. RX FIFO Trigger Level Setting Summary

SCR[7]	TLR[7:4]	Description
0	0	Defined by FCR[7:6] (either 8, 16, 56, 60 characters).
0	≠ 0000	Defined by TLR[7:4] (from 4 to 60 characters with a granularity of 4 characters).
1	any value	Defined by the concatenated value of TLR[7:4] and FCR[7:6] (from 1 to 63 characters with a granularity of 1 character). **Note:** the combination of TLR[7:4] = 0000 and FCR[7:6] = 00 (all zeros) is not supported (minimum of 1 character is required). All zeros results in unpredictable behavior.

Table 23-29. TX FIFO Trigger Level Setting Summary

SCR[6]	TLR[3:0]	Description
0	0	Defined by FCR[5:4] (either 8, 16, 32, 56 characters).
0	≠ 0000	Defined by TLR[3:0] (from 4 to 60 characters with a granularity of 4 characters).
1	any value	Defined by the concatenated value of TLR[3:0] and FCR[5:4] (from 1 to 63 characters with a granularity of 1 character). **Note:** the combination of TLR[3:0] = 0000 and FCR[5:4] = 00 (all zeros) is not supported (minimum of 1 character is required). All zeros results in unpredictable behavior.

23.3.19 Mode Definition Register 1 (MDR1)

The mode of operation is programmed by writing to MDR1[2:0]; therefore, the mode definition register 1 (MDR1) must be programmed on startup after configuration of the configuration registers (DLL, DLH, and LCR). The value of MDR1[2:0] must not be changed again during normal operation. The mode definition register 1 (MDR1) is shown in Figure 23-46 and described in Table 23-30.

NOTE: If the module is disabled by setting the MODESELECT field to 7h, interrupt requests can still be generated unless disabled through the interrupt enable register (IER). In this case, UART mode interrupts are visible. Reading the interrupt identification register (IIR) shows the UART mode interrupt flags.

Figure 23-46. Mode Definition Register 1 (MDR1)

15						8
Reserved						
R-0						
7	6	5	4	3	2	0
FRAMEENDMODE	SIPMODE	SCT	SETTXIR	IRSLEEP	MODESELECT	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-7h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-30. Mode Definition Register 1 (MDR1) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	FRAMEENDMODE		IrDA mode only.
		0	Frame-length method.
		1	Set EOT bit method.
6	SIPMODE		MIR/FIR modes only.
		0	Manual SIP mode: SIP is generated with the control of ACREG[3].
		1	Automatic SIP mode: SIP is generated after each transmission.
5	SCT		Store and control the transmission.
		0	Starts the infrared transmission when a value is written to the THR register.
		1	Starts the infrared transmission with the control of ACREG[2]. **Note:** Before starting any transmission, there must be no reception ongoing.
4	SETTXIR		Used to configure the infrared transceiver.
		0	If MDR2[7] = 0, no action; if MDR2[7] = 1, TXIR pin output is forced low.
		1	TXIR pin output is forced high (not dependant of MDR2[7] value).
3	IRSLEEP		IrDA/CIR sleep mode.
		0	IrDA/CIR sleep mode disabled.
		1	IrDA/CIR sleep mode enabled.
2-0	MODESELECT	0-7h	UART/IrDA/CIR mode selection.
		0	UART 16× mode.
		1	SIR mode.
		2h	UART 16× auto-baud.
		3h	UART 13× mode.
		4h	MIR mode.
		5h	FIR mode.
		6h	CIR mode.
		7h	Disable (default state).

23.3.20 Mode Definition Register 2 (MDR2)

The MDR2[0] bit describes the status of the TX status interrupt in IIR[5]. The IRTXUNDERRUN bit must be read after a TX status interrupt occurs. The MDR2[2:1] bits set the trigger level for the frame status FIFO (8 entries) and must be programmed before the mode is programmed in MDR1[2:0]. The mode definition register 2 (MDR2) is shown in Figure 23-47 and described in Table 23-31.

NOTE: The MDR2[6] bit gives the flexibility to invert the RX pin inside the UART module to ensure that the protocol at the input of the transceiver module has the same polarity at module level. By default, the RX pin is inverted because most of transceiver invert the IR receive pin.

Figure 23-47. Mode Definition Register 2 (MDR2)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
SETTXIRALT	IRRXINVERT	CIRPULSEMODE		UARTPULSE	STSFIFOTRIG		IRTXUNDERRUN
R/W-0	R/W-0	R/W-0		R/W-0	R/W-0		R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 23-31. Mode Definition Register 2 (MDR2) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	SETTXIRALT		Provides alternate functionality for MDR1[4].
		0	Normal mode
		1	Alternate mode for SETTXIR
6	IRRXINVERT		Only for IR mode (IrDA and CIR). Invert RX pin in the module before the voting or sampling system logic of the infrared block. This does not affect the RX path in UART modem modes.
		0	Inversion is performed.
		1	No inversion is performed.
5-4	CIRPULSEMODE	0-3h	CIR pulse modulation definition. Defines high level of the pulse width associated with a digit:
		0	Pulse width of 3 from 12 cycles.
		1h	Pulse width of 4 from 12 cycles.
		2h	Pulse width of 5 from 12 cycles.
		3h	Pulse width of 6 from 12 cycles.
3	UARTPULSE		UART mode only. Used to allow pulse shaping in UART mode.
		0	Normal UART mode.
		1	UART mode with pulse shaping.
2-1	STSFIFOTRIG	0-3h	Only for IrDA mode. Frame status FIFO threshold select:
		0	1 entry
		1h	4 entries
		2h	7 entries
		3h	8 entries
0	IRTXUNDERRUN		IrDA transmission status interrupt. When the TX status interrupt (IIR[5]) occurs, the meaning of the interrupt is:
		0	The last bit of the frame was transmitted successfully without error.
		1	An underrun occurred. The last bit of the frame was transmitted but with an underrun error. The bit is reset to 0 when the RESUME register is read.

23.3.21 Mode Definition Register 3 (MDR3)

The MDR3[0] DISABLE_CIR_RX_DEMOD bit will force the CIR receiver to bypass demodulation of received data if set.

The MDR3[1] NONDEFAULT_FREQ bit allows the user to set sample per bi. Set this bit if non default (48MHz) clock frequency is used to achieve less than 2 % error rate. Changing this bit automatically disables the device by setting MDR1[2:0] to 3h.

MDR3[2] SET_DMA_THRESHOLD bit enables the usage of different TX DMA threshold then 64-trigger, when set to 1h.

The mode definition register 3 (MDR3) is shown in Figure 23-48 and described in Table 23-32.

Figure 23-48. Mode Definition Register 3 (MDR3)

31	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved					SET_DMA_TX_THRESHOLD	NONDEFAULT_FREQ	DISABLE_CIR_RX_DEMOD
R-0					R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-32. Mode Definition Register 3 (MDR3) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved.
2	SET_DMA_THRESHOLD	0	Use the value of 64-tx trigger.
		1h	Sets different TX DMA threshold then 64-trigger
1	NONDEFAULT_FREQ	0	Default fclk frequencies
		1h	Nondefault fclk frequencies
0	DISABLE_CIR_RX_DEMOD	0	Enalbes CIR RX demodulation
		1h	Disables CIR RX demodulation

23.3.22 Status FIFO Line Status Register (SFLSR)

Reading the status FIFO line status register (SFLSR) effectively reads frame status information from the status FIFO. This register does not physically exist. Reading this register increments the status FIFO read pointer (SFREGL and SFREGH must be read first). The status FIFO line status register (SFLSR) is shown in Figure 23-49 and described in Table 23-33.

NOTE: Top of RX FIFO = Next frame to be read from RX FIFO.

Figure 23-49. Status FIFO Line Status Register (SFLSR)

15	8
Reserved	
R-0	

7	5	4	3	2	1	0
Reserved		OE_ERROR	FRAME_TOO_LONG_ERROR	ABORT_DETECT	CRC_ERROR	Reserved
R-0		R-unknown	R-unknown	R-unknown	R-unknown	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-33. Status FIFO Line Status Register (SFLSR) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	Reserved.
4	OE_ERROR	0	No error
		1	Overrun error in RX FIFO when frame at top of RX FIFO was received.
3	FRAME_TOO_LONG_ERROR	0	No error
		1	Frame-length too long error in frame at top of RX FIFO.
2	ABORT_DETECT	0	No error
		1	Abort pattern detected in frame at top of RX FIFO.
1	CRC_ERROR	0	No error
		1	CRC error in frame at top of RX FIFO.
0	Reserved	0	Reserved.

23.3.23 RESUME Register

The RESUME register is used to clear internal flags, which halt transmission/reception when an underrun/overrun error occurs. Reading this register resumes the halted operation. This register does not physically exist and always reads as 00. The RESUME register is shown in Figure 23-50 and described in Table 23-34.

Figure 23-50. RESUME Register

15	8	7	0
Reserved		RESUME	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-34. RESUME Register Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	RESUME	0-FFh	Dummy read to restart the TX or RX.

23.3.24 Status FIFO Register Low (SFREGL)

The frame lengths of received frames are written into the status FIFO. This information can be read by reading the status FIFO register low (SFREGL) and the status FIFO register high (SFREGH). These registers do not physically exist. The LSBs are read from SFREGL and the MSBs are read from SFREGH. Reading these registers does not alter the status FIFO read pointer. These registers must be read before the pointer is incremented by reading the SFLSR. The status FIFO register low (SFREGL) is shown in Figure 23-51 and described in Table 23-35.

Figure 23-51. Status FIFO Register Low (SFREGL)

15	8	7	0
Reserved		SFREGL	
R-0		R-unknown	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-35. Status FIFO Register Low (SFREGL) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	SFREGL	0-FFh	LSB part of the frame length.

23.3.25 Status FIFO Register High (SFREGH)

The frame lengths of received frames are written into the status FIFO. This information can be read by reading the status FIFO register low (SFREGL) and the status FIFO register high (SFREGH). These registers do not physically exist. The LSBs are read from SFREGL and the MSBs are read from SFREGH. Reading these registers does not alter the status FIFO read pointer. These registers must be read before the pointer is incremented by reading the SFLSR. The status FIFO register high (SFREGH) is shown in Figure 23-52 and described in Table 23-36.

Figure 23-52. Status FIFO Register High (SFREGH)

15	4	3	0
Reserved		SFREGH	
R-0		R-unknown	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-36. Status FIFO Register High (SFREGH) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved	0	Reserved.
3-0	SFREGH	0-Fh	MSB part of the frame length.

23.3.26 BOF Control Register (BLR)

The BLR[6] bit is used to select whether C0h or FFh start patterns are to be used, when multiple start flags are required in SIR mode. If only one start flag is required, this is always C0h. If *n* start flags are required, either (n – 1) C0h or (n –1) FFh flags are sent, followed by a single C0h flag (immediately preceding the first data byte). The BOF control register (BLR) is shown in Figure 23-53 and described in Table 23-37.

Figure 23-53. BOF Control Register (BLR)

15	8	7	6	5	0
Reserved		STSFIFORESET	XBOFTYPE	Reserved	
R-0		R/W-0	R/W-1	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-37. BOF Control Register (BLR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	STSFIFORESET		Status FIFO reset. This bit is self-clearing..
6	XBOFTYPE		SIR xBOF select.
		0	FFh start pattern is used.
		1	C0h start pattern is used.
0	Reserved	0	Reserved.

23.3.27 Auxiliary Control Register (ACREG)

The auxiliary control register (ACREG) is shown in Figure 23-54 and described in Table 23-38.

NOTE:

- If transmit FIFO is not empty and MDR1[5] = 1, IrDA starts a new transfer with data of previous frame as soon as abort frame has been sent. Therefore, TX FIFO must be reset before sending an abort frame.
- It is recommended to disable TX FIFO underrun capability by masking corresponding underrun interrupt. When disabling underrun by setting ACREG[4] = 1, unknown data is sent over TX line.

Figure 23-54. Auxiliary Control Register (ACREG)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
PULSETYPE	SDMOD	DISIRRX	DISTXUNDERRUN	SENDSIP	SCTXEN	ABORTEN	EOTEN
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-38. Auxiliary Control Register (ACREG) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	PULSETYPE		SIR pulse-width select:
		0	3/16 of baud-rate pulse width
		1	1.6 µs
6	SDMOD		Primary output used to configure transceivers. Connected to the SD/MODE input pin of IrDA transceivers.
		0	SD pin is set to high.
		1	SD pin is set to low.
5	DISIRRX		Disable RX input.
		0	Normal operation (RX input automatically disabled during transmit, but enabled outside of transmit operation).
		1	Disables RX input (permanent state; independent of transmit).
4	DISTXUNDERRUN		Disable TX underrun.
		0	Long stop bits cannot be transmitted. TX underrun is enabled.
		1	Long stop bits can be transmitted.
3	SENDSIP		MIR/FIR modes only. Send serial infrared interaction pulse (SIP).
			If this bit is set during an MIR/FIR transmission, the SIP is sent at the end of it. This bit is automatically cleared at the end of the SIP transmission.
		0	No action.
		1	Send SIP pulse.
2	SCTXEN		Store and control TX start. When MDR1[5] = 1 and the LH writes 1 to this bit, the TX state-machine starts frame transmission. This bit is self-clearing.
1	ABORTEN		Frame abort. The LH can intentionally abort transmission of a frame by writing 1 to this bit. Neither the end flag nor the CRC bits are appended to the frame.
0	EOTEN		EOT (end-of-transmission) bit. The LH writes 1 to this bit just before it writes the last byte to the TX FIFO in the set-EOT bit frame-closing method. This bit is automatically cleared when the LH writes to the THR (TX FIFO).

23.3.28 Supplementary Control Register (SCR)

The supplementary control register (SCR) is shown in Figure 23-55 and described in Table 23-39.

NOTE: Bit 4 enables the wake-up interrupt, but this interrupt is not mapped into the IIR register. Therefore, when an interrupt occurs and there is no interrupt pending in the IIR register, the SSR[1] bit must be checked. To clear the wake-up interrupt, bit SCR[4] must be reset to 0.

Figure 23-55. Supplementary Control Register (SCR)

15							8
Reserved							
R-0							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXTRIGGRANU1	TXTRIGGRANU1	DSRIT	RXCTSDSRWAKEUPENABLE	TXEMPTYCTLIT	DMAMODE2		DMAMODECTL
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-39. Supplementary Control Register (SCR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	RXTRIGGRANU1	0	Disables the granularity of 1 for trigger RX level.
		1	Enables the granularity of 1 for trigger RX level.
6	TXTRIGGRANU1	0	Disables the granularity of 1 for trigger TX level.
		1	Enables the granularity of 1 for trigger TX level.
5	DSRIT	0	Disables $\overline{\text{DSR}}$ interrupt.
		1	Enables $\overline{\text{DSR}}$ interrupt.
4	RXCTSDSRWAKEUPENABLE		RX CTS wake-up enable.
		0	Disables the WAKE UP interrupt and clears SSR[1].
		1	Waits for a falling edge of RX, $\overline{\text{CTS}}$, or $\overline{\text{DSR}}$ pins to generate an interrupt.
3	TXEMPTYCTLIT	0	Normal mode for THR interrupt.
		1	THR interrupt is generated when TX FIFO and TX shift register are empty.
2-1	DMAMODE2	0-3h	Specifies the DMA mode valid if SCR[0] = 1:
		0	DMA mode 0 (no DMA).
		1h	DMA mode 1 (UARTnDMAREQ[0] in TX, UARTnDMAREQ[1] in RX)
		2h	DMA mode 2 (UARTnDMAREQ[0] in RX)
		3h	DMA mode 3 (UARTnDMAREQ[0] in TX)
0	DMAMODECTL	0	The DMAMODE is set with FCR[3].
		1	The DMAMODE is set with SCR[2:1].

23.3.29 Supplementary Status Register (SSR)

The supplementary status register (SSR) is shown in Figure 23-56 and described in Table 23-40.

NOTE: Bit 1 is reset only when SCR[4] is reset to 0.

Figure 23-56. Supplementary Status Register (SSR)

15				8
Reserved				
R-0				
7	3	2	1	0
Reserved		DMACOUNTERRST	RXCTSDSRWAKEUPSTS	TXFIFOFULL
R-0		R/W-1	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-40. Supplementary Status Register (SSR) Field Descriptions

Bit	Field	Value	Description
15-3	Reserved	0	Reserved.
2	DMACOUNTERRST	0	The DMA counter will not be reset, if the corresponding FIFO is reset (via FCR[1] or FCR[2]).
		1	The DMA counter will be reset, if the corresponding FIFO is reset (via FCR[1] or FCR[2]).
1	RXCTSDSRWAKEUPSTS		Pin falling edge detection: Reset only when SCR[4] is reset to 0.
		0	No falling-edge event on RX, $\overline{CTS}$, and $\overline{DSR}$.
		1	A falling edge occurred on RX, $\overline{CTS}$, or $\overline{DSR}$.
0	TXFIFOFULL	0	TX FIFO is not full.
		1	TX FIFO is full.

23.3.30 BOF Length Register (EBLR)

In IrDA SIR operation, the BOF length register (EBLR) specifies the number of BOF + xBOFs to transmit. The value set into this register must consider the BOF character; therefore, to send only one BOF with no XBOF, this register must be set to 1. To send one BOF with *n* XBOFs, this register must be set to *n* + 1. Furthermore, the value 0 sends 1 BOF plus 255 XBOFs.

In IrDA MIR mode, the BOF length register (EBLR) specifies the number of additional start flags (MIR protocol mandates a minimum of 2 start flags).

In CIR mode, the BOF length register (EBLR) specifies the number of consecutive zeros to be received before generating the RXSTOP interrupt (IIR[2]). All the received zeros are stored in the RX FIFO. When the register is cleared to 0, this feature is deactivated and always in reception state, which is disabled by setting the ACREG[5] bit to 1.

The BOF length register (EBLR) is shown in Figure 23-57 and described in Table 23-41.

NOTE: If the RX_STOP interrupt occurs before a byte boundary, the remaining bits of the last byte are filled with zeros and then passed into the RX FIFO.

Figure 23-57. BOF Length Register (EBLR)

15	8	7	0
Reserved		EBLR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-41. BOF Length Register (EBLR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	EBLR		IrDA mode: This register allows definition of up to 176 xBOFs, the maximum required by IrDA specification. CIR mode: This register specifies the number of consecutive zeros to be received before generating the RXSTOP interrupt (IIR[2]).
		0	Feature disabled.
		1h	Generate RXSTOP interrupt after receiving 1 zero bit.
		...	
		FFh	Generate RXSTOP interrupt after receiving 255 zero bits.

23.3.31 Module Version Register (MVR)

The reset value is fixed by hardware and corresponds to the RTL revision of this module. A reset has no effect on the value returned. The module version register (MVR) is shown in Figure 23-58 and described in Table 23-42.

Figure 23-58. Module Version Register (MVR)

15	8	7	4	3	0
Reserved		MAJORREV		MINORREV	
R-0		R-unknown		R-unknown	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-42. Module Version Register (MVR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-4	MAJORREV	0-Fh	Major revision number of the module.
3-0	MINORREV	0-Fh	Minor revision number of the module.

23.3.32 System Configuration Register (SYSC)

The AUTOIDLE bit controls a power-saving technique to reduce the logic power consumption of the module interface; that is, when the feature is enabled, the interface clock is gated off until the module interface is accessed. When the SOFTRESET bit is set high, it causes a full device reset. The system configuration register (SYSC) is shown in Figure 23-59 and described in Table 23-43.

Figure 23-59. System Configuration Register (SYSC)

15	5	4	3	2	1	0
Reserved		IDLEMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE
R-0		R/W-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-43. System Configuration Register (SYSC) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	Reserved.
4-3	IDLEMODE	0-3h	Power management req/ack control.
		0	Force idle: Idle request is acknowledged unconditionally.
		1h	No-idle: Idle request is never acknowledged.
		2h	Smart idle: Acknowledgement to an idle request is given based in the internal activity of the module.
		3h	Smart idle Wakeup: Acknowledgement to an idle request is given based in the internal activity of the module. The module is allowed to generate wakeup request.
2	ENAWAKEUP		Wakeup control.
		0	Wakeup is disabled.
		1	Wakeup capability is enabled.
1	SOFTRESET		Software reset. Set this bit to 1 to trigger a module reset. This bit is automatically reset by the hardware. Read returns 0.
		0	Normal mode.
		1	Module is reset.
0	AUTOIDLE		Internal interface clock-gating strategy.
		0	Clock is running.
		1	Automatic interface clock-gating strategy is applied based on interface activity.

23.3.33 System Status Register (SYSS)

The system status register (SYSS) is shown in Figure 23-60 and described in Table 23-44.

Figure 23-60. System Status Register (SYSS)

15	1	0
Reserved		RESETDONE
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-44. System Status Register (SYSS) Field Descriptions

Bit	Field	Value	Description
15-1	Reserved	0	Reserved.
0	RESETDONE		Internal reset monitoring.
		0	Internal module reset is ongoing.
		1	Reset complete.

23.3.34 Wake-Up Enable Register (WER)

The wake-up enable register (WER) is used to mask and unmask a UART event that subsequently notifies the system. An event is any activity in the logic that can cause an interrupt and/or an activity that requires the system to wake up. Even if wakeup is disabled for certain events, if these events are also an interrupt to the UART, the UART still registers the interrupt as such. The wake-up enable register (WER) is shown in Figure 23-61 and described in Table 23-45.

Figure 23-61. Wake-Up Enable Register (WER)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
TXWAKEUPEN	RLS_INTERRUPT	RHR_INTERRUPT	RX_ ACTIVITY	DCD_ACTIVITY	RI_ ACTIVITY	DSR_ACTIVITY	CTS_ACTIVITY
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-45. Wake-Up Enable Register (WER) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	TXWAKEUPEN		Wake-up interrupt.
		0	Event is not allowed to wake up the system.
		1	Event can wake up the system: Event can be: THRIT or TXDMA request and/or TXSATUSIT.
6	RLS_ INTERRUPT		Receiver line status interrupt.
		0	Event is not allowed to wake up the system.
		1	Event can wake up the system.
5	RHR_ INTERRUPT		RHR interrupt.
		0	Event is not allowed to wake up the system.
		1	Event can wake up the system.
4	RX_ ACTIVITY	0	Event is not allowed to wake up the system.
		1	Event can wake up the system.
3	DCD_ACTIVITY	0	Event is not allowed to wake up the system.
		1	Event can wake up the system.
2	RI_ ACTIVITY	0	Event is not allowed to wake up the system.
		1	Event can wake up the system.
1	DSR_ACTIVITY	0	Event is not allowed to wake up the system.
		1	Event can wake up the system.
0	CTS_ ACTIVITY	0	Event is not allowed to wake up the system.
		1	Event can wake up the system.

23.3.35 Carrier Frequency Prescaler Register (CFPS)

Since the consumer IR (CIR) works at modulation rates of 30–56.8 kHz, the 48 MHz clock must be prescaled before the clock can drive the IR logic. The carrier frequency prescaler register (CFPS) sets the divisor rate to give a range to accommodate the remote control requirements in BAUD multiples of 12×. The value of the CFPS at reset is 105 decimal (69h), which equates to a 38.1 kHz output from starting conditions. The 48 MHz carrier is prescaled by the CFPS that is then divided by the 12× BAUD multiple. The carrier frequency prescaler register (CFPS) is shown in Figure 23-62 and described in Table 23-46.

Figure 23-62. Carrier Frequency Prescaler Register (CFPS)

15	8	7	0
Reserved		CFPS	
R-0		R/W-69h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-46. Carrier Frequency Prescaler Register (CFPS) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	CFPS	0-FFh	System clock frequency prescaler at (12× multiple). CFPS = 0 is not supported. Examples for CFPS values:

Target Frequency (kHz)	CFPS (decimal)	Actual Frequency (kHz)
30	133	30.08
32.75	122	32.79
36	111	36.04
36.7	109	36.69
38	105	38.1
40	100	40
56.8	70	57.14

23.3.36 Divisor Latches Low Register (DLL)

The divisor latches low register (DLL) with the DLH register stores the 14-bit divisor for generation of the baud clock in the baud rate generator. DLH stores the most-significant part of the divisor, DLL stores the least-significant part of the divisor. The DLL register is shown in Figure 23-63 and described in Table 23-47.

NOTE: DLL and DLH can be written to only before sleep mode is enabled (before IER[4] is set).

Figure 23-63. Divisor Latches Low Register (DLL)

15	8	7	0
Reserved		CLOCK_LSB	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-47. Divisor Latches Low Register (DLL) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	CLOCK_LSB	0-FFh	Divisor latches low. Stores the 8 LSB divisor value.

23.3.37 Divisor Latches High Register (DLH)

The divisor latches high register (DLH) with the DLL register stores the 14-bit divisor for generation of the baud clock in the baud rate generator. DLH stores the most-significant part of the divisor, DLL stores the least-significant part of the divisor. The DLH register is shown in Figure 23-64 and described in Table 23-48.

NOTE: DLL and DLH can be written to only before sleep mode is enabled (before IER[4] is set).

Figure 23-64. Divisor Latches High Register (DLH)

15	6	5	0
Reserved		CLOCK_MSB	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-48. Divisor Latches High Register (DLH) Field Descriptions

Bit	Field	Value	Description
15-6	Reserved	0	Reserved.
5-0	CLOCK_MSB	0-3Fh	Divisor latches high. Stores the 6 MSB divisor value.

23.3.38 Enhanced Feature Register (EFR)

The enhanced feature register (EFR) enables or disables enhanced features. Most enhanced functions apply only to UART modes, but EFR[4] enables write accesses to FCR[5:4], the TX trigger level, which is also used in IrDA modes. The enhanced feature register (EFR) is shown in Figure 23-65 and described in Table 23-49.

Figure 23-65. Enhanced Feature Register (EFR)

15					8
Reserved					
R-0					
7	6	5	4	3	0
AUTOCTSEN	AUTORTSEN	SPECIALCHARDETECT	ENHANCEDEN	SWFLOWCONTROL	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-49. Enhanced Feature Register (EFR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	AUTOCTSEN		Auto-CTS enable bit (UART mode only).
		0	Normal operation.
		1	Auto-CTS flow control is enabled; transmission is halted when the $\overline{CTS}$ pin is high (inactive).
6	AUTORTSEN		Auto-RTS enable bit (UART mode only).
		0	Normal operation.
		1	Auto-RTS flow control is enabled; $\overline{RTS}$ pin goes high (inactive) when the receiver FIFO HALT trigger level, TCR[3:0], is reached and goes low (active) when the receiver FIFO RESTORE transmission trigger level is reached.
5	SPECIALCHARDETECT		Special character detect (UART mode only).
		0	Normal operation.
		1	Special character detect enable. Received data is compared with XOFF2 data. If a match occurs, the received data is transferred to RX FIFO and the IIR[4] bit is set to 1 to indicate that a special character was detected.
4	ENHANCEDEN		Enhanced functions write enable bit.
		0	Disables writing to IER[7:4], FCR[5:4], and MCR[7:5].
		1	Enables writing to IER[7:4], FCR[5:4], and MCR[7:5].
3-0	SWFLOWCONTROL	0-Fh	Combinations of software flow control can be selected by programming this bit. XON1 and XON2 should be set to different values if the software flow control is enabled. See Table 23-50. In IrDA mode, EFR[1:0] selects the IR address to check. See *IR Address Checking*.

Table 23-50. EFR[3:0] Software Flow Control Options

EFR[3:0]				
Bit 3	Bit 2	Bit 1	Bit 0	TX/RX Software Flow Control
0	0	X	X	No transmit flow control
1	0	X	X	Transmit XON1, XOFF1
0	1	X	X	Transmit XON2, XOFF2
1	1	X	X	Transmit XON1, XON2: XOFF1, XOFF2(1)
X	X	0	0	No receive flow control
X	X	1	0	Receiver compares XON1, XOFF1
X	X	0	1	Receiver compares XON2, XOFF2
X	X	1	1	Receiver compares XON1, XON2: XOFF1, XOFF2(1)

(1) The XON1 and XON2 characters or the XOFF1 and XOFF2 characters must be transmitted/received sequentially with XON1/XOFF1 followed by XON2/XOFF2.

23.3.39 XON1/ADDR1 Register

In UART mode, XON1 character; in IrDA mode, ADDR1 address 1. The XON1/ADDR1 register is shown in Figure 23-66 and described in Table 23-51.

Figure 23-66. XON1/ADDR1 Register

15	8	7	0
Reserved		XONWORD1	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-51. XON1/ADDR1 Register Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	XONWORD1	0-FFh	Stores the 8-bit XON1 character in UART modes and ADDR1 address 1 in IrDA modes.

23.3.40 XON2/ADDR2 Register

In UART mode, XON2 character; in IrDA mode, ADDR2 address 2. The XON2/ADDR2 register is shown in Figure 23-67 and described in Table 23-52.

Figure 23-67. XON2/ADDR2 Register

15	8	7	0
Reserved		XONWORD2	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-52. XON2/ADDR2 Register Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
5	XONWORD2	0-FFh	Stores the 8-bit XON2 character in UART modes and ADDR2 address 2 in IrDA modes.

23.3.41 XOFF1 Register

In UART mode, XOFF1 character. The XOFF1 register is shown in Figure 23-68 and described in Table 23-53.

Figure 23-68. XOFF1 Register

15	8	7	0
Reserved		XOFFWORD1	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-53. XOFF1 Register Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	XOFFWORD1	0	Stores the 8-bit XOFF1 character in UART modes.

23.3.42 XOFF2 Register

In UART mode, XOFF2 character. The XOFF2 register is shown in Figure 23-69 and described in Table 23-54.

Figure 23-69. XOFF2 Register

15	8	7	0
Reserved		XOFFWORD2	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-54. XOFF2 Register Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	XOFFWORD2	0-FFh	Stores the 8-bit XOFF2 character in UART modes.

23.3.43 Transmit Frame Length Low Register (TXFLL)

The transmit frame length low register (TXFLL) and the TXFLH register hold the 13-bit transmit frame length (expressed in bytes). TXFLL holds the LSBs and TXFLH holds the MSBs. The frame length value is used if the frame length method of frame closing is used. The transmit frame length low register (TXFLL) is shown in Figure 23-70 and described in Table 23-55.

Figure 23-70. Transmit Frame Length Low Register (TXFLL)

15	8	7	0
Reserved		TXFLL	
R-0		W-0	

LEGEND: R/W = Read/Write; W = Write only; -*n* = value after reset

Table 23-55. Transmit Frame Length Low Register (TXFLL) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	TXFLL	0-FFh	LSB register used to specify the frame length.

23.3.44 Transmit Frame Length High Register (TXFLH)

The transmit frame length high register (TXFLH) and the TXFLL register hold the 13-bit transmit frame length (expressed in bytes). TXFLL holds the LSBs and TXFLH holds the MSBs. The frame length value is used if the frame length method of frame closing is used. The transmit frame length high register (TXFLH) is shown in Figure 23-71 and described in Table 23-56.

Figure 23-71. Transmit Frame Length High Register (TXFLH)

15	5	4	0
Reserved		TXFLH	
R-0		W-0	

LEGEND: R/W = Read/Write; W = Write only; -*n* = value after reset

Table 23-56. Transmit Frame Length High Register (TXFLH) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	Reserved.
4-0	TXFLH	0-1Fh	MSB register used to specify the frame length.

23.3.45 Received Frame Length Low Register (RXFLL)

The received frame length low register (RXFLL) and the RXFLH register hold the 12-bit receive maximum frame length. RXFLL holds the LSBs and RXFLH holds the MSBs. If the intended maximum receive frame length is *n* bytes, program RXFLL and RXFLH to be *n* + 3 in SIR or MIR modes and *n* + 6 in FIR mode (+3 and +6 are the result of frame format with CRC and stop flag; two bytes are associated with the FIR stop flag). The received frame length low register (RXFLL) is shown in Figure 23-72 and described in Table 23-57.

Figure 23-72. Received Frame Length Low Register (RXFLL)

15	8	7	0
Reserved		RXFLL	
R-0		W-0	

LEGEND: R/W = Read/Write; W = Write only; -*n* = value after reset

Table 23-57. Received Frame Length Low Register (RXFLL) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	RXFLL	0-FFh	LSB register used to specify the frame length in reception.

23.3.46 Received Frame Length High Register (RXFLH)

The received frame length high register (RXFLH) and the RXFLL register hold the 12-bit receive maximum frame length. RXFLL holds the LSBs and RXFLH holds the MSBs. If the intended maximum receive frame length is *n* bytes, program RXFLL and RXFLH to be *n* + 3 in SIR or MIR modes and *n* + 6 in FIR mode (+3 and +6 are the result of frame format with CRC and stop flag; two bytes are associated with the FIR stop flag). The received frame length high register (RXFLH) is shown in Figure 23-73 and described in Table 23-58.

Figure 23-73. Received Frame Length High Register (RXFLH)

15	4	3	0
Reserved		RXFLH	
R-0		W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-58. Received Frame Length High Register (RXFLH) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved	0	Reserved.
3-0	RXFLH	0-Fh	MSB register used to specify the frame length in reception.

23.3.47 UART Autobauding Status Register (UASR)

The UART autobauding status register (UASR) returns the speed, the number of bits by characters, and the type of parity in UART autobauding mode. In autobauding mode, the input frequency of the UART modem must be fixed to 48 MHz. Any other module clock frequency results in incorrect baud rate recognition. The UART autobauding status register (UASR) is shown in Figure 23-74 and described in Table 23-59.

NOTE:

- This register is used to set up transmission according to characteristics of previous reception, instead of LCR, DLL, and DLH registers when UART is in autobauding mode.
- To reset the autobauding hardware (to start a new “AT” detection) or to set the UART in standard mode (no autobaud), MDR1[2:0] must be set to 7h (reset state), then set to 2h (UART in autobaud mode) or cleared to 0 (UART in standard mode).

Usage limitation:

- Only 7 and 8 bits character (5 and 6 bits not supported).
- 7 bits character with space parity not supported.
- Baud rate between 1200 and 115 200 bp/s (10 possibilities).

Figure 23-74. UART Autobauding Status Register (UASR)

15	8	7	6	5	4	0
Reserved		PARITYTYPE		BITBYCHAR	SPEED	
R-0		R-0		R-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-59. UART Autobauding Status Register (UASR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-6	PARITYTYPE	0-3h	Type of the parity in UART autobauding mode.
		0	No parity identified
		1h	Parity space
		2h	Even parity
		3h	Odd parity
5	BITBYCHAR		Number of bits by characters.
		0	7-bit character identified.
		1	8-bit character identified.
4-0	SPEED	0-1Fh	Speed
		0	No speed identified.
		1h	115 200 baud
		2h	57 600 baud
		3h	38 400 baud
		4h	28 800 baud
		5h	19 200 baud
		6h	14 400 baud
		7h	9600 baud
		8h	4800 baud
		9h	2400 baud
		Ah	1200 baud
		Bh-1Fh	Reserved

Universal Serial Bus (USB)

The universal serial bus (USB) is an external serial bus supporting a data transfer speed rates ranging up to 480 Mbits/second. The device supports a dual port USB controller with both ports complying with USB2.0 standard. Both ports support dual role functionality and as such they are both configurable as Host or Device. Even though both ports are made of OTG Controllers, OTG feature is not supported. The two ports operate independent of each other.

Topic **Page**

24.1 Introduction

24.1.1 Overview

The USB controller provides a low-cost connectivity solution for numerous consumer portable devices by providing a mechanism for data transfer between USB devices with a line/bus speed up to 480 Mbps. The device USB subsystem has two independent USB 2.0 Modules built around two OTG controllers. Even though the controllers are OTG controllers, OTG features are not supported by any of the ports. Each port has the support for a dual-role feature allowing for additional versatility enabling operation capability as a host or peripheral. Both ports have identical capabilities and operate independent of each other.

Each USB controller is built around the Mentor USB OTG controller (musbmhdrc) and Synopsys SR70LX PHY. Each USB controller has user configurable 32K Bytes of Endpoint FIFO and has the support for 15 'Transmit' endpoints and 15 'Receive' endpoints in addition to Endpoint 0. The USB subsystem makes use of the CPPI 4.1 DMA for accelerating data movement via a dedicated DMA hardware.

The two USB modules share the CPPI DMA controller and accompanying Queue Manager, Interrupt Pacer, Power Management module, and PHY/UTMI clock.

Within the descriptions of the USB subsystem that would follow, the term USB controller or USB PHY is used to mean/refer to any of the two USB controllers or PHYs existing within the USB subsystem. The term USB module is used to mean/refer to any of the two USB modules. USB0(1) is used to refer to one of the two USB Modules.

24.1.2 Features Supported

The main features of the USB Subsystem are:

- Has two USB modules where each USB module is built around a mentor USB controller and Synopsys PHY.
- Includes SR70LX Synopsys USB 2.0 OTG nanoPHY per module in C014.P process.
- Supports USB 2.0 device operations at HS (480 Mb/s) and FS (12 Mb/s).
- Supports USB 2.0 Host operations at HS (480 Mb/s), FS (12 Mb/s) and LS (1.5Mb/s).
- Supports all modes of transfers (control, bulk, interrupt, and isochronous) in both host and device mode.
- Supports high bandwidth ISO mode.
- Supports 15 Transmit (Tx) and 15 Receive (Rx) Endpoints including Endpoint 0.
- Includes a 32KB Endpoint FIFO RAM per Module (64Kbytes per USB Subsystem) with user programmable FIFO sizes.
- Includes RNDIS mode for accelerating RNDIS type protocols using short packet termination over USB.
- Includes CDC Linux mode for accelerating CDC type protocols using short packet termination over USB.
- Includes Generic RNDIS mode, an RNDIS like mode for terminating RNDIS type protocols without using short packet termination for support for MSC class applications.
- Includes a CPPI 4.1 compliant DMA controller sub-module with 30 Rx and 30 Tx simultaneous data connections/channels.
- Includes CPPI 4.1 DMA scheduler.
- DMA supports host and packet descriptors formats.
- DMA supports stall on buffer starvation.
- DMA supports capability of allocating data buffer sizes up to 4Mbytes per single transfer.
- Provides a CPPI Queue Manager Module with 156 Queues for Queueing and De-Queueing packets.
- Supports DMA pacing logic for interrupts.
- Supports Loopback MGC test using at the UTMI interface level.

24.1.3 Features Not Supported

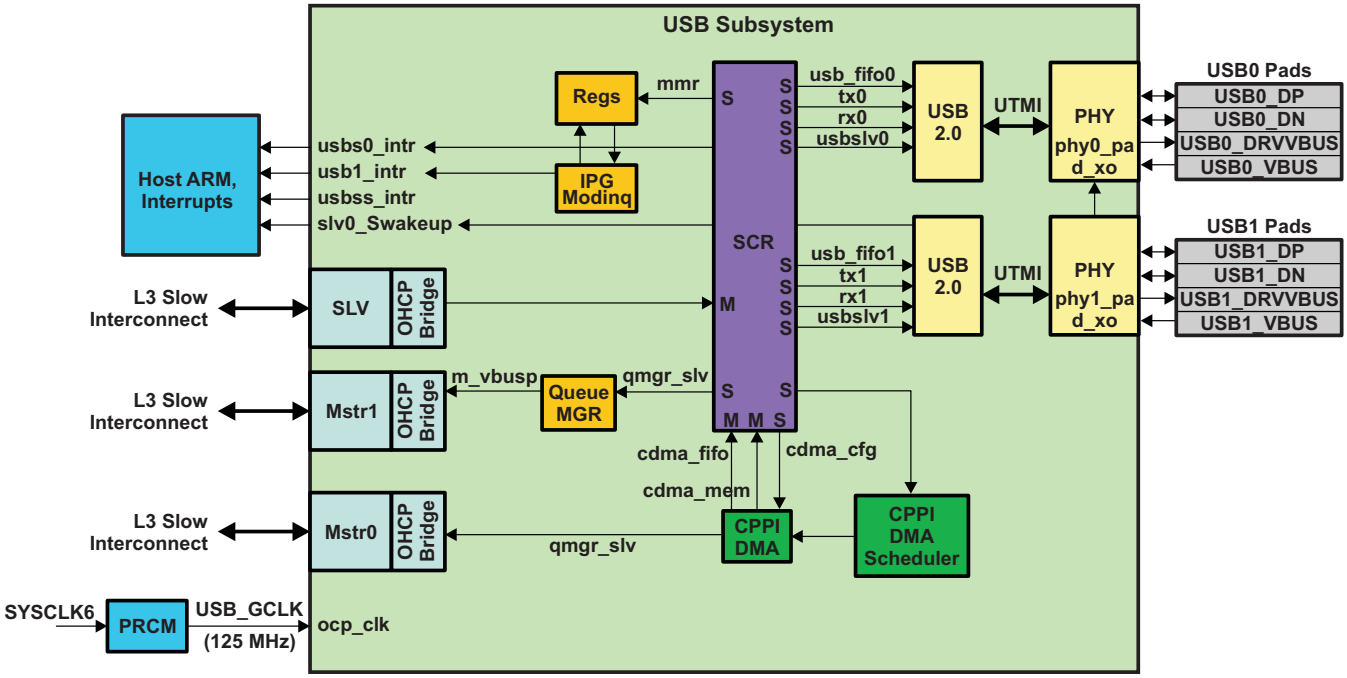
- USB2.0 extensions for OTG Session Request Protocol (SRP)
- USB2.0 extensions for OTG Host Request Protocol (HRP)

NOTE: The USB subsystem is built around two USB OTG controllers. However, OTG functionalities are not supported. What this means is that the USB Modules support mini-host and device capabilities but is not capable of changing roles (device/host) on the fly.

24.1.4 Functional Block Diagram

Figure 1 displays a high-level functional block diagram covering the major blocks of the USB subsystem. It captures the general data paths and displays the interaction between each blocks of the subsystem at a high level.

Figure 24-1. USB Subsystem Block Diagram



24.1.4.1 L3 Slow Interconnect

This is the OCP interface to the MMRs and system resource. CPPI DMA and Queue Manager have also their own dedicated interface to the system resource through the master interfaces. Cortex-A8 accesses the USB peripheral through the save OCP Interface.

24.1.4.2 Reg

The USB register file (USB core and subsystem registers and CPPI DMA registers are accessible to the Cortex-A8) reside in a block within the USBSS.

24.1.4.3 USB2.0 Controller

The USB subsystem supports two independent USB Modules. Each USB module comprises of a Dual Role USB2.0 compliant Mentor OTG controller without the support for OTG functionality.

24.1.4.4 USB PHY

The USB subsystem is made of two USB modules. Each USB module comprises a Synopsys Designware core USB 2.0 PHY. The PHY is connected to the USB controller via 8-bit UTMI Interface. The PHY does not have a built-in charge pump and an external power source is required to source the 5V VBUS power when operating as a Host.

24.1.4.5 USB Pads

The USB subsystem external interface signals comprises two differential data lines, two Voltage Buses (VBUS), and two USB DRVVBUS signals (enable/disable external 5V Power Source). No USB ID pad exists. The Register field that is user software controlled is used as a replacement for the ID pad usage. For more details about the USB0 and USB1 mode registers, see Section 24.9.2.1.22 and Section 24.9.2.2.22, respectively.

24.1.4.6 DMA Resources

The CPPI DMA alongside with the Queue Manager and the Scheduler are used to perform data transfer for Endpoints 1 to 15.

24.1.4.7 Interrupts

There are three interrupt interfaces: one for the USB subsystem, one for the USB0 controller, and one for the USB1 controller. All interrupts for that specific unit are aggregated into one interrupt interface.

Each interrupt has two basic registers: STATUS and ENABLE. Hardware can set the register via internal logic. Software can set or clear the register by accessing the appropriate locations in the memory map. The enabling (or masking) of the interrupt is controlled via software. Writing 1's (WR1) to either the IRQENABLE_SET or IRQENABLE_CLR addresses will enable or disable the interrupt respectively. An EOI function exists to re-enable the detection of active interrupts. If the system attempts to write to any of the interrupt registers, any still active interrupts will trigger the EOI output signal. The EOI input signal indicates when external logic requires the USB to re-evaluate its pending sources and send another pulse. For more information see section "Interrupt Support."

Note: When generating an interrupt by writing to one of the IRQ_ENABLE_SET registers; the interrupt can occur several cycles before the OCP write status complete has occurred.

24.1.5 Supported Use Case(s)

The USB subsystem supports two USB OTG modules that are capable of operating as a mini-host and device. OTG features are not supported.

24.1.6 Industry Standard(s) Compliance

The USB subsystem complies with the following standards:

- Complies with the USB 2.0 standard for high-speed (480 Mbps) functions and with the On-The-Go supplement to the USB 2.0 Specification. However it does not support OTG functionality.

24.1.7 Non-Industry Standard(s)

The USB subsystem complies with the following proprietary standards:

- Mentor Dual Role Controller (MUSBMHDRC) – USB Controller
- SR70LX – Physical Layer: Synopsys DesignWare Core USB 2.0 nanoPHY
- TI CPPI 4.1 – Built in 1st party DMA and Structure

24.1.8 Terminology Used in This Document

The following is a brief explanation of some terms that are used in this document:

AHB— Advanced High-Performance Bus

CBA— Common Bus Architecture

CDC— Change Data Capture

CPPI— Communications Port Programming Interface

DFT— Design for Test

DMA— Direct Memory Access

DV— Design Verification

EOI— End of Interrupt

EOP— End of Packet

FIFO— First-In First-Out

FS— Full-Speed USB data rate

HNP— Host Negotiation

HS— High-Speed USB data rate

INTD— Interrupt Distributor

IP— Intellectual Property

ISO— Isochronous transfer type

LS— Low-Speed USB data rate

MP— Middle of Packet

OCP— Open Core Protocol

OCP HD— OCP High Performance

OCP MMR— OCP Memory-Mapped Registers

OTG— On-the-Go

PDR— Physical Design Requirements

PHY— Physical Layer Device

PPU— Packet Processing Unit

RNDIS— Remote Network Driver Interface Specification

RX— Receive

SCR— Switched Central Resource

SOC— System On a Chip

SOP— Start of Packet

SRP— Session Resume

TX— Transmit

USB— Universal Serial Bus

USB0— One of the two USB 2.0-Compliant USB Module

USB1— One of the two USB2.0-Compliant USB Module

USBSS— USB Subsystem (contains USB0 and USB1)

UTMI— USB 2.0 Transceiver Macrocell Interface

XDMA— Transfer DMA (DMA other than CPPI DMA used within the Controller)

24.2 Architecture

The USB controller within the device supports 15 Transmit endpoints and 15 Receive endpoints in addition to Endpoint 0. (The use of these endpoints for IN and OUT transactions depends on whether the USB controller is being used as a device/peripheral or as a host. When used as a peripheral, IN transactions are processed through TX endpoints and OUT transactions are processed through Rx endpoints. When used as a host, IN transactions is processed through Rx endpoints and OUT transactions is processed through TX endpoints.)

These additional endpoints can be individually configured in software to handle either Bulk transfers (which also allows them to handle Interrupt transfers), Isochronous transfers or Control transfers. Further, the endpoints can also be allocated to different target device functions on the fly – maximizing the number of devices that can be simultaneously supported.

Each endpoint requires a chunk of memory allocated within the FIFO to be associated with it. The USB module has a single block of FIFO RAM (32 Kbytes in size) to be shared by all endpoints. The FIFO for Endpoint 0 is required to be 64 bytes deep and will buffer 1 packet. The first 64 Bytes of the FIFO RAM is reserved by hardware for Endpoint 0 usage and for this reason user software does not need (nor has the capability) to allocate FIFO for Endpoint 0. The rest of the FIFO RAM is user configurable with regard to the other endpoint FIFOs, which may be from 8 to 8192 bytes in size and can buffer either 1 or 2 packets.

Separate FIFOs may be associated with each endpoint: alternatively a TX endpoint and the Rx endpoint with the same Endpoint number can be configured to use the same FIFO, for example to reduce the size of RAM block needed, provided they can never be active at the same time.

The role (host or device) that the USB controller assumes is chosen by user firmware programming the respective USB controller MODE register defined within the USB subsystem space. Note that some USB pins have not been bonded out and the functions of these pins are controlled by user software via dedicated registers.

The user has access to the controller through the OCP slave interface via the CPU. The user can process USB transactions entirely from the CPU or can also use the DMA to perform data transfer. The CPPI DMA can be used to service Endpoints 1 to 15 not Endpoint 0. CPU access method is used to service Endpoint 0 transactions.

24.2.1 Clock Control

Two main clocks are used to by the USB subsystem, the OCP clock and the UTMI/PHY clock.

The OCP clock (SYSCLK6), also referred as the System Clock, is the clock that is used to clock the controller and all other blocks of the USB subsystem with the exception of the PHY. This clock is controlled by the PRCM. A clock rate of 60 MHz or greater must be used to guarantee core operation can deliver full USB 2.0 bandwidth (480 Mb/s). In other words, SYSCLK6 should not be less than 60 MHz. SYSCLK6 is derived from the main PLL.

The UTMI/PHY clock is derived from a 24 MHz clock sourced to the PHY. PHY internal PLL will derive the necessary bit clock and 60MHz UTMI clock

24.2.2 Signal Descriptions

The USB subsystem external interface signals are displayed within Table 24-1.

Table 24-1. USBSS Interface Signals

Pin (x=0/1)	Type	Description
USBx_DP USBx_DN	I/O	USBx data differential pair
USBx_DRVVBUS	O	USBx VBUS supply control
USBx_VBUS	I	USBx VBUS (input only for voltage sensing)

24.2.3 VBUS Voltage Sourcing Control

When any of the USB controllers assumes the role of a host, the USB is required to supply a 5V power source to an attached device through its VBUS line. In order to achieve this task, the USB controller requires the use of an external power logic (or charge pump) capable of sourcing 5V power. A USB_DRVVBUS is used as a control signal to enable/disable this external power logic to either source or disable power on the VBUS line. The control on the USB_DRVVBUS is automatic and is handled by the USB controller. The control should be transparent to the user so long as the proper hardware connection and software initialization are in place. The USB controller drives the USB_DRVVBUS signal high when it assumes the role of a host while the controller is in session. When assuming the role of a device, the controller drives the USB_DRVVBUS signal low disabling the external charge pump/power logic; hence, no power is driven on the VBUS line (in this case, power is expected to be sourced by the external host).

Note that both USBs are self-powered and the device does not rely on the voltage on the VBUS line sourced by an external host for controller operation when assuming the role of a device. The power on the VBUS is used to identify the presence of a Host. It is also used to power up the pull-up on the D+ line. The USB PHY would continually monitor the voltage on the VBUS and report the status to USB controller.

24.2.4 Pull-up/Pull-Down Resistors

Since the USB controllers are dual role controllers, capable of assuming a role of a host or device, the necessary required pull-up/pull-down resistors can not exist external to the device. These pull-up/pull-down resistors exist internal to the device, within the PHY to be more specific, and are enabled and disabled based on the role the controller assumes allowing dynamic hardware configuration.

When assuming the role of a host, the data lines are pulled low by the PHY enabling the internal 15KOhms resistors. When assuming the role of a device the required 1.5Kohm pull-up resistor on the D+ line is enabled automatically to signify the USB capability to the external host as a FS device (HS operation is negotiated during reset bus condition).

24.2.5 Role Assuming Method

For an OTG controller, the usual method followed by the controller to assume the role of a host or a device is governed by the state of the ID pin which in turn is controlled by the USB cable connector type. Since no ID pin signal has been bonded out on the device, there exists a register field within the USB subsystem register file that is programmed by the user indirectly controlling the state of the ID pin and ultimately selecting the role of the USB controller.

Two registers, USB0 Mode Register at offset 10E8h and USB1 Mode Register at offset 18E8h, are used for a user to select the role the USB controller assumes. The user is required to program the corresponding register prior to the USB controller is in session.

24.2.6 USB Signal Conditioning

The device supports a built in logic to condition the signal quality of the transmit signal by adjusting transmit parameters including transmit level, pre-emphasis, etc using the USB_CTRL0/1 Registers.

24.2.7 USB PHY Initialization

Prior to configuring the USB Module Registers, release the USB SubSystem and PHY from reset, enable the interconnect and controller clocks, and as configure PHY values. Registers related for this task are captured within this doc for ease of reference. Only related information for this task is included within this document. The actual program values are shown at the end.

The USB registers within Device Configuration register space are used to control the PHY. The Cortex-A8 CPU must be in Supervisory Mode when accessing the USB related registers within the Device Configuration Spaces. The registers used to control the PHY are USB_CTRL, USBPHY_CTRL0, and USBPHY_CTRL1. See USB Related Clock/PHY Control Registers for the definitions of these registers. Consult the device manual for more information.

```
*0x48180B10 &= 0xFFFFFF9F; // RSTCTRL (Release USB Module from Reset)
while ((0x48180B14 & 0x00000060)>>5)!=0x3); // Wait until Modules come out Reset
*0x48180B14 |= 0x00000060; // RSTST (Clear Reset Presence)
*0x48180514 |= 0x2; // L3_SLOW_CLKSTCTRL (Enable Interconnect Clock)
*0x48180558 |= 0x2; // USB_CLKCTRL (Enable USB OCP Clock)
while(((*0x48180558) & 0x70000)>>16)==0x7); // Wait until USB Module is Ready

// Configure Access Capability to be in Supervisory Mode
*0x48140620 =0x0000_0003 //USB_CTRL Use PLL Ref Clock, Wake USB PHY1/00

USBPHY_CTRL0 default configuration is good as is and works with out the need for
adjusting parameters for most boards (boards with good h/w design).

USBPHY_CTRL1 default configuration is good as is and works with out the need for
adjusting parameters for most boards (boards with good h/w design).

// USB0 Controller Role Assumption Configuration
*0x4740_10E8 = 0x0000_0000. //A-Type, Normal Phy Mode, Normal Loopback Mode
*0x4740_10E8 = 0x0000_0100. //B-Type, Normal Phy Mode, Normal Loopback Mode

// USB1 Controller Role Assumption Configuration
*0x4740_18E8 = 0x0000_0000. //A-Type, Normal Phy Mode, Normal Loopback Mode
*0x4740_18E8 = 0x0000_0100. //B-Type, Normal Phy Mode, Normal Loopback Mode
```

24.2.8 Indexed and Non-Indexed Register Spaces

The USB controller provides two mechanisms for accessing endpoint control and status registers; Indexed and Non-Indexed method.

Indexed Endpoint Control/Status Register Space. This register space can be thought of as a region populated with Proxy Registers. This register space is a memory-mapped region at offset 1410h to 141Fh for USB0 and 1C10h to 1C1Fh for USB1. The endpoint register space mapped at this region is selected by programming the corresponding INDEX register (@ offsets 140Eh and 1C0Eh for USB0 and USB1 respectively) of the controller.

By programming the INDEX register with the corresponding endpoint number, the control and status register corresponding to that particular endpoint is accessible from this Indexed Region. In other words, Index Register region behaves as a proxy to access a selected endpoint registers.

Non-indexed Endpoint Control/Status Register Space. These regions are dedicated endpoint registers memory-mapped residing at offsets 1500h to 16FFh for USB0 and 1D00h to 1EFFh for USB1. Registers at offset 1500h to 150Fh belong to Endpoint 0; registers at offset 1510h to 151Fh belong to Endpoint 1… and lastly registers at offset 16F0h to 16FFh belong to Endpoint 15 for USB0. Similarly registers at offset 1D00h to 1E0Fh belong to Endpoint 0; registers at offset 1D10h to 1D1Fh belong to Endpoint 1… and finally registers at offset 1DF0h to 1DFFh belong to Endpoint 15 for USB1.

This allows the user/firmware to access an endpoint register region directly from its unique space, as specified within a non-indexed region or from a proxy space by programming the corresponding Index register.

Note: The control and status registers/regions apply to both Tx and Rx of a given Endpoint. That is, Endpoint 1 Tx and Endpoint 1 Rx registers occupy the same register space.

For detailed information about the USB controller registers, see Section 24.9.

24.2.9 Dynamic FIFO Sizing

Each USB module supports a total of 32K bytes of FIFO RAM to dynamically allocate FIFO to all endpoints in use. Each endpoint requires a FIFO to be associated with it. There is one set of register per endpoint available for FIFO allocation, excluding endpoint 0. FIFO configuration registers are Indexed registers and cannot be accessed directly; configuration takes place via accessing proxy registers. The INDEX register at addresses 140Eh or 1C0Eh must be set to the appropriate endpoint. In other words, the Endpoint FIFO configuration registers do not have non-indexed regions.

The allocation of FIFO space to the different endpoints requires programming for each Tx and Rx endpoint with the following information:

- The start address of the FIFO within the RAM block
- The maximum size of packet to be supported
- Whether double-buffering is required

(These last two together define the amount of space that needs to be allocated to the FIFO.)

It is the responsibility of the firmware to ensure that all the Tx and Rx endpoints (other than endpoint 0) that are active in the current USB configuration have a block of FIFO RAM allocated for their use. The maximum FIFO size for endpoint 0 required is 64 bytes deep and which is large enough to buffer one packet. For this reason, the first 64 bytes of FIFO memory is reserved for Endpoint 0 usage and this resource allocation is implied to be available at all times and no FIFO allocation is required for Endpoint 0. For endpoints other than endpoint 0, the FIFO RAM interface is configurable and must have a minimum size of 8 bytes and should be capable of buffering either 1 or 2 packets. Separate FIFOs may be associated with each endpoint: alternatively a Tx endpoint and the Rx endpoint with the same Endpoint number can be configured to use the same FIFO, to reduce the size of RAM block needed, provided they can never be active at the same time.

NOTE: The option of dynamically setting FIFO sizes only applies to Endpoints 1-15. Endpoint 0 FIFO has a fixed size (64 bytes) and a fixed location which is the first 64 Bytes of FIFO (FIFO addresses 0-63).

24.2.10 USB Controller Host and Peripheral Modes Operation

The two USB modules can be used in a range of different environments. They can be used as either a high-speed or a full-speed USB peripheral device attached to a conventional USB host (such as a PC) in a point-to-point type of arrangement. As a host, the USB modules can also be used with another peripheral device of any speed (high-,full-, or low-speed) in a point-to-point arrangement or can be used as the host to a range of peripheral devices in a multi-point setup; one-to-many via hub. Note that the role each USB controller is assuming is independent of each other.

Since no USB ID pin has been bonded out on the device, the USB2.0 controller role adaptation of a host or peripheral (device) is dependent upon the setting made by the firmware to the respective USB Mode Register IDDIG field. If the IDDIG field is programmed by the firmware with a value of ‘1’ prior to the USB going into session, it would assume the role of a peripheral/device. However, if the IDDIG field is programmed with the value of ‘0’, the USB controller will assume the role of a host. What this means is that the USB cable end, connector, will not be able to control the role of the OTG controller and the user needs to be aware of the firmware program setting prior to performing a USB connection.

The procedure for the USB2.0 OTG controller determining its operating modes (usb controller assuming a role of a host or a peripheral) starts when the USB 2.0 controller goes to session. The USB 2.0 controller is in session when either it senses a voltage (>= 4.4V) on the USB0_VBUS pin and the controller sets DEVCTL[SESSION] or when the firmware sets the DEVCTL[SESSION] bit; assuming that it will operating as a host.

When the SESSION bit is set, the controller will start sensing the state of the Iddig signal controlled by the IDDIG bit field of the MODE Register. If the state of the iddig signal is found to be low (IDDIG is programmed with ‘0’) then the USB2.0 controller will assume the role of a host and when it finds the iddig signal to be high (IDDIG is programmed with ‘1’) then it will assume the role of a device. Note that iddig is an internal signal that could be driven to high or low via firmware from dedicated registers.

Upon the USB controller determining its role as a host, it will drive the USB0/1_DRVVBUS pin high to enable the external power logic so that it start sourcing the required 5V power (must be ≥ 4.4V to account for the voltage drop on the cable it is suggested to be in the neighborhood of 4.75V). The USB2.0 controller will then wait for the voltage of the USB0/1_VBUS to go high. Upon sampling a little time later, if it does not see the voltage on the USB0/1_VBUS pin to be greater than Vbus Valid voltage (4.4V), it will generate a Vbus error interrupt. Assuming that the voltage level of the USB0/1_VBUS is found to be above Vbus Valid, the USB 2.0 host controller will wait for a device to connect; that is for it to see one of its data lines USB0/1_DP/DM be pulled high.

When assuming the role of a peripheral, assuming that the firmware has set the POWER[SOFTCONN] bit, and the USB0/1 Mode Register IDDIG bit field is programmed with '1' and an external host is sourcing power on the USB0/1_VBUS line then the USB2.0 controller will set the DEVCTL[SESSION] bit instructing the controller to go into session. Note that the voltage on the USB0/1_VBUS pin must be greater or equal to 4.4V. When the controller goes into session, it will force the USB2.0 Controller to sense the state of the iddig signal. Once it senses iddig signal to be high, it will enable its 1.5 Kohm pull-up resistor to signify the external host that it is a Full-Speed device. Note that even when operating as a High-Speed peripheral; the USB controller has to first come up as Full-Speed and then later transition to High-Speed. The USB2.0 controller will then waits for a reset bus condition from the external host. Then after, if High-Speed option has been selected it will negotiate for a High-Speed operation and if its request has been accepted by the host, it will enable its precision 45 ohm resistors on its data lines and disable the 1.5 Kohm resistor.

24.3 Protocol Description(s)

This section describes the implementation of the USB protocol(s) by the USB modules.

24.3.1 USB Controller Peripheral Mode Operation

The USB controller assumes the role of a peripheral when the USB Mode Register[iddig=bit8] is set to 1 by the user application prior to the controller goes into session. When the USB controller go into session it will assume the role of a device.

Soft connect –After a POR or USB Module soft reset, the SOFTCONN bit of POWER register (bit 6) is cleared to 0. The controller will therefore appear disconnected until the software has set the SOFTCONN bit to 1. The application software can then choose when to set the PHY into its normal mode. Systems with a lengthy initialization procedure may use this to ensure that initialization is complete and the system is ready to perform enumeration before connecting to the USB. Once the SOFTCONN bit has been set, the software can also simulate a disconnect by clearing this bit to 0.

Entry into suspend mode –When operating as a peripheral device, the controller monitors activity on the bus and when no activity has occurred for 3 ms, it goes into Suspend mode. If the Suspend interrupt has been enabled, an interrupt will be generated at this time.

At this point, the controller can then be left active (and hence able to detect when Resume signaling occurs on the USB), or the application may arrange to disable the controller by stopping its clock. However, the controller will not then be able to detect Resume signaling on the USB if clocks are not running. If such is the case, external hardware will be needed to detect Resume signaling (by monitoring the DM and DP signals), so that the clock to the controller can be restarted.

Resume Signaling –When resume signaling occurs on the bus, first the clock to the controller must be restarted if necessary. Then the controller will automatically exit Suspend mode. If the Resume interrupt is enabled, an interrupt will be generated.

Initiating a remote wakeup –If the software wants to initiate a remote wakeup while the controller is in Suspend mode, it should set the POWER[RESUME] bit to 1. The software should leave then this bit set for approximately 10 ms (minimum of 2 ms, a maximum of 15 ms) before resetting it to 0.

NOTE: No resume interrupt will be generated when the software initiates a remote wakeup.

Reset Signaling –When reset signaling or bus condition occurs on the bus, the controller performs the following actions:

- Clears FADDR register to 0
- Clears INDEX register to 0
- Flushes all endpoint FIFOs

- Clears all controller control/status registers
- Generates a reset interrupt
- Enables all interrupts at the core level.

If the HSENA bit within the POWER register (bit 5) is set, the controller also tries to negotiate for high-speed operation. Whether high-speed operation is selected is indicated by HSMODE bit of POWER register (bit 4). When the application software receives a reset interrupt, it should close any open pipes and wait for bus enumeration to begin.

24.3.1.1 Control Transactions:Peripheral Mode

Endpoint 0 is the main control endpoint of the core. The software is required to handle all the standard device requests that may be sent or received via endpoint 0. These are described in Universal Serial Bus Specification, Revision 2.0, Chapter 9. The protocol for these device requests involves different numbers and types of transactions per request/command. To accommodate this, the software needs to take a state machine approach to command decoding and handling.

The Standard Device Requests received by a USB peripheral device can be divided into three categories: Zero Data Requests (in which all the information is included in the command; no additional data is required), Write Requests (in which the command will be followed by additional data), and Read Requests (in which the device is required to send data back to the host).

This section looks at the sequence of actions that the software must perform to process these different types of device request.

NOTE: The Setup packet associated with any standard device request should include an 8-byte command. Any setup packet containing a command field other than 8 bytes will be automatically rejected by the controller.

24.3.1.1.1 Zero Data Requests: Peripheral Mode

Zero data requests have all their information included in the 8-byte command and require no additional data to be transferred. Examples of Zero Data standard device requests are:

- SET_FEATURE
- CLEAR_FEATURE
- SET_ADDRESS
- SET_CONFIGURATION
- SET_INTERFACE

The sequence of events will begin, as with all requests, when the software receives an endpoint 0 interrupt. The RXPKTRDY bit of PERI_CSR0 (bit 0) will also have been set. The 8-byte command should then be read from the endpoint 0 FIFO, decoded and the appropriate action taken.

For example, if the command is SET_ADDRESS, the 7-bit address value contained in the command should be written to the FADDR register at the completion of the command. The PERI_CSR0 register should be written by setting the SERV_RXPKTRDY bit (bit 6) (indicating that the command has been read from the FIFO) and also setting the DATAEND bit (bit 3) (indicating that no further data is expected for this request). The interval between setting SERV_RXPKTRDY bit and DATAEND bit should be very small to avoid getting a SETUPEND error condition. It is highly recommended to set both bits at the same time.

When the host moves to the status stage of the request, a second endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software. The second interrupt is just a confirmation that the request completed successfully. For SET_ADDRESS command, the address should be written to FADDR register at the completion of the command, that is, when the status stage interrupt is received.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the PERI_CSR0 register should be written to set the SERV_RXPKTRDY bit (bit 6) and to set the SENDSTALL bit (bit 5). When the host moves to the status stage of the request, the controller will send a STALL packet telling the host that the request was not executed. A second endpoint 0 interrupt will be generated and the SENTSTALL bit (bit 2 of PERI_CSR0) will be set.

If the host sends more data after the DATAEND bit has been set, then the controller will send a STALL packet automatically. An endpoint 0 interrupt will be generated and the SENTSTALL bit (bit 2 of PERI_CSR0) will be set.

NOTE: DMA is not supported for endpoint 0; endpoint 0 is always serviced via CPU.

24.3.1.1.2 Write Requests: Peripheral Mode

Write requests involve an additional packet (or packets) of data being sent from the host after the 8-byte command. An example of a Write standard device request is: SET_DESCRIPTOR.

The sequence of events will begin, as with all requests, when the software receives an endpoint 0 interrupt. The RXPKTRDY bit of PERI_CSR0 will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO and decoded.

As with a zero data request, the PERI_CSR0 register should then be written to set the SERV_RXPKTRDY bit (bit 6) (indicating that the command has been read from the FIFO) but in this case the DATAEND bit (bit 3) should not be set (indicating that more data is expected).

When a second endpoint 0 interrupt is received, the PERI_CSR0 register should be read to check the endpoint status. The RXPKTRDY bit of PERI_CSR0 should be set to indicate that a data packet has been received. The COUNT0 register should then be read to determine the size of this data packet. The data packet can then be read from the endpoint 0 FIFO.

If the length of the data associated with the request (indicated by the wLength field in the command) is greater than the maximum packet size for endpoint 0, further data packets will be sent. In this case, PERI_CSR0 should be written to set the SERV_RXPKTRDY bit, but the DATAEND bit should not be set.

When all the expected data packets have been received, the PERI_CSR0 register should be written to set the SERV_RXPKTRDY bit and to set the DATAEND bit (indicating that no more data is expected).

When the host moves to the status stage of the request, another endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software, the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the PERI_CSR0 register should be written to set the SERV_RXPKTRDY bit (bit 6) and to set the SENDSTALL bit (bit 5). When the host sends more data, the controller will send a STALL to tell the host that the request was not executed. An endpoint 0 interrupt will be generated and the SENTSTALL bit of PERI_CSR0 (bit 2) will be set.

If the host sends more data after the DATAEND has been set, then the controller will send a STALL. An endpoint 0 interrupt will be generated and the SENTSTALL bit of PERI_CSR0 (bit 2) will be set.

24.3.1.1.3 Read Requests: Peripheral Mode

Read requests have a packet (or packets) of data sent from the function to the host after the 8-byte command. Examples of Read Standard Device Requests are:

- GET_CONFIGURATION
- GET_INTERFACE
- GET_DESCRIPTOR
- GET_STATUS
- SYNCH_FRAME

The sequence of events will begin, as with all requests, when the software receives an endpoint 0 interrupt. The RXPKTRDY bit of PERI_CSR0 (bit 0) will also have been set. The 8-byte command should then be read from the endpoint 0 FIFO and decoded. The PERI_CSR0 register should then be written to set the SERV_RXPKTRDY bit (bit 6) (indicating that the command has read from the FIFO).

The data to be sent to the host should then be written to the endpoint 0 FIFO. If the data to be sent is greater than the maximum packet size for endpoint 0, only the maximum packet size should be written to the FIFO. The PERI_CSR0 register should then be written to set the TXPKTRDY bit (bit 1) (indicating that there is a packet in the FIFO to be sent). When the packet has been sent to the host, another endpoint 0 interrupt will be generated and the next data packet can be written to the FIFO.

When the last data packet has been written to the FIFO, the PERI_CSR0 register should be written to set the TXPKTRDY bit and to set the DATAEND bit (bit 3) (indicating that there is no more data after this packet).

When the host moves to the status stage of the request, another endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software: the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the PERI_CSR0 register should be written to set the SERV_RXPKTRDY bit (bit 6) and to set the SENDSTALL bit (bit 5). When the host requests data, the controller will send a STALL to tell the host that the request was not executed. An endpoint 0 interrupt will be generated and the SENTSTALL bit of PERI_CSR0 (bit 2) will be set.

If the host requests more data after DATAEND (bit 3) has been set, then the controller will send a STALL. An endpoint 0 interrupt will be generated and the SENTSTALL bit of PERI_CSR0 (bit 2) will be set.

24.3.1.1.4 Endpoint 0 States: Peripheral Mode

When the USB controller is operating as a peripheral device, the endpoint 0 control needs three modes – IDLE, TX and RX – corresponding to the different phases of the control transfer and the states endpoint 0 enters for the different phases of the transfer (described in later sections).

The default mode on power-up or reset should be IDLE. RXPKTRDY bit of PERI_CSR0 (bit 0) becoming set when endpoint 0 is in IDLE state indicates a new device request. Once the device request is unloaded from the FIFO, the controller decodes the descriptor to find whether there is a data phase and, if so, the direction of the data phase of the control transfer (in order to set the FIFO direction). See Figure 24-2.

Depending on the direction of the data phase, endpoint 0 goes into either TX state or RX state. If there is no Data phase, endpoint 0 remains in IDLE state to accept the next device request

The actions that the CPU needs to take at the different phases of the possible transfers (for example, loading the FIFO, setting TXPKTRDY) are indicated in Figure 24-3.

Figure 24-2. CPU Actions at Transfer Phases

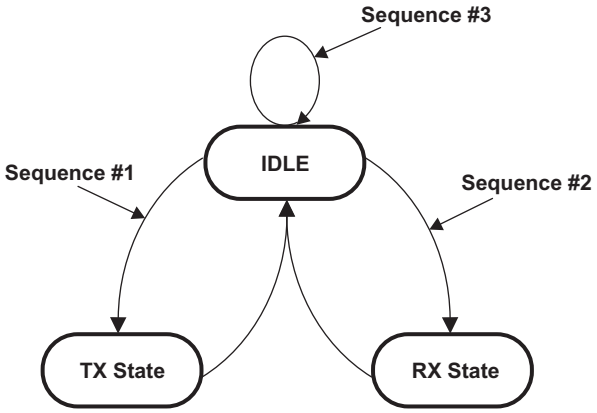
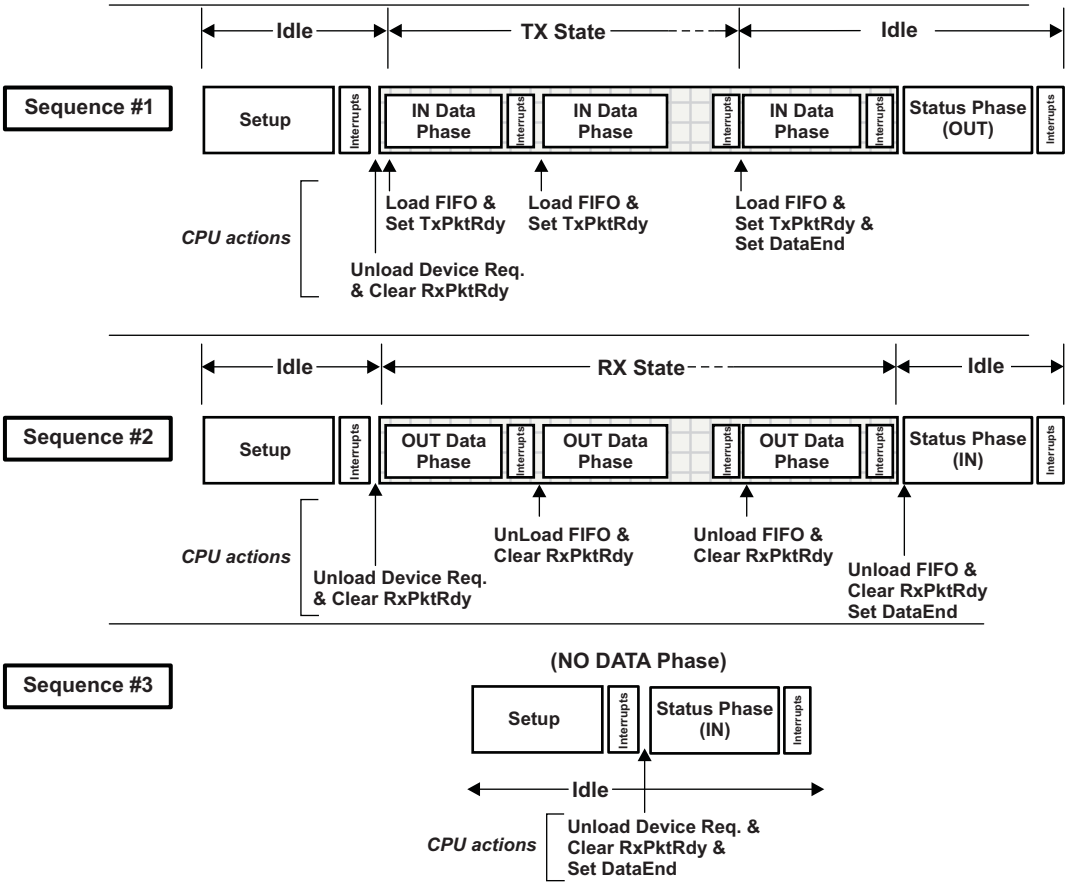


Figure 24-3. Sequence of Transfer



24.3.1.1.4.1 Endpoint 0 Interrupt Generating Events: Peripheral Mode

An Endpoint 0 interrupt is generated when any of the below events occur and Interrupt Service handler should determine the right event and launch the proper handler

The following events will cause Endpoint 0 Interrupt to be generated while the controller is operating assuming the role of a device/peripheral:

- The controller sets the RXPKTRDY bit of PERI_CSR0 (bit 0) after a valid token has been received and data has been written to the FIFO (legal status condition).
- The controller clears the TXPKTRDY bit of PERI_CSR0 (bit 1) after the packet of data in the FIFO has been successfully transmitted to the host (legal status condition).
- The controller sets the SENTSTALL bit of PERI_CSR0 (bit 2) after a control transaction is ended due to a protocol violation (illegal status condition).
- The controller sets the SETUPEND bit of PERI_CSR0 (bit 4) because a control transfer has ended before DATAEND (bit 3 of PERI_CSR0) is set (illegal status condition).

In order to determine the cause of the interrupt, the detail below recommends the order of cause determination and execution.

Whenever the endpoint 0 service routine is entered, the software must first check to see if the current control transfer has been ended due to either a STALL condition or a premature end of control transfer. If the control transfer ends due to a STALL condition, the SENTSTALL bit would be set. If the control transfer ends due to a premature end of control transfer, the SETUPEND bit would be set. In either case, the software should abort processing the current control transfer and set the state to IDLE.

Once the software has determined that the interrupt was not generated by an illegal bus state, the next action taken depends on the endpoint state.

If endpoint 0 is in IDLE state, the only valid reason an interrupt can be generated is as a result of the controller receiving data from the bus. The service routine must check for this by testing the RXPKTRDY bit of PERI_CSR0 (bit 0). If this bit is set, then the controller has received a SETUP packet. This must be unloaded from the FIFO and decoded to determine the action the controller must take. Depending on the command contained within the SETUP packet, endpoint 0 will enter one of three states:

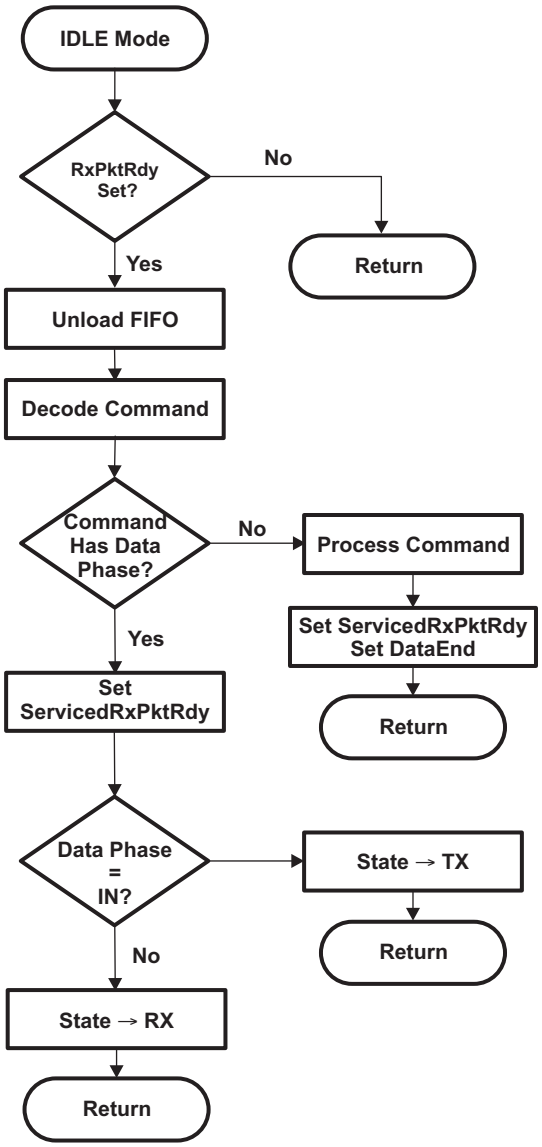
- If the command is a single packet transaction (SET_ADDRESS, SET_INTERFACE etc.) without any data phase, the endpoint will remain in IDLE state.
- If the command has an OUT data phase (SET_DESCRIPTOR etc.), the endpoint will enter RX state.
- If the command has an IN data phase (GET_DESCRIPTOR etc.), the endpoint will enter TX state.
- If the endpoint 0 is in TX state, the interrupt indicates that the core has received an IN token and data from the FIFO has been sent. The software must respond to this either by placing more data in the FIFO if the host is still expecting more data or by setting the DATAEND bit to indicate that the data phase is complete. Once the data phase of the transaction has been completed, endpoint 0 should be returned to IDLE state to await the next control transaction (status stage).
- If the endpoint is in RX state, the interrupt indicates that a data packet has been received. The software must respond by unloading the received data from the FIFO. The software must then determine whether it has received all of the expected data. If it has, the software should set the DATAEND bit and return endpoint 0 to IDLE state. If more data is expected, the firmware should set the SERV_RXPKTRDY bit of PERI_CSR0 (bit 6) to indicate that it has read the data in the FIFO and leave the endpoint in RX state.

24.3.1.1.4.2 Idle Mode: Control Transfer of Peripheral Mode

IDLE mode is the mode the Endpoint 0 control needs to select at power-on or reset and is the mode to which the Endpoint 0 control should return when the RX and TX modes are terminated.

It is also the mode in which the SETUP phase of control transfer is handled (as outlined in Figure 24-4).

Figure 24-4. Flow Chart of Setup Stage of a Control Transfer in Peripheral Mode



24.3.1.1.4.3 TX Mode: Control Transfer of Peripheral Mode

When the endpoint is in TX state, all arriving IN tokens need to be treated as part of a data phase until the required amount of data has been sent to the host. If either a SETUP or an OUT token is received while the endpoint is in the TX state, this will cause a SETUPEND condition to occur as the core expects only IN tokens.

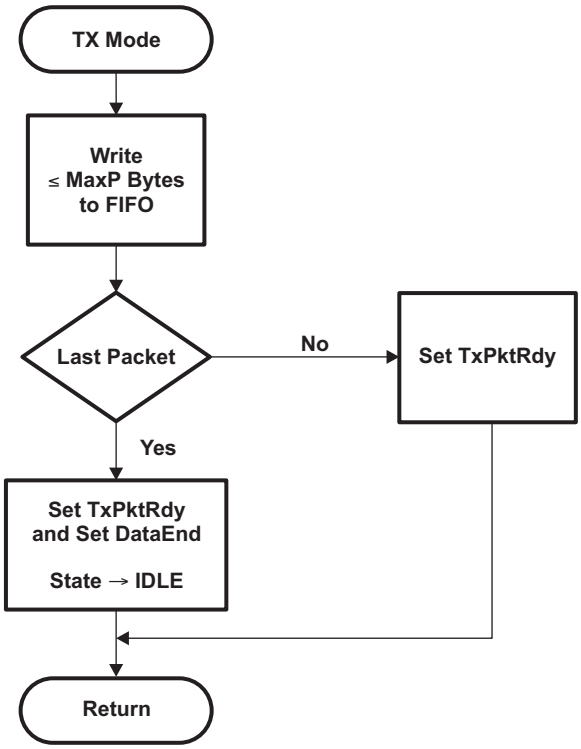
Three events can cause TX mode to be terminated before the expected amount of data has been sent as shown in Figure 24-5:

1. The host sends an invalid token causing a SETUPEND condition (bit 4 of PERI_CSR0 is set).
2. The firmware sends a packet containing less than the maximum packet size for Endpoint 0.
3. The firmware sends an empty data packet.

Until the transaction is terminated, the firmware simply needs to load the FIFO when it receives an interrupt which indicates that a packet has been sent from the FIFO. An interrupt is generated when TXPKTRDY is cleared.

When the firmware forces the termination of a transfer (by sending a short or empty data packet), it should set the DATAEND bit of PERI_CSR0 (bit 3) to indicate to the core that the data phase is complete and that the core should next receive an acknowledge packet.

Figure 24-5. Flow Chart of Transmit Data Stage of a Control Transfer in Peripheral Mode



24.3.1.1.4.4 Rx Mode: Control Transfer of Peripheral Mode

In RX mode, all arriving data should be treated as part of a data phase until the expected amount of data has been received. If either a SETUP or an IN token is received while the endpoint is in RX state, a SETUPEND condition will occur as the controller expects only OUT tokens.

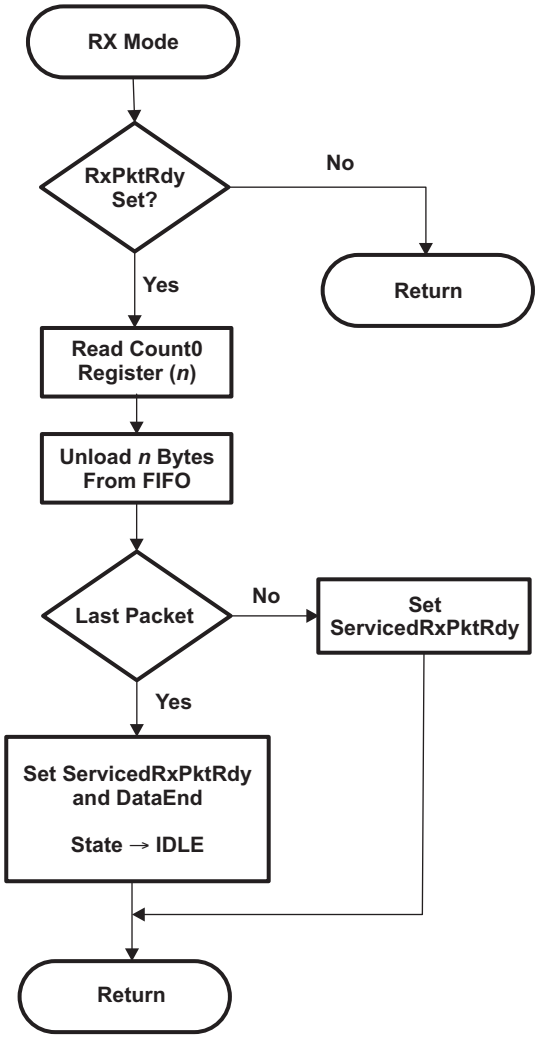
Three events can cause RX mode to be terminated before the expected amount of data has been received as shown in Figure 24-6:

1. The host sends an invalid token causing a SETUPEND condition (setting bit 4 of PERI_CSR0).
2. The host sends a packet which contains less than the maximum packet size for endpoint 0.
3. The host sends an empty data packet.

Until the transaction is terminated, the software unloads the FIFO when it receives an interrupt that indicates new data has arrived (setting RXPKTRDY bit of PERI_CSR0) and to clear RXPKTRDY by setting the SERV_RXPKTRDY bit of PERI_CSR0 (bit 6).

When the software detects the termination of a transfer (by receiving either the expected amount of data or an empty data packet), it should set the DATAEND bit (bit 3 of PERI_CSR0) to indicate to the controller that the data phase is complete and that the core should receive an acknowledge packet next.

Figure 24-6. Flow Chart of Receive Data Stage of a Control Transfer in Peripheral Mode



24.3.1.1.4.5 Error Handling: Control Transfer of Peripheral Mode

A control transfer may be aborted due to a protocol error on the USB, the host prematurely ending the transfer, or if the software wishes to abort the transfer (for example, because it cannot process the command).

The controller automatically detects protocol errors and sends a STALL packet to the host under the following conditions:

- The host sends more data during the OUT Data phase of a write request than was specified in the command. This condition is detected when the host sends an OUT token after the DATAEND bit (bit 3 of PERI_CSR0) has been set.
- The host requests more data during the IN Data phase of a read request than was specified in the command. This condition is detected when the host sends an IN token after the DATAEND bit in the PERI_CSR0 register has been set.
- The host sends more than Max Packet Size data bytes in an OUT data packet.
- The host sends a non-zero length DATA1 packet during the STATUS phase of a read request.

When the controller has sent the STALL packet, it sets the SENTSTALL bit (bit 2 of PERI_CSR0) and generates an interrupt. When the software receives an endpoint 0 interrupt with the SENTSTALL bit set, it should abort the current transfer, clear the SENTSTALL bit, and return to the IDLE state.

If the host prematurely ends a transfer by entering the STATUS phase before all the data for the request has been transferred, or by sending a new SETUP packet before completing the current transfer, then the SETUPEND bit (bit 4 of PERI_CSR0) will be set and an endpoint 0 interrupt generated. When the software receives an endpoint 0 interrupt with the SETUPEND bit set, it should abort the current transfer, set the SERV_SETUPEND bit (bit 7 of PERI_CSR0), and return to the IDLE state. If the RXPKTRDY bit (bit 0 of PERI_CSR0) is set this indicates that the host has sent another SETUP packet and the software should then process this command.

If the software wants to abort the current transfer, because it cannot process the command or has some other internal error, then it should set the SENDSTALL bit (bit 5 of PERI_CSR0). The controller will then send a STALL packet to the host, set the SENTSTALL bit (bit 2 of PERI_CSR0) and generate an endpoint 0 interrupt.

24.3.1.1.5 Additional Conditions: Control transfer of Peripheral Mode

When working as a peripheral device, the controller automatically responds to certain conditions on the USB bus or actions by the host. The details are:

- Stall Issued to Control Transfers
  1. The host sends more data during an OUT Data phase of a Control transfer than was specified in the device request during the SETUP phase. This condition is detected by the controller when the host sends an OUT token (instead of an IN token) after the software has unloaded the last OUT packet and set DATAEND.
  2. The host requests more data during an IN data phase of a Control transfer than was specified in the device request during the SETUP phase. This condition is detected by the controller when the host sends an IN token (instead of an OUT token) after the software has cleared TXPKTRDY and set DATAEND in response to the ACK issued by the host to what should have been the last packet.
  3. The host sends more than MaxPktSize data with an OUT data token.
  4. The host sends the wrong PID for the OUT Status phase of a Control transfer.
  5. The host sends more than a zero length data packet for the OUT Status phase.
- Zero Length Out Data Packets In Control Transfer
  A zero length OUT data packet is used to indicate the end of a Control transfer. In normal operation, such packets should only be received after the entire length of the device request has been transferred (after the software has set DataEnd). If, however, the host sends a zero length OUT data packet before the entire length of device request has been transferred, this signals the premature end of the transfer. In this case, the controller will automatically flush any IN token loaded by software ready for the Data phase from the FIFO and set SETUPEND bit (bit 4 of PERI_CSR0).

24.3.1.2 Bulk Transfer: Peripheral Mode

Bulk transactions are handled by endpoints other than endpoint 0. It is used to handle non-periodic, large bursty communication typically used for a transfer that use any available bandwidth and can also be delayed until bandwidth is available.

24.3.1.2.1 Bulk IN Transactions: Peripheral Mode

A Bulk IN transaction is used to transfer non-periodic data from the USB peripheral device to the host.

The following optional features are available for use with a Tx endpoint used in peripheral mode for Bulk IN transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature allows the DMA controller to load packets into the FIFO without processor intervention

24.3.1.2.1.1 Bulk IN Transaction Setup: Peripheral Mode

In configuring a TX endpoint for bulk transactions, the TXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint and the PERI_TXCSR register (DMAEN and DMAMODE bit fields should be set when using DMA. Table 24-2 displays the PERI_TXCSR setting when used for Bulk transfer.

Table 24-2. PERI_TXCSR Register Bit Configuration for Bulk IN Transactions

Bit	Bit Name	Description
Bit 15	AUTOSET	Cleared to 0 if using DMA. For CPU Mode use, if AUTOSET bit is set, the TXPKTRDY bit will be automatically set when data of the maximum packet size is loaded into the FIFO.
Bit 14	ISO	Cleared to 0 for bulk mode operation.
Bit 13	MODE	Set to 1 to make sure the FIFO is enabled (only necessary if the FIFO is shared with an RX endpoint)
Bit 12	DMAEN	Set to 1 to enable DMA usage; not needed if CPU is being used to service the Tx Endpoint
Bit 11	FRCDATATOG	Cleared to 0 to allow normal data toggle operations.
Bit 10	DMAMODE	Set to 1 when DMA is used to service Tx FIFO.

When the endpoint is first configured (following a SET_CONFIGURATION or SET_INTERFACE command on Endpoint 0), the lower byte of PERI_TXCSR should be written to set the CLRDATATOG bit (bit 6). This will ensure that the data toggle (which is handled automatically by the controller) starts in the correct state.

Also if there are any data packets in the FIFO, indicated by the FIFONOTEMPTY bit (bit 1 of PERI_TXCSR) being set, they should be flushed by setting the FLUSHFIFO bit (bit 3 of PERI_TXCSR).

NOTE: It may be necessary to set this bit twice in succession if double buffering is enabled.

24.3.1.2.1.2 Bulk IN Operation: Peripheral Mode

When data is to be transferred over a Bulk IN pipe, a data packet needs to be loaded into the FIFO and the PERI_TXCSR register written to set the TXPKTRDY bit (bit 0). When the packet has been sent, the TXPKTRDY bit will be cleared by the USB controller and an interrupt generated to signify to the user/application that the next packet can be loaded into the FIFO. If double packet buffering is enabled, then after the first packet has been loaded and the TXPKTRDY bit set, the TXPKTRDY bit will immediately be cleared (will not wait for the loaded first packet to be driven out) by the USB controller and an interrupt generated so that a second packet can be loaded into the FIFO. The software should operate in the same way, loading a packet when it receives an interrupt, regardless of whether double packet buffering is enabled or not.

In the general case, the packet size must not exceed the size specified by the lower 11 bits of the TXMAXP register. This part of the register defines the payload (packet size) for transfers over the USB and is required by the USB Specification to be either 8, 16, 32, 64 (Full-Speed or High-Speed) or 512 bytes (High-Speed only).

The host may determine that all the data for a transfer has been sent by knowing the total amount of data that is expected. Alternatively it may infer that all the data has been sent when it receives a packet which is smaller than the maximum packet size configuration for that particular endpoint (TXMAXP[10-0]). In the latter case, if the total size of the data block is a multiple of this payload, it will be necessary for the function to send a null packet after all the data has been sent. This is done by setting TXPKTRDY when the next interrupt is received, without loading any data into the FIFO.

If large blocks of data are being transferred, then the overhead of calling an interrupt service routine to load each packet can be avoided by using the CPPI DMA. A separate section detailing the use of the DMA is discussed within a latter section. Suffix is to say that the PERI_TXCSR (DMAEN and DMAMODE) bit fields need to be set and the PERI_TXCSR[AUTOSET] bit cleared at the core level when using the DMA with an endpoint configured for any IN transaction/transfer.

24.3.1.2.1.3 Error Handling of Bulk IN Transfer: Peripheral Mode

If the software wants to shut down the Bulk IN pipe, it should set the SENDSTALL bit (bit 4 of PERI_TXCSR). When the controller receives the next IN token, it will send a STALL to the host, set the SENTSTALL bit (bit 5 of PERI_TXCSR) and generate an interrupt.

When the software receives an interrupt with the SENTSTALL bit (bit 5 of PERI_TXCSR) set, it should clear the SENTSTALL bit. It should however leave the SENDSTALL bit set until it is ready to re-enable the Bulk IN pipe.

NOTE: If the host failed to receive the STALL packet for some reason, it will send another IN token, so it is advisable to leave the SENDSTALL bit set until the software is ready to re-enable the Bulk IN pipe. When a pipe is re-enabled, the data toggle sequence should be restarted by setting the CLRDATATOG bit in the PERI_TXCSR register (bit 6).

24.3.1.2.2 Bulk OUT Transfer: Peripheral Mode

A Bulk OUT transaction is used to transfer non-periodic data from the host to the function controller.

The following optional features are available for use with an Rx endpoint used in peripheral mode for Bulk OUT transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. Double packet buffering is enabled by setting the DPB bit of the RXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention.

NOTE: When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

24.3.1.2.2.1 Bulk OUT Transfer Setup: Peripheral Mode

In configuring an Rx endpoint for Bulk OUT transactions, the RXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. When using DMA for Rx Endpoints, the PERI_RXCSR needs to have its DMAEN bit set and have its AUTOCLEAR and DMAMODE bits cleared. In addition, the relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint) and the PERI_RXCSR register should be set as shown in Table 24-3.

Table 24-3. PERI_RXCSR Register Bit Configuration for Bulk OUT Transactions

Bit	Bit Name	Description
Bit 15	AUTOCLEAR	Cleared to 0 if using DMA. In CPU Mode of usage, if the CPU sets AUTOCLEAR bit, the RXPKTRDY bit will be automatically cleared when a packet of RXMAXP bytes has been unloaded from the Receive FIFO.
Bit 14	ISO	Cleared to 0 to enable Bulk protocol.
Bit 13	DMAEN	Set to 1 if a DMA request is required for this Rx endpoint.
Bit 12	DISNYET	Cleared to 0 to allow normal PING flow control. This will affect only high speed transactions.
Bit 11	DMAMODE	Always clear this bit to 0.

When the Rx endpoint is first configured (following a SET_CONFIGURATION or SET_INTERFACE command on Endpoint 0), the lower byte of PERI_RXCSR should be written to set the CLRDATATOG bit (bit 7). This will ensure that the data toggle (which is handled automatically by the USB controller) starts in the correct state.

Also if there are any data packets in the FIFO (indicated by the RXPKTRDY bit (bit 0 of PERI_RXCSR) being set), they should be flushed by setting the FLUSHFIFO bit (bit 4 of PERI_RXCSR).

NOTE: It may be necessary to set this bit twice in succession if double buffering is enabled.

24.3.1.2.2.2 Bulk OUT Operation: Peripheral Mode

When a data packet is received by a Bulk Rx endpoint, the RXPKTRDY bit (bit 0 of PERI_RXCSR) is set and an interrupt is generated. The software should read the RXCOUNT register for the endpoint to determine the size of the data packet. The data packet should be read from the FIFO, and then the RXPKTRDY bit should be cleared.

The packets received should not exceed the size specified in the RXMAXP register (as this should be the value set in the wMaxPacketSize field of the endpoint descriptor sent to the host). When a block of data larger than wMaxPacketSize needs to be sent to the function, it will be sent as multiple packets. All the packets will be wMaxPacketSize in size, except the last packet which will contain the residue. The software may use an application specific method of determining the total size of the block and hence when the last packet has been received. Alternatively it may infer that the entire block has been received when it receives a packet which is less than wMaxPacketSize in size. (If the total size of the data block is a multiple of wMaxPacketSize, a null data packet will be sent after the data to signify that the transfer is complete.)

In the general case, the application software will need to read each packet from the FIFO individually. If large blocks of data are being transferred, the overhead of calling an interrupt service routine to unload each packet can be avoided by using DMA.

24.3.1.2.2.3 Bulk OUT Error Handling: Peripheral Mode

If the software wants to shut down the Bulk OUT pipe, it should set the SENDSTALL bit (bit 5 of PERI_RXCSR). When the controller receives the next packet it will send a STALL to the host, set the SENTSTALL bit (bit 6 of PERI_RXCSR) and generate an interrupt.

When the software receives an interrupt with the SENTSTALL bit (bit 6 of PERI_RXCSR) set, it should clear this bit. It should however leave the SENDSTALL bit set until it is ready to re-enable the Bulk OUT pipe.

NOTE: If the host failed to receive the STALL packet for some reason, it will send another packet, so it is advisable to leave the SENDSTALL bit set until the software is ready to re-enable the Bulk OUT pipe. When a Bulk OUT pipe is re-enabled, the data toggle sequence should be restarted by setting the CLRDATATOG bit (bit 7) in the PERI_RXCSR register.

24.3.1.3 Interrupt Transfer: Peripheral Mode

An Interrupt IN transaction uses the same protocol as a Bulk IN transaction and the configuration and operation mentioned on prior sections apply to Interrupt transfer too with minor changes. Similarly, an Interrupt OUT transaction uses almost the same protocol as a Bulk OUT transaction and can be used the same way.

Tx endpoints in the USB controller have one feature for Interrupt IN transactions that they do not support in Bulk IN transactions. In Interrupt IN transactions, the endpoints support continuous toggle of the data toggle bit.

This feature is enabled by setting the FRCDATATOG bit in the PERI_TXCSR register (bit 11). When this bit is set, the controller will consider the packet as having been successfully sent and toggle the data bit for the endpoint, regardless of whether an ACK was received from the host.

Another difference is that interrupt endpoints do not support PING flow control. This means that the controller should never respond with a NYET handshake, only ACK/NAK/STALL. To ensure this, the DISNYET bit in the PERI_RXCSR register (bit 12) should be set to disable the transmission of NYET handshakes in high-speed mode.

Though DMA can be used with an interrupt OUT endpoint, it generally offers little benefit as interrupt endpoints are usually expected to transfer all their data in a single packet.

24.3.1.4 Isochronous Transfer: Peripheral Mode

Isochronous transfers are used when working with isochronous data. Isochronous transfers provide periodic, continuous communication between host and device.

24.3.1.4.1 Isochronous IN Transactions: Peripheral Mode

An Isochronous IN transaction is used to transfer periodic data from the function controller to the host.

The following optional features are available for use with a Tx endpoint used in Peripheral mode for Isochronous IN transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).

  NOTE: Double packet buffering is generally advisable for isochronous transactions in order to avoid underrun errors as described in later section.
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature allows the DMA controller to load packets into the FIFO without processor intervention.
  However, this feature is not particularly useful with Isochronous endpoints because the packets transferred are often not maximum packet size and the PERI_TXCSR register needs to be accessed following every packet to check for Underrun errors.
  When DMA is enabled and DMAMODE bit of PERI_TXCSR is set, endpoint interrupt will not be generated for completion of packet transfer. Endpoint interrupt will be generated only in the error conditions.

24.3.1.4.1.1 Isochronous IN Transfer Setup: Peripheral Mode

In configuring a Tx endpoint for Isochronous IN transactions, the TXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRTXE register should be set (if an interrupt is required for this endpoint) and the PERI_TXCSR register should be set as shown in Table 24-4.

Table 24-4. PERI_TXCSR Register Bit Configuration for Isochronous IN Transactions

Bit	Bit Name	Description
Bit 14	ISO	Set to 1 to enable Isochronous transfer protocol.
Bit 13	MODE	Set to 1 to ensure the FIFO is enabled (only necessary if the FIFO is shared with an Rx endpoint).
Bit 12	DMAEN	Set to 1 if DMA Requests have to be enabled.
Bit 11	FRCDATATOG	Ignored in Isochronous mode.
Bit 10	DMAMODE	Set to 1 when DMA is enabled.

24.3.1.4.1.2 Isochronous IN Transfer Operation: Peripheral Mode

An isochronous endpoint does not support data retries, so if data underrun is to be avoided, the data to be sent to the host must be loaded into the FIFO before the IN token is received. The host will send one IN token per frame (or microframe in High-speed mode), however the timing within the frame (or microframe) can vary. If an IN token is received near the end of one frame and then at the start of the next frame, there will be little time to reload the FIFO. For this reason, double buffering of the endpoint is usually necessary.

An interrupt is generated whenever a packet is sent to the host and the software may use this interrupt to load the next packet into the FIFO and set the TXPKTRDY bit in the PERI_TXCSR register (bit 0) in the same way as for a Bulk Tx endpoint. As the interrupt could occur almost any time within a frame(/microframe), depending on when the host has scheduled the transaction, this may result in irregular timing of FIFO load requests. If the data source for the endpoint is coming from some external hardware, it may be more convenient to wait until the end of each frame(/microframe) before loading the FIFO as this will minimize the requirement for additional buffering. This can be done by using either the SOF interrupt or the external SOF_PULSE signal from the controller to trigger the loading of the next data packet. The SOF_PULSE is generated once per frame(/microframe) when a SOF packet is received. (The controller also maintains an external frame(/microframe) counter so it can still generate a SOF_PULSE when the SOF packet has been lost.) The interrupts may still be used to set the TXPKTRDY bit in PERI_TXCSR (bit 0) and to check for data overruns/underruns.

Starting up a double-buffered Isochronous IN pipe can be a source of problems. Double buffering requires that a data packet is not transmitted until the frame(/microframe) after it is loaded. There is no problem if the function loads the first data packet at least a frame(/microframe) before the host sets up the pipe (and therefore starts sending IN tokens). But if the host has already started sending IN tokens by the time the first packet is loaded, the packet may be transmitted in the same frame(/microframe) as it is loaded, depending on whether it is loaded before, or after, the IN token is received. This potential problem can be avoided by setting the ISOUPDATE bit in the POWER register (bit 7). When this bit is set, any data packet loaded into an Isochronous Tx endpoint FIFO will not be transmitted until after the next SOF packet has been received, thereby ensuring that the data packet is not sent too early.

24.3.1.4.1.3 Isochronous IN Error Handling: Peripheral Mode

If the endpoint has no data in its FIFO when an IN token is received, it will send a null data packet to the host and set the UNDERRUN bit in the PERI_TXCSR register (bit 2). This is an indication that the software is not supplying data fast enough for the host. It is up to the application to determine how this error condition is handled.

If the software is loading one packet per frame(/microframe) and it finds that the TXPKTRDY bit in the PERI_TXCSR register (bit 0) is set when it wants to load the next packet, this indicates that a data packet has not been sent (perhaps because an IN token from the host was corrupted). It is up to the application how it handles this condition: it may choose to flush the unsent packet by setting the FLUSHFIFO bit in the PERI_TXCSR register (bit 3), or it may choose to skip the current packet.

24.3.1.4.2 Isochronous OUT Transactions: Peripheral Mode

An isochronous OUT transaction is used to transfer periodic data from the host to the function controller.

Following optional features are available for use with an Rx endpoint used in Peripheral mode for Isochronous OUT transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. Double packet buffering is enabled by setting the DPB bit of RXFIFOSZ register (bit 4).
  NOTE: Double packet buffering is generally advisable for Isochronous transactions in order to avoid overrun errors.
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention.
  However, this feature is not particularly useful with Isochronous endpoints because the packets transferred are often not maximum packet size and the PERI_RXCSR register needs to be accessed following every packet to check for Overrun or CRC errors.
  When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

24.3.1.4.2.1 Isochronous OUT Setup: Peripheral Mode

In configuring an Rx endpoint for Isochronous OUT transactions, the RXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint) and the PERI_RXCSR register should be set as shown in Table 24-5.

Table 24-5. PERI_RXCSR Register Bit Configuration for Isochronous OUT Transactions

Bit	Bit Name	Description
Bit 15	AUTOCLEAR	Cleared to 0 if using DMA. In CPU Mode of usage, if the CPU sets AUTOCLEAR bit, the RXPKTRDY bit will be automatically cleared when a packet of RXMAXP bytes has been unloaded from the Receive FIFO.
Bit 14	ISO	Set to 1 to configure endpoint usage for Isochronous transfer.
Bit 13	DMAEN	Set to 1 if a DMA request is required for this Rx endpoint.
Bit 12	DISNYET	Ignored in Isochronous Mode.
Bit 11	DMAMODE	Always clear this bit to 0.

24.3.1.4.2.2 Isochronous OUT Operation: Peripheral Mode

An Isochronous endpoint does not support data retries so, if a data overrun is to be avoided, there must be space in the FIFO to accept a packet when it is received. The host will send one packet per frame (or microframe in High-speed mode); however, the time within the frame can vary. If a packet is received near the end of one frame(/microframe) and another arrives at the start of the next frame, there will be little time to unload the FIFO. For this reason, double buffering of the endpoint is usually necessary.

An interrupt is generated whenever a packet is received from the host and the software may use this interrupt to unload the packet from the FIFO and clear the RXPKTRDY bit in the PERI_RXCSR register (bit 0) in the same way as for a Bulk Rx endpoint. As the interrupt could occur almost any time within a frame(/microframe), depending on when the host has scheduled the transaction, the timing of FIFO unload requests will probably be irregular. If the data sink for the endpoint is going to some external hardware, it may be better to minimize the requirement for additional buffering by waiting until the end of each frame(/microframe) before unloading the FIFO. This can be done by using either the SOF interrupt or the external SOF_PULSE signal from the controller to trigger the unloading of the data packet. The SOF_PULSE is generated once per frame(/microframe) when a SOF packet is received. (The controller also maintains an external frame(/microframe) counter so it can still generate a SOF_PULSE when the SOF packet has been lost.) The interrupts may still be used to clear the RXPKTRDY bit in PERI_RXCSR and to check for data overruns/underruns.

24.3.1.4.2.3 Isochronous OUT Error Handling: Peripheral Mode

If there is no space in the FIFO to store a packet when it is received from the host, the OVERRUN bit in the PERI_RXCSR register (bit 2) will be set. This is an indication that the software is not unloading data fast enough for the host. It is up to the application to determine how this error condition is handled

If the controller finds that a received packet has a CRC error, it will still store the packet in the FIFO and set the RXPKTRDY bit (bit 0 of PERI_RXCSR) and the DATAERROR bit (bit 3 of PERI_RXCSR). It is left up to the application to determine how this error condition is handled.

The number of USB packets sent in any microframe will depend on the amount of data to be transferred, and is indicated through the PIDs used for the individual packets. If the indicated number of packets have not been received by the end of a microframe, the INCOMPRX bit (bit 8) bit in the PERI_RXCSR register will be set to indicate that the data in the FIFO is incomplete. Equally, if a packet of the wrong data type is received, then the PID Error bit is the PERI_RXCSR register will be set. In each case, an interrupt will, however, still be generated to allow the data that has been received to be read from the FIFO.

Note: The circumstances in which a PID Error or INCOMPRX is reported depends on the precise sequence of packets received.

When the core is operating in peripheral mode, the details are in Isochronous OUT Error Handling: Peripheral Mode.

Isochronous OUT Error Handling: Peripheral Mode

No. Packet(s) Expected	Data Packet(s) Received	Response
1	DATA0	OK
	DATA1	PID Error set
	DATA2	PID Error set
	MDATA	PID Error set
2	DATA0	OK
	DATA1	INCOMPRX Set
	DATA2	INCOMPRX Set + PID Error set
	MDATA	INCOMPRX Set
	MDATA DATA0	PID Error Set
	MDATA DATA1	OK
	MDATA DATA2	PID Error Set
	MDATA MDATA	PID Error Set

Isochronous OUT Error Handling: Peripheral Mode (continued)

No. Packet(s) Expected	Data Packet(s) Received	Response
3	DATA0	OK
	DATA1	INCOMPRX Set
	DATA2	INCOMPRX Set
	MDATA	INCOMPRX Set
	MDATA DATA0	PID Error set
	MDATA DATA1	OK
	MDATA DATA2	INCOMPRX Set
	MDATA MDATA	INCOMPRX Set
	MDATA MDATA DATA0	PID Error set
	MDATA MDATA DATA1	PID Error set
	MDATA MDATA DATA2	OK
	MDATA MDATA MDATA	PID Error set

24.3.2 USB Controller Host Mode Operation

The USB controller assumes the role of a host when the USB Mode Register[iddig=bit8] is cleared to 0 by the firmware prior to the controller goes into session. When the USB controller go into session, application/firmware sets the DEVCTL[SESSION] bit to 1, it will assume the role of a host.

- Entry into Suspend mode. When operating as a host, the USB controller can be prompted to go into Suspend mode by setting the SUSPENDM bit in the POWER register. When this bit is set, the controller will complete the current transaction then stop the transaction scheduler and frame counter. No further transactions will be started and no SOF packets will be generated.
  If the ENSUSPM bit (bit 0 of POWER register) is set, the UTMI+ PHY will go into low-power mode when the controller goes into suspend mode.
- Sending Resume Signaling. When the application requires the controller to leave suspend mode, it needs to clear the SUSPENDM bit in the POWER register, set the RESUME bit and leave it set for 20ms. While the RESUME bit is high, the controller will generate Resume signaling on the bus. After 20 ms, the CPU should clear the RESUME bit, at which point the frame counter and transaction scheduler will be started.
- Responding to Remote Wake-up. If Resume signaling is detected from the target while the controller is in suspend mode, the UTMI+ PHY will be brought out of low-power mode and UTMI clock is restarted. The controller will then exit suspend mode and automatically set the RESUME bit in the POWER register (bit 2) to ‘1’ to take over generating the Resume signaling from the target. If the resume interrupt is enabled, an interrupt will be generated.
- Reset Signaling. If the RESET bit in the POWER register (bit 3) is set while the controller is in Host mode, it will generate Reset signaling on the bus. If the HSENAB bit in the POWER register (bit 5) was set, it will also try to negotiate for high-speed operation. The software should keep the RESET bit set for at least 20 ms to ensure correct resetting of the target device. After the software has cleared the bit, the controller will start its frame counter and transaction scheduler. Whether high-speed operation is selected will be indicated by HSMODE bit of POWER register (bit 4).

24.3.2.1 Control Transactions: Host Mode

Host control transactions are conducted through Endpoint 0 and the software is required to handle all the Standard Device Requests that may be sent or received via Endpoint 0 (as described in Universal Serial Bus Specification, Revision 2.0). Endpoint 0 can only be serviced via CPU and DMA mode can not be used.

There are three categories of Standard Device Requests to be handled: Zero Data Requests (in which all the information is included in the command); Write Requests (in which the command will be followed by additional data); and Read Requests (in which the device is required to send data back to the host).

1. Zero Data Requests comprise a SETUP command followed by an IN Status Phase.
2. Write Requests comprise a SETUP command, followed by an OUT Data Phase which is in turn followed by an IN Status Phase.
3. Read Requests comprise a SETUP command, followed by an IN Data Phase which is in turn followed by an OUT Status Phase.

A timeout may be set to limit the length of time for which the host controller will retry a transaction which is continually NAKed by the target. This limit can be between 2 and 2 15 frames/microframes and is set through the NAKLIMIT0 register.

The following sections describe the actions that the CPU needs to take in issuing these different types of request through looking at the steps to take in the different phases of a control transaction.

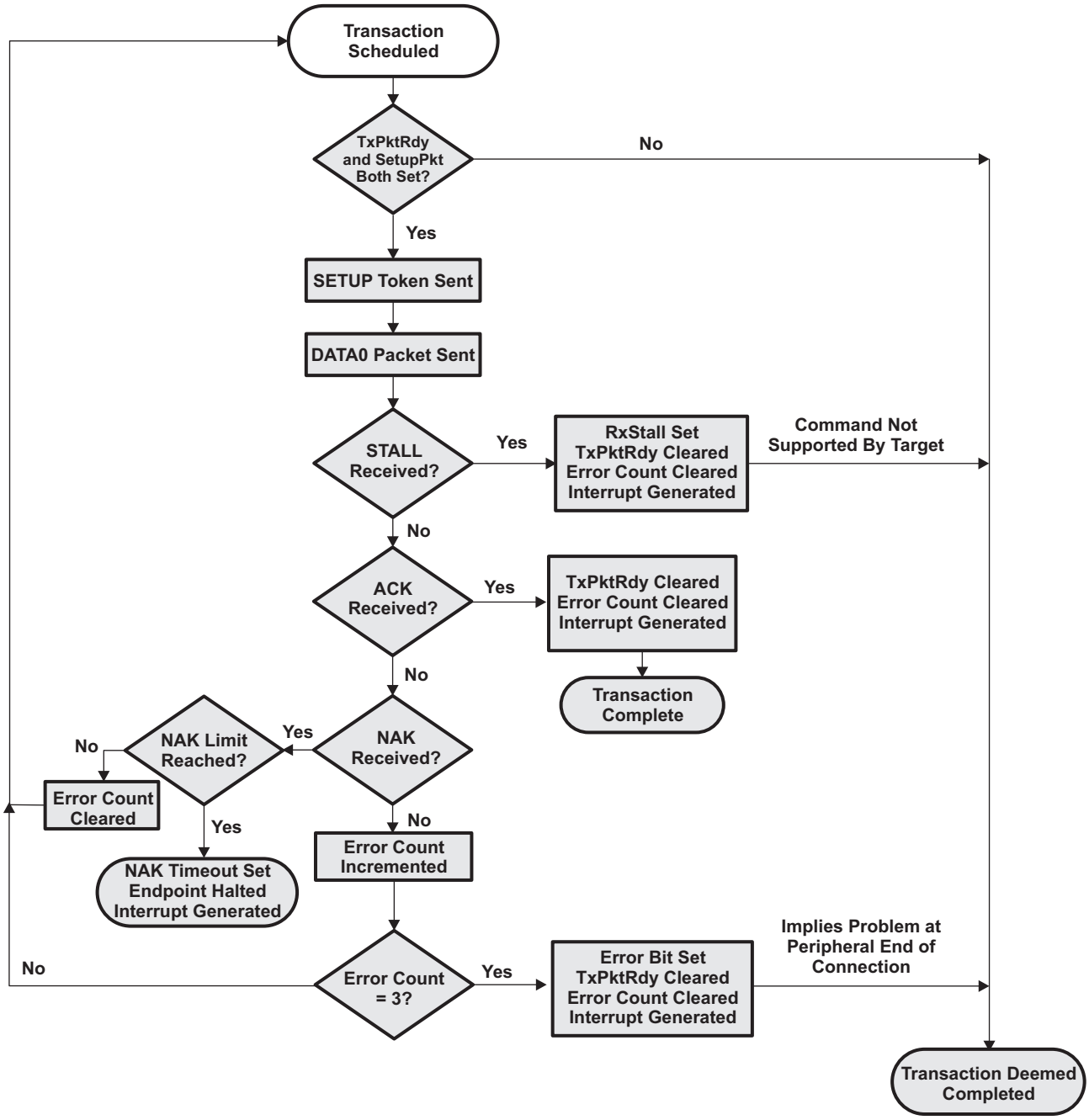
Note: Before initiating any transactions as a Host, the FADDR register needs to be set to address the peripheral device. When the device is first connected, FADDR should be cleared to 0. After a SET_ADDRESS command is issued, FADDR should be set the target's new address.

24.3.2.1.1 Setup Phase of Control Transaction: Host Mode

For the SETUP Phase of a control transaction (Figure 24-7), the software driving the USB host device needs to:

1. Load the 8 bytes of the required Device request command into the Endpoint 0 FIFO
2. Set SETUPPKT and TXPKTRDY (bits 3 and 1 of HOST_CSR0, respectively).
   NOTE: These bits must be set together.
   The controller then proceeds to send a SETUP token followed by the 8-byte command/request to Endpoint 0 of the addressed device, retrying as necessary. *Note:* On errors, the controller retries the transaction three times.
3. At the end of the attempt to send the 8-byte request data, the controller will generate an Endpoint 0 interrupt. The software should then read HOST_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4) or the NAK_TIMEOUT bit (bit 7) has been set.
   If RXSTALL is set, it indicates that the target did not accept the command (for example, because it is not supported by the target device) and so has issued a STALL response. If ERROR is set, it means that the controller has tried to send the SETUP Packet and the following data packet three times without getting any response.
   If NAK_TIMEOUT is set, it means that the controller has received a NAK response to each attempt to send the SETUP packet, for longer than the time set in HOST_NAKLIMIT0. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK_TIMEOUT bit or to abort the transaction by flushing the FIFO before clearing the NAK_TIMEOUT bit.
4. If none of RXSTALL, ERROR or NAK_TIMEOUT is set, the SETUP Phase has been correctly ACKed and the software should proceed to the following IN Data Phase, OUT Data Phase or IN Status Phase specified for the particular Standard Device Request.

Figure 24-7. Flow Chart of Setup Stage of a Control Transfer in Host Mode



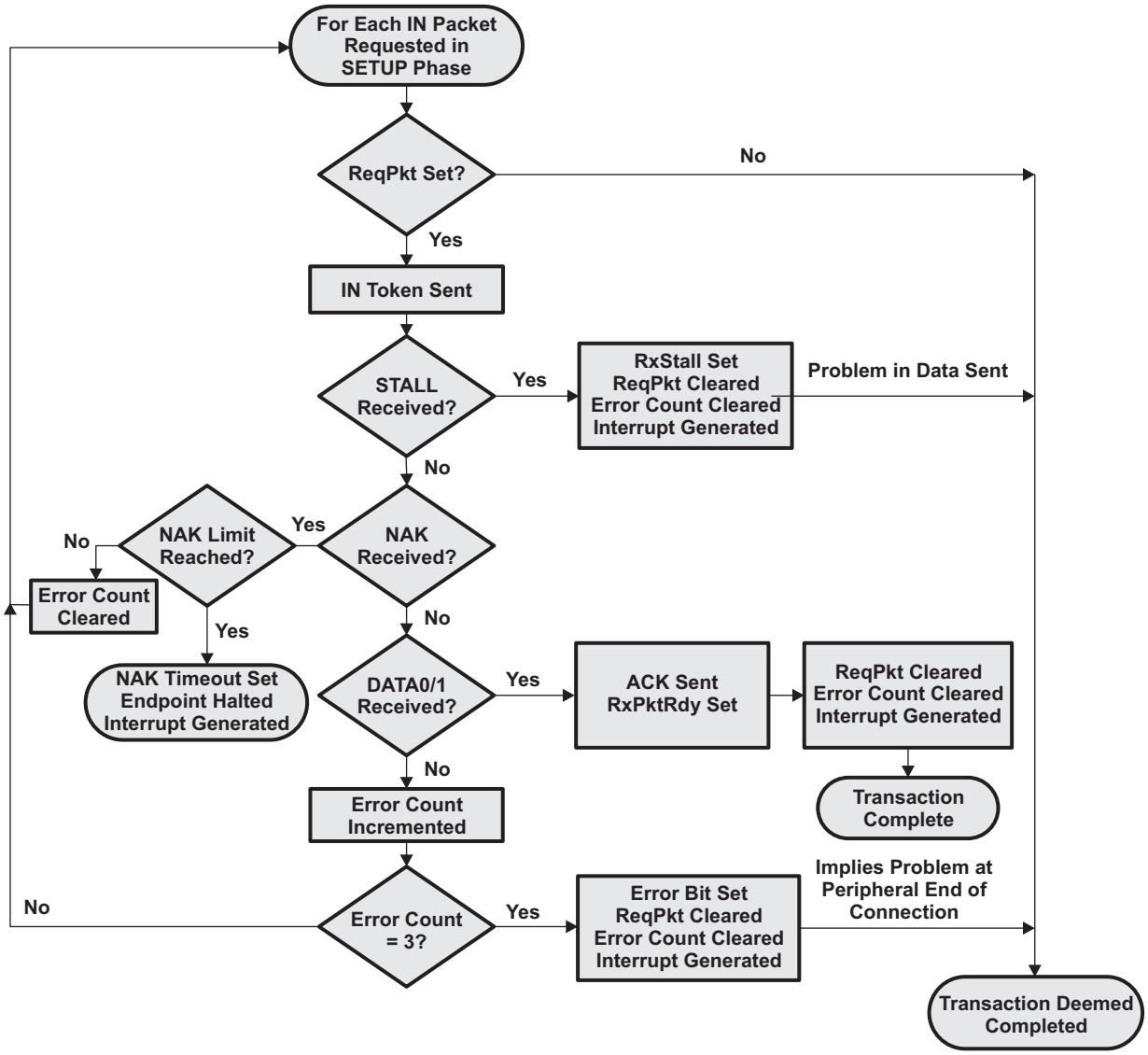
24.3.2.1.2 Data Phase (IN Data Phase) of a Control Transaction: Host Mode

For the IN Data Phase of a control transaction (Figure 24-8), the software driving the USB host device needs to

1. Set REQPKT bit of HOST_CSR0 (bit 5)
2. Wait while the controller sends the IN token and receives the required data back.
3. When the controller generates the Endpoint 0 interrupt, read HOST_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4), the NAK_TIMEOUT bit (bit 7) or RXPKTRDY bit (bit 0) has been set.
   If RXSTALL is set, it indicates that the target has issued a STALL response.
   If ERROR is set, it means that the controller has tried to send the required IN token three times without getting any response. If NAK_TIMEOUT bit is set, it means that the controller has received a NAK response to each attempt to send the IN token, for longer than the time set in HOST_NAKLIMIT0. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK_TIMEOUT bit or to abort the transaction by clearing REQPKT before clearing the NAK_TIMEOUT bit
4. If RXPKTRDY has been set, the software should read the data from the Endpoint 0 FIFO, then clear RXPKTRDY.
5. If further data is expected, the software should repeat Steps 1-4.

When all the data has been successfully received, the CPU should proceed to the OUT Status Phase of the Control Transaction.

Figure 24-8. Flow Chart of Data Stage (IN Data Phase) of a Control Transfer in Host Mode

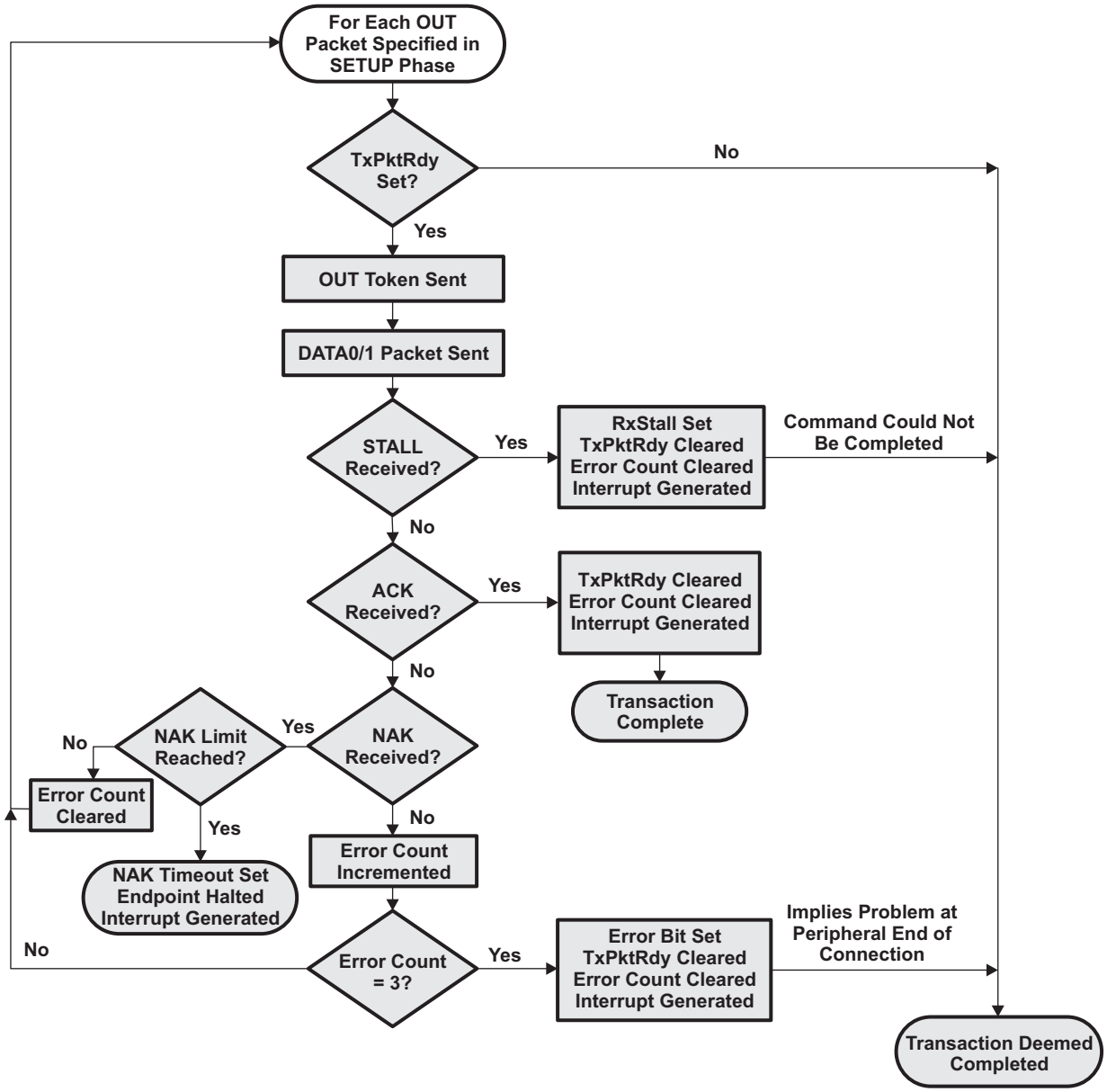


24.3.2.1.3 Data Phase (OUT Data Phase) of a Control Transaction: Host Mode

For the OUT Data Phase of a control transaction (Figure 24-9), the software driving the USB host device needs to:

1. Load the data to be sent into the endpoint 0 FIFO.
2. Set the TXPKTRDY bit of HOST_CSR0 (bit 1). The controller then proceeds to send an OUT token followed by the data from the FIFO to Endpoint 0 of the addressed device, retrying as necessary.
3. At the end of the attempt to send the data, the controller will generate an Endpoint 0 interrupt. The software should then read HOST_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4) or the NAK_TIMEOUT bit (bit 7) has been set.
   If RXSTALL bit is set, it indicates that the target has issued a STALL response.
   If ERROR bit is set, it means that the controller has tried to send the OUT token and the following data packet three times without getting any response. If NAK_TIMEOUT is set, it means that the controller has received a NAK response to each attempt to send the OUT token, for longer than the time set in the HOST_NAKLIMIT0 register. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK_TIMEOUT bit or to abort the transaction by flushing the FIFO before clearing the NAK_TIMEOUT bit.
   If none of RXSTALL, ERROR or NAKLIMIT is set, the OUT data has been correctly ACKed.
4. If further data needs to be sent, the software should repeat Steps 1-3.
   When all the data has been successfully sent, the software should proceed to the IN Status Phase of the Control Transaction.

Figure 24-9. Flow Chart of Data Stage (OUT Data Phase) of a Control Transfer in Host Mode



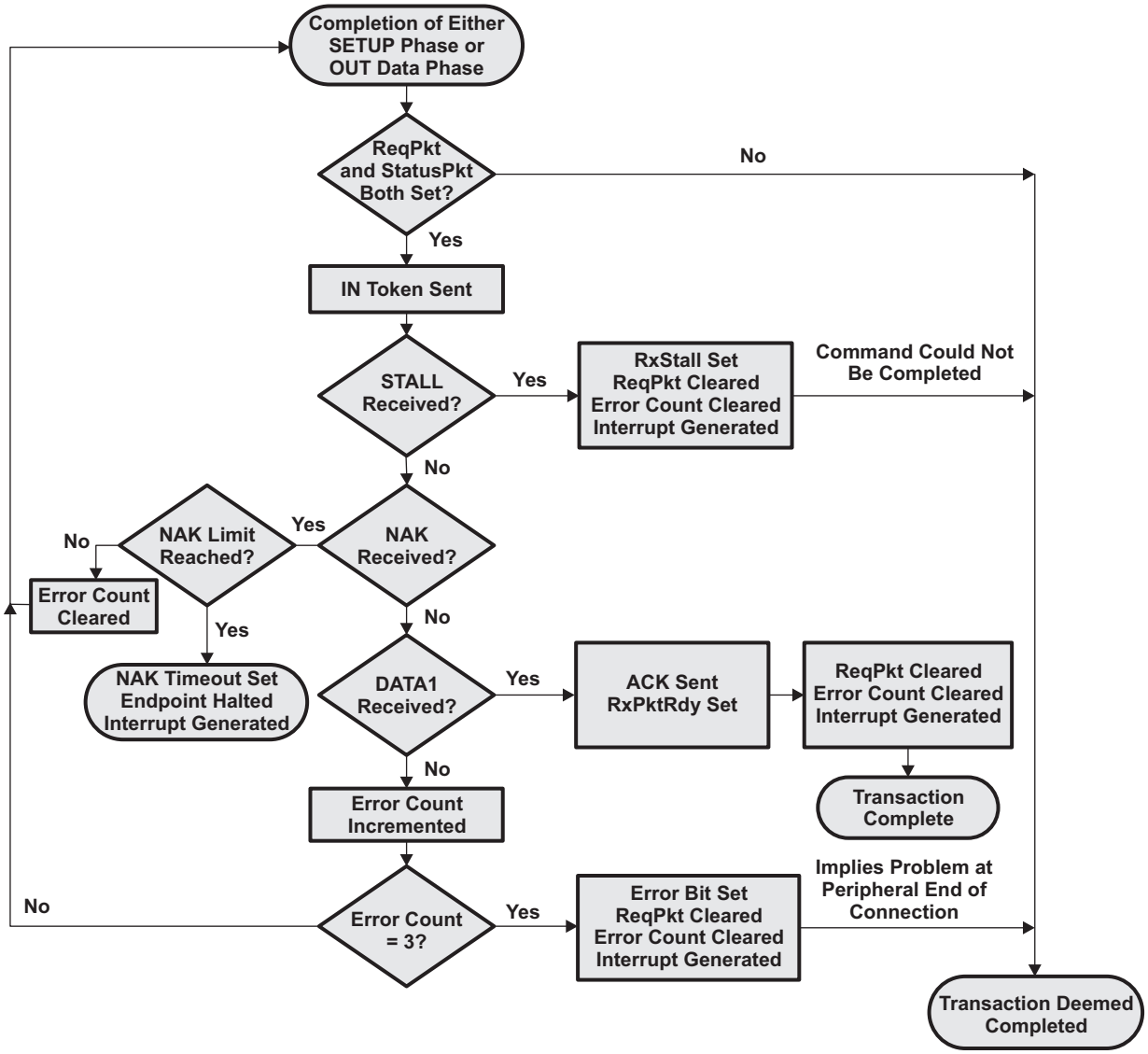
24.3.2.1.4 Status Phase (Status for OUT Data Phase or Setup Phase) of a Control Transaction: Host Mode

IN Status Phase of a control transfer exists for a Zero Data Request or for a Write Request of a control transfer. The IN Status Phase follows the Setup Stage, if no Data Stage of a control transfer exists, or OUT Data Phase of a Data Stage of a control transfer.

For the IN Status Phase of a control transaction (Figure 24-10), the software driving the USB Host device needs to:

1. Set the STATUSPKT and REQPKT bits of HOST_CSR0 (bit 6 and bit 5, respectively).
2. Wait while the controller sends an IN token and receives a response from the USB peripheral device.
3. When the controller generates the Endpoint 0 interrupt, read HOST_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4), the NAK_TIMEOUT bit (bit 7) or RXPKTRDY bit (bit 0) has been set.
   If RXSTALL bit is set, it indicates that the target could not complete the command and so has issued a STALL response.
   If ERROR bit is set, it means that the controller has tried to send the required IN token three times without getting any response.
   If NAK_TIMEOUT bit is set, it means that the controller has received a NAK response to each attempt to send the IN token, for longer than the time set in the HOST_NAKLIMIT0 register. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK_TIMEOUT bit or to abort the transaction by clearing REQPKT bit and STATUSPKT bit before clearing the NAK_TIMEOUT bit.
4. The CPU should clear the STATUSPPKT bit of HOST_CSR0 together with (in the same write operation as) RxPktRdy if this has been set.

Figure 24-10. Flow Chart of Status Stage of Zero Data Request or Write Request of a Control Transfer in Host Mode



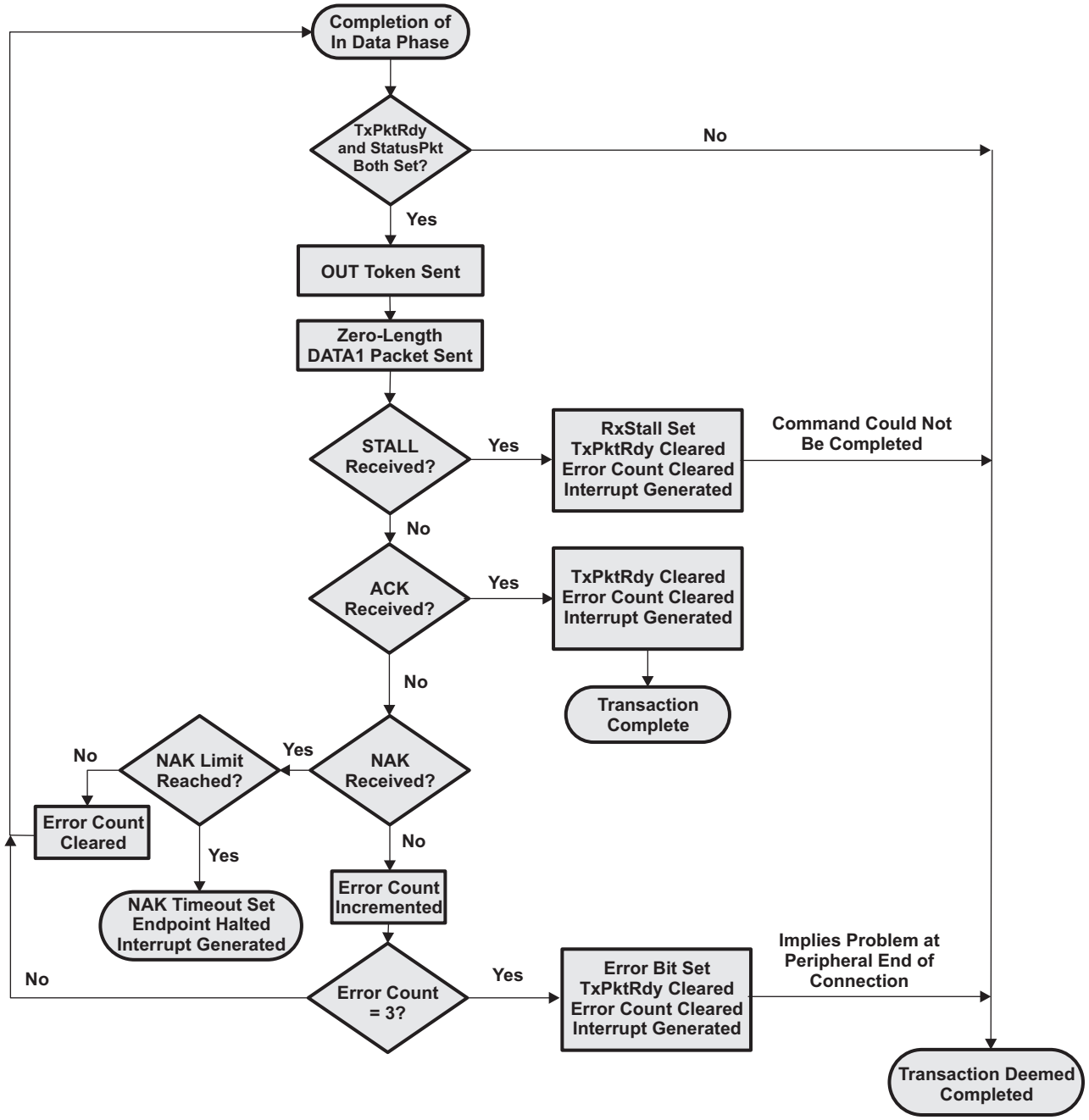
24.3.2.1.5 Status Phase for a Read Request of a Control Transaction: Host Mode

OUT Status Phase of a control transfer exist for a Read Request or a control transfer where data was received by the host controller. The OUT Status phase follows the IN Data Stage of a control transfer.

For the OUT Status Phase of a control transaction (Figure 24-11), the CPU driving the host device needs to:

1. Set STATUSPKT and TXPKTRDY bits of HOST_CSR0 (bit 6 and bit 1, respectively).
   NOTE: These bits need to be set together
2. Wait while the controller sends the OUT token and a zero-length DATA1 packet.
3. At the end of the attempt to send the data, the controller will generate an Endpoint 0 interrupt. The software should then read HOST_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4) or the NAK_TIMEOUT bit (bit 7) has been set.
   If RXSTALL bit is set, it indicates that the target could not complete the command and so has issued a STALL response.
   If ERROR bit is set, it means that the controller has tried to send the STATUS Packet and the following data packet three times without getting any response.
   If NAK_TIMEOUT bit is set, it means that the controller has received a NAK response to each attempt to send the IN token, for longer than the time set in the HOST_NAKLIMIT0 register. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK_TIMEOUT bit or to abort the transaction by flushing the FIFO before clearing the NAK_TIMEOUT bit.
4. If none of RXSTALL, ERROR or NAK_TIMEOUT bits is set, the STATUS Phase has been correctly ACKed.

Figure 24-11. Chart of Status Stage of a Read Request of a Control Transfer in Host Mode



24.3.2.2 Bulk Transfer: Host Mode

Bulk transactions are handled by endpoints other than endpoint 0. It is used to handle non-periodic, large bursty communication typically used for a transfer that use any available bandwidth and can also be delayed until bandwidth is available.

24.3.2.2.1 Bulk IN Transactions: Host Mode

A Bulk IN transaction may be used to transfer non-periodic data from the external USB peripheral to the host.

The following optional features are available for use with an Rx endpoint used in host mode to receive the data:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. This allows that one packet can be received while another is being read. Double packet buffering is enabled by setting the DPB bit of RXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention.
  When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions. For more information see section on CPPI DMA

24.3.2.2.1.1 Bulk IN Setup: Host Mode

Before initiating any Bulk IN Transactions in Host mode:

- The HOST_RXTYPE register for the endpoint that is to be used needs to be programmed as:
  - Operating speed in the SPEED bit field (bits 7 and 6).
  - Set 10 (binary value) in the PROT field for bulk transfer.
  - Endpoint Number of the target device in RENDPN field. This is the endpoint number contained in the Rx endpoint descriptor returned by the target device during enumeration.
- The RXMAXP register for the controller endpoint must be written with the maximum packet size (in bytes) for the transfer. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the target endpoint.
- The HOST_RXINTERVAL register needs to be written with the required value for the NAK limit (2-215 frames/microframes), or cleared to 0 if the NAK timeout feature is not required.
- The relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint).

  Note: Whether interrupt is handled from PDR level or Core level, interrupt is required to be enabled at the core level. See details on Core Interrupt registers, CTRLR register, and sections on Interrupt Handling
- The AUTOREQ bit field should be used only when servicing transfers using CPU mode and must be cleared when using DMA Mode. For DMA Mode, dedicated registers USB0/1 Req Registers exist and the HOST_RXCSR[AUTOREQ] should be cleared.
  When DMA is enabled, the following bits of HOST_RXCSR register should be set as:
  - Clear AUTOCLEAR.
  - Set DMAEN (bit 13) to 1 if a DMA request is required for this endpoint.
  - Clear DSINYET (bit 12) to 0 to allow normal PING flow control. This will affect only High Speed transactions.
  - Clear DMAMODE (bit 11) to 0.

    Note: If DMA is enabled, the USB0/1 Auto Req register can be set for generating IN tokens automatically after receiving the data. Set the bit field RXn_AUTOREQ (where n is the endpoint number) with binary value 01 or 11.

When the endpoint is first configured, the endpoint data toggle should be cleared to 0 either by using the DATATOGWREN and DATATOG bits of HOST_RXCSR (bit 10 and bit 9) to toggle the current setting or by setting the CLRDATATOG bit of HOST_RXCSR (bit 7). This will ensure that the data toggle (which is handled automatically by the controller) starts in the correct state. Also if there are any data packets in the FIFO (indicated by the RXPKTRDY bit (bit 0 of HOST_RXCSR) being set), they should be flushed by setting the FLUSHFIFO bit of HOST_RXCSR (bit 4).

NOTE: It may be necessary to set this bit twice in succession if double buffering is enabled.

24.3.2.2.1.2 Bulk IN Operation: Host Mode

When Bulk data is required from the USB peripheral device, the software should set the REQPKT bit in the corresponding HOST_RXCSR register (bit 5). The controller will then send an IN token to the selected peripheral endpoint and waits for data to be returned.

If data is correctly received, RXPKTRDY bit of HOST_RXCSR (bit 0) is set. If the USB peripheral device responds with a STALL, RXSTALL bit (bit 6 of HOST_RXCSR) is set. If a NAK is received, the controller tries again and continues to try until either the transaction is successful or the POLINTVL_NAKLIMIT set in the HOST_RXINTERVAL register is reached. If no response at all is received, two further attempts are made before the controller reports an error by setting the ERROR bit of HOST_RXCSR (bit 2).

The controller then generates the appropriate endpoint interrupt, whereupon the software should read the corresponding HOST_RXCSR register to determine whether the RXPKTRDY, RXSTALL, ERROR or DATAERR_NAKTIMEOUT bit is set and act accordingly. If the DATAERR_NAKTIMEOUT bit is set, the controller can be directed either to continue trying this transaction (until it times out again) by clearing the DATAERR_NAKTIMEOUT bit or to abort the transaction by clearing REQPKT bit before clearing the DATAERR_NAKTIMEOUT bit.

The packets received should not exceed the size specified in the RXMAXP register (as this should be the value set in the wMaxPacketSize field of the endpoint descriptor sent to the host).

In the general case, the application software (if CPU is servicing the endpoint) will need to read each packet from the FIFO individually. If large blocks of data are being transferred, the overhead of calling an interrupt service routine to unload each packet can be avoided by using DMA.

Note: When using DMA, see CPPI DMA section for the proper configuration of the core register, HOST_RXCSR.

24.3.2.2.1.3 Bulk IN Error Handling: Host Mode

If the target wants to shut down the Bulk IN pipe, it will send a STALL response to the IN token. This will result in the RXSTALL bit of HOST_RXCSR (bit 6) being set.

24.3.2.2.2 Bulk OUT Transactions: Host Mode

A Bulk OUT transaction may be used to transfer non-periodic data from the host to the USB peripheral.

Following optional features are available for use with a Tx endpoint used in Host mode to transmit this data:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the peripheral device. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature can be used to allow the DMA controller to load packets into the FIFO without processor intervention. For more information on using DMA, consult the section discussing CPPI DMA.
  When DMA is enabled and DMAMODE bit in HOST_TXCSR register is set, an endpoint interrupt will not be generated for completion of packet reception. An endpoint interrupt will be generated only in the error conditions.

24.3.2.2.2.1 Bulk OUT Setup: Host Mode

Before initiating any bulk OUT transactions:

- The target function address needs to be set in the TXFUNCADDR register for the selected controller endpoint. (TXFUNCADDR register is available for all endpoints from EP0 to EP15.)
- The HOST_TXTYPE register for the endpoint that is to be used needs to be programmed as:
  - Operating speed in the SPEED bit field (bits 7 and 6).
  - Set PROT field to 10b for bulk transfer.
  - Enter Endpoint Number of the target device in TENDPN field. This is the endpoint number contained in the OUT(Tx) endpoint descriptor returned by the target device during enumeration.
- The TXMAXP register for the controller endpoint must be written with the maximum packet size (in bytes) for the transfer. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the target endpoint.
- The HOST_TXINTERVAL register needs to be written with the required value for the NAK limit (2-215 frames/microframes), or cleared to 0 if the NAK timeout feature is not required.
- The relevant interrupt enable bit in the INTRTXE register should be set (if an interrupt is required for this endpoint).
- The following bits of HOST_TXCSR register should be set as:
  - Set the MODE bit (bit 13) to 1 to ensure the FIFO is enabled (only necessary if the FIFO is shared with an Rx endpoint).
  - Clear the FRCDATATOG bit (bit 11) to 0 to allow normal data toggle operations.
  - Clear/Set AUTOSET bit (bit 15). The setting of this bit depends on the desire of the user/application in automatically setting the TXPKTRDY bit when servicing transactions using CPU.

Note: If DMA needs to be used in place of the CPU, HOST_TXCSR Register Bit Configuration in Host Mode displays the setting of the HOST_TXCSR register in Host mode. For the CPPI DMA registers settings, see Section 24.4.

HOST_TXCSR Register Bit Configuration in Host Mode

Bit	Bit Name	Description
Bit 15	AUTOSET	Cleared to 0 if using DMA. For CPU Mode use, if AUTOSET bit is set, the TXPKTRDY bit will be automatically set when data of the maximum packet size is loaded into the FIFO.
Bit 14	ISO	Cleared to 0 for bulk mode operation.
Bit 13	MODE	Set to 1 to make sure the FIFO is enabled (only necessary if the FIFO is shared with an RX endpoint)
Bit 12	DMAEN	Set to 1 to enable DMA usage; not needed if CPU is being used to service the Tx Endpoint
Bit 11	FRCDATATOG	Cleared to 0 to allow normal data toggle operations.
Bit 10	DMAMODE	Set to 1 when DMA is used to service Tx FIFO.

When the endpoint is first configured, the endpoint data toggle should be cleared to 0 either by using the DATATOGWREN bit and DATATOG bit of HOST_TXCSR (bit 9 and bit 8) to toggle the current setting or by setting the CLRDATATOG bit of HOST_TXCSR (bit 6). This will ensure that the data toggle (which is handled automatically by the controller) starts in the correct state. Also, if there are any data packets in the FIFO (indicated by the FIFONOTEMPTY bit of HOST_TXCSR register (bit 1) being set), they should be flushed by setting the FLUSHFIFO bit (bit 3 of HOST_TXCSR).

NOTE: It may be necessary to set this bit twice in succession if double buffering is enabled.

24.3.2.2.2.2 Bulk OUT Operation: Host Mode

When Bulk data is required to be sent to the USB peripheral device, the software should write the first packet of the data to the FIFO (or two packets if double-buffered) and set the TXPKTRDY bit in the corresponding HOST_TXCSR register (bit 0). The controller will then send an OUT token to the selected peripheral endpoint, followed by the first data packet from the FIFO.

If data is correctly received by the peripheral device, an ACK should be received whereupon the controller will clear TXPKTRDY bit of HOST_TXCSR (bit 0). If the USB peripheral device responds with a STALL, the RXSTALL bit (bit 5) of HOST_TXCSR is set. If a NAK is received, the controller tries again and continues to try until either the transaction is successful or the NAK limit set in the HOST_TXINTERVAL register is reached. If no response at all is received, two further attempts are made before the controller reports an error by setting ERROR bit in HOST_TXCSR (bit 2).

The controller then generates the appropriate endpoint interrupt, whereupon the software should read the corresponding HOST_TXCSR register to determine whether the RXSTALL (bit 5), ERROR (bit 2) or NAK_TIMEOUT (bit 7) bit is set and act accordingly. If the NAK_TIMEOUT bit is set, the controller can be directed either to continue trying this transaction (until it times out again) by clearing the NAK_TIMEOUT bit or to abort the transaction by flushing the FIFO before clearing the NAK_TIMEOUT bit.

If large blocks of data are being transferred, then the overhead of calling an interrupt service routine to load each packet can be avoided by using DMA.

24.3.2.2.2.3 Bulk OUT Error Handling: Host Mode

If the target wants to shut down the Bulk OUT pipe, it will send a STALL response. This is indicated by the RXSTALL bit of HOST_TXCSR register (bit 5) being set.

24.3.2.3 Interrupt Transfer: Host Mode

When the controller is operating as the host, interactions with an Interrupt endpoint on the USB peripheral device are handled in very much the same way as the equivalent Bulk transactions (described in previous sections).

The principal difference as far as operational steps are concerned is that the PROT field of HOST_RXTYPE and HOST_TXTYPE (bits 5-4) need to be set (binary value) to represent an Interrupt transaction. The required polling interval also needs to be set in the HOST_RXINTERVAL and HOST_TXINTERVAL registers.

24.3.2.4 Isochronous Transfer: Host Mode

Isochronous transfers are used when working with isochronous data. Isochronous transfers provide periodic, continuous communication between host and device.

An Isochronous IN transaction is used to transfer periodic data from the USB peripheral to the host.

24.3.2.4.1 Isochronous IN Transactions: Host Mode

The following optional features are available for use with an Rx endpoint used in Host mode to receive this data:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. This allows that one packet can be received while another is being read. Double packet buffering is enabled by setting the DPB bit of RXFIFOSZ register (bit 4).
- AutoRequest: When the AutoRequest feature is enabled, the REQPKT bit of HOST_RXCSR (bit 5) will be automatically set when the RXPKTRDY bit is cleared. This only applies when the CPU is being used to service the endpoint. When using DMA, this bit field needs to be cleared to Zero. CPPI DMA has its own configuration registers that renders a similar task, USB0/1 Auto Req registers, that needs to be used to have a similar effect. For more information, see section on CPPI DMA.

- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention. However, this feature is not particularly useful with isochronous endpoints because the packets transferred are often not maximum packet size.
- When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

24.3.2.4.1.1 Isochronous IN Transfer Setup: Host

Before initiating an Isochronous IN Transactions in Host mode:

- The target function address needs to be set in the RXFUNCADDR register for the selected controller endpoint (RXFUNCADDR register is available for all endpoints from EP0 to EP4).
- The HOST_RXTYPE register for the endpoint that is to be used needs to be programmed as:
  - Operating speed in the SPEED bit field (bits 7 and 6).
  - Set 01 (binary value) in the PROT field for isochronous transfer.
  - Endpoint Number of the target device in RENDPN field. This is the endpoint number contained in the Rx endpoint descriptor returned by the target device during enumeration.
- The RXMAXP register for the controller endpoint must be written with the maximum packet size (in bytes) for the transfer. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the target endpoint.
- The HOST_RXINTERVAL register needs to be written with the required transaction interval (usually one transaction per frame/microframe).
- The relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint).
- The following bits of HOST_RXCSR register should be set as:
  - Clear AUTOCLEAR
  - Set the DMAEN bit (bit 13) to 1 if a DMA request is required for this endpoint.
  - Clear the DISNYET it (bit 12) to 0 to allow normal PING flow control. This will only affect High Speed transactions.
  - Clear DMAMODE bit (bit 11) to 0.
- If DMA is enabled, AUTOREQ register can be set for generating IN tokens automatically after receiving the data. Set the bit field RXn_AUTOREQ (where n is the endpoint number) with binary value 01 or 11. For detailed information on using CPPI DMA, consult related section within this document.

24.3.2.4.1.2 Isochronous IN Operation: Host Mode

The operation starts with the software setting REQPKT bit of HOST_RXCSR (bit 5). This causes the controller to send an IN token to the target.

When a packet is received, an interrupt is generated which the software may use to unload the packet from the FIFO and clear the RXPKTRDY bit in the HOST_RXCSR register (bit 0) in the same way as for a Bulk Rx endpoint. As the interrupt could occur almost any time within a frame(/microframe), the timing of FIFO unload requests will probably be irregular. If the data sink for the endpoint is going to some external hardware, it may be better to minimize the requirement for additional buffering by waiting until the end of each frame before unloading the FIFO. This can be done by using the SOF_PULSE signal from the controller to trigger the unloading of the data packet. The SOF_PULSE is generated once per frame(/microframe). The interrupts may still be used to clear the RXPKTRDY bit in HOST_RXCSR.

24.3.2.4.1.3 Isochronous IN Error Handling: Host Mode

If a CRC or bit-stuff error occurs during the reception of a packet, the packet will still be stored in the FIFO but the DATAERR_NAKTIMEOUT bit of HOST_RXCSR (bit 3) is set to indicate that the data may be corrupt.

Note: The number of USB packets sent in any microframe will depend on the amount of data to be transferred, and is indicated through the PIDs used for the individual packets. If the indicated numbers of packets have not been received by the end of a microframe, the INCOMPRX bit in the HOST_RXCSR register will be set to indicate that the data in the FIFO is incomplete. Equally, if a packet of the wrong data type is received, then the PID Error bit is the HOST_RXCSR register will be set. In each case, an interrupt will, however, still be generated to allow the data that has been received to be read from the FIFO.

24.3.2.4.2 Isochronous OUT Transactions: Host Mode

An Isochronous OUT transaction may be used to transfer periodic data from the host to the USB peripheral.

Following optional features are available for use with a Tx endpoint used in Host mode to transmit this data:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the peripheral device. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).

DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature can be used to allow the DMA controller to load packets into the FIFO without processor intervention.

However, this feature is not particularly useful with isochronous endpoints because the packets transferred are often not maximum packet size.

When DMA is enabled and endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

24.3.2.4.2.1 Isochronous OUT Transfer Setup: Host Mode

Before initiating any Isochronous OUT transactions:

- The target function address needs to be set in the TXFUNCADDR register for the selected controller endpoint (TXFUNCADDR register is available for all endpoints from EP0 to EP4).
- The HOST_TXTYPE register for the endpoint that is to be used needs to be programmed as:
  - Operating speed in the SPEED bit field (bits 7 and 6).
  - Set 01 (binary value) in the PROT field for isochronous transfer.
  - Endpoint Number of the target device in TENDPN field. This is the endpoint number contained in the OUT(Tx) endpoint descriptor returned by the target device during enumeration.
- The TXMAXP register for the controller endpoint must be written with the maximum packet size (in bytes) for the transfer. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the target endpoint.
- The HOST_TXINTERVAL register needs to be written with the required transaction interval (usually one transaction per frame/microframe).
- The relevant interrupt enable bit in the INTRTXE register should be set (if an interrupt is required for this endpoint).

- The following bits of HOST_TXCSR register should be set as:
  - Set the MODE bit (bit 13) to 1 to ensure the FIFO is enabled (only necessary if the FIFO is shared with an Rx endpoint).
  - Set the DMAEN bit (bit 12) to 1 if a DMA request is required for this endpoint
  - The FRCDATATOG bit (bit 11) is ignored for isochronous transactions.
  - Set the DMAMODE bit (bit 10) to 1 when DMA is enabled.
    For more details in using DMA, consult CPPI DMA section within this document.

24.3.2.4.2.2 Isochronous OUT Transfer Operation: Host Mode

The operation starts when the software writes to the FIFO and sets TXPKTRDY bit of HOST_TXCSR (bit 0). This triggers the controller to send an OUT token followed by the first data packet from the FIFO.

An interrupt is generated whenever a packet is sent and the software may use this interrupt to load the next packet into the FIFO and set the TXPKTRDY bit in the HOST_TXCSR register (bit 0) in the same way as for a Bulk Tx endpoint. As the interrupt could occur almost any time within a frame, depending on when the host has scheduled the transaction, this may result in irregular timing of FIFO load requests. If the data source for the endpoint is coming from some external hardware, it may be more convenient to wait until the end of each frame before loading the FIFO as this will minimize the requirement for additional buffering. This can be done by using the SOF_PULSE signal from the controller to trigger the loading of the next data packet. The SOF_PULSE is generated once per frame(/microframe). The interrupts may still be used to set the TXPKTRDY bit in HOST_TXCSR.

24.4 Communications Port Programming Interface (CPPI) 4.1 DMA

The CPPI DMA module supports the transmission and reception of USB packets. The CPPI DMA is designed to facilitate the segmentation and reassembly of CPPI compliant packets to/from smaller data blocks that are natively compatible with the specific requirements of each networking port. Multiple Tx and Rx channels are provided for all endpoints (excluding endpoint 0) within the DMA allowing multiple segmentation or reassembly operations to be effectively performed in parallel (but not actually simultaneously). The DMA controller maintains state information for each of the ports/channels which allows packet segmentation and reassembly operations to be time division multiplexed between channels in order to share the underlying DMA hardware. A DMA scheduler is used to control the ordering and rate at which this multiplexing occurs.

The CPPI (version 4.1) DMA controller sub-module is a common 15 port DMA controller. It supports 15 Tx and 15 Rx channels and each port attaches to the associated endpoint in the controller. Port 1 maps to endpoint 1 and Port 2 maps to endpoint 2 and so on with Port 15 mapped to endpoint 15; endpoint 0 can not utilize the DMA and the firmware is responsible to load or offload the endpoint 0 FIFO via CPU.

24.4.1 CPPI Terminology

Host — The host is an intelligent system resource that configures and manages each communications control module. The host is responsible for allocating memory, initializing all data structures, and responding to port interrupts.

Main Memory — The area of data storage managed by the CPU. The CPPI DMA (CDMA) reads and writes CPPI packets from and to main memory. This memory can exist internal or external from the device.

Queue Manager (QM) — The QM is responsible for accelerating management of a variety of Packet Queues and Free Descriptor / Buffer Queues. It provides status indications to the CDMA Scheduler when queues are empty or full.

CPPI DMA (CDMA) — The CDMA is responsible for transferring data between the CPPI FIFO and Main Memory. It acquires free Buffer Descriptor from the QM (Receive Submit Queue) for storage of received data, posts received packets pointers to the Receive Completion Queue, transmits packets stored on the Transmit Submit Queue (Transmit Queue) , and posts completed transmit packets to the Transmit Completion Queue.

CDMA Scheduler (CDMAS) — The CDMAS is responsible for scheduling CDMA transmit and receive operations. It uses Queue Indicators from the QM and the CDMA to determine the types of operations to schedule.

CPPI FIFO — The CPPI FIFO provides FIFO interfaces (for each of the 15 transmit and 15 receive endpoints).

Transfer DMA (XDMA) — The XDMA receives DMA requests from the Mentor USB 2.0 Core and initiates DMAs to the CPPI FIFO.

Endpoint FIFOs — The Endpoint FIFOs are the USB packet storage elements used by the Mentor USB 2.0 Core for packet transmission or reception. The XDMA transfers data between the CPPI FIFO and the Endpoint FIFOs for transmit operations and between the Endpoint FIFOs and the CPPI FIFO for receive operations.

Port — A port is the communications module (peripheral hardware) that contains the control logic for Direct Memory Access for a single transmit/receive interface or set of interfaces. Each port may have multiple communication channels that transfer data using homogenous or heterogeneous protocols. A port is usually subdivided into transmit and receive pairs which are independent of each other. Each endpoint, excluding endpoint 0, has its own dedicated port.

Channel — A channel refers to the sub-division of information (flows) that is transported across ports. Each channel has associated state information. Channels are used to segregate information flows based on the protocol used, scheduling requirements (example: CBR, VBR, ABR), or concurrency requirements (that is, blocking avoidance). All fifteen ports per USB Module have dedicated single channels, channel 0, associated for their use in a USB application.

Data Buffer — A data buffer is a single data structure that contains payload information for transmission to or reception from a port. A data buffer is a byte aligned contiguous block of memory used to store packet payload data. A data buffer may hold any portion of a packet and may be linked together (via descriptors) with other buffers to form packets. Data buffers may be allocated anywhere within the 32-bit memory space. The Buffer Length field of the packet descriptor indicates the number of valid data bytes in the buffer. There may be from 1 to 4M-1 valid data bytes in each buffer.

Host Buffer Descriptor (Buffer Descriptor) — A buffer descriptor is a single data structure that contains information about one or more data buffers. This type of descriptor is required when more than one descriptor is needed to define an entire packet, that is, it either defines the middle of a packet or end of a packet.

Start of Packet (SOP) — Packet along side Buffer Descriptors are used to hold information about Packets. When multiple Descriptors are used to hold a single packet information or a single Descriptor is used to hold a single packet information, the first descriptor is referred to as a Packet Descriptor which is also Start-of-Packet Descriptor.

Host Packet Descriptor (Packet Descriptor) — A packet descriptor is another name for the first buffer descriptor within a packet. Some fields within a data buffer descriptor are only valid when it is a packet descriptor including the tags, packet length, packet type, and flags. This type of descriptor is always used to define a packet since it provides packet level information that is useful to both the ports and the Host in order to properly process the packet. It is the only descriptor used when single descriptor solely defines a packet. When multiple descriptors are needed to define a packet, the packet descriptor is the first descriptor used to define a packet.

Free Descriptor/Buffer Queue — A free descriptor/buffer queue is a hardware managed list of available descriptors with pre-linked empty buffers that are to be used by the receive ports for host type descriptors. Free Descriptor/Buffer Queues are implemented by the Queue Manager.

Teardown Descriptor — Teardown Descriptor is a special structure which is not used to describe either a packet or a buffer but is instead used to describe the completion of a channel halt and teardown event. Channel teardown is an important function because it ensures that when a connection is no longer needed that the hardware can be reliably halted and any remaining packets which had not yet been transmitted can be reclaimed by the Host without the possibility of losing buffer or descriptor references (which results in a memory leak).

Packet Queue — A packet queue is hardware managed list of valid (populated) packet descriptors that is used for forwarding a packet from one entity to another for any number of purposes.
NOTE: All descriptors (regardless of type) must be allocated at addresses that are naturally aligned to the smallest power of 2 that is equal to or greater than the descriptor size.

24.4.2 Data Structures

Two data structures are mainly used to identify data buffers used by packet and buffer descriptors. A third Descriptor, Teardown Descriptor, exists. The purpose of this Descriptor is a channel halt and teardown event. Each of these Descriptor layouts as well as bit fields are shown below.

24.4.2.1 Host Packet Descriptor/ Packet Descriptor (SOP Descriptor)

Descriptors are designed to be used when USB like application requires support for true, unlimited fragment count scatter/gather type operations. The Packet Descriptor is the first descriptor on multiple descriptors setup or the only descriptor in a single descriptors setup. The Packet Descriptor contains the following information:

- Indicator which identifies the descriptor as a packet descriptor (always 10h
- Source and destination tags (reserved)
- Packet type
- Packet length
- Protocol specific region size
- Protocol specific control/statusbits
- Pointer to the first valid byte in the SOP data buffer
- Length of the SOP data buffer
- Pointer to the next buffer descriptor in the

Packet descriptors can vary in size by design of their defined fields from 32 bytes up to 104 bytes. Within this range, packet descriptors always contain 32 bytes of required information and may also contain 8 bytes of software specific tagging information and up to 64 bytes (indicated in 4 byte increments) of protocol specific information. How much protocol specific information (and therefore the allocated size of the descriptors) is application dependent. Port will make use of the first 32 bytes only.

From a general USB use perspective, a 32-byte descriptor size is suffix and the use of this size is expected for a normal USB usage.

The packet descriptor layout is shown in Figure 24-12 and described within Table 24-6 through Table 24-13.

Figure 24-12. Packet Descriptor Layout

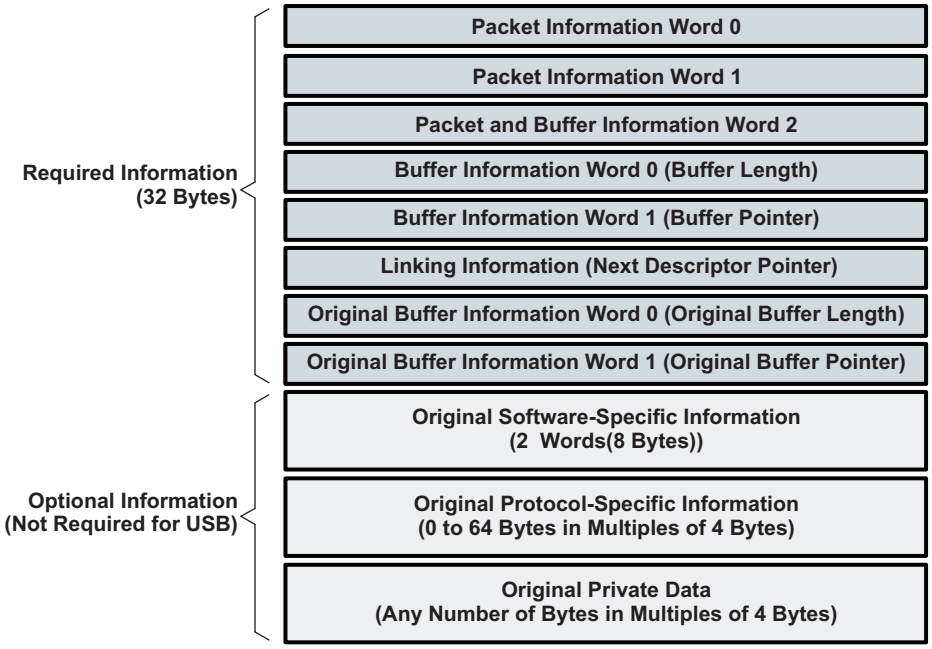


Table 24-6. Packet Descriptor Word 0 (PD0) Bit Field Descriptions

Bits	Field	Description
31-27	Descriptor type	The host packet descriptor type is 16 decimal (10h). The CPU initializes this field.
26-22	Protocol-specific valid word count	This field indicates the valid number of 32-bit words in the protocol-specific region. The CPU initializes this field. This is encoded in increments of 4 bytes as: 0 = 0 byte 1 = 4 bytes ... 16 = 64 bytes 17-31 = Reserved
21-0	Packet length	The length of the packet in bytes. If the packet length is less than the sum of the buffer lengths, then the packet data will be truncated. A packet length greater than the sum of the buffers is an error. The valid range for the packet length is 0 to (4M - 1) bytes. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 24-7. Packet Descriptor Word 1 (PD1) Bit Field Descriptions

Bits	Field	Description
31-27	Source Tag; Port #	This field indicates the port number (0-31) from which the packet originated. The DMA overwrites this field on packet reception. This is the RX Endpoint number from which the packet originated.
26-21	Source Tag; Channel #	This field indicates the channel number within the port from which the packet originated. The DMA overwrites this field on packet reception. This field is always 0-67.
20-16	Source Tag: Sub-channel #	This field indicates the sub-channel number (0-31) within the channel from which the packet originated. The DMA overwrites this field on packet reception. This field is always 0.
15-0	Destination Tag	This field is application-specific. This field is always 0.

Table 24-8. Packet Descriptor Word 2 (PD2) Bit Field Descriptions

Bits	Field	Description
31	Packet error	This bit indicates if an error occurred during reception of this packet (0 = no error occurred; 1 = error occurred). The DMA overwrites t his field on packet reception. Additional information about different errors may be encoded in the protocol specific fields in the descriptor.
30-26		This field indicates the type of this packet. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception. This field is encoded as: 5 = USB 8-31 = Reserved
25-20	Reserved	Reserved
19	Zero-length packet indicator	If a zero-length USB packet is received, the XDMA will send the CDMA a data block with a byte count of 0 and this bit is set. The CDMA will then perform normal EOP termination of the packet without transferring data. For transmit, if a packet has this bit set, the XDMA will ignore the CPPI packet size and send a zero-length packet to the USB controller.
18-16	Protocol-specific	This field contains protocol-specific flags/information that can be assigned based on the packet type. Not used for USB.
15	Return policy	This field indicates the return policy for this packet. The CPU initializes this field. 0 = Entire packet (still linked together) should be returned to the queue specified in bits 13-0. 1 = Each buffer should be returned to the queue specified in bits 13-0 of Word 2 in their respective descriptors. The Tx DMA will return each buffer in sequence.
14	On-chip	This field indicates whether or not this descriptor is in a region which is in on-chip memory space (1) or in external memory (0).
31-12	Packet return queue mgr #	This field indicates which queue manager in the system the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmissionor reception and is initialized by the CPU. There is only one queue manager in the USB HS/FS device controller. This field must always be 0.
11-0	Packet return queue #	This field indicates the queue number within the selected queue manager that the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU.

Table 24-9. Packet Descriptor Word 3 (PD3) Bit Field Descriptions

Bits	Field	Description
31-22	Reserved	Reserved
21-0	Buffer 0 length	The buffer length field indicates how many valid data bytes are in the buffer. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 24-10. Packet Descriptor Word 4 (PD4) Bit Field Descriptions

Bits	Field	Description
31-0	Buffer 0 pointer	The buffer pointer is the byte-aligned memory address of the buffer associated with the buffer descriptor. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 24-11. Packet Descriptor Word 5 (PD5) Bit Field Descriptions

Bits	Field	Description
31-0	Next descriptor pointer	The 32-bit word aligned memory address of the next buffer descriptor in the packet. If the value of this pointer is zero, then the current buffer is the last buffer in the packet. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 24-12. Packet Descriptor Word 6 (PD6) Bit Field Descriptions

Bits	Field	Description
31-22	Reserved	Reserved
21-0	Original buffer 0 length	The buffer length field indicates the original size of the buffer in bytes. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer size as allocated by the CPU at initialization. Since the buffer length in Word 3 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer size information.

Table 24-13. Packet Descriptor Word 7 (PD7) Bit Field Descriptions

Bits	Field	Description
31-22	Reserved	Reserved
21-0	Original buffer 0 pointer	The buffer pointer is the byte-aligned memory address of the buffer associated with the buffer descriptor. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer location as allocated by the CPU at initialization. Since the buffer pointer in Word 4 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer pointer information.

24.4.2.2 Host Buffer Descriptor/Buffer Descriptor (BD)

The buffer descriptor is identical in size and organization to a Packet Descriptor but does not include valid information in the packet level fields and does not include a populated region for protocol specific information. The packet level fields are not needed since the SOP descriptor contains this information and additional copy of this data is not needed/necessary.

Host buffer descriptors are designed to be linked onto a host packet descriptor or another host buffer descriptor to provide support for unlimited scatter / gather type operations. Host buffer descriptors provide information about a single corresponding data buffer. Every host buffer descriptor stores the following information:

- Pointer to the first valid byte in the data
- Length of the data buffer
- Pointer to the next buffer descriptor in the packet

Buffer descriptors always contain 32 bytes of required information. Since it is a requirement that it is possible to convert a descriptor between a Buffer Descriptor and a Packet Descriptor (by filling in the appropriate fields) in practice, Buffer Descriptors will be allocated using the same sizes as Packet Descriptors. In addition, since the 5 LSBs of the Descriptor Pointers are used in CPPI 4.1 for the purpose of indicating the length of the descriptor, the minimum size of a descriptor is always 32 bytes

The buffer descriptor layout is shown in Figure 24-13 and described within Table 24-14 through Table 24-21 .

Figure 24-13. Buffer Descriptor (BD) Layout

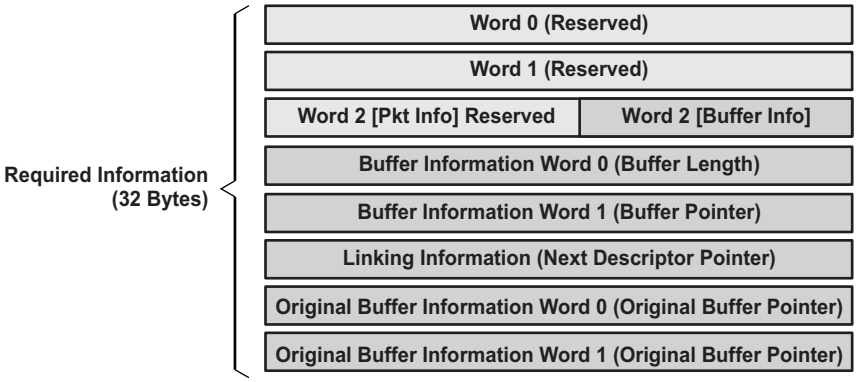


Table 24-14. Buffer Descriptor Word 0 (BD0) Bit Field Descriptions

Bits	Field	Description
31-0	Reserved	Reserved

Table 24-15. Buffer Descriptor Word 1 (BD1) Bit Field Descriptions

Bits	Field	Description
31-0	Reserved	Reserved

Table 24-16. Buffer Descriptor Word 2 (BD2) Bit Field Descriptions

Bits	Field	Description
31-15	Reserved	Reserved
14	On-chip	This field indicates whether or not this descriptor is in a region which is on-chip memory space (1) or in external memory (0).
13-12	Packet return queue mgr #	This field indicates which queue manager in the system the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU. There is only 1 queue manager in the USB HS/FS device controller. This field must always be 0.
11-0	Packet return queue #	This field indicates the queue number within the selected queue manager that the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU.

Table 24-17. Buffer Descriptor Word 3 (BD3) Bit Field Descriptions

Bits	Field	Description
31-22	Reserved	Reserved
21-0	Buffer 0 length	The buffer length field indicates how many valid data bytes are in the buffer. The CPU initializes this field for transmitted packets. The DMA overwrites this field upon packet reception.

Table 24-18. Buffer Descriptor Word 4 (BD4) Bit Field Descriptions

Bits	Field	Description
31-0	Buffer 0 pointer	The buffer pointer is the byte-aligned memory address of the buffer associated with the buffer descriptor. The CPU initializes this field for transmitted packets. The DMA overwrites this field upon packet reception.

Table 24-19. Buffer Descriptor Word 5 (BD5) Bit Field Descriptions

Bits	Field	Description
31-0	Next description pointer	The 32-bit word aligned memory address of the next buffer descriptor in the packet. If the value of this pointer is zero, then the current descriptor is the last descriptor in the packet. The CPU initializes this field for transmitted packets. The DMA overwrites this field upon packet reception.

Table 24-20. Buffer Descriptor Word 6 (BD6) Bit Field Descriptions

Bits	Field	Description
31-22	Reserved	Reserved
21-0	Original buffer 0 length	The buffer length field indicates the original size of the buffer in bytes. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer size as allocated by the CPU at initialization. Since the buffer length in Word 3 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer size information.

Table 24-21. Buffer Descriptor Word 7 (BD7) Bit Field Descriptions

Bits	Field	Description
31-0	Original buffer 0 pointer	The buffer pointer is the byte-aligned memory address of the buffer associated with the buffer descriptor. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer location as allocated by the CPU at initialization. Since the buffer pointer in Word 4 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer pointer information.

24.4.2.3 Teardown Descriptor

The Teardown Descriptor is not like the packet or buffer descriptors since it is not used to describe either a packet or a buffer. The teardown descriptor is always 32 bytes long and is comprised of 4 bytes of actual teardown information and 28 bytes of pad. The teardown descriptor layout and associated bit field descriptions are shown in the Figure below and Tables that follows. Since the 5 LSBs of the descriptor pointers are used in CPPI 4.1 for the purpose of indicating the length of the descriptor, the minimum size of a descriptor is 32 bytes.

The teardown descriptor is used to describe a channel halt and teardown event. Channel teardown ensures that when a connection is no longer needed that the hardware can be reliably halted and any remaining packets which had not yet been transmitted can be reclaimed by the host without the possibility of losing buffer or descriptor references (which results in a memory leak).

Theteardown descriptor contains the following information:

- Indicator which identifies the descriptor as a teardown packet descriptor
- DMA controller number where teardown
- Channel number within DMA where teardown occurred
- Indicator of whether this teardown was for the Tx or Rx channel

Figure 24-14. Teardown Descriptor Layout

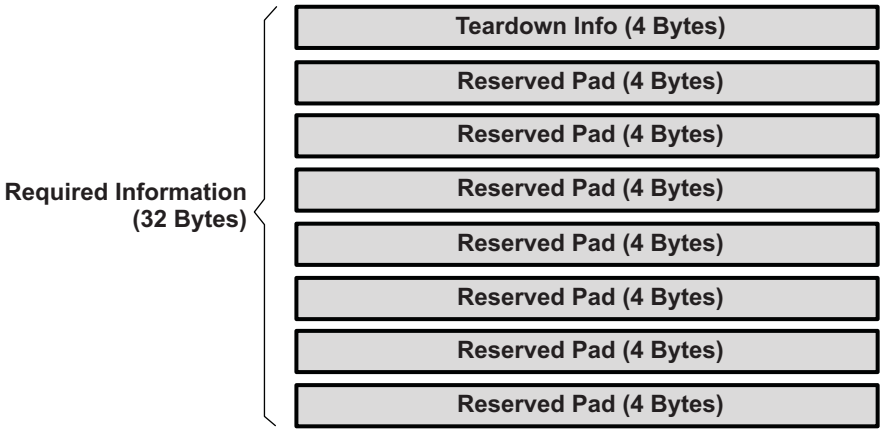


Table 24-22. Teardown Descriptor Word 0 Bit Field Descriptions

Bits	Field	Description
31-27	Descriptor type	The teardown descriptor type is 19 decimal (13h)
26-17	Reserved	Reserved
16	TX_RX	Indicates whether teardown is a TX (0) or RX (1).
15-10	DMA number	Indicates the DMA number for this teardown.
9-6	Reserved	Reserved
5-0	Channel number	Indicates the channel number within the DMA that was torn down.

Table 24-23. Teardown Descriptor Words 1 to 7 Bit Field Descriptions

Bits	Field	Description
31-0	Reserved	Reserved

Teardown operation of an endpoint requires three operations. The teardown register in the CPPI DMA must be written, the corresponding endpoint bit in TEARDOWN of the USB module must be set, and the FlushFIFO bit in the Mentor USB controller Tx/RxCSR register must be set.

The following is the Transmit teardown procedure highlighting the steps required to be followed:

1. Set the TX_TEARDOWN bit in the CPPI DMA TX channel n global configuration register (TXGCRn).
2. Set the appropriate TX_TDOWN bit in the USBOTG controller's USB teardown register (TEARDOWN). Write Tx Endpoint Number to teardown to TEARDOWN[TX_TDOWN] field.
3. Check if the teardown descriptor has been received on the teardown queue: The completion queue (see Queue Assignment table) is usually used as the Teardown queue when the Teardown descriptor has been received, the descriptor address will be loaded onto CTRLD[Completion Queue #] register:
   (a) If not, go to step 2
   (b) If so, go to step 4
4. Set the appropriate TX_TDOWN bit in the USBOTG controller's USB teardown register (TEARDOWN). Set the bit corresponding to the Channel Number within TEARDOWN[TX_TDOWN] field.
5. Flush the TX FIFO in the Mentor OTG core: Set PERI_TXCSR[FLUSHFIFO] for the corresponding Endpoint.
6. Re-enable the Tx DMA channel.
   (a) Clear TXGCRn[TX_TEARDOWN and TX_ENABLE] bit.
   (b) Set TXGCRn[TX_ENABLE] bit.

24.4.3 Queue Manager

The queue manager (QM) is a hardware module that is responsible for accelerating management of the queues, that is, queues are maintained within a queue manager module. Packets are added to a queue by writing the 32-bit descriptor address to a particular memory mapped location in the queue manager module. Packets are de-queued by reading the same location for that particular queue. A single queue manager exists within the device and handles all available 156 queues within the USB subsystem

Descriptors are queued onto a logical queue (pushed) by writing the descriptor pointer to the Queue N Register D. When the Queue N Register D is written, this kicks off the queue manager causing it to add the descriptor to the tail of queue.

The QM keeps track of the order of the descriptors pushed within a queue or the linking status of the descriptors. To accomplish this linking status, QM will first resolve the 32-bit descriptor pointer into a 16-bit index which is used for linking a queue pointer purposes. Once the address to index computation is performed, that is, the physical index information is determined, the QM will link that descriptor onto the descriptor chain that is maintained for that logical queue by writing the linking information out to a linking RAM which is external to the queue manager. More on linking RAM discussion would follow in latter sections. The QM will also update the queue tail pointers appropriately. Since logical queues within the queue manager are maintained using linked lists, queues cannot become full and no check for fullness is required before queuing a packet descriptor.

24.4.3.1 Queue Types

Several types of queues exist (a total of 156 queues) within the CPPI 4.1 DMA. Regardless of the type of queue, queues are used to hold pointers to Packet or Buffer Descriptors while they are being passed between the Host and / or any of the ports in the system. All queues are maintained within the single Queue Manager module.

The following types of Queues exist:

- Free descriptor queue (unassigned to specific endpoint but assigned to specific endpoint type; receive endpoints – can be used as a receive submit queue)
- Transmit submit queue
- Transmit completion (return) queue
- Receive completion (return) queue
- Teardown queue

Dedicated queues exist where one or more queues are assigned for the use of a single endpoint and non-dedicated queues exist where no specific queue assignment to an endpoint is required (but pertains to receive endpoints only). Transmit endpoints are not allowed to use free descriptor queues.

Dedicated queues exist for each specific endpoint use and non-dedicated queues exist that can be used by any/all receive endpoints. Three queues are reserved for each endpoint/port for transmit actions with two of the three queues being transmit submit queues while the remaining queue is used as a completion/return queue. For receive actions, the only dedicated queues are completion/return queues; one queue is assigned/reserved for each receive endpoint. 32 free descriptor queues that do not have dedicated endpoint-queue assignment exist and these queues are used to service any receive endpoint as receive submit queues.

Table 24-24 displays queue-endpoint assignments.

Table 24-24. Queue-Endpoint Assignments

Queue Start Number	Queue Count	Queue-Endpoint Association
0	32	Free Descriptor Queues/Rx Submit Queues (USB0/1 Rx Endpoints)
32	2	USB0 Tx Endpoint 1
34	2	USB0 Tx Endpoint 2
36	2	USB0 Tx Endpoint 3
38	2	USB0 Tx Endpoint 4

Table 24-24. Queue-Endpoint Assignments (continued)

Queue Start Number	Queue Count	Queue-Endpoint Association
40	2	USB0 Tx Endpoint 5
42	2	USB0 Tx Endpoint 6
44	2	USB0 Tx Endpoint 7
46	2	USB0 Tx Endpoint 8
48	2	USB0 Tx Endpoint 9
50	2	USB0 Tx Endpoint 10
52	2	USB0 Tx Endpoint 11
54	2	USB0 Tx Endpoint 12
56	2	USB0 Tx Endpoint 13
58	2	USB0 Tx Endpoint 14
60	2	USB0 Tx Endpoint 15
62	2	USB1 Tx Endpoint 1
64	2	USB1 Tx Endpoint 2
66	2	USB1 Tx Endpoint 3
68	2	USB1 Tx Endpoint 4
70	2	USB1 Tx Endpoint 5
72	2	USB1 Tx Endpoint 6
74	2	USB1 Tx Endpoint 7
76	2	USB1 Tx Endpoint 8
78	2	USB1 Tx Endpoint 9
80	2	USB1 Tx Endpoint 10
82	2	USB1 Tx Endpoint 11
84	2	USB1 Tx Endpoint 12
86	2	USB1 Tx Endpoint 13
88	2	USB1 Tx Endpoint 14
90	2	USB1 Tx Endpoint 15
92	1	Reserved
93	1	USB0 Tx Endpoint 1 completion queue
94	1	USB0 Tx Endpoint 2 completion queue
95	1	USB0 Tx Endpoint 3 completion queue
96	1	USB0 Tx Endpoint 4 completion queue
97	1	USB0 Tx Endpoint 5 completion queue
98	1	USB0 Tx Endpoint 6 completion queue
99	1	USB0 Tx Endpoint 7 completion queue
100	1	USB0 Tx Endpoint 8 completion queue
101	1	USB0 Tx Endpoint 9 completion queue
102	1	USB0 Tx Endpoint 10 completion queue
103	1	USB0 Tx Endpoint 11 completion queue
104	1	USB0 Tx Endpoint 12 completion queue
104	1	USB0 Tx Endpoint 13 completion queue
106	1	USB0 Tx Endpoint 14 completion queue
107	1	USB0 Tx Endpoint 15 completion queue
108	1	Reserved
109	1	USB0 Rx Endpoint 1 completion queue
110	1	USB0 Rx Endpoint 2 completion queue
111	1	USB0 Rx Endpoint 3 completion queue
112	1	USB0 Rx Endpoint 4 completion queue

Table 24-24. Queue-Endpoint Assignments (continued)

Queue Start Number	Queue Count	Queue-Endpoint Association
113	1	USB0 Rx Endpoint 5 completion queue
114	1	USB0 Rx Endpoint 6 completion queue
115	1	USB0 Rx Endpoint 7 completion queue
116	1	USB0 Rx Endpoint 8 completion queue
117	1	USB0 Rx Endpoint 9 completion queue
118	1	USB0 Rx Endpoint 10 completion queue
119	1	USB0 Rx Endpoint 11 completion queue
120	1	USB0 Rx Endpoint 12 completion queue
121	1	USB0 Rx Endpoint 13 completion queue
122	1	USB0 Rx Endpoint 14 completion queue
123	1	USB0 Rx Endpoint 15 completion queue
124	1	Reserved
125	1	USB1 Tx Endpoint 1 completion queue
126	1	USB1 Tx Endpoint 2 completion queue
127	1	USB1 Tx Endpoint 3 completion queue
128	1	USB1 Tx Endpoint 4 completion queue
129	1	USB1 Tx Endpoint 5 completion queue
130	1	USB1 Tx Endpoint 6 completion queue
131	1	USB1 Tx Endpoint 7 completion queue
132	1	USB1 Tx Endpoint 8 completion queue
133	1	USB1 Tx Endpoint 9 completion queue
134	1	USB1 Tx Endpoint 10 completion queue
135	1	USB1 Tx Endpoint 11 completion queue
136	1	USB1 Tx Endpoint 12 completion queue
137	1	USB1 Tx Endpoint 13 completion queue
138	1	USB1 Tx Endpoint 14 completion queue
139	1	USB1 Tx Endpoint 15 completion queue
140	1	Reserved
141	1	USB1 Rx Endpoint 1 completion queue
142	1	USB1 Rx Endpoint 2 completion queue
143	1	USB1 Rx Endpoint 3 completion queue
144	1	USB1 Rx Endpoint 4 completion queue
145	1	USB1 Rx Endpoint 5 completion queue
146	1	USB1 Rx Endpoint 6 completion queue
147	1	USB1 Rx Endpoint 7 completion queue
148	1	USB1 Rx Endpoint 8 completion queue
149	1	USB1 Rx Endpoint 9 completion queue
150	1	USB1 Rx Endpoint 10 completion queue
151	1	USB1 Rx Endpoint 11 completion queue
152	1	USB1 Rx Endpoint 12 completion queue
153	1	USB1 Rx Endpoint 13 completion queue
154	1	USB1 Rx Endpoint 14 completion queue
155	1	USB1 Rx Endpoint 15 completion queue

24.4.3.2 Free Descriptor Queue (Receive Submit Queue)

Receive endpoints/channels use queues, referred to as free descriptor queues, or receive submit queues, to forward completed receive packets to the host. The entries on the free descriptor queues have pre-attached empty buffers whose size and location are described in the original buffer information fields in the descriptor. The host is required to allocate both the descriptor and buffer and pre-link them prior to adding (submitting) a descriptor to one of the available free descriptor queue. The first 32 queues (queue 0 up to queue 31) are reserved for all 30 USB0/1 (endpoints 1 to 15 of each USB module) receive endpoints/channels to handle incoming packets to the device.

24.4.3.3 Transmit Submit Queue

Transmit ports use packet queues, referred to as transmit submit queues, to store the packets that are waiting to be transmitted. Each transmit endpoint has dedicated queues (2 queues per port) that are reserved exclusively for a use by a single endpoint. Multiple queues per port/channel are allocated to facilitate quality of service (QoS) for applications that require QoS. The first 30 queues, queue 32 up to queue 61, are allocated for USB0 transmit endpoints 1 to 15 with two queues per endpoint.queues 62 up to queue 91, are allocated for transmit USB1 endpoints 1 to 15.

24.4.3.4 Transmit Completion (Return) Queue

Transmit ports use packet queues, referred to as "transmit completion queues, to return packet descriptors to the host after packets have been transmitted. A singe queue is reserved to be used by a single transmit endpoint. Application s/w needs to insure that the right set of queues is used to return the Descriptors after the completion of a packet transmission based on the endpoint number used. For USB0 transmit endpoints 1 to 15 Queues 92 up to queue 106 are reserved/assigned for use as a completion queue respectively. Similarly for USB1 transmit endpoints 1 to 15, queues 107 up to queue 121 is used as completion queue, respectively.

Transmit Completion Queues are also used to return packet Descriptors when performing a Transmit channel teardown operation.

24.4.3.5 Receive Completion (Return) Queue

Receive ports use packet queues, referred to as receive completion queues, to return packet descriptors to the port after packets have been received. A singe queue is reserved to be used by a single receive endpoint. Application s/w needs to insure that the right set of queues is used to return the descriptors after the completion of a packet reception based on the endpoint number used. For USB0 receive endpoints 1 to 15 queues 122 up to queue 136 are reserved/assigned for use as a receive completion queue respectively. Similarly for USB1 receive endpoints 1 to 15, queues 137 up to queue 151 is used as completion queue respectively.

Receive channel teardown is not necessary for receive transactions and no channel teardown information nor resource is available

24.4.3.6 Diverting Queue Packets from one Queue to Another

The host can move the entire contents of one queue to another queue by writing the source queue number and the destination queue number to the Queue Diversion register. When diverting packets, the descriptors are pushed onto the tail of the destination queue.

24.4.3.7 Teardown Queue

The Teardown Queue is used by the DMA to communicate a completion of a channel teardown after a channel teardown is invoked on to a channel. The pointer to the teardown descriptor is written to the teardown queue, which is also the Completion Queue, when the channel teardown completes.

24.4.4 Memory Regions and Linking RAM

In addition to allocating buffer for raw data and memory for Descriptors, the host is responsible for allocating additional memory for exclusive use of the CPPI DMA Queue Manager to be used as a scratch PAD RAM. The queue manager uses this memory to manage states of descriptors submitted within the submit queues. In other words, this memory needs not to be managed by the user software and firmware responsibility lies only for allocation/reserving a chunk of memory for the use of the queue manager. The allocated memory can be a single block of memory that is contiguous or two blocks of memory that are not contiguous. These two blocks of memory are referred to as a Linking RAM Regions and should not be confused with memory regions that are used to store descriptors. That is, the use of the term *region* should be used in the context of its use.

To accomplish the linking of submitted descriptors, the queue manager will first resolve the 32-bit descriptor pointer into a 16-bit index which is used for linking and queue pointer purposes. Once the physical index information is determined, the queue manager will link that descriptor onto the descriptor chain that is maintained for that logical queue by writing the linking information out to a linking RAM which is external to the queue manager. The queue manager will also update the queue tail pointer appropriately. Since logical queues within the queue manager are maintained using linked lists, queues cannot become full and no check for fullness is required before queuing a packet descriptor.

The actual physical size of the Linking RAM region(s) to be allocated depends on the total number of descriptors defined within all memory regions. A minimum of four bytes of memory needs to be allocated for each Descriptor defined within all 16 memory regions.

The queue manager has the capability of managing up to 16 memory regions. These memory regions are used to store descriptors of variable sizes. The total number of descriptors that can be managed by the queue manager should not exceed 64K. Each memory region has descriptors of one configurable size, that is, descriptors with different sizes cannot be housed within a single memory region. These 64K descriptors are referenced internally in the queue manager by a 16-bit quantity index.

The information about the Linking RAM regions and the size that are allocated is communicated to the CPPI DMA via three registers dedicated for this purpose. Two of the three registers are used to store the 32-bit aligned start addresses of the Linking RAM regions. The remaining one register is used to store the size of the first Linking RAM. The linking RAM size value stored here is the number of descriptors that is to be managed by the queue manager within that region not the physical size of the buffer, which is four times the number of descriptors.

Note that application is not required to use both linking RAM regions, if the size of the Linking RAM for the first region is large enough to accommodate all descriptors defined. No linking RAM size register for linking RAM region 2 exists. The size of the second linking RAM, when used, is indirectly computed from the total number of descriptors defined less the number of descriptors managed by the first linking RAM

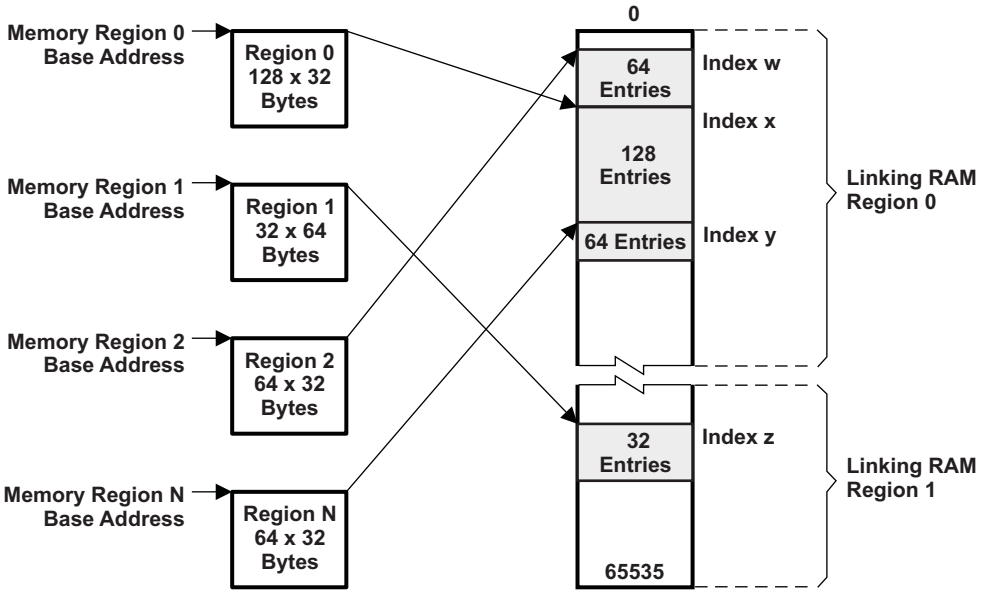
Figure 24-15 displays the relationship between several memory regions and linking RAM.

24.4.5 Zero Length Packets

A special case is the handling of null packets with the CPPI 4.1 compliant DMA controller. Upon receiving a zero length USB packet, the XFER DMA will send a data block to the DMA controller with byte count of zero and the zero byte packet bit of INFO Word 2 set. The DMA controller will then perform normal End of Packet termination of the packet, without transferring data.

If a zero-length USB packet is received, the XDMA will send the CDMA a data block with a byte count of 0 and this bit set. The CDMA will then perform normal EOP termination of the packet without transferring data. For transmit, if a packet has this bit set, the XDMA will ignore the CPPI packet size and send a zero-length packet to the USB controller.

Figure 24-15. Relationship Between Memory Regions and Linking RAM



24.4.6 CPPI DMA Scheduler

The CPPI DMA scheduler is responsible for controlling the rate and order between the different Tx and Rx threads/transfers that are provided in the CPPI DMA controller. The scheduler table RAM exists within the scheduler.

The DMA controller maintains state information for each of the channels which allows packet segmentation and reassembly operations to be time division multiplexed between channels in order to share the underlying DMA hardware. A DMA scheduler is used to control the ordering and rate at which this multiplexing occurs.

24.4.6.1 CPPI DMA Scheduler Operation

Once the scheduler is enabled it will begin processing the entries in the table. When appropriate the scheduler will pass credits to the DMA controller to perform a Tx or Rx operation. The operation of the DMA controller is as follows:

1. After the DMA scheduler is enabled it begins with the table index cleared to 0.
2. The scheduler reads the entry pointed to by the index and checks to see if the channel in question is currently in a state where a DMA operation can be accepted.
   (a) The DMA channel must be enabled AND
   (b) The CPPI FIFO that the channel talks to has free space on TX (FIFO full signal is not asserted) or a valid block on Rx (FIFO empty signal is not asserted)
3. If the DMA channel is capable of processing a credit to transfer a block, the DMA scheduler will issue that credit via the DMA scheduling interface which is a point to point connection between the DMA Scheduler and the DMA Controller.
   (a) The DMA controller may not be ready to accept the credit immediately and is provided a sched_ready signal which is used to stall the scheduler until it can accept the credit. The DMA controller only asserts the sched_ready signal when it is in the IDLE state.
   (b) Once a credit has been accepted (indicated by sched_req and sched_ready both asserted), the scheduler will increment the index to the next entry and will start again at step 2.
4. If the channel in question is not currently capable of processing a credit, the scheduler will increment the index in the scheduler table to the next entry and will start at step 2.
5. Note that when the scheduler attempts to increment its index, to the value programmed in the table size register, the index will be reset to 0.

24.4.6.1.1 CPPI DMA Scheduler Initialization

Before the scheduler can be used, the host is required to initialize and enable the block. This initialization is performed as follows:

1. The Host initializes entries within an internal memory array in the scheduler. This array contains up to 256 entries (4 entries per table word n where n=0-63) and each entry consists of a DMA channel number and a bit indicating if this is a Tx or Rx opportunity. These entries represent both the order and frequency that various Tx and Rx channels will be processed. A table size of 256 entries allows channel bandwidth to be allocated with a maximum precision of 1/256th of the total DMA bandwidth. The more entries that are present for a given channel, the bigger the slice of the bandwidth that channel will be given. Larger tables can be accommodated to allow for more precision. This array can only be written by the Host, it cannot be read.
2. If the application does not need to use the entire 256 entries, firmware can initialize the portion of the 256 entries and indicate the size of the entries used by writing onto an internal register in the scheduler which sets the actual size of the array (it can be less than 256 entries).
3. The host writes an internal register bit to enable the scheduler. The scheduler is not required to be disabled in order to change the scheduler array contents.

24.4.6.1.2 CPPI DMA Scheduler Programming Examples

Consider a three endpoints use on a system with the following configurations: EP1-Tx, EP2-Rx, and EP2-Tx. Two assumptions are considered:

Case 1: Assume that you would like to service each enabled endpoints (EP1-Tx, EP2-Rx, and EP2-Tx) with equal priority.

The scheduler handles the rate at which an endpoint is serviced by the number of credits programmed (entries) for that particular endpoint within the scheduler Table Words. The scheduler has up to 256 credits that it can grant and for this example the user can configure the number of entries/credits to be anywhere from 3 to 256 with a set of 3 entries.

However, the optimum and direct programming for this scenario would be programming only the first three entries of the scheduler via scheduler Table WORD[0]. Since this case expects the Scheduler to use only the first three entries, you communicate that by programming DMA_SCHED_CTRL.LAST_ENTRY with 2 (that is, 3 -1). The Enabled Endpoint numbers and the data transfer direction is then communicated by programming the first three entries of WORD[0] (ENTRY0_CHANNEL = 1: ENTRY0_RXTX = 0; ENTRY1_CHANNEL = 2: ENTRY1_RXTX = 1;ENTRY2_CHANNEL = 2: ENTRY2_RXTX = 0). With this programming, the Scheduler will only service the first three entries in a round-robin fashion, checking each credited endpoint for transfer one after the other, and servicing the endpoint that has data to transfer.

Case 2: Enabled endpoint EP1-Tx is serviced at twice the rate as the other enabled endpoints (EP2-Rx and EP2-Tx).

The number of entries/credit that has to be awarded to EP1-Tx has to be twice as much of the others. Four entries/credits would suffix to satisfy our requirement with two credits for EP1-Tx, one credit for EP2-Rx, and one credit for EP2-Tx. This requirement is satisfied by allocating any 2 of the 4 entries to EP1-Tx endpoint. Again for this example, scheduler Table WORD[0] would suffice since it can handle the first 4 entries. Even though several scenarios exist to programming the order of service for this case, one scenario would be to allow servicing EP1-Tx to back-to-back followed by the other enabled endpoints. Program DMA_SCHED_CTRL.LAST_ENTRY with 3 (that is, 4 -1). Program WORD[0] (ENTRY0_CHANNEL = 1: ENTRY0_RXTX = 0; ENTRY1_CHANNEL = 1: ENTRY1_RXTX = 0; ENTRY2_CHANNEL = 2: ENTRY2_RXTX = 1; ENTRY3_CHANNEL = 2: ENTRY3_RXTX = 0).

24.4.7 CPPI DMA State Registers

The port must store and maintain state information for each transmit and receive port/channel. The state information is referred to as the Tx DMA State and Rx DMA State.

24.4.7.1 Transmit DMA State Registers

The Tx DMA State is a combination of control fields and protocol specific port scratchpad space used to manipulate data structures and transmit packets. Each transmit channel has two queues. Each queue has a one head descriptor pointer and one completion pointer. There are thirty Tx DMA State registers; one for each port/channel.

The following information is stored in the Tx DMA State:

- Tx Queue Head Descriptor Pointer(s)
- Tx Completion Pointer(s)
- Protocol specific control/status (port scratchpad)

24.4.7.2 Receive DMA State Registers

The Rx DMA State is a combination of control fields and protocol specific port scratchpad space used to manipulate data structures in order to receive packets. Each receive channel has only one queue. Each channel queue has one head descriptor pointer and one completion pointer. There are thirty Rx DMA State registers; one for each port/channel.

The following information is stored in the Rx DMA State:

- Rx Queue Head Descriptor Pointer
- Rx Queue Completion Pointer
- Rx Buffer Offset

24.4.8 CPPI DMA Protocols Supported

Four different type of DMA transfers are supported by the CPPI 4.1 DMA; Transparent, RNDIS, Generic RNDIS, and Linux CDC. The following sections will outline the details on these DMA transfer types.

24.4.8.1 Transparent DMA Transfer

Transparent Mode DMA operation is the default DMA mode where DMA interrupt is generated whenever a DMA packet is transferred. In the transparent mode, DMA packet size cannot be greater than USB MaxPktSize for the endpoint. This transfer type is ideal for transfer (not packet) sizes that are less than a max packet size.

24.4.8.1.1 Transparent DMA Transfer Setup

The following will configure all 30 ports/channels for Transparent DMA Transfer type.

Make sure that RNDIS Mode is disabled globally. This allows the application to configure the CPPI DMA protocol in use to be configured per endpoint need. To disable RNDIS operation, clear RNDIS bit in the USB Control Registers corresponding to the USB Module (default setting), that is, CTRL0[RNDIS] = 0 and CTRL1[RNDIS] = 0.

Configure the USB0/1 Tx/Rx DMA Mode Registers (USB0/1 TX(RX)MODE0/1) for the Endpoint field in use is programmed for Transparent Mode (TXMODE0/1[TXn_MODE] = 00b and RXMODE0/1[RXn_MODE] = 00b).

24.4.8.2 RNDIS DMA Transfer

RNDIS mode DMA is used for large transfers (total data size to be transferred is greater than USB MaxPktSize where the MzxPktSize is a multiple of 64 bytes) that requires multiple USB packets. This is accomplished by breaking the larger packet into smaller packets, where each packet size being USB MaxPktSize except the last packet where its size is less than USB MaxPktSize, including zero bytes. This implies that multiple USB packets of MaxPktSize will be received and transferred together as a single large DMA transfer and the DMA interrupt is generated only at the end of the complete reception of DMA transfer. The protocol defines the end of the complete transfer by receiving a short USB packet (smaller than USB MaxPktSize as mentioned in USB specification 2.0). If the DMA packet size is an exact multiple of USB MaxPktSize, the DMA controller waits for a zero byte packet at the end of complete transfer to signify the completion of the transfer.

NOTE: RNDIS Mode DMA is supported only when USB MaxPktSize is an integral multiple of 64 bytes.

24.4.8.2.1 RNDIS DMA Transfer Setup

If all endpoints in use are desired to operate in RNDIS mode, it is only suffix to configure RNDIS DMA operation in RNDIS mode at the global level and application can ignore individual endpoint DMA mode configuration. This is achieved by programming CTRLR0/1[RNDIS] = 1.

However if DMA mode is required to be configured at the endpoint level, it is required to disable the use of RNDIS at the global level, this is achieved by clearing RNDIS bit field (CTRLR0/1[RNDIS] = 0), since global configuration over-rides endpoint configuration.

To configure RNDIS DMA mode use, configure the field that correspond to the USB module endpoint using the corresponding USB0/1 TX(RX) Mode Register (TXMODE0/1[TXn_MODE] = 01b and RXMODE0/1[RXn_MODE] = 01b).

24.4.8.3 Generic RNDIS DMA Transfer

Generic RNDIS DMA transfer mode is identical to the normal RNDIS mode in nearly all respects, except for the exception case where the last packet of the transfer can either be a short packet or the MaxPktSize. When the last packet size is equal to the MaxPktSize then no additional zero-byte packet is sent when using Generic RNDIS transfer. Generic RNDIS transfer makes use of a USB0/1 GENERIC RNDIS EPn Size register (there exists a register for each endpoint) that must be programmed with the transfer size (in bytes) of the transfer for the USB Module (USB0 or USB1) prior to posting a transfer transaction. If the transfer size is an integer multiple of USB MaxPktSize then no additional zero-byte packet is sent when using Generic RNDIS transfer. However, if the a short packet has been sent prior to programmed size count, the transfer would end in a similar fashion an RDIS transfer would behave. For example, it the USB MaxPktSize (Tx/RxMaxP) is programmed with a value of 64, the Generic RNDIS EP Size register for that endpoint must be programmed with a value that is an integer multiple of 64 (for example, 64, 128, 192, 256, etc.) for it behave differently than RNDIS transfer. In other words, when using Generic RNDIS mode and the DMA is tasked to transfer data transfer size that is less or equal the size value programmed within the USB0/1 GENERIC RNDIS EPn Size register.

This means that Generic RNDIS mode will perform data transfer in the same manner as RNDIS mode, closing the CPPI packet if a USB packet with a size less than the USB MaxPktSize size value is received. Otherwise, the packet will be closed when the value in the Generic RNDIS EP Size register is reached.

Using USB0/1 GENERIC RNDIS EPn Size register, a packet of up to 64K bytes (65536 bytes) can be transferred. This is to allow the host software to program the USB module to transfer data that is an exact multiple of the USB MaxPktSize (Tx/RxMaxP programmed value) without having to send an additional short packet to terminate.

NOTE: As in RNDIS mode, the USB max packet size (Tx/RxMaxp programmed value) of any Generic RNDIS mode enabled endpoints must be a multiple of 64 bytes. Generic RNDIS acceleration should not be enabled for endpoints where the max packet size is not a multiple of 64 bytes. Only transparent mode should be used for such endpoints.

24.4.8.3.1 Generic RNDIS DMA Transfer Setup

Disable the use of RNDIS at the global level, this is achieved by clearing RNDIS bit field (CTRLR0/1[RNDIS]=0), since global configuration over-rides endpoint configuration.

Configure the field that correspond to the USB module endpoint using the corresponding USB0/1 TX(RX) Mode Register (TXMODE0/1[TXn_MODE] = 11b and RXMODE0/1[RXn_MODE] = 11b).

24.4.8.4 Linux CDC DMA Transfer

Linux CDC DMA transfer mode acts in the same manner as RNDIS packets, except for the case where the last data matches the max USB packet size requiring additional zero-byte packet transfer in RNDIS mode and this is not the case for Linux CDC. If the last data packet of a transfer is a short packet where the data size is greater than zero and less the USB MaxPktSize, then the behavior of the Linux CDC DMA transfer type is identical with the RNDIS DMA transfer type. The only exception is when the short packet length terminating the transfer is a Null Packet. In this case, instead of transferring the Null Packet, it will transfer a data packet of size 1 byte with the data value of 00h.

In transmit operation, if an endpoint is configured or CDC Linux mode, upon receiving a Null Packet from the CPPI DMA, the XFER DMA will then generate a packet containing 1 byte of data, whose value is 00h, indicating the end of the transfer. During receive operation, the XFER DMA will recognize the one byte zero packet as a termination of the data transfer, and sends a block of data with the EOP indicator set and a byte count of one to the CPPI DMA controller. The CPPI DMA realizing the end of the transfer termination will not update/increase the packet size count of the Host Packet Descriptor.

24.4.8.4.1 Linux CDC DMA Transfer Setup

Disable the use of RNDIS at the global level, this is achieved by clearing RNDIS bit field (CTRLR0/1[RNDIS]=0), since global configuration over-rides endpoint configuration.

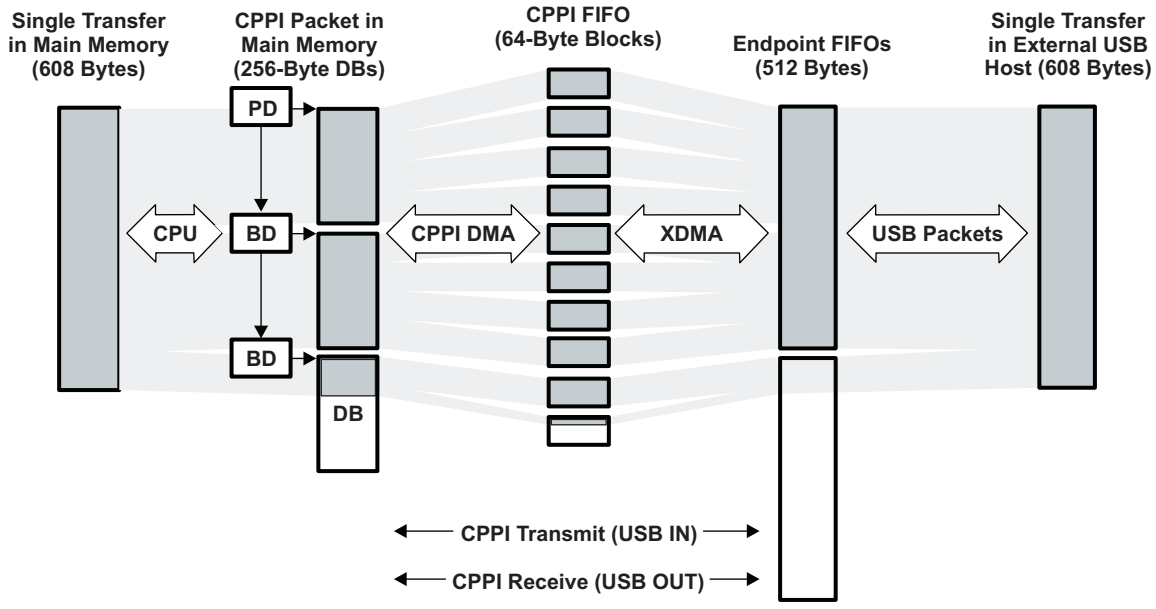
Configure the field that correspond to the USB module endpoint using the corresponding USB0/1 TX(RX) Mode Register (TXMODE0/1[TXn_MODE] = 10b and RXMODE0/1[RXn_MODE] = 10b).

24.4.9 USB Data Flow Using DMA

The necessary steps required to perform a USB data transfer using the CPPI 4.1 DMA is expressed using an example for both transmit and receive cases. Assume USB0 is ready to perform a USB data transfer of size 608 bytes (see Figure 24-16).

The 608 bytes data to be transferred is described using three descriptors. Since each data buffer defined within a single descriptor has a size of 256 bytes, a 608 bytes data buffer would require three descriptors.

Figure 24-16. High-level Transmit and Receive Data Transfer Example



Example assumptions:

- The CPPI data buffers are 256 bytes in length.
- USB0 module is to be used to perform this transfer. Note that the steps required is similar for USB1 use.
- The USB0 endpoint 1 Tx and Rx endpoint 1 size are programmed to handle max USB packet size of 512 bytes.
- A single transfer length is 608 bytes.
- The SOP offset is 0.

The previous example translates to the following multi-descriptor setup:

Transmit Case

Transmit setup is as follows:

- One packet descriptor with packet length (this is NOT data buffer length, the term packet used here is to mean a transfer length not USB packet) field of 608 bytes and a Data Buffer of size 256 Bytes linked to the 1st host buffer descriptor.
- Two buffer descriptors with first buffer descriptor (this is the one linked to the packet descriptor) defining the second data buffer size of 256 Bytes which in turn is linked to the next (second) buffer descriptor.
- Second buffer fescriptor with a fata buffer size of 96 bytes (can be greater, the packet descriptor contain the size of the packet) linked to no other descriptor (NULL).

Receive Case

For this example since each data buffer size is 256 bytes, we would require a minimum of three descriptors that would define data buffer size of 608 bytes. The receive setup is as follows:

- Two buffer descriptors with 256 bytes of data buffer size
- One buffer descriptor with 96 bytes (can be greater) of data buffer size

Within the rest of this section, the following nomenclature is used:

BD — Buffer Descriptor or Host Buffer Descriptor

DB — Data Buffer Size of 256 Bytes

PBD — Pointer to Host Buffer Descriptor

PD — Host Packet Descriptor

PPD — Pointer to Host Packet Descriptor

TXSQ — Transmit Queue or Transmit Submit Queue (for USB0 EP1, use Queues 32 or 33, for USB1 EP1 use Queues 62 or 63)

TXCQ — Transmit Completion Queue or Transmit Return Queue (for USB0 Tx EP1, use Queue 93, and for USB1 Tx EP1 use Queue 125)

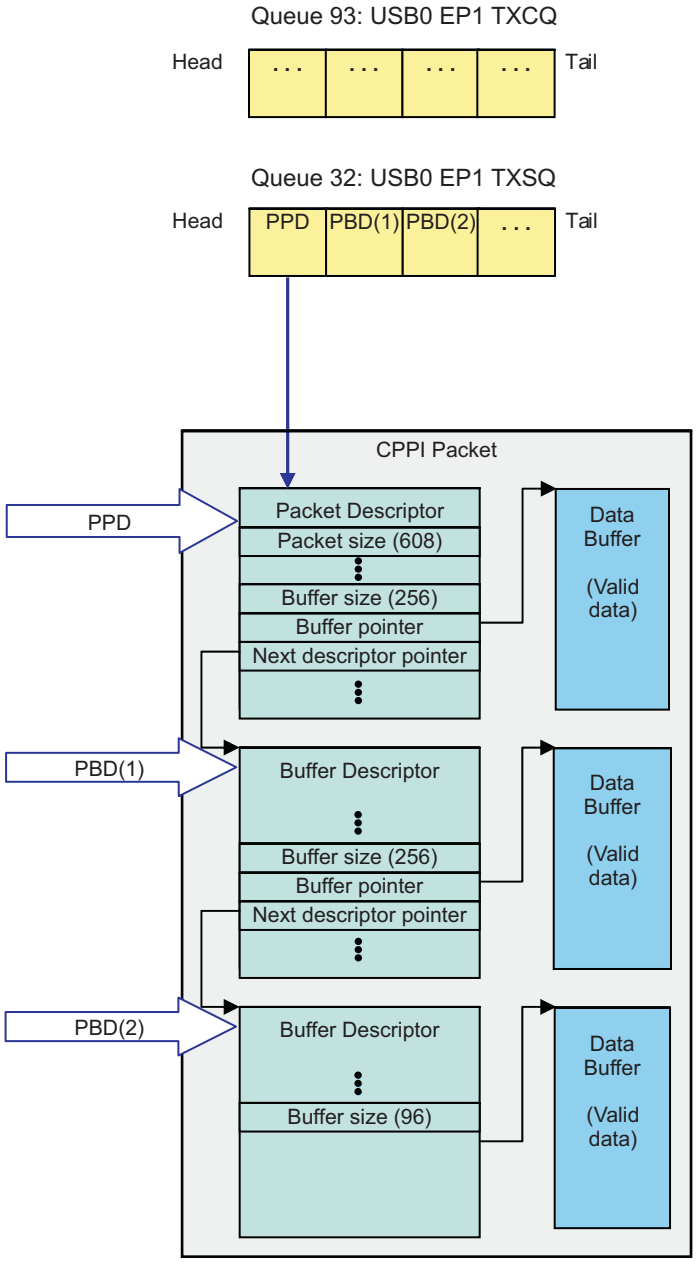
RXCQ — Receive Completion Queue or Receive Return Queue (for USB0 Rx EP1, use Queue 109, for USB1 Rx EP1 use Queue 141)

RXSQ — Receive Free/Buffer Descriptor Queue or Receive Submit Queue. (For USB0 Rx EP1 Queue 0 is used and for USB1 Rx EP1 Queue 16 should be used

24.4.9.1 Transmit USB Data Flow Using DMA

The transmit descriptors and queue status configuration prior to the transfer taking place is shown in Figure 24-17.

Figure 24-17. Transmit Descriptors and Queue Status Configuration



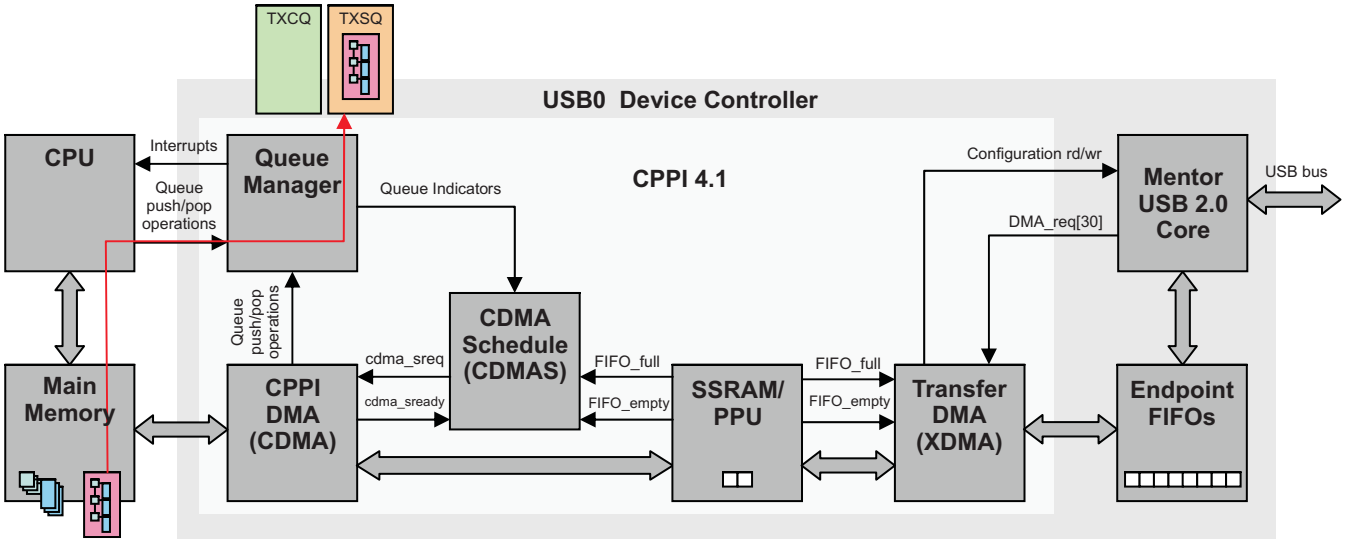
24.4.9.1.1 Transmit Initialization (Step 1)

The CPU performs the following steps for transmit initialization:

1. Initializes Memory Region 0 base address and Memory Region 0 size, Link RAM0 Base address, Link RAM0 data size, and Link RAM1 Base address.
2. Creates PD, BDs, and DBs in main memory and link as indicated in Figure 24-18.
3. Initializes and configures the Queue Manager, Channel Setup, DMA Scheduler, and Mentor USB 2.0 Core.
4. Adds (pushes) the PPD and the two PBDs to the TXSQ by writing the Packet Descriptor address to the TXSQ CTRL D Register.

Figure 24-18 captures the BD/DB pair in main memory and later submitted within the TXSQ.

Figure 24-18. Transmit USB Data Flow Example (Initialization)



24.4.9.1.2 CDMA and XDMA Transfer Packet Data Into Endpoint FIFO (Step 2)

The steps for CDMA and XDMA to transfer packet data into endpoint FIFO are as follows:

1. The Queue Manager informs the CDMAS that the TXSQ is not empty.
2. CDMAS checks that the CPPI FIFO FIFO_full is not asserted, then issues a credit to the CDMA.
3. CDMA reads the Packet Descriptor pointer (PPD) and descriptor size hint from the queue manager
4. For each 64-byte block of data in the packet data payload (note that packet refers here to CPPI packet which is not the same as USB packet and it means to refer to data transfer size):
   (a) The CDMA transfers a max burst of 64-byte block (OCP burst) from the data to be transferred in main memory to the CPPI FIFO
   (b) The XDMA sees FIFO_empty not asserted and transfers 64-byte block from CPPI FIFO to Endpoint FIFO.
   (c) The CDMA performs the above 2 steps (‘a’ and ‘b’) 3 more times since the data size of the HPD is 256 bytes.
5. The CDMA reads the first buffer descriptor pointer (PBD).
6. For each 64-byte block of data in the packet data payload:
   (a) The CDMA transfers a max burst of 64-byte block from the data to be transferred in main memory to the CPPI FIFO.
   (b) The XDMA sees FIFO_empty not asserted and transfers 64-byte block from CPPI FIFO to Endpoint FIFO.
   (c) The CDMA performs the above 2 steps (‘a’ and ‘b’) 2 more times since data size of the HBD is 256 bytes.
7. The CDMA reads the second Buffer Descriptor pointer (PBD
8. For each 64-byte block of data in the packet data payload:
   (a) The CDMA transfers a max burst of 64-byte block from the data to be transferred in main memory to the CPPI FIFO.
   (b) The XDMA sees FIFO_empty not asserted and transfers 64-byte block from CPPI FIFO to Endpoint FIFO.
   (c) The CDMA transfers the last remaining 32-byte from the data to be transferred in main memory to the CPPI FIFO.
   (d) The XDMA sees FIFO_empty not asserted and transfers 32-byte block from CPPI FIFO to Endpoint FIFO.

24.4.9.1.3 USB 2.0 Core Transmits USB Packets for Tx (Step 3)

1. Once the XDMA has transferred enough 64-byte blocks of burst of data from the CPPI FIFO to fill the Endpoint FIFO (accumulated 512 bytes), it signals the Mentor USB 2.0 Core that a TX packet is ready (sets the endpoint's TxPktRdy bit).
2. The USB 2.0 Core will transmit the packet from the Endpoint FIFO out on the USB BUS when it receives a corresponding IN request from the attached USB Host.
3. After the USB packet is transferred, the USB 2.0 Core issues a TX DMA_req to the XDMA.
4. This process is repeated until (the above steps '1', '2', and '3') the entire packet has been transmitted. The XDMA will also generate the required termination packet depending on the termination mode configured for the endpoint.

24.4.9.1.4 Return Packet to Completion Queue and Interrupt CPU for Tx (Step 4)

1. After all data for the packet has been transmitted (as specified by the packet size field), the CDMA will write the pointer of the Packet Descriptor only to the TX Completion Queue specified in the return queue manager / queue number fields of the packet descriptor.
2. The Queue Manager then indicates the status of the TXSQ (empty) to the CDMAS and the TXCQ to the CPU via an interrupt.

24.4.9.2 Receive USB Data Flow Using DMA

The receive buffer descriptors and queue status configuration prior to the transfer taking place is shown in Figure 24-19.

24.4.9.2.1 Receive Initialization (Step 1)

1. The CPU initializes Queue Manager with the Memory Region 0 base address and Memory Region 0 size, Link RAM0 Base address, Link RAM0 data size, and Link RAM1 Base address.
2. The CPU creates BDs, and DBs in main memory and link them creating a BD and DB pairs.
3. CPU then initializes the RXCQ queue and configures the Queue Manager, Channel Setup, DMA Scheduler, and USB 2.0 Core.
4. It then adds (pushes) the address of the addresses/pointers of three Buffer Descriptors (PBDs) into the RXSQ by writing onto CTRL D register.

Figure 24-20 displays receive operation Initialization.

24.4.9.2.2 USB 2.0 Core Receives a Packet, XDMA Starts Data Transfer for Receive (Step 2)

1. The USB 2.0 Core receives a USB packet from the USB Host and stores it in the Endpoint FIFO.
2. It then asserts a DMA_req to the XDMA informing it that data is available in the Endpoint FIFO.
3. The XDMA verifies the corresponding CPPI FIFO is not full via the FIFO_full signal, then starts transferring 64-byte data blocks (burst) from the Endpoint FIFO into the CPPI FIFO.

Figure 24-19. Receive Buffer Descriptors and Queue Status Configuration

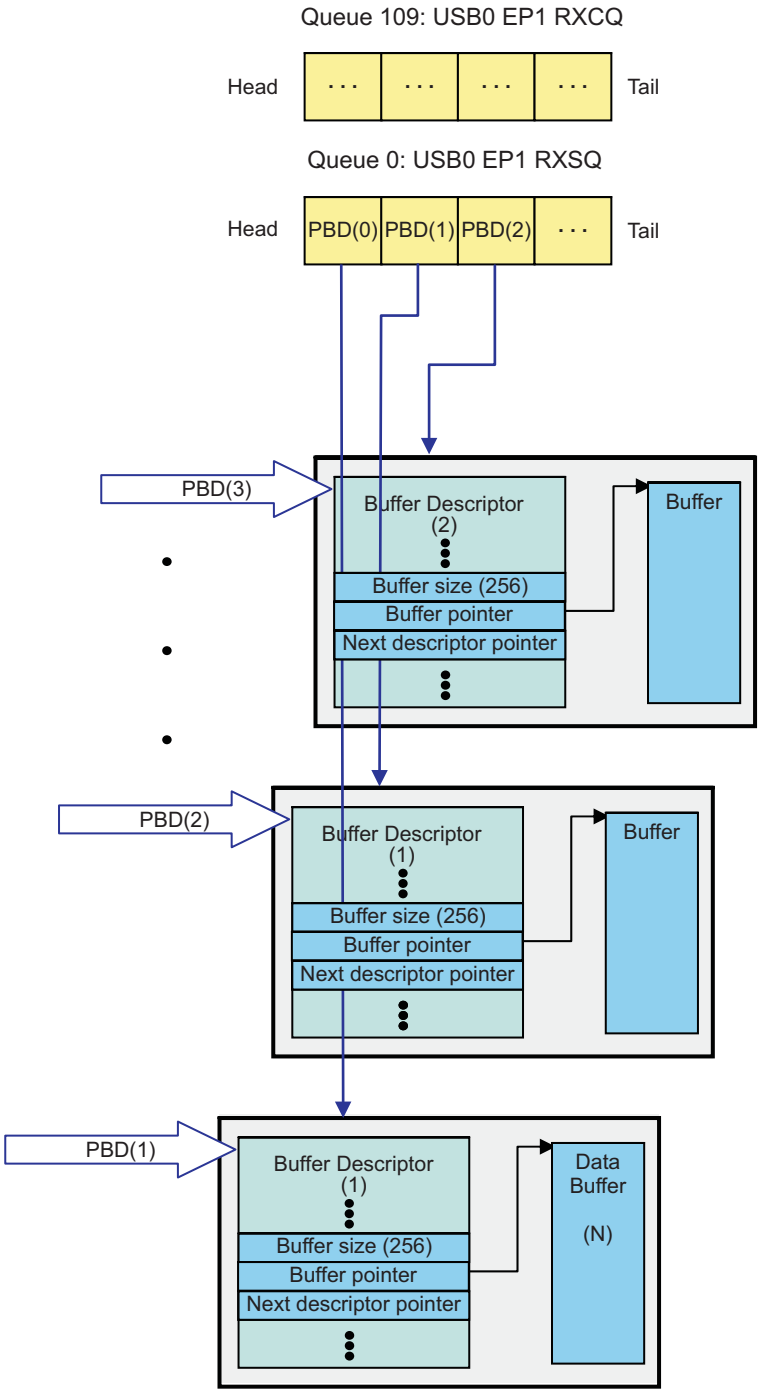
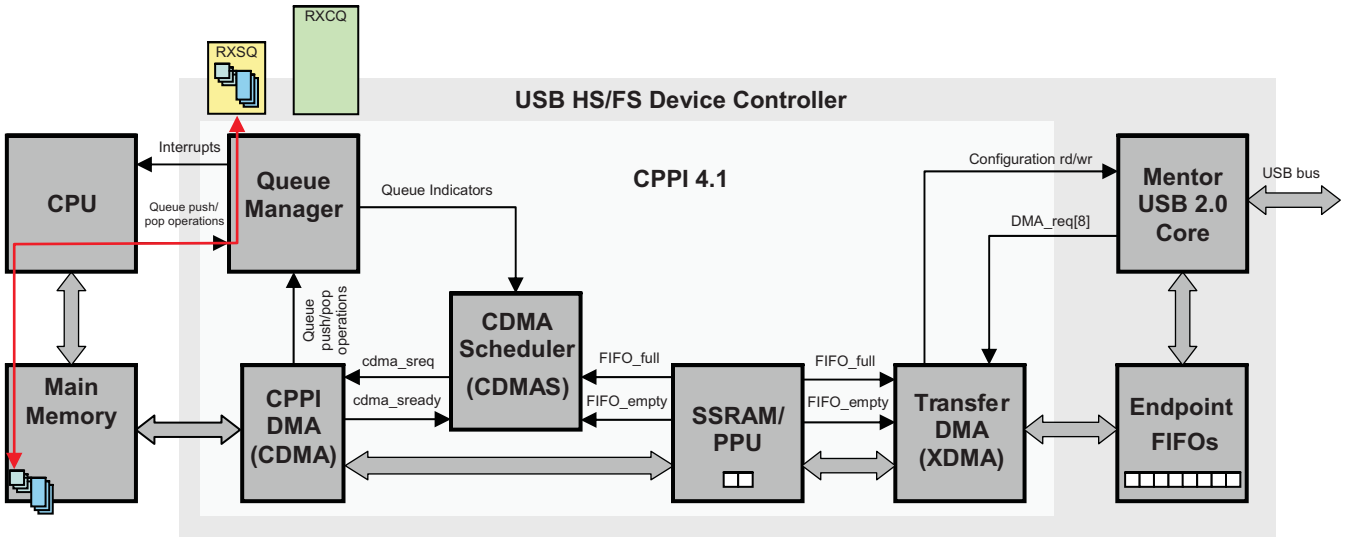


Figure 24-20. Receive USB Data Flow Example (Initialization)



24.4.9.2.3 CDMA Transfers Data from SSRAM / PPU to Main Memory for Receive (Step 3)

1. The CDMAS see FIFO_empty de-asserted (there is RX data in the FIFO) and issues a transaction credit to the CDMA.
2. The CDMA begins packet reception by fetching the first PBD from the Queue Manager using the Free Descriptor / Buffer Queue 0 (Rx Submit Queue) index that was initialized in the RX port DMA state for that channel.
3. The CDMA will then begin writing the 64-byte block of packet data into this DB.
4. The CDMA will continue filling the buffer with additional 64-byte blocks of data from the CPPI FIFO and will fetch additional PBD as needed using the Free Descriptor / Buffer Queue 1, 2, and 3 indexes for the 2nd, 3rd, and remaining buffers in the packet. After each buffer is filled, the CDMA writes the buffer descriptor to main memory.

24.4.9.2.4 CDMA Completes the Packet Transfer for Receive (Step 4)

1. After the entire packet has been received, the CDMA writes the packet descriptor to main memory.
2. The CDMA then writes the packet descriptor to the RXCQ specified in the Queue Manager / Queue Number fields in the RX Global Configuration Register.
3. The Queue Manager then indicates the status of the RXCQ to the CPU via an interrupt.
4. The CPU can then process the received packet by popping the received packet information from the RXCQ and accessing the packet's data from main memory.

24.5 USB 2.0 Test Modes

The USB2.0 controller supports the four USB 2.0 test modes (Test_SE0_NAK, Test_J, Test_K, and Test_Packet) defined for high-speed functions. It also supports an additional “FIFO access” test mode that can be used to test the operation of the CPU interface, the DMA controller and the RAM block.

The test modes are entered by writing to the TESTMODE register. A test mode is usually requested by the host sending a SET_FEATURE request to Endpoint 0. When the software receives the request, it should wait until the Endpoint 0 transfer has completed (when it receives the Endpoint 0 interrupt indicating the status phase has completed) then write to the TESTMODE register.

Note: These test modes have no purpose in normal operation.

24.5.1 TEST_SE0_NAK

To enter the Test_SE0_NAK test mode, the software should set the TEST_SE0_NAK bit in the TESTMODE register to 1. The USB controller will then go into a mode in which it responds to any valid IN token with a NAK.

24.5.2 TEST_J

To enter the Test_J test mode, the software should set the TEST_J bit in the TESTMODE register to 1. The USB controller will then go into a mode in which it transmits a continuous J on the bus.

24.5.3 TEST_K

To enter the Test_K test mode, the software should set the TEST_K bit in the TESTMODE register to 1. The USB controller will then go into a mode in which it transmits a continuous K on the bus.

24.5.4 TEST_PACKET

To execute the Test_Packet, the software should:

1. Start a session (if the core is being used in Host mode.
2. Write the standard test packet (shown below) to the Endpoint 0 FIFO.
3. Write 8h to the TESTMODE register (TEST_PACKET = 1) to enter Test_Packet test mode.
4. Set the TxPktRdy bit in the HOST/PERI_CSR0 register (bit 1).

The 53 byte test packet to load is as follows (all bytes in hex). The test packet only has to be loaded once; the USB controller will keep re-sending the test packet without any further intervention from the software.

Table 24-25 displays the 53 Bytes test packet content.

Table 24-25. 53 Bytes Test Packet Content

0	0	0	0	0	0	0	0
0	AA	AA	AA	AA	AA	AA	AA
AA	EE	EE	EE	EE	EE	EE	EE
EE	FE	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	7F	BF	DF
EF	F7	FB	FD	FC	7E	BF	DF
EF	F7	FB	FD	7E			

This data sequence is defined in Universal Serial Bus Specification Revision 2.0, Section 7.1.20. The USB controller will add the DATAA0 PID to the head of the data sequence and the CRC to the end.

24.5.5 FIFO_ACCESS

The FIFO Access test mode allows you to test the operation of CPU Interface, the DMA controller (if configured), and the RAM block by loading a packet of up to 64 bytes into the Endpoint 0 FIFO and then reading it back out again. Endpoint 0 is used because it is a bidirectional endpoint that uses the same area of RAM for its Tx and Rx FIFOs.

NOTE: The core does not need to be connected to the USB bus to run this test. If it is connected, then no session should be in progress when the test is run.

The test procedure is as follows:

1. Load a packet of up to 64 bytes into the Endpoint 0 Tx FIFO.
2. Set HOST/PERI_CSR0[TXPKTRDY].
3. Write 40h to the TESTMODE register (FIFO_ACCESS = 1).
4. Unload the packet from the Endpoint Rx FIFO, again.
5. Set HOST/PERI_CSR0[SERVICEDRXPKTRDY].

Writing 40h to the TESTMODE register causes the following sequence of events:

The Endpoint 0 CPU pointer (that records the number of bytes to be transmitted) is copied to the Endpoint 0 USB pointer (that records the number of bytes received).

1. The Endpoint 0 CPU pointer is reset.
2. HOST/PERI_CSR0[TXPKTRDY] is cleared.
3. HOST/PERI_CSR0[RXPKTRDY] is set.
4. An Endpoint 0 interrupt is generated (if enabled).

The effect of these steps is to make the Endpoint 0 controller act as if the packet loaded into the Tx FIFO has flushed and the same packet received over the USB. The data that was loaded in the Tx FIFO can now be read out of the Rx FIFO.

24.5.6 FORCE_HOST

The Force Host test mode enables you to instruct the core to operate in Host mode, regardless of whether it is actually connected to any peripheral; that is, the state of the CID input and the LINESTATE and HOSTDISCON signals are ignored. (While in this mode, the state of the HOSTDISCON signal can be read from the BDEVICE bit in the device control register (DEVCTL)) .

This mode, which is selected by writing 80h to the TESTMODE register (FORCE_HOST = 1), allows implementation of the USB Test_Force_Enable (USB 2.0 Specification Section 7.1.20). It can also be used for debugging PHY problems in hardware.

While the FORCE_HOST bit remains set, the core enters the Host mode when the SESSION bit in DEVCTL is set to 1 and remains in the Host mode until the SESSION bit is cleared to 0 even if a connected device is disconnected during the session. The operating speed while in this mode is determined by the FORCE_HS and FORCE_FS bits in the TESTMODE register.

24.6 Reset Considerations

The USB controller has two reset sources: hardware reset and the soft reset.

24.6.1 Software Reset Considerations

When the RESET bit in the control register (CTRLR) is set, all the USB controller registers and DMA operations are reset. The bit is cleared automatically.

A software reset on the DSP or ARM CPU does not affect the register values and operation of the USB controller.

24.6.2 Hardware Reset Considerations

When a hardware reset is asserted, all the registers are set to their default values.

24.7 Interrupt Support

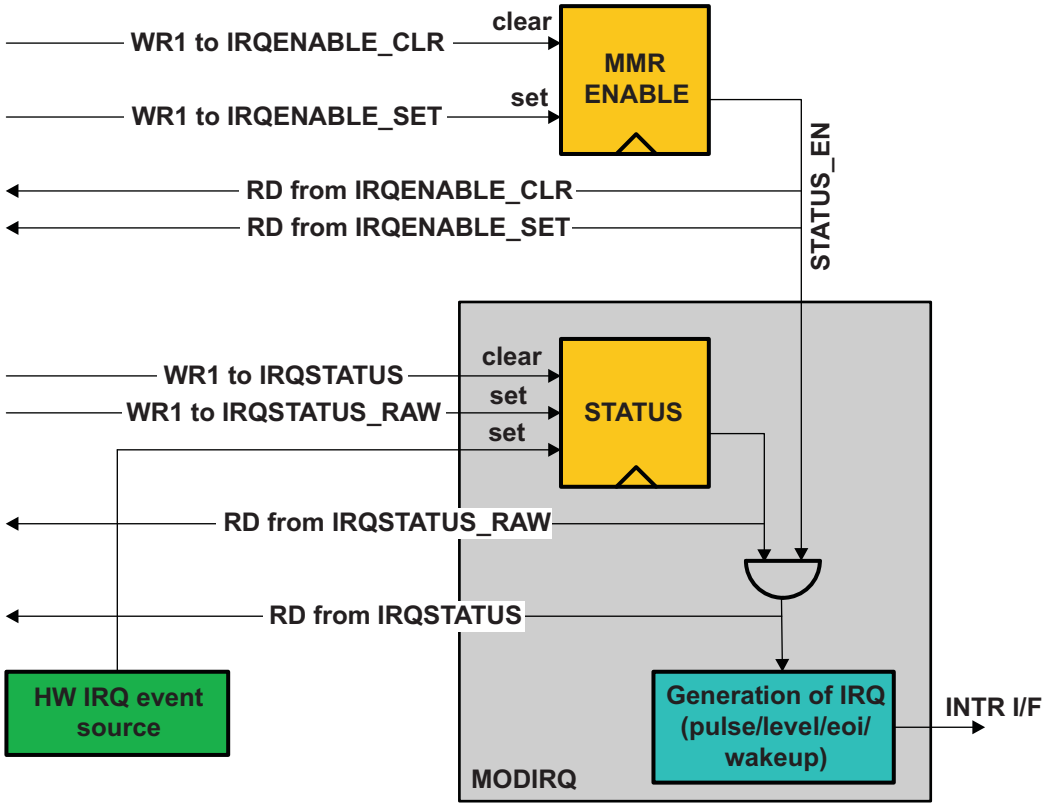
This section defines the module interrupt capabilities and requirements.

24.7.1 CPU Interrupts

There are three interrupt interfaces: one for the USB subsystem, one for USB0 controller, and one for USB1 controller. All interrupts for that specific unit are aggregated into one interrupt interface.

Each interrupt interface will use the IP generic MODIRQ as shown in Figure 24-21 which is shown with the grey box. Each interrupt has two basic registers: STATUS and ENABLE. The STATUS register is located in the MODIRQ module and can be set two ways. Hardware can set the register via internal logic. Software can set or clear the register by accessing the appropriate locations in the memory map. The enabling (or masking) of the interrupt is controlled via software. Writing 1's (WR1) to either the IRQENABLE_SET or IRQENABLE_CLR addresses will enable or disable the interrupt.

Figure 24-21. Functional Representation of Interrupts



The output interrupt interface has both a pulse and a level output. The pulse signal is active for one OCP clock; while the level output will be active high when asserted. An EOI function exists to re-enable the detection of active interrupts. If the system attempts to write to any of the interrupt registers, any still active interrupts will trigger the EOI output signal. The EOI input signal indicates when external logic requires the USB to re-evaluate its pending sources and send another pulse.

NOTE: When generating an interrupt by writing to one of the IRQ_ENABLE_SET registers; the interrupt can occur several cycles before the OCP write status complete has occurred.

Subsystem Interrupts

USB Subsystem Interrupts lists the interrupts generated by the USB subsystem.

USB Subsystem Interrupts

Interrupt	Description
rx_sop_starvation	Queue Manager cannot obtain a valid descriptor at SOP and has starvation status.
rx_mop_starvation	Queue Manager cannot obtain a valid descriptor during MOP and has starvation status.
pd_cmp_flag	When the packet is completed and the Packet Descriptor is pushed into the Queue Manager.
tx_pkt_cmp_0	USB0 Tx CPPI DMA Packet completion status
rx_pkt_cmp_0	USB0 Rx CPPI DMA Packet completion status
tx_pkt_cmp_1	USB1 Tx CPPI DMA Packet completion status
rx_pkt_cmp_1	USB1 Rx CPPI DMA Packet completion status

The pd_cmp_flag is only set when bit 31 of the original buffer length word (word 6) of a packet descriptor (not buffer descriptor) has been set.

The USBSS QMGR outputs 60 packet completion signals for the four Tx/Rx CPPI DMA Packet Completion Status interrupts as shown in USB Subsystem Interrupts. The mapping of the packet completion signals is shown in CPPI DMA Packet Completion Hardware Interrupt Groupings. Therefore, each packet completion consists of the combination of the data from the 15 individual packet completion signals. When one of the 15 packet completion signals was active (level high) then an interrupt would be generated (if enabled).

CPPI DMA Packet Completion Hardware Interrupt Groupings

Interrupt	CPPI DMA Packet Completion Grouping (EP)	Description
tx_pkt_cmp_0	1, 2, …, 15	USB0 Tx CPPI DMA Packet completion status
rx_pkt_cmp_0	1, 2, …, 15	USB0 Rx CPPI DMA Packet completion status
tx_pkt_cmp_1	1, 2, …, 15	USB1 Tx CPPI DMA Packet completion status
rx_pkt_cmp_1	1, 2, …, 15	USB1 Rx CPPI DMA Packet completion status

In order to control the frequency of the generated interrupts, interrupt pacing has been added to each of the 60 individual packet completions. Each packet completion has its own 8-bit unsigned threshold. Each threshold can range from 0 to 255. The packet descriptor for each of the packet completion signals will have a differed bit. This bit will indicate whether the packet completion will be incremented or not. If the count of the packet completion exceeds the threshold; then an interrupt will be generated. This assumes that the packet completion is enabled. The thresholds for the 60 packet completions are stored in one of the following registers: USBSS IRQ_DMA_THRESHOLD_ab_c. Where:

(a) Tx or Rx

(b) 0 or 1

(c) 0, 1, 2, or 3

The 60 DMA enables are stored in registers: USBSS IRQ_DMA_ENABLE_0 and USBSS IRQ_DMA_ENABLE_1.

The individual packet completion count registers will be reset to zero when one of the below conditions occurs:

- Reset signal is active.
- Packet completion threshold has been exceeded.
- Packet completion count is equal to 255.
- Frame count threshold has been exceeded.
- Frame count is equal to 255.

It is possible that the generation of interrupts has been reduced to an unacceptable delay due a lack of differed bits or thresholds set to too large of a number. A watch dog timer has been added to the design which provides a secondary method of generating the interrupts. This system uses the USB start of frame pulse from the Mentor core, a counter, a threshold, and an enable. This logic is similar as compared with the differed bit logic. The frame sync pulse frequency is 1kHz in full-speed or low-speed mode, or 8kHz is high-speed mode.

Example 1: The DMA threshold is set to 10 and the frame threshold is set to 3 for a specific endpoint. Let's assume that the expected packet completion frequency is 5 completed packets per frame. In this example the first frame had 5 packets completed. The second frame had 5 packets completed. An interrupt is generated after the 2nd frame.

Example 2: The DMA threshold is set to 10 and the frame threshold is set to 3 for a specific endpoint. Let's assume that the expected packet completion frequency is 5 completed packets per frame. In this example the first frame had 5 packets completed. The second frame had only 2 packets completed and the third frame had 0 packets completed. There was some error or delay that caused the remaining packets to not complete. An interrupt is generated after the 3rd frame.

In both examples software will not be able to determine whether the interrupt was generated due to DMA threshold or frame threshold.

Each of the 60 Packet completions will have a frame counter, frame threshold, and frame enable. If the frame counter exceeds the frame threshold; then an interrupt will be generated. This assumes the frame enable counter has been enabled.

The registers for the 60 frame thresholds are stored in: USBSS_IRQ_FRAME_THRESHOLD_ab_c. Where:

(a) Tx or Rx

(b) 0 or 1

(c) 0, 1, 2, or 3

The 60 frame enables are stored in registers: USBSS IRQ_FRAME_ENABLE_0 and USBSS IRQ_FRAME_ENABLE_1.

The individual frame count registers will be reset to zero when one of the below conditions occurs:

- Reset signal is active.
- Packet completion threshold has been exceeded.
- Packet completion count is equal to 255.
- Frame count threshold has been exceeded.
- Frame count is equal to 255.

When the low to high transition occurs for any of the four CPPI DMA Packet completions, a hardware signal is sent to the MODIRQ module. This will cause an interrupt (if enabled via software). When all packets have been removed from the queue, the pending signal is returned to a 0. Software may choose to use one of the unassigned queues if an interrupt is not wanted upon the completion of a packet.

The starvation interrupts occur when the queue manager cannot allocate a buffer for an Rx buffer. This can occur either at the start of a packet or in the middle of a series of packets.

The interrupts listed in USB Subsystem Interrupts can be enabled (or disabled) by setting (or clearing) the appropriate IRQENABLE_SET (or IRQENABLE_CLR) bits in the MMR registers.

To clear the interrupts it is required to write 1's to the IRQSTAUS registers. It is possible to manually set the interrupts by writing 1's to the IRQSTATUS_RAW registers.

Controller Interrupts

Each of the controllers has 42 interrupts that are generated by the USB 2.0 OTG Controller. Actually, there are only 41 real interrupts; but 42 are designed into the logic. This makes the logic easier to build. RX_ENDP[0] will never be used.

Controller Interrupts

Interrupt	Description
TX_ENDP[15:0]	TX endpoint ready or error for endpoints 15 to 0 (endpoint 0 is used for both TX and RX)
RX_ENDP[15:1]	RX endpoint ready or error for endpoints 15 to 0 (there is no interrupt for endpoint 0)
USB[8] DRVVBUS level change	Signal from the Mentor core
USB[7:0]	8 USB conditions
USB_INT	Single interrupt signal from the Mentor core
TX_FIFO[15:0]	TX FIFO endpoint ready for endpoints 15 to 0

The interrupts listed in Controller Interrupts are generated by the Mentor core (assumes that DMAReqEnab and DMAReqMode are not both set) and follow the following procedure to generate the interrupt (for all interrupts except USB[8] and TX_FIFO[15:0]):

- Mentor core will signal F_INTR module that there is an interrupt via the USB_INT signal.
- F_INTR will generate 3 VBUSP read transactions for the Mentor core.
  - Read INTRTX register
  - Read INTRRX register
  - Read INTRUSB register
- F_INTR will temporary store the results and transfer the data to the F_REGS module.
- The F_REGS module will generate the hardware IRQ event source.

Interrupts for TX_ENDP[15:0] and RX_ENDP[15:1] are not generated when both DMAReqEnab and DMAReqMode are both set. This is the typical case when the USBSS DMA is used (CPPI).

Interrupts for TX_FIFO[15:0] are derived from the Mentor controller and sent directly to the F_REGS module which generates the hardware interrupts event source. These interrupts indicate when the TX Fifo is ready to accept new data.

The interrupt for USB[8] is from the signal DRVVBUS. This interrupt comes from the Mentor core; but does not have an internal register within the Mentor core. The signal DRVVBUS goes directly to the F_REGS module which generates the hardware interrupt event source.

The interrupts listed in Controller Interrupts can be enabled (or disabled) by setting (or clearing) the appropriate IRQENABLE_SET (or IRQENABLE_CLR) bits in the MMR registers.

To clear the interrupts it is required to write 1's to the IRQSTAUS registers. It is possible to manually set the interrupts by writing 1's to the IRQSTATUS_RAW registers.

The interrupt registers for all the interrupts in Table 132 are listed in both the USB0 and USB1 controller memory maps. The interrupt registers for TX_ENDP[15:0], RX_ENDP[15:1], and USB[7:0] are listed in the Mentor Core memory map.

Interrupt Description

USB Core Highlander Interrupts

There are 15 general-purpose TX endpoints supported in addition to control endpoint 0, whose interrupt encapsulates both TX and RX readiness. The core will generate a corresponding EP0 or TXEPn (n=1…15) interrupt when that TX endpoint successfully completes transmitting a packet and the buffer is empty and ready to send another packet or when the TX endpoint had an error condition. If the TXCSR is setup as required for the DMA (with DMAReqEnab and DMAReqMode set) then the TX endpoint interrupt will only generate error condition interrupts, and not packet completion interrupts, which would allow software to keep the interrupt enabled for monitoring error conditions. This is the setup when the USBSS CPPI DMA controller is used.

There are 15 general-purpose RX endpoints supported. The core will generate a corresponding RXEPn (n=1…15) interrupt when that RX endpoint successfully receives a packet and all the data is ready in the RX endpoint FIFO or when a receive error occurs. If the RXCSR is setup as required for the DMA (with DMAReqEnab set and DMAReqMode clear) then the RX endpoint interrupt will only generate error condition interrupts, and not packet readiness interrupts, which would allow software to keep the interrupt enabled for monitoring error conditions.

Interrupts USB[0] to USB[7] are generated by the Mentor controller, and converted to HIGHLANDER 0.8 format by the Highlander interrupt sub-module. Refer to the Mentor core documentation for more details on these interrupts. Interrupt USB[8] is generated outside the Mentor core by the interrupt sub-module whenever the level of DRVVBUS changes. This is usually due to entering or leaving host mode, or entering or leaving power saving mode.

It is important to clear the Mentor controller interrupts by reading the respective Mentor controller INTRUSB Register before clearing the IRQ_ENABLE_CLR Register and setting the IRQ_EOI Register.

16 TX FIFO endpoint interrupts exist one for each endpoint. These interrupts indicate when the TX FIFO is ready to accept new data.

The USB20OTG interrupts are also aggregated by the MODIRQ module. Refer to the memory mapped registers for a list of USB Interrupts that the MODIRQ processes. Only one USB interrupt interface is provided on the outside for each USB controller and MODIRQ provides the interrupt aggregation logic for these interrupts. When the EOI register in the USB module is written to, the eoi_write signal is sent to MODIRQ which in turn re-evaluates the interrupt and triggers it if necessary.

USB Core (Non-Highlander) Interrupts

If the non-Highlander interrupt mode is selected for the USB interrupts, then the previous list of USB Highlander sources are not valid, and the core generates the CPU interrupt directly. The signal from the core will be connected to the module F_REGS and all interactions to clear and mask interrupts must be done with the core registers.

The additional USB interrupt for DRVVBUS level changes is not part of the core interrupt set, and therefore is not available when using non-Highlander interrupt mode.

Interrupt Condition Control

USB Core Interrupts

The USB core interrupts are originally generated inside the Mentor core. The core provides enable registers for each of the three types of interrupts. The INTRTXE register allows enabling of each TX endpoint interrupt The INTRRXE register allows enabling of each RX endpoint interrupt. The INTRUSBE register allows enabling of each general USB interrupt. Any enabled interrupt will generate CPU interrupts in the form of the corresponding interrupt source, the module generated interrupt pulse, and the interrupt pulse from the Mentor core directly.

To comply with Highlander interrupts, additional logic is built around the core. This adds a mask register along with mask set and mask clear registers. The USB Interrupt Mask register shows the current mask value, where each bit enables the corresponding interrupt source in the USB Interrupt Source register. The USB Interrupt Mask Set register allows writing a '1' in bit positions to enable the corresponding interrupt source. Those bits written with a '0' will not be modified. The USB Interrupt Mask Clear register allows writing a '1' in bit positions to disable the corresponding interrupt source. Those bit written with a '0' will not be modified. The mask is used to enable interrupt sources and generate the masked interrupt sources which are used to generate the CPU interrupt or by external logic to generate CPU interrupts.

24.8 Supported Use Cases

The USB Subsystem supports two independent USB Modules where each USB module can operate as a host or peripheral. When operating as a host, it is capable of interfacing to a single target/peripheral directly or to multiple targets via hub at low-, full-, or high-speed. When operating as a peripheral, it is capable of interfacing to a host at a full- or high-speed.

24.9 USB Registers

The USB20OTG_REGS submodule contains the MMRs that are used in the submodules within the USB controller. The registers required to access and control the USBSS are partitioned in blocks based on their functions. Table 24-26 lists the base address offset of the USBSS and its submodules. For the base address of these registers, see Table 1-11.

Table 24-26. USBSS Submodule Base Addresses and Size

Submodule Name	Base Address Offset	Section
USBSS Registers	0000h	Section 24.9.1
USB0 Controller Registers	1000h	Section 24.9.2.1
USB1 Controller Registers	1800h	Section 24.9.2.2
CPPI DMA Controller Registers	2000h	Section 24.9.3
CPPI DMA Scheduler Registers	3000h	Section 24.9.4
CPPI DMA Queue Manager Registers	4000h	Section 24.9.5

24.9.1 USBSS Registers

The USBSS registers contain the registers that are used to control at the global level and apples to all submodules. Table 24-27 lists the registers for the USBSS submodule.

Table 24-27. USBSS Registers

Address Offset	Acronym	USBSS Register
0h	REVREG	USBSS REVISION
10h	SYSCONFIG	USBSS SYSCONFIG
20h	EOI	USBSS IRQ_EOI
24h	IRQSTATRAW	USBSS IRQ_STATUS_RAW
28h	IRQSTAT	USBSS IRQ_STATUS
2Ch	IRQENABLER	USBSS IRQ_ENABLE_SET
30h	IRQCLEARR	USBSS IRQ_ENABLE_CLR
100h	IRQDMATHOLDTX00	USBSS IRQ_DMA_THRESHOLD_TX0_0
104h	IRQDMATHOLDTX01	USBSS IRQ_DMA_THRESHOLD_TX0_1
108h	IRQDMATHOLDTX02	USBSS IRQ_DMA_THRESHOLD_TX0_2
10Ch	IRQDMATHOLDTX03	USBSS IRQ_DMA_THRESHOLD_TX0_3
110h	IRQDMATHOLDRX00	USBSS IRQ_DMA_THRESHOLD_RX0_0
114h	IRQDMATHOLDRX01	USBSS IRQ_DMA_THRESHOLD_RX0_1
118h	IRQDMATHOLDRX02	USBSS IRQ_DMA_THRESHOLD_RX0_2
11Ch	IRQDMATHOLDRX03	USBSS IRQ_DMA_THRESHOLD_RX0_3
120h	IRQDMATHOLDTX10	USBSS IRQ_DMA_THRESHOLD_TX1_0
124h	IRQDMATHOLDTX11	USBSS IRQ_DMA_THRESHOLD_TX1_1
128h	IRQDMATHOLDTX12	USBSS IRQ_DMA_THRESHOLD_TX1_2
12Ch	IRQDMATHOLDTX13	USBSS IRQ_DMA_THRESHOLD_TX1_3
130h	IRQDMATHOLDRX10	USBSS IRQ_DMA_THRESHOLD_RX1_0
134h	IRQDMATHOLDRX11	USBSS IRQ_DMA_THRESHOLD_RX1_1
138h	IRQDMATHOLDRX12	USBSS IRQ_DMA_THRESHOLD_RX1_2
13Ch	IRQDMATHOLDRX13	USBSS IRQ_DMA_THRESHOLD_RX1_3
140h	IRQDMAENABLE0	USBSS IRQ_DMA_ENABLE_0
144h	IRQDMAENABLE1	USBSS IRQ_DMA_ENABLE_1
200h	IRQFRAMETHOLDTX00	USBSS IRQ_FRAME_THRESHOLD_TX0_0
204h	IRQFRAMETHOLDTX01	USBSS IRQ_FRAME_THRESHOLD_TX0_1
208h	IRQFRAMETHOLDTX02	USBSS IRQ_FRAME_THRESHOLD_TX0_2
20Ch	IRQFRAMETHOLDTX03	USBSS IRQ_FRAME_THRESHOLD_TX0_3
210h	IRQFRAMETHOLDRX00	USBSS IRQ_FRAME_THRESHOLD_RX0_0
214h	IRQFRAMETHOLDRX01	USBSS IRQ_FRAME_THRESHOLD_RX0_1
218h	IRQFRAMETHOLDRX02	USBSS IRQ_FRAME_THRESHOLD_RX0_2
21Ch	IRQFRAMETHOLDRX03	USBSS IRQ_FRAME_THRESHOLD_RX0_3
220h	IRQFRAMETHOLDTX10	USBSS IRQ_FRAME_THRESHOLD_TX1_0
224h	IRQFRAMETHOLDTX11	USBSS IRQ_FRAME_THRESHOLD_TX1_1
228h	IRQFRAMETHOLDTX12	USBSS IRQ_FRAME_THRESHOLD_TX1_2
22Ch	IRQFRAMETHOLDTX13	USBSS IRQ_FRAME_THRESHOLD_TX1_3
230h	IRQFRAMETHOLDRX10	USBSS IRQ_FRAME_THRESHOLD_RX1_0
234h	IRQFRAMETHOLDRX11	USBSS IRQ_FRAME_THRESHOLD_RX1_1
238h	IRQFRAMETHOLDRX12	USBSS IRQ_FRAME_THRESHOLD_RX1_2
23Ch	IRQFRAMETHOLDRX13	USBSS IRQ_FRAME_THRESHOLD_RX1_3
240h	IRQFRAMEENABLE0	USBSS IRQ_FRAME_ENABLE_0
244h	IRQFRAMEENABLE1	USBSS IRQ_FRAME_ENABLE_1

24.9.1.1 USBSS Revision Register (REVREG)

The USBSS revision register (REVREG) contains the major and minor revisions for the USB subsystem module. This register is shown in Figure 24-22 and described in Table 24-28.

Figure 24-22. USBSS Revision Register (REVREG)

31–30	29–28	27–16
Scheme	Reserved	func
R-1	R-0	R-EA2h

15–11	10–8	7–6	5–0
R_rtl	X_major	Custom	Y_minor
R-1	R-0	R-0	R-7h

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-28. USBSS Revision Register (REVREG) Field Description

Bits	Field	Value	Description
31-30	Scheme	0-3h	Used to distinguish between old scheme and current.
29-28	Reserved	0	Reserved
27-16	func	0-FFFh	Function indicates a software compatible module family.
15-11	R_rtl	0-1Fh	RTL revision. Will vary depending on release.
10-8	X_major	0-7h	Major revision.
7-6	Custom	0-3h	Custom revision
5-0	Y_minor	0-3Fh	Minor revision.

24.9.1.2 USBSS SYSCONFIG Register (SYSCONFIG)

The USBSS SYSCONFIG register (SYSCONFIG) is the clock management configuration register for the USBSS.

The clock enable bits 8 to 11 should only be configured/modified during the firmware initialization stage of the USBSS. In other words, these bits should not be modified while the system is running or after enabling the clock. These bits control the clock generation modules. Dynamically setting these bits can disable the power management logic.

The USBSS SYSCONFIG register is shown in shown in Figure 24-23 and described in Table 24-29.

Figure 24-23. USBSS SYSCONFIG Register (SYSCONFIG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved				usb0 ocp en_n	phy0 utmi en_n	usb1 ocp en_n	phy1 utmi en_n
R-0				R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved		Standby mode		Idlemode		Freeemu	Soft reset
R-0		R/W-2h		R/W-2h		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-29. USBSS SYSCONFIG Register (SYSCONFIG) Field Descriptions

Bits	Field	Value	Description
31-12	Reserved	0	Reserved
11	usb0 ocp en_n		Active low clock enable for usb0_ocp_clk
		0	Enable
		1	Disable
10	phy0 utmi en_n		Active low clock enable for phy0_utmi_clk
		0	Enable
		1	Disable
9	usb1 ocp en_n		Active low clock enable for usb1_ocp_clk
		0	Enable
		1	Disable
8	phy1 utmi en_n		Active low clock enable for phy1_utmi_clk
		0	Enable
		1	Disable
7-6	Reserved	0	Reserved
5-4	Standby mode		Configuration of the local initiator state management mode.
		0	Force-standby mode
		1h	Reserved
		2h	Smart-standby mode
		3h	Smart-standby wakeup capable mode

Table 24-29. USBSS SYSCONFIG Register (SYSCONFIG) Field Descriptions (continued)

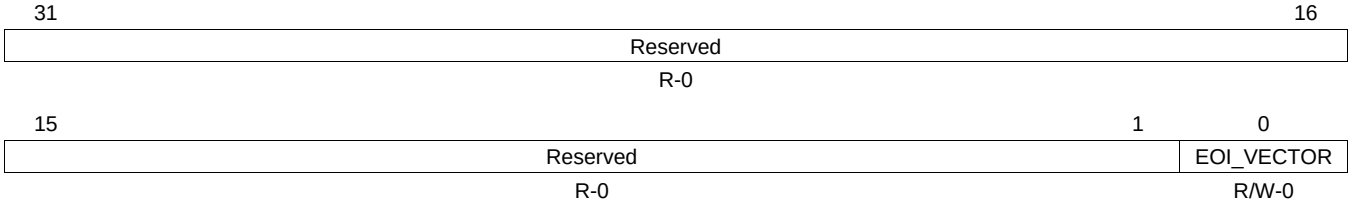
Bits	Field	Value	Description
3-2	Idlemode		Configuration of the local target state management mode.
		0	Reserved
		1h	Reserved
		2h	Smart-idle mode
		3h	Smart-idle wakeup capable mode
1	Freeemu		Sensitivity to emulation (debug) suspend input signal.
		0	Sensitive to emulation suspend
		1	NOT sensitive to emulation suspend
0	Soft reset		Software reset of USBSS, USB0, and USB1 modules
		Write 0	No action
		Write 1	Initiate software reset
		Read 0	Reset done, no action
		Read 1	Reset ongoing

24.9.1.3 USBSS End of Interrupt Register (EOI)

The USBSS end of interrupt register (EOI) allows the CPU to acknowledge completion of an interrupt by writing 0, zero, to the EOI_VECTOR field. An eoi_write signal will be generated and another interrupt will be triggered if interrupt sources remain.

This register will be reset one cycle after it has been written to. The USBSS end of interrupt register is shown in Figure 24-24 and described in Table 24-30.

Figure 24-24. USBSS End of Interrupt Register (EOI)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-30. USBSS End of Interrupt Register (EOI) Field Descriptions

Bits	Field	Description
31-1	Reserved	Always read as 0. Writes have no effect.
0	EOI_VECTOR	EOI for USBSS interrupt.

24.9.1.4 USBSS IRQ_STATUS_RAW (IRQSTATRAW)

The USBSS IRQSTATRAW register allows the USBSS interrupt sources to be manually triggered when writing a 1 to a specific bit. A read from this register returns the USBSS interrupt event pending value.

General actions per bit:
Write 0: No action
Write 1: Set event
Read 0: No event pending
Read 1: Event pending

This register is shown in Figure 24-25 and described in Table 24-31.

Figure 24-25. USBSS IRQ_STATUS_RAW (IRQSTATRAW)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved				rx_pkt_cmp_1	tx_pkt_cmp_1	rx_pkt_cmp_0	tx_pkt_cmp_0
R-0				R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved					pd_cmp_flag	rx_mop_starvation	rx_sop_starvation
R-0					R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-31. USBSS IRQ_STATUS_RAW (IRQSTATRAW) Field Descriptions

Bits	Field	Description
31-12	Reserved	Always read as 0. Writes have no effect.
11	rx_pkt_cmp_1	Interrupt status for USB1 Rx CPPI DMA packet completion status.
10	tx_pkt_cmp_1	Interrupt status for USB1 Tx CPPI DMA packet completion status.
9	rx_pkt_cmp_0	Interrupt status for USB0 Rx CPPI DMA packet completion status.
8	tx_pkt_cmp_0	Interrupt status for USB0 Tx CPPI DMA packet completion status.
7-3	Reserved	Always read as 0. Writes have no effect.
2	pd_cmp_flag	Interrupt status when the packet is completed and the packet descriptor is pushed into the queue manager.
1	rx_mop_starvation	Interrupt status when queue manager cannot allocate an Rx buffer in the middle of a packet.
0	rx_sop_starvation	Interrupt status when queue manager cannot allocate an Rx buffer at the start of a packet.

24.9.1.5 USBSS IRQ_STATUS Register (IRQSTAT)

The USBSS IRQSTAT register contains the interrupt sources status of the enabled interrupt. IRQSTAT register is a subset of IRQSTATRAW register. Writing a '1' to a bit field will clear a pending interrupt event.

General actions per bit:
Write 0: No action
Write 1: Clear event
Read 0: No event pending
Read 1: Event pending

This register is shown in Figure 24-26 and described in Table 24-32.

Figure 24-26. USBSS IRQ_STATUS (IRQSTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved				rx_pkt_cmp_1	tx_pkt_cmp_1	rx_pkt_cmp_0	tx_pkt_cmp_0
R-0				R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved					pd_cmp_flag	rx_mop_starvation	rx_sop_starvation
R-0					R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-32. USBSS IRQ_STATUS (IRQSTAT) Field Descriptions

Bits	Field	Description
31-12	Reserved	Always read as 0. Writes have no effect.
11	rx_pkt_cmp_1	Interrupt status for USB1 Rx CPPI DMA packet completion status.
10	tx_pkt_cmp_1	Interrupt status for USB1 Tx CPPI DMA packet completion status.
9	rx_pkt_cmp_0	Interrupt status for USB0 Rx CPPI DMA packet completion status.
8	tx_pkt_cmp_0	Interrupt status for USB0 Tx CPPI DMA packet completion status.
7-3	Reserved	Always read as 0. Writes have no effect.
2	pd_cmp_flag	Interrupt status when the packet is completed and the packet descriptor is pushed into the queue manager.
1	rx_mop_starvation	Interrupt status when queue manager cannot allocate an Rx buffer in the middle of a packet.
0	rx_sop_starvation	Interrupt status when queue manager cannot allocate an Rx buffer at the start of a packet.

24.9.1.6 USBSS IRQ_ENABLE_SET Register (IRQENABLER)

The USB interrupt mask set register (IRQENABLER) allows the USB interrupts to be enabled (unmasked). A read from this register returns the USB interrupts enabled.

General actions per bit:
Write 0: No action
Write 1: Interrupt enabled
Read 0: Interrupt disabled
Read 1: Interrupt enabled

The USBSS IRQ_ENABLE_SET register is shown in Figure 24-27 and described in Table 24-33.

Figure 24-27. USBSS IRQ_ENABLE_SET Register (IRQENABLER)

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		rx_pkt_cmp_1	tx_pkt_cmp_1	rx_pkt_cmp_0	tx_pkt_cmp_0
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	3	2	1	0
Reserved		pd_cmp_flag	rx_mop_starvation	rx_sop_starvation
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-33. USBSS IRQ_ENABLE_SET Register (IRQENABLER) Field Descriptions

Bits	Field	Description
31-12	Reserved	Always read as 0. Writes have no effect.
11	rx_pkt_cmp_1	Interrupt enable for USB1 Rx CPPI DMA packet completion status.
10	tx_pkt_cmp_1	Interrupt enable for USB1 Tx CPPI DMA packet completion status.
9	rx_pkt_cmp_0	Interrupt enable for USB0 Rx CPPI DMA packet completion status.
8	tx_pkt_cmp_0	Interrupt enable for USB0 Tx CPPI DMA packet completion status.
7-3	Reserved	Always read as 0. Writes have no effect.
2	pd_cmp_flag	Interrupt enable when the packet is completed and the packet descriptor is pushed into the queue manager.
1	rx_mop_starvation	Interrupt enable when queue manager cannot allocate an Rx buffer in the middle of a packet.
0	rx_sop_starvation	Interrupt enable when queue manager cannot allocate an Rx buffer at the start of a packet.

24.9.1.7 USBSS IRQ_ENABLE_CLR Register (IRQCLEARR)

The USBSS IRQCLEARR register allows the USBSS interrupt sources to be masked when writing a 1 to a specific bit. A read of this register returns the USBSS interrupt enabled value.

General actions per bit:
Write 0: No action
Write 1: Disable interrupt
Read 0: Interrupt disabled
Read 1: Interrupt enabled

The USBSS IRQ_ENABLE_CLR register is shown in Figure 24-28 and described in Table 24-34.

Figure 24-28. USBSS IRQ_ENABLE_CLR Register (IRQCLEARR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved				rx_pkt_cmp_1	tx_pkt_cmp_1	rx_pkt_cmp_0	tx_pkt_cmp_0
R-0				R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved					pd_cmp_flag	rx_mop_starvation	rx_sop_starvation
R-0					R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-34. USBSS IRQ_ENABLE_CLR Register (IRQCLEARR) Field Descriptions

Bits	Field	Description
31-12	Reserved	Always read as 0. Writes have no effect.
11	rx_pkt_cmp_1	Interrupt enable for USB1 Rx CPPI DMA packet completion status.
10	tx_pkt_cmp_1	Interrupt enable for USB1 Tx CPPI DMA packet completion status
9	rx_pkt_cmp_0	Interrupt enable for USB0 Rx CPPI DMA packet completion status.
8	tx_pkt_cmp_0	Interrupt enable for USB0 Tx CPPI DMA packet completion status.
7-3	Reserved	Always read as 0. Writes have no effect.
2	pd_cmp_flag	Interrupt enable when the packet is completed and the packet descriptor is pushed into the queue manager.
1	rx_mop_starvation	Interrupt enable when queue manager cannot allocate an Rx buffer in the middle of a packet.
0	rx_sop_starvation	Interrupt enable when queue manager cannot allocate an Rx buffer at the start of a packet.

24.9.1.8 USBSS IRQ_DMA_THRESHOLD_TX0_0 Register (IRQDMATHOLDTX00)

The USBSS IRQ_DMA_THRESHOLD_TX0_0 register (IRQDMATHOLDTX00) defines the size of the four DMA thresholds for interrupt pacing for USB0. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_TX0_0 register is shown in Figure 24-29 and described in Table 24-35.

Figure 24-29. USBSS IRQ_DMA_THRESHOLD_TX0_0 Register (IRQDMATHOLDTX00)

31 24	23 16	15 8	7 0
dma_thres_tx0_3	dma_thres_tx0_2	dma_thres_tx0_1	Reserved
R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-35. USBSS IRQ_DMA_THRESHOLD_TX0_0 Register (IRQDMATHOLDTX00) Field Descriptions

Bits	Field	Description
31-24	dma_thres_tx0_3	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 3.
23-16	dma_thres_tx0_2	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 2.
15-8	dma_thres_tx0_1	DMA threshold value for tx_pkt_cmp_0 for USB0 v 1.
7-0	Reserved	Always read as 0. Writes have no effect.

24.9.1.9 USBSS IRQ_DMA_THRESHOLD_TX0_1 Register (IRQDMATHOLDTX01)

The USBSS IRQ_DMA_THRESHOLD_TX0_1 register (IRQDMATHOLDTX01) defines the size of the four DMA thresholds for interrupt pacing for USB0. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_TX0_1 register is shown in Figure 24-30 and described in Table 24-36.

Figure 24-30. USBSS IRQ_DMA_THRESHOLD_TX0_1 Register (IRQDMATHOLDTX01)

31 24	23 16	15 8	7 0
dma_thres_tx0_7	dma_thres_tx0_6	dma_thres_tx0_5	dma_thres_tx0_4
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-36. USBSS IRQ_DMA_THRESHOLD_TX0_1 Register (IRQDMATHOLDTX01) Field Descriptions

Bits	Field	Description
31-24	dma_thres_tx0_7	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 7.
23-16	dma_thres_tx0_6	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 6.
15-8	dma_thres_tx0_5	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 5.
7-0	dma_thres_tx0_4	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 4.

24.9.1.10 USBSS IRQ_DMA_THRESHOLD_TX0_2 Register (IRQDMATHOLDTX02)

The USBSS IRQ_DMA_THRESHOLD_TX0_2 register (IRQDMATHOLDTX02) defines the size of the four DMA thresholds for interrupt pacing for USB0. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_TX0_2 register is shown in Figure 24-31 and described in Table 24-37.

Figure 24-31. USBSS IRQ_DMA_THRESHOLD_TX0_2 Register (IRQDMATHOLDTX02)

31 24	23 16	15 8	7 0
dma_thres_tx0_11	dma_thres_tx0_10	dma_thres_tx0_9	dma_thres_tx0_8
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-37. USBSS IRQ_DMA_THRESHOLD_TX0_2 Register (IRQDMATHOLDTX02) Field Descriptions

Bits	Field	Description
31-24	dma_thres_tx0_11	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 11.
23-16	dma_thres_tx0_10	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 10.
15-8	dma_thres_tx0_9	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 9.
7-0	dma_thres_tx0_8	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 8.

24.9.1.11 USBSS IRQ_DMA_THRESHOLD_TX0_3 Register (IRQDMATHOLDTX03)

The USBSS IRQ_DMA_THRESHOLD_TX0_3 register (IRQDMATHOLDTX03) defines the size of the four DMA thresholds for interrupt pacing for USB0. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_TX0_3 register is shown in Figure 24-31 and described in Table 24-37.

Figure 24-32. USBSS IRQ_DMA_THRESHOLD_TX0_3 Register (IRQDMATHOLDTX03)

31 24	23 16	15 8	7 0
dma_thres_tx0_15	dma_thres_tx0_14	dma_thres_tx0_13	dma_thres_tx0_12
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-38. USBSS IRQ_DMA_THRESHOLD_TX0_3 Register (IRQDMATHOLDTX03) Field Descriptions

Bits	Field	Description
31-24	dma_thres_tx0_15	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 15.
23-16	dma_thres_tx0_14	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 14.
15-8	dma_thres_tx0_13	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 13.
7-0	dma_thres_tx0_12	DMA threshold value for tx_pkt_cmp_0 for USB0 endpoint 12.

24.9.1.12 USBSS IRQ_DMA_THRESHOLD_RX0_0 Register (IRQDMATHOLDRX00)

The USBSS IRQ_DMA_THRESHOLD_RX0_0 register (IRQDMATHOLDRX00) defines the size of the four DMA thresholds for interrupt pacing for USB0. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_RX0_0 register is shown in Figure 24-33 and described in Table 24-39.

Figure 24-33. USBSS IRQ_DMA_THRESHOLD_RX0_0 Register (IRQDMATHOLDRX00)

31 24	23 16	15 8	7 0
dma_thres_rx0_3	dma_thres_rx0_2	dma_thres_rx0_1	Reserved
R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-39. USBSS IRQ_DMA_THRESHOLD_RX0_0 Register (IRQDMATHOLDRX00) Field Descriptions

Bits	Field	Description
31-24	dma_thres_rx0_3	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 3.
23-16	dma_thres_rx0_2	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 2.
15-8	dma_thres_rx0_1	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 1.
7-0	Reserved	Always read as 0. Writes have no effect.

24.9.1.13 USBSS IRQ_DMA_THRESHOLD_RX0_1 Register (IRQDMATHOLDRX01)

The USBSS IRQ_DMA_THRESHOLD_RX0_1 register (IRQDMATHOLDRX01) defines the size of the four DMA thresholds for interrupt pacing for USB0. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_RX0_1 register is shown in Figure 24-34 and described in Table 24-40.

Figure 24-34. USBSS IRQ_DMA_THRESHOLD_RX0_1 Register (IRQDMATHOLDRX01)

31 24	23 16	15 8	7 0
dma_thres_rx0_7	dma_thres_rx0_6	dma_thres_rx0_5	dma_thres_rx0_4
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-40. USBSS IRQ_DMA_THRESHOLD_RX0_1 Register (IRQDMATHOLDRX00) Field Descriptions

Bits	Field	Description
31-24	dma_thres_rx0_7	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 7.
23-16	dma_thres_rx0_6	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 6.
15-8	dma_thres_rx0_5	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 5.
7-0	dma_thres_rx0_4	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 4.

24.9.1.14 USBSS IRQ_DMA_THRESHOLD_RX0_2 Register (IRQDMATHOLDRX02)

The USBSS IRQ_DMA_THRESHOLD_RX0_2 register defines the size of the four DMA thresholds for interrupt pacing. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_RX0_2 register is shown in Figure 24-35 and described in Table 24-41.

Figure 24-35. USBSS IRQ_DMA_THRESHOLD_RX0_2 Register (IRQDMATHOLDRX02)

31 24	23 16	15 8	7 0
dma_thres_rx0_11	dma_thres_rx0_10	dma_thres_rx0_9	dma_thres_rx0_8
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-41. USBSS IRQ_DMA_THRESHOLD_RX0_2 Register (IRQDMATHOLDRX02) Field Descriptions

Bits	Field	Description
31-24	dma_thres_rx0_11	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 11.
23-16	dma_thres_rx0_10	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 10.
15-8	dma_thres_rx0_9	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 9.
7-0	dma_thres_rx0_8	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 8.

24.9.1.15 USBSS IRQ_DMA_THRESHOLD_RX0_3 Register (IRQDMATHOLDRX03)

The USBSS IRQDMATHOLDRX03 register defines the size of the four DMA thresholds for interrupt pacing for USB0. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_RX0_3 register is shown in Figure 24-36 and described in Table 24-42.

Figure 24-36. USBSS IRQ_DMA_THRESHOLD_RX0_3 Register (IRQDMATHOLDRX03)

31 24	23 16	15 8	7 0
dma_thres_rx0_15	dma_thres_rx0_14	dma_thres_rx0_13	dma_thres_rx0_12
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-42. USBSS IRQ_DMA_THRESHOLD_RX0_3 Register (IRQDMATHOLDRX03) Field Descriptions

Bits	Field	Description
31-24	dma_thres_rx0_15	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 15.
23-16	dma_thres_rx0_14	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 14.
15-8	dma_thres_rx0_13	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 13.
7-0	dma_thres_rx0_12	DMA threshold value for rx_pkt_cmp_0 for USB0 endpoint 12.

24.9.1.16 USBSS IRQ_DMA_THRESHOLD_TX1_0 Register (IRQDMATHOLDTX10)

The USBSS IRQ_DMA_THRESHOLD_TX1_0 register (IRQDMATHOLDTX10) defines the size of the four DMA thresholds for interrupt pacing for USB1. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_TX1_0 register is shown in Figure 24-37 and described in Table 24-43.

Figure 24-37. USBSS IRQ_DMA_THRESHOLD_TX1_0 Register (IRQDMATHOLDTX10)

31 24	23 16	15 8	7 0
dma_thres_tx1_3	dma_thres_tx1_2	dma_thres_tx1_1	Reserved
R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-43. USBSS IRQ_DMA_THRESHOLD_TX1_0 Register (IRQDMATHOLDTX10) Field Descriptions

Bits	Field	Description
31-24	dma_thres_tx1_3	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 3.
23-16	dma_thres_tx1_2	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 2.
15-8	dma_thres_tx1_1	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 1.
7-0	Reserved	Always read as 0. Writes have no effect.

24.9.1.17 USBSS IRQ_DMA_THRESHOLD_TX1_1 Register (IRQDMATHOLDTX11)

The USBSS IRQ_DMA_THRESHOLD_TX1_1 register (IRQDMATHOLDTX11) defines the size of the four DMA thresholds for interrupt pacing for USB1. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_TX1_1 register is shown in Figure 24-38 and described in Table 24-44.

Figure 24-38. USBSS IRQ_DMA_THRESHOLD_TX1_1 Register (IRQDMATHOLDTX11)

31 24	23 16	15 8	7 0
dma_thres_tx1_7	dma_thres_tx1_6	dma_thres_tx1_5	dma_thres_tx1_4
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-44. USBSS IRQ_DMA_THRESHOLD_TX1_1 Register (IRQDMATHOLDTX11) Field Descriptions

Bits	Field	Description
31-24	dma_thres_tx1_7	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 7.
23-16	dma_thres_tx1_6	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 6.
15-8	dma_thres_tx1_5	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 5.
7-0	dma_thres_tx1_4	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 4.

24.9.1.18 USBSS IRQ_DMA_THRESHOLD_TX1_2 Register (IRQDMATHOLDTX12)

The USBSS IRQ_DMA_THRESHOLD_TX1_2 register (IRQDMATHOLDTX12) defines the size of the four DMA thresholds for interrupt pacing for USB1. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_TX1_2 register is shown in Figure 24-39 and described in Table 24-45.

Figure 24-39. USBSS IRQ_DMA_THRESHOLD_TX1_2 Register (IRQDMATHOLDTX12)

31 24	23 16	15 8	7 0
dma_thres_tx1_11	dma_thres_tx1_10	dma_thres_tx1_9	dma_thres_tx1_8
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-45. USBSS IRQ_DMA_THRESHOLD_TX1_2 Register (IRQDMATHOLDTX12) Field Descriptions

Bits	Field	Description
31-24	dma_thres_tx1_11	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 11.
23-16	dma_thres_tx1_10	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 10.
15-8	dma_thres_tx1_9	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 9.
7-0	dma_thres_tx1_8	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 8.

24.9.1.19 USBSS IRQ_DMA_THRESHOLD_TX1_3 Register (IRQDMATHOLDTX13)

The USBSS IRQ_DMA_THRESHOLD_TX1_3 register (IRQDMATHOLDTX13) defines the size of the four DMA thresholds for interrupt pacing for USB1. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_TX1_3 register is shown in Figure 24-40 and described in Table 24-46.

Figure 24-40. USBSS IRQ_DMA_THRESHOLD_TX1_3 Register (IRQDMATHOLDTX13)

31 24	23 16	15 8	7 0
dma_thres_tx1_15	dma_thres_tx1_14	dma_thres_tx1_13	dma_thres_tx1_12
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-46. USBSS IRQ_DMA_THRESHOLD_TX1_3 Register (IRQDMATHOLDTX13) Field Descriptions

Bits	Field	Description
31-24	dma_thres_tx1_15	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 15.
23-16	dma_thres_tx1_14	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 14.
15-8	dma_thres_tx1_13	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 13.
7-0	dma_thres_tx1_12	DMA threshold value for tx_pkt_cmp_0 for USB1 endpoint 12.

24.9.1.20 USBSS IRQ_DMA_THRESHOLD_RX1_0 Register (IRQDMATHOLDRX10)

The USBSS IRQ_DMA_THRESHOLD_RX1_0 register (IRQDMATHOLDRX10) defines the size of the four DMA thresholds for interrupt pacing for USB1. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_RX1_0 register is shown in Figure 24-41 and described in Table 24-47.

Figure 24-41. USBSS IRQ_DMA_THRESHOLD_RX1_0 Register (IRQDMATHOLDRX10)

31 24	23 16	15 8	7 0
dma_thres_rx1_3	dma_thres_rx1_2	dma_thres_rx1_1	Reserved
R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-47. USBSS IRQ_DMA_THRESHOLD_RX1_0 Register (IRQDMATHOLDRX10) Field Descriptions

Bits	Field	Description
31-24	dma_thres_rx1_3	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 3.
23-16	dma_thres_rx1_2	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 2.
15-8	dma_thres_rx1_1	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 1.
7-0	Reserved	Always read as 0. Writes have no effect.

24.9.1.21 USBSS IRQ_DMA_THRESHOLD_RX1_1 Register (IRQDMATHOLDRX11)

The USBSS IRQ_DMA_THRESHOLD_RX1_1 register (IRQDMATHOLDRX11) defines the size of the four DMA thresholds for interrupt pacing for USB1. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_RX1_1 register is shown in Figure 24-42 and described in Table 24-48.

Figure 24-42. USBSS IRQ_DMA_THRESHOLD_RX1_1 Register (IRQDMATHOLDRX11)

31 24	23 16	15 8	7 0
dma_thres_rx1_7	dma_thres_rx1_6	dma_thres_rx1_5	dma_thres_rx1_4
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-48. USBSS IRQ_DMA_THRESHOLD_RX1_1 Register (IRQDMATHOLDRX11) Field Descriptions

Bits	Field	Description
31-24	dma_thres_rx1_7	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 7.
23-16	dma_thres_rx1_6	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 6.
15-8	dma_thres_rx1_5	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 5.
7-0	dma_thres_rx1_4	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 4.

24.9.1.22 USBSS IRQ_DMA_THRESHOLD_RX1_2 Register (IRQDMATHOLDRX12)

The USBSS IRQ_DMA_THRESHOLD_RX1_2 register (IRQDMATHOLDRX12) defines the size of the four DMA thresholds for interrupt pacing for USB1. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_RX1_2 register is shown in Figure 24-43 and described in Table 24-49.

Figure 24-43. USBSS IRQ_DMA_THRESHOLD_RX1_2 Register (IRQDMATHOLDRX12)

31 24	23 16	15 8	7 0
dma_thres_rx1_11	dma_thres_rx1_10	dma_thres_rx1_9	dma_thres_rx1_8
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-49. USBSS IRQ_DMA_THRESHOLD_RX1_2 Register (IRQDMATHOLDRX12) Field Descriptions

Bits	Field	Description
31-24	dma_thres_rx1_11	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 11.
23-16	dma_thres_rx1_10	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 10.
15-8	dma_thres_rx1_9	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 9.
7-0	dma_thres_rx1_8	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 8.

24.9.1.23 USBSS IRQ_DMA_THRESHOLD_RX1_3 Register (IRQDMATHOLDRX13)

The USBSS IRQ_DMA_THRESHOLD_RX1_3 register (IRQDMATHOLDRX13) defines the size of the four DMA thresholds for interrupt pacing for USB1. Each threshold contains an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_DMA_THRESHOLD_RX1_3 register is shown in Figure 24-44 and described in Table 24-50.

Figure 24-44. USBSS IRQ_DMA_THRESHOLD_RX1_3 Register (IRQDMATHOLDRX13)

31 24	23 16	15 8	7 0
dma_thres_rx1_15	dma_thres_rx1_14	dma_thres_rx1_13	dma_thres_rx1_12
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-50. USBSS IRQ_DMA_THRESHOLD_RX1_3 Register (IRQDMATHOLDRX13) Field Descriptions

Bits	Field	Description
31-24	dma_thres_rx1_15	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 15.
23-16	dma_thres_rx1_14	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 14.
15-8	dma_thres_rx1_13	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 13.
7-0	dma_thres_rx1_12	DMA threshold value for rx_pkt_cmp_0 for USB1 endpoint 12.

24.9.1.24 USBSS IRQ_DMA_ENABLE_0 Register (IRQDMAENABLE0)

The USBSS IRQ_DMA_ENABLE_0 register (IRQDMAENABLE0) contains 32 enables. Sixteen enables are assigned to the 16 endpoints associated with the tx_pkt_cmp_0 interrupt and 16 enables are assigned to the 16 endpoints associated with the rx_pkt_cmp_0 interrupt. Endpoint 0 for both the tx and rx does not exist; but it is included for completeness of this register.

If the enable is set; then that specific endpoint can generate an interrupt if the event pending count has exceeded the threshold.

If the enable is cleared; then that specific endpoint cannot generate an interrupt regardless if the event pending count has exceeded the threshold or not. The USBSS IRQ_DMA_ENABLE_0 Register is shown in Figure 24-45 and described in Table 24-51.

Figure 24-45. USBSS IRQ_DMA_ENABLE_0 Register (IRQDMAENABLE0)

31	18	17	16
dma_en_rx0_15		dma_en_rx0_1	Rsvd
R/W-0		R/W-0	R-0

15	3	2	1	0
dma_en_tx0_15		dma_en_tx0_2	dma_en_tx0_1	Rsvd
R/W-0		R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 24-51. USBSS IRQ_DMA_ENABLE_0 Register (IRQDMAENABLE0) Field Descriptions

Bits	Field	Description
31-18	dma_en_rx0_15	DMA threshold enable value for rx_pkt_cmp_0 for USB0 endpoint 15.
17	dma_en_rx0_1	DMA threshold enable value for rx_pkt_cmp_0 for USB0 endpoint 1.
16	Reserved	Always read as 0. Writes have no effect.
15-3	dma_en_tx0_15	DMA threshold enable value for tx_pkt_cmp_0 for USB0 endpoint 15.
2	dma_en_tx0_2	DMA threshold enable value for tx_pkt_cmp_0 for USB0 endpoint 2.
1	dma_en_tx0_1	DMA threshold enable value for tx_pkt_cmp_0 for USB0 endpoint 1.
0	Reserved	Always read as 0. Writes have no effect.

24.9.1.25 USBSS IRQ_DMA_ENABLE_1 Register (IRQDMAENABLE1)

The USBSS IRQ_DMA_ENABLE_1 register (IRQDMAENABLE1) contains 32 enables. Sixteen enables are assigned to the 16 endpoints associated with the tx_pkt_cmp_1 interrupt and 16 enables are assigned to the 16 endpoints associated with the rx_pkt_cmp_1 interrupt. Endpoint 0 for both the tx and rx does not exist; but it is included for completeness of this register.

If the enable is set; then that specific endpoint can generate an interrupt if the event pending count has exceeded the threshold.

If the enable is cleared; then that specific endpoint cannot generate an interrupt regardless if the event pending count has exceeded the threshold or not. The USBSS IRQ_DMA_ENABLE_0 Register is shown in Figure 24-46 and described in Table 24-52.

Figure 24-46. USBSS IRQ_DMA_ENABLE_1 Register (IRQDMAENABLE1)

31	18	17	16
dma_en_rx1_15		dma_en_rx1_1	Rsvd
R/W-0		R/W-0	R-0

15	2	1	0
dma_en_tx1_15		dma_en_tx1_1	Rsvd
R/W-0		R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-52. USBSS IRQ_DMA_ENABLE_1 Register (IRQDMAENABLE1) Field Descriptions

Bits	Field	Description
31-18	dma_en_rx1_15	DMA threshold enable value for rx_pkt_cmp_1 for USB1 endpoint 15.
17	dma_en_rx1_1	DMA threshold enable value for rx_pkt_cmp_1 for USB1 endpoint 1.
16	Reserved	Always read as 0. Writes have no effect.
15-2	dma_en_tx1_15	DMA threshold enable value for tx_pkt_cmp_1 for USB1 endpoint 15.
1	dma_en_tx1_1	DMA threshold enable value for tx_pkt_cmp_1 for USB1 endpoint 1.
0	Reserved	Always read as 0. Writes have no effect.

24.9.1.26 USBSS IRQ_FRAME_THRESHOLD_TX0_0 Register (IRQFRAMETHOLD00)

The USBSS IRQ_FRAME_THRESHOLD_TX0_0 register (IRQFRAMETHOLD00) defines the size of the four FRAME thresholds for interrupt pacing for USB0. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_TX0_0 register is shown in Figure 24-47 and described in Table 24-53.

Figure 24-47. USBSS IRQ_FRAME_THRESHOLD_TX0_0 Register (IRQFRAMETHOLD00)

31 24	23 16	15 8	7 0
frame_thres_tx1_3	frame_thres_tx1_2	frame_thres_tx1_1	Reserved
R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-53. USBSS IRQ_FRAME_THRESHOLD_TX0_0 Register (IRQFRAMETHOLD00) Field Descriptions

Bits	Field	Description
31-24	frame_thres_tx1_3	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 3.
23-16	frame_thres_tx1_2	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 2.
15-8	frame_thres_tx1_1	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 1.
7-0	Reserved	Always read as 0. Writes have no effect.

24.9.1.27 USBSS IRQ_FRAME_THRESHOLD_TX0_1 Register (IRQFRAMETHOLDTX01)

The USBSS IRQ_FRAME_THRESHOLD_TX0_1 register (IRQFRAMETHOLDTX01) defines the size of the four FRAME thresholds for interrupt pacing for USB0. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_TX0_1 register is shown in Figure 24-48 and described in Table 24-54.

Figure 24-48. USBSS IRQ_FRAME_THRESHOLD_TX0_1 Register (IRQFRAMETHOLD01)

31 24	23 16	15 8	7 0
frame_thres_tx1_7	frame_thres_tx1_6	frame_thres_tx1_5	frame_thres_tx1_4
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-54. USBSS IRQ_FRAME_THRESHOLD_TX0_1 Register (IRQFRAMETHOLDTX01) Field Descriptions

Bits	Field	Description
31-24	frame_thres_tx1_7	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 7.
23-16	frame_thres_tx1_6	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 6.
15-8	frame_thres_tx1_5	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 5.
7-0	frame_thres_tx1_4	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 4.

24.9.1.28 USBSS IRQ_FRAME_THRESHOLD_TX0_2 Register (IRQFRAMETHOLDTX02)

The USBSS IRQ_FRAME_THRESHOLD_TX0_2 register (IRQFRAMETHOLDTX02) defines the size of the four FRAME thresholds for interrupt pacing for USB0. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_TX0_2 register is shown in Figure 24-49 and described in Table 24-55.

Figure 24-49. USBSS IRQ_FRAME_THRESHOLD_TX0_2 Register (IRQFRAMETHOLD02)

31 24	23 16	15 8	7 0
frame_thres_tx1_11	frame_thres_tx1_10	frame_thres_tx1_9	frame_thres_tx1_8
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-55. USBSS IRQ_FRAME_THRESHOLD_TX0_2 Register (IRQFRAMETHOLDTX02) Field Descriptions

Bits	Field	Description
31-24	frame_thres_tx1_11	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 11.
23-16	frame_thres_tx1_10	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 10.
15-8	frame_thres_tx1_9	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 9.
7-0	frame_thres_tx1_8	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 8.

24.9.1.29 USBSS IRQ_FRAME_THRESHOLD_TX0_3 Register (IRQFRAMETHOLDTX03)

The USBSS IRQ_FRAME_THRESHOLD_TX0_3 register (IRQFRAMETHOLDTX03) defines the size of the four FRAME thresholds for interrupt pacing for USB0. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_TX0_3 register is shown in Figure 24-50 and described in Table 24-56.

Figure 24-50. USBSS IRQ_FRAME_THRESHOLD_TX0_3 Register (IRQFRAMETHOLD03)

31 24	23 16	15 8	7 0
frame_thres_tx1_15	frame_thres_tx1_14	frame_thres_tx1_13	frame_thres_tx1_12
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-56. USBSS IRQ_FRAME_THRESHOLD_TX0_3 Register (IRQFRAMETHOLDTX03) Field Descriptions

Bits	Field	Description
31-24	frame_thres_tx1_15	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 15.
23-16	frame_thres_tx1_14	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 14.
15-8	frame_thres_tx1_13	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 13.
7-0	frame_thres_tx1_12	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 12.

24.9.1.30 USBSS IRQ_FRAME_THRESHOLD_RX0_0 Register (IRQFRAMETHOLDRX00)

The USBSS IRQ_FRAME_THRESHOLD_RX0_0 register (IRQFRAMETHOLDRX00) defines the size of the four FRAME thresholds for interrupt pacing for USB0. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_RX0_0 register is shown in Figure 24-51 and described in Table 24-57.

Figure 24-51. USBSS IRQ_FRAME_THRESHOLD_RX0_0 Register (IRQFRAMETHOLDRX00)

31 24	23 16	15 8	7 0
frame_thres_rx1_3	frame_thres_rx1_2	frame_thres_rx1_1	Reserved
R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-57. USBSS IRQ_FRAME_THRESHOLD_RX0_0 Register (IRQFRAMETHOLDRX00) Field Descriptions

Bits	Field	Description
31-24	frame_thres_rx1_3	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 3.
23-16	frame_thres_rx1_2	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 2.
15-8	frame_thres_rx1_1	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 1.
7-0	Reserved	Always read as 0. Writes have no effect.

24.9.1.31 USBSS IRQ_FRAME_THRESHOLD_RX0_1 Register (IRQFRAMETHOLDRX01)

The USBSS IRQ_FRAME_THRESHOLD_RX0_1 register (IRQFRAMETHOLDRX01) defines the size of the four FRAME thresholds for interrupt pacing for USB0. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_RX0_1 register is shown in Figure 24-52 and described in Table 24-58.

Figure 24-52. USBSS IRQ_FRAME_THRESHOLD_RX0_1 Register (IRQFRAMETHOLDRX01)

31 24	23 16	15 8	7 0
frame_thres_rx1_7	frame_thres_rx1_6	frame_thres_rx1_5	frame_thres_rx1_4
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-58. USBSS IRQ_FRAME_THRESHOLD_RX0_1 Register (IRQFRAMETHOLDRX01) Field Descriptions

Bits	Field	Description
31-24	frame_thres_rx1_7	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 7.
23-16	frame_thres_rx1_6	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 6.
15-8	frame_thres_rx1_5	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 5.
7-0	frame_thres_rx1_4	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 4.

24.9.1.32 USBSS IRQ_FRAME_THRESHOLD_RX0_2 Register (IRQFRAMETHOLDRX02)

The USBSS IRQ_FRAME_THRESHOLD_RX0_2 register (IRQFRAMETHOLDRX02) defines the size of the four FRAME thresholds for interrupt pacing for USB0. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_RX0_2 register is shown in Figure 24-52 and described in Table 24-59.

Figure 24-53. USBSS IRQ_FRAME_THRESHOLD_RX0_2 Register (IRQFRAMETHOLDRX02)

31 24	23 16	15 8	7 0
frame_thres_rx1_11	frame_thres_rx1_10	frame_thres_rx1_9	frame_thres_rx1_8
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-59. USBSS IRQ_FRAME_THRESHOLD_RX0_2 Register (IRQFRAMETHOLDRX02) Field Descriptions

Bits	Field	Description
31-24	frame_thres_rx1_11	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 11.
23-16	frame_thres_rx1_10	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 10.
15-8	frame_thres_rx1_9	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 9.
7-0	frame_thres_rx1_8	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 8.

24.9.1.33 USBSS IRQ_FRAME_THRESHOLD_RX0_3 Register (IRQFRAMETHOLDRX03)

The USBSS IRQ_FRAME_THRESHOLD_RX0_3 register (IRQFRAMETHOLDRX03) defines the size of the four FRAME thresholds for interrupt pacing for USB0. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_RX0_3 register is shown in Figure 24-54 and described in Table 24-60.

Figure 24-54. USBSS IRQ_FRAME_THRESHOLD_RX0_3 Register (IRQFRAMETHOLDRX03)

31 24	23 16	15 8	7 0
frame_thres_rx1_15	frame_thres_rx1_14	frame_thres_rx1_13	frame_thres_rx1_12
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-60. USBSS IRQ_FRAME_THRESHOLD_RX0_3 Register (IRQFRAMETHOLDRX03) Field Descriptions

Bits	Field	Description
31-24	frame_thres_rx1_15	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 15.
23-16	frame_thres_rx1_14	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 14.
15-8	frame_thres_rx1_13	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 13.
7-0	frame_thres_rx1_12	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 12.

24.9.1.34 USBSS IRQ_FRAME_THRESHOLD_TX1_0 Register (IRQFRAMETHOLD10)

The USBSS IRQ_FRAME_THRESHOLD_TX1_0 register (IRQFRAMETHOLD10) defines the size of the four FRAME thresholds for interrupt pacing for USB1. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_TX1_0 register is shown in Figure 24-55 and described in Table 24-61.

Figure 24-55. USBSS IRQ_FRAME_THRESHOLD_TX1_0 Register (IRQFRAMETHOLD10)

31 24	23 16	15 8	7 0
frame_thres_tx1_3	frame_thres_tx1_2	frame_thres_tx1_1	Reserved
R/W-0	R/W-0	R/W-0	R/0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-61. USBSS IRQ_FRAME_THRESHOLD_TX1_0 Register (IRQFRAMETHOLD10) Field Descriptions

Bits	Field	Description
31-24	frame_thres_tx1_3	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 3.
23-16	frame_thres_tx1_2	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 2.
15-8	frame_thres_tx1_1	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 1.
7-0	Reserved	Always read as 0. Writes have no effect.

24.9.1.35 USBSS IRQ_FRAME_THRESHOLD_TX1_1 Register (IRQFRAMETHOLDTX11)

The USBSS IRQ_FRAME_THRESHOLD_TX1_1 register (IRQFRAMETHOLDTX11) defines the size of the four FRAME thresholds for interrupt pacing for USB1. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_TX1_1 register is shown in Figure 24-56 and described in Table 24-62.

Figure 24-56. USBSS IRQ_FRAME_THRESHOLD_TX1_1 Register (IRQFRAMETHOLD11)

31 24	23 16	15 8	7 0
frame_thres_tx1_7	frame_thres_tx1_6	frame_thres_tx1_5	frame_thres_tx1_4
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-62. USBSS IRQ_FRAME_THRESHOLD_TX1_1 Register (IRQFRAMETHOLDTX11) Field Descriptions

Bits	Field	Description
31-24	frame_thres_tx1_7	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 7.
23-16	frame_thres_tx1_6	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 6.
15-8	frame_thres_tx1_5	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 5.
7-0	frame_thres_tx1_4	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 4.

24.9.1.36 USBSS IRQ_FRAME_THRESHOLD_TX1_2 Register (IRQFRAMETHOLDTX12)

The USBSS IRQ_FRAME_THRESHOLD_TX1_2 register (IRQFRAMETHOLDTX12) defines the size of the four FRAME thresholds for interrupt pacing for USB1. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_TX1_2 register is shown in Figure 24-57 and described in Table 24-63.

Figure 24-57. USBSS IRQ_FRAME_THRESHOLD_TX1_2 Register (IRQFRAMETHOLD12)

31 24	23 16	15 8	7 0
frame_thres_tx1_11	frame_thres_tx1_10	frame_thres_tx1_9	frame_thres_tx1_8
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-63. USBSS IRQ_FRAME_THRESHOLD_TX1_2 Register (IRQFRAMETHOLDTX12) Field Descriptions

Bits	Field	Description
31-24	frame_thres_tx1_11	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 11.
23-16	frame_thres_tx1_10	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 10.
15-8	frame_thres_tx1_9	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 9.
7-0	frame_thres_tx1_8	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 8.

24.9.1.37 USBSS IRQ_FRAME_THRESHOLD_TX1_3 Register (IRQFRAMETHOLDTX13)

The USBSS IRQ_FRAME_THRESHOLD_TX1_3 register (IRQFRAMETHOLDTX13) defines the size of the four FRAME thresholds for interrupt pacing for USB1. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_TX1_3 register is shown in Figure 24-58 and described in Table 24-64.

Figure 24-58. USBSS IRQ_FRAME_THRESHOLD_TX1_3 Register (IRQFRAMETHOLD13)

31 24	23 16	15 8	7 0
frame_thres_tx1_15	frame_thres_tx1_14	frame_thres_tx1_13	frame_thres_tx1_12
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-64. USBSS IRQ_FRAME_THRESHOLD_TX1_3 Register (IRQFRAMETHOLDTX13) Field Descriptions

Bits	Field	Description
31-24	frame_thres_tx1_15	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 15.
23-16	frame_thres_tx1_14	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 14.
15-8	frame_thres_tx1_13	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 13.
7-0	frame_thres_tx1_12	FRAME threshold value for tx_pkt_cmp_0 for USB1 endpoint 12.

24.9.1.38 USBSS IRQ_FRAME_THRESHOLD_RX1_0 Register (IRQFRAMETHOLDRX10)

The USBSS IRQ_FRAME_THRESHOLD_RX1_0 register (IRQFRAMETHOLDRX10) defines the size of the four FRAME thresholds for interrupt pacing for USB1. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_RX1_0 register is shown in Figure 24-59 and described in Table 24-65.

Figure 24-59. USBSS IRQ_FRAME_THRESHOLD_RX1_0 Register (IRQFRAMETHOLDRX10)

31 24	23 16	15 8	7 0
frame_thres_rx1_3	frame_thres_rx1_2	frame_thres_rx1_1	Reserved
R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-65. USBSS IRQ_FRAME_THRESHOLD_RX1_0 Register (IRQFRAMETHOLDRX10) Field Descriptions

Bits	Field	Description
31-24	frame_thres_rx1_3	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 3.
23-16	frame_thres_rx1_2	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 2.
15-8	frame_thres_rx1_1	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 1.
7-0	Reserved	Always read as 0. Writes have no effect.

24.9.1.39 USBSS IRQ_FRAME_THRESHOLD_RX1_1 Register (IRQFRAMETHOLDRX11)

The USBSS IRQ_FRAME_THRESHOLD_RX1_1 register (IRQFRAMETHOLDRX11) defines the size of the four FRAME thresholds for interrupt pacing for USB1. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_RX1_1 register is shown in Figure 24-60 and described in Table 24-66.

Figure 24-60. USBSS IRQ_FRAME_THRESHOLD_RX1_1 Register (IRQFRAMETHOLDRX11)

31 24	23 16	15 8	7 0
frame_thres_rx1_7	frame_thres_rx1_6	frame_thres_rx1_5	frame_thres_rx1_4
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-66. USBSS IRQ_FRAME_THRESHOLD_RX1_1 Register (IRQFRAMETHOLDRX11) Field Descriptions

Bits	Field	Description
31-24	frame_thres_rx1_7	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 7.
23-16	frame_thres_rx1_6	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 6.
15-8	frame_thres_rx1_5	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 5.
7-0	frame_thres_rx1_4	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 4.

24.9.1.40 USBSS IRQ_FRAME_THRESHOLD_RX1_2 Register (IRQFRAMETHOLDRX12)

The USBSS IRQ_FRAME_THRESHOLD_RX1_2 register (IRQFRAMETHOLDRX12) defines the size of the four FRAME thresholds for interrupt pacing for USB1. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_RX1_2 register is shown in Figure 24-61 and described in Table 24-67.

Figure 24-61. USBSS IRQ_FRAME_THRESHOLD_RX1_2 Register (IRQFRAMETHOLDRX12)

31 24	23 16	15 8	7 0
frame_thres_rx1_11	frame_thres_rx1_10	frame_thres_rx1_9	frame_thres_rx1_8
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-67. USBSS IRQ_FRAME_THRESHOLD_RX1_2 Register (IRQFRAMETHOLDRX12) Field Descriptions

Bits	Field	Description
31-24	frame_thres_rx1_11	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 11.
23-16	frame_thres_rx1_10	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 10.
15-8	frame_thres_rx1_9	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 9.
7-0	frame_thres_rx1_8	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 8.

24.9.1.41 USBSS IRQ_FRAME_THRESHOLD_RX1_3 Register (IRQFRAMETHOLDRX13)

The USBSS IRQ_FRAME_THRESHOLD_RX1_3 register (IRQFRAMETHOLDRX13) defines the size of the four FRAME thresholds for interrupt pacing for USB1. Each threshold contains is an 8-bit unsigned number and can range from 0 to 255. A possible interrupt can be triggered if the count for that specific endpoint has exceeded the value of the threshold. The counter for the compared value is also an 8-bit unsigned number; therefore, setting the threshold to 255 prevents the possibility of a trigger.

The USBSS IRQ_FRAME_THRESHOLD_RX1_3 register is shown in Figure 24-62 and described in Table 24-68.

Figure 24-62. USBSS IRQ_FRAME_THRESHOLD_RX1_3 Register (IRQFRAMETHOLDRX13)

31 24	23 16	15 8	7 0
frame_thres_rx1_15	frame_thres_rx1_14	frame_thres_rx1_13	frame_thres_rx1_12
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-68. USBSS IRQ_FRAME_THRESHOLD_RX1_3 Register (IRQFRAMETHOLDRX13) Field Descriptions

Bits	Field	Description
31-24	frame_thres_rx1_15	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 15.
23-16	frame_thres_rx1_14	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 14.
15-8	frame_thres_rx1_13	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 13.
7-0	frame_thres_rx1_12	FRAME threshold value for rx_pkt_cmp_0 for USB1 endpoint 12.

24.9.1.42 USBSS IRQ_FRAME_ENABLE_0 Register (IRQFRAMEENABLE0)

The USBSS IRQ_FRAME_ENABLE_0 register (IRQFRAMEENABLE0) contains 30 enables for USB0. Fifteen enables are assigned to the 15 endpoints associated with the tx_pkt_cmp_0 interrupt and 15 enables are assigned to the 15 endpoints associated with the rx_pkt_cmp_0 interrupt. Endpoint 0 for both the tx and rx does not exist; but it is included for completeness of this register.

If the enable is set; then that specific endpoint can generate an interrupt if the frame count has exceeded the threshold.

If the enable is not set; then no hardware interrupt will be generated for that specific endpoint. It is still possible to generate the interrupt via software. If the enable is cleared; then that specific endpoint cannot generate an interrupt regardless if the frame count has exceeded the threshold or not. The USBSS IRQ_FRAME_ENABLE_0 register is shown in Figure 24-63 and described in Table 24-69.

Figure 24-63. USBSS IRQ_FRAME_ENABLE_0 Register (IRQFRAMEENABLE0)

31	18	17	16
frame_en_rx0_15		frame_en_rx0_1	Rsvd
R/W-0		R/W-0	R-0

15	2	1	0
frame_en_tx0_15		frame_en_tx0_1	Rsvd
R/W-0		R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 24-69. USBSS IRQ_FRAME_ENABLE_0 Register (IRQFRAMEENABLE0) Field Descriptions

Bits	Field	Description
31-18	frame_en_rx0_15	FRAME threshold enable value for rx_pkt_cmp_0 for USB0 endpoint 15.
17	frame_en_rx0_1	FRAME threshold enable value for rx_pkt_cmp_0 for USB0 endpoint 1.
16	Reserved	Always read as 0. Writes have no effect.
15-2	frame_en_tx0_15	FRAME threshold enable value for tx_pkt_cmp_0 for USB0 endpoint 15.
1	frame_en_tx0_1	FRAME threshold enable value for tx_pkt_cmp_0 for USB0 endpoint 1.
0	Reserved	Always read as 0. Writes have no effect.

24.9.1.43 USBSS IRQ_FRAME_ENABLE_1 Register (IRQFRAMEENABLE1)

The USBSS IRQ_FRAME_ENABLE_1 register (IRQFRAMEENABLE1) contains 30 enables for USB1. Fifteen enables are assigned to the 15 endpoints associated with the tx_pkt_cmp_1 interrupt and 15 enables are assigned to the 15 endpoints associated with the rx_pkt_cmp_1 interrupt. Endpoint 0 for both the tx and rx does not exist; but it is included for completeness of this register.

If the enable is set; then that specific endpoint can generate an interrupt if the frame count has exceeded the threshold. If the enable is not set; then no hardware interrupt will be generated for that specific endpoint. It is still possible to generate the interrupt via software.

If the enable is cleared; then that specific endpoint cannot generate an interrupt regardless if the frame count has exceeded the threshold or not. The USBSS IRQ_FRAME_ENABLE_1 register is shown in Figure 24-64 and described in Table 24-70.

Figure 24-64. USBSS IRQ_FRAME_ENABLE_1 Register (IRQFRAMEENABLE1)

31	18	17	16
frame_en_rx1_15		frame_en_rx1_1	Rsvd
R/W-0		R/W-0	R-0

15	2	1	0
frame_en_tx1_15		frame_en_tx1_1	Rsvd
R/W-0		R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 24-70. USBSS IRQ_FRAME_ENABLE_1 Register (IRQFRAMEENABLE1) Field Descriptions

Bits	Field	Description
31-18	frame_en_rx1_15	FRAME threshold enable value for rx_pkt_cmp_1 for USB1 endpoint 15.
17	frame_en_rx1_1	FRAME threshold enable value for rx_pkt_cmp_1 for USB1 endpoint 1.
16	Reserved	Always read as 0. Writes have no effect.
15-2	frame_en_tx1_15	FRAME threshold enable value for tx_pkt_cmp_1 for USB1 endpoint 15.
1	frame_en_tx1_1	FRAME threshold enable value for tx_pkt_cmp_1 for USB1 endpoint 1.
0	Reserved	Always read as 0. Writes have no effect.

24.9.2 USB0 and USB1 Controller Registers

The USB0 and USB1 register descriptions are included in this section.

24.9.2.1 USB0 Controller Registers

Table 24-71 lists the registers for the USB0 Controller submodule.

Table 24-71. USB0 Controller Registers

Address Offset	Acronym	USB0 Controller Register
1000h	USB0REV	USB0 REVISION
1014h	USB0CTRL	USB0 Control Register
1018h	USB0STAT	USB0 Status Register
1020h	USB0IRQMSTAT	USB0 IRQ_MERGED_STATUS
1024h	USB0IRQEOI	USB0 IRQ_EOI
1028h	USB0IRQSTATRAW0	USB0 IRQ_STATUS_RAW_0
102Ch	USB0IRQSTATRAW1	USB0 IRQ_STATUS_RAW_1
1030h	USB0IRQSTAT0	USB0 IRQ_STATUS_0
1034h	USB0IRQSTAT1	USB0 IRQ_STATUS_1
1038h	USB0IRQENABLESET0	USB0 IRQ_ENABLE_SET_0
103Ch	USB0IRQENABLESET1	USB0 IRQ_ENABLE_SET_1
1040h	USB0IRQENABLECLR0	USB0 IRQ_ENABLE_CLR_0
1044h	USB0IRQENABLECLR1	USB0 IRQ_ENABLE_CLR_1
1070h	USB0TXMODE	USB0 Tx Mode Register
1074h	USB0RXMODE	USB0 Rx Mode Register
1080h	USB0GENRNDISEP1	USB0 Generic RNDIS Size EP1
1084h	USB0GENRNDISEP2	USB0 Generic RNDIS Size EP2
1088h	USB0GENRNDISEP3	USB0 Generic RNDIS Size EP3
108Ch	USB0GENRNDISEP4	USB0 Generic RNDIS Size EP4
1090h	USB0GENRNDISEP5	USB0 Generic RNDIS Size EP5
1094h	USB0GENRNDISEP6	USB0 Generic RNDIS Size EP6
1098h	USB0GENRNDISEP7	USB0 Generic RNDIS Size EP7
109Ch	USB0GENRNDISEP8	USB0 Generic RNDIS Size EP8
10A0h	USB0GENRNDISEP9	USB0 Generic RNDIS Size EP9
10A4h	USB0GENRNDISEP10	USB0 Generic RNDIS Size EP10
10A8h	USB0GENRNDISEP11	USB0 Generic RNDIS Size EP11
10ACh	USB0GENRNDISEP12	USB0 Generic RNDIS Size EP12
10B0h	USB0GENRNDISEP13	USB0 Generic RNDIS Size EP13
10B4h	USB0GENRNDISEP14	USB0 Generic RNDIS Size EP14
10B8h	USB0GENRNDISEP15	USB0 Generic RNDIS Size EP15
10D0h	USB0AUTOREQ	USB0 Auto Req Register
10D4h	USB0SRPFIXTIME	USB0 SRP Fix Time
10D8h	USB0TDOWN	USB0 Teardown Register
10E0h	USB0UTMI	USB0 PHY UTMI Register
10E4h	USB0UTMILB	USB0 MGC UTMI Loopback Register
10E8h	USB0MODE	USB0 Mode Register
1400h–1468Ch	...	USB0 Mentor Core Registers

24.9.2.1.1 USB0 Revision Register (USB0REV)

The USB0 revision register (USB0REV) contains the major and minor revisions for the USB0 module. The USB0 revision register is shown in Figure 24-65 and described in Table 24-72.

Figure 24-65. USB0 Revision Register (USB0REV)

31	30	29	28	27	16
Scheme		Reserved		func	
R-0		R-0		R-0	

15	11	10	8	7	6	5	0
R_rtl		X_major		Custom		Y_minor	
R-0		R-0		R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 24-72. USB0 Revision Register (USB0REV) Field Descriptions

Bits	Field	Value	Description
31-30	Scheme	0-3h	Used to distinguish between old scheme and current.
29-28	Reserved	0	Reserved
27-16	func	0-FFFh	Function indicates a software compatible module family.
15-11	R_rtl	0-1Fh	RTL revision.
10-8	X_major	0-7h	Major revision.
7-6	Custom	0-3h	Custom revision
5-0	Y_minor	0-3Fh	Minor revision

24.9.2.1.2 USB0 Control Register (USB0CTRL)

The USB0 control register (USB0CTRL) allows the CPU to control various aspects of the module. If uint is set high, then the Mentor controller generic interrupt for USB[9] will be generated (if enabled). This requires software to read the Mentor controller's registers to determine which interrupt USB[0] to USB[7] occurred. If uint is set low, then the usb20otg_f module will automatically read the Mentor controller's registers and set the appropriate interrupt USB[0] to USB[7] (if enabled). The generic interrupt for USB[9] will not be generated.

The USB0 control register is shown in Figure 24-66 and described in Table 24-73.

Figure 24-66. USB0 Control Register (USB0CTRL)

31	30	29	16
dis_deb	dis_srp	Reserved	
R/W-0	R/W-0	R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved		soft reset isolation	rndis	uint	Reserved	clkfack	soft reset
R-0		R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-73. USB0 Control Register (USB0CTRL) Field Descriptions

Bits	Field	Value	Description
31	dis_deb	0-1	Disable the VBUS debouncer circuit fix
30	dis_srp	0-1	Disable the OTG Session Request Protocol (SRP) AVALID circuit fix. When enabled (=0) this allows additional time for the VBUS signal to be measured against the VBUS thresholds. The time is specified in the USB0 SRP Fix time register.
29-6	Reserved	0	Always read as 0. Writes have no effect.
5	soft reset isolation	0-1	Soft reset isolation. When high, this bit forces all USB0 signals that connect to the USBSS to zeros during a soft reset via bit 0 of this register. This bit should be set high prior to setting bit 0 and cleared after bit 0 is cleared.
4	rndis	0-1	Global RNDIS mode enable for all endpoints.
3	uint		USB non-highlander interrupt enable
		1	Non-highlander
		0	Highlander
2	Reserved	0	Always read as 0. Writes have no effect.
1	clkfack	0-1	Clock stop fast ack enable.
0	soft reset		Software reset of USB0.
		Write 0	No action
		Write 1	Initiate software reset
		Read 0	Reset done, no action
		Read 1	Reset ongoing

24.9.2.1.3 USB0 Status Register (USB0STAT)

The USB0 status register (USB0STAT) allows the CPU to check the voltage state level of USB0_DRVVBUS pin. The USB0 status register is shown in Figure 24-67 and described in Table 24-74.

Figure 24-67. USB0 Status Register (USB0STAT)

31	1	0
Reserved		drvvbus
R-0		R-0

LEGEND: R = Read only; -*n* = value after reset

Table 24-74. USB0 Status Register (USB0STAT) Field Descriptions

Bits	Field	Description
31-1	Reserved	Always read as 0. Writes have no effect.
0	drvvbus	Current USB0_DRVVBUS value

24.9.2.1.4 USB0 IRQ_MERGED_STATUS Register (USB0IRQMSTAT)

The USB0 IRQ_MERGED_STATUS register (USB0IRQMSTAT) contains a merged status for all the events for USB0. This register optimizes software accesses in a module where a large number of events are associated to one IRQ line. The merged status is read-only, and does not need to be actively cleared like the IRQ status. It will clear automatically.

The USB0 IRQ_MERGED_STATUS register is shown in Figure 24-68 and described in Table 24-75.

Figure 24-68. USB0 IRQ_MERGED_STATUS Register (USB0IRQMSTAT)

31	1	2	1	0
Reserved			Bank1	Bank1
R-0			R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 24-75. USB0 IRQ_MERGED_STATUS Register (USB0IRQMSTAT) Field Descriptions

Bits	Field	Value	Description
31-2	Reserved	0	Always read as 0. Writes have no effect.
1	Bank1	0	No events pending from IRQ_STATUS_1
		1	At least one event is pending from IRQ_STATUS_1
0	Bank0	0	No events pending from IRQ_STATUS_0
		1	At least one event is pending from IRQ_STATUS_0

24.9.2.1.5 USB0 IRQ_EOI Register (USB0IRQEOI)

The USB0 IRQ_EOI register (USB0IRQEOI) allows the CPU to acknowledge completion of an interrupt by writing to the EOI. An eoi_write signal will be generated and another interrupt will be triggered if interrupt sources remain.

This register will be reset one cycle after it has been written to.

The USB0 IRQ_EOI register is shown in Figure 24-69 and described in Table 24-76.

Figure 24-69. USB0 IRQ_EOI Register (USB0IRQEOI)

31	1	0
Reserved		EOI for USB0
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-76. USB0 IRQ_EOI Register (USB0IRQEOI) Field Descriptions

Bits	Field	Description
31-1	Reserved	Always read as 0. Writes have no effect.
0	EOI for USB0	EOI for USB0 interrupt

24.9.2.1.6 USB0 IRQ_STATUS_RAW_0 Register (USB0IRQSTATRAW0)

The USB0 IRQ_STATUS_RAW_0 register (USB0IRQSTATRAW0) allows the USB0 interrupt sources to be manually triggered when writing a 1 to a specific bit. A read of this register returns the USB0 interrupt event pending value.

General actions per bit:
Write 0: No action
Write 1: Set event
Read 0: No event pending
Read 1: Event pending

The USB0 IRQ_STATUS_RAW_0 register is shown in Figure 24-70 and described in Table 24-77.

Figure 24-70. USB0 IRQ_STATUS_RAW_0 Register (USB0IRQSTATRAW0)

31	30	29	28	27	26	25	24
RXEP15	RXEP14	RXEP13	RXEP12	RXEP11	RXEP10	RXEP9	RXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
RXEP7	RXEP6	RXEP5	RXEP4	RXEP3	RXEP2	RXEP1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
15	14	13	12	11	10	9	8
TXEP15	TXEP14	TXEP13	TXEP12	TXEP11	TXEP10	TXEP9	TXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TXEP7	TXEP6	TXEP5	TXEP4	TXEP3	TXEP2	TXEP1	TXEP0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-77. USB0 IRQ_STATUS_RAW_0 Register (USB0IRQSTATRAW0) Field Descriptions

Bits	Field	Description
31-17	RXEP*n*	Interrupt status for RX endpoint *n*
16	Reserved	Always read 0. Writes have no effect.
15-0	TXEP*n*	Interrupt status for TX endpoint *n*

24.9.2.1.7 USB0 IRQ_STATUS_RAW_1 Register (USB0IRQSTATRAW1)

The USB0 IRQ_STATUS_RAW_1 register (USB0IRQSTATRAW1) allows the USB0 interrupt sources to be manually triggered when writing a 1 to a specific bit. A read of this register returns the USB0 interrupt event pending value.

General actions per bit:
Write 0: No action
Write 1: Set event
Read 0: No event pending
Read 1: Event pending

The USB0 IRQ_STATUS_RAW_1 register is shown in Figure 24-71 and described in Table 24-78.

Figure 24-71. USB0 IRQ_STATUS_RAW_1 Register (USB0IRQSTATRAW1)

31	30	29	28	27	26	25	24
TXFIFO15	TXFIFO14	TXFIFO13	TXFIFO12	TXFIFO11	TXFIFO10	TXFIFO9	TXFIFO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
TXFIFO7	TXFIFO6	TXFIFO5	TX FIFO4	TXFIFO3	TXFIFO2	TXFIFO1	TXFIFO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15					10	9	8
Reserved						USB[9]	USB[8]
R-0						R/W-0	R/W-0
7	6	5	4	3	2	1	0
USB[7]	USB[6]	USB[5]	USB[4]	USB[3]	USB[2]	USB[1]	USB[0]
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-78. USB0 IRQ_STATUS_RAW_1 Register (USB0IRQSTATRAW1) Field Descriptions

Bits	Field	Description
31-16	TXFIFO*n*	Interrupt status for TX FIFO endpoint *n*
15-10	Reserved	Always read 0. Writes have no effect.
9	USB[9]	Interrupt status for Mentor controller USB_INT generic interrupt
8	USB[8]	Interrupt status for DRVVBUS level change
7	USB[7]	Interrupt status for VBUS < VBUS valid threshold
6	USB[6]	Interrupt status for SRP detected
5	USB[5]	Interrupt status for device disconnected (host mode)
4	USB[4]	Interrupt status for device connected (host mode)
3	USB[3]	Interrupt status for SOF started
2	USB[2]	Interrupt status for Reset signaling detected (peripheral mode) Babble detected (host mode)
1	USB[1]	Interrupt status for Resume signaling detected
0	USB[0]	Interrupt status for Suspend signaling detected

24.9.2.1.8 USB0 IRQ_STATUS_0 Register (USB0IRQSTAT0)

The USB0 IRQ_STATUS_0 register (USB0IRQSTAT0) allows the USB0 interrupt sources to be manually cleared when writing a 1 to a specific bit. A read of this register returns the USB0 interrupt event pending value.

General actions per bit:
Write 0: No action
Write 1: Clear event
Read 0: No event pending
Read 1: Event pending

The USB0 IRQ_STATUS_0 register is shown in Figure 24-72 and described in Table 24-79.

Figure 24-72. USB0 IRQ_STATUS_0 Register (USB0IRQSTAT0)

31	30	29	28	27	26	25	24
RXEP15	RXEP14	RXEP13	RXEP12	RXEP11	RXEP10	RXEP9	RXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
RXEP7	RXEP6	RXEP5	RXEP4	RXEP3	RXEP2	RXEP1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
15	14	13	12	11	10	9	8
TXEP15	TXEP14	TXEP13	TXEP12	TXEP11	TXEP10	TXEP9	TXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TXEP7	TXEP6	TXEP5	TXEP4	TXEP3	TXEP2	TXEP1	TXEP0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-79. USB0 IRQ_STATUS_0 Register (USB0IRQSTAT0) Field Descriptions

Bits	Field	Description
31-17	RXEP*n*	Interrupt status for RX endpoint *n*
16	Reserved	Always read 0. Writes have no effect.
15-0	TXEP*n*	Interrupt status for TX endpoint *n*

24.9.2.1.9 USB0 IRQ_STATUS_1 Register (USB0IRQSTAT1)

The USB0 IRQ_STATUS_1 register (USB0IRQSTAT1) allows the USB0 interrupt sources to be manually cleared when writing a 1 to a specific bit. A read of this register returns the USB0 interrupt event pending value.

General actions per bit:
Write 0: No action
Write 1: Clear event
Read 0: No event pending
Read 1: Event pending

The USB0 IRQ_STATUS_1 register is shown in Figure 24-73 and described in Table 24-80.

Figure 24-73. USB0 IRQ_STATUS_1 Register (USB0IRQSTAT1)

31	30	29	28	27	26	25	24
TXFIFO15	TXFIFO14	TXFIFO13	TXFIFO12	TXFIFO11	TXFIFO10	TXFIFO9	TXFIFO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
TXFIFO7	TXFIFO6	TXFIFO5	TX FIFO4	TXFIFO3	TXFIFO2	TXFIFO1	TXFIFO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15					10	9	8
Reserved						USB[9]	USB[8]
R-0						R/W-0	R/W-0
7	6	5	4	3	2	1	0
USB[7]	USB[6]	USB[5]	USB[4]	USB[3]	USB[2]	USB[1]	USB[0]
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-80. USB0 IRQ_STATUS_1 Register (USB0IRQSTAT1) Field Descriptions

Bits	Field	Description
31-16	TXFIFO*n*	Interrupt status for TX FIFO endpoint *n*
15-10	Reserved	Always read 0. Writes have no effect.
9	USB[9]	Interrupt status for Mentor controller USB_INT generic interrupt
8	USB[8]	Interrupt status for DRVVBUS level change
7	USB[7]	Interrupt status for VBUS < VBUS valid threshold
6	USB[6]	Interrupt status for SRP detected
5	USB[5]	Interrupt status for device disconnected (host mode)
4	USB[4]	Interrupt status for device connected (host mode)
3	USB[3]	Interrupt status for SOF started
2	USB[2]	Interrupt status for Reset signaling detected (peripheral mode) Babble detected (host mode)
1	USB[1]	Interrupt status for Resume signaling detected
0	USB[0]	Interrupt status for Suspend signaling detected

24.9.2.1.10 USB0 IRQ_ENABLE_SET_0 Register (USB0IRQENABLESET0)

The USB0 IRQ_ENABLE_SET_0 register (USB0IRQENABLESET0) allows the USB0 interrupt sources to be manually enabled when writing a 1 to a specific bit. A read of this register returns the USB0 interrupt enabled value. General actions per bit:
Write 0: No action
Write 1: Interrupt enabled
Read 0: Interrupt disabled
Read 1: Interrupt enabled

The USB0 IRQ_ENABLE_SET_0 register is shown in Figure 24-74 and described in Table 24-81.

Figure 24-74. USB0 IRQ_ENABLE_SET_0 Register (USB0IRQENABLESET0)

31	30	29	28	27	26	25	24
RXEP15	RXEP14	RXEP13	RXEP12	RXEP11	RXEP10	RXEP9	RXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
RXEP7	RXEP6	RXEP5	RXEP4	RXEP3	RXEP2	RXEP1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
15	14	13	12	11	10	9	8
TXEP15	TXEP14	TXEP13	TXEP12	TXEP11	TXEP10	TXEP9	TXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TXEP7	TXEP6	TXEP5	TXEP4	TXEP3	TXEP2	TXEP1	TXEP0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-81. USB0 IRQ_ENABLE_SET_0 Register (USB0IRQENABLESET0) Field Descriptions

Bits	Field	Description
31-17	RXEP*n*	Interrupt enable for RX endpoint *n*
16	Reserved	Always read 0. Writes have no effect.
15-0	TXEP*n*	Interrupt enable for TX endpoint *n*

24.9.2.1.11 USB0 IRQ_ENABLE_SET_1 Register (USB0IRQENABLESET1)

The USB0 IRQ_ENABLE_SET_1 register (USB0IRQENABLESET1) allows the USB0 interrupt sources to be manually enabled when writing a 1 to a specific bit. A read of this register returns the USB0 interrupt enabled value. General actions per bit:
Write 0: No action
Write 1: Interrupt enabled
Read 0: Interrupt disabled
Read 1: Interrupt enabled

The USB0 IRQ_ENABLE_SET_1 register is shown in Figure 24-75 and described in Table 24-82.

Figure 24-75. USB0 IRQ_ENABLE_SET_1 Register (USB0IRQENABLESET1)

31	30	29	28	27	26	25	24
TXFIFO15	TXFIFO14	TXFIFO13	TXFIFO12	TXFIFO11	TXFIFO10	TXFIFO9	TXFIFO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
TXFIFO7	TXFIFO6	TXFIFO5	TX FIFO4	TXFIFO3	TXFIFO2	TXFIFO1	TXFIFO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15					10	9	8
Reserved						USB[9]	USB[8]
R-0						R/W-0	R/W-0
7	6	5	4	3	2	1	0
USB[7]	USB[6]	USB[5]	USB[4]	USB[3]	USB[2]	USB[1]	USB[0]
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-82. USB0 IRQ_ENABLE_SET_1 Register (USB0IRQENABLESET1) Field Descriptions

Bits	Field	Description
31-16	TXFIFO*n*	Interrupt enable for TX FIFO endpoint *n*
15-10	Reserved	Always read 0. Writes have no effect.
9	USB[9]	Interrupt enable for Mentor controller USB_INT generic interrupt
8	USB[8]	Interrupt enable for DRVVBUS level change
7	USB[7]	Interrupt enable for VBUS < VBUS valid threshold
6	USB[6]	Interrupt enable for SRP detected
5	USB[5]	Interrupt enable for device disconnected (host mode)
4	USB[4]	Interrupt enable for device connected (host mode)
3	USB[3]	Interrupt enable for SOF started
2	USB[2]	Interrupt enable for Reset signaling detected (peripheral mode) Babble detected (host mode)
1	USB[1]	Interrupt enable for Resume signaling detected
0	USB[0]	Interrupt enable for Suspend signaling detected

24.9.2.1.12 USB0 IRQ_ENABLE_CLR_0 Register (USB0IRQENABLECLR0)

The USB0 IRQ_ENABLE_CLR_0 register (USB0IRQENABLECLR0) allows the USB0 interrupt sources to be manually disabled when writing a 1 to a specific bit. A read of this register returns the USB0 interrupt enabled value. General actions per bit:
Write 0: No action
Write 1: Disable interrupt
Read 0: Interrupt disabled
Read 1: Interrupt enabled

The USB0 IRQ_ENABLE_CLR_0 register is shown in Figure 24-76 and described in Table 24-83.

Figure 24-76. USB0 IRQ_ENABLE_CLR_0 Register (USB0IRQENABLECLR0)

31	30	29	28	27	26	25	24
RXEP15	RXEP14	RXEP13	RXEP12	RXEP11	RXEP10	RXEP9	RXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
RXEP7	RXEP6	RXEP5	RXEP4	RXEP3	RXEP2	RXEP1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
15	14	13	12	11	10	9	8
TXEP15	TXEP14	TXEP13	TXEP12	TXEP11	TXEP10	TXEP9	TXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TXEP7	TXEP6	TXEP5	TXEP4	TXEP3	TXEP2	TXEP1	TXEP0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-83. USB0 IRQ_ENABLE_CLR_0 Register (USB0IRQENABLECLR0) Field Descriptions

Bits	Field	Description
31-17	RXEP*n*	Interrupt enable for RX endpoint *n*
16	Reserved	Always read 0. Writes have no effect.
15-0	TXEP*n*	Interrupt enable for TX endpoint *n*

24.9.2.1.13 USB0 IRQ_ENABLE_CLR_1 Register (USB0IRQENABLECLR1)

The USB0 IRQ_ENABLE_CLR_1 register (USB0IRQENABLECLR1) allows the USB0 interrupt sources to be manually disabled when writing a 1 to a specific bit. A read of this register returns the USB0 interrupt enabled value. General actions per bit:
Write 0: No action
Write 1: Disable interrupt
Read 0: Interrupt disabled
Read 1: Interrupt enabled

The USB0 IRQ_ENABLE_CLR_1 register is shown in Figure 24-77 and described in Table 24-84.

Figure 24-77. USB0 IRQ_ENABLE_SET_1 Register (USB0IRQENABLECLR1)

31	30	29	28	27	26	25	24
TXFIFO15	TXFIFO14	TXFIFO13	TXFIFO12	TXFIFO11	TXFIFO10	TXFIFO9	TXFIFO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
TXFIFO7	TXFIFO6	TXFIFO5	TX FIFO4	TXFIFO3	TXFIFO2	TXFIFO1	TXFIFO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15					10	9	8
Reserved						USB[9]	USB[8]
R-0						R/W-0	R/W-0
7	6	5	4	3	2	1	0
USB[7]	USB[6]	USB[5]	USB[4]	USB[3]	USB[2]	USB[1]	USB[0]
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-84. USB0 IRQ_ENABLE_CLR_1 Register (USB0IRQENABLECLR1) Field Descriptions

Bits	Field	Description
31-16	TXFIFO*n*	Interrupt enable for TX FIFO endpoint *n*
15-10	Reserved	Always read 0. Writes have no effect.
9	USB[9]	Interrupt enable for Mentor controller USB_INT generic interrupt
8	USB[8]	Interrupt enable for DRVVBUS level change
7	USB[7]	Interrupt enable for VBUS < VBUS valid threshold
6	USB[6]	Interrupt enable for SRP detected
5	USB[5]	Interrupt enable for device disconnected (host mode)
4	USB[4]	Interrupt enable for device connected (host mode)
3	USB[3]	Interrupt enable for SOF started
2	USB[2]	Interrupt enable for Reset signaling detected (peripheral mode) Babble detected (host mode)
1	USB[1]	Interrupt enable for Resume signaling detected
0	USB[0]	Interrupt enable for Suspend signaling detected

24.9.2.1.14 USB0 Tx Mode Register (USB0TXMODE)

The USB0 Tx mode register (USB0TXMODE) allows the CPU to individually enable RNDIS/Generic/CDC modes for each endpoint. Using the global RNDIS enable (rndis) bit in the control register (USB0CTRL) overrides this register and enables the RNDIS mode for all endpoints.

The USB0 Tx mode register is shown in Figure 24-78 and described in Table 24-85.

Figure 24-78. USB0 Tx Mode Register (USB0TXMODE)

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Reserved	Tx15_mode	Tx14_mode	Tx13_mode	Tx12_mode	Tx11_mode	Tx10_mode	Tx9_mode
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Tx8_mode	Tx7_mode	Tx6_mode	Tx5_mode	Tx4_mode	Tx3_mode	Tx2_mode	Tx1_mode
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-85. USB0 Tx Mode Register (USB0TXMODE) Field Descriptions

Bits	Field	Value	Description
31-30	Reserved	0	Always read as 0. Writes have no effect.
29-28	Tx15_mode		TX endpoint 15 mode.
		0	Transparent mode on TX endpoint 15
		1h	RNDIS MODE on TX endpoint 15
		2h	CDC mode on TX endpoint 15
		3h	Generic RNDIS mode on TX endpoint 15
27-26	Tx14_mode		TX endpoint 14 mode.
		0	Transparent mode on TX endpoint 14
		1h	RNDIS MODE on TX endpoint 14
		2h	CDC mode on TX endpoint 14
		3h	Generic RNDIS mode on TX endpoint 14
25-24	Tx13_mode		TX endpoint 13 mode.
		0	Transparent mode on TX endpoint 13
		1h	RNDIS MODE on TX endpoint 13
		2h	CDC mode on TX endpoint 13
		3h	Generic RNDIS mode on TX endpoint 13
23-22	Tx12_mode		TX endpoint 12 mode.
		0	Transparent mode on TX endpoint 12
		1h	RNDIS MODE on TX endpoint 12
		2h	CDC mode on TX endpoint 12
		3h	Generic RNDIS mode on TX endpoint 12
21-20	Tx11_mode		TX endpoint 11 mode.
		0	Transparent mode on TX endpoint 11
		1h	RNDIS MODE on TX endpoint 11
		2h	CDC mode on TX endpoint 11
		3h	Generic RNDIS mode on TX endpoint 11
19-18	Tx10_mode		TX endpoint 10 mode.
		0	Transparent mode on TX endpoint 10
		1h	RNDIS MODE on TX endpoint 10
		2h	CDC mode on TX endpoint 10
		3h	Generic RNDIS mode on TX endpoint 10

Table 24-85. USB0 Tx Mode Register (USB0TXMODE) Field Descriptions (continued)

Bits	Field	Value	Description
17-16	Tx9_mode		TX endpoint 9 mode.
		0	Transparent mode on TX endpoint 9
		1h	RNDIS MODE on TX endpoint 9
		2h	CDC mode on TX endpoint 9
		3h	Generic RNDIS mode on TX endpoint 9
15-14	Tx8_mode		TX endpoint 8 mode.
		0	Transparent mode on TX endpoint 8
		1h	RNDIS MODE on TX endpoint 8
		2h	CDC mode on TX endpoint 8
		3h	Generic RNDIS mode on TX endpoint 8
13-12	Tx7_mode		TX endpoint 7 mode.
		0	Transparent mode on TX endpoint 7
		1h	RNDIS MODE on TX endpoint 7
		2h	CDC mode on TX endpoint 7
		3h	Generic RNDIS mode on TX endpoint 7
11-10	Tx6_mode		TX endpoint 6 mode.
		0	Transparent mode on TX endpoint 6
		1h	RNDIS MODE on TX endpoint 6
		2h	CDC mode on TX endpoint 6
		3h	Generic RNDIS mode on TX endpoint 6
9-8	Tx5_mode		TX endpoint 5 mode.
		0	Transparent mode on TX endpoint 5
		1h	RNDIS MODE on TX endpoint 5
		2h	CDC mode on TX endpoint 5
		3h	Generic RNDIS mode on TX endpoint 5
7-6	Tx4_mode		TX endpoint 4 mode.
		0	Transparent mode on TX endpoint 4
		1h	RNDIS MODE on TX endpoint 4
		2h	CDC mode on TX endpoint 4
		3h	Generic RNDIS mode on TX endpoint 4
5-4	Tx3_mode		TX endpoint 3 mode.
		0	Transparent mode on TX endpoint 3
		1h	RNDIS MODE on TX endpoint 3
		2h	CDC mode on TX endpoint 3
		3h	Generic RNDIS mode on TX endpoint 3
3-2	Tx2_mode		TX endpoint 2 mode.
		0	Transparent mode on TX endpoint 2
		1h	RNDIS MODE on TX endpoint 2
		2h	CDC mode on TX endpoint 2
		3h	Generic RNDIS mode on TX endpoint 2
1-0	Tx1_mode		TX endpoint 1 mode.
		0	Transparent mode on TX endpoint 1
		1h	RNDIS MODE on TX endpoint 1
		2h	CDC mode on TX endpoint 1
		3h	Generic RNDIS mode on TX endpoint 1

24.9.2.1.15 USB0 Rx Mode Register (USB0RXMODE)

The USB0 Rx mode register (USB0RXMODE) allows the CPU to individually enable RNDIS/Generic/CDC modes for each endpoint. Using the global RNDIS enable (rndis) bit in the control register (USB0CTRL) overrides this register and enables the RNDIS mode for all endpoints.

The USB0 Rx mode register is shown in Figure 24-79 and described in Table 24-86.

Figure 24-79. USB0 Rx Mode Register (USB0RXMODE)

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Reserved	Rx15_mode	Rx14_mode	Rx13_mode	Rx12_mode	Rx11_mode	Rx10_mode	Rx9_mode
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	**13 12**	**11 10**	**9 8**	**7 6**	**5 4**	**3 2**	**1 0**
Rx8_mode	Rx7_mode	Rx6_mode	Rx5_mode	Rx4_mode	Rx3_mode	Rx2_mode	Rx1_mode
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-86. USB0 Rx Mode Register (USB0RXMODE) Field Descriptions

Bits	Field	Value	Description
31-30	Reserved	0	Always read as 0. Writes have no effect.
29-28	Rx15_mode		RX endpoint 15 mode.
		0	Transparent mode on RX endpoint 15
		1h	RNDIS MODE on RX endpoint 15
		2h	CDC mode on RX endpoint 15
		3h	Generic RNDIS mode on RX endpoint 15
27-26	Rx14_mode		RX endpoint 14 mode.
		0	Transparent mode on RX endpoint 14
		1h	RNDIS MODE on RX endpoint 14
		2h	CDC mode on RX endpoint 14
		3h	Generic RNDIS mode on RX endpoint 14
25-24	Rx13_mode		RX endpoint 13 mode.
		0	Transparent mode on RX endpoint 13
		1h	RNDIS MODE on RX endpoint 13
		2h	CDC mode on RX endpoint 13
		3h	Generic RNDIS mode on RX endpoint 13
23-22	Rx12_mode		RX endpoint 12 mode.
		0	Transparent mode on RX endpoint 12
		1h	RNDIS MODE on RX endpoint 12
		2h	CDC mode on RX endpoint 12
		3h	Generic RNDIS mode on RX endpoint 12
21-20	Rx11_mode		RX endpoint 11 mode.
		0	Transparent mode on RX endpoint N+10
		1h	RNDIS MODE on RX endpoint N+10
		2h	CDC mode on RX endpoint N+10
		3h	Generic RNDIS mode on RX endpoint N+10
19-18	Rx10_mode		RX endpoint 10 mode.
		0	Transparent mode on RX endpoint 10
		1h	RNDIS MODE on RX endpoint 10
		2h	CDC mode on RX endpoint 10
		3h	Generic RNDIS mode on RX endpoint 10

Table 24-86. USB0 Rx Mode Register (USB0RXMODE) Field Descriptions (continued)

Bits	Field	Value	Description
17-16	Rx9_mode		RX endpoint 9 mode.
		0	Transparent mode on RX endpoint 9
		1h	RNDIS MODE on RX endpoint 9
		2h	CDC mode on RX endpoint 9
		3h	Generic RNDIS mode on RX endpoint 9
15-14	Rx8_mode		RX endpoint 8 mode.
		0	Transparent mode on RX endpoint 8
		1h	RNDIS MODE on RX endpoint 8
		2h	CDC mode on RX endpoint 8
		3h	Generic RNDIS mode on RX endpoint 8
13-12	Rx7_mode		RX endpoint 7 mode.
		0	Transparent mode on RX endpoint 7
		1h	RNDIS MODE on RX endpoint 7
		2h	CDC mode on RX endpoint 7
		3h	Generic RNDIS mode on RX endpoint 7
11-10	Rx6_mode		RX endpoint 6 mode.
		0	Transparent mode on RX endpoint 6
		1h	RNDIS MODE on RX endpoint 6
		2h	CDC mode on RX endpoint 6
		3h	Generic RNDIS mode on RX endpoint 6
9-8	Rx5_mode		RX endpoint 5 mode.
		0	Transparent mode on RX endpoint 5
		1h	RNDIS MODE on RX endpoint 5
		2h	CDC mode on RX endpoint 5
		3h	Generic RNDIS mode on RX endpoint 5
7-6	Rx4_mode		RX endpoint 4 mode.
		0	Transparent mode on RX endpoint 4
		1h	RNDIS MODE on RX endpoint 4
		2h	CDC mode on RX endpoint 4
		3h	Generic RNDIS mode on RX endpoint 4
5-4	Rx3_mode		RX endpoint 3 mode.
		0	Transparent mode on RX endpoint 3
		1h	RNDIS MODE on RX endpoint 3
		2h	CDC mode on RX endpoint 3
		3h	Generic RNDIS mode on RX endpoint 3
3-2	Rx2_mode		RX endpoint 2 mode.
		0	Transparent mode on RX endpoint 2
		1h	RNDIS MODE on RX endpoint 2
		2h	CDC mode on RX endpoint 2
		3h	Generic RNDIS mode on RX endpoint 2
1-0	Rx1_mode		RX endpoint 1 mode.
		0	Transparent mode on RX endpoint 1
		1h	RNDIS MODE on RX endpoint 1
		2h	CDC mode on RX endpoint 1
		3h	Generic RNDIS mode on RX endpoint 1

24.9.2.1.16 USB0 Generic RNDIS EPn Size Register (USB0GENRNDISEPn)

The USB0 generic RNDIS EPn size register (USB0GENRNDISEPn) is programmed with a RNDIS packet size in bytes. When EP(n) is in Generic RNDIS mode, the received USB packets will be collected into a single CPPI packet that is completed when the number of bytes equal to the value of this register has been received, or a “short” packet is received. Note that this register must be programmed with a value that is an integer multiple of the endpoint size. The maximum value this register can be programmed with is 10000h, or 65536. N can range from 1 to 15.

The USB0 generic RNDIS EPn size register is shown in Figure 24-80 and described in Table 24-87.

Figure 24-80. USB0 Generic RNDIS EPn Size Register (USB0GENRNDISEPn)

31	17	16	0
Reserved		Ep(n)_size	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-87. USB0 Generic RNDIS EPn Size Register (USB0GENRNDISEPn) Field Descriptions

Bits	Field	Description
31-17	Reserved	Always read as 0. Writes have no effect.
16-0	Ep(n)_size	Generic RNDIS packet size. n = 1-15

24.9.2.1.17 USB0 Auto Req Register (USB0AUTOREQ)

The USB0 auto req register (USB0AUTOREQ) allows the CPU to enable an automatic IN token request generation for host mode RX operation per each RX endpoint. This features has the DMA set the ReqPkt bit in the RXCSR when it clears the RxPktRdy bit after reading out a packet. The ReqPkt bit is used by the core to generate an IN token to receive data. By using this feature, the host can automatically generate an IN token after the DMA finishes receiving data and empties an endpoint buffer, thus receiving the next data packet as soon as possible from the connected device. Without this feature, the CPU will have to manually set the ReqPkt bit for every USB packet.

There are two modes that Auto Req can function: always or all except an EOP. The always mode will set the ReqPkt bit after every USB packet the DMA receives thus generating a new IN token after each USB packet. The EOP mode will set the ReqPkt bit after every USB packet that isn’t an EOP (end of packet) in the CPPI descriptor. For RNDIS, CDC, and Generic RNDIS modes the auto req will stop when the EOP is received (either via a short packet for RNDIS,CDC, and Generic RNDIS or the count is reached for Generic RNDIS), making it useful for starting a large RNDIS packet and having it auto generate IN tokens until the end of the RNDIS packet. For transparent mode every USB packet is an EOP CPPI packet so the auto req never functions and acts like auto req is disabled.

The USB0 auto req register is shown in Figure 24-81 and described in Table 24-88.

Figure 24-81. USB0 Auto Req Register (USB0AUTOREQ)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		Rx15_autoreq		Rx14_autoreq		Rx13_autoreq		Rx12_autoreq		Rx11_autoreq		Rx10_autoreq		Rx9_autoreq	
R-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rx8_autoreq		Rx7_autoreq		Rx6_autoreq		Rx5_autoreq		Rx4_autoreq		Rx3_autoreq		Rx2_autoreq		Rx1_autoreq	
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-88. USB0 Auto Req Register (USB0AUTOREQ) Field Descriptions

Bits	Field	Values	Description
31-30	Reserved	0	Always read as 0. Writes have no effect.
29-28	Rx15_autoreq		RX endpoint 15 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
27-26	Rx14_autoreq		RX endpoint 14 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
25-24	Rx13_autoreq		RX endpoint 13 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
23-22	Rx12_autoreq		RX endpoint 12 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
21-20	Rx11_autoreq		RX endpoint 11 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
19-18	Rx10_autoreq		RX endpoint 10 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
17-16	Rx9_autoreq		RX endpoint 9 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always

Table 24-88. USB0 Auto Req Register (USB0AUTOREQ) Field Descriptions (continued)

Bits	Field	Values	Description
15-14	Rx8_autoreq		RX endpoint 8 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
13-12	Rx7_autoreq		RX endpoint 7 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
11-10	Rx6_autoreq		RX endpoint 6 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
9-8	Rx5_autoreq		RX endpoint 5 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
7-6	Rx4_autoreq		RX endpoint 4 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
5-4	Rx3_autoreq		RX endpoint 3 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
3-2	Rx2_autoreq		RX endpoint 2 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
1-0	Rx1_autoreq		RX endpoint 1 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always

24.9.2.1.18 USB0 SRP Fix Time Register (USB0SRPFIXTIME)

The USB0 SRP fix time register (USB0SRPFIXTIME) allows the CPU to configure the maximum amount of time the SRP fix logic blocks the AVAID from the PHY to the OTG core. This time allows the VBUS signal the ability to get below the thresholds and therefore remove the chance of voltage bounces which could give false threshold measurements.

The USB0 SRP fix time register is shown in Figure 24-82 and described in Table 24-89.

Figure 24-82. USB0 SRP Fix Time Register (USB0SRPFIXTIME)

31	0
srpfixtime	
R/W-280 DE80h	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 24-89. USB0 SRP Fix Time Register (USB0SRPFIXTIME) Field Descriptions

Bits	Field	Description
31-0	srpfixtime	SRP Fix maximum time in 60 MHz cycles. Default is 700 ms.

24.9.2.1.19 USB0 Teardown Register (USB0TDOWN)

The USB0 teardown register (USB0TDOWN) controls the tearing down of Rx and Tx FIFOs in the USB controller. When a 1 is written to a valid bit in this register, the CPPI FIFO pointers for that endpoint are cleared, and the register automatically clears itself after 1 clock cycle. This register must be used in conjunction with the CPPI DMA Teardown mechanism. The host should also write the FlushFIFO bits in the TXCSR and RXCSR Mentor USB Controller registers to ensure a complete teardown of the endpoint (see the Mentor specification for details).

The USB0 teardown register is shown in Figure 24-83 and described in Table 24-90.

Figure 24-83. USB0 Teardown Register (USB0TDOWN)

31	17	16	15	1	0
tx_tdown[*n*]		Rsvd	rx_tdown[*n*]		Rsvd
R/W-0		R-0	R/W-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-90. USB0 Teardown Register (USB0TDOWN) Field Descriptions

Bits	Field	Value	Description
31-17	tx_tdown[*n*]		Tx endpoint teardown. Write 1 to corresponding bit *n* to set. Read as 0.
		Bit 31	Endpoint 15
		...	...
		Bit 17	Endpoint 1
16	Reserved	0	Always read as 0. Writes have no effect.
15-1	rx_tdown[*n*]		RX endpoint teardown. Write 1 to corresponding bit *n* to set. Read as 0.
		Bit 15	Endpoint 15
		...	...
		Bit 1	Endpoint 1
0	Reserved	0	Always read as 0. Writes have no effect.

24.9.2.1.20 USB0 PHY UTMI Register (USB0UTMI)

The USB0 PHY UTMI register (USB0UTMI) controls various PHY UTMI signals. The UTMI interface is located between the PHY module and the Mentor Graphics Controller. These signals are inputs to the PHY but are not outputs from the Mentor Graphics Controller.

It is important to set the otgdisable bit in to a high value to operate the PHY in a non-OTG mode. The USB0 PHY UTMI register is shown in Figure 24-84 and described in Table 24-91.

Figure 24-84. USB0 PHY UTMI Register (USB0UTMI)

31							24
Reserved							
R-0							
23	22	21	20	19	18	17	16
txbitstuffen	txbitstuffenh	otgdisable	vbusvldextsel	vbusvldext	txenablen	fsxcvrowner	txvalidh
R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15							8
datainh							
R/W-0							
7				3	2	1	0
Reserved					wordinterface	fsdataext	fsse0ext
R-0					R/W-0	R/W-1	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-91. USB0 PHY UTMI Register (USB0UTMI) Field Descriptions

Bits	Field	Description
31-24	Reserved	Always read as 0. Writes have no effect.
23	txbitstuffen	PHY UTMI input for signal txbitstuffen
22	txbitstuffenh	PHY UTMI input for signal txbitstuffenh
21	otgdisable	PHY UTMI input for signal otgdisable
20	vbusvldextsel	PHY UTMI input for signal vbusvldextsel
19	vbusvldext	PHY UTMI input for signal vbusvldext
18	txenablen	PHY UTMI input for signal txenablen
17	fsxcvrowner	PHY UTMI input for signal fsxcvrowner
16	txvalidh	PHY UTMI input for signal txvalidh
15-8	datainh	PHY UTMI input for signal datainh
7-3	Reserved	Always read as 0. Writes have no effect.
2	wordinterface	PHY UTMI input for signal wordinterface
1	fsdataext	PHY UTMI input for signal fsdataext
0	fsse0ext	PHY UTMI input for signal fsse0ext

24.9.2.1.21 USB0 MGC UTMI Loopback Register (USB0UTMILB)

The USB0 MGC UTMI loopback register (USB0UTMILB) contains most of the input and output UTMI signals for the Mentor Graphics Controller. The UTMI interface is located between the PHY module and the Mentor Graphics Controller.

In the loopback mode, test register bits 11 to 0 control various UTMI signals that are input to the Mentor Graphics Controller.

In the loopback mode, test register bits 28 to 16 observe various UTMI signals that are output from the Mentor Graphics Controller.

The USB0 MGC UTMI loopback register is shown in Figure 24-85 and described in Table 24-92.

Figure 24-85. USB0 MGC UTMI Loopback Register (USB0UTMILB)

31	30	29	28	27	26	25	24
Reserved			suspendm	suspendm		txvalid	xcvrsel
R-0			R-0	R-0		R-0	R-0
23	22	21	20	19	18	17	16
xcvrsel	termsel	drvvbus	chrgvbus	dischrgvbus	dppulldown	dmpulldown	idpullup
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
Reserved				iddig	hostdiscon	sessend	avalid
R-0				R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
vbusvalid	rxerror	Reserved		linestate		Reserved	
R/W-0	R/W-0	R-0		R/W-1		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-92. USB0 MGC UTMI Loopback Register (USB0UTMILB) Field Descriptions

Bits	Field	Description
31-29	Reserved	Always read as 0. Writes have no effect.
28	suspendm	Loopback test observed value for suspendm
27-26	opmode	Loopback test observed value for opmode
25	txvalid	Loopback test observed value for txvalid
24-23	xcvrsel	Loopback test observed value for xcvrsel
22	termsel	Loopback test observed value for termsel
21	drvvbus	Loopback test observed value for drvvbus
20	chrgvbus	Loopback test observed value for chrgvbus
19	dischrgvbus	Loopback test observed value for dischrgvbus
18	dppulldown	Loopback test observed value for dppulldown
17	dmpulldown	Loopback test observed value for dmpulldown
16	idpullup	Loopback test observed value for idpullup
15-12	Reserved	Always read as 0. Writes have no effect.
11	iddig	Loopback test value for iddig
10	hostdiscon	Loopback test value for hostdiscon
9	sessend	Loopback test value for sessend
8	avalid	Loopback test value for avalid
7	vbusvalid	Loopback test value for vbusvalid
6	rxerror	Loopback test value for rxerror
5-4	Reserved	Always read as 0. Writes have no effect.
3-2	linestate	Loopback test value for linestate
1-0	Reserved	Always read as 0. Writes have no effect.

24.9.2.1.22 USB0 Mode Register (USB0MODE)

The USB0 mode register (USB0MODE) supports operating the PHY in a non-OTG mode. This requires the user to set the MGC UTMI input signal iddig. OTG interfaces have an id external pin which control UTMI signal iddig. Since this external pin is not available, the user should set iddig to either a 0 (A-type) or 1 (B-type). This value will be the initial setting for the Mentor controller. But, the controller will determine whether it is operating as a host or device via its protocols. To determine the function of the controller this information can be found by reading the Mentor Controller DEVCTL register (80h) bit 2.

The loopback bit enables the loopback test. This test allows MGC0 to be connected to MGC1. It is important to set both loopback bits in both USB0/1 Mode registers. The USB0 MGC UTMI loopback register contains the various UTMI signals to be controlled and observed during loopback test.

The phy_test bit enables the phy_test mode. This test mode is intended to allow additional control of the UTMI inputs to the PHY. Currently, these inputs are drvvbus, dppulldown, dmpulldown, and idpullup. When phy_test is high, then the pin inputs for these signals control the inputs to the PHY instead of the Mentor controller outputs. The phy_test mode is not active with loopback mode is active.

When phy_test is active than the PHY inputs datainh is equal to datain. And txvalidh is equal to txvalid.

The USB0 mode register is shown in Figure 24-86 and described in Table 24-93.

Figure 24-86. USB0 Mode Register (USB0MODE)

31	16
Reserved	
R-0	

15	9	8	7	6	2	1	0
Reserved		iddig	Rsvd	Reserved		phy_test	loopback
R-0		R/W-0	R/W-0	R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-93. USB0 Mode Register (USB0MODE) Field Descriptions

Bits	Field	Value	Description
31-9	Reserved	0	Always read as 0. Writes have no effect.
8	iddig		MGC input value for iddig
		0	A type
		1	B type
7	Reserved	0-1	For a PG1.x device, read as 0. For a PG2.x device, a 1 must be written to this bit.
6-2	Reserved	0	Always read as 0. Writes have no effect.
1	phy_test		PHY test
		0	Normal mode
		1	PHY test mode
0	loopback		Loopback test mode
		0	Normal mode
		1	Loopback test mode

24.9.2.1.23 USB0 Mentor Core Registers

A description of the Mentor core registers is available in Section 24.9.6. Two instantiations of the USB core exists, USB0 and USB1. These registers reside back to back within USB0 space. The core registers that are used by USB0 subsystem are defined within offsets 1400h-159Ch, while the core registers that are used by USB1 subsystem are defined within offset 1C00h-1D9Ch.

24.9.2.2 USB1 Controller Registers

Table 24-94 lists the registers for the USB1 controller submodule.

Table 24-94. USB1 Controller Registers

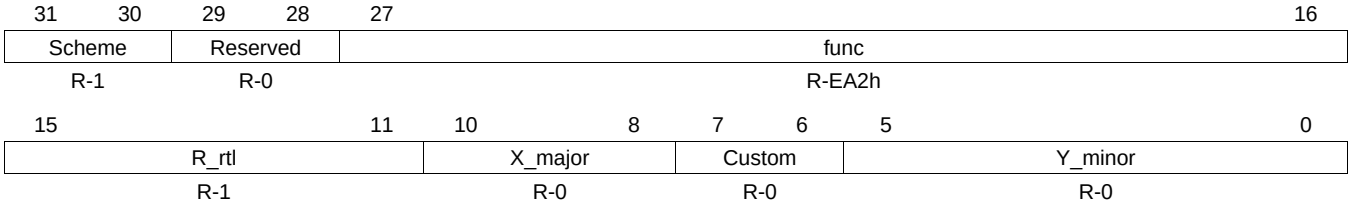
Address Offset	Acronym	USB1 Controller Register
1800h	USB1REV	USB1 Revision Register
1814h	USB1CTRL	USB1 Control Register
1818h	USB1STAT	USB1 Status Register
1820h	USB1IRQMSTAT	USB1 IRQ_MERGED_STATUS
1824h	USB1IRQEOI	USB1 IRQ_EOI
1828h	USB1IRQSTATRAW0	USB1 IRQ_STATUS_RAW_0
182Ch	USB1IRQSTATRAW1	USB1 IRQ_STATUS_RAW_1
1830h	USB1IRQSTAT0	USB1 IRQ_STATUS_0
1834h	USB1IRQSTAT1	USB1 IRQ_STATUS_1
1838h	USB1IRQENABLESET0	USB1 IRQ_ENABLE_SET_0
183Ch	USB1IRQENABLESET1	USB1 IRQ_ENABLE_SET_1
1840h	USB1IRQENABLECLR0	USB1 IRQ_ENABLE_CLR_0
1844h	USB1IRQENABLECLR1	USB1 IRQ_ENABLE_CLR_1
1870h	USB1TXMODE	USB1 Tx Mode Register
1874h	USB1RXMODE	USB1 Rx Mode Register
1880h	USB1GENRNDISEP1	USB1 Generic RNDIS Size EP1
1884h	USB1GENRNDISEP2	USB1 Generic RNDIS Size EP2
1888h	USB1GENRNDISEP3	USB1 Generic RNDIS Size EP3
188Ch	USB1GENRNDISEP4	USB1 Generic RNDIS Size EP4
1890h	USB1GENRNDISEP5	USB1 Generic RNDIS Size EP5
1894h	USB1GENRNDISEP6	USB1 Generic RNDIS Size EP6
1898h	USB1GENRNDISEP7	USB1 Generic RNDIS Size EP7
189Ch	USB1GENRNDISEP8	USB1 Generic RNDIS Size EP8
18A0h	USB1GENRNDISEP9	USB1 Generic RNDIS Size EP9
18A4h	USB1GENRNDISEP10	USB1 Generic RNDIS Size EP10
18A8h	USB1GENRNDISEP11	USB1 Generic RNDIS Size EP11
18ACh	USB1GENRNDISEP12	USB1 Generic RNDIS Size EP12
18B0h	USB1GENRNDISEP13	USB1 Generic RNDIS Size EP13
18B4h	USB1GENRNDISEP14	USB1 Generic RNDIS Size EP14
18B8h	USB1GENRNDISEP15	USB1 Generic RNDIS Size EP15
18D0h	USB1AUTOREQ	USB1 Auto Req Register
18D4h	USB1SRPFIXTIME	USB1 SRP Fix Time
18D8h	USB1TDOWN	USB1 Teardown Register
18E0h	USB1UTMI	USB1 PHY UTMI Register
18E4h	USB1UTMILB	USB1 MGC UTMI Loopback Register
18E8h	USB1MODE	USB1 Mode Register
1C00h–1D9Ch	...	USB1 Mentor Core Registers

24.9.2.2.1 USB1 Revision Register (USB1REV)

The USB1 revision register (USB1REV) contains the major and minor revisions for the USB1 module.

The USB1 revision register is shown in Figure 24-87 and described in Table 24-95.

Figure 24-87. USB1 Revision Register (USB1REV)



LEGEND: R = Read only; -*n* = value after reset

Table 24-95. USB1 Revision Register (USB1REV) Field Descriptions

Bits	Field	Value	Description
31-30	Scheme	0-3h	Used to distinguish between old scheme and current.
29-28	Reserved	0	Reserved
27-16	func	0-FFFh	Function indicates a software compatible module family.
15-11	R_rtl	0-1Fh	RTL revision.
10-8	X_major	0-7h	Major revision.
7-6	Custom	0-3h	Custom revision
5-0	Y_minor	0-3Fh	Minor revision.

24.9.2.2.2 USB1 Control Register (USB1CTRL)

The USB1 control register (USB1CTRL) allows the CPU to control various aspects of the module. If uint is set high, then the Mentor controller generic interrupt for USB[9] will be generated (if enabled). This requires S/W to read the Mentor controller's registers to determine which interrupt USB[0] to USB[7] occurred. If uint is set low, then the usb20otg_f module will automatically read the Mentor controller's registers and set the appropriate interrupt USB[0] to USB[7] (if enabled). The generic interrupt for USB[9] will not be generated.

The USB1 control register is shown in Figure 24-88 and described in Table 24-96.

Figure 24-88. USB1 Control Register (USB1CTRL)

31	30	29	16
dis_deb	dis_srp	Reserved	
R/W-0	R/W-0	R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved		soft reset isolation	rndis	uint	Reserved	clkfack	soft reset
R-0		R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-96. USB1 Control Register (USB1CTRL) Field Descriptions

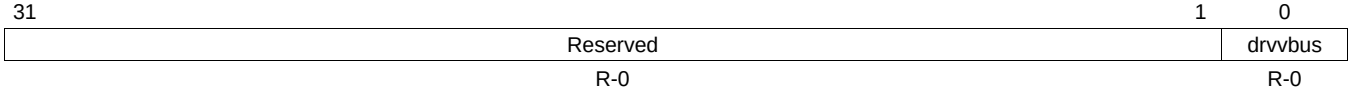
Bits	Field	Value	Description
31	dis_deb	0-1	Disable the VBUS debouncer circuit fix
30	dis_srp	0-1	Disable the OTG Session Request Protocol (SRP) AVALID circuit fix. When enabled (=0), this allows additional time for the VBUS signal to be measured against the VBUS thresholds. The time is specified in the USB1 SRP Fix Time Register.
29-6	Reserved	0	Always read as 0. Writes have no effect.
5	soft reset isolation	0-1	Soft reset isolation. When high, this bit forces all USB1 signals that connect to the USBSS to zeros during a soft reset via bit 0 of this register. This bit should be set high prior to setting bit 0 and cleared after bit 0 is cleared.
4	rndis	0-1	Global RNDIS mode enable for all endpoints.
3	uint		USB non-highlander interrupt enable
		0	Highlander
		1	Non-highlander
2	Reserved	0	Always read as 0. Writes have no effect.
1	clkfack	0-1	Clock stop fast ack enable.
0	soft reset		Software reset of USB1.
		Write 0	No action
		Write 1	Initiate software reset
		Read 0	Reset done, no action
		Read 1	Reset ongoing

24.9.2.2.3 USB1 Status Register (USB1STAT)

The USB1 status register (USB1STAT) allows the CPU to check the voltage state level of USB1_DRVVBUS pin.

Ths USB1 status register is shown in Figure 24-89 and described in Table 24-97.

Figure 24-89. USB1 Status Register (USB1STAT)



LEGEND: R = Read only; -*n* = value after reset

Table 24-97. USB1 Status Register (USB1STAT) Field Descriptions

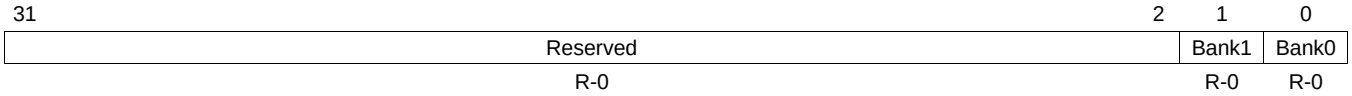
Bits	Field	Description
31-1	Reserved	Always read as 0. Writes have no effect.
0	drvvbus	Current USB1_DRVVBUS value

24.9.2.2.4 USB1 IRQ_MERGED_STATUS Register (USB1IRQMSTAT)

The USB1 IRQ_MERGED_STATUS register (USB1IRQMSTAT) contains a merged status for all the events for USB1. This register optimizes software accesses in a module where a large number of events are associated to one IRQ line. The merged status is read-only, and does not need to be actively cleared like the IRQ status. It will clear automatically.

The USB1 IRQ_MERGED_STATUS register is shown in Figure 24-90 and described in Table 24-98.

Figure 24-90. USB1 IRQ_MERGED_STATUS Register (USB1IRQMSTAT)



LEGEND: R = Read only; -*n* = value after reset

Table 24-98. USB1 IRQ_MERGED_STATUS Register (USB1IRQMSTAT) Field Descriptions

Bits	Field	Value	Description
31-2	Reserved	0	Always read as 0. Writes have no effect.
1	Bank1	0	No events pending from IRQ_STATUS_1
		1	At least one event is pending from IRQ_STATUS_1
0	Bank0	0	No events pending from IRQ_STATUS_0
		1	At least one event is pending from IRQ_STATUS_0

24.9.2.2.5 USB1 IRQ_EOI Register (USB1IRQEOI)

The USB1 IRQ_EOI register (USB1IRQEOI) allows the CPU to acknowledge completion of an interrupt by writing to the end of interrupt (EOI). An eoi_write signal will be generated and another interrupt will be triggered if interrupt sources remain. This register will be reset one cycle after it has been written to.

The USB1 IRQ_EOI register is shown in Figure 24-91 and described in Table 24-99.

Figure 24-91. USB1 IRQ_EOI Register (USB1IRQEOI)

31	1	0
Reserved		EOI for USB1
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-99. USB1 IRQ_EOI Register (USB1IRQEOI) Field Descriptions

Bits	Field	Description
31-1	Reserved	Always read as 0. Writes have no effect.
0	EOI for USB1	EOI for USB1 interrupt

24.9.2.2.6 USB1 IRQ_STATUS_RAW_0 Register (USB1IRQSTATRAW0)

The USB1 IRQ_STATUS_RAW_0 register (USB1IRQSTATRAW0) allows the USB1 interrupt sources to be manually triggered when writing a 1 to a specific bit. A read of this register returns the USB1 interrupt event pending value.

General actions per bit:
Write 0: No action
Write 1: Set event
Read 0: No event pending
Read 1: Event pending

The USB1 IRQ_STATUS_RAW_0 register is shown in Figure 24-92 and described in Table 24-100.

Figure 24-92. USB1 IRQ_STATUS_RAW_0 Register (USB1IRQSTATRAW0)

31	30	29	28	27	26	25	24
RXEP15	RXEP14	RXEP13	RXEP12	RXEP11	RXEP10	RXEP9	RXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
RXEP7	RXEP6	RXEP5	RXEP4	RXEP3	RXEP2	RXEP1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
15	14	13	12	11	10	9	8
TXEP15	TXEP14	TXEP13	TXEP12	TXEP11	TXEP10	TXEP9	TXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TXEP7	TXEP6	TXEP5	TXEP4	TXEP3	TXEP2	TXEP1	TXEP0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-100. USB1 IRQ_STATUS_RAW_0 Register (USB1IRQSTATRAW0) Field Descriptions

Bits	Field	Description
31-17	RXEP*n*	Interrupt status for RX endpoint *n*
16	Reserved	Always read 0. Writes have no effect.
15-0	TXEP*n*	Interrupt status for TX endpoint *n*

24.9.2.2.7 USB1 IRQ_STATUS_RAW_1 Register (USB1IRQSTATRAW1)

The USB1 IRQ_STATUS_RAW_1 register (USB1IRQSTATRAW1) allows the USB1 interrupt sources to be manually triggered when writing a 1 to a specific bit. A read of this register returns the USB1 interrupt event pending value.

General actions per bit:
Write 0: No action
Write 1: Set event
Read 0: No event pending
Read 1: Event pending

The USB1 IRQ_STATUS_RAW_1 register is shown in Figure 24-93 and described in Table 24-101.

Figure 24-93. USB1 IRQ_STATUS_RAW_1 Register (USB1IRQSTATRAW1)

31	30	29	28	27	26	25	24
TXFIFO15	TXFIFO14	TXFIFO13	TXFIFO12	TXFIFO11	TXFIFO10	TXFIFO9	TXFIFO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
TXFIFO7	TXFIFO6	TXFIFO5	TX FIFO4	TXFIFO3	TXFIFO2	TXFIFO1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
15					10	9	8
Reserved						USB[9]	USB[8]
R-0						R/W-0	R/W-0
7	6	5	4	3	2	1	0
USB[7]	USB[6]	USB[5]	USB[4]	USB[3]	USB[2]	USB[1]	USB[0]
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-101. USB1 IRQ_STATUS_RAW_1 Register (USB1IRQSTATRAW1) Field Descriptions

Bits	Field	Description
31-17	TXFIFO*n*	Interrupt status for TX FIFO endpoint *n*
16-10	Reserved	Always read 0. Writes have no effect.
9	USB[9]	Interrupt status for Mentor controller USB_INT generic interrupt
8	USB[8]	Interrupt status for DRVVBUS level change
7	USB[7]	Interrupt status for VBUS < VBUS valid threshold
6	USB[6]	Interrupt status for SRP detected
5	USB[5]	Interrupt status for device disconnected (host mode)
4	USB[4]	Interrupt status for device connected (host mode)
3	USB[3]	Interrupt status for SOF started
2	USB[2]	Interrupt status for Reset signaling detected (peripheral mode) Babble detected (host mode)
1	USB[1]	Interrupt status for Resume signaling detected
0	USB[0]	Interrupt status for Suspend signaling detected

24.9.2.2.8 USB1 IRQ_STATUS_0 Register (USBIRQSTAT0)

The USB1 IRQ_STATUS_0 register (USBIRQSTAT0) allows the USB1 interrupt sources to be manually cleared when writing a 1 to a specific bit. A read of this register returns the USB1 interrupt event pending value.

General actions per bit:
Write 0: No action
Write 1: Clear event
Read 0: No event pending
Read 1: Event pending

The USB1 IRQ_STATUS_0 register is shown in Figure 24-94 and described in Table 24-102.

Figure 24-94. USB1 IRQ_STATUS_0 Register (USB1IRQSTAT0)

31	30	29	28	27	26	25	24
RXEP15	RXEP14	RXEP13	RXEP12	RXEP11	RXEP10	RXEP9	RXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
RXEP7	RXEP6	RXEP5	RXEP4	RXEP3	RXEP2	RXEP1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
15	14	13	12	11	10	9	8
TXEP15	TXEP14	TXEP13	TXEP12	TXEP11	TXEP10	TXEP9	TXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TXEP7	TXEP6	TXEP5	TXEP4	TXEP3	TXEP2	TXEP1	TXEP0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-102. USB1 IRQ_STATUS_0 Register (USBIRQSTAT0) Field Descriptions

Bits	Field	Description
31-17	RXEP*n*	Interrupt status for RX endpoint *n*
16	Reserved	Always read 0. Writes have no effect.
15-0	TXEP*n*	Interrupt status for TX endpoint *n*

24.9.2.2.9 USB1 IRQ_STATUS_1 Register (USB0IRQSTAT1)

The USB1 IRQ_STATUS_1 register (USB0IRQSTAT1) allows the USB1 interrupt sources to be manually cleared when writing a 1 to a specific bit. A read of this register returns the USB1 interrupt event pending value.

General actions per bit:
Write 0: No action
Write 1: Clear event
Read 0: No event pending
Read 1: Event pending

The USB1 IRQ_STATUS_1 register is shown in Figure 24-95 and described in Table 24-103.

Figure 24-95. USB1 IRQ_STATUS_1 Register (USB0IRQSTAT1)

31	30	29	28	27	26	25	24
TXFIFO15	TXFIFO14	TXFIFO13	TXFIFO12	TXFIFO11	TXFIFO10	TXFIFO9	TXFIFO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
TXFIFO7	TXFIFO6	TXFIFO5	TX FIFO4	TXFIFO3	TXFIFO2	TXFIFO1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
15					10	9	8
Reserved						USB[9]	USB[8]
R-0						R/W-0	R/W-0
7	6	5	4	3	2	1	0
USB[7]	USB[6]	USB[5]	USB[4]	USB[3]	USB[2]	USB[1]	USB[0]
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-103. USB1 IRQ_STATUS_1 Register (USB0IRQSTAT1) Field Descriptions

Bits	Field	Description
31-17	TXFIFO*n*	Interrupt status for TX FIFO endpoint *n*
16-10	Reserved	Always read 0. Writes have no effect.
9	USB[9]	Interrupt status for Mentor controller USB_INT generic interrupt
8	USB[8]	Interrupt status for DRVVBUS level change
7	USB[7]	Interrupt status for VBUS < VBUS valid threshold
6	USB[6]	Interrupt status for SRP detected
5	USB[5]	Interrupt status for device disconnected (host mode)
4	USB[4]	Interrupt status for device connected (host mode)
3	USB[3]	Interrupt status for SOF started
2	USB[2]	Interrupt status for Reset signaling detected (peripheral mode) Babble detected (host mode)
1	USB[1]	Interrupt status for Resume signaling detected
0	USB[0]	Interrupt status for Suspend signaling detected

24.9.2.2.10 USB1 IRQ_ENABLE_SET_0 Register (USB1IRQENABLESET0)

The USB1 IRQ_ENABLE_SET_0 register (USB1IRQENABLESET0) allows the USB1 interrupt sources to be manually enabled when writing a 1 to a specific bit. A read of this register returns the USB1 interrupt enabled value.

General actions per bit:
Write 0: No action
Write 1: Interrupt enabled
Read 0: Interrupt disabled
Read 1: Interrupt enabled

The USB1 IRQ_ENABLE_SET_0 register is shown in Figure 24-96 and described in Table 24-104.

Figure 24-96. USB1 IRQ_ENABLE_SET_0 Register (USB1IRQENABLESET0)

31	30	29	28	27	26	25	24
RXEP15	RXEP14	RXEP13	RXEP12	RXEP11	RXEP10	RXEP9	RXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
RXEP7	RXEP6	RXEP5	RXEP4	RXEP3	RXEP2	RXEP1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
15	14	13	12	11	10	9	8
TXEP15	TXEP14	TXEP13	TXEP12	TXEP11	TXEP10	TXEP9	TXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TXEP7	TXEP6	TXEP5	TXEP4	TXEP3	TXEP2	TXEP1	TXEP0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-104. USB1 IRQ_ENABLE_SET_0 Register (USB1IRQENABLESET0) Field Descriptions

Bits	Field	Description
31-17	RXEP*n*	Interrupt enable for RX endpoint *n*
16	Reserved	Always read 0. Writes have no effect.
15-0	TXEP*n*	Interrupt enable for TX endpoint *n*

24.9.2.2.11 USB1 IRQ_ENABLE_SET_1 Register (USB1IRQENABLESET1)

The USB1 IRQ_ENABLE_SET_1 register (USB1IRQENABLESET1) allows the USB1 interrupt sources to be manually enabled when writing a 1 to a specific bit. A read of this register returns the USB1 interrupt enabled value.

General actions per bit:
Write 0: No action
Write 1: Interrupt enabled
Read 0: Interrupt disabled
Read 1: Interrupt enabled

The USB1 IRQ_ENABLE_SET_1 register is shown in Figure 24-97 and described in Table 24-105.

Figure 24-97. USB1 IRQ_ENABLE_SET_1 Register (USB1IRQENABLESET1)

31	30	29	28	27	26	25	24
TXFIFO15	TXFIFO14	TXFIFO13	TXFIFO12	TXFIFO11	TXFIFO10	TXFIFO9	TXFIFO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
TXFIFO7	TXFIFO6	TXFIFO5	TX FIFO4	TXFIFO3	TXFIFO2	TXFIFO1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
15					10	9	8
Reserved						USB[9]	USB[8]
R-0						R/W-0	R/W-0
7	6	5	4	3	2	1	0
USB[7]	USB[6]	USB[5]	USB[4]	USB[3]	USB[2]	USB[1]	USB[0]
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-105. USB1 IRQ_ENABLE_SET_1 Register (USB1IRQENABLESET1) Field Descriptions

Bits	Field	Description
31-17	TXFIFO*n*	Interrupt enable for TX FIFO endpoint *n*
16-10	Reserved	Always read 0. Writes have no effect.
9	USB[9]	Interrupt enable for Mentor controller USB_INT generic interrupt
8	USB[8]	Interrupt enable for DRVVBUS level change
7	USB[7]	Interrupt enable for VBUS < VBUS valid threshold
6	USB[6]	Interrupt enable for SRP detected
5	USB[5]	Interrupt enable for device disconnected (host mode)
4	USB[4]	Interrupt enable for device connected (host mode)
3	USB[3]	Interrupt enable for SOF started
2	USB[2]	Interrupt enable for Reset signaling detected (peripheral mode) Babble detected (host mode)
1	USB[1]	Interrupt enable for Resume signaling detected
0	USB[0]	Interrupt enable for Suspend signaling detected

24.9.2.2.12 USB1 IRQ_ENABLE_CLR_0 Register (USB1IRQENABLECLR0)

The USB1 IRQ_ENABLE_CLR_0 register (USB1IRQENABLECLR0) allows the USB1 interrupt sources to be manually disabled when writing a 1 to a specific bit. A read of this register returns the USB1 interrupt enabled value.

General actions per bit:
Write 0: No action
Write 1: Disable interrupt
Read 0: Interrupt disabled
Read 1: Interrupt enabled

The USB1 IRQ_ENABLE_CLR_0 register is shown in Figure 24-98 and described in Table 24-106.

Figure 24-98. USB1 IRQ_ENABLE_CLR_0 Register (USB1IRQENABLECLR0)

31	30	29	28	27	26	25	24
RXEP15	RXEP14	RXEP13	RXEP12	RXEP11	RXEP10	RXEP9	RXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
RXEP7	RXEP6	RXEP5	RXEP4	RXEP3	RXEP2	RXEP1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
15	14	13	12	11	10	9	8
TXEP15	TXEP14	TXEP13	TXEP12	TXEP11	TXEP10	TXEP9	TXEP8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TXEP7	TXEP6	TXEP5	TXEP4	TXEP3	TXEP2	TXEP1	TXEP0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-106. USB1 IRQ_ENABLE_CLR_0 Register (USB1IRQENABLECLR0) Field Descriptions

Bits	Field	Description
31-17	RXEP*n*	Interrupt enable for RX endpoint *n*
16	Reserved	Always read 0. Writes have no effect.
15-0	TXEP*n*	Interrupt enable for TX endpoint *n*

24.9.2.2.13 USB1 IRQ_ENABLE_CLR_1 Register (USB1IREENABLECLR1)

The USB1 IRQ_ENABLE_CLR_1 register (USB1IREENABLECLR1) allows the USB1 interrupt sources to be manually disabled when writing a 1 to a specific bit. A read of this register returns the USB1 interrupt enabled value.

General actions per bit:
Write 0: No action
Write 1: Disable interrupt
Read 0: Interrupt disabled
Read 1: Interrupt enabled

The USB1 IRQ_ENABLE_CLR_1 register is shown in Figure 24-97 and described in Table 24-107.

Figure 24-99. USB1 IRQ_ENABLE_CLR_1 Register (USB1IREENABLECLR1)

31	30	29	28	27	26	25	24
TXFIFO15	TXFIFO14	TXFIFO13	TXFIFO12	TXFIFO11	TXFIFO10	TXFIFO9	TXFIFO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
TXFIFO7	TXFIFO6	TXFIFO5	TX FIFO4	TXFIFO3	TXFIFO2	TXFIFO1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
15					10	9	8
Reserved						USB[9]	USB[8]
R-0						R/W-0	R/W-0
7	6	5	4	3	2	1	0
USB[7]	USB[6]	USB[5]	USB[4]	USB[3]	USB[2]	USB[1]	USB[0]
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-107. USB1 IRQ_ENABLE_CLR_1 Register (USB1IREENABLECLR1) Field Descriptions

Bits	Field	Description
31-17	TXFIFO*n*	Interrupt enable for TX FIFO endpoint *n*
16-10	Reserved	Always read 0. Writes have no effect.
9	USB[9]	Interrupt enable for Mentor controller USB_INT generic interrupt
8	USB[8]	Interrupt enable for DRVVBUS level change
7	USB[7]	Interrupt enable for VBUS < VBUS valid threshold
6	USB[6]	Interrupt enable for SRP detected
5	USB[5]	Interrupt enable for device disconnected (host mode)
4	USB[4]	Interrupt enable for device connected (host mode)
3	USB[3]	Interrupt enable for SOF started
2	USB[2]	Interrupt enable for Reset signaling detected (peripheral mode) Babble detected (host mode)
1	USB[1]	Interrupt enable for Resume signaling detected
0	USB[0]	Interrupt enable for Suspend signaling detected

24.9.2.2.14 USB1 Tx Mode Register (USB1TXMODE)

The USB1 Tx mode register (USB1TXMODE) allows the CPU to individually enable RNDIS/Generic/CDC modes for each endpoint. Using the global RNDIS enable (rndis) bit in the control register (USB1CTRL) overrides this register and enables the RNDIS mode for all endpoints.

The USB1 Tx mode register is shown in Figure 24-100 and described in Table 24-108.

Figure 24-100. USB1 Tx Mode Register (USB1TXMODE)

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Reserved	Tx15_mode	Tx14_mode	Tx13_mode	Tx12_mode	Tx11_mode	Tx10_mode	Tx9_mode
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	**13 12**	**11 10**	**9 8**	**7 6**	**5 4**	**3 2**	**1 0**
Tx8_mode	Tx7_mode	Tx6_mode	Tx5_mode	Tx4_mode	Tx3_mode	Tx2_mode	Tx1_mode
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-108. USB1 Tx Mode Register (USB1TXMODE) Field Descriptions

Bits	Field	Value	Description
31-30	Reserved	0	Always read as 0. Writes have no effect.
29-28	Tx15_mode		TX endpoint 15 mode.
		0	Transparent Mode on TX endpoint 15
		1h	RNDIS MODE on TX endpoint 15
		2h	CDC Mode on TX endpoint 15
		3h	Generic RNDIS Mode on TX endpoint 15
27-26	Tx14_mode		TX endpoint 14 mode.
		0	Transparent Mode on TX endpoint 14
		1h	RNDIS MODE on TX endpoint 14
		2h	CDC Mode on TX endpoint 14
		3h	Generic RNDIS Mode on TX endpoint 14
25-24	Tx13_mode		TX endpoint 13 mode.
		0	Transparent Mode on TX endpoint 13
		1h	RNDIS MODE on TX endpoint 13
		2h	CDC Mode on TX endpoint 13
		3h	Generic RNDIS Mode on TX endpoint 13
23-22	Tx12_mode		TX endpoint 12 mode.
		0	Transparent Mode on TX endpoint 12
		1h	RNDIS MODE on TX endpoint 12
		2h	CDC Mode on TX endpoint 12
		3h	Generic RNDIS Mode on TX endpoint 12
21-20	Tx11_mode		TX endpoint 11 mode.
		0	Transparent Mode on TX endpoint 11
		1h	RNDIS MODE on TX endpoint 11
		2h	CDC Mode on TX endpoint 11
		3h	Generic RNDIS Mode on TX endpoint 11
19-18	Tx10_mode		TX endpoint 10 mode.
		0	Transparent Mode on TX endpoint 10
		1h	RNDIS MODE on TX endpoint 10
		2h	CDC Mode on TX endpoint 10
		3h	Generic RNDIS Mode on TX endpoint 10

Table 24-108. USB1 Tx Mode Register (USB1TXMODE) Field Descriptions (continued)

Bits	Field	Value	Description
17-16	Tx9_mode		TX endpoint 9 mode.
		0	Transparent Mode on TX endpoint 9
		1h	RNDIS MODE on TX endpoint 9
		2h	CDC Mode on TX endpoint 9
		3h	Generic RNDIS Mode on TX endpoint 9
15-14	Tx8_mode		TX endpoint 8 mode.
		0	Transparent Mode on TX endpoint 8
		1h	RNDIS MODE on TX endpoint 8
		2h	CDC Mode on TX endpoint 8
		3h	Generic RNDIS Mode on TX endpoint 8
13-12	Tx7_mode		TX endpoint 7 mode.
		0	Transparent Mode on TX endpoint 7
		1h	RNDIS MODE on TX endpoint 7
		2h	CDC Mode on TX endpoint 7
		3h	Generic RNDIS Mode on TX endpoint 7
11-10	Tx6_mode		TX endpoint 6 mode.
		0	Transparent Mode on TX endpoint 6
		1h	RNDIS MODE on TX endpoint 6
		2h	CDC Mode on TX endpoint 6
		3h	Generic RNDIS Mode on TX endpoint 6
9-8	Tx5_mode		TX endpoint 5 mode.
		0	Transparent Mode on TX endpoint 5
		1h	RNDIS MODE on TX endpoint 5
		2h	CDC Mode on TX endpoint 5
		3h	Generic RNDIS Mode on TX endpoint 5
7-6	Tx4_mode		TX endpoint 4 mode.
		0	Transparent Mode on TX endpoint 4
		1h	RNDIS MODE on TX endpoint 4
		2h	CDC Mode on TX endpoint 4
		3h	Generic RNDIS Mode on TX endpoint 4
5-4	Tx3_mode		TX endpoint 3 mode.
		0	Transparent Mode on TX endpoint 3
		1h	RNDIS MODE on TX endpoint 3
		2h	CDC Mode on TX endpoint 3
		3h	Generic RNDIS Mode on TX endpoint 3
3-2	Tx2_mode		TX endpoint 2 mode.
		0	Transparent Mode on TX endpoint 2
		1h	RNDIS MODE on TX endpoint 2
		2h	CDC Mode on TX endpoint 2
		3h	Generic RNDIS Mode on TX endpoint 2
1-0	Tx1_mode		TX endpoint 1 mode.
		0	Transparent Mode on TX endpoint 1
		1h	RNDIS MODE on TX endpoint 1
		2h	CDC Mode on TX endpoint 1
		3h	Generic RNDIS Mode on TX endpoint 1

24.9.2.2.15 USB1 Rx Mode Register (USB1RXMODE)

The USB1 Rx mode register (USB1RXMODE) allows the CPU to individually enable RNDIS/Generic/CDC modes for each endpoint. Using the global RNDIS enable (rndis) bit in the control register (USB1CTRL) overrides this register and enables the RNDIS mode for all endpoints.

The USB1 Rx mode register is shown in Figure 24-101 and described in Table 24-109.

Figure 24-101. USB1 Rx Mode Register (USB1RXMODE)

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Reserved	Rx15_mode	Rx14_mode	Rx13_mode	Rx12_mode	Rx11_mode	Rx10_mode	Rx9_mode
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	**13 12**	**11 10**	**9 8**	**7 6**	**5 4**	**3 2**	**1 0**
Rx8_mode	Rx7_mode	Rx6_mode	Rx5_mode	Rx4_mode	Rx3_mode	Rx2_mode	Rx1_mode
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-109. USB1 Rx Mode Register (USB1RXMODE) Field Descriptions

Bits	Field	Value	Description
31-30	Reserved	0	Always read as 0. Writes have no effect.
29-28	Rx15_mode		RX endpoint 15 mode.
		0	Transparent mode on RX endpoint 15
		1h	RNDIS MODE on RX endpoint 15
		2h	CDC mode on RX endpoint 15
		3h	Generic RNDIS mode on RX endpoint 15
27-26	Rx14_mode		RX endpoint 14 mode.
		0	Transparent mode on RX endpoint 14
		1h	RNDIS MODE on RX endpoint 14
		2h	CDC mode on RX endpoint 14
		3h	Generic RNDIS mode on RX endpoint 14
25-24	Rx13_mode		RX endpoint 13 mode.
		0	Transparent mode on RX endpoint 13
		1h	RNDIS MODE on RX endpoint 13
		2h	CDC mode on RX endpoint 13
		3h	Generic RNDIS mode on RX endpoint 13
23-22	Rx12_mode		RX endpoint 12 mode.
		0	Transparent mode on RX endpoint 12
		1h	RNDIS MODE on RX endpoint 12
		2h	CDC mode on RX endpoint 12
		3h	Generic RNDIS mode on RX endpoint 12
21-20	Rx11_mode		RX endpoint 11 mode.
		0	Transparent mode on RX endpoint N+10
		1h	RNDIS MODE on RX endpoint N+10
		2h	CDC mode on RX endpoint N+10
		3h	Generic RNDIS mode on RX endpoint N+10
19-18	Rx10_mode		RX endpoint 10 mode.
		0	Transparent mode on RX endpoint 10
		1h	RNDIS MODE on RX endpoint 10
		2h	CDC mode on RX endpoint 10
		3h	Generic RNDIS mode on RX endpoint 10

Table 24-109. USB1 Rx Mode Register (USB1RXMODE) Field Descriptions (continued)

Bits	Field	Value	Description
17-16	Rx9_mode		RX endpoint 9 mode.
		0	Transparent mode on RX endpoint 9
		1h	RNDIS MODE on RX endpoint 9
		2h	CDC mode on RX endpoint 9
		3h	Generic RNDIS mode on RX endpoint 9
15-14	Rx8_mode		RX endpoint 8 mode.
		0	Transparent mode on RX endpoint 8
		1h	RNDIS MODE on RX endpoint 8
		2h	CDC mode on RX endpoint 8
		3h	Generic RNDIS mode on RX endpoint 8
13-12	Rx7_mode		RX endpoint 7 mode.
		0	Transparent mode on RX endpoint 7
		1h	RNDIS MODE on RX endpoint 7
		2h	CDC mode on RX endpoint 7
		3h	Generic RNDIS mode on RX endpoint 7
11-10	Rx6_mode		RX endpoint 6 mode.
		0	Transparent mode on RX endpoint 6
		1h	RNDIS MODE on RX endpoint 6
		2h	CDC mode on RX endpoint 6
		3h	Generic RNDIS mode on RX endpoint 6
9-8	Rx5_mode		RX endpoint 5 mode.
		0	Transparent mode on RX endpoint 5
		1h	RNDIS MODE on RX endpoint 5
		2h	CDC mode on RX endpoint 5
		3h	Generic RNDIS mode on RX endpoint 5
7-6	Rx4_mode		RX endpoint 4 mode.
		0	Transparent mode on RX endpoint 4
		1h	RNDIS MODE on RX endpoint 4
		2h	CDC mode on RX endpoint 4
		3h	Generic RNDIS mode on RX endpoint 4
5-4	Rx3_mode		RX endpoint 3 mode.
		0	Transparent mode on RX endpoint 3
		1h	RNDIS MODE on RX endpoint 3
		2h	CDC mode on RX endpoint 3
		3h	Generic RNDIS mode on RX endpoint 3
3-2	Rx2_mode		RX endpoint 2 mode.
		0	Transparent mode on RX endpoint 2
		1h	RNDIS MODE on RX endpoint 2
		2h	CDC mode on RX endpoint 2
		3h	Generic RNDIS mode on RX endpoint 2
1-0	Rx1_mode		RX endpoint 1 mode.
		0	Transparent mode on RX endpoint 1
		1h	RNDIS MODE on RX endpoint 1
		2h	CDC mode on RX endpoint 1
		3h	Generic RNDIS mode on RX endpoint 1

24.9.2.2.16 USB1 Generic RNDIS EP N Size Register (USB1GENRNDISEPn)

The USB1 generic RNDIS EP N size register (USB1GENRNDISEPn) is programmed with a RNDIS packet size in bytes. When EP(N) is in Generic RNDIS mode, the received USB packets will be collected into a single CPPI packet that is completed when the number of bytes equal to the value of this register has been received, or a “short” packet is received. Note that this register must be programmed with a value that is an integer multiple of the endpoint size. N can range from 1 to 15.

The USB1 generic RNDIS EP N size register is shown in Figure 24-102 and described in USB1 Generic RNDIS EP N Size Register (USB1GENRNDISEPn) Field Descriptions.

Figure 24-102. USB1 Generic RNDIS EP N Size Register (USB1GENRNDISEPn)

31	17	16	0
Reserved		Ep(N)_size	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

USB1 Generic RNDIS EP N Size Register (USB1GENRNDISEPn) Field Descriptions

Bits	Field	Description
31-17	Reserved	Always read as 0. Writes have no effect.
16-0	Ep(N)_size	Generic RNDIS packet size.

24.9.2.2.17 USB1 Auto Req Register (USB1AUTOREQ)

The USB1 auto req register (USB1AUTOREQ) allows the CPU to enable an automatic IN token request generation for host mode RX operation per each RX endpoint. This features has the DMA set the ReqPkt bit in the RXCSR when it clears the RxPktRdy bit after reading out a packet. The ReqPkt bit is used by the core to generate an IN token to receive data. By using this feature, the host can automatically generate an IN token after the DMA finishes receiving data and empties an endpoint buffer, thus receiving the next data packet as soon as possible from the connected device. Without this feature, the CPU will have to manually set the ReqPkt bit for every USB packet.

There are two modes that Auto Req can function: always or all except an EOP. The always mode will set the ReqPkt bit after every USB packet the DMA receives thus generating a new IN token after each USB packet. The EOP mode will set the ReqPkt bit after every USB packet that isn’t an EOP (end of packet) in the CPPI descriptor. For RNDIS, CDC, and Generic RNDIS modes the auto req will stop when the EOP is received (either via a short packet for RNDIS,CDC, and Generic RNDIS or the count is reached for Generic RNDIS), making it useful for starting a large RNDIS packet and having it auto generate IN tokens until the end of the RNDIS packet. For transparent mode, every USB packet is an EOP CPPI packet, so the auto req never functions and acts like auto req is disabled.

The USB1 auto req register is shown in Figure 24-103 and described in Table 24-110.

Figure 24-103. USB1 Auto Req Register (USB1AUTOREQ)

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Reserved	Rx(N+14)_ autoreq	Rx(N+13)_ autoreq	Rx(N+12)_ autoreq	Rx(N+11)_ autoreq	Rx(N+10)_ autoreq	Rx(N+9)_ autoreq	Rx(N+8)_ autoreq
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Rx(N+7)_ autoreq	Rx(N+6)_ autoreq	Rx(N+5)_ autoreq	Rx(N+4)_ autoreq	Rx(N+3)_ autoreq	Rx(N+2)_ autoreq	Rx(N+1)_ autoreq	Rx(N+0)_ autoreq
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-110. USB1 Auto Req Register (USB1AUTOREQ) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Always read as 0. Writes have no effect.
29-28	Rx(N+14)_autoreq		RX endpoint N+14 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
27-26	Rx(N+13)_autoreq		RX endpoint N+13 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
25-24	Rx(N+12)_autoreq		RX endpoint N+12 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
23-22	Rx(N+11)_autoreq		RX endpoint N+11 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
21-20	Rx(N+10)_autoreq		RX endpoint N+10 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
19-18	Rx(N+9)_autoreq		RX endpoint N+9 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
17-16	Rx(N+8)_autoreq		RX endpoint N+8 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always

Table 24-110. USB1 Auto Req Register (USB1AUTOREQ) Field Descriptions (continued)

Bit	Field	Value	Description
15-14	Rx(N+7)_autoreq		RX endpoint N+7 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
13-12	Rx(N+6)_autoreq		RX endpoint N+6 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
11-10	Rx(N+5)_autoreq		RX endpoint N+5 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
9-8	Rx(N+4)_autoreq		RX endpoint N+4 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
7-6	Rx(N+3)_autoreq		RX endpoint N+3 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
5-4	Rx(N+2)_autoreq		RX endpoint N+2 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
3-2	Rx(N+1)_autoreq		RX endpoint N+1 Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always
1-0	Rx(N)_autoreq		RX endpoint N Auto Req enable
		0	No auto req
		1h	Auto req on all but EOP
		2h	Reserved
		3h	Auto req always

24.9.2.2.18 USB1 SRP Fix Time Register (USB1SRPFIXTIME)

The USB1 SRP fix time register (USB1SRPFIXTIME) allows the CPU to configure the maximum amount of time the SRP fix logic blocks the AVAID from the PHY to the OTG core. This time allows the VBUS signal the ability to get below the thresholds and therefore remove the chance of voltage bounces which could give false threshold measurements.

The USB1 SRP fix time register is shown in Figure 24-104 and described in Table 24-111.

Figure 24-104. USB1 SRP Fix Time Register (USB1SRPFIXTIME)

31	0
srpfixtime	
R/W-0280 DE80h	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 24-111. USB1 SRP Fix Time Register (USB1SRPFIXTIME) Field Descriptions

Bits	Field	Description
31-0	srpfixtime	SRP Fix maximum time in 60 MHz cycles. Default is 700 ms.

24.9.2.2.19 USB1 Teardown Register (USB1TDOWN)

The USB1 teardown register (USB1TDOWN) controls the tearing down of rx and tx fifos in the USB controller. When a '1' is written to a valid bit in this register, the CPPI FIFO pointers for that endpoint are cleared , and the register automatically clears itself after 1 clock cycle. This register must be used in conjunction with the CPPI DMA Teardown mechanism. The Host should also write the FlushFIFO bits in the TXCSR and RXCSR Mentor USB Controller registers to ensure a complete teardown of the endpoint. See the Mentor specification for details.

The USB1 teardown register is shown in Figure 24-105 and described in Table 24-112.

Figure 24-105. USB1 Teardown Register (USB1TDOWN)

31	17	16	15	1	0
tx_tdown[*n*]		Rsvd	rx_tdown[*n*]		Rsvd
R/W-0		R-0	R/W-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-112. USB1 Teardown Register (USB1TDOWN) Field Descriptions

Bits	Field	Value	Description
31-17	tx_tdown[*n*]		Tx Endpoint Teardown. Write 1 to corresponding bit *n* to set. Read as 0.
		Bit 31	Endpoint 15
		...	...
		Bit 17	Endpoint 1
16	Reserved	0	Always read as 0. Writes have no effect.
15-1	rx_tdown[*n*]		RX Endpoint Teardown. Write 1 to corresponding bit *n* to set. Read as 0.
		Bit 15	Endpoint 15
		...	...
		Bit 1	Endpoint 1
0	Reserved	0	Always read as 0. Writes have no effect.

24.9.2.2.20 USB1 PHY UTMI Register (USB1UTMI)

The USB1 PHY UTMI register (USB1UTMI) controls various PHY UTMI signals. The UTMI interface is located between the PHY module and the Mentor Graphics Controller. These signals are inputs to the PHY but are not outputs from the Mentor Graphics Controller. It is important to set the otgdisable bit in to a high value to operate the PHY in a non-OTG mode.

The USB1 PHY UTMI register is shown in Figure 24-106 and described in Table 24-113.

Figure 24-106. USB1 PHY UTMI Register (USB1UTMI)

31							24
Reserved							
R-0							
23	22	21	20	19	18	17	16
txbitstuffen	txbitstuffenh	otgdisable	vbusvldextsel	vbusvldext	txenablen	fsxcvrowner	txvalidh
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15							8
datainh							
R/W-0							
7					3	2	1	0
Reserved						wordinterface	fsdataext	fsse0ext
R-0						R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-113. USB1 PHY UTMI Register (USB1UTMI) Field Descriptions

Bits	Field	Description
31-24	Reserved	Always read as 0. Writes have no effect.
23	txbitstuffen	PHY UTMI input for signal txbitstuffen
22	txbitstuffenh	PHY UTMI input for signal txbitstuffenh
21	otgdisable	PHY UTMI input for signal otgdisable
20	vbusvldextsel	PHY UTMI input for signal vbusvldextsel
19	vbusvldext	PHY UTMI input for signal vbusvldext
18	txenablen	PHY UTMI input for signal txenablen
17	fsxcvrowner	PHY UTMI input for signal fsxcvrowner
16	txvalidh	PHY UTMI input for signal txvalidh
15-8	datainh	PHY UTMI input for signal datainh
7-3	Reserved	Always read as 0. Writes have no effect.
2	wordinterface	PHY UTMI input for signal wordinterface
1	fsdataext	PHY UTMI input for signal fsdataext
0	fsse0ext	PHY UTMI input for signal fsse0ext

24.9.2.2.21 USB1 MGC UTMI Loopback Register (USB1UTMILB)

The USB1 MGC UTMI loopback register (USB1UTMILB) contains most of the input and output UTMI signals for the Mentor Graphics Controller. The UTMI interface is located between the PHY module and the Mentor Graphics Controller.

In the loopback mode, test register bits 11 to 0 control various UTMI signals that are input to the Mentor Graphics Controller.

In the loopback mode, test register bits 28 to 16 observe various UTMI signals that are output from the Mentor Graphics Controller.

The USB1 MGC UTMI loopback register is shown in Figure 24-107 and described in Table 24-114.

Figure 24-107. USB1 MGC UTMI Loopback Register (USB1UTMILB)

31	30	29	28	27	26	25	24
Reserved			suspendm	opmode		txvalid	xcvrsel
R-0			R-0	R-0		R-0	R-0
23	22	21	20	19	18	17	16
xcvrsel	termsel	drvvbus	chrgvbus	dischrgvbus	dppulldown	dmpulldown	idpullup
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
Reserved				iddig	hostdiscon	sessend	avalid
R-0				R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
vbusvalid	rxerror	Reserved		linestate		Reserved	
R/W-0	R/W-0	R-0		R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-114. USB1 MGC UTMI Loopback Register (USB1UTMILB) Field Descriptions

Bits	Field	Description
31-29	Reserved	Always read as 0. Writes have no effect.
28	suspendm	Loopback test observed value for suspendm
27-26	opmode	Loopback test observed value for opmode
25	txvalid	Loopback test observed value for txvalid
24-23	xcvrsel	Loopback test observed value for xcvrsel
22	termsel	Loopback test observed value for termsel
21	drvvbus	Loopback test observed value for drvvbus
20	chrgvbus	Loopback test observed value for chrgvbus
19	dischrgvbus	Loopback test observed value for dischrgvbus
18	dppulldown	Loopback test observed value for dppulldown
17	dmpulldown	Loopback test observed value for dmpulldown
16	idpullup	Loopback test observed value for idpullup
15-12	Reserved	Always read as 0. Writes have no effect.
11	iddig	Loopback test value for iddig
10	hostdiscon	Loopback test value for hostdiscon
9	sessend	Loopback test value for sessend
8	avalid	Loopback test value for avalid
7	vbusvalid	Loopback test value for vbusvalid
6	rxerror	Loopback test value for rxerror
5-4	Reserved	Always read as 0. Writes have no effect.
3-2	linestate	Loopback test value for linestate
1-0	Reserved	Always read as 0. Writes have no effect.

24.9.2.2.22 USB1 Mode Register (USB1MODE)

The USB1 mode register (USB1MODE) supports operating the PHY in a non-OTG mode. This requires the user to set the MGC UTMI input signal iddig. OTG interfaces have an id external pin which control UTMI signal iddig. Since this external pin is not available, the user should set iddig to either a 0 (A-type) or 1 (B-type). This value will be the initial setting for the Mentor controller. But, the controller will determine whether it is operating as a host or device via its protocols. To determine the function of the controller this information can be found by reading the Mentor Controller DEVCTL register (80h) bit 2.

The loopback bit enables the loopback test. This test allows MGC1 to be connected to MGC0. It is important to set both loopback bits in both USB0/1 Mode registers. The USB0 MGC UTMI Loopback Register contains the various UTMI signals to be controlled and observed during loopback test.

The phy_test bit enables the phy_test mode. This test mode is intended to allow additional control of the UTMI inputs to the PHY. Currently, these inputs are drvvbus, dppulldown, dmpulldown, and idpullup. When phy_test is high, then the pin inputs for these signals control the inputs to the PHY instead of the Mentor controller outputs. The phy_test mode is not active with loopback mode is active.

When phy_test is active than the PHY inputs datainh is equal to datain. And txvalidh is equal to txvalid.

The USB1 mode register is shown in Figure 24-108 and described in Table 24-115.

Figure 24-108. USB1 Mode Register (USB1MODE)

31	16
Reserved	
R-0	

15	9	8	7	6	2	1	0
Reserved		iddig	Rsvd	Reserved		phy_test	loopback
R-0		R/W-1	R/W-0	R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-115. USB1 Mode Register (USB1MODE) Field Descriptions

Bits	Field	Value	Description
31-9	Reserved	0	Always read as 0. Writes have no effect.
8	iddig		MGC input value for iddig
		0	A type
		1	B type
7	Reserved	0-1	For a PG1.x device, read as 0. For a PG2.x device, a 1 must be written to this bit.
6-2	Reserved	0	Always read as 0. Writes have no effect.
1	phy_test		PHY test
		0	Normal mode
		1	PHY test mode
0	loopback		Loopback test mode
		0	Normal mode
		1	Loopback test mode

24.9.2.2.23 USB1 Mentor Core Registers

A description of the mentor core registers is available in Section 24.9.6. Two instantiations of the USB core exists, USB0 and USB1. These registers reside back to back within USB0 space. The core registers that are used by USB0 subsystem are defined within offsets 1400h-159Ch, while the core registers that are used by USB1 subsystem are defined within offsets 1C00h–1D9Ch.

24.9.3 CPPI DMA Controller Registers

Table 24-116 lists the registers for the CPPI DMA Controller submodule.

Table 24-116. CPPI DMA Controller Registers

Address Offset	Acronym	CPPI DMA Controller Register
2000h	DMAREVID	Revision Register
2004h	TDFDQ	Teardown Free Descriptor Queue Control Register
2008h	DMAEMU	Emulation Control Register
2010h	DMAMEM1BA	CPPI Mem1 Base Address Register
2014h	DMAMEM1MASK	CPPI Mem1 Mask Address Register
2800h	TXGCR0	Tx Channel 0 Global Configuration Register
2808h	RXGCR0	Rx Channel 0 Global Configuration Register
280Ch	RXHPCRA0	Rx Channel 0 Host Packet Configuration Register A
2810h	RXHPCRB0	Rx Channel 0 Host Packet Configuration Register B
2820h	TXGCR1	Tx Channel 1 Global Configuration Register
2828h	RXGCR1	Rx Channel 1 Global Configuration Register
282Ch	RXHPCRA1	Rx Channel 1 Host Packet Configuration Register A
2830h	RXHPCRB1	Rx Channel 1 Host Packet Configuration Register B
2840h	TXGCR2	Tx Channel 2 Global Configuration Register
2848h	RXGCR2	Rx Channel 2 Global Configuration Register
284Ch	RXHPCRA2	Rx Channel 2 Host Packet Configuration Register A
2850h	RXHPCRB2	Rx Channel 2 Host Packet Configuration Register B
2860h	TXGCR3	Tx Channel 3 Global Configuration Register
2868h	RXGCR3	Rx Channel 3 Global Configuration Register
286Ch	RXHPCRA3	Rx Channel 3 Host Packet Configuration Register A
2870h	RXHPCRB3	Rx Channel 3 Host Packet Configuration Register B
2880h-2B9Fh	...	...
2BA0h	TXGCR29	Tx Channel 29 Global Configuration Register
2BA8h	RXGCR29	Rx Channel 29 Global Configuration Register
2BACh	RXHPCRA29	Rx Channel 29 Host Packet Configuration Register A
2BB0h	RXHPCRB29	Rx Channel 29 Host Packet Configuration Register B

24.9.3.1 CPPI DMA Revision Register (DMAREVID)

The CPPI DMA revision register (DMAREVID) contains the major and minor revisions for the module. The CPPI DMA revision register is shown in Figure 24-109 and described in Table 24-117.

Figure 24-109. CPPI DMA Revision Register (DMAREVID)

31 30	29 16	15 11	10 8	7 0
Rsvd	modID	revrtl	revmaj	revmin
R-0	R-53h	R-2h	R-1	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-117. CPPI DMA Revision Registe (DMAREVID) Field Descriptions

Bits	Field	Value	Description
31-30	Reserved	0	Always read as 0. Writes have no effect.
29-16	modID	0-3FFFh	Module ID field
15-11	revrtl	0-1Fh	RTL revision.
10-8	revmaj	0-7h	Major revision.
7-0	revmin	0-FFh	Minor revision.

24.9.3.2 CPPI DMA Teardown Free Descriptor Queue Control Register (TDFDQ)

The CPPI DMA teardown free descriptor queue control register (TDFDQ) is used to inform the DMA of the location in memory or descriptor array which is to be used for signaling of a teardown complete for each Tx and Rx channel. The CPPI DMA teardown free descriptor queue control register is shown in Figure 24-110 and described in Table 24-118.

Figure 24-110. CPPI DMA Teardown Free Descriptor Queue Control Register (TDFDQ)

31 14	13 12	11 0
Reserved	td_desc_qmgr	td_desc_qnum
R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-118. CPPI DMA Teardown Free Descriptor Queue Control Register (TDFDQ) Field Descriptions

Bits	Field	Description
31-14	Reserved	Reserved
13-12	td_desc_qmgr	This field controls which of the 4 Queue Managers the DMA will access in order to allocate a channel teardown descriptor from the teardown descriptor queue.
11-0	td_desc_qnum	This field controls which of the 2K queues in the indicated queue manager should be read in order to allocate channel teardown descriptors.

24.9.3.3 CPPI DMA Emulation Control Register (DMAEMU)

The CPPI DMA emulation control register (DMAEMU) is used to control the behavior of the DMA when the emususp input is asserted. The CPPI DMA emulation control register is shown in Figure 24-111 and described in Table 24-119.

Figure 24-111. CPPI DMA Emulation Control Register (DMAEMU)

31	2	1	0
Reserved		soft	free
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-119. CPPI DMA Emulation Control Register (DMAEMU) Field Descriptions

Bits	Field	Value	Description
31-2	Reserved	0	Always read as 0. Writes have no effect.
1	soft		Control for emulation pause request
		0	Forces emu_pause_req low
		1	Does not force emu_pause_req low
0	free	0-1	Enable for emulation suspend

24.9.3.4 CPPI Mem1 Base Address Register (DMAMEM1BA)

The CPPI Mem1 base address register (DMAMEM1BA) is the base address for the CPPI mem1 accesses. This base address points to the location of the data that will be transferred from external memory to the USB DMA RAM. This 16-bit field is the 16 most-significant bits of the 32-bit base address.

The CPPI Mem1 base address register is shown in Figure 24-112 and described in Table 24-120.

Figure 24-112. CPPI Mem1 Base Address Register (DMAMEM1BA)

31	16	15	0
mem1_base		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-120. CPPI Mem1 Base Address Register (DMAMEM1BA) Field Descriptions

Bits	Field	Description
31-16	mem1_base	CPPI mem1 base address
15-0	Reserved	Always read as 0. Writes have no effect.

24.9.3.5 CPPI Mem1 Mask Address Register (DMAMEM1MASK)

The CPPI Mem1 mask address register (DMAMEM1MASK) is the mask address for the CPPI mem1 base register (DMAMEM1BA). The 16-bit mem1_mask is ANDed with the 16-bit mem1_base register. The 16-bit resultant signal is the 16 most-significant bits of the 32-bit address for external memory accesses during DMA transactions.

The CPPI Mem1 mask address register is shown in Figure 24-113 and described in Table 24-121.

Figure 24-113. CPPI Mem1 Mask Address Register (DMAMEM1MASK)

31	16	15	0
mem1_mask		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-121. CPPI Mem1 Mask Address Register (DMAMEM1MASK) Field Descriptions

Bits	Field	Description
31-16	mem1_mask	CPPI mem1 mask address
15-0	Reserved	Always read as 0. Writes have no effect.

24.9.3.6 Tx Channel N Global Configuration Register (TXGCRn)

The Tx channel N global configuration register (TXGCRn) are used to initialize the behavior of each of the Tx CPPI DMA channels. N,n ranges from 0 to 14. The Tx channel N global configuration register is shown in Figure 24-114 and described in Table 24-122.

Figure 24-114. Tx Channel N Global Configuration Register (TXGCRn)

31	30	29	16
tx_enable	tx_teardown	Reserved	
R/W-0	R/W-0	R-0	

15	14	13	12	11	0
Reserved		tx_default_qmgr		tx_default_qnum	
R-0		W-0		W-0	

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 24-122. Tx Channel N Global Configuration Register (TXGCRn) Field Descriptions

Bits	Field	Value	Description
31	tx_enable		This field enables or disables the channel
		0	Channel is disabled
		1	Channel is enabled
			This field will be cleared after a channel teardown is complete.
30	tx_teardown	0-1	Setting this bit will request the channel to be torn down. This field will remain set after a channel teardown is complete.
29-14	Reserved	0	Reserved
13-12	tx_default_qmgr	0-3h	This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.
11-0	tx_default_qnum	0-FFFh	This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.

24.9.3.7 Rx Channel N Global Configuration Register (RXGCRn)

The Rx channel N global configuration register (RXGCRn) are used to initialize the global (non descriptor type specific) behavior of each of the Rx CPPI DMA channels. If the enable bit is being set, the Rx Channel Global Configuration Register should only be written after all of the other Rx Configuration Registers have been initialized. N,n ranges from 0 to 14.

The Rx channel N global configuration register is shown in Figure 24-115 and described in Table 24-123.

Figure 24-115. Rx Channel N Global Configuration Register (RXGCRn)

31	30	29 25	24	23 16
rx_enable	rx_teardown	Reserved	rx_error_handling	rx_sop_offset
R/W-0	R/W-0	R-0	W-0	W-0

15 14	13 12	11 0
rx_default_desc_type	rx_default_rq_qmgr	rx_default_rq_qnum
W-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 24-123. Rx Channel N Global Configuration Register (RXGCRn) Field Descriptions

Bits	Field	Value	Description
31	rx_enable		This field enables or disables the channel
		0	Channel is disabled
		1	Channel is enabled
			This field will be cleared after a channel teardown is complete.
30	rx_teardown	0-1	This field indicates whether or not an Rx teardown operation is complete. This field should be cleared when a channel is initialized. This field will be set after a channel teardown is complete.
29-25	Reserved	0	Reserved
24	rx_error_handling		This bit controls the error handling mode for the channel and is only used when channel errors (descriptor or buffer starvation occurs):
		0	Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to
		1	Starvation errors result in subsequent re-try of the descriptor allocation operation. In this mode, the DMA will return to the IDLE state without saving it's internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface. This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.
			Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when
23-16	rx_sop_offset	0-FFh	This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload. This value must be less than the minimum size of a buffer in the system. Valid values are 0 – 255 bytes.
15-14	rx_default_desc_type		This field indicates the default descriptor type to use:
		0	Reserved
		1h	Host
		2h-3h	Reserved
			The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.
13-12	rx_default_rq_qmgr	0-3h	This field indicates the default receive queue manager that this channel should use. The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.
11-0	rx_default_rq_qnum	0-FFFh	This field indicates the default receive queue that this channel should use. The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.

24.9.3.8 Rx Channel N Host Packet Configuration Register A (RXHPCRAn)

The Rx channel N host packet configuration register A (RXHPCRAn) are used to initialize the behavior of each of the Rx CPPI DMA channels for reception of host type packets. N,n ranges from 0 to 14. The Rx channel N host packet configuration register A is shown in Figure 24-116 and described in Figure 24-116.

Figure 24-116. Rx Channel N Host Packet Configuration Register A (RXHPCRAn)

31	30	29	28	27	16
Reserved		rx_host_fdq1_qmgr		rx_host_fdq1_qnum	
R-0		W-0		W-0	

15	14	13	12	11	0
Reserved		rx_host_fdq0_qmgr		rx_host_fdq0_qnum	
R-0		W-0		W-0	

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 24-124. Rx Channel N Host Packet Configuration Register A (RXHPCRAn) Field Descriptions

Bits	Field	Description
31-30	Reserved	Reserved
29-28	rx_host_fdq1_qmgr	This field specifies which Buffer Manager should be used for the 2nd Rx buffer in a host type packet
27-16	rx_host_fdq1_qnum	This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet
15-14	Reserved	Reserved
13-12	rx_host_fdq0_qmgr	This field specifies which Buffer Manager should be used for the 1st Rx buffer in a host type packet
11-0	rx_host_fdq0_qnum	This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet

24.9.3.9 Rx Channel N Host Packet Configuration Register B (RXHPCRBn)

The Rx channel N host packet configuration register B (RXHPCRBn) are used to initialize the behavior of each of the Rx CPPI DMA channels for reception of host type packets. N,n ranges from 0 to 14. The Rx channel N host packet configuration register B is shown in Figure 24-117 and described in Table 24-125.

Figure 24-117. Rx Channel N Host Packet Configuration Register B (RXHPCRBn)

31 30	29 28	27 16
Reserved	rx_host_fdq3_qmgr	rx_host_fdq3_qnum
R-0	W-0	W-0

15 14	13 12	11 0
Reserved	rx_host_fdq2_qmgr	rx_host_fdq2_qnum
R-0	W-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 24-125. Rx Channel N Host Packet Configuration Register B (RXHPCRBn) Field Descriptions

Bits	Field	Description
31-30	Reserved	Reserved
29-28	rx_host_fdq3_qmgr	This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet
27-16	rx_host_fdq3_qnum	This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet
15-14	Reserved	Reserved
13-12	rx_host_fdq2_qmgr	This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet
11-0	rx_host_fdq2_qnum	This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet

24.9.4 CPPI DMA Scheduler Registers

Table 24-126 lists the registers for the CPPI DMA Scheduler submodule.

Table 24-126. CPPI DMA Scheduler Registers

Address Offset	Acronym	CPPI DMA Scheduler Register
3000h	DMA_SCHED_CTRL	CPPI DMA Scheduler Control Register
3800h-38FCh	WORD0-WORD63	CPPI DMA Scheduler Table Word 0-Word 63

24.9.4.1 CPPI DMA Scheduler Control Register (DMA_SCHED_CTRL)

The CPPI DMA scheduler control register (DMA_SCHED_CTRL) contains the major and minor revisions for the module. This register is shown in Figure 24-118 and described in Table 24-127.

Figure 24-118. CPPI DMA Scheduler Control Register (DMA_SCHED_CTRL)

31	30	16
enable	Reserved	
R/W-0	R-0	

15	8	7	0
Reserved		last_entry	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-127. CPPI DMA Scheduler Control Register (DMA_SCHED_CTRL) Field Descriptions

Bits	Field	Value	Description
31	enable		This is the enable bit for the scheduler and is encoded as follows:
		0	Scheduler is disabled and will no longer fetch entries from the scheduler table or pass credits to the DMA controller
		1	Scheduler is enabled
			This bit should only be set after the table has been initialized.
30-8	Reserved	0	Reserved
7-0	last_entry		This field indicates the last valid entry in the scheduler table. There are 64 words in the table and there are 4 entries in each word. The table can be programmed with any integer number of entries from 1 to 256. The corresponding encoding for this field is as follows:
		0	1 entry
		1h	2 entries
		…	
		FEh	255 entries
		FFh	256 entries

24.9.4.2 CPPI DMA Scheduler Table Word N (WORDn)

The Tx channel configuration registers are used to initialize the behavior of each of the Tx DMA channels. N ranges from 0 to 63. The CPPI DMA scheduler table word N register (WORDn) is shown in Figure 24-119 and described in Table 24-128.

Figure 24-119. CPPI DMA Scheduler Table Word N Register (WORDn)

31	30 29	28 24	23	22 21	20 16
entry3_rxtx	Reserved	entry3_channel	entry2_rxtx	Reserved	entry2_channel
W-0	R-0	W-0	W-0	R-0	W-0

15	14 13	12 8	7	6 5	4 0
entry1_rxtx	Reserved	entry1_channel	entry0_rxtx	Reserved	entry0_channel
W-0	R-0	W-0	W-0	R-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 24-128. CPPI DMA Scheduler Table Word N Register (WORDn) Field Descriptions

Bits	Field	Value	Description
31	entry3_rxtx		This bit indicates if this entry is for a Tx or an Rx channel and is encoded as follows:
		0	Tx Channel
		1	Rx Channel
30-29	Reserved	0	Reserved
28-24	entry3_channel	0-1Fh	This field indicates the channel # that is to be given an opportunity to transfer data. If this is a Tx entry, the DMA will be presented with a scheduling 'credit' for that exact Tx channel. If this is an Rx entry, the DMA will be presented with a scheduling credit for the Rx FIFO that is associated with this channel. For Rx FIFOs which carry traffic for more than 1 Rx DMA channel, the exact channel number that is given in the Rx credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry.
23	entry2_rxtx		This bit indicates if this entry is for a Tx or an Rx channel and is encoded as follows:
		0	Tx Channel
		1	Rx Channel
22-21	Reserved	0	Reserved
20-16	entry2_channel	0-1Fh	This field indicates the channel # that is to be given an opportunity to transfer data. If this is a Tx entry, the DMA will be presented with a scheduling 'credit' for that exact Tx channel. If this is an Rx entry, the DMA will be presented with a scheduling credit for the Rx FIFO that is associated with this channel. For Rx FIFOs which carry traffic for more than 1 Rx DMA channel, the exact channel number that is given in the Rx credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry.
15	entry1_rxtx		This bit indicates if this entry is for a Tx or an Rx channel and is encoded as follows:
		0	Tx Channel
		1	Rx Channel
14-13	Reserved		Reserved
12-8	entry1_channel	0-1Fh	This field indicates the channel # that is to be given an opportunity to transfer data. If this is a Tx entry, the DMA will be presented with a scheduling 'credit' for that exact Tx channel. If this is an Rx entry, the DMA will be presented with a scheduling credit for the Rx FIFO that is associated with this channel. For Rx FIFOs which carry traffic for more than 1 Rx DMA channel, the exact channel number that is given in the Rx credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry.
7	entry0_rxtx		This bit indicates if this entry is for a Tx or an Rx channel and is encoded as follows:
		0	Tx Channel
		1	Rx Channel
6-5	Reserved	0	Reserved

Table 24-128. CPPI DMA Scheduler Table Word N Register (WORDn) Field Descriptions (continued)

Bits	Field	Value	Description
4-0	entry0_channel	0-1Fh	This field indicates the channel # that is to be given an opportunity to transfer data. If this is a Tx entry, the DMA will be presented with a scheduling 'credit' for that exact Tx channel. If this is an Rx entry, the DMA will be presented with a scheduling credit for the Rx FIFO that is associated with this channel. For Rx FIFOs which carry traffic for more than 1 Rx DMA channel, the exact channel number that is given in the Rx credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry.

24.9.5 CPPI DMA Queue Manager Registers

Table 24-129 lists the registers for the CPPI DMA Queue Manager submodule.

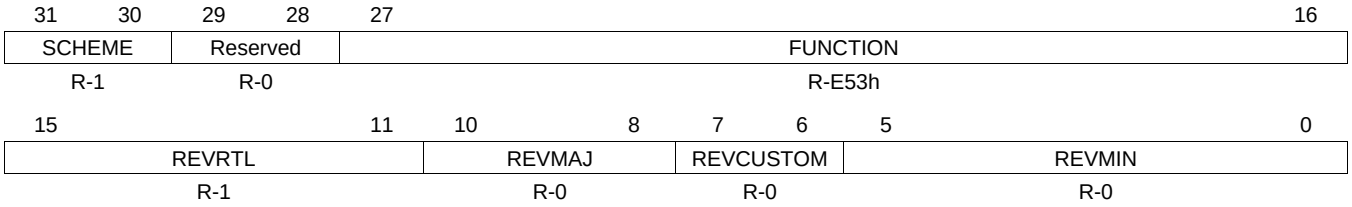
Table 24-129. CPPI DMA Queue Manager Registers

Address Offset	Acronym	CPPI DMA Queue Manager Register
4000h	QMGRREVID	Revision Register
4008h	DIVERSIO	Queue Diversion Register
4020h	FDBSC0	Free Descriptor/Buffer Starvation Count Qmgr Register 0
4024h	FDBSC1	Free Descriptor/Buffer Starvation Count Register 1
4028h	FDBSC2	Free Descriptor/Buffer Starvation Count Register 2
402Ch	FDBSC3	Free Descriptor/Buffer Starvation Count Register 3
4030h	FDBSC4	Free Descriptor/Buffer Starvation Count Register 4
4034h	FDBSC5	Free Descriptor/Buffer Starvation Count Register 5
4038h	FDBSC6	Free Descriptor/Buffer Starvation Count Register 6
403Ch	FDBSC7	Free Descriptor/Buffer Starvation Count Register 7
4080h	LRAM0BASE	Linking RAM Region 0 Base Address Register
4084h	LRAM0SIZE	Linking RAM Region 0 Size Register
4088h	LRAM1BASE	Linking RAM Region 1 Base Address Register
4090h	PEND0	Queue Pending Register 0
4094h	PEND1	Queue Pending Register 1
4098h	PEND2	Queue Pending Register 2
409Ch	PEND3	Queue Pending Register 3
40A0h	PEND4	Queue Pending Register 4
5000h + 16 x R	QMEMRBASEr	Memory Region R Base Address Register
5000h + 16 x R + 4	QMEMRCTRLr	Memory Region R Control Register
6000h + 16 x N	CTRLAn	Queue Manager Queue N Register A
6004h + 16 x N	CTRLBn	Queue Manager Queue N Register B
6008h + 16 x N	CTRLCn	Queue Manager Queue N Register C
600Ch + 16 x N	CTRLDn	Queue N Register D
7000h + 16 x N	QSTATAn	Queue N Status Register A
7004h + 16 x N	QSTATBn	Queue N Status Register B
7008h + 16 x N	QSTATCn	Queue N Status Register C

24.9.5.1 Queue Manager Revision Register (QMGRREVID)

The queue manager revision register (QMGRREVID) contains the major and minor revisions for the module. It does not support byte accesses. This register is shown in Figure 24-120 and described in Table 24-130.

Figure 24-120. Queue Manager Revision Register (QMGRREVID)



LEGEND: R = Read only; -*n* = value after reset

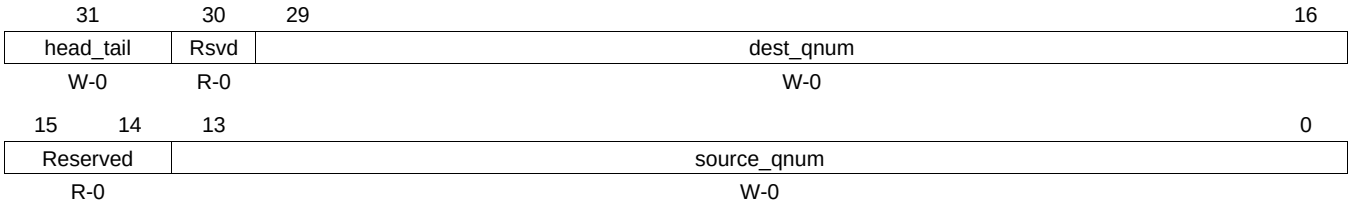
Table 24-130. Queue Manager Revision Register (QMGRREVID) Field Descriptions

Bits	Field	Value	Description
31-30	SCHEME	0-3h	Used to distinguish between old scheme and current.
29-28	Reserved	0	Reserved
27-16	FUNCTION	0-FFFh	Function indicates a software compatible module family.
15-11	REVRTL	0-1Fh	RTL revision
10-8	REVMAJ	0-7h	Major revision
7-6	REVCUSTOM	0-3h	Custom revision
5-0	REVMIN	0-3Fh	Minor revision

24.9.5.2 Queue Manager Queue Diversion Register (DIVERSION)

The queue manager queue diversion register (DIVERSION) is used to transfer the contents of one queue onto another queue. It does not support byte accesses. This register is shown in Figure 24-121 and described in Table 24-131.

Figure 24-121. Queue Manager Queue Diversion Register (DIVERSION)



LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 24-131. Queue Manager Queue Diversion Register (DIVERSION) Field Descriptions

Bits	Field	Description
31	head_tail	Indicates whether queue contents should be merged on to head or tail of destination queue. Clear this field for head and set for tail.
30	Reserved	Reserved
29-16	dest_qnum	Destination Queue Number
15-14	Reserved	Reserved
13-0	source_qnum	Source Queue Number

24.9.5.3 Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (FDBSC0)

The queue manager free descriptor/buffer starvation count register 0 (FDBSC0) provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses. This register is shown in Figure 24-122 and described in Table 24-132.

Figure 24-122. Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (FDBSC0)

31 24	23 16	15 8	7 0
fdbq3_starve_cnt	fdbq2_starve_cnt	fdbq1_starve_cnt	fdbq0_starve_cnt
RC-0	RC-0	RC-0	RC-0

LEGEND: RC = Clear on read; -*n* = value after reset

Table 24-132. Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (FDBSC0) Field Descriptions

Bits	Field	Description
31-24	fdbq3_starve_cnt	This field increments each time the free descriptor/buffer queue 3 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
23-16	fdbq2_starve_cnt	This field increments each time the free descriptor/buffer queue 2 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
15-8	fdbq1_starve_cnt	This field increments each time the free descriptor/buffer queue 1 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
7-0	fdbq0_starve_cnt	This field increments each time the free descriptor/buffer queue 0 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.

24.9.5.4 Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (FDBSC1)

The queue manager free descriptor/buffer starvation count register 1 (FDBSC1) provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses. This register is shown in Figure 24-123 and described in Table 24-133.

Figure 24-123. Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (FDBSC1)

31 24	23 16	15 8	7 0
fdbq7_starve_cnt	fdbq6_starve_cnt	fdbq5_starve_cnt	fdbq4_starve_cnt
RC-0	RC-0	RC-0	RC-0

LEGEND: RC = Clear on read; -*n* = value after reset

Table 24-133. Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (FDBSC1) Field Descriptions

Bits	Field	Description
31-24	fdbq7_starve_cnt	This field increments each time the free descriptor/buffer queue 7 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
23-16	fdbq6_starve_cnt	This field increments each time the free descriptor/buffer queue 6 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
15-8	fdbq5_starve_cnt	This field increments each time the free descriptor/buffer queue 5 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
7-0	fdbq4_starve_cnt	This field increments each time the free descriptor/buffer queue 4 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.

24.9.5.5 Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (FDBSC2)

The queue manager free descriptor/buffer starvation count register 2 (FDBSC2) provides statistics about how many starvation events are occurring on the Rx Free Descriptor/Buffer Queues. It does not support byte accesses. This register is shown in Figure 24-124 and described in Table 24-134.

Figure 24-124. Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (FDBSC2)

31 24	23 16	15 8	7 0
fdbq11_starve_cnt	fdbq10_starve_cnt	fdbq9_starve_cnt	fdbq8_starve_cnt
RC-0	RC-0	RC-0	RC-0

LEGEND: RC = Clear on read; -*n* = value after reset

Table 24-134. Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (FDBSC2) Field Descriptions

Bits	Field	Description
31-24	fdbq11_starve_cnt	This field increments each time the free descriptor/buffer queue 11 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
23-16	fdbq10_starve_cnt	This field increments each time the free descriptor/buffer queue 10 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
15-8	fdbq9_starve_cnt	This field increments each time the free descriptor/buffer queue 9 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
7-0	fdbq8_starve_cnt	This field increments each time the free descriptor/buffer queue 8 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.

24.9.5.6 Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (FDBSC3)

The queue manager free descriptor/buffer starvation count register 3 (FDBSC3) provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses. This register is shown in Figure 24-125 and described in Table 24-135.

Figure 24-125. Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (FDBSC3)

31 24	23 16	15 8	7 0
fdbq15_starve_cnt	fdbq14_starve_cnt	fdbq13_starve_cnt	fdbq12_starve_cnt
RC-0	RC-0	RC-0	RC-0

LEGEND: RC = Clear on read; -*n* = value after reset

Table 24-135. Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (FDBSC3) Field Descriptions

Bits	Field	Description
31-24	fdbq15_starve_cnt	This field increments each time the free descriptor/buffer queue 15 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
23-16	fdbq14_starve_cnt	This field increments each time the free descriptor/buffer queue 14 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
15-8	fdbq13_starve_cnt	This field increments each time the free descriptor/buffer queue 13 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
7-0	fdbq12_starve_cnt	This field increments each time the free descriptor/buffer queue 12 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.

24.9.5.7 Queue Manager Free Descriptor/Buffer Starvation Count Register 4 (FDBSC4)

The queue manager free descriptor/buffer starvation count register 4 (FDBSC4) provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses. This register is shown in Figure 24-126 and described in Table 24-136.

Figure 24-126. Queue Manager Free Descriptor/Buffer Starvation Count Register 4 (FDBSC4)

31 24	23 16	15 8	7 0
fdbq19_starve_cnt	fdbq18_starve_cnt	fdbq17_starve_cnt	fdbq16_starve_cnt
RC-0	RC-0	RC-0	RC-0

LEGEND: RC = Clear on read; -*n* = value after reset

Table 24-136. Queue Manager Free Descriptor/Buffer Starvation Count Register 4 (FDBSC4) Field Descriptions

Bits	Field	Description
31-24	fdbq19_starve_cnt	This field increments each time the free descriptor/buffer queue 19 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
23-16	fdbq18_starve_cnt	This field increments each time the free descriptor/buffer queue 18 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
15-8	fdbq17_starve_cnt	This field increments each time the free descriptor/buffer queue 17 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
7-0	fdbq16_starve_cnt	This field increments each time the free descriptor/buffer queue 16 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.

24.9.5.8 Queue Manager Free Descriptor/Buffer Starvation Count Register 5 (FDBSC5)

The queue manager free descriptor/buffer starvation count register 5 (FDBSC5) provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses. This register is shown in Figure 24-127 and described in Table 24-137.

Figure 24-127. Queue Manager Free Descriptor/Buffer Starvation Count Register 5 (FDBSC5)

31 24	23 16	15 8	7 0
fdbq23_starve_cnt	fdbq22_starve_cnt	fdbq21_starve_cnt	fdbq20_starve_cnt
RC-0	RC-0	RC-0	RC-0

LEGEND: RC = Clear on read; -*n* = value after reset

Table 24-137. Queue Manager Free Descriptor/Buffer Starvation Count Register 5 (FDBSC5) Field Descriptions

Bits	Field	Description
31-24	fdbq23_starve_cnt	This field increments each time the free descriptor/buffer queue 23 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
23-16	fdbq22_starve_cnt	This field increments each time the free descriptor/buffer queue 22 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
15-8	fdbq21_starve_cnt	This field increments each time the free descriptor/buffer queue 21 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
7-0	fdbq20_starve_cnt	This field increments each time the free descriptor/buffer queue 20 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.

24.9.5.9 Queue Manager Free Descriptor/Buffer Starvation Count Register 6 (FDBSC6)

The queue manager free descriptor/buffer starvation count register 6 (FDBSC6) provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses. This register is shown in Figure 24-128 and described in Table 24-138.

Figure 24-128. Queue Manager Free Descriptor/Buffer Starvation Count Register 6 (FDBSC6)

31 24	23 16	15 8	7 0
fdbq27_starve_cnt	fdbq26_starve_cnt	fdbq25_starve_cnt	fdbq24_starve_cnt
RC-0	RC-0	RC-0	RC-0

LEGEND: RC = Clear on read; -*n* = value after reset

Table 24-138. Queue Manager Free Descriptor/Buffer Starvation Count Register 6 (FDBSC6) Field Descriptions

Bits	Field	Description
31-24	fdbq27_starve_cnt	This field increments each time the free descriptor/buffer queue 27 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
23-16	fdbq26_starve_cnt	This field increments each time the free descriptor/buffer queue 26 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
15-8	fdbq25_starve_cnt	This field increments each time the free descriptor/buffer queue 25 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
7-0	fdbq24_starve_cnt	This field increments each time the free descriptor/buffer queue 24 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.

24.9.5.10 Queue Manager Free Descriptor/Buffer Starvation Count Register 7 (FDBSC7)

The queue manager free descriptor/buffer starvation count register 7 (FDBSC7) provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses. This register is shown in Figure 24-129 and described in Table 24-139.

Figure 24-129. Queue Manager Free Descriptor/Buffer Starvation Count Register 7 (FDBSC7)

31 24	23 16	15 8	7 0
fdbq31_starve_cnt	fdbq30_starve_cnt	fdbq29_starve_cnt	fdbq28_starve_cnt
RC-0	RC-0	RC-0	RC-0

LEGEND: RC = Clear on read; -*n* = value after reset

Table 24-139. Queue Manager Free Descriptor/Buffer Starvation Count Register 7 (FDBSC7) Field Descriptions

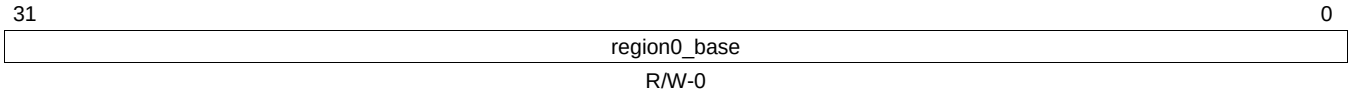
Bits	Field	Description
31-24	fdbq31_starve_cnt	This field increments each time the free descriptor/buffer queue 31 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
23-16	fdbq30_starve_cnt	This field increments each time the free descriptor/buffer queue 30 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
15-8	fdbq29_starve_cnt	This field increments each time the free descriptor/buffer queue 29 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.
7:0	fdbq28_starve_cnt	This field increments each time the free descriptor/buffer queue 28 is read while it is empty via the CPPI DMA. This field is cleared when read via the CPU.

24.9.5.11 Queue Manager Linking RAM Region 0 Base Address Register (LRAM0BASE)

The queue manager linking RAM region 0 base address register (LRAM0BASE) is used to set the base address for the first portion of the Linking RAM. This address must be 32-bit aligned. It is used by the queue manager to calculate the 32-bit linking address for a given descriptor index. It does not support byte accesses.

The queue manager linking RAM region 0 base address register is shown in Figure 24-130 and described in Table 24-140.

Figure 24-130. Queue Manager Linking RAM Region 0 Base Address Register (LRAM0BASE)



LEGEND: R/W = Read/Write; -*n* = value after reset

Table 24-140. Queue Manager Linking RAM Region 0 Base Address Register (LRAM0BASE) Field Descriptions

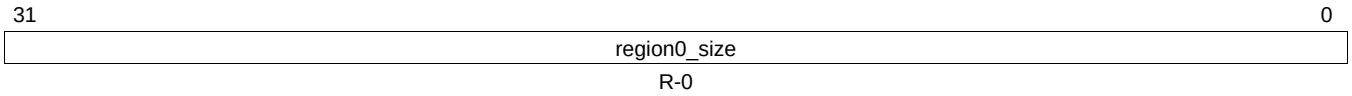
Bits	Field	Description
31-0	region0_base	This field stores the base address for the first region of the linking RAM. This may be anywhere in 32-bit address space but would be typically located in on-chip memory.

24.9.5.12 Queue Manager Linking RAM Region 0 Size Register (LRAM0SIZE)

The queue manager linking RAM region 0 size register (LRAM0SIZE) is used to set the size of the array of linking pointers that are located in region 0 of linking RAM. The size specified the number of descriptors for which linking information is stored in this region. It does not support byte accesses.

The queue manager linking RAM region 0 size register is shown in Figure 24-131 and described in Table 24-141.

Figure 24-131. Queue Manager Linking RAM Region 0 Size Register (LRAM0SIZE)



LEGEND: R = Read only; -*n* = value after reset

Table 24-141. Queue Manager Linking RAM Region 0 Size Register (LRAM0SIZE) Field Descriptions

Bits	Field	Description
31-0	region0_size	This field indicates the number of entries that are contained in the linking RAM region 0. A descriptor with index less than region0_size value has its linking location in region 0. A descriptor with index greater than region0_size has its linking location in region 1. The queue manager adds the index (left shifted by 2 bits) to the appropriate regionX_base_addr to get the absolute 32-bit address to the linking location for a descriptor.

24.9.5.13 Queue Manager Linking RAM Region 1 Base Address Register (LRAM1BASE)

The queue manager linking RAM region 1 base address register (LRAM1BASE) is used to set the base address for the second portion of the linking RAM. This base address is used by the queue manager to calculate the 32-bit linking address from the descriptor index. All descriptors with index higher than that given in linking RAM 0 size register have linking information stored in linking RAM region 1. It does not support byte accesses.

The queue manager linking RAM region 1 base address register is shown in Figure 24-132 and described in Table 24-142.

Figure 24-132. Queue Manager Linking RAM Region 1 Base Address Register (LRAM1BASE)

31	0
region1_base	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 24-142. Queue Manager Linking RAM Region 1 Base Address Register (LRAM1BASE) Field Descriptions

Bits	Field	Description
31-0	region1_base	This field stores the base address for the second region of the linking RAM. This may be anywhere in 32-bit address space but would be typically located in off-chip memory.

24.9.5.14 Queue Manager Queue Pending Register 0 (PEND0)

The queue manager queue pending register 0 (PEND0) can be read to find the pending status for queues 31 to 0. It does not support byte accesses. This register is shown in Figure 24-133 and described in Table 24-143.

Figure 24-133. Queue Manager Queue Pending Register 0 (PEND0)

31	0
qpend0	
R-0	

LEGEND: R = Read only; -*n* = value after reset

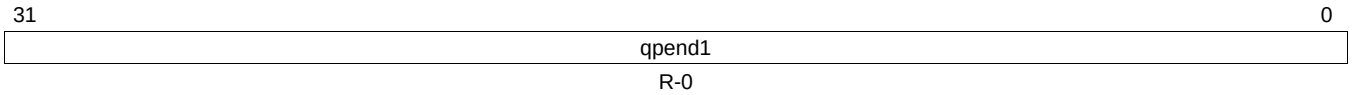
Table 24-143. Queue Manager Queue Pending Register 0 (PEND0) Field Descriptions

Bits	Field	Description
31-0	qpend0	This field indicates the queue pending status for queues 31-0

24.9.5.15 Queue Manager Queue Pending Register 1 (PEND1)

The queue manager queue pending register 1 (PEND1) can be read to find the pending status for queues 63 to 32. It does not support byte accesses. This register is shown in Figure 24-134 and described in Table 24-144.

Figure 24-134. Queue Manager Queue Pending Register 1 (PEND1)



LEGEND: R = Read only; -*n* = value after reset

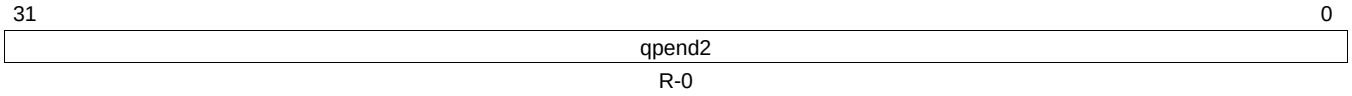
Table 24-144. Queue Manager Queue Pending Register 1 (PEND1) Field Descriptions

Bits	Field	Description
31-0	qpend1	This field indicates the queue pending status for queues 63-32

24.9.5.16 Queue Manager Queue Pending Register 2 (PEND2)

The queue manager queue pending register 2 (PEND2) can be read to find the pending status for queues 95 to 64. It does not support byte accesses. This register is shown in Figure 24-135 and described in Table 24-145.

Figure 24-135. Queue Manager Queue Pending Register 2 (PEND2)



LEGEND: R = Read only; -*n* = value after reset

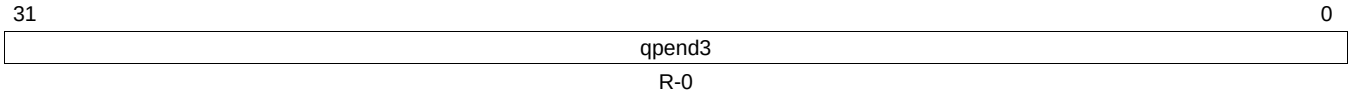
Table 24-145. Queue Manager Queue Pending Register 2 (PEND1) Field Descriptions

Bits	Field	Description
31-0	qpend2	This field indicates the queue pending status for queues 95-64

24.9.5.17 Queue Manager Queue Pending Register 3 (PEND3)

The queue manager queue pending register 3 (PEND3) can be read to find the pending status for queues 127 to 96. It does not support byte accesses. This register is shown in Figure 24-136 and described in Table 24-146.

Figure 24-136. Queue Manager Queue Pending Register 3 (PEND3)



LEGEND: R = Read only; -*n* = value after reset

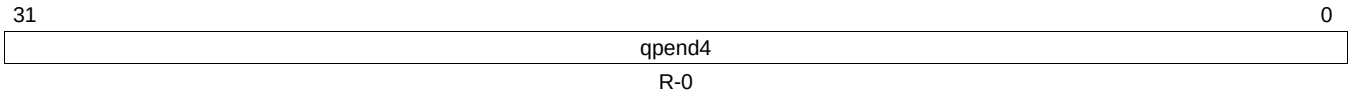
Table 24-146. Queue Manager Queue Pending Register 3 (PEND3) Field Descriptions

Bits	Field	Description
31-0	qpend3	This field indicates the queue pending status for queues 127-96

24.9.5.18 Queue Manager Queue Pending Register 4 (PEND4)

The queue manager queue pending register 4 (PEND4) can be read to find the pending status for queues 159 to 128. It does not support byte accesses. This register is shown in Figure 24-137 and described in Table 24-147.

Figure 24-137. Queue Manager Queue Pending Register 4 (PEND4)



LEGEND: R = Read only; -*n* = value after reset

Table 24-147. Queue Manager Queue Pending Register 4 (PEND4) Field Descriptions

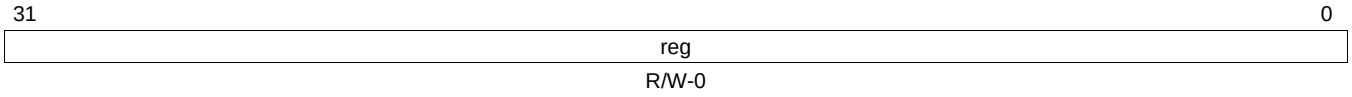
Bits	Field	Description
31-0	qpend4	This field indicates the queue pending status for queues 159-128

24.9.5.19 Queue Manager Memory Region R Base Address Register (QMEMRBASEr)

The queue manager memory region R base address register (QMEMRBASEr) is written by the Host to set the base address of memory region R. This memory region stores a number of descriptors of a particular size as determined by the memory region R control register. It does not support byte accesses. R ranges from 0 to 15.

The queue manager memory region R base address register is shown in Figure 24-138 and described in Table 24-148.

Figure 24-138. Queue Manager Memory Region R Base Address Register (QMEMRBASEr)



LEGEND: R/W = Read/Write; -*n* = value after reset

Table 24-148. Queue Manager Memory Region R Base Address Register (QMEMRBASEr) Field Descriptions

Bits	Field	Description
31-0	reg	This field contains the base address of the memory region R.

24.9.5.20 Queue Manager Memory Region R Control Register (QMEMRCTRLr)

The queue manager memory region R control register (QMEMRCTRLr) is written by the Host to configure various parameters of this memory region. It does not support byte accesses. R ranges from 0 to 15. This register is shown in Figure 24-139 and described in Table 24-149.

Figure 24-139. Queue Manager Memory Region R Control Register (QMEMRCTRLr)

31–30	29–16
Reserved	start_index
R-0	R/W-0

15–12	11–8	7–3	2–0
Reserved	desc_size	Reserved	reg_size
R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-149. Queue Manager Memory Region R Control Register (QMEMRCTRLr) Field Descriptions

Bits	Field	Description
31-30	Reserved	Reserved
29-16	start_index	This field indicates where in linking RAM does the descriptor linking information corresponding to memory region R starts.
15-12	Reserved	Reserved
11-08	desc_size	This field indicates the size of each descriptor in this memory region. It is an encoded value that specifies descriptor size as 2^(5+desc_size) number of bytes. The settings of desc_size from 9-15 are reserved.
7-3	Reserved	Reserved
2-0	reg_size	This field indicates the size of the memory region (in terms of number of descriptors). It is an encoded value that specifies region size as 2^(5+reg_size) number of descriptors.

24.9.5.21 Queue Manager Queue N Register A (CTRLAn)

The queue manager queue N register A (CTRLAn) is an optional register that is only implemented for a queue if the queue supports entry/byte count feature. The entry count feature provides a count of the number of entries that are currently valid in the queue. It does not support byte accesses. N ranges from 0 to 155.

The queue manager queue N register A is shown in Figure 24-140 and described in Table 24-150.

Figure 24-140. Queue Manager Queue N Register A (CTRLAn)

31–14	13–0
Reserved	queue_entry_count
R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 24-150. Queue Manager Queue N Register A (CTRLAn) Field Descriptions

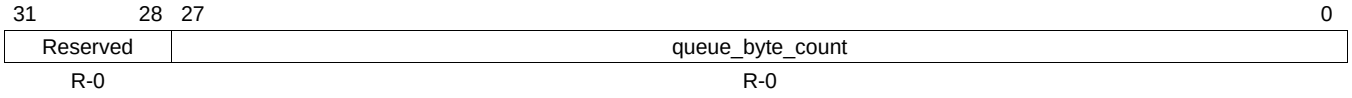
Bits	Field	Description
31-14	Reserved	Reserved
13-0	queue_entry_count	This field indicates how many packets are currently queued on the queue. This count is incremented by 1 whenever a packet is added to the queue. This count is decremented by 1 whenever a packet is popped from the queue.

24.9.5.22 Queue Manager Queue N Register B (CTRLBn)

The queue manager queue N register B (CTRLBn) is an optional register that is only implemented for a queue if the queue supports a total byte count feature. The total byte count feature provides a count of the total number of bytes in all of the packets that are currently valid in the queue. This register must be read prior to reading queue N register D during packet pop operation if the total size information is desired. It does not support byte accesses. N ranges from 0 to 155.

The queue manager queue N register B is shown in Figure 24-141 and described in Table 24-151.

Figure 24-141. Queue Manager Queue N Register B (CTRLBn)



LEGEND: R = Read only; -*n* = value after reset

Table 24-151. Queue Manager Queue N Register B (CTRLBn) Field Descriptions

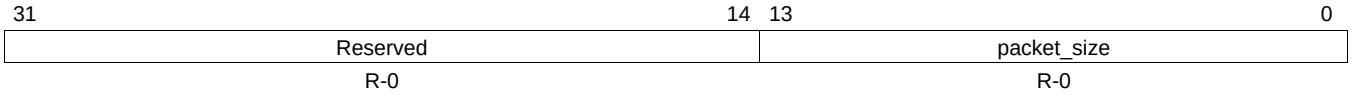
Bits	Field	Description
31-28	Reserved	Reserved
27-0	queue_byte_count	This field indicates how many bytes total are contained in all of the packets that are currently queued on this queue.

24.9.5.23 Queue Manager Queue N Register C (CTRLCn)

The queue manager queue N register C (CTRLCn) is used to provide additional information about the packet that is being pushed or popped from the queue. This register provides an option for the packet to be pushed onto either the tail of the queue (default) or the head of the queue (override). This register must be written prior to writing the queue N register D during packet write operations. This register must be read prior to reading queue N register D during pop operations if the packet size information is desired. It does not support byte accesses. N ranges from 0 to 155.

The queue manager queue N register C is shown in Figure 24-142 and described in Table 24-152.

Figure 24-142. Queue Manager Queue N Register C (CTRLCn)



LEGEND: R = Read only; -*n* = value after reset

Table 24-152. Queue Manager Queue N Register C (CTRLCn) Field Descriptions

Bits	Field	Description
31-14	Reserved	Reserved
13-0	packet_size	This field indicates packet size of the head element of a queue. This field indicates packet size for packet pop operation.

24.9.5.24 Queue Manager Queue N Register D (CTRLDn)

The queue manager queue N register D (CTRLDn) is written to add a packet to the queue and read to pop a packets off a queue. The packet is only pushed or popped to/from the queue when the queue register D is written. It does not support byte accesses. N ranges from 0 to 155.

The queue manager queue N register D is shown in Figure 24-143 and described in Table 24-153.

Figure 24-143. Queue Manager Queue N Register D (CTRLDn)

31	5	4	0
desc_ptr		desc_size	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 24-153. Queue Manager Queue N Register D (CTRLDn) Field Descriptions

Bits	Field	Description
31-5	desc_ptr	Descriptor pointer. It is read as zero if the queue is empty. It indicates a 32-bit aligned address that points to a descriptor when the queue is not empty.
4-0	desc_size	Descriptor Size. It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes. This field returns a 0x0 when an empty queue is read.

24.9.5.25 Queue Manager Queue N Status Register A (QSTATAn)

The queue manager queue N status register A (QSTATAn) is an optional register that is only implemented for a queue if the queue supports entry/byte count feature. The entry count feature provides a count of the number of entries that are currently valid in the queue. It does not support byte accesses. N ranges from 0 to 155.

The queue manager queue N status register A is shown in Figure 24-144 and described in Table 24-154.

Figure 24-144. Queue Manager Queue N Status Register A (QSTATAn)

31	14	13	0
Reserved		queue_entry_count	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 24-154. Queue Manager Queue N Status Register A (QSTATAn) Field Descriptions

Bits	Field	Description
31-14	Reserved	Reserved
13-0	queue_entry_count	This field indicates how many packets are currently queued on the queue.

24.9.5.26 Queue Manager Queue N Status Register B (QSTATBn)

The queue manager queue N status register B (QSTATBn) is an optional register that is only implemented for a queue if the queue supports a total byte count feature. The total byte count feature provides a count of the total number of bytes in all of the packets that are currently valid in the queue. It does not support byte accesses. N ranges from 0 to 155.

The queue manager queue N status register B is shown in Figure 24-145 and described in Table 24-155.

Figure 24-145. Queue Manager Queue N Status Register B (QSTATBn)

31 28	27 0
Reserved	queue_byte_count
R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 24-155. Queue Manager Queue N Status Register B (QSTATBn) Field Descriptions

Bits	Field	Description
31-28	Reserved	Reserved
27-0	queue_byte_count	This field indicates how many bytes total are contained in all of the packets that are currently queued on this queue.

24.9.5.27 Queue Manager Queue N Status Register C (QSTATCn)

The queue manager queue N status register C (QSTATCn) specifies the packet size for the head element of a queue. It does not support byte accesses. N ranges from 0 to 155. This register is shown in Figure 24-146 and described in Table 24-156.

Figure 24-146. Queue Manager Queue N Status Register C (QSTATCn)

31 14	13 0
Reserved	packet_size
R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 24-156. Queue Manager Queue N Status Register C (QSTATCn) Field Descriptions

Bits	Field	Description
31-14	Reserved	Reserved
13-0	packet_size	This field indicates packet size of the head element of a queue.

24.9.6 USB Mentor Core Registers

The Mentor Control register map is divided into the following sections:

- **Common USB registers (0h–Fh)** – These registers provide control and status for the complete core.
- **Indexed Endpoint Control/Status registers (10h–1Fh)** – These registers provide control and status for the currently selected endpoint. The registers mapped into this section depend on whether the core is in Peripheral mode or in Host mode and on the value of the Index register.
- **FIFOs (20h–5Fh)** – This address range provides access to the endpoint FIFOs.
- Additional Control and Configuration registers (60h–7Fh) – These registers provide additional device status and control.
- **Target Endpoint Control Registers (80h–FFh)** – These registers provide target function and hub address details for each of the endpoints.
- **Non-Indexed Endpoint Control/Status registers (100h and above)** – The registers available at 10h–1Fh, accessible independently of the setting of the Index register. 100h–10Fh, EP0 registers; 110h–11Fh, EP1 registers; 120h–12Fh, EP2; and so on.

Two instances of the mentor core registers exist within USB0 space. Since both USB modules operate independent of each other, each USB core has its own set of registers. USB0 mentor base address is 4740 1400h; USB1 mentor core registers base address is 4740 1C00h. The core registers the absolute address is computed by summing the CORE_OFFSET with USBn_OFFSET.

24.9.6.1 Common USB Registers

These registers provide control and status for the complete core. Table 24-157 lists the registers.

Table 24-157. Common USB Registers

Core Address Offset	Acronym	Common USB Registers
0h	USBn_FADDR	Function Address Register
1h	USBn_POWER	Power Management Register
2h	USBn_INTRTX	Interrupt Register for Endpoint 0 Plus Transmit Endpoints 1 to 15
4h	INTRRX	Interrupt Register for Receive Endpoints 1 to 15
6h	INTRTXE	Interrupt Enable Register for INTRTX
8h	INTRRXE	Interrupt Enable Register for INTRRX
Ah	USBn_INTRUSB	Interrupt Register for Common USB Interrupts
Bh	USBn_INTRUSBE	Interrupt Enable Register for INTRUSB
Ch	USBn_FRAME	Frame Number Register
Eh	USBn_INDEX	Index Register for Selecting the Endpoint Status and Control Registers
Fh	USBn_TESTMODE	Register to Enable the USB 2.0 Test Modes

24.9.6.1.1 Function Address Register (USBn_FADDR)

The function address register (USBn_FADDR) is an 8-bit register that should be written with the 7-bit address of the peripheral part of the transaction.

Since USB cores are configured with multipoint support, this register only applies to operations carried out when the controller is in peripheral mode. In Host mode, this register is ignored.

The function address register is shown in Figure 24-147 and described in Table 24-158.

Figure 24-147. Function Address Register (USBn_FADDR)

7	6	0
Reserved	FUNCADDR	
R-0	W-0	

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 24-158. Function Address Register (USBn_FADDR) Field Descriptions

Bits	Field	Description
7	Reserved	Reserved
6-0	FUNCADDR	7_bit address of the peripheral part of the transaction. When used in peripheral mode, this register should be written with the address received through a SET_ADDRESS command, which is then used for decoding the function address in subsequent token packets.

24.9.6.1.2 Power Management Register (USBn_POWER)

The power management register (USBn_POWER) is an 8-bit register that is used for controlling suspend and resume signaling, and some basic operational aspects of the USB core. This register is shown in Figure 24-148 and described in Table 24-159.

Figure 24-148. Power Management Register (USBn_POWER)

7	6	5	4	3	2	1	0
ISOUPDATE	SOFTCONN	HSEN	HSMODE	RESET	RESUME	SUSPENDM	ENSUSPM
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 24-159. Power Management Register (USBn_POWER) Field Descriptions

Bits	Field	Description
7	ISOUPDATE	When set, the USB controller will wait for an SOF token from the time TxPktRdy is set before sending the packet. If an IN token is received before an SOF token, then a zero length data packet will be sent. Note: this is only valid in Peripheral Mode. This bit only affects endpoints performing Isochronous transfers.
6	SOFTCONN	If Soft Connect/Disconnect feature is enabled, then the USB D+/D-lines are enabled when this bit is set and tri-stated when this bit is cleared. Note: This is only valid in Peripheral Mode.
5	HSEN	When set, the USB controller negotiates for high-speed mode when the device is reset by the hub. If not set, the device will only operate in full-speed mode.
4	HSMODE	This bit is set when the USB controller has successfully negotiated for high-speed mode.
3	RESET	This bit is set when reset signaling is present on the bus. Note: this bit is Read/Write in Host Mode, but read-only in peripheral mode.
2	RESUME	Set to generate resume signaling when the controller is in suspend mode. The bit should be cleared after 10 ms (a maximum of 15 ms) to end resume signaling. In Host mode, this bit is also automatically set when resume signaling from the target is detected while the USB controller is suspended.
1	SUSPENDM	In Host mode, this bit should be set to enter suspend mode. In peripheral mode, this bit is set on entry into suspend mode. It is cleared when the interrupt register is read, or the RESUME bit is set.
0	ENSUSPM	Set to enable the SUSPENDM output.

24.9.6.1.3 Interrupt Register for Endpoint 0 Plus Transmit Endpoints 1 to 15 (USBn_INTRTX)

The interrupt register for endpoint 0 plus transmit endpoints 1 to 15 (USBn_INTRTX) is a 16-bit read-only register that indicates which interrupts are currently active for endpoint 0 and the TX endpoints 1-15. Note also that all active interrupts are cleared when this register is read.

The interrupt register for endpoint 0 plus transmit endpoints 1 to 15 is shown in Figure 24-149 and described in Table 24-160.

Figure 24-149. Interrupt Register for Endpoint 0 Plus Transmit Endpoints 1 to 15 (USBn_INTRTX)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP15TX	EP14TX	EP13TX	EP12TX	EP11TX	EP10TX	EP9TX	EP8TX	EP7TX	EP6TX	EP5TX	EP4TX	EP3TX	EP2TX	EP1TX	EP0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

LEGEND: RC = Clear on read; -*n* = value after reset

Table 24-160. Interrupt Register for Endpoint 0 Plus Transmit Endpoints 1 to 15 (USBn_INTRTX) Field Descriptions

Bits	Field	Description
15-1	EP*n*TX	Transmit endpoint *n* interrupt active
0	EP0	Endpoint 0 (transmit or receive) interrupt active

24.9.6.1.4 Interrupt Register for Receive Endpoints 1 to 15 (INTRRX)

The interrupt register for receive endpoints 1 to 15 (INTRRX) is a 16-bit read-only register that indicates which of the interrupts for Rx Endpoints 1-15 are currently active. Note also that all active interrupts are cleared when this register is read.

The interrupt register for receive endpoints 1 to 15 is shown in Figure 24-150 and described in Table 24-161.

Figure 24-150. Interrupt Register for Receive Endpoints 1 to 15 (INTRRX)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP15RX	EP14RX	EP13RX	EP12RX	EP11RX	EP10RX	EP9RX	EP8RX	EP7RX	EP6RX	EP5RX	EP4RX	EP3RX	EP2RX	EP1RX	Rsvd
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

LEGEND: RC = Clear on read; -*n* = value after reset

Table 24-161. Interrupt Register for Receive Endpoints 1 to 15 (INTRRX) Field Descriptions

Bits	Field	Description
15-1	EP*n*RX	Receive endpoint *n* interrupt active
0	Reserved	Reserved

24.9.6.1.5 Interrupt Enable Register for INTRTX (INTRTXE)

The interrupt enable register for INTRTX (INTRTXE) is a 16-bit register that provides interrupt enable bits for the interrupts in IntrTx. On reset, the bits corresponding to endpoint 0 and the TX endpoints are set to 1. This register is shown in Figure 24-151 and described in Table 24-162.

Figure 24-151. Interrupt Enable Register for INTRTX (INTRTXE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP15TX	EP14TX	EP13TX	EP12TX	EP11TX	EP10TX	EP9TX	EP8TX	EP7TX	EP6TX	EP5TX	EP4TX	EP3TX	EP2TX	EP1TX	EP0
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 24-162. Interrupt Enable Register for INTRTX (INTRTXE) Field Descriptions

Bits	Field	Description
15-1	EP*n*TX	Transmit endpoint *n* interrupt enable
0	EP0	Endpoint 0 interrupt active

24.9.6.1.6 Interrupt Enable Register for INTRRX (INTRRXE)

The interrupt enable register for INTRRX (INTRRXE) is a 16-bit register that provides interrupt enable bits for the interrupts in USBn_INTRRX. On reset, the bits corresponding to the Rx endpoints are set to 1. This register is shown in Figure 24-152 and described in Table 24-163.

Figure 24-152. Interrupt Enable Register for INTRRX (INTRRXE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP15RX	EP14RX	EP13RX	EP12RX	EP11RX	EP10RX	EP9RX	EP8RX	EP7RX	EP6RX	EP5RX	EP4RX	EP3RX	EP2RX	EP1RX	Rsvd
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-163. Interrupt Enable Register for INTRRX (INTRRXE) Field Descriptions

Bits	Field	Description
15-1	EP*n*RX	Receive endpoint *n* interrupt enable
0	Reserved	Reserved

24.9.6.1.7 Interrupt Register for Common USB Interrupts (USBn_INTRUSB)

The interrupt register for common USB interrupts (USBn_INTRUSB) is an 8-bit read-only register that indicates which USB interrupts are currently active. All active interrupts will be cleared when this register is read. This register is shown in Figure 24-153 and described in Table 24-164.

Figure 24-153. Interrupt Register for Common USB Interrupts (USBn_INTRUSB)

7	6	5	4	3	2	1	0
VBUSERR	SESSREQ	DISCON	CONN	SOF	RESET_BABBLE	RESUME	SUSPEND
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

LEGEND: RC = Clear on read; -*n* = value after reset

Table 24-164. Interrupt Register for Common USB Interrupts (USBn_INTRUSB) Field Descriptions

Bits	Field	Description
7	VBUSERR	Set when VBus drops below the VBus valid threshold during a session. Only valid when the USB controller is 'A' device. All active interrupts will be cleared when this register is read.
6	SESSREQ	Set when session request signaling has been detected. Only valid when USB controller is 'A' device.
5	DISCON	Set in host mode when a device disconnect is detected. Set in peripheral mode when a session ends.
4	CONN	Set when a device connection is detected. Only valid in host mode.
3	SOF	Set when a new frame starts.
2	RESET_BABBLE	Set in peripheral mode when reset signaling is detected on the bus set in host mode when babble is detected.
1	RESUME	Set when resume signaling is detected on the bus while the USB controller is in suspend mode.
0	SUSPEND	Set when suspend signaling is detected on the bus only valid in peripheral mode.

24.9.6.1.8 Interrupt Enable Register for INTRUSB (USBn_INTRUSBE)

The interrupt enable register for INTRUSB (USBn_INTRUSBE) is an 8-bit register that provides interrupt enable bits for each of the interrupts in USBn_INTRUSB. This register is shown in Figure 24-154 and described in Table 24-165.

Figure 24-154. Interrupt Enable Register for INTRUSB (USBn_INTRUSBE)

7	6	5	4	3	2	1	0
VBUSERR	SESSREQ	DISCON	CONN	SOF	RESET_BABBLE	RESUME	SUSPEND
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-165. Interrupt Enable Register for INTRUSB (USBn_INTRUSBE) Field Descriptions

Bits	Field	Description
7	VBUSERR	Vbus error interrupt enable
6	SESSREQ	Session request interrupt enable
5	DISCON	Disconnect interrupt enable
4	CONN	Connect interrupt enable
3	SOF	Start of frame interrupt enable
2	RESET_BABBLE	Reset interrupt enable
1	RESUME	Resume interrupt enable
0	SUSPEND	Suspend interrupt enable

24.9.6.1.9 Frame Number Register (USBn_FRAME)

The frame number register (USBn_FRAME) is a 16-bit read-only register that holds the last received frame number. This register is shown in Figure 24-155 and described in Table 24-166.

Figure 24-155. Frame Number Register (USBn_FRAME)

15	11	10	0
Reserved		FRAMENUMBER	
R-0		R-7FFh	

LEGEND: R = Read only; -*n* = value after reset

Table 24-166. Frame Number Register (USBn_FRAME) Field Descriptions

Bits	Field	Description
15-11	Reserved	Reserved
10-0	FRAMENUMBER	Last received frame number

24.9.6.1.10 Index Register for Selecting the Endpoint Status and Control Registers (USBn_INDEX)

Each TX endpoint and each Rx endpoint have their own set of control/status registers located between Core_Offset 100h-1FFh. In addition one set of TX control/status and one set of Rx control/status registers appear at Core_Offset 10h-19h. This block of memory can be used as a proxy to access endpoint configuration registers. Index is a 4-bit register that determines which endpoint control/status registers are accessed.

Before accessing an endpoint's control/status registers at Core_Offset 10h-19h, the endpoint number should be written to the Index register to ensure that the correct control/status registers appear in the memory map.

The index register for selecting the endpoint status and control registers is shown in Figure 24-156 and described in Table 24-167.

Figure 24-156. Index Register for Selecting the Endpoint Status and Control Registers (USBn_INDEX)

7	4	3	0
Reserved		EPSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-167. Index Register for Selecting the Endpoint Status and Control Registers (USBn_INDEX) Field Descriptions

Bits	Field	Description
7-4	Reserved	Reserved
3-0	EPSEL	Each transmit endpoint and each receive endpoint have their own set of control/status registers. EPSEL determines which endpoint control/status registers are accessed. Before accessing an endpoint's control/status registers, the endpoint number should be written to the Index register to ensure that the correct control/status registers appear in the memory-map.

24.9.6.1.11 Register to Enable the USB 2.0 Test Modes (USBn_TESTMODE)

The register to enable the USB 2.0 test modes (USBn_TESTMODE) is an 8-bit register that is primarily used to put the USB controller into one of the four test modes for high-speed operation described in the USB 2.0 specification – in response to a SET FEATURE: TESTMODE command. It is not used in normal operation.

The register to enable the USB 2.0 test modes is shown in Figure 24-157 and described in Table 24-168.

Figure 24-157. Register to Enable the USB 2.0 Test Modes (USBn_TESTMODE)

7	6	5	4	3	2	1	0
FORCE_HOST	FIFO_ACCESS	FORCE_FS	FORCE_HS	TEST_PACKET	TEST_K	TEST_J	TEST_SE0_NAK
R/W-0	W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 24-168. Register to Enable the USB 2.0 Test Modes (USBn_TESTMODE) Field Descriptions

Bits	Field	Description
7	FORCE_HOST	Set this bit to forcibly put the USB controller into Host mode when SESSION bit is set, regardless of whether it is connected to any peripheral. The controller remains in Host mode until the session bit is cleared, even if a device is disconnected. And if the FORCE_HOST but remains set, it re-enters Host mode next time the SESSION bit is set. The operating speed is determined using the FORCE_HS and FORCE_FS bits.
6	FIFO_ACCESS	Set this bit to transfer the packet in EP0 Tx FIFO to EP0 Receive FIFO. It is cleared automatically.
5	FORCE_FS	Set this bit to force the USB controller into full-speed mode when it receives a USB reset.
4	FORCE_HS	Set this bit to force the USB controller into high-speed mode when it receives a USB reset.
3	TEST_PACKET	Set this bit to enter the Test_Packet test mode. In this mode, the USB controller repetitively transmits a 53-byte test packet on the bus, the form of which is defined in the Universal Serial Bus Specification Revision 2.0. Note: The test packet has a fixed format and must be loaded into the endpoint 0 FIFO before the test mode is entered.
2	TEST_K	Set this bit to enter the Test_K test mode. In this mode, the USB controller transmits a continuous K on the bus.
1	TEST_J	Set this bit to enter the Test_J test mode. In this mode, the USB controller transmits a continuous J on the bus.
0	TEST_SE0_NAK	Set this bit to enter the Test_SE0_NAK test mode. In this mode, the USB controller remains in high-speed mode, but responds to any valid IN token with a NAK.

24.9.6.2 Indexed Register Space

This is a block of the core register space that is used as a proxy to access the configuration and status registers of an endpoint. Mapping is done by programming the INDEX register with the endpoint number. Table 24-169 lists the registers.

Table 24-169. Indexed Region Registers

Core Address Offset	Acronym	Indexed Region Registers
10h	USBn_TXMAXP	Maximum Packet Size for Peripheral/Host Transmit Endpoint
12h	USBn_PERI_CSR0	Control Status Register for Endpoint 0 in Peripheral Mode
12h	USBn_HOST_CSR0	Control Status Register for Endpoint 0 in Host Mode
12h	USBn_PERI_TXCSR	Control Status Register for Peripheral Transmit Endpoint
12h	USBn_HOST_TXCSR	Control Status Register for Host Transmit Endpoint
14h	USBn_RXMAXP	Maximum Packet Size for Peripheral/Host Receive Endpoint
16h	USBn_PERI_RXCSR	Control Status Register for Peripheral Receive Endpoint
16h	USBn_HOST_RXCSR	Control Status Register for Host Receive Endpoint
18h	USBn_COUNT0	Count 0 Register
18h	USBn_RXCOUNT	Receive Count Register
1Ah	USBn_HOST_TYPE0	Type Register (Host mode only)
1Ah	USBn_HOST_TXTYPE	Transmit Type Register (Host mode only)
1Bh	USBn_HOST_NAKLIMIT0	NAKLimit0 Register (Host mode only)
1Bh	USBn_HOST_TXINTERVAL	Transmit Interval Register (Host mode only)
1Ch	USBn_HOST_RXTYPE	Receive Type Register (Host mode only)
1Dh	USBn_HOST_RXINTERVAL	Receive Interval Register (Host mode only)
1Fh	USBn_CONFIGDATA	Configuration Data Register

24.9.6.2.1 Maximum Packet Size for Peripheral/Host Transmit Endpoint Register (USBn_TXMAXP)

The maximum packet size for peripheral/host transmit endpoint register (USBn_TXMAXP) defines the maximum amount of data that can be transferred through the selected Tx endpoint in a single operation. There is a TXMAXP register for each Tx endpoint (except Endpoint 0). Endpoint 0 resource is hardened in design and max packet size is set to 64 bytes and no user configuration is required or does not exist.

The maximum packet size for peripheral/host transmit endpoint register is shown in Figure 24-158 and described in Table 24-170.

Figure 24-158. Maximum Packet Size for Peripheral/Host Transmit Endpoint Register (USBn_TXMAXP)

15	11	10	0
Reserved		MAXPAYLOAD	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-170. Maximum Packet Size for Peripheral/Host Transmit Endpoint (USBn_TXMAXP) Field Descriptions

Bits	Field	Description
15-11	Reserved	Reserved

Table 24-170. Maximum Packet Size for Peripheral/Host Transmit Endpoint (USBn_TXMAXP) Field Descriptions (continued)

Bits	Field	Description
10-0	MAXPAYLOAD	The maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes, but is subject to the constraints placed by the USB specification on packet sizes for bulk, interrupt, and isochronous transfers in full-speed and high-speed operations. The value written to this register should match the wMaxPacketSize field of the standard endpoint descriptor for the associated endpoint. A mismatch could cause unexpected results.

24.9.6.2.2 Control Status Register for Endpoint 0 in Peripheral Mode (USBn_PERI_CSR0)

The control status register for endpoint 0 in peripheral mode (USBn_PERI_CSR0) is a 16-bit register that provides control and status bits for endpoint 0 when USB controller assumes the role of a peripheral. This register is shown in Figure 24-159 and described in Table 24-171.

Figure 24-159. Control Status Register for Endpoint 0 in Peripheral Mode (USBn_PERI_CSR0)

15						9	8
Reserved							FLUSHFIFO
R-0							W-0
7	6	5	4	3	2	1	0
SERV_ SETUPEND	SERV_ RXPKTRDY	SENDSTALL	SETUPEND	DATAEND	SENTSTALL	TXPKTRDY	RXPKTRDY
W-0	W-0	W-0	R-0	W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 24-171. Control Status Register for Endpoint 0 in Peripheral Mode (USBn_PERI_CSR0) Field Descriptions

Bits	Field	Description
15-9	Reserved	Reserved
8	FLUSHFIFO	Set this bit to flush the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXPKTRDY/RXPKTRDY bit is cleared. Note: FLUSHFIFO has no effect unless TXPKTRDY/RXPKTRDY is set.
7	SERV_SETUPEND	Set this bit to clear the SETUPEND bit. It is cleared automatically.
6	SERV_RXPKTRDY	Set this bit to clear the RXPKTRDY bit. It is cleared automatically.
5	SENDSTALL	Set this bit to terminate the current transaction. The STALL handshake will be transmitted and then this bit will be cleared automatically.
4	SETUPEND	This bit will be set when a control transaction ends before the DATAEND bit has been set. An interrupt is generated, and the FIFO will be flushed at this time. The bit is cleared by the writing a 1 to the SERV_SETUPEND bit.
3	DATAEND	Set this bit to 1: • When setting TXPKTRDY for the last data packet. • When clearing RXPKTRDY after unloading the last data packet. • When setting TXPKTRDY for a zero length data packet. It is cleared automatically.
2	SENTSTALL	This bit is set when a STALL handshake is transmitted. This bit should be cleared.
1	TXPKTRDY	Set this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.
0	RXPKTRDY	This bit is set when a data packet has been received. An interrupt is generated when this bit is set. This bit is cleared by setting the SERV_RXPKTRDY bit.

24.9.6.2.3 Control Status Register for Endpoint 0 in Host Mode (USBn_HOST_CSR0)

The control status register for endpoint 0 in host mode (USBn_HOST_CSR0) is a 16-bit register that provides control and status bits for endpoint 0 when USB controller assumes the role of a host. This register is shown in Figure 24-160 and described in Table 24-172.

Figure 24-160. Control Status Register for Endpoint 0 in Host Mode (USBn_HOST_CSR0)

15	14	13	12	11	10	9	8
Reserved				DSPING	DATATOGWREN	DATATOG	FLUSHFIFO
R-0				R/W-0	W-0	R/W-0	W-0

7	6	5	4	3	2	1	0
NAK_TIMEOUT	STATUSPKT	REQPKT	ERROR	SETUPPKT	RXSTALL	TXPKTRDY	RXPKTRDY
W-0	R/W-0	R/W-0	W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 24-172. Control Status Register for Endpoint 0 in Host Mode (USBn_HOST_CSR0) Field Descriptions

Bits	Field	Description
15-12	Reserved	Reserved
11	DSPING	The CPU writes a 1 to the DSPING bit to instruct the core not to issue PING tokens in the data and status phases of a high-speed control transfer (for use with devices that do not respond to PING).
10	DATATOGWREN	Write 1 to this bit to enable the DATATOG bit to be written. This bit is automatically cleared once the new value is written to DATATOG.
9	DATATOG	When read, this bit indicates the current state of the EP0 data toggle. If DATATOGWREN is high, this bit can be written with the required setting of the data toggle. If DATATOGWREN is low, any value written to this bit is ignored.
8	FLUSHFIFO	Write 1 to this bit to flush the next packet to be transmitted/read from the Endpoint 0 FIFO. The FIFO pointer is reset and the TXPKTRDY/RXPKTRDY bit is cleared. Note: FLUSHFIFO has no effect unless TXPKTRDY/RXPKTRDY is set.
7	NAK_TIMEOUT	This bit will be set when endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the NAKLIMIT0 register. This bit should be cleared to allow the endpoint to continue.
6	STATUSPKT	Set this bit at the same time as the TXPKTRDY or REQPKT bit is set, to perform a status stage transaction. Setting this bit ensures that the data toggle is set so that a DATA1 packet is used for the Status Stage transaction.
5	REQPKT	Set this bit to request an IN transaction. It is cleared when RXPKTRDY is set.
4	ERROR	This bit is set when three attempts have been made to perform a transaction with no response from the peripheral. You should clear this bit. An interrupt is generated when this bit is set.
3	SETUPPKT	Set this bit, at the same time as the TXPKTRDY bit is set, to send a SETUP token instead of an OUT token for the transaction.
2	RXSTALL	This bit is set when a STALL handshake is received. You should clear this bit.
1	TXPKTRDY	Set this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.
0	RXPKTRDY	This bit is set when a data packet has been received. An interrupt is generated when this bit is set. Clear this bit by setting the SERV_RXPKTRDY bit.

24.9.6.2.4 Control Status Register for Peripheral Transmit Endpoint (USBn_PERI_TXCSR)

The control status register for peripheral transmit endpoint (USBn_PERI_TXCSR) is a 16-bit register that provides control and status bits for transfers through the currently-selected Tx endpoint when controller assumes the role of a peripheral. There is a TXCSR register for each configured Tx endpoint (not including Endpoint 0).

The control status register for peripheral transmit endpoint is shown in Figure 24-161 and described in Table 24-173.

Figure 24-161. Control Status Register for Peripheral Transmit Endpoint (USBn_PERI_TXCSR)

15	14	13	12	11	10	9	8
AUTOSET	ISO	MODE	DMAEN	FRCDATATOG	DMAMODE	Reserved	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	
7	6	5	4	3	2	1	0
Reserved	CLRDATATOG	SENTSTALL	SENDSTALL	FLUSHFIFO	UNDERRUN	FIFO NOTEMPTY	TXPKTRDY
R-0	W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 24-173. Control Status Register for Peripheral Transmit Endpoint (USBn_PERI_TXCSR) Field Descriptions

Bits	Field	Description
15	AUTOSET	DMA Mode: The CPU needs to set the AUTOSET bit prior to enabling the Tx DMA. CPU Mode: If the CPU sets the AUTOSET bit, the TXPKTRDY bit will be automatically set when data of the maximum packet size (value in TXMAXP) is loaded into the Tx FIFO. If a packet of less than the maximum packet size is loaded, then the TXPKTRDY bit will have to be set manually.
14	ISO	Set this bit to enable the Tx endpoint for Isochronous transfers, and clear it to enable the Tx endpoint for Bulk or Interrupt transfers.
13	MODE	Set this bit to enable the endpoint direction as Tx, and clear the bit to enable it as Rx. Note: This bit has any effect only where the same endpoint FIFO is used for both Transmit and Receive transactions.
12	DMAEN	Set this bit to enable the DMA request for the Tx endpoint.
11	FRCDATATOG	Set this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by Interrupt Tx endpoints that are used to communicate rate feedback for Isochronous endpoints.
10	DMAMODE	This bit should always be set to 1 when the DMA is enabled.
9-7	Reserved	Reserved
6	CLRDATATOG	Write a 1 to this bit to reset the endpoint data toggle to 0.
5	SENTSTALL	This bit is set automatically when a STALL handshake is transmitted. The FIFO is flushed and the TXPKTRDY bit is cleared. You should clear this bit.
4	SENDSTALL	Write a 1 to this bit to issue a STALL handshake to an IN token. Clear this bit to terminate the stall condition. Note: This bit has no effect where the endpoint is being used for Isochronous transfers.
3	FLUSHFIFO	Write a 1 to this bit to flush the next packet to be transmitted from the endpoint Tx FIFO. The FIFO pointer is reset and the TXPKTRDY bit is cleared. Note: FlushFIFO has no effect unless the TXPKTRDY bit is set. Also note that, if the FIFO is double-buffered, FlushFIFO may need to be set twice to completely clear the FIFO.
2	UNDERRUN	This bit is set automatically if an IN token is received when TXPKTRDY is not set. You should clear this bit.
1	FIFONOTEMPTY	This bit is set when there is at least 1 packet in the Tx FIFO. You should clear this bit.
0	TXPKTRDY	Set this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.

24.9.6.2.5 Control Status Register for Host Transmit Endpoint (USBn_HOST_TXCSR)

The control status register for host transmit endpoint (USBn_HOST_TXCSR) is a 16-bit register that provides control and status bits for transfers through the currently-selected Tx endpoint when controller assumes the role of a host. There is a TXCSR register for each configured Tx endpoint (not including Endpoint 0).

The control status register for host transmit endpoint is shown in Figure 24-162 and described in Figure 24-162.

Figure 24-162. Control Status Register for Host Transmit Endpoint (USBn_HOST_TXCSR)

15	14	13	12	11	10	9	8
AUTOSET	Reserved	MODE	DMAEN	FRCDATATOG	DMAMODE	DATATOG WREN	DATATOG
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	W-0	R/W-0
7	6	5	4	3	2	1	0
NAK_TIMEOUT	CLRDATATOG	RXSTALL	SETUPPKT	FLUSHFIFO	ERROR	FIFO NOTEMPTY	TXPKTRDY
R/W-0	W-0	R/W-0	R/W-0	W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 24-174. Control Status Register for Host Transmit Endpoint (USBn_HOST_TXCSR) Field Descriptions

Bits	Field	Description
15	AUTOSET	DMA Mode: The CPU needs to set the AUTOSET bit prior to enabling the Tx DMA. CPU Mode: If the CPU sets the AUTOSET bit, the TXPKTRDY bit will be automatically set when data of the maximum packet size (value in TXMAXP) is loaded into the Tx FIFO. If a packet of less than the maximum packet size is loaded, then the TXPKTRDY bit will have to be set manually.
14	Reserved	Reserved
13	MODE	Set this bit to enable the endpoint direction as Tx, and clear the bit to enable it as Rx. Note: This bit has any effect only where the same endpoint FIFO is used for both Transmit and Receive transactions.
12	DMAEN	Set this bit to enable the DMA request for the Tx endpoint.
11	FRCDATATOG	Set this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by Interrupt Tx endpoints that are used to communicate rate feedback for Isochronous endpoints.
10	DMAMODE	This bit should always be set to 1 when the DMA is enabled.
9	DATATOGWREN	Write 1 to this bit to enable the DATATOG bit to be written. This bit is automatically cleared once the new value is written to DATATOG.
8	DATATOG	When read, this bit indicates the current state of the Tx EP data toggle. If DATATOGWREN is high, this bit can be written with the required setting of the data toggle. If DATATOGWREN is low, any value written to this bit is ignored.
7	NAK_TIMEOUT	This bit will be set when the Tx endpoint is halted following the receipt of NAK responses for longer than the time set as the NAKLIMIT by the TXINTERVAL register. It should be cleared to allow the endpoint to continue. Note: This is valid only for Bulk endpoints.
6	CLRDATATOG	Write a 1 to this bit to reset the endpoint data toggle to 0.
5	RXSTALL	This bit is set when a STALL handshake is received. The FIFO is flushed and the TXPKTRDY bit is cleared (see below). You should clear this bit.
4	SETUPPKT	Set this bit at the same time as TXPKTRDY is set, to send a SETUP token instead of an OUT token for the transaction. Note: Setting this bit also clears the DATATOG bit.
3	FLUSHFIFO	Write a 1 to this bit to flush the next packet to be transmitted from the endpoint Tx FIFO. The FIFO pointer is reset and the TXPKTRDY bit (below) is cleared. Note: FlushFIFO has no effect unless the TXPKTRDY bit is set. Also note that, if the FIFO is double-buffered, FLUSHFIFO may need to be set twice to completely clear the FIFO.
2	ERROR	The USB controller sets this bit when 3 attempts have been made to send a packet and no handshake packet has been received. You should clear this bit. An interrupt is generated when the bit is set. This is valid only when the endpoint is operating in Bulk or Interrupt mode.
1	FIFONOTEMPTY	The USB controller sets this bit when there is at least 1 packet in the Tx FIFO.

Table 24-174. Control Status Register for Host Transmit Endpoint (USBn_HOST_TXCSR) Field Descriptions (continued)

Bits	Field	Description
0	TXPKTRDY	Set this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.

24.9.6.2.6 Maximum Packet Size for Peripheral/Host Receive Endpoint Register (USBn_RXMAXP)

The maximum packet size for peripheral/host receive endpoint register (USBn_RXMAXP) defines the maximum amount of data that can be transferred through the selected Rx endpoint in a single operation. There is a RXMAXP register for each Rx endpoint (except Endpoint 0). Endpoint 0 resource is hardened in design and max packet size is set to 64 bytes and no user configuration is required or does not exist.

The maximum packet size for peripheral/host receive endpoint register is shown in Figure 24-163 and described in Table 24-175.

Figure 24-163. Maximum Packet Size for Peripheral/Host Receive Endpoint Register (USBn_RXMAXP)

15	11	10	0
Reserved		MAXPAYLOAD	
R-0		RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-175. Maximum Packet Size for Peripheral Host Receive Endpoint (USBn_RXMAXP) Field Descriptions

Bit	Field	Description
15-11	Reserved	Reserved
10-0	MAXPAYLOAD	Defines the maximum amount of data that can be transferred through the selected Receive endpoint in a single frame/microframe (high-speed transfers). The value set can be up to 1024 bytes, but is subject to the constraints placed by the USB specification on packet sizes for bulk, interrupt, and isochronous transfers in full-speed and high-speed operations. The value written to this register should match the wMaxPacketSize field of the standard endpoint descriptor for the associated endpoint. A mismatch could cause unexpected results.

24.9.6.2.7 Control Status Register for Peripheral Receive Endpoint (USBn_PERI_RXCSR)

The control status register for peripheral receive endpoint (USBn_PERI_RXCSR) is a 16-bit register that provides control and status bits for transfers through the currently-selected Tx endpoint when controller assumes the role of a peripheral. There is a RXCSR register for each configured Rx endpoint (not including endpoint 0).

The control status register for peripheral receive endpoint is shown in Figure 24-164 and described in Table 24-176.

Figure 24-164. Control Status Register for Peripheral Receive Endpoint (USBn_PERI_RXCSR)

15	14	13	12	11	10	9	8
AUTOCLEAR	ISO	DMAEN	DISNYET	DMAMODE	DATATOG WREN	DATATOG	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	W-0	R/W-0	R-0
7	6	5	4	3	2	1	0
CLRDATATOG	SENTSTALL	SENDSTALL	FLUSHFIFO	DATAERROR	OVERRUN	FIFOFULL	RXPKTRDY
W-0	R/W-0	R/W-0	W-0	R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 24-176. Control Status Register for Peripheral Receive Endpoint (USBn_PERI_RXCSR) Field Descriptions

Bit	Field	Description
15	AUTOCLEAR	DMA Mode: The CPU sets the AUTOCLEAR bit prior to enabling the Rx DMA. CPU Mode: If the CPU sets the AUTOCLEAR bit, then the RXPKTRDY bit will be automatically cleared when a packet of RXMAXP bytes has been unloaded from the Receive FIFO. When packets of less than the maximum packet size are unloaded, RXPKTRDY will have to be cleared manually.
14	ISO	Set this bit to enable the Receive endpoint for Isochronous transfers, and clear it to enable the Receive endpoint for bulk/interrupt transfers.
13	DMAEN	Set this bit to enable the DMA request for the Receive endpoints.
12	DISNYET	DISNYET: Applies only for bulk/interrupt transactions: The CPU sets this bit to disable the sending of NYET handshakes. When set, all successfully received Rx packets are ACK'd including at the point at which the FIFO becomes full. Note: This bit only has any effect in high-speed mode, in which mode it should be set for all Interrupt endpoints. PID_ERROR: Applies only for ISO Transactions: The core sets this bit to indicate a PID error in the received packet.
11	DMAMODE	Always clear this bit to 0.
10	DATATOGWREN	Write 1 to this bit to enable the DATATOG bit to be written. This bit is automatically cleared once the new value is written to DATATOG.
9	DATATOG	When read, this bit indicates the current state of the Receive EP data toggle. If DATATOGWREN is high, this bit can be written with the required setting of the data toggle. If DATATOGWREN is low, any value written to this bit is ignored.
8	Reserved	Reserved
7	CLRDATATOG	Write a 1 to this bit to reset the endpoint data toggle to 0.
6	SENTSTALL	This bit is set when a STALL handshake is transmitted. The FIFO is flushed and the TXPKTRDY bit is cleared. You should clear this bit.
5	SENDSTALL	Write a 1 to this bit to issue a STALL handshake. Clear this bit to terminate the stall condition. Note: This bit has no effect where the endpoint is being used for Isochronous transfers.
4	FLUSHFIFO	Write a 1 to this bit to flush the next packet to be read from the endpoint Receive FIFO. The FIFO pointer is reset and the RXPKTRDY bit is cleared. Note: FLUSHFIFO has no effect unless RXPKTRDY is set. Also note that, if the FIFO is double-buffered, FLUSHFIFO may need to be set twice to completely clear the FIFO.
3	DATAERROR	This bit is set when RXPKTRDY is set if the data packet has a CRC or bit-stuff error. It is cleared when RXPKTRDY is cleared. Note: This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.
2	OVERRUN	This bit is set if an OUT packet cannot be loaded into the Receive FIFO. You should clear this bit. Note: This bit is only valid when the endpoint is operating in ISO mode. In bulk mode, it always returns zero.
1	FIFOFULL	This bit is set when no more packets can be loaded into the Receive FIFO.

Table 24-176. Control Status Register for Peripheral Receive Endpoint (USBn_PERI_RXCSR) Field Descriptions (continued)

Bit	Field	Description
0	RXPKTRDY	This bit is set when a data packet has been received. You should clear this bit when the packet has been unloaded from the Receive FIFO. An interrupt is generated when the bit is set.

24.9.6.2.8 Control Status Register for Host Receive Endpoint (USBn_HOST_RXCSR)

The control status register for host receive endpoint (USBn_HOST_RXCSR) is a 16-bit register that provides control and status bits for transfers through the currently-selected Rx endpoint when controller assumes the role of a host. There is a RXCSR register for each configured Rx endpoint (not including endpoint 0).

The control status register for host receive endpoint is shown in Figure 24-165 and described in Table 24-177.

Figure 24-165. Control Status Register for Host Receive Endpoint (USBn_HOST_RXCSR)

15	14	13	12	11	10	9	8
AUTOCLEAR	AUTOREQ	DMAEN	DISNYET	DMAMODE	DATATOG WREN	DATATOG	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	W-0	R/W-0	R-0
7	6	5	4	3	2	1	0
CLRDATATOG	RXSTALL	REQPKT	FLUSHFIFO	DATAERR_ NAKTIMEOUT	ERROR	FIFOFULL	RXPKTRDY
W-0	R/W-0	R/W-0	W-0	R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 24-177. Control Status Register for Host Receive Endpoint (USBn_HOST_RXCSR) Field Descriptions

Bit	Field	Description
15	AUTOCLEAR	DMA Mode: The CPU sets the AUTOCLEAR bit prior to enabling the Rx DMA. CPU Mode: If the CPU sets the AUTOCLEAR bit, then the RXPKTRDY bit will be automatically cleared when a packet of RXMAXP bytes has been unloaded from the Receive FIFO. When packets of less than the maximum packet size are unloaded, RXPKTRDY will have to be cleared manually.
14	AUTOREQ	If the CPU sets the AUTOREQ bit, then the REQPKT bit will be automatically set when the RXPKTRDY bit is cleared. Note: This bit is automatically cleared when a short packet is received.
13	DMAEN	Set this bit to enable the DMA request for the Receive endpoints.
12	DISNYET	Set this bit to disable the sending of NYET handshakes. When set, all successfully received Receive packets are ACKED including at the point at which the FIFO becomes full. Note: This bit only has any effect in high-speed mode, in which mode it should be set for all Interrupt endpoints.
11	DMAMODE	Always clear this bit to 0.
10	DATATOGWREN	Write 1 to this bit to enable the DATATOG bit to be written. This bit is automatically cleared once the new value is written to DATATOG.
9	DATATOG	When read, this bit indicates the current state of the Receive EP data toggle. If DATATOGWREN is high, this bit can be written with the required setting of the data toggle. If DATATOGWREN is low, any value written to this bit is ignored.
8	Reserved	Reserved
7	CLRDATATOG	Write a 1 to this bit to reset the endpoint data toggle to 0.
6	RXSTALL	When a STALL handshake is received, this bit is set and an interrupt is generated. You should clear this bit.
5	REQPKT	Write a 1 to this bit to request an IN transaction. It is cleared when RXPKTRDY is set.

Table 24-177. Control Status Register for Host Receive Endpoint (USBn_HOST_RXCSR) Field Descriptions (continued)

Bit	Field	Description
4	FLUSHFIFO	Write a 1 to this bit to flush the next packet to be read from the endpoint Receive FIFO. The FIFO pointer is reset and the RXPKTRDY bit is cleared. Note: FLUSHFIFO has no effect unless RXPKTRDY is set. Also note that, if the FIFO is double-buffered, FLUSHFIFO may need to be set twice to completely clear the FIFO.
3	DATAERR_NAKTIMEOUT	When operating in ISO mode, this bit is set when RXPKTRDY is set if the data packet has a CRC or bit-stuff error and cleared when RXPKTRDY is cleared. In Bulk mode, this bit will be set when the Receive endpoint is halted following the receipt of NAK responses for longer than the time set as the NAK Limit by the RXINTERVAL register. You should clear this bit to allow the endpoint to continue.
2	ERROR	The USB controller sets this bit when 3 attempts have been made to receive a packet and no data packet has been received. You should clear this bit. An interrupt is generated when the bit is set. Note: This bit is only valid when the transmit endpoint is operating in Bulk or Interrupt mode. In ISO mode, it always returns zero.
1	FIFOFULL	This bit is set when no more packets can be loaded into the Receive FIFO.
0	RXPKTRDY	This bit is set when a data packet has been received. You should clear this bit when the packet has been unloaded from the Receive FIFO. An interrupt is generated when the bit is set.

24.9.6.2.9 Count 0 Register (USBn_COUNT0)

The count 0 register (USBn_COUNT0) is a 7-bit read-only register that indicates the number of received data bytes in the Endpoint 0 FIFO. The value returned changes as the contents of the FIFO change and is only valid while the RXPKTRDY bit in the control status register for host receive endpoint (USBn_HOST_RXCSR) is set. This register is shown in Figure 24-166 and described in Table 24-178.

Figure 24-166. Count 0 Register (USBn_COUNT0)

15	7	6	0
Reserved		EP0RXCOUNT	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 24-178. Count 0 Register (USBn_COUNT0) Field Descriptions

Bit	Field	Description
15-7	Reserved	Reserved
6-0	EP0RXCOUNT	Indicates the number of received data bytes in the endpoint 0 FIFO. The value returned changes as the contents of the FIFO change and is only valid while RXPKTRDY of USBn_PERI_CSR0 or USBn_HOST_CSR0 is set.

24.9.6.2.10 Receive Count Register (USBn_RXCOUNT)

The receive count register (USBn_RXCOUNT) is a 13-bit read-only register that holds the number of received data bytes in the packet currently in line to be read from the Rx FIFO for endpoints other than endpoint 0. The value returned changes as the FIFO is unloaded and is only valid while the RXPKTRDY bit in the control status register for host receive endpoint (USBn_HOST_RXCSR) is set.

The receive count register is shown in Figure 24-167 and described in Table 24-179.

Figure 24-167. Receive Count Register (USBn_RXCOUNT)

15	13	12	0
Reserved		EPRXCOUNT	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 24-179. Receive Count Register (USBn_RXCOUNT) Field Descriptions

Bit	Field	Description
15-13	Reserved	Reserved
12-0	EPRXCOUNT	Holds the number of received data bytes in the packet in the Receive FIFO. The value returned changes as the contents of the FIFO change and is only valid while RXPKTRDY of PERI_RXCSR or HOST_RXCSR is set.

24.9.6.2.11 Type Register (Host mode only) (USBn_HOST_TYPE0)

The type register (Host mode only) (USBn_HOST_TYPE0) is an 8-bit register, meaningful only when controller assumes the role of a host, of which only bits 6 and 7 are implemented. These bits should be written with the operating speed of the targeted device. This register is shown in Figure 24-168 and described in Table 24-180.

Figure 24-168. Type Register (Host mode only) (USBn_HOST_TYPE0)

7	6	5	0
SPEED		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-180. Type Register (Host mode only) (USBn_HOST_TYPE0) Field Descriptions

Bit	Field	Value	Description
7-6	SPEED		Operating Speed of Target Device
		0	Reserved
		1h	High
		2h	Full
		3h	Low
5-0	Reserved	0	Reserved

24.9.6.2.12 Transmit Type Register (Host mode only) (USBn_HOST_TXTYPE)

The transmit type register (Host mode only) (USBn_HOST_TXTYPE) is an 8-bit register that should be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently-selected Tx endpoint, and its operating speed. There is a TXTYPE register for each configured Tx endpoint (except Endpoint 0).

The transmit type register is shown in Figure 24-169 and described in Table 24-181.

Figure 24-169. Transmit Type Register (Host mode only) (USBn_HOST_TXTYPE)

7	6	5	4	3	0
SPEED		PROT		TENDPN	
R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 24-181. Transmit Type Register (Host mode only) (USBn_HOST_TXTYPE) Field Descriptions

Bit	Field	Value	Description
7-6	SPEED		Operating Speed of Target Device
		0	Reserved
		1h	High
		2h	Full
		3h	Low
5-4	PROT		Set this to select the required protocol for the transmit endpoint
		0	Control
		1h	Isochronous
		2h	Bulk
		3h	Interrupt
3-0	TENDPN	0-Fh	Set this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during device enumeration.

24.9.6.2.13 NAKLimit0 Register (Host mode only) (USBn_HOST_NAKLIMIT0)

The NAKLIMIT0 register (Host mode only) (USBn_HOST_NAKLIMIT0) is a 5-bit register that sets the number of frames/microframes (high-speed transfers) after which endpoint 0 should timeout on receiving a stream of NAK responses. (Equivalent settings for other endpoints can be made through their TXINTERVAL and RXINTERVAL registers.) The number of frames/microframes selected is 2(m - 1) (where m is the value set in the register, valid values 2-16). If the host receives NAK responses from the target for more frames than the number represented by the Limit set in this register, the endpoint will be halted. Note: A value of 0 or 1 disables the NAK timeout function.

The NAKLIMIT0 register is shown in Figure 24-170 and described in Table 24-182.

Figure 24-170. NAKLimit0 Register (Host mode only) (USBn_HOST_NAKLIMIT0)

7	5	4	0
Reserved		EP0NAKLIMIT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-182. NAKLimit0 Register (Host mode only) (USBn_HOST_NAKLIMIT0) Field Descriptions

Bit	Field	Description
7-5	Reserved	Reserved
4-0	EP0NAKLIMIT	Sets the number of frames/microframes (high-speed transfers) after which endpoint 0 should time out on receiving a stream of NAK responses. The number of frames/microframes selected is 2(-1) (where m is the value set in the register, valid values 2-16). If the host receives NAK responses from the target for more frames than the number represented by the Limit set in this register, the endpoint will be halted. Note: A value of 0 or 1 disables the NAK timeout function.

24.9.6.2.14 Transmit Interval Register (Host mode only) (USBn_HOST_TXINTERVAL)

The transmit interval register (Host mode only) (USBn_HOST_TXINTERVAL) is an 8-bit register (meaningful only when controller assumes the role of a host) that, for interrupt and isochronous transfers, defines the polling interval for the currently-selected Tx endpoint. For bulk endpoints, this register sets the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses. There is a TXINTERVAL register for each configured Tx endpoint (except endpoint 0).

The transmit interval register is shown in Figure 24-171 and described in Table 24-183.

Figure 24-171. Transmit Interval Register (Host mode only) (USBn_HOST_TXINTERVAL)

7							0
POLINTVL_NAKLIMIT							
R/W-0							

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 24-183. Transmit Interval Register (Host mode only) (USBn_HOST_TXINTERVAL) Field Descriptions

Bit	Field	Description			
7-0	POLINTVL_ NAKLIMIT	For interrupt and isochronous transfers, defines the polling interval for the currently-selected transmit endpoint. For bulk endpoints, sets the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses. There is a transmit interval register for each configured transmit endpoint (except endpoint 0). In each case, the value that is set defines a number of frames/microframes (high-speed transfers), as follows:			
		Transfer Type	Speed	Valid values (m)	Interpretation
		Interrupt	Low Speed or Full Speed	1-255	Polling interval is m frames
				1-16	Polling interval is 2(-1) microframes
		Isochronous	Full Speed or High Speed	1-16	Polling interval is 2(-1) frames/microframes
		Bulk	Full Speed or High Speed	2-16	NAK Limit is 2(-1)frames/microframes
		Note: A value of 0 or 1 disables the NAK timeout function.			

24.9.6.2.15 Receive Type Register (Host mode only) (USBn_HOST_RXTYPE)

The receive type register (Host mode only) (USBn_HOST_RXTYPE) is an 8-bit register that should be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently-selected Rx endpoint, and its operating speed. There is a RXTYPE register for each configured Rx endpoint (except endpoint 0).

The receive type register is shown in Figure 24-172 and described in Figure 24-172.

Figure 24-172. Receive Type Register (Host mode only) (USBn_HOST_RXTYPE)

7	6	5	4	3	0
SPEED		PROT		RENDPN	
R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 24-184. Receive Type Register (Host mode only) (USBn_HOST_RXTYPE) Field Descriptions

Bits	Field	Value	Description
7-6	SPEED		Operating Speed of Target Device
		0	Reserved
		1h	High
		2h	Full
		3h	Low
5-4	PROT		Set this to select the required protocol for the transmit endpoint
		0	Control
		1h	Isochronous
		2h	Bulk
		3h	Interrupt
3-0	RENDPN	0-Fh	Set this value to the endpoint number contained in the Receive endpoint descriptor returned to the USB controller during device enumeration

24.9.6.2.16 Receive Interval Register (Host mode only) (USBn_HOST_RXINTERVAL)

The receive interval register (Host mode only) (USBn_HOST_RXINTERVAL) is an 8-bit register (only meaningful in host mode only) that, for Interrupt and Isochronous transfers, defines the polling interval for the currently-selected Rx endpoint. For bulk endpoints, this register sets the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses. There is a RXINTERVAL register for each configured Rx endpoint (except endpoint 0).

The receive interval register is shown in Figure 24-173 and described in Table 24-185.

Figure 24-173. Receive Interval Register (Host mode only) (USBn_HOST_RXINTERVAL)

7	0
POLINTVL_NAKLIMIT	
R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 24-185. Transmit Interval Register (Host mode only) (USBn_HOST_RXINTERVAL) Field Descriptions

Bit	Field	Description			
7-0	POLINTVL_ NAKLIMIT	For interrupt and isochronous transfers, defines the polling interval for the currently-selected transmit endpoint. For bulk endpoints, sets the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses. There is a transmit interval register for each configured transmit endpoint (except endpoint 0). In each case, the value that is set defines a number of frames/microframes (high-speed transfers), as follows:			
		Transfer Type	Speed	Valid values (m)	Interpretation
		Interrupt	Low Speed or Full Speed	1-255	Polling interval is m frames
			High Speed	1-16	Polling interval is 2(-1) microframes
		Isochronous	Full Speed or High Speed	1-16	Polling interval is 2(-1) frames/microframes
		Bulk	Full Speed or High Speed	2-16	NAK Limit is 2(-1)frames/microframes
		Note: A value of 0 or 1 disables the NAK timeout function.			

24.9.6.2.17 Configuration Data Register (USBn_CONFIGDATA)

The configuration data register (USBn_CONFIGDATA) is an 8-bit read-only register that returns information about the selected core configuration. This register information can only be accessed from Indexed Region and INDEX has to be programmed, with zero, selecting endpoint 0. This register is shown in Figure 24-174 and described in Table 24-186.

Figure 24-174. Configuration Data Register (USBn_CONFIGDATA)

7	6	5	4	3	2	1	0
MPRXE	MPTXE	BIGENDIAN	HBRXE	HBTXE	DYNFIFO	SOFTCONE	UTMI DATA WIDTH
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 24-186. Configuration Data Register (USBn_CONFIGDATA) Field Descriptions

Bit	Field	Value	Description
7	MPRXE		Indicates automatic amalgamation of bulk packets. (INDEX register needs to be cleared to 0.)
		0	Automatic amalgamation of bulk packets is not selected.
		1	Automatic amalgamation of bulk packets is selected.
6	MPTXE		Indicates automatic splitting of bulk packets. (INDEX register needs to be cleared to 0.)
		0	Automatic splitting of bulk packets is not selected.
		1	Automatic splitting of bulk packets is selected.
5	BIGENDIAN		Indicates endian ordering. (INDEX register needs to be cleared to 0.)
		0	Little-endian ordering is selected.
		1	Big-endian ordering is selected.
4	HBRXE		Indicates high-bandwidth Rx ISO endpoint support. (INDEX register needs to be cleared to 0.)
		0	High-bandwidth Rx ISO endpoint support is not selected.
		1	High-bandwidth Rx ISO endpoint support is selected.
3	HBTXE		Indicates high-bandwidth Tx ISO endpoint support. (INDEX register needs to be cleared to 0.)
		0	High-bandwidth Tx ISO endpoint support is not selected.
		1	High-bandwidth Tx ISO endpoint support is selected.
2	DYNFIFO		Indicates dynamic FIFO sizing. (INDEX register needs to be cleared to 0.)
		0	Dynamic FIFO sizing option is not selected.
		1	Dynamic FIFO sizing option is selected.
1	SOFTCONE		Indicates soft connect/disconnect. (INDEX register needs to be cleared to 0.)
		0	Soft connect/disconnect option is not selected.
		1	Soft connect/disconnect option is selected.
0	UTMIDATAWIDTH		Indicates selected UTMI data width. (INDEX register needs to be cleared to 0.)
		0	8 bits
		1	16 bits

24.9.7 FIFOs

This address range provides access to the endpoint FIFOs. Register offset addresses for this block is 20h to 5Fh.

24.9.7.1 Transmit and Receive FIFO Register for Endpoint 0 - 15 (USBn_FIFO0 – USBn_FIFO15)

This address range provides 16 addresses for CPU access to the FIFOs for each endpoint. Writing to these addresses loads data into the TxFIFO for the corresponding endpoint. Reading from these addresses unloads data from the RxFIFO for the corresponding endpoint. The address range is 20h – 5Fh and the FIFOs are located on 32-bit double-word boundaries (endpoint 0 at 20h, endpoint 1 at 24h ... Endpoint 15 at 5Ch).

Note 1: Transfers to and from FIFOs may be 8-bit, 16-bit or 32-bit as required, and any combination of access is allowed provided the data accessed is contiguous. However, all the transfers associated with one packet must be of the same width so that the data is consistently byte-, word- or double-word-aligned. The last transfer may however contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

Note 2: Depending on the size of the FIFO and the expected maximum packet size, the FIFOs support either single-packet or doublepacket buffering. However, burst writing of multiple packets is not supported as flags need to be set after each packet is written.

Note 3: Following a STALL response or a Tx strike out error on dndpoint 1 – 15, the associated FIFO is completely flushed.

The transmit and receive FIFO register for endpoint 0 - 15 (USBn_FIFO0 – USBn_FIFO15) is shown in Figure 24-175 and described in Table 24-187.

Figure 24-175. Transmit and Receive FIFO Register for Endpoint 0 - 15 (USBn_FIFO0 – USBn_FIFO15)

31	0
DATA	
R/W-0-FFFF FFFFh	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 24-187. FIFOs: Transmit and Receive FIFO Register for Endpoint 0 - 15 (USBn_FIFO0 – USBn_FIFO15) Field Descriptions

Bit	Field	Description
31-0	DATA	Writing to these addresses loads data into the Transmit FIFO for the corresponding endpoint. Reading from these addresses unloads data from the Receive FIFO for the corresponding endpoint.

24.9.7.2 Additional Control and Configuration Registers

Table 24-188 lists the additional Control and Configuration registers.

Table 24-188. Additional Control and Configuration Registers

Core Address Offset	Acronym	Control and Configuration Registers
60h	USBn_DEVCTL	Device Control Register
62h	USBn_TXFIFOSZ	Transmit Endpoint FIFO Size Register
63h	USBn_RXFIFOSZ	Receive Endpoint FIFO Size Register
64h	USBn_TXFIFOADDR	Transmit Endpoint FIFO Address Register
66h	USBn_RXFIFOADDR	Receive Endpoint FIFO Address Register
6Ch	USBn_HWVERS	Hardware Version Register

24.9.7.2.1 Device Control Register (USBn_DEVCTL)

The device control register (USBn_DEVCTL) is an 8-bit read-only register that is used to select whether the USB controller is operating in peripheral mode or in host mode, and for controlling and monitoring the USB VBus line. If the PHY is suspended no PHY clock is received and the VBus is not sampled. This register is shown in Figure 24-176 and described in Table 24-189.

Figure 24-176. Device Control Register (USBn_DEVCTL)

7	6	5	4	3	2	1	0
BDEVICE	FSDEV	LSDEV	VBUS		HOSTMODE	HOSTREQ	SESSION
R-0	R-0	R-0	R-0		R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 24-189. Device Control Register (USBn_DEVCTL) Field Descriptions

Bit	Field	Value	Description
7	BDEVICE		This read-only bit indicates whether the USB controller is operating as the 'A' device or the 'B' device.
		0	A device
		1	B device
6	FSDEV	0-1	This read-only bit is set when a full-speed or high-speed device has been detected being connected to the port (high-speed devices are distinguished from full-speed by checking for high-speed chirps when the device is reset). Only valid in Host mode.
5	LSDEV	0-1	This read-only bit is set when a low-speed device has been detected being connected to the port. Only valid in Host mode.
4-3	VBUS		These read-only bits encode the current VBus level as follows:
		0	Below session end
		1h	Above session dnd, below AValid
		2h	Above AValid, below VBusValid
		3h	Above VBusValid
2	HOSTMODE	0-1	This read-only bit is set when the USB controller is acting as a Host.
1	HOSTREQ	0-1	When set, the USB controller initiates the Host negotiation when suspend mode is entered. It is cleared when Host negotiation is completed. ('B' device only)
0	SESSION	0-1	When operating as an 'A' device, you must set or clear this bit start or end a session. When operating as a 'B' device, this bit is set/cleared by the USB controller when a session starts/ends. You must also set this bit to initiate the session request protocol. When the USB controller is in suspend mode, you may clear the bit to perform a software disconnect. A special software routine is required to perform SRP. Details will be made available in a later document version.

24.9.7.2.2 Transmit Endpoint FIFO Size Register (USBn_TXFIFOSZ)

The allocation of FIFO space to the different endpoints requires the specification for each Tx endpoint:

- The start address of the FIFO within the RAM block
- The maximum size of packet to be supported
- Whether double-buffering is required

The transmit endpoint FIFO size register (USBn_TXFIFOSZ) defines the amount of space that needs to be allocated to the FIFO in either single or double buffering context.

The transmit endpoint FIFO size register is shown in Figure 24-177 and described in Table 24-190.

Figure 24-177. Transmit Endpoint FIFO Size Register (USBn_TXFIFOSZ)

7	5	4	3	0
Reserved		DPB	SZ	
R-0		R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-190. Transmit Endpoint FIFO Size Register (USBn_TXFIFOSZ) Field Descriptions

Bit	Field	Description
7-5	Reserved	Reserved
4	DPB	Double packet buffering enable. Single packet buffering is supported. Double packet buffering is enabled
3-0	SZ	Maximum packet size to be allowed (before any splitting within the FIFO of Bulk packets prior to transmission). If m = SZ, the FIFO size is calculated as 2(m+3) for single packet buffering and 2(m+4) for dual packet buffering.

24.9.7.2.3 Receive Endpoint FIFO Size Register (USBn_RXFIFOSZ)

The allocation of FIFO space to the different endpoints requires the specification for each Rx endpoint

- The start address of the FIFO within the RAM block
- The maximum size of packet to be supported
- Whether double-buffering is required

The receive endpoint FIFO size register (USBn_RXFIFOSZ) defines the amount of space that needs to be allocated to the FIFO in either single or double buffering context.

The receive endpoint FIFO size register is shown in Figure 24-178 and described in Table 24-191.

Figure 24-178. Receive Endpoint FIFO Size Register (USBn_RXFIFOSZ)

7	6	5	4	3	0
Reserved			DPB	SZ	
R-0			R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-191. Receive Endpoint FIFO Size Register (USBn_RXFIFOSZ) Field Descriptions

Bit	Field	Description
7-5	Reserved	Reserved
4	DPB	Double packet buffering enable. Single packet buffering is supported. Double packet buffering is enabled
3-0	SZ	Maximum packet size to be allowed (before any splitting within the FIFO of Bulk packets prior to transmission). If m = SZ, the FIFO size is calculated as 2(m+3) for single packet buffering and 2(m+4) for dual packet buffering.

24.9.7.2.4 Transmit Endpoint FIFO Address Register (USBn_TXFIFOADDR)

The allocation of FIFO space to the different endpoints requires the specification for each Tx endpoint:

- The start address of the FIFO within the RAM block
- The maximum size of packet to be supported
- Whether double-buffering is required

The transmit endpoint FIFO address register (USBn_TXFIFOADDR) the coding for the start address of the FIFO. Note that the 1st 64 bytes of the FIFO RAM is reserved for the use of endpoint 0. Allocation of FIFO should exclude this reserved location.

The transmit endpoint FIFO address register is shown in Figure 24-179 and described in Table 24-192.

Figure 24-179. Transmit Endpoint FIFO Address Register (USBn_TXFIFOADDR)

15	13	12	0
Reserved		ADDR	
R-0		R/W-1FFFh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-192. Transmit Endpoint FIFO Address Register (USBn_TXFIFOADDR) Field Descriptions

Bit	Field	Description
15-13	Reserved	Reserved
12-0	ADDR	Start address of endpoint FIFO in units of 8 bytes If m = ADDR, then the start address is 8 × m

24.9.7.2.5 Receive Endpoint FIFO Address Register (USBn_RXFIFOADDR)

The allocation of FIFO space to the different endpoints requires the specification for each Rx endpoint:

- The start address of the FIFO within the RAM block
- The maximum size of packet to be supported
- Whether double-buffering is required

The receive endpoint FIFO address register (USBn_RXFIFOADDR) the coding for the start address of the FIFO. Note that the 1st 64 bytes of the FIFO RAM is reserved for the use of endpoint 0. Allocation of FIFO should exclude this reserved location.

The receive endpoint FIFO address register is shown in Figure 24-180 and described in Table 24-193.

Figure 24-180. Receive Endpoint FIFO Address Register (USBn_RXFIFOADDR)

15 13	12 0
Reserved	ADDR
R-0	R/W-1FFFh

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-193. Receive Endpoint FIFO Address Register (USBn_RXFIFOADDR) Field Descriptions

Bit	Field	Description
15-13	Reserved	Reserved
12-0	ADDR	Start address of endpoint FIFO in units of 8 bytes If m = ADDR, then the start address is 8 × m

24.9.7.2.6 Hardware Version Register (USBn_HWVERS)

The hardware version register (USBn_HWVERS) is a 16-bit read-only register that returns information about the version of the RTL from which the core hardware was generated, in particular the RTL version number (REVMAJ.REVMIN). This register is shown in Figure 24-181 and described in Table 24-194.

Figure 24-181. Hardware Version Register (USBn_HWVERS)

15	14 10	9 0
RC	REVMAJ	REVMIN
R-0	R-1Fh	R-3E7h

LEGEND: R = Read only; -*n* = value after reset

Table 24-194. Hardware Version Register (USBn_HWVERS) Field Descriptions

Bit	Field	Description
15	RC	Set to 1 if RTL is used from a Release Candidate, rather than from a full release of the core.
14-10	REVMAJ	Major version of RTL. Range is 0-31.
9-0	REVMIN	Minor version of RTL. Range is 0-999.

24.9.7.3 Target Endpoint Control Registers

Table 24-195 lists the target endpoint control registers. These registers provide target function and hub address details for each of the endpoints. Register addresses consumed by this block is 0x0080 to 0x00FF.

Table 24-195. Target Endpoint Control Registers

Core Address Offset	Acronym	Target Endpoint Control Registers
80h	USBn_TXFUNCADDRm	Transmit Function Address Register
82h	USBn_TXHUBADDRm	Transmit Hub Address Register
83h	USBn_TXHUBPORTm	Transmit Hub Port Register
84h	USBn_RXFUNCADDRm	Receive Function Address Register
86h	USBn_RXHUBADDRm	Receive Hub Address Register
87h	USBn_RXHUBPORTm	Receive Hub Port Register

24.9.7.3.1 Transmit Function Address Register (USBn_TXFUNCADDRm)

The transmit function address register (USBn_TXFUNCADDRm) is a 7-bit register (meaningful in host mode operation only) that records the address of the target function that is to be accessed through the associated endpoint (EPn). USBn_ TXFUNCADDR needs to be defined for each Tx endpoint that is used.

USBn_TXFUNCADDRm needs to be defined for endpoint 0. This register, with the USBn_TXHUBADDRm and USBn_TXHUBPORTm registers, allows the allocation of multiple devices to the USB controller endpoints.

The transmit function address register is shown in Figure 24-182 and described in Table 24-196.

Figure 24-182. Transmit Function Address Register (USBn_TXFUNCADDRm)

7	6	0
Reserved	FUNCADDR	
R-0	R/W-7Fh	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 24-196. Transmit Function Address (USBn_TXFUNCADDRm) Field Descriptions

Bit	Field	Description
7	Reserved	Reserved
6-0	FUNCADDR	Address of target function

24.9.7.3.2 Transmit Hub Address Register (USBn_TXHUBADDRm)

The transmit hub address register (USBn_TXHUBADDRm) is an 8-bit register, like USBn_TXHUBPORTm, that only needs to be written where a full- or low-speed device is connected to Tx endpoint EPn via a high-speed USB 2.0 hub that carries out the necessary transaction translation to convert between high-speed transmission and full-/low-speed transmission. In such circumstances:

- The lower 7 bits should record the address of this USB 2.0 hub
- The top bit should record whether the hub has multiple transaction translators (clear to 0 if single transaction translator; set to 1 if multiple transaction translators).

If endpoint 0 is connected to a hub, then USBn_TXHUBADDRm need to be defined for this endpoint.

The transmit hub address register is shown in Figure 24-183 and described in Table 24-197.

Figure 24-183. Transmit Hub Address Register (USBn_TXHUBADDRm)

7	6	0
MULT_TRANS	HUBADDR	
R/W-0	R/W-7Fh	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 24-197. Transmit Hub Address Register (USBn_TXHUBADDRm) Field Descriptions

Bit	Field	Description
7	MULT_TRANS	Set to 1 if hub has multiple transaction translators. Cleared to 0 if only single transaction translator is available.
6-0	HUBADDR	Address of hub

24.9.7.3.3 Transmit Hub Port Register (USBn_TXHUBPORTm)

The transmit hub port register (USBn_TXHUBPORTm) (meaningful in host mode operation only) only needs to be written where a full- or low-speed device is connected to Tx endpoint EPn via a high-speed USB 2.0 hub, which carries out the necessary transaction translation. In such circumstances, these 7-bit read/write registers need to be used to record the port of that USB 2.0 hub through which the target associated with the endpoint is accessed.

If Endpoint 0 is connected to a hub, then USBn_TXHUBPORT needs to be defined for this endpoint.

The transmit hub port register is shown in Figure 24-184 and described in Table 24-198.

Figure 24-184. Transmit Hub Port (USBn_TXHUBPORTm)

7	6	0
Reserved	HUBPORT	
R-0	R/W-7Fh	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 24-198. Transmit Hub Port Register (USBn_TXHUBPORTm) Field Descriptions

Bit	Field	Description
7	Reserved	Reserved
6-0	HUBPORT	Port number of the hub

24.9.7.3.4 Receive Function Address Register (USBn_RXFUNCADDRm)

The receive function address register (USBn_RXFUNCADDRm) is a 7-bit register (meaningful in host mode operation only) that records the address of the target function that is to be accessed through the associated endpoint (EPn). USBn_ RXFUNCADDR needs to be defined for each Rx endpoint that is used.

USBn_RXFUNCADDRm needs to be defined for endpoint 0. This register, with the USBn_RXHUBADDRm and USBn_RXHUBPORTm registers, allows the allocation of multiple devices to the USB controller endpoints.

The receive function address register is shown in Figure 24-185 and described in Table 24-199.

Figure 24-185. Receive Function Address Register (USBn_RXFUNCADDRm)

7	6	0
Reserved	FUNCADDR	
R-0	R/W-7Fh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-199. Receive Function Address Register (USBn_RXFUNCADDRm) Field Descriptions

Bit	Field	Description
7	Reserved	Reserved
6-0	FUNCADDR	Address of target function.

24.9.7.3.5 Receive Hub Address Register (USBn_RXHUBADDRm)

The receive hub address register (USBn_RXHUBADDRm) is an 8-bit register (meaningful in host mode operation only), like USBn_RXHUBPORTm, that only needs to be written where a full- or low-speed device is connected to Rx endpoint EPn via a high-speed USB 2.0 hub that carries out the necessary transaction translation to convert between high-speed transmission and full-/low-speed transmission. In such circumstances:

- The lower 7 bits should record the address of this USB 2.0 hub
- The top bit should record whether the hub has multiple transaction translators (clear to 0 if single transaction translator; set to 1 if multiple transaction translators).

If endpoint 0 is connected to a hub, then USBn_RXHUBADDRm need to be defined for this endpoint.

The receive hub address register is shown in Figure 24-186 and described in Table 24-200.

Figure 24-186. Receive Hub Address Register (USBn_RXHUBADDRm)

7	6	0
MULT_TRANS	HUBADDR	
R/W-0	R/W-7Fh	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 24-200. Receive Hub Address Register (USBn_RXHUBADDRm) Field Descriptions

Bit	Field	Description
7	MULT_TRANS	Set to 1 if hub has multiple transaction translators. Cleared to 0 if only single transaction translator is available.
6-0	HUBADDR	Address of HUB.

24.9.7.3.6 Receive Hub Port Register (USBn_RXHUBPORTm)

The receive hub port register (USBn_RXHUBPORTm) (meaningful in host mode operation only) only needs to be written where a full- or low-speed device is connected to Rx endpoint EPn via a high-speed USB 2.0 hub that carries out the necessary transaction translation. In such circumstances, these 7-bit read/write registers need to be used to record the port of that USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: If endpoint 0 is connected to a hub, then USBn_RXHUBPORT needs to be defined for this endpoint.

The receive hub port register is shown in Figure 24-187 and described in Table 24-201

Figure 24-187. Receive Hub Port Register (USBn_RXHUBPORTm)

7	6	0
Reserved	HUBPORT	
R-0	R/W-7Fh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-201. Receive Hub Port Register (USBn_RXHUBPORTm) Field Descriptions

Bit	Field	Description
7	Reserved	Reserved
6-0	HUBPORT	Port number of HUB.

24.9.7.4 Non-Indexed Endpoint Control/Status Registers

The registers available at indexed region, offset 10h–1Fh, are accessible independently of the setting of the Index register. The Non-Indexed region has dedicated block of register space for each endpoint with memory block between 100h–10Fh are reserved for endpoint 0 registers; memory block between 110h–11Fh are reserved for endpoint 1 registers; memory block between 120h–12Fh are reserved for endpoint 2 registers; and so on.

Table 24-202 and Table 24-203 display the set of endpoint registers that are accessible from their dedicated block of memory in both peripheral and host mode, respectively. Note that the definitions of these register is discussed in detail in Section 24.9.6.2.

Table 24-202. Peripheral Mode Configuration and Status Register Mapping where n = 0,1 (USB0 or USB1) and m = 1-15 (endpoint number)

Offset	Name	Description (within Indexed Region)
100h + 10h × m	USBn_TXMAXPm	See USBn_TXMAXP
102h	USBn_ PERI_CSR0	See USBn_PERI_CSR0
102h + 10h × m	USBn_PERI_TXCSRm	See USBn_PERI_TXCSR
104h + 10h × m	USBn_RXMAXPm	See USBn_RXMAXP
106h + 10h × m	USBn_ PERI_RXCSRm	See USBn_PERI_RXCSR
108h	USBn_COUNT0	See USBn_COUNT0
108h + 10h × m	USBn_ RXCOUNTm	See USBn_RXCOUNT
10Fh	USBn_CONFIGDATA	See USBn_CONFIGDATA

Table 24-203. Host Mode Configuration and Status Register Mapping where n = 0,1 (USB0 or USB1) and m = 1-15 (endpoint number)

Offset	Name	Description
100h +10h × m	USBn_TXMAXPm	See USBn_TXMAXP
102h	USBn_HOST_CSR0	See USBn_HOST_CSR0
102h + 10h × m	USBn_HOST_TXCSRm	See USBn_HOST_TXCSR
104h + 10h × m	USBn_RXMAXPm	See USBn_RXMAXP
106h + 10h × m	USBn_HOST _RXCSRm	See USBn_HOST_RXCSR
108h	USBn_COUNT0	See USBn_COUNT0
108h + 10h × m	USBn_RXCOUNTm	See USBn_RXCOUNT
10Ah	USBn_HOST_TYPE0	See USBn_HOST_TYPE0
10Ah + 10h × m	USBn_HOST_TXTYPEm	See USBn_HOST_TXTYPE
10Bh	USBn_HOST_NAKLIMIT0	See USBn_HOST_NAKLIMIT0
10Bh + 10h × m	USBn_HOST_TXINTERVALm	See USBn_HOST_TXINTERVAL
10Ch	USBn_HOST_TYPE0	See USBn_HOST_TYPE0
10Ch + 10h × m	USBn_HOST_RXTYPEm	See USBn_HOST_RXTYPE
10Dh + 10h × m	USBn_HOST_RXINTERVALm	See USBn_HOST_RXINTERVAL
10Fh	USBn_CONFIGDATA	See USBn_CONFIGDATA

USB Related Clock/PHY Control Registers

This section contains information on USB PHY registers, which are also used for USB Interface Clocking and USB Controller Clocking. For details on USB Interface Clocking and USB Controller Clocking, see the appropriate chapter.

24.9.7.1 CM_DEFAULT_L3_SLOW_CLKSTCTRL Register

The CM_DEFAULT_L3_SLOW_CLKSTCTRL register enables the domain power state transition. It controls the software supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also holds one status bit per clock input of the domain.

This register is shown in Figure 24-188 and described in Table 24-204.

Figure 24-188. CM_DEFAULT_L3_SLOW_CLKSTCTRL Register

31 9	8	7 2	1 0
Reserved	CLKACTIVITY_USB_GCLK	Reserved	CLKTRCTRL
R-0	R-0	R-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-204. CM_DEFAULT_L3_SLOW_CLKSTCTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	CLKACTIVITY_USB_GCLK		This field indicates the state of the L3_SLOW_GCLK clock in the domain.
		0	Corresponding clock is gated
		1	Corresponding clock is active
7-2	Reserved	0	Reserved
1-0	CLKTRCTRL		Controls the clock state transition of the L3_SLOW clock domain in DEFAULT power domain.
		0	Reserved
		1h	SW_SLEEP: Start a software forced sleep transition on the domain.
		2h	SW_WKUP: Start a software forced wake-up transition on the domain.
		3h	Reserved

24.9.7.2 USB_CTRL Register

The USB_CTRL register controls various features of the USB subsystem. This register is shown in Figure 24-189 and described in Table 24-205.

Figure 24-189. USB_CTRL Register

31	16
Reserved	
R-0	

15	9	8	7	2	1	0
Reserved		PHYCLKSRC	Reserved		PHYSLEEP1	PHYSLEEP0
R-0		R/W-0	R-0		R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-205. USB_CTRL Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	PHYCLKSRC		USB PHY reference clock source
		0	Use PLL reference clock
		1	Use USB oscillator
7-2	Reserved	0	Reserved
1	PHYSLEEP1		USB PHY1 sleep mode control. Places Phy1 in Sleep mode per USB 2.0 LPM addendum.
		0	Sleep mode
		1	Normal operating mode
0	PHYSLEEP0		USB PHY0 sleep mode control. Places Phy0 in Sleep mode per USB 2.0 LPM addendum.
		0	Sleep mode
		1	Normal operating mode

24.9.7.3 USBPHY_CTRL0 Register

The USBPHY_CTRL0 register controls various features of the USB0 Phy. This register is shown in Figure 24-190 and described in Table 24-206.

Figure 24-190. USBPHY_CTRL0 Register

31	30 28	27 23	22 20	19 16
Rsvd	COMPDISTUNE	Reserved	SQRXTUNE	TXFSLSTUNE
R-0	R/W-4h	R-0	R/W-3h	R/W-3h

15	14 13	12	11 8	7 6	5 4	3 0
Rsvd	TXPREEMTUNE	TXRISETUNE	TXVREFTUNE	Reserved	TXHSXVTUNE	Reserved
R-0	R/W-0	R/W-0	R/W-8h	R-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-206. USBPHY_CTRL0 Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	COMPDISTUNE	1-0	Disconnect Threshold Adjust.
27-23	Reserved	0	Reserved
22-20	SQRXTUNE	1-0	Squelch Threshold Adjust.
19-16	TXFSLSTUNE	1-0	FS/LS Source Impedance Adjust.
15	Reserved	0	Reserved
14-13	TXPREEMTUNE	1-0	HS Transmit Pre-Emphasis Enable.
12	TXRISETUNE	1-0	HS Transmit Rise/Fall Time Adjust.
11-8	TXVREFTUNE	1-0	HS DC Voltage Level Adjust.
7-6	Reserved	0	Reserved
5-4	TXHSXVTUNE	1-0	Transmit High-Speed Crossover Adjust.
3-0	Reserved	0	Reserved

24.9.7.4 USBPHY_CTRL1 Register

The USBPHY_CTRL1 register controls various features of the USB1 Phy. This register is shown in Figure 24-191 and described in Table 24-207.

Figure 24-191. USBPHY_CTRL1 Register

31	30 28	27 23	22 20	19 16
Rsvd	COMPDISTUNE	Reserved	SQRXTUNE	TXFSLSTUNE
R-0	R/W-4h	R-0	R/W-3h	R/W-3h

15	14 13	12	11 8	7 6	5 4	3 0
Rsvd	TXPREEMTUNE	TXRISETUNE	TXVREFTUNE	Reserved	TXHSXVTUNE	Reserved
R-0	R/W-0	R/W-0	R/W-8h	R-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-207. USBPHY_CTRL1 Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-28	COMPDISTUNE	1-0	Disconnect Threshold Adjust.
27-23	Reserved	0	Reserved
22-20	SQRXTUNE	1-0	Squelch Threshold Adjust.
19-16	TXFSLSTUNE	1-0	FS/LS Source Impedance Adjust.
15	Reserved	0	Reserved
14-13	TXPREEMTUNE	1-0	HS Transmit Pre-Emphasis Enable.
12	TXRISETUNE	1-0	HS Transmit Rise/Fall Time Adjust.
11-8	TXVREFTUNE	1-0	HS DC Voltage Level Adjust.
7-6	Reserved	0	Reserved
5-4	TXHSXVTUNE	1-0	Transmit High-Speed Crossover Adjust.
3-0	Reserved	0	Reserved

ROM Code Memory and Peripheral Booting

This chapter describes the booting functionality of the ROM Code. The booting functionality covers the following features:

- *Memory Booting* – booting the device by starting code stored on permanent memories like flash-memory or memory cards. This process is usually performed after either device cold or warm reset.
- *Peripheral Booting* – booting the device by downloading the executable code over a communication interface like UART or Ethernet. This process is intended for flashing a device.

This chapter is intended for all who design systems based on this device, develop boot loader images and flashing tools.

Topic **Page**

25.1 Introduction

25.1.1 Acronyms

Table 25-1. Acronyms and Abbreviations

Acronym/Abbreviation	Definition
ASIC	Application Specific Integrated Circuit
B	Byte (8 bits)
CS	Chip Select
CH	Configuration Header
DPLL	Digital PLL
EMIF	External Memory InterFace (SDRAM Controller)
FIQ	Fast Interrupt Request
FS USB	Full Speed USB
GPMC	General Purpose Memory Controller
HLOS	High Level Operating System
HS USB	High Speed USB
HW	Hardware
HWA	Hardware Accelerator
I2C	Inter-Integrated Circuit
IRQ	Interrupt
ISR	Interrupt Service Routine
JTAG	Joint Test Action Group. This term is used to refer to the Debugging and Testing interface.
Kbps	Kilobits per second
KB	Kilobyte, 1024 B
MB	Megabyte, 1024 KB
MPU	Micro Processor Unit
OCM	On-Chip Memory
OneNAND™	A type of Flash memory
PLL	Phase Locked Loop
POR	Power On Reset
PMU	Power Management Unit
PRCM	Power, Reset and Clock Management module
RAM	Random Access Memory. Refers to the internal static RAM.
ROM	Read Only Memory
RSA	Rivest, Shamir, Adleman encryption algorithm
SAR	Save & Restore
SD card	Secure Digital card
SDRAM	Synchronous Dynamic Random Access Memory
SWI	Software Interrupt
TOC	Table Of Contents – a structure within an executable image
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
XIP	Execute In Place
WDT	Watchdog timer

25.1.2 Naming Conventions

The following is a brief explanation of some terms used in this document:

Bootstrap— Initial software that is launched by the ROM Code during the Memory Booting phase.

Downloaded SW— Initial software that is downloaded into internal RAM by the ROM Code during Peripheral Booting phase.

E-Fuse— A one-time programmable memory location usually set at the factory.

Emulator Device— A type of device where some of the security rules are relaxed. Intended for development only.

Flash Loader— Downloaded software launched by the ROM Code in Pre-Flashing and which programs an image into external memories.

Initial SW— Software that is executed by any of the ROM Code mechanisms (Memory Booting or Peripheral Booting). Initial SW is a generic term for Bootstrap and Downloaded SW.

Manager— A modular software component with finite and focused functionality.

Memory Booting— ROM Code mechanism that consists of executing an Initial SW from external memory.

Peripheral Booting— ROM Code mechanism that consists of polling selected interfaces, downloading and executing an Initial SW (in this case called Downloaded SW) in internal RAM.

Pre-Flashing— Specific case of Peripheral Booting, where the ROM Code mechanism is used to program external Flash memories.

25.2 Overview

25.2.1 Architecture

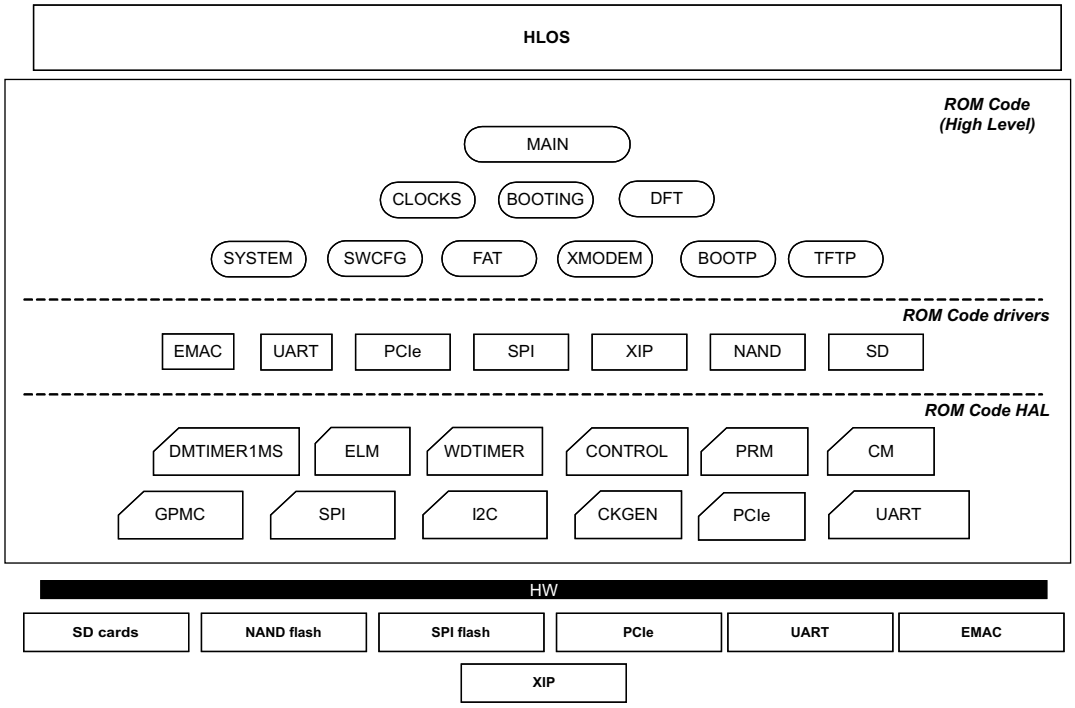
The architecture of the ROM Code is shown in Figure 25-1. It is split into three main layers with a top-down approach: high-level, drivers, and hardware abstraction layer (HAL). One layer communicates with a lower level layer through a unified interface.

The high-level layer is in charge of the main tasks of the ROM Code: watchdog and clocks configuration and main booting routine.

The drivers layer implements the logical and communication protocols for any booting device in accordance with the interface specification.

Finally the HAL implements the lowest level code for interacting with the hardware infrastructure IPs. End booting devices are attached to device IO pads.

Figure 25-1. ROM Code Architecture



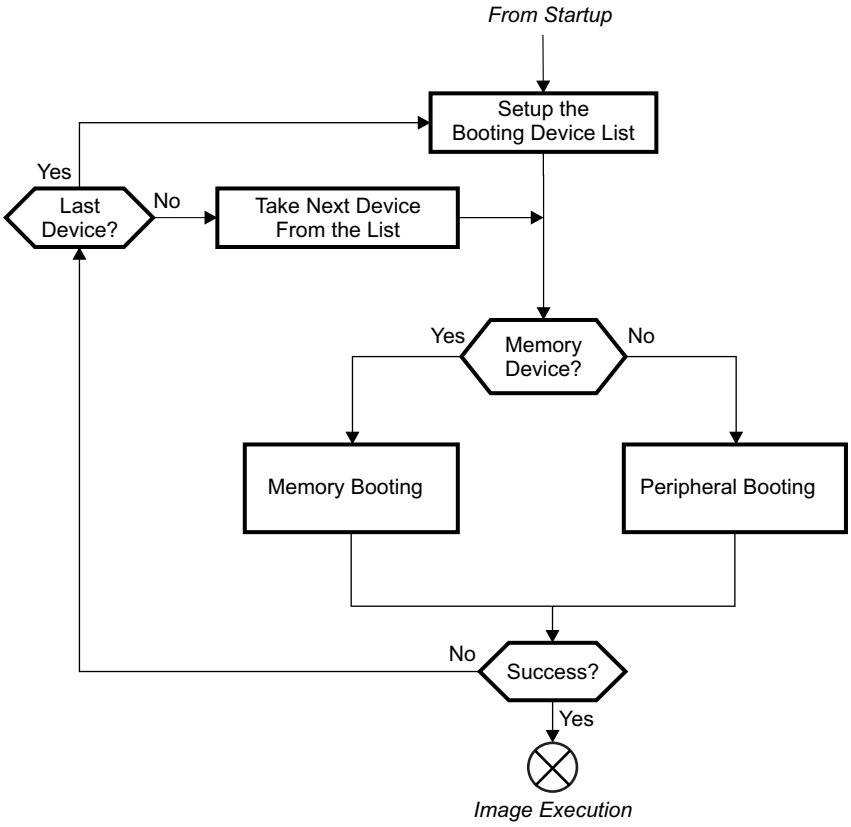
25.2.2 Functionality

Figure 25-2 illustrates the high-level flow for the ROM Code booting procedure. The ROM Code performs platform configuration and initialization as part of the start-up procedure.

The booting device list is created based on the MBOOT.A booting device can be a memory booting device (soldered flash memory or temporarily booting device like memory card) or a peripheral interface connected to a host.

The main loop of the booting procedure goes through the booting device list and tries to search for an image from the currently selected booting device. This loop is exited if a valid booting image is found and successfully executed or upon watchdog expiration.

Figure 25-2. ROM Code Boot Procedure



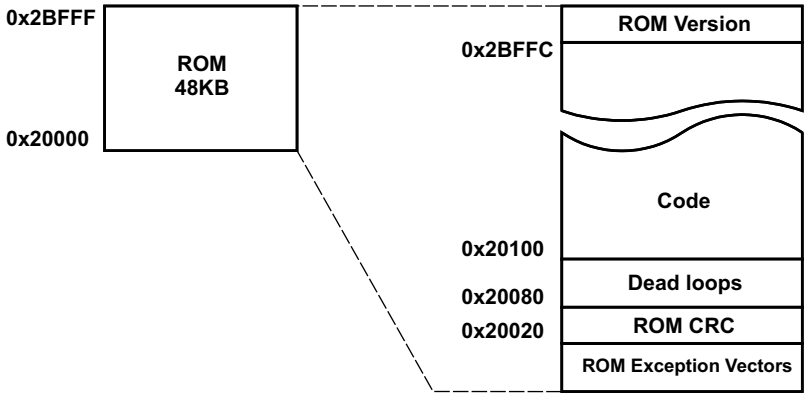
25.3 Memory Map

25.3.1 ROM Memory Map

The on-chip ROM memory map is shown in Figure 25-3. The ROM Code mapping consists of the following:

- ROM Exception vectors
- CRC
- Dead loops collection
- Code and const data sections
- API Table
- ROM Version

Figure 25-3. ROM Memory Map



25.3.1.1 ROM Exception Vectors

Table 25-2 lists the ROM exception vectors. The reset exception is redirected to the ROM Code startup. Other exceptions are redirected to their RAM handlers by loading appropriate addresses into the PC register.

Table 25-2. ROM Exception Vectors

Address	Exception	Content
20000h	Reset	Branch to the ROM Code startup
20004h	Undefined	PC = 4030 D004h
20008h	SWI	PC = 4030 D008h
2000Ch	Pre-fetch abort	PC = 4030 D00Ch
20010h	Data abort	PC = 4030 D010h
20014h	Unused	PC = 4030 D014h
20018h	IRQ	PC = 4030 D018h
2001Ch	FIQ	PC = 4030 D01Ch

25.3.1.2 ROM Code CRC

The ROM Code CRC is calculated as 32 bit CRC code (CRC-32-IEEE 802.3) for the address range 20000h–2BFFFh. The four bytes CRC code is stored at location 20020h.

25.3.1.3 Dead Loops

Built-in dead loops are used for different purposes as shown in Table 25-3. All dead loops are branch instructions coded in ARM mode. The fixed location of these dead loops facilitates debugging and testing. The first seven dead loops are default exception handlers linked with RAM exception vectors. The dead loops might be called directly from the user code. However there exists a special function which can be called from ROM Code in order to execute a dead loop. The function is an assembly code in ARM mode which takes the dead loop address from R0 register. The main purpose of the function is to issue a global software reset before going to a dead loop. The function is located at address 200C0h. In addition, the function clears global cold reset status upon issuing the global SW reset.

Table 25-3. Dead Loops

Address	Purpose
20080h	Undefined exception default handler
20084h	SWI exception default handler
20088h	Pre-fetch abort exception default handler
2008Ch	Data abort exception default handler
20090h	Unused exception default handler
20094h	IRQ exception default handler
20098h	FIQ exception default handler
2009Ch	Validation tests PASS
200A0h	Validation tests FAIL
200A4h	Reserved
200A8h	Image not executed or returned.
200ACh	Reserved
200B0h	Reserved
200B4h	Reserved
200B8h	Reserved
200BCh	Reserved

25.3.1.4 Code

This space is used to hold code and constant data

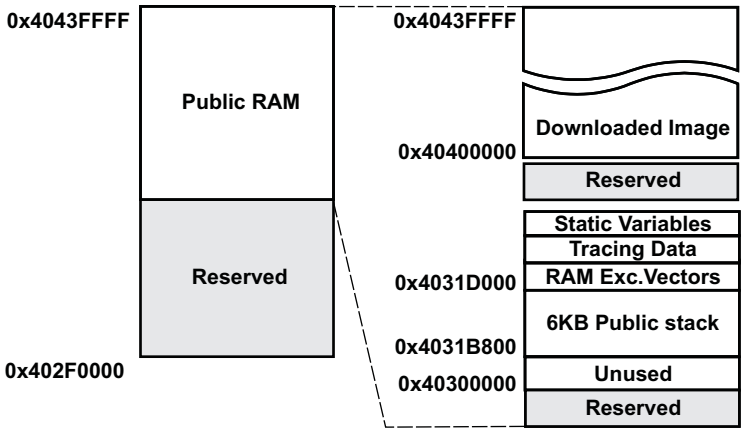
25.3.1.5 ROM Code Version

The ROM Code version consists of two decimal numbers: major and minor. It can be used to identify the ROM Code release version burned in a given IC (for example, useful to identify an IC version). The ROM Code version is a 32-bit hexadecimal value located at address 2BFFCh.

25.3.2 RAM Memory Map

The ROM Code makes use of the on-chip RAM module connected to the L3 interconnect (referred to as L3 RAM in this document). Its usage is shown in Figure 25-4. The RAM memory-map ranges from address 4030 0000h to 4043 FFFFh.

Figure 25-4. RAM Memory Map



25.3.2.1 Downloaded Image

This area is used by the ROM Code to store the downloaded boot image. It can be up to 255 KB.

25.3.2.2 Public Stack

Space reserved for stack.

25.3.2.3 RAM Exception Vectors

The RAM exception vectors enable a simple means for redirecting exceptions to custom handlers. Table 25-4 shows content of the RAM space reserved for RAM vectors. The first seven addresses are ARM instructions which load the value located in the subsequent seven addresses into the PC register. Theses instructions are executed when an exception occurs since they are called from the ROM exception vectors. Undefined, SWI, Unused and FIQ exceptions are redirected to a hardcoded dead loop. Pre-fetch abort, data abort, and IRQ exception are redirected to pre-defined ROM handlers. User code can redirect any exception to a custom handler either by writing its address to the appropriate location from 4031 D024h to 4031 D03Ch or by overriding the branch (load into PC) instruction between addresses from 4031 D004h to 4031 D01Ch.

Table 25-4. RAM Exception Vectors

Address	Exception	Content
4031 D000h	Reserved	Reserved
4031 D004h	Undefined	PC = [4031 D024h]
4031 D008h	SWI	PC = [4031 D028h]
4031 D00Ch	Pre-fetch abort	PC = [4031 D02Ch]
4031 D010h	Data abort	PC = [4031 D030h]
4031 D014h	Unused	PC = [4031 D034h]
4031 D018h	IRQ	PC = [4031 D038h]
4031 D01Ch	FIQ	PC = [4031 D03Ch]
4031 D020h	Reserved	20090h

Table 25-4. RAM Exception Vectors (continued)

Address	Exception	Content
4031 D024h	Undefined	20080h
4031 D028h	SWI	20084h
4031 D02Ch	Pre-fetch abort	Address of default pre-fetch abort handler (*)
4031 D030h	Data abort	Address of default data abort handler (*)
4031 D034h	Unused	20090h
4031 D038h	IRQ	Address of default IRQ handler
4031 D03Ch	FIQ	20098h

(*) the default handlers for pre-fetch and data abort are performing reads from CP15 debug registers to retrieve the reason of the abort:

- In case of pre-fetch abort: the IFAR register is read from CP15 and stored into R0. The IFSR register is read and stored into the R1 register. Then the ROM Code jumps to the pre-fetch abort dead loop (20088h).
- In case of data abort: the DFAR register is read from CP15 and stored into R0. The DFSR register is read and stored into the R1 register. Then the ROM Code jumps to the data abort dead loop (2008Ch).

25.3.2.4 Tracing Data

This section contains trace vectors reflecting the execution path of the boot. Section 25.12 describes the usage of the different trace vectors and lists all the possible trace codes.

Table 25-5. Tracing Data

Address	Size [bytes]	Description
4031 D040h	4	Current tracing vector, word 1
4031 D044h	4	Current tracing vector, word 2
4031 D048h	4	Current tracing vector, word 3
4031 D04Ch	4	Current copy of the PRM_RSTST register (reset reasons)
4031 D050h	4	Cold reset run tracing vector, word 1
4031 D054h	4	Cold reset run tracing vector, word 2
4031 D058h	4	Cold reset run tracing vector, word 3
4031 D05Ch	4	Reserved
4031 D060h	4	Reserved
4031 D064h	4	Reserved

25.3.2.5 Static Variables

This area contains the ROM Code static variables used during boot time (and possibly during run-time, if calling the ROM API functions).

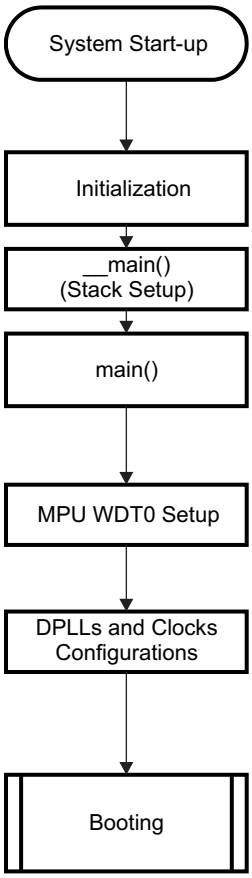
25.4 Startup and Configuration

25.4.1 ROM Code Start-up

The ROM Code is physically located at the address 20000h.

The CPU performs the initialization and stack setup (compiler auto-generated C-initialization or “scatter loading”) as shown in Figure 25-5. Then it configures the watchdog timer (set to 3 minutes) and performs system clocks configuration. Finally, it jumps to the booting routine.

Figure 25-5. ROM Code Start-up Sequence



25.4.2 CPU State at Startup

The CPU L1 instruction cache and branch prediction mechanisms are not activated as part of the boot process. The vector base address is configured to the reset vector of the ROM Code (20000h). The MMU is left switched off during the boot (hence, L1 data cache off).

No specific configuration is performed for the slave CPU which keeps its default configuration after reset (L1 instruction and data caches off, branch prediction off, MMU off, no remap of vectors base address).

25.4.3 Clocking Configuration

The supported system clock frequencies are DEV_CLKIN = 27 MHz.

The ROM Code configures the clocks and DPLLs that are necessary for ROM Code execution:

- MAIN PLL locked to provide 220 MHz clocks for ARM and peripheral blocks.
- DDR DPLL locked to provide 400 MHz for L3, UART clocks.

Table 25-6 summarizes the ROM Code default settings for clocks. This default configuration enables all the ROM Code functionalities with minimized needs on power during boot.

CAUTION

The SYSCLK10 divisor that the ROM Code configures is different between PG1.x and PG2.x devices. If SYSCLK10 is used, the original default POR values described in this document should not be depended on, but the PLLs and divisors must be explicitly configured. This is applicable for both PG1.x and PG2.x devices.

The DPLLs and PRCM clock dividers are configured with the ROM Code default values after cold or warm reset in order to give the same working conditions to the ROM Code sequence.

Table 25-6. ROM Code Default Clock Settings

Clock	Frequency (MHz)	Source
SYSCLK2	1000	MAIN PLL
SYSCLK5	250	MAIN PLL
SYSCLK6	125	MAIN PLL
SYSCLK10	48	DDR PLL

25.5 Booting

25.5.1 Overview

Figure 25-6 shows the booting procedure. First, a booting device list is created. The list consists of all devices which will be searched for a booting image. The list is filled in based on the SYSBOOT.

Once the booting device list is set up, the booting routine examines the devices enumerated in the list sequentially and either executes the memory booting or peripheral booting procedure depending on the booting device type. The memory booting procedure is executed when the booting device type is one of NOR, NAND, or SPI-EEPROM. The peripheral booting is executed when the booting device type is Ethernet, PCIe or UART.

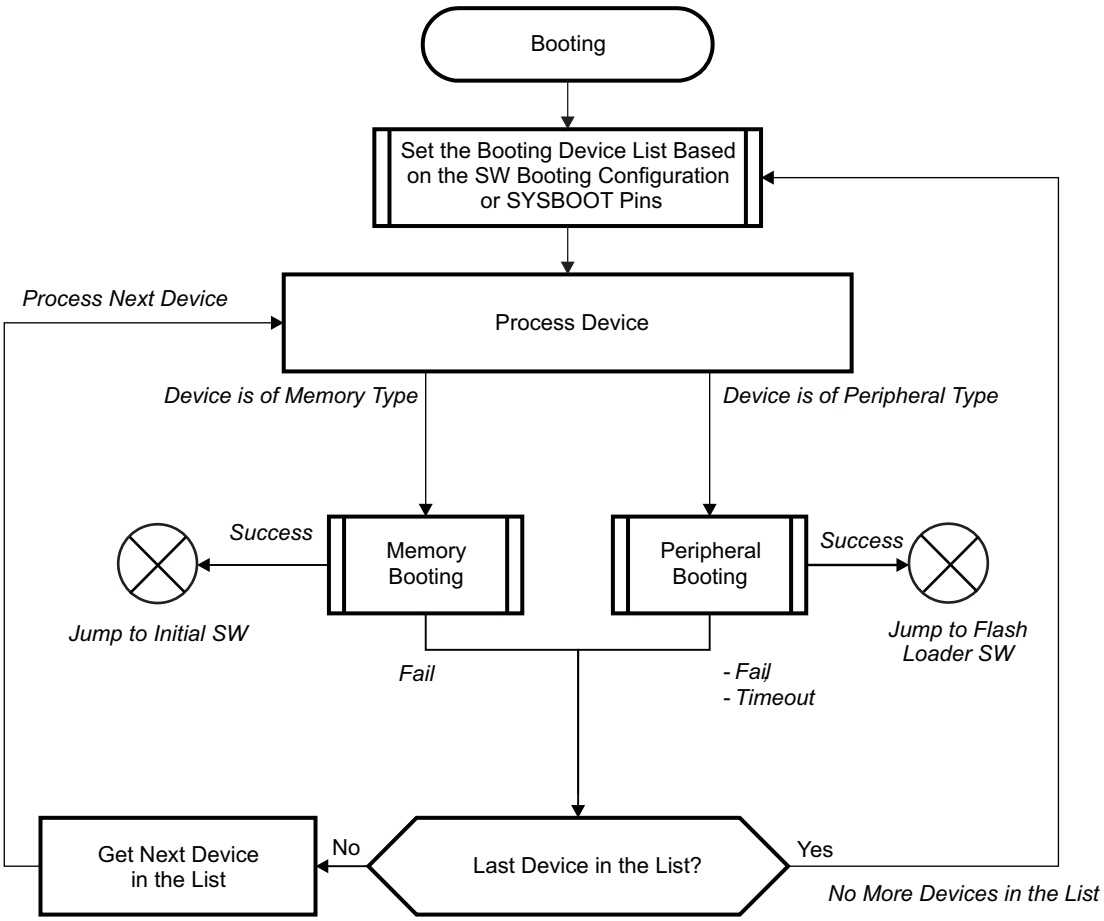
The memory booting procedure reads data from a memory type device. If a valid booting image is found and successfully read from the external memory:

- the Initial SW is simply started.

The peripheral booting procedure downloads data from a host (commonly a PC) to the device by means of Ethernet, PCIe or UART links. The ROM Code uses a host-slave logical protocol for synchronization. Upon successful PCIe enumeration (or UART or Ethernet connection) the host sends the image binary contents. The peripheral booting procedure is described in detail in Section 25.8.

If the memory or peripheral booting fails for all devices enumerated in the device list, then the ROM Code gets back to the first device in the list. The device list is then processed again in a loop. This loop shall be further interrupted by the MPU.

Figure 25-6. . ROM Code Booting Procedure



25.5.2 Device List

The ROM Code creates the device list based on information gathered from the SYSBOOT configuration pins sensed in the control module. The pins are used to index the device table from which the list of devices is extracted.

25.5.2.1 MBOOT Configuration Pins

Table 25-7. MBOOT Configuration Pins

Boot Modes				MBOOT[4:0]
1st	2nd	3rd	4th	
reserved	reserved	reserved	reserved	0
UART	XIP w/ WAIT	SD card	SPI	1
UART	SPI	NAND	NANDI2C	10
UART	SPI	XIP	SD card	11
Ethernet(GMII)	SPI	NAND	NANDI2C	100
reserved	reserved	reserved	reserved	101
reserved	reserved	reserved	reserved	110
Ethernet(GMII)	SD card	SPI	XIP	111
PCIE_32	reserved	reserved	reserved	1000
PCIE_64	reserved	reserved	reserved	1001
reserved	reserved	reserved	reserved	1010
reserved	reserved	reserved	reserved	1011
reserved	reserved	reserved	reserved	1100
reserved	reserved	reserved	reserved	1101
reserved	reserved	reserved	reserved	1110
Fast External Boot	UART	Ethernet(GMII)	PCIE_64	1111
XIP	UART	Ethernet(GMII)	SD card	10000
XIP w/WAIT	UART	Ethernet(GMII)	SD card	10001
NAND	NANDI2C	SPI	UART	10010
NAND	NANDI2C	SD card	UART	10011
NAND	NAND12C	SPI	Ethernet (GMII)	10100
NANDI2C	SD card	Ethernet(GMII)	UART	10101
SPI	SD card	UART	Ethernet(GMII)	10110
SD card	SPI	UART	Ethernet(GMII)	10111
SPI	SD card	PCIE_32	reserved	11000
SPI	SD card	PCIe_64	reserved	11001
reserved	reserved	reserved	reserved	11010
reserved	reserved	reserved	reserved	11011
reserved	reserved	reserved	reserved	11100
reserved	reserved	reserved	reserved	11101
reserved	reserved	reserved	reserved	11110
Fast External Boot	Ethernet(GMII)	UART	PCIE_32	11111

The ROM Code uses the row pointed by the MBOOT configuration value .The device list is filled in with the 1st to 4th devices.

Table 25-7 is the decoding table for MBOOT configuration pins. The following shortcuts are used in the table:

NAND / NANDI2C	NAND flash memory / read flash geometry from I2C EEPROM
XIP / XIP w/ WAIT	NOR or other XIP device with or without wait monitoring
UART	UART interface (UART port 0)
Ethernet	Ethernet GMII0 interface (EMAC port 0)
SPI	SPI EEPROM (SPI 0, CS0)
PCIE_32 / PCIE_64	PCI Express interface with 32 bit / 64 bit addressing

Fast XIP boot mechanism (as described in Section 25.6) is provided where minimal execution is performed from ROM Code for configuring the GPMC interface and then directly jump to the code contained in the connected NOR flash device.

The NOR device must fulfill the following requirements for EMU external boot mode and fast XIP mode:

- Addr/Data muxed device or a non-muxed device connected in MUX0 configuration
- Bus width
- CS0 chip select
- Device wait signal connected to the WAIT0 GPMC signal (if used)
- Wait monitoring enable/disable

25.6 Fast Internal Booting

25.6.1 Overview

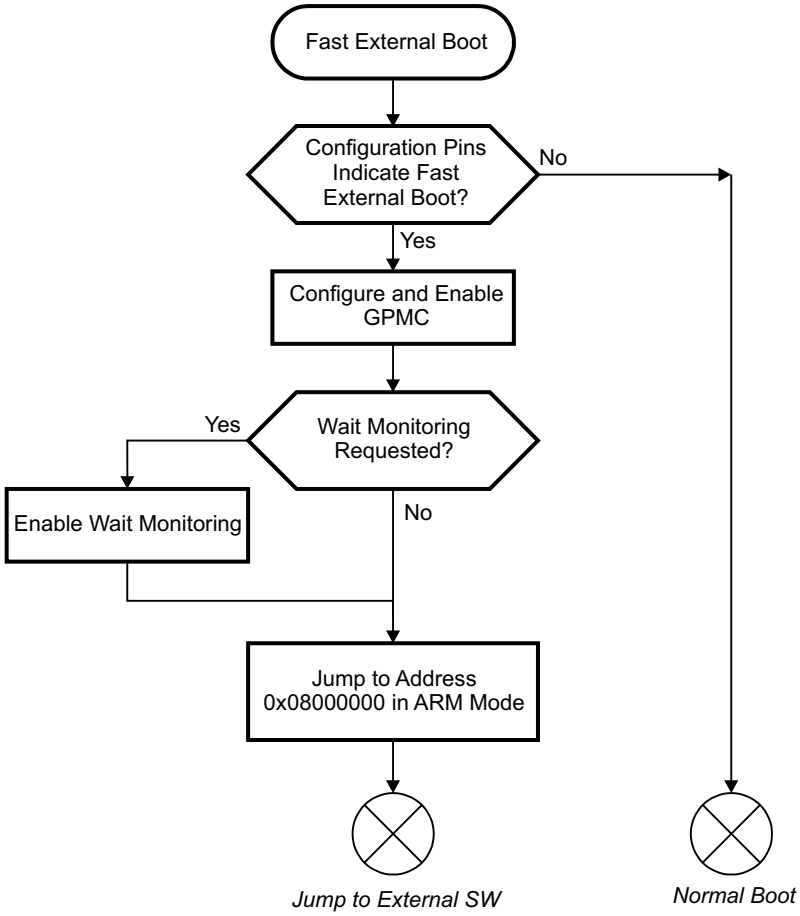
The fast external boot feature:

- Consists of a blind jump in ARM mode to a code located in an external XIP device connected to CS0
- The jump is performed with minimum on-chip ROM Code execution, without configuring any PLL
- Allows the customer to create its own booting code
- Is set up by means of the configuration pins, see Table 25-7

25.6.2 External Booting

Figure 25-7 shows the Fast External Boot procedure. The code does not make use of RAM and is designed for fast execution.

Figure 25-7. Fast External Boot

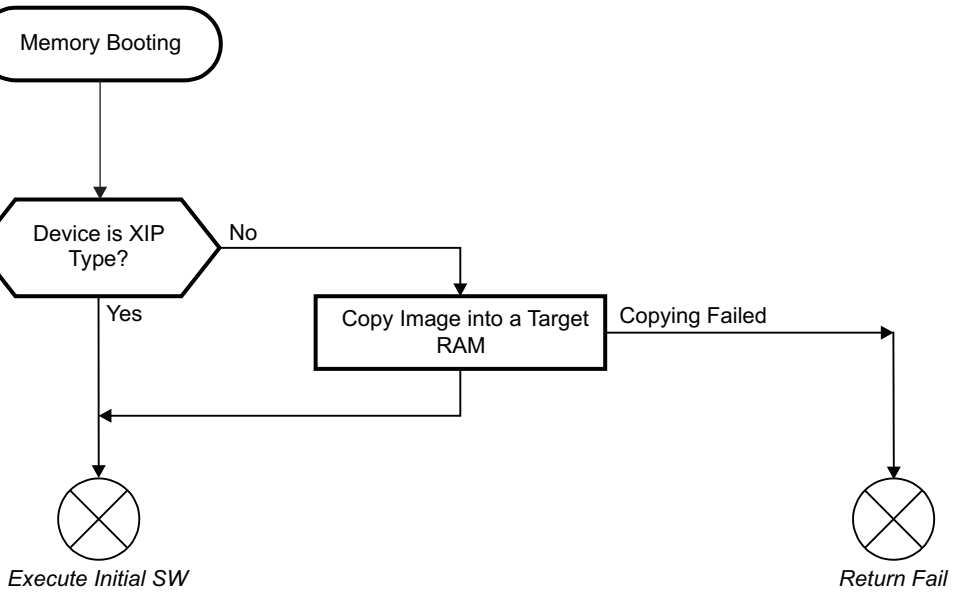


25.7 Memory Booting

25.7.1 Overview

The Memory Booting procedure takes care of starting an external code located in memory device types. These devices are also called permanent booting devices since they are used for booting permanently.

Figure 25-8. Memory Booting



The permanent booting devices supported are:

- SD cards
- NOR flash
- NAND flash
- SPI EEPROMs

There are two groups of permanent booting devices distinguished by the need of code shadowing. The code shadowing means copying a code from a non-directly addressable device into a location (typically a RAM area) from where the code can be executed. Devices that are directly addressable are called eXecute In Place (XIP) devices.

The memory booting flowchart is shown in Figure 25-8. The second step is about performing the shadowing of the image, that is copying the image from external mass storage (non-XIP) into internal RAM. Failure in image copy results in memory booting returning to the main booting procedure that will select the next device for booting. The next sections detail procedures for device initialization and detection in addition to the description of the sector read routine for each supported device type. A sector is a logical unit of 512 bytes. A booting image is considered to be present when the first 4 bytes word of the sector is not equal to 0000 0000h or FFFF FFFFh.

During the first read sector call, sectors are copied to a temporary RAM buffer. Once the image is found and destination address is known, the content of the temporary buffer is moved to the target RAM location so it is needed to re-read the first image sector.The header is discarded, therefore only executable code is located in RAM with the first executable instruction located at the destination address.

The image execution is detailed in Section 25.10. For more information about image formats and contents refer to Section 25.9.

SD cards and NAND devices can hold up to four copies of the booting image. Therefore, the ROM Code searches for one valid image out of the four if present by walking over the first four blocks of the mass storage space. Other XIP devices (NOR) use only one copy of the booting image.

25.7.2 XIP Memory

The ROM Code can boot directly from XIP devices. A typical XIP device is a NOR flash memory. Support for XIP devices is performed under the following assumptions:

- Uses GPMC as the communication interface
- Can connect up to 1 Gbit (128 Mbytes) memories
- Uses both ×8 and ×16 data bus width
- Follows asynchronous protocol
- Supports address / data multiplexed mode and non-multiplexed mode
- GPMC clock is 55 MHz
- Device is connected to CS0 mapped to address 800 0000h
- Wait pin signal WAIT0 is monitored depending on the MBOOT configuration pins (XIP/XIPWAIT).

Depending on the MBOOT option the GPMC is configured to use the WAIT signal connected on the WAIT0 pin or not. Wait pin polarity is set to stall accessing memory when the WAIT0 pin is low. The wait monitoring is intended to be used with memories which require long time for initialization after reset or need to pause while reading data. The boot procedure from XIP device can be described as such:

1. Configure GPMC for XIP device access
2. Set the image location to 800 0000h
3. Verify if bootable image is present at the image location.
4. If the image has been found, start it.
5. If the image has not been found, return from XIP booting to the main booting loop.

25.7.2.1 XIP Initialization and Detection

GPMC Initialization

Figure 25-9 and Table 25-8 describes the GPMC timing settings set for XIP boot and other address-data accessible devices (for example, OneNAND).

Figure 25-9. GPMC XIP Timings

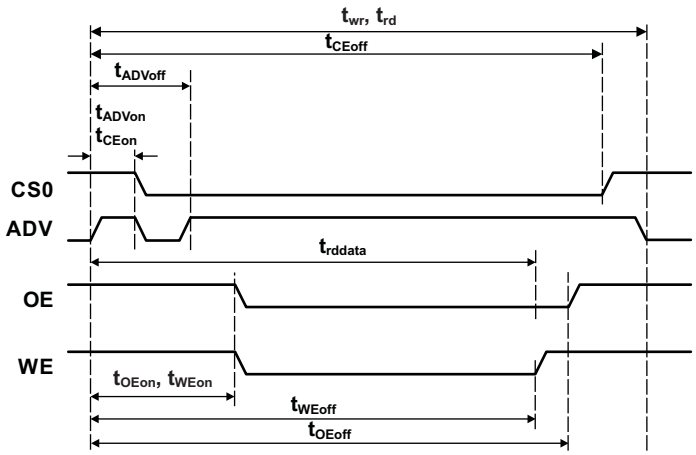


Table 25-8. XIP Timings Parameters

Parameter	Description	Value [clock cycles]
twr	write cycle period	17
trd	read cycle period	17
tCEon	CE low time	1
tCEoff	CE high time	16
tADVon	ADV low time	1
tADVoff	ADV high time	2
tOEon	OE low time	3
tWEon	WE low time	3
trddata	data latch time	15
tOEoff	OE high time	16
tWEoff	WE high time	15

The one clock cycle is 18.182 ns which corresponds to 55 MHz frequency.

There is no specific identification routine executed prior to booting from an XIP device.

The list of pins that are configured by the ROM in the case of NOR boot mode are listed in Table 25-9. Note that all the pins might not be driven at boot time. The decision as to which pins need to be driven is done based on the type of NOR flash selected.

The pins that are not listed in Table 25-9 are not configured by the ROM code, and are left at power on defaults. Specifically, external logic is needed to isolate the upper address lines (A12-A27) of the NOR flash from the device pins and drive them low during boot. Once the Initial Software starts running, it can configure the pinmux setting appropriately for the lines and remove the isolation to allow the GPMC to drive all the address lines.

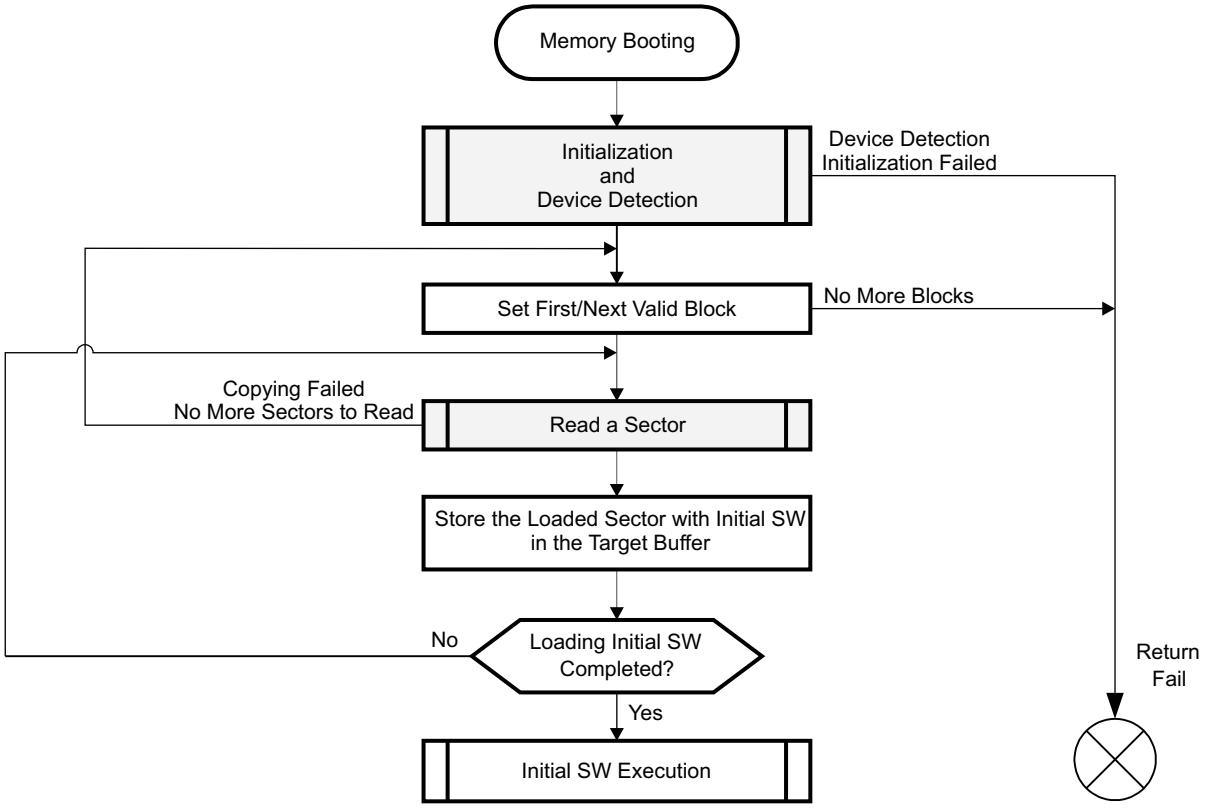
Table 25-9. Pins Used for NOR Boot

Signal Name	Pin Used in XIP Mode
cs0	$\overline{\text{GPMC\_CS[0]}}$
advn_ale	GPMC_$\overline{\text{ADV}}$_ALE
oen_ren	$\overline{\text{GPMC\_OE\_RE}}$
be0n_cle	GPMC_$\overline{\text{BE0}}$_CLE
Wen	$\overline{\text{GPMC\_WE}}$
Wait	GPMC_WAIT
Clk	GPMC_CLK
ad0 - ad15	GPMC_D[15:0]
a0	GPMC_A[0]
a1	GPMC_A[1]
a2	GPMC_A[2]
a3	GPMC_A[3]
a4	GPMC_A[4]
a5	GPMC_A[5]
a6	GPMC_A[6]
a7	GPMC_A[7]
a8	GPMC_A[8]
a9	GPMC_A[9]
a10	GPMC_A[10]
a11	GPMC_A[11]

Image Shadowing for non-XIP memories

- The image shadowing uses the approach shown in Figure 25-10.

Figure 25-10. Image Shadowing



25.7.3 NAND

NOTE: For PG2.x devices, CS0BW is not needed for NAND boot. The boot ROM detects the size of the NAND device and configures itself appropriately.

The NAND flash memory is not XIP and requires shadowing before the code can be executed. The features include:

- GPMC as the communication interface
- Device from 512Mbit (64 MByte)
- ×8 and ×16 bus width
- Support for large page size (2048 bytes + 64 spare bytes) or very large page size 4096 bytes + 128 / 218 spare bytes)
- CE don't care devices only
- Single Level Cell (SLC) and Multiple Level Cell (MLC) devices
- Device Identification based on ONFI or ROM table
- ECC correction : 8 bits/sector for most devices (16b/sector for devices with large spare area)
- GPMC timings adjusted for NAND access
- 55 MHz GPMC clock
- Device connected to CS0
- Wait pin signal WAITPIN0 connected to NAND BUSY output
- Four physical blocks are searched for an image. The block size depends on device.

The initialization routine for NAND devices consists in three parts: GPMC initialization, device detection with parameters determination and finally bad block detection.

- **ONFI support.** The NAND identification starts with ONFI detection.
- **GPMC initialization.** The GPMC interface is configured as such it can be used for accessing NAND devices. The address bus is released since a NAND device does not use it. The data bus width is initially set to 8 bits; and changed to 16 bits if needed after device parameters determination. The following scheme is applied since NAND devices require different timings when compared to regular NOR devices:

Figure 25-11 and Table 25-10 describes the timings configured for NAND device access. The one clock cycle is 18.181 ns which correspond to 55 MHz frequency.

Figure 25-11. GPMC NAND Timings

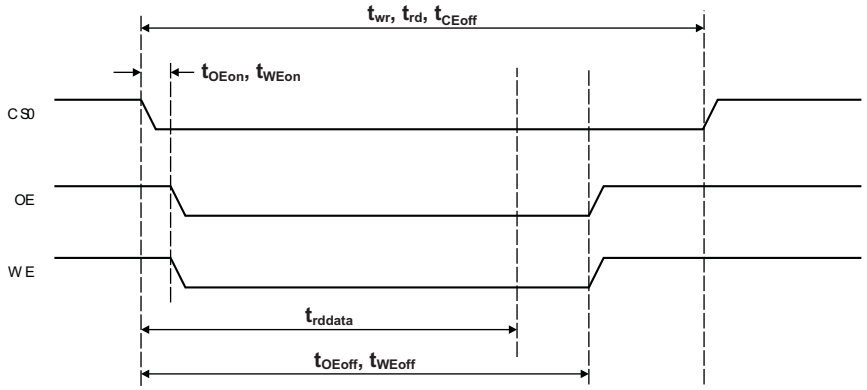


Table 25-10. NAND Timings Parameters

Parameter	Description	Value [clock cycles]
twr	write cycle period	30
trd	read cycle period	30
tCEon	CE low (not marked on the figure)	0
tOEon	CE low to OE low time	7
tWEon	CE low to WE low time	5
trddata	CE low to data latch time	21
tOEoff	CE low to OE high time	24
tWEoff	CE low to WE high time	22

- **Device detection and parameters.** The ROM Code first performs an initial wait for device auto initialization (with 250 ms timeout) with polling of the ready information. Then, it needs to identify the NAND type connected to the GPMC interface. The GPMC is initialized using 8 bits, asynchronous mode. The NAND device is reset (command FFh) and its status is polled until ready for operation (with 100 ms timeout). The ONFI Read ID (command 90h / address 20h) is sent to the NAND device. If it replies with the ONFI signature (4 bytes), then a Read parameters page (command ECh) is sent. The information shown in Table 25-11 is then extracted: page size, spare area size, number of pages per block, and the addressing mode. The remaining data bytes from the parameters page stream are simply ignored.

NOTE: PG2.x devices also check for the ONFI signature on the parameters page. Some NAND devices that did not work properly with PG1.x devices could work with PG2.x devices.

Table 25-11. ONFI Parameters Page Description

Offset	Description	Size (bytes)
6	Features supported	2
80	Number of data bytes per page	4
84	Number of spare bytes per page	2
92	Number of pages per block	4
101	Number of address cycles	1

If the ONFI Read ID command fails (it will be the case with any device not supporting ONFI) then the device is reset again with polling for device to be ready (with 100ms timeout). Then, the standard Read ID (command 90h / address 00h) is sent. If the Device ID (2nd byte of the ID byte stream) is recognized as being a supported device, then the device parameters are extracted from an internal ROM Code table. The list of supported devices is shown in Table 25-12.

Table 25-12. Supported NAND Devices

Capacity	Device ID	Bus Width	Page size
512 Mb	F0	x8	2048
512 Mb	C0	x16	2048
512 Mb	A0	x8	2048
512 Mb	B0	x16	2048
512 Mb	F2	x8	2048
512 Mb	C2	x16	2048
512 Mb	A2	x8	2048
512 Mb	B2	x16	2048
1 Gb	F1	x8	2048

Table 25-12. Supported NAND Devices (continued)

Capacity	Device ID	Bus Width	Page size
1 Gb	C1	x16	2048
1 Gb	A1	x8	2048
1 Gb	B1	x16	2048
2 Gb	DA	x8	2048
2 Gb	CA	x16	2048
2 Gb	AA	x8	2048
2 Gb	BA	x16	2048
2 Gb	83	x8	2048
2 Gb	93	x16	2048
4 Gb	DC	x8	2048
4 Gb	CC	x16	2048
4 Gb	AC	x8	2048/4096
4 Gb	BC	x16	2048/4096
4 Gb	84	x8	2048
4 Gb	94	x16	2048
8 Gb	D3	x8	2048/4096
8 Gb	C3	x16	2048/4096
8 Gb	A3	x8	2048/4096
8 Gb	B3	x16	2048/4096
8 Gb	85	x8	2048
8 Gb	95	x16	2048
16 Gb	D5	x8	2048/4096
16 Gb	C5	x16	2048/4096
16 Gb	A5	x8	2048/4096
16 Gb	B5	x16	2048/4096
16 Gb	86	x8	2048
16 Gb	96	x16	2048
32 Gb	D7	x8	2048/4096
32 Gb	C7	x16	2048/4096
32 Gb	A7	x8	2048/4096
32 Gb	B7	x16	2048/4096
32 Gb	87	x8	2048
32 Gb	97	x16	2048
64 Gb	DE	x8	2048/4096
64 Gb	CE	x16	2048/4096
64 Gb	AE	x8	2048/4096
64 Gb	BE	x16	2048/4096

When the parameters are retrieved from the ROM table: page size and block size is updated based on 4th byte of NAND ID data. Due to inconsistency amongst different manufacturers, only devices which has been recognized to be at least 2Gb (included) have these parameters updated. Therefore, the ROM Code supports 4kB page devices but only if their size, according to the table, is at least 2Gb. Devices which are smaller than 2Gb have the block size parameter set to 128kB (when page size is 2KB). Table 25-13 shows the 4th ID Data byte encoding used in ROM Code.

Table 25-13. 4th NAND ID Data Byte

Item	Description	I/O #							
		7	6	5	4	3	2	1	0
Page Size	1kB							0	0
	2kB							0	1
	4kB							1	0
	8kB							1	1
Cell type	2 levels					0	0		
	4 levels					0	1		
	8 levels					1	0		
	16 levels					1	1		
Block Size	64kB			0	0				
	128kB			0	1				
	256kB			1	0				
	512kB			1	1				

- **Reading NAND geometry from I2C EEPROM.** ROM supports a special boot mode called NANDI2C to support NAND devices whose geometry cannot be detected by the ROM automatically using methods described in the previous section (Figure 25-12). If this boot mode is selected, the ROM code tries to read NAND geometry from an I2C EEPROM. If the read is successful, ROM code then proceeds to next steps of NAND boot, beginning with reading bad blocks information.

  The list of pins that are configured by the ROM incase of NANDI2C boot mode (This is in addition to the NAND boot pins described in Table 25-16).

Table 25-14. Pins used for NANDI2C boot for I2C EEPROM access

Signal name	Pin used
I2C SCL	iic0_scl
I2C SDA	iic0_sda

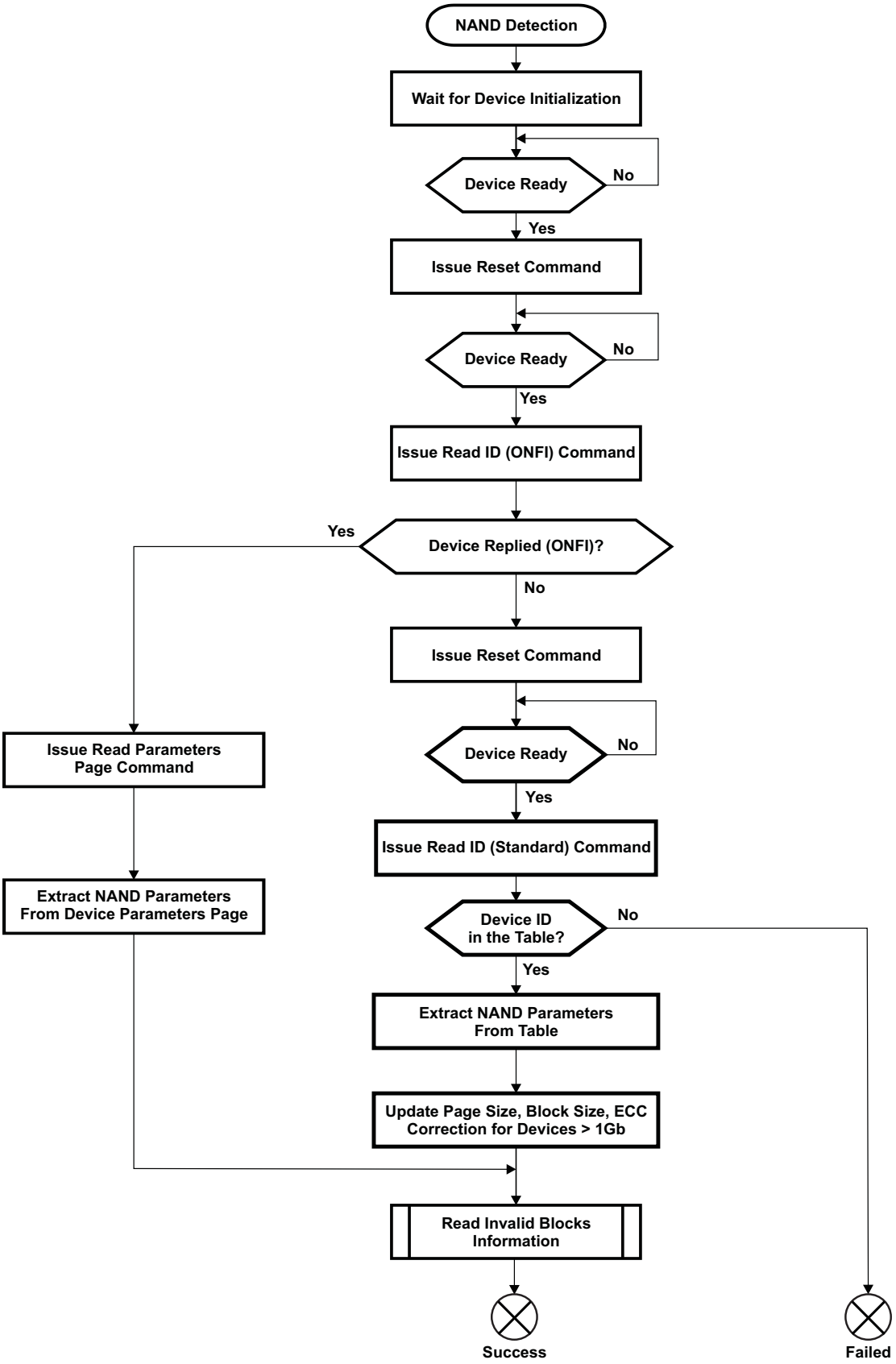
ROM accesses the I2C EEPROM at I2C slave address 50h and reads 7 bytes starting from address offset 80h. The format of this (NAND geometry information) is shown in Table 25-15.

The detection procedure is described in Figure 25-12. Once the device has been successfully detected, the ROM Code changes GPMC to 16-bit bus width if necessary.

Table 25-15. NAND Geometry Information on I2C EEPROM

Byte address	Information	
	Upper nibble	Lower nibble
80h	Magic Number – 10h	
81h	Magic Number – B3h	
82h	Magic Number – 57h	
83h	Magic Number – A6h	
84h	NAND column address (word/byte offset within a page) size in bytes, Example: 2	NAND row address (page offset) size in bytes. Example: 3
85h	Page size (2N) exponent “N”. Example (for page size of 2048): 11	Pages per block (2N) exponent “N” Example (for number of blocks 64): 6
86h	NAND bus width 0 → 8-bit, 1 → 16-bit	ECC Type 0→ No ECC, 1 → BCH8, 2 → BCH16

Figure 25-12. NAND Device Detection

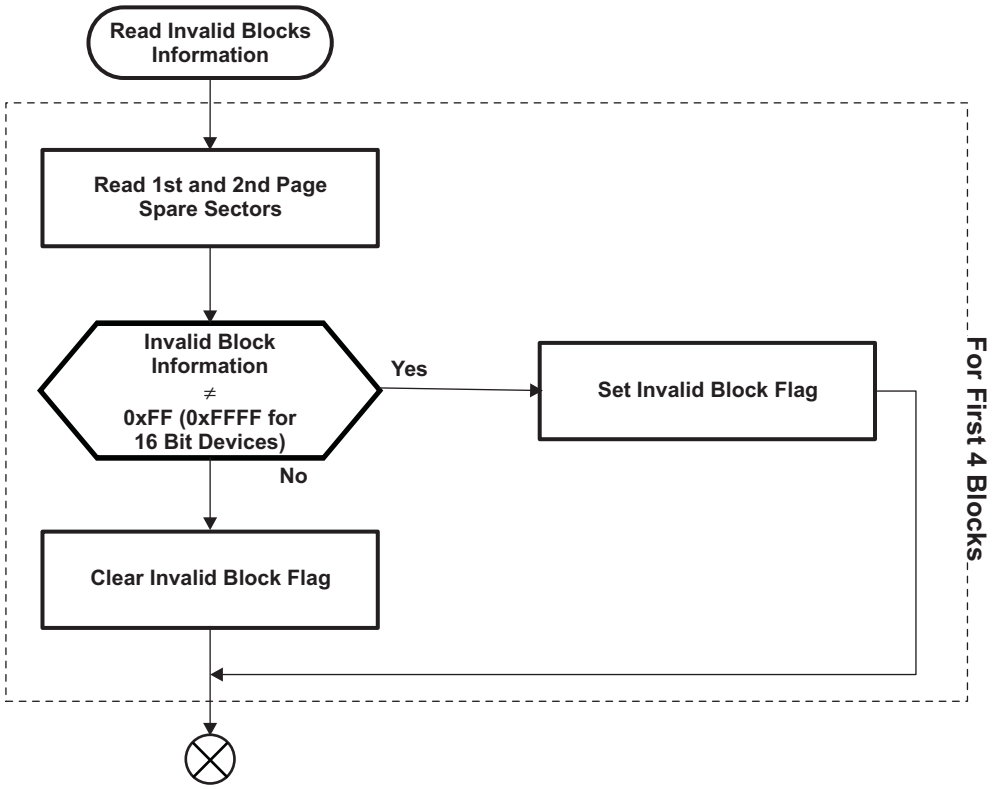


- **ECC correction.** The default ECC correction applied is BCH 8b/sector using the GPMC and ELM hardware.

  For device ID codes D3h, C3h, D5h, C5h, D7h, C7h, DEh, CEh when manufacturer code (first ID byte) is 98h the Cell type information is checked in the 4th byte of ID data. If it is equal to 10b then the ECC correction applied is BCH 16b/sector.
- **Bad block verification.** Invalid blocks are blocks which contain invalid bits whose reliability cannot be guaranteed by the manufacturer. Those bits are identified in the factory or during the programming and reported in the initial invalid block information located in the spare area on the 1st and 2nd page of each block. Since the ROM Code is looking for an image in the first four blocks, it must detect block validity status of these blocks. Blocks which are detected as invalid are not accessed later on. Blocks validity status is coded in the spare areas of the first two pages of a block (first byte equal to FFh in 1st and 2nd pages for an 8 bits device / first word equal to FFFFh in 1st and 2nd page for a 16-bit device).

  Figure 25-13 depicts the invalid block detection routine. The routine consists in reading spare areas and checking validity data pattern.

Figure 25-13. NAND Invalid Blocks Detection



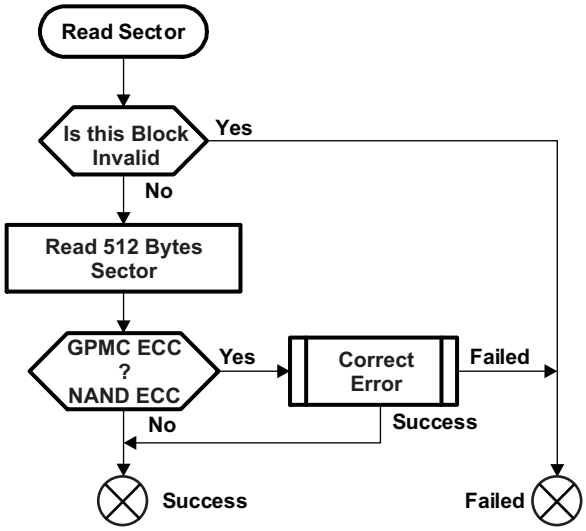
25.7.3.1 NAND Read Sector Procedure

The ROM Code reads data from NAND devices in 512 bytes sectors. The read function fails in two cases:

- The accessed sector is within a block marked as invalid
- The accessed sector contains an error which cannot be corrected with ECC

Figure 25-14 shows the read sector routine for NAND devices. The ROM Code uses normal read (command 00h 30h) for reading NAND page data.

Figure 25-14. NAND Read Sector Procedure



Page data can contain errors due to memory alteration. The ROM Code uses an ECC correction algorithm to detect and possibly correct those errors. The ECC algorithm used is BCH with capability for correcting 8b or 16b errors per sector.

The BCH data is automatically calculated by the GPMC on reading each 512 bytes sector. The computed ECC is compared against ECC stored in the spare area for the corresponding page. Depending on the page size, the amount of ECC data bytes stored in the corresponding spare area is different. Figure 25-15 and Figure 25-16 show the mapping of ECC data inside the spare area for 2KB-page and 4KB-page devices, respectively. If both ECC data are equal then the Read Sector function returns the read 512 bytes sector without error. Otherwise, the ROM Code tries to correct error(s) in the corresponding sector (this procedure is assisted by the ELM hardware) and returns the data if successful. If errors are uncorrectable, the function returns with FAIL.

The first two bytes in the spare area are always reserved for block state information:

- First word equal to FFFFh for a 16-bit device to check block validity
- First byte equal to FFh for an 8-bit device to check block validity and the second byte is a padding

Concerning the length of ECC section in bytes, extra bytes can be added to make the access ×16 compatible:

- For 8b BCH, there are 14 bytes ECC for each 512 bytes sector (that is, 13 bytes ECC + 1 byte padding)
- For 16b BCH, there are 26 bytes ECC for each 512 bytes sector (that is, no padding is needed)

Figure 25-15. ECC Data Mapping for 2KB Page and 8b BCH Encoding

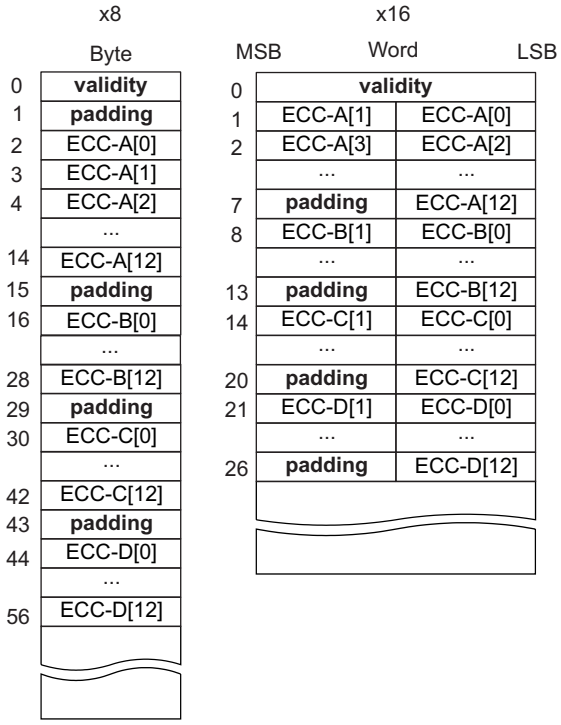
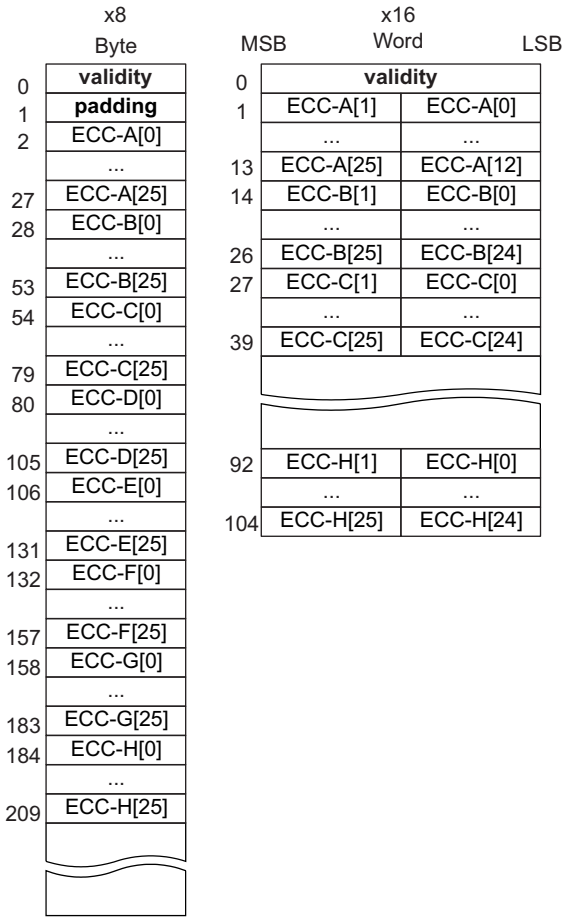


Figure 25-16. ECC Data Mapping for 4KB Page and 16b BCH Encoding



25.7.3.2 Pins Used

The list of pins that are configured by the ROM in case of NAND boot mode are in Table 25-16. Note that all the pins might not be driven at boot time.

Table 25-16. Pins Used for NAND Boot

Signal Name	Pin Used
cs0	$\overline{\text{GPMC\_CS[0]}}$
advn_ale	GPMC_$\overline{\text{ADV}}$_ALE
oen_ren	$\overline{\text{GPMC\_OE\_RE}}$
be0n_cle	GPMC_$\overline{\text{BE0}}$_CLE
wen	$\overline{\text{GPMC\_WE}}$
wait	GPMC_WAIT
clk	GPMC_CLK
ad0 - ad15	GPMC_D[15:0]

25.7.4 SD Cards

NOTE: PG2.x devices support booting from SD cards larger than 4 GB in size.

25.7.4.1 Overview

The ROM code supports booting from SD cards in the following conditions:

- SD cards compliant to low and high capacities.
- SD cards connected to interface #1.
- 3V VCC power supply, support for 3V I/O voltages.
- Clock Frequency: identification mode: 400 kHz; data transfer mode up to 10 MHz.
- Only one card connected to the bus.
- Raw mode, image data read directly from sectors in the user area.
- File system mode (FAT12/16/32 supported with or without Master Boot Record), image data is read from a booting file.

25.7.4.2 System Interconnection

An SD card can be connected to interface #1 typically through a card cage.

Notes:

- It may be possible to design the system so that an eSD memory type is connected to interface #1. However, the dual voltage IOs feature of interface #1 dedicates it preferably to cards.
- The ROM Code does not handle the card detection feature on card cage.

25.7.4.3 Pins Used

The list of device pins that are configured by the ROM in the case of SD boot mode are shown in Table 25-17. Note that all the pins might not be driven at boot time.

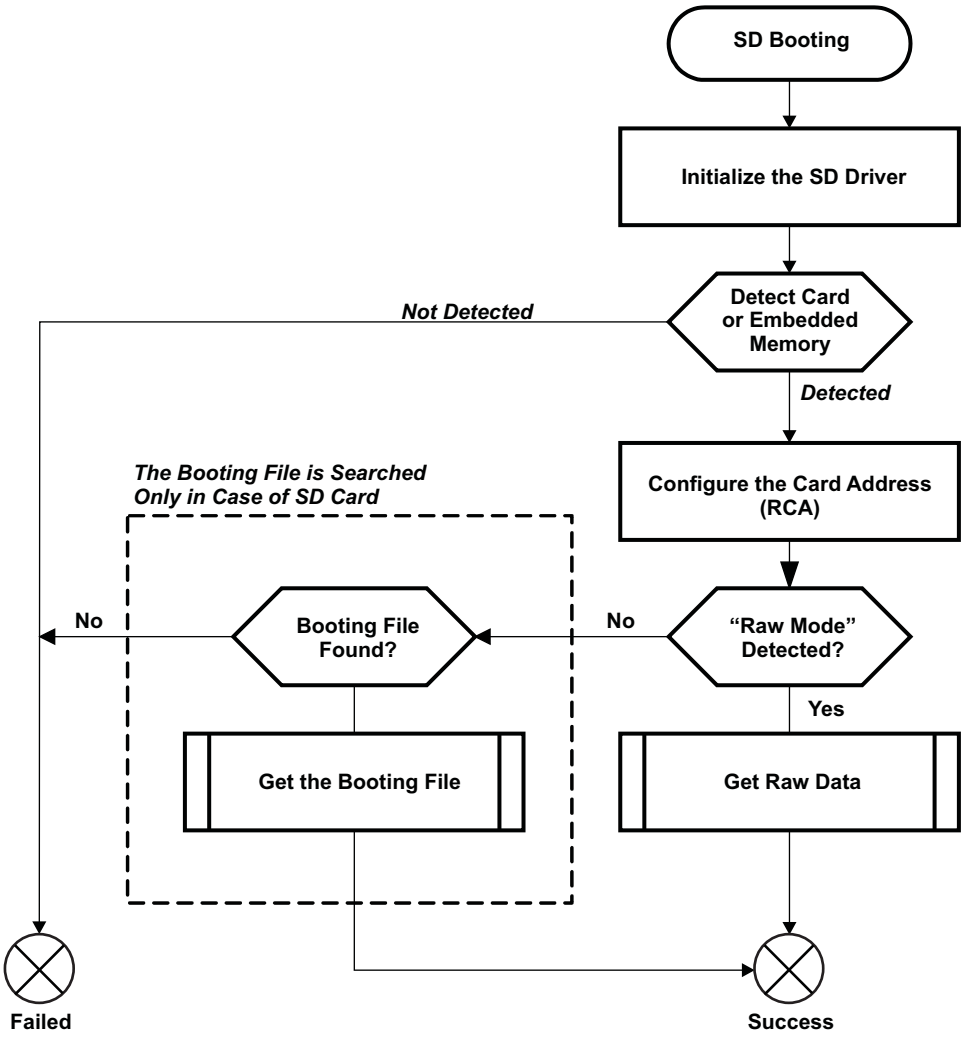
Table 25-17. Pins Used for SD Card Boot

Signal Name	Pin Used
clk	SD_CLK
cmd	SD_CMD
dat0	SD_DAT[0]
dat1	SD_DAT[1]
dat2	SD_DAT[2]
dat3	SD_DAT[3]

25.7.4.4 Booting Procedure

The high level flowchart of the eSD and SD booting procedure is depicted in Figure 25-17. The booting file is searched only in case of booting from a card. eSD embedded memories only support raw mode.

Figure 25-17. SD Booting



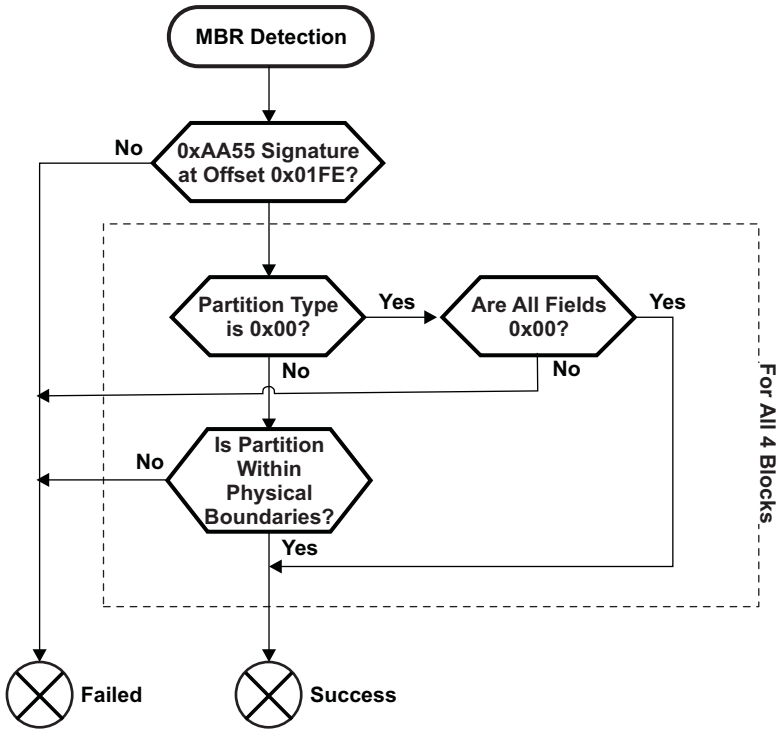
25.7.4.5 Initialization and Detection

The ROM Code initializes the memory device or card connected on interface #1 using the standard High-Voltage range (3.0V). If neither memory device nor card is detected then the ROM Code carries on to the next booting device. The standard identification process and Relative Card Address (RCA) assignment are used. However, the ROM Code assumes that only one memory or card is present on the bus. This first sequence is done using the CMD signal that is common to SD devices.

SD standards detail this phase as initialization phase. ACMD41 is only supported by the SD standard. ACMD41 (ACMD41 is made out of CMD55 and ACMD41) is sent and a response is expected from an SD device. If no response is received then it is assumed that no device is connected and the ROM Code exits the SD Booting procedure with FAIL. This is detection procedure is shown in Figure 25-18.

As previously mentioned, the contents of an SD card may be formatted as raw binary or within a FAT file system. eSD devices only support raw mode. The ROM Code reads out raw sectors from image or the booting file within the file system and boots from it.

Figure 25-18. SD Detection Procedure



25.7.4.6 SD Read Sector Procedure in Raw Mode

In raw mode the booting image can be located at one of the four consecutive locations in the main area: offset 0/20000h (128KB)/40000h (256KB)/60000h (384KB). For this reason a booting image shall not exceed 128KB in size. However it is possible to flash a device with an image greater than 128KB starting at one of the aforementioned locations. Therefore the ROM Code does not check the image size. The only drawback is that the image will cross the subsequent image boundary.

The raw mode is detected by reading sectors #0, #256, #512, #768. The content of these sectors is then verified for presence of a TOC structure as described in Section 25.10. The Configuration Header (CH) must be located in the first sector followed by a header. The CH might be void (only containing a CHSETTINGS item for which the Valid field is zero).

25.7.4.7 SD Read Sector Procedure in FAT Mode

SD cards may hold a FAT file system which ROM Code is able to read and process. The image used by the booting procedure is taken from a specific booting file named “MLO”. This file has to be located in the root directory on an active primary partition of type FAT12/16 or FAT32.

An SD card can be configured either as floppy-like or hard-drive-like.

- When acting as floppy-like, the content of the card is a single file system without any master boot record (MBR) holding a partition table
- When acting as hard-drive-like, an MBR is present in the first sector of the card. This MBR holds a table of partitions, one of which must be FAT12/16/32, primary and active.

The card should always hold an MBR. However, depending on the operating system, the SD card will be formatted either with partition(s) (using an MBR) or without. The ROM Code supports both types; this is described in the following section.

The ROM Code retrieves a map of the booting file from the FAT table. The booting file map is a collection of all FAT table entries related to the booting file (a FAT entry points to a cluster holding part of the file). The booting procedure uses this map to access any 512 byte sector within the booting file without involving ROM Code FAT module.

The sector read procedure utilizes standard SD raw data read function. The sector address is generated based on the booting memory file map collected during the initialization. Hence the ROM Code can address sectors freely within the booting file space.

25.7.4.8 FAT File System

This paragraph describes functionalities which are used by the ROM Code. It is not intended to fully describe the Master Boot Record and the FAT file system, but does include:

- How to recognize if a sector is the 1st sector of an MBR
- How to recognize if a sector is the 1st sector of a FAT12/16/32
- How to find the 1st cluster of the booting file
- How to buffer the booting file FAT entries

Some memory devices which support file systems can be formatted with or without MBR, therefore the first task of the ROM Code is to detect whether or not the device is holding an MBR in the first sector. If this is the case, an active FAT12/16/32 partition is searched in all four MBR partition entries, based on the Type field. If the MBR entries are not valid or if no useable partition is found then the ROM Code returns to the Booting procedure with FAIL. The Extended partitions are not checked, the booting file must reside in a primary partition.

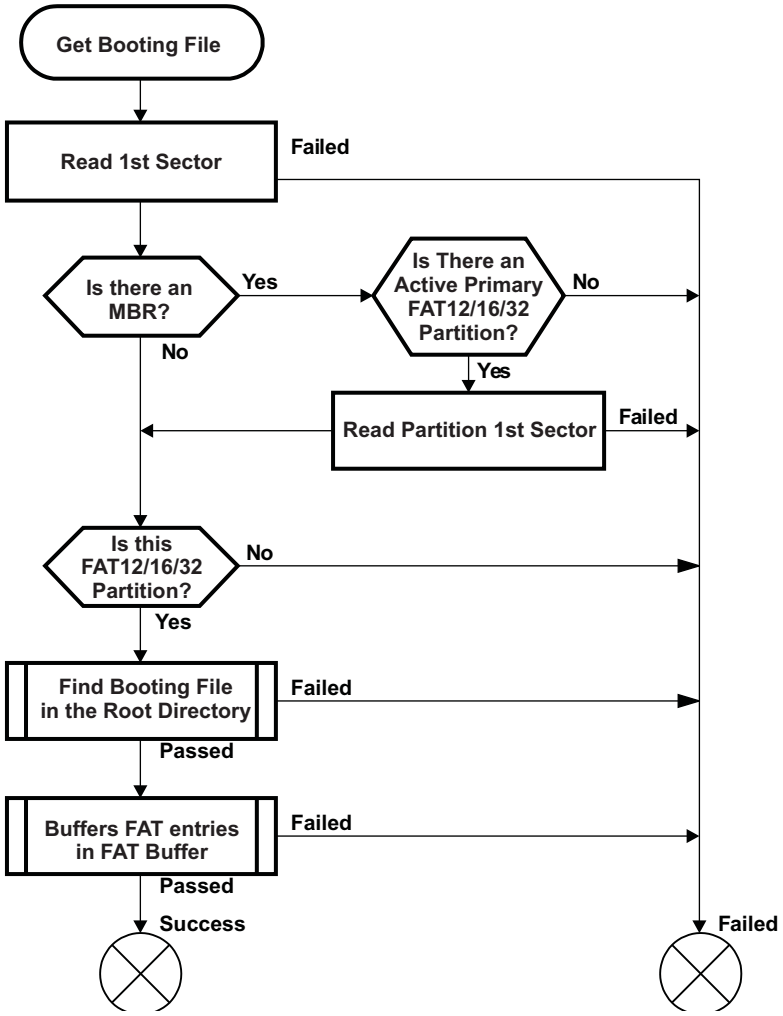
If a partition is found then its first sector is read and used further on. If no MBR is present (in case of a floppy-like system), the first sector of the device is read and used further on. The read sector is checked to be a valid FAT12/16 or FAT32 partition. If this fails, in case another partition type is used (Linux FS or any other) or if the partition is not valid, the ROM Code returns with FAIL.

Otherwise, the Root Directory entries are searched for a file named depending on the booting device. The Long File Names (LFN) format is not used and only File Names in 8.3 Format are searched for. If no valid file is found, the ROM Code returns with FAIL.

Once the file has been found, the ROM Code reads the File Allocation Table (FAT) and buffers the singly-linked chain of clusters in a FAT Buffer which will be used by the Booting Procedure to access the file directly sector by sector. For FAT12/16 and for FAT32 (valid if a specific flag has been set in the FAT32 Boot Sector), there exist multiples copies of the FAT (ROM Code supports only two copies). When buffering FAT entries, the two FATs are compared. If they are not the same, only entries from the last FAT are used. The FAT Buffer holds sector numbers and not cluster numbers. The ROM Code converts each cluster entry to one or several sector entries if applicable.

The whole process is described in Figure 25-19. Every part related to MBR or FAT12/16/32 is described in the next paragraph.

Figure 25-19. SD Booting, Get Booting File



25.7.4.8.1 Master Boot Record: MBR

The Master Boot Record is the 1st sector of a memory device. It is made out of some executable code and four partition entries. The aim of such a structure is to be able to divide the hard disk in partitions mostly used to boot different systems (Microsoft Windows™, Linux, …). Its structure is described in Table 25-18 and Table 25-19. The valid partition types searched by the ROM Code are described in Table 25-20.

Table 25-18. Master Boot Record Structure

Offset	Length [bytes]	Entry Description	Value
0000h	446	Optional Code	
01BEh	16	Partition Table Entry	(see Table 25-19)
01CEh	16	Partition Table Entry	(see Table 25-19)
01DEh	16	Partition Table Entry	(see Table 25-19)
01EEh	16	Partition Table Entry	(see Table 25-19)
01FEh	2	Signature	AA55h

Table 25-19. Partition Entry

Offset	Length [bytes]	Entry Description	Value
0000h	1	Partition State	00h: Inactive 80h: Active
0001h	1	Partition Start Head	H_s
0002h	2	Partition Start Cylinder and Sector	$C_s[7:0]$-$C_s[9:8]$-$S_s[5:0]$
0004h	1	Partition Type	Refer to Table 25-20 for partial partition types
0005h	16	Partition End Head	H_e
0006h	2	Partition End Cylinder and Sector	$C_e[7:0]$ - $C_e[9:8]$ - $S_e[5:0]$
0008h	4	First sector position relative to the beginning of media	$LBA_s = C_s.H.S + H_s.S + S_s - 1$
000Ch	4	Number of sectors in partition	$LBA_e = C_e.H.S + H_e.S + S_e - 1$ $Nb_s = LBA_e - LBA_s + 1$

Table 25-20. Partition Types

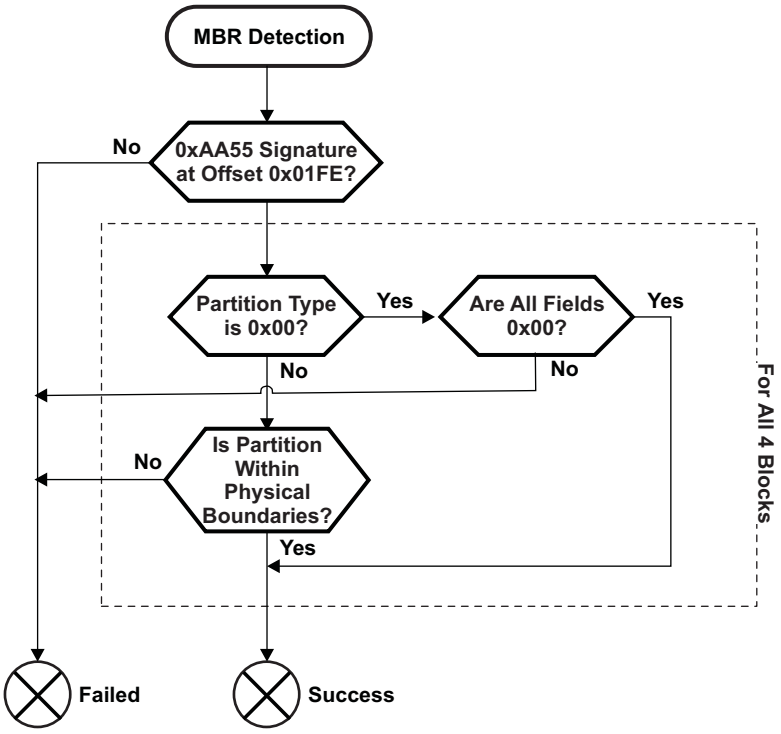
Partition Type	Description
01h	FAT12
04h, 06h, 0Eh	FAT16
0Bh, 0Ch, 0Fh	FAT32

The way the ROM Code detects whether a sector is the 1st sector of an MBR or not is described in Figure 25-20.

The ROM Code first checks if the signature is present. Each partition entry is checked:

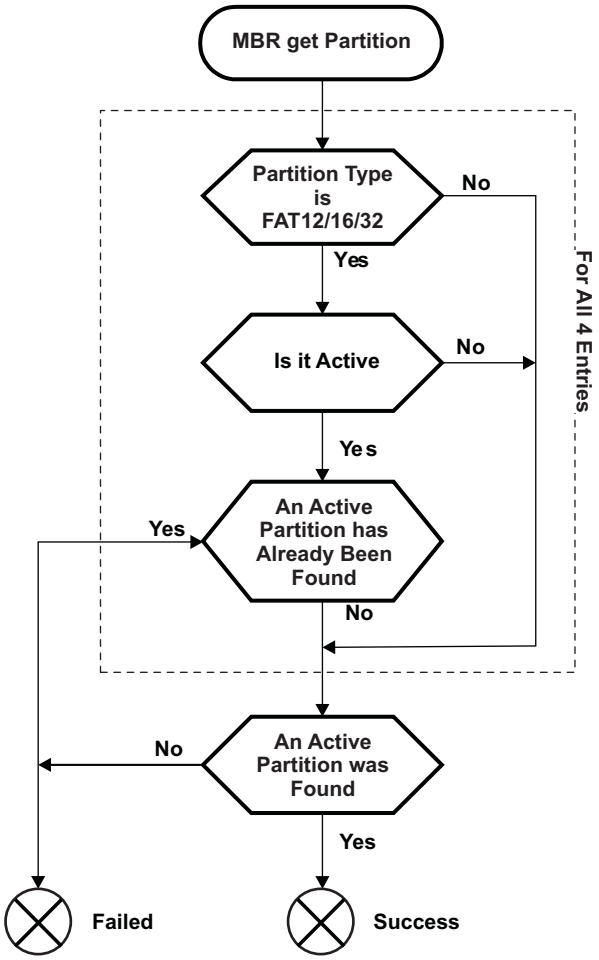
- If its type is set to 00h then all fields in the entry must be 00h
- The partition is checked to be within physical boundaries (the partition is located inside and its size fits the total physical sectors)

Figure 25-20. MBR Detection Procedure



Once identified, the ROM Code gets the partition using the procedure described in Figure 25-21. The partition type is checked to be FAT12/16 or FAT32. Its state must be 00h or 80h (if there is more than one active partition the test fails). The ROM Code returns with FAIL if no active primary FAT12/16/32 could be found.

Figure 25-21. MBR, Get Partition



25.7.4.8.2 FAT12/16/32 Boot Sector

The FAT file system is made out of several parts:

- Boot Sector which holds the BIOS Parameter Block (BPB)
- File Allocation Table (FAT) which describes the use of each cluster of the partition
- Data Area which holds the Files, Directories and Root Directory (for FAT12/16, the Root Directory has a specific fixed location)

The boot sector is described in Table 25-21. In the following description, all the fields whose names start with BPB_ are part of the BPB. All the fields whose names start with BS_ are part of the Boot Sector and not really part of the BPB (not mandatory); they are not used at all by the ROM Code.

Table 25-21. FAT Boot Sector

	Offset	Length [bytes]	Name	Description
	0000h	3	BS_jmpBoot	Jump Instruction to Boot Code (not used)
	0003h	8	BS_OEMName	Name of the System which created the partition
	000Bh	2	BPB_BytsPerSec	Counts of Bytes per sector (usually 512)
	000Dh	1	BPB_SecPerClus	Number of sectors per allocation unit
	000Eh	2	BPB_RsvdSecCnt	Number of reserved sectors for the Boot Sector For FAT12/16 is 1, for FAT32, usually 32
	0010h	1	BPB_NumFATs	Number of copies of FAT, usually 2
	0011h	2	BPB_RootEntCnt	For FAT12/16, number of 32 bytes entries in the Root Directory (multiple of BPB_BytsPerSec/32) For FAT32 this value is 0
	0013h	2	BPB_TotSec16	Total Count of sectors on the volume. If the size is bigger than 10000h or for FAT32, this field is 0 and BPB_TotSec32 holds the value
	0015h	1	BPB_Media	Media Type, usually F8h: fixed, non-removable
	0016h	2	BPB_FATSz16	For FAT12/16, size in sectors of one FAT For FAT32, holds 0
	0018h	2	BPB_SecPerTrk	Number of sectors per track, 63 for SD
	001Ah	2	BPB_NumHeads	Number of heads, 255 for SD
	001Ch	4	BPB_HiddSec	Number of sectors preceding the partition
	0020h	4	BPB_TotSec32	Total Count of sectors on the volume. If the size is smaller than 10000h (for FAT12/16), this field is 0 and BPB_TotSec16 is valid
FAT12/16	0024h	1	BS_DrvNum	Drive Number
	0025h	1	BS_Reserved1	00h
	0026h	1	BS_BootSig	Extended Boot Signature 29h. Indicates that the following 3 fields are present
	0027h	4	BS_VolID	Volume Serial Number
	002Bh	11	BS_VolLab	Volume Label
	0036h	8	BS_FilSysType	File system Type: “FAT12”, “FAT16”, “FAT32”. Note: This field is not mandatory (BS_), it cannot be used to indentify the partition type.
FAT32	0024h	4	BPB_FATSz32	Size in sectors of one FAT. Field BPB_FATSz16 must be 0
	0028h	2	BPB_ExtFlags	FAT Flags: [7]: 0 = FAT is mirrored; 1 = Only one FAT is used [3:0]: Number of used FAT if no mirroring used
	002Ah	2	BPB_FSVer	File system Version Number
	002Ch	4	BPB_RootClus	First Cluster number of the Root Directory
	0030h	2	BPB_FSInfo	Sector number of FSINFO Structure in the reserved-area, usually 1
	0032h	2	BPB_BkBootSec	If non-zero, indicates the sector number in the reserved-area of a copy of the Boot Sector
	0034h	12	BPB_Reserved	Reserved, set to 00h
	0040h	1	BS_DrvNum	Drive Number
	0041h	1	BS_Reserved1	00h
	0042h	1	BS_BootSig	Extended Boot Signature 29h. Indicates that the following 3 fields are present
	0043h	4	BS_VolID	Volume Serial Number
	0047h	11	BS_VolLab	Volume Label
	0052h	8	BS_FilSysType	File system Type: “FAT12”, “FAT16”, “FAT32”. Note: This field is not mandatory (BS_), it cannot be used to indentify the partition type.
	01FEh	2	BPB_Signature	AA55h

25.7.5 SPI

SPI EEPROMs or SPI flashes have an EEPROM or NOR Flash back-end and they connect to the device using the serial SPI protocol. These typical devices operate in three stages – the command stage, the address stage, and the data transfer stage. The command is usually an 8-bit value followed by the address (depending on the size of the device), followed by the data to be read or written. Because of the need for fewer pins, these devices are comparatively inexpensive, easy for board layout, and are the devices of choice when cost, complexity, and form factor are critical considerations.

25.7.5.1 Features

- Supports 12 MHz clock (50% duty cycle)
- Supports only SPI Mode 3 (clock polarity = 1, clock phase = 1)
- Supports only 24-bit addressable EEPROMs
- Supports only 4-pin SPI mode (CS, CLK, Serial Input, Serial Output)
- The boot devices must be connected to channel 0 and must support the Read Command (03h)
- The boot image is copied into internal memory and then executed

25.7.5.2 Initialization and Detection

The ROM Code initializes the SPI controller, pin muxing, and clocks for communicating with the SPI device. The controller is initialized in Mode 3 and the clock is setup to operate at 12 MHz. There is no specific device identification routine that is executed by the ROM code to identify whether a boot device is preset or not. If no SPI device is present, the sector read will return only FFFF FFFFh and the SPI boot will be treated as failed.

25.7.5.3 SPI Read Sector Procedure

The ROM Code reads SPI data from the boot device in 512-byte sectors. For each call to the SPI Read Sector routine, the SPI Read Command (03h) is sent along with the 24-bit start address of the data to be read.

From the next iteration onwards, a dummy value is transmitted on the master out line and the data is received on the master in line. This needs to be done because the SPI protocol always operates in full duplex mode. The dummy data transmitted by the ROM is the Read Command appended to the start address. The data from the boot device is received MSB first.

As the Cortex-A8 is a little-endian processor and the SPI operates in a big-endian format, this means that while writing to the Flash, care needs to be taken to write the image in a big-endian format. This avoids doing the endian conversion at boot time, thus improving boot performance.

25.7.5.4 Pins Used

The list of device pins that are configured by the ROM in the case of SPI boot mode are shown in Table 25-22. Note that all the pins might not be driven at boot time. The default state of the other SPI chip select pins could be low at POR, so care must be taken if any other devices are connected to CS[1], CS[2], or CS[3]. External logic must be used to ensure that the chip select to these devices are not enabled by default.

Table 25-22. Pins Used for SPI Boot

Signal Name	Pin Used
cs	$\overline{\text{SPI\_SCS[0]}}$
miso	SPI_D[0]
mosi	SPI_D[1]
clk	SPI_SCLK

25.7.5.5 Blocks and Sectors Search Summary

Table 25-23 summarizes the number of blocks and sectors that are searched during the memory booting from devices requiring image shadowing. NAND is organized with blocks, which are erasable units.

Table 25-23. Blocks and Sectors Searched on Non-XIP Memories

Memory	Maximum Number of Blocks Checked	Number of Sectors Searched
NAND	First 4	Number of sectors in one block(1)
SPI, eSD and SD cards (raw mode)	First 4	1(2)

(1) Depends on NAND geometry
(2) Regarding SD card booting in FAT mode, the file system area is searched for one file.

25.8 Peripheral Booting

25.8.1 Overview

The ROM Code can boot from three different peripheral interfaces:

- ETHERNET: 1000/100/10 Mbps Ethernet, using standard TCP/IP network boot protocols BOOTP and TFTP over the GMII interface
- PCIe: Two lanes at 250 MB/s
- UART: 115.2Kbps, 8 bits, even parity, 1 stop bit, no flow control

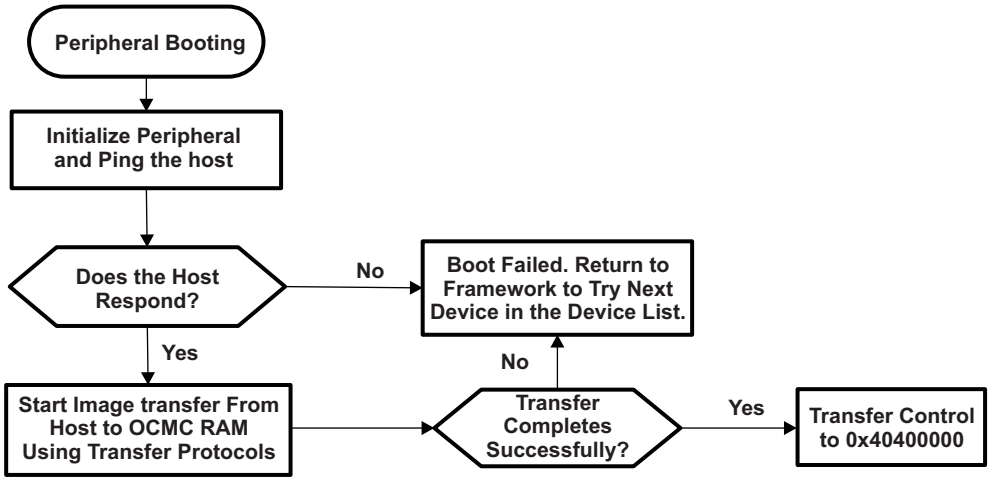
The purpose of booting from a peripheral interface is to download a boot image from an external host (typically a PC). This booting method is mostly used for programming flash memories connected to the device (for example, in the case of initial flashing, firmware update or servicing).

25.8.2 Boot Image Location and Size

The boot image is downloaded directly into internal RAM at the location 40400000h. The maximum size of downloaded image is 256KB.

25.8.3 Peripheral Boot Procedure Overview

Figure 25-22. Peripheral Booting Procedure



25.8.4 Ethernet Boot Procedure

25.8.4.1 Device Initialization

- Ethernet boot uses the CPGMAC0 subsystem of the device, connected to external Ethernet PHY using the GMII0 and MDIO pins.
- Device uses EFUSE register MAC_ID0 for Ethernet MAC address of the device.
- Device detects if the PHY is alive on the MDIO interface and
  - Reads the STATUS register to check if Ethernet link is active
  - Reads the CONTROL register to detect the auto-negotiated mode of operation
    - Is the mode full-duplex or half duplex
    - Speed of operation, 1000/100/10 Mbps

25.8.4.2 BOOTP (RFC 951)

The device then proceeds to obtain the IP and Boot information using BOOTP protocol. The device prepares and broadcasts the BOOTP message that has the following information:

- Device MAC address in “chaddr” field – to uniquely identify the device to the server.
- “Vender-class-identifier” option number 60 (RFC 1497, RFC 1533). Servers could use this information to identify the device type. The value present is "DM816x ROM v1.0"
- ”Client-identifier” option number 61 (RFC 1497, RFC 1533). This has the ASIC-ID structure which contains additional info for the device.

The device then expects a BOOTP response that provides the following information for the booting to proceed:

- Device IP address from “yiaddr” field
- Subnetmask from extended option 1 (RFC 1497, RFC 1533)
- Gateway IP from extended option number 3 (RFC 1497, RFC 1533) or from “giaddr” field of BOOTP response.
- Boot image filename from “file” field

  Timeouts and retries
- Exponentially increasing timeouts starting from 4s
- Five retries

25.8.4.3 FTP (RFC 1350)

After a successful BOOTP completion, the device initiates the TFTP download of the boot image into SRAM. The device has the capability to reach TFTP server within the local subnet or outside, though the gateway.

Timeouts and retries

- Timeout of 1s to receive a response for the READ request
- 5 retries for the READ request
- Retries are managed by server once data transfer starts (server re-sends a data packet if the ACK was not received within a timeout value)
- Device has a 60s timeout to complete the data transfer, to handle the scenario if the server dies in the middle of a data transfer

25.8.4.4 Pins Used

The list of pins that are configured by the ROM in case of Ethernet boot mode are in Table 25-24. Note that all the pins might not be driven at boot time.

Table 25-24. Pins Used for Ethernet Boot

Signal Name	Pin Used
col	gmii0_col
crs	gmii0_crs
gtx_clk	gmii0_gtxclk
rx_clk	gmii0_rxclk
rxd0-rxd7	gmii0_rxd0-rxd7
rxdv	gmii0_rxdv
rxer	gmii0_rxer
tx_clk	gmii0_txclk
txd_0-txd7	gmii0_txd0-txd7
txen	gmii0_txen
mclk	gmii0_mclk
mdio	gmii0_mdio

25.8.5 PCIe Boot Procedure

25.8.5.1 Device Initialization

- ROM code configures the PCIe vendor ID as 104Ch
- ROM code configures the PCIe device ID as B800h.
- The BAR window sizes for inbound translation are configurable as per the following tables.
- Only legacy interrupt A is enabled. MSI is disabled.
- The device is configured in the D0 power state.
- The link and device capability is configured for maximum flexibility.
- The PCIe settings used by the ROM can be overridden after boot by the Root Complex.

Figure 25-23. PCIe Peripheral Booting Procedure

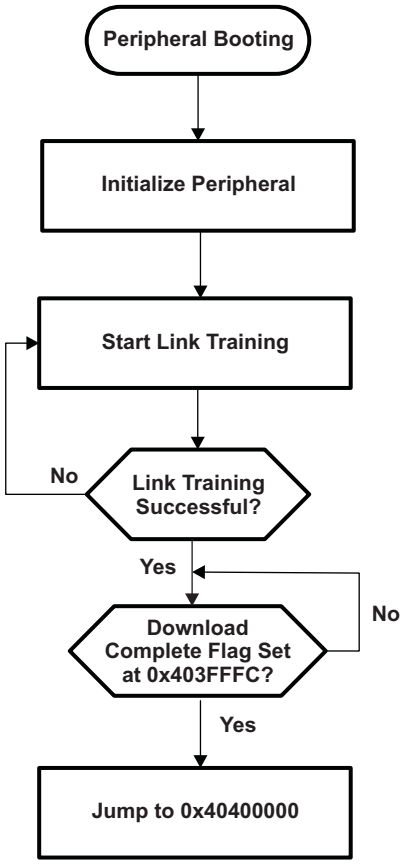


Table 25-25. PCIe 32 BAR Window Size Configuration (PG1.x Devices)

BAR0	BAR1	BAR2		BAR3		BAR4	
		CS0BW	Size	CS0MUX[1:0]	Size	CS0WAIT	Size
4 KB	8 MB	0	0	00	0	0	0
		1	8 MB	01	64 MB	1	256 MB
				10	128 MB		
				11	256 MB		

Table 25-26. PCIe 32 BAR Window Size Configuration (PG2.x Devices)

BAR0	BAR1	BAR2		BAR3		BAR4	
		CS0BW	Size	CS0MUX[1:0]	Size	CS0WAIT	Size
4 KB	8 MB	0	0	00	0	0	0
		1	16 MB	01	32 MB	1	256 MB
				10	64 MB		
				11	128 MB		

Table 25-27. PCIe 64 BAR Window Size Configuration (PG1.x Devices)

BAR0/1	BAR 2/3		BAR 4/5	
	CS0WAIT :CS0BW	Size	CS0MUX[1:0]	Size
4 KB	00	0	00	0
	01	8 MB	01	64 MB
	10	64 MB	10	128 MB
	11	128 MB	11	256 MB

Table 25-28. PCIe 64 BAR Window Size Configuration (PG2.x Devices)

BAR0/1	BAR 2/3		BAR 4/5	
	CS0WAIT :CS0BW	Size	CS0MUX[1:0]	Size
4 KB	00	0	00	0
	01	256 MB	01	1 GB
	10	512 MB	10	2 GB
	11	1 GB	11	4 GB

Table 25-29. PCIe BAR Window Base Address and Offset Configuration

BAR	Base Address	Offset	Comments
0	0000 0000h	0000 0000h	
1 (64bit BAR0)	8080 0000h	4040 0000h	OCMC RAM1
2	8100 0000h	0800 0000h	GPMC
3 (64 Bit BAR2)	8180 0000h	8000 0000h	DDR 0
4	8200 0000h	C000 0000h	DDR 1

Note that the Base Address and Offset configurations can be changed from the host after the enumeration is complete.

25.8.6 UART Boot Procedure

25.8.6.1 Device Initialization

- UART boot uses UART0
- UART0 is configured to run at 115200 baud, 8-bits, even parity, 1 stop bit and no flow control.

25.8.6.2 Boot Image Download

- UART boot uses x-modem client protocol to receive the boot image.
- Utilities like hyperterm, teraterm, minicom can be used on the PC side to download the boot image to the board
- With x-modem packet size of 1K throughout is roughly about 4KBytes/Sec.
- The ROM code will ping the host 10 times in 3s to start x-modem transfer. If host does not respond, UART boot will timeout.
- Once the transfer has started, if the host does not send any packet for 3s, UART boot will time out
- If the delay between two consecutive bytes of the same packet is more than 2ms, the host is requested to re-transmit the entire packet again
- Error checking using the CRC-16 support in x-modem. If an error is detected, the host is requested to re-transmit the packet again

25.8.6.3 Pins Used

The list of pins that are configured by the ROM incase of UART boot mode are in Table 25-30. Note that all the pins might not be driven at boot time

Table 25-30. Pins Used for UART Boot

Signal Name	Pin Used
rx	UART0_RXD
tx	UART0_TXD

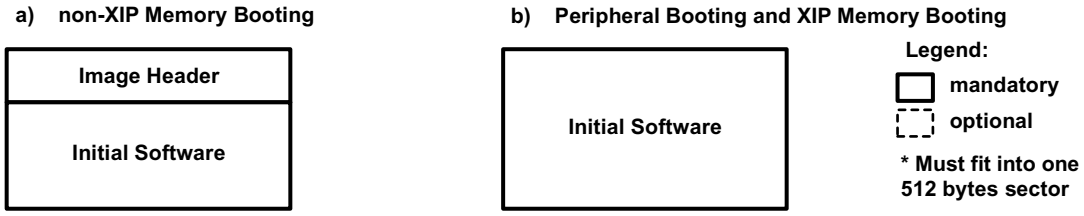
25.9 Image Format

25.9.1 Overview

All preceding sections describe how the ROM Code searches and detects a boot image from a memory or a peripheral device type. This section describes the format of the boot image. A boot image is a header containing the destination address and size of the image for non-XIP memory devices.

The mandatory section of a boot image contains the software that will be loaded into the memory and executed. An overview of the image formats is shown in Figure 25-24.

Figure 25-24. Image Formats



(a) Non-XIP Memory Booting: Image must begin with a header that contains information on image size and destination address

(b) Peripheral Booting and XIP Memory Booting: When memory device is of XIP type (for example, NOR), the header is not needed and the image can contain code for direct execution. The same image format is used for peripheral booting (where the code is transferred to internal RAM).

25.9.1.1 Image Format

When the booting memory device is non-XIP, the image must contain a small header with the size of the software to load and the destination address where to store it.

The header is not needed when booting from an XIP memory device (for example, NOR) or in case of peripheral booting. In this case, the peripheral or memory booting image starts directly with executable code.

Table 25-31. Image Format

Field	Non-XIP Device (offset)	XIP Device (offset)	Size (bytes)	Description
Size	0000h	-	4	Size of the image
Destination	0004h	-	4	Address where to store the image / code entry point
Image	0008h	0000h	x	Executable code

Note: the “Destination” address field stands for both:

- • Target address for the image copy from the non-XIP storage to the target XIP location (for example, internal RAM or SDRAM)
- • Entry point for image code

In other words, you must take care to locate the code entry point to the target address for image copy.

25.10 Image Execution

25.10.1 Overview

One of the early steps of the ROM Code execution is to search for a boot image from the requested medium (configured by the MBOOT pins) and copy it to RAM if needed. If the boot interface is non-XIP, then the image is simply copied to the provided destination address (internal or external RAM) and then executed. If the boot interface is of XIP type, then the image copy is not needed and execution is directly given to the XIP memory.

25.10.2 Execution

The image is executed at the time the ROM Code performs the branch to the first executable instruction inside the Initial SW. In non-XIP, the execution address is the first word after the header. The branch is performed in public ARM supervisor mode. The R0 register points to the Booting Parameters structure which contains various information about the booting execution. Table 25-32 details this structure.

Table 25-32. Booting Parameters Structure

Offset	Field	Size (bytes)	Description
0h	Booting Message	4	Last received Booting Message.
4h	Memory booting device descriptor address	4	Pointer to the memory device descriptor that has been used during the memory booting process
5h	Current Booting Device	1	Code of device used for booting 0h – void, no device 1h – XIP memory 2h – XIPWAIT memory (wait monitoring on) 3h – NAND 4h – OneNAND 5h – SD port 1 (Card) 6h – SD port 2 (JC64) 7h – EMIF NVM 43h – UART3 45h – USB (internal) 46h – USB-ULPI
6h	Reset Reason	1	Current reset reason bit mask (bit = 1-event present) [0] – power-on reset [1] – global software reset [4] – MPU watchdog reset [6] – external warm reset Other bits – reserved
7h	Reserved	1	Reserved

25.11 Services for HLOS Support

The Cortex-A8 in the device restrict accesses to few ARM coprocessor registers. In order for the HLOS to access secure registers for L2 cache maintenance and wake up of slave CPU(s), the ROM Code provides primitives that can be called.

The list of services is:

- L2 cache maintenance
  - L2 Cache Set Debug Register
  - L2 Cache Clean & Invalidate Range of PA
  - L2 Cache Set Control Register
- Multicore infrastructure maintenance
  - Read AuxCoreBoot 0 and 1 Register
  - Modify AuxCoreBoot 0
  - Write AuxCoreBoot 1
  - Read Control and Status Register
  - Clear Control and Status Register

25.12 Tracing

Tracing in the ROM Code consist in three 32 bit vectors for which each bit corresponds to a particular “way point” in the ROM Code execution sequence (refer to Table 25-5). Tracing vectors are initialized at the very beginning of the startup phase and updated all along the boot process.

There are two sets of tracing vectors. The first set is the current trace information (after cold or warm reset). The second set holds a copy of trace vectors collected at the first ROM Code run after cold reset. As a consequence after a warm reset, it is possible to have visibility on the boot scenario that occurred during cold reset.

Table 25-33. Tracing Vectors

Trace vector	Bit #	Group	Meaning
1	0	General	Passed the public reset vector
1	1	General	Entered main function
1	2	General	Running after the cold reset
1	3	Boot	Main booting routine entered
1	4	Memory Boot	Memory booting started
1	5	Peripheral Boot	Peripheral booting started
1	6	Boot	Booting loop reached last device
1	7	Boot	Header found
1	8	Boot	Booting Message “Skip Peripheral Booting” received
1	9	Boot	Booting Message “Change Device” received
1	10	Peripheral Boot	Booting Message “Peripheral booting” received
1	11	Peripheral Boot	Booting Message “Get Asic Id”
1	12	Peripheral Boot	Device initialized
1	13	Peripheral Boot	Asic Id sent
1	14	Peripheral Boot	Image received
1	15	Peripheral Boot	Peripheral booting failed
1	16	Peripheral Boot	Booting Message not received (timeout)
1	17	Peripheral Boot	Image size not received (timeout)
1	18	Peripheral Boot	Image not received (timeout)
1	19	Reserved	Reserved
1	20	Configuration Header	CHSETTINGS found
1	21	Configuration Header	CHSETTINGS executed

Table 25-33. Tracing Vectors (continued)

Trace vector	Bit #	Group	Meaning
1	22	Configuration Header	CHRAM executed
1	23	Configuration Header	CHFLASH executed
1	24	Configuration Header	CHMMCSD clocks executed
1	25	Configuration Header	CHMMCSD bus width executed
1	26	Reserved	Reserved
1	27	Reserved	Reserved
1	28	Reserved	Reserved
1	29	Reserved	Reserved
1	30	Reserved	Reserved
1	31	Reserved	Reserved
2	0	Companion chip	Phoenix detected
2	1	Companion chip	VBUS detected
2	2	Companion chip	VMMC switched on
2	3	Companion chip	VUSB switched on
2	4	USB	USB connect
2	5	USB	USB configured state
2	6	USB	USB VBUS valid
2	7	USB	USB session valid
2	8	Reserved	Reserved
2	9	Reserved	Reserved
2	10	Reserved	Reserved
2	11	Reserved	Reserved
2	12	Memory Boot	Memory booting trial 0
2	13	Memory Boot	Memory booting trial 1
2	14	Memory Boot	Memory booting trial 2
2	15	Memory Boot	Memory booting trial 3
2	16	Memory Boot	Execute image
2	17	Reserved	Reserved
2	18	Memory Peripheral Boot	Jumping to Initial SW
2	19	Reserved	Reserved
2	20	Reserved	Reserved
2	21	Reserved	Reserved
2	22	Reserved	Reserved
2	23	Reserved	Reserved
2	24	Reserved	Reserved
2	25	Reserved	Reserved
2	26	Reserved	Reserved
2	27	Reserved	Reserved
2	28	Reserved	Reserved
2	29	Reserved	Reserved
2	30	Reserved	Reserved
2	31	Reserved	Reserved
3	0	Memory Boot	Memory booting device NULL
3	1	Memory Boot	Memory booting device XIP
3	2	Memory Boot	Memory booting device XIPWAIT
3	3	Memory Boot	Memory booting device NAND
3	4	Memory Boot	Memory booting device OneNAND

Table 25-33. Tracing Vectors (continued)

Trace vector	Bit #	Group	Meaning
3	5	Memory Boot	Memory booting device MMCSD1
3	6	Reserved	Reserved
3	7	Memory Boot	Memory booting device MMCSD2
3	8	Reserved	Reserved
3	9	Reserved	Reserved
3	10	Memory Boot	Memory booting device EMIF LPDDR2-NVM
3	11	Reserved	Reserved
3	12	Reserved	Reserved
3	13	Reserved	Reserved
3	14	Reserved	Reserved
3	15	Reserved	Reserved
3	16	Reserved	Reserved
3	17	Reserved	Reserved
3	18	Peripheral Boot	Peripheral booting device UART3
3	19	Reserved	Reserved
3	20	Peripheral Boot	Peripheral booting device USB
3	21	Peripheral Boot	Peripheral booting device USB ULPI
3	22	Peripheral Boot	Peripheral booting device NULL
3	23	Reserved	Reserved
3	24	Reserved	Reserved
3	25	Reserved	Reserved
3	26	Reserved	Reserved
3	27	Reserved	Reserved
3	28	Reserved	Reserved
3	29	Reserved	Reserved
3	30	Reserved	Reserved
3	31	Reserved	Reserved

On-Chip Debug Support

This chapter describes the on-chip debug support.

Topic **Page**

26.1 Introduction

Debugging a system containing an embedded processor involves an environment that connects high-level debugging software running on a host computer, to a low-level debug interface supported by the target device. An emulator device facilitates communication between the host debugger and the emulation logic on the target chip.

The emulation layer is a combination of hardware and software that connects the host debugger to the target system. It utilizes one or more hardware interfaces and/or protocols to convert actions dictated by the debugger user to JTAG commands and scans that exercise the core hardware.

The device implements the following debug features:

- Debug interface with 5 standard JTAG pins + RTCK pin + EMU[4:0] pins, which supports:
  - Standard JTAG (IEEE1149.1) protocol, and adaptive clocking scheme of ARM® ARM9™ processor
  - Cross-triggering between devices, and debug boot mode control via EMU[1:0] pins
  - Export of system trace data over EMU[4:0] pins
- Support debug capabilities of individual processor, including:
  - Advanced Event Triggering (AET) for C674x DSP
  - Six ARM968™ processor cores enhanced with TI's ICECrusher-9 modules, residing in the three HDVICP subsystems
  - Cortex-A8 core enhanced with TI's ICECrusher-CS module, residing in the MPU subsystem
  - Debug activities shall not been interrupted by events of clock-gate or power-down of an individual subsystem.
- Support limited debug capabilities of Hardware Accelerators (HWAs), residing in the three HDVICP subsystems (6 HWAs per HDVICP)
- Support multi-core debug capabilities, including:
  - Two cross-triggering channels (Trigger0, and Trigger1), shared by:
    - All processors in the device
    - External device (via EMU[1:0] pins)
    - Other debug modules that have only input trigger line - L3 Statistics Collector, MIPI® System Trace Module (STM), and HDVICP HWAs + Software Message and System Event Trace (SMSET) module
  - Synchronized (global) run with C674x DSP, Cortex-A8, Media Controller, ARM968s, and all HWAs
- Support debug related reset features, including:
  - Debugger-generated system reset and subsystem resets via ICEPick-D
  - Debugger-generated local resets to individual processor (both DSP Subsystem and ICECrusher can generate a local reset)
  - Blocking certain system reset generated by applications
  - Blocking subsystem local resets generated by applications
  - Debug logics shall survive all resets except global cold resets or nTRST/TLR
  - nTRST/TLR resets only affect debug and test logics. It does not affect any functional operation of the device
  - Wait-In-Reset (WIR) debug boot mode
- Support system memory accesses and debug components access via the DAP port without halting any processors
- Support Embedded Trace Buffer (ETB) based CPU trace for Cortex-A8 and C674x DSP, including:
  - Support Cortex-A8 program trace, timing trace, and data trace (address only)
  - Support C674x DSP program trace, timing trace, and data trace
  - A single 32KB ETB shared by Cortex-A8, C674x DSP, and system trace

- Support STM-based system trace as following:
  - SW messages generated by application code running on each processor
  - HW messages generated by bus monitors and system event monitors
  - Trace data can be exported to either EMU[4:0] pins or ETB buffer
- Peripherals, memories, as well as MMU units are aware of debug activities:
  - Debug access to invalid memory locations, which might be reserved, clock-gated, or powered-down, does not cause system hang
  - Debug activities does not affect the correctness of an application:
    - Peripheral, which is sensitive to a debug access, distinguishes access initiated from debugger versus from application
    - Peripheral, which is sensitive to a debug event (CPU halt), behaves properly during processor suspension
    - Errors triggered by a debug access does not cause any undesired interrupt or exception to the normal execution of a processor
    - The result of MMU table-walk caused by a debug access is not written into TLB; MMU translation fault caused by debug access does not trigger the translation fault interrupt/exception

The following commonly used debug features are not supported in the device:

- Off-chip trace of C674x DSP (via pins)
- Off-chip trace of Cortex-A8 (via pins)

26.2 Debug Interface

26.2.1 IEEE1149.1 JTAG Mode

The target debug interface has the following signals:

- Five standard IEEE1149.1 JTAG signals:
  - nTRST
  - TCK
  - TMS
  - TDI
  - TDO
- A return clock (RTCK) due to the clocking requirements of the ARM968 processor
- Two EMU[1:0] or five EMU[4:0] TI extensions, depending on the pin count (14 pins or 20 pins) in the JTAG header of the device.

Table 26-1 describes the IEEE1149.1 signals.

Table 26-1. IEEE1149.1 Signals

Device Pin Name	Internal Signal Name	Type[1]	Function	Description
$\overline{\text{TRST}}$	nTRST	I	Test reset	When asserted (active low), causes all test and debug logic in the device to be reset along with the IEEE1149.1 interface
TCLK	TCK	I	Test clock	This is the test clock used to drive an IEEE1149.1 TAP state-machine and logic. Depending on the emulator attached to the device, this is a free running clock or a gated clock depending on RTCK monitoring.
RTCK	RTCK	O	Returned (synchronized) test clock	Depending on the emulator attached to the device, the JTAG signals are clocked from RTCK or RTCK is monitored by the emulator to gate TCK.
TMS	TMS	I	Test mode select	Directs the next state of the IEEE1149.1 TAP state-machine
TDI	TDI	I	Test data input	Scan data input to the device
TDO	TDO	O	Test data output	Scan data output by the device
EMU0	EMU0	I/O	Emulation 0	Channel 0 trigger, debug boot mode, or system trace port
EMU1	EMU1	I/O	Emulation 1	Channel 0 trigger, debug boot mode, or system trace port
EMU2[2]	EMU2	I/O	Emulation 2	System trace port
EMU3[2]	EMU3	I/O	Emulation 3	
EMU4[2]	EMU4	I/O	Emulation 4	

(1) I = Input; O = Output; I/O = Bidirectional
(2) 20-pin JTAG header only

For information about internal pullup/pulldown resistors on the debug interface pins, see the device datasheet.

For information about the JTAG ID code value, see the *TMS320DM816x DaVinci Digital Media Processors Silicon Errata* (SPRZ329).

26.2.2 Trace Connector and Board Layout Considerations

For information about board design guidelines for Trace Advanced Emulation, see *Emulation and Trace Headers Technical Reference Manual* (SPRU655).

26.3 Debugger Connection

26.3.1 ICEPick Module

The debugger connects to the device through its JTAG interface. The first level of debug interface seen by the debugger is the ICEPick module embedded in the Debug Subsystem.

NOTE: ICEPick version D (ICEPick-D) is used in the device.

System-on-chip (SoC) designs typically have multiple processors, each having a JTAG TAP embedded in the processor. The ICEPick module manages these TAPs and the power, reset, and clock controls for modules that have TAPs.

The ICEPick module is visible only from the debugger point of view, and thus cannot be programmed by application software. The debugger can configure ICEPick through its own TAP controller. The ICEPick TAP has an instruction length of 6 bits and is the primary TAP. It is always visible in the scan chain and is used to control and monitor the other secondary TAPs.

ICEPick provides the following debug capabilities:

- Debug connect logic for enabling or disabling most ICEPick instructions
- Dynamic TAP insertion:
  - Serially linking up to 32 TAP controllers
  - Individually selecting one or more of the TAPs for scan without disrupting the instruction register (IR) state of other TAPs
- Power, reset, and clock management:
  - Provides the power and clock states of each domain
  - Provides debugger control of the power domain of a processor. Can force the domain power and clocks on, and prohibit the domain from being clock-gated or powered down while a debugger is connected.
  - Applies system reset
  - Provides wait-in-reset (WIR) boot mode
  - Provides global and local WIR release
  - Provides global and local reset blocking

The ICEPick module implements a connect register, which must be configured with a predefined key to enable the full set of JTAG instructions. When the debug connect key is properly programmed, ICEPick signals and subsystems emulation logics should be turned on.

For more information about ICEPick dynamic TAP insertion, see Section 26.3.3.

For more information about ICEPick power, reset, and clock management features, see Section 26.6.

26.3.2 Debug Boot Modes

ICEPick supports debug boot modes determined by the level of the EMU0 and EMU1 pins upon a rising edge of the Power-On Reset (POR) signal. At POR, EMU0 and EMU1 are automatically configured as inputs. The EMU0 and EMU1 pins are free when POR is released.

Table 26-2 summarizes the debug boot modes.

Table 26-2. Debug Boot Modes Upon POR

EMU1	EMU0	TAPs in the TDI → TDO Path	Other Effects/Comments
0	0	None	Reserved (do not use)
0	1	None	Reserved (do not use)
1	0	ICEPick	TAP only + WIR mode
1	1	ICEPick	TAP only (default mode)

In ICEPick-only configuration, none of the secondary TAPs are selected. The ICEPick TAP is the only TAP between device-level TDI and TDO pins. This is the recommended boot mode.

The device can also boot to invoke WIR mode. If the device is booted in this mode, all processors within the device that support a TAP through ICEPick are held in reset until released. Individual processors may be released from reset (local), or all processors held in the reset state may be released at the same time (global).

26.3.3 Dynamic TAP Insertion

To include secondary TAPs in the scan chain, the debugger must use the ICEPick TAP router to program the TAPs. At its root, ICEPick is a scan-path linker that lets the emulator selectively choose which subsystem TAPs are accessible through the device-level debug interface. Each secondary TAP can be dynamically included in or excluded from the scan path. From external JTAG interface point of view, secondary TAPs that are not selected appear not to exist.

Table 26-3 shows the secondary debug and test TAPs connected to the ICEPick scan chain along with the modules that can be accessed. The TAP ID indicates the position of the TAP in the scan chain.

Table 26-3. ICEPick Secondary Debug and Test TAP Mapping

Secondary JTAG Port	TAP ID	IR Scan Length	Modules Accessed Through That JTAG Port
Test Bank			
DFT-SS	0	N/A	For DFT testing (P1500 type)
CATSCAN	1	N/A	For CATSCAN (JTAG type)
Reserved	3	N/A	–
Debug Bank			
Reserved	0	N/A	–
DSP subsystem	1	38	C674x; ICEMaker
HDVICP2-0 iCONT1	2	4	ARM968; ICECrusher-9
HDVICP2-0 iCONT2	3	4	ARM968; ICECrusher-9
HDVICP2-1 iCONT1	4	4	ARM968; ICECrusher-9
HDVICP2-1 iCONT2	5	4	ARM968; ICECrusher-9
HDVICP2-2 iCONT1	6	4	ARM968; ICECrusher-9
HDVICP2-2 iCONT2	7	4	ARM968; ICECrusher-9
Reserved	9	N/A	–
Media Controller (HDVPSS control)	10	4	–
Media Controller (HDVICP control)	11	4	–
Reserved	12	N/A	–
DAP APB-AP	13	4	MPUSS (Cortex-A8; ICECrusher-CS; ETM)
			DebugSS (ETB; DRM; STM; Trace Funnel)
			DSPSS (ADTF)
DAP AHB-AP			HDVICP SMSET (all instances)
			HDVICP HWAs (all instances)
			Statistics collectors (all instances)
			OCP_WP

NOTE:

- The DAP provides a way for the debugger to access:
  - Debug peripherals inside and outside the DebugSS for configuration purposes (through the APB port)
  - Device-level resources for program downloading and access to application peripherals without any requirement to halt or be intrusive to a CPU in the device (through the AHB port)

Besides secondary debug TAPs, ICEPick supports also power, reset, and clock controls for non-JTAG debug cores. The debug cores are accessible through the Debug Access Port (DAP).

Table 26-4 summarizes the ICEPick debug core mapping.

Table 26-4. ICEPick Debug Core Mapping

Debug Core	Debug Core ID	IR Scan Length	Description
Cortex-A8	0	N/A	For supporting SyncRun, debug attention, and clock/power/reset control.
IME3_HDVICP2-0	1	N/A	
ILF3_HDVICP2-0	2	N/A	
IPE3_HDVICP2-0	3	N/A	
ECD3_HDVICP2-0	4	N/A	
MC3_HDVICP2-0	5	N/A	
CALC3_HDVICP2-0	6	N/A	
IME3_HDVICP2-1	7	N/A	
ILF3_HDVICP2-1	8	N/A	
IPE3_HDVICP2-1	9	N/A	
ECD3_HDVICP2-1	10	N/A	
MC3_HDVICP2-1	11	N/A	
CALC3_HDVICP2-1	12	N/A	
IME3_HDVICP2-2	13	N/A	
ILF3_HDVICP2-2	14	N/A	
IPE3_HDVICP2-2	15	N/A	
ECD3_HDVICP2-2	16	N/A	
MC3_HDVICP2-2	17	N/A	
CALC3_HDVICP2-2	18	N/A	

26.4 Basic Debug Support

26.4.1 Processors Native Debug Support

26.4.1.1 Cortex-A8

The Cortex-A8 processor supports the following native debug features:

- Halt mode and monitor mode debugging
- Six hardware breakpoints and two watchpoints
- Performance monitoring
- Trace

For more information about Cortex-A8 native debug features, refer to the ARM® Cortex®-A8 Technical Reference Manual.

Additionally, the native debug features of Cortex-A8 are enhanced by an ICECrusher-CS module as follows:

- Two cross-triggering channels
- Debug reset support

26.4.1.2 C674x

The C674x processor supports the following debug features through the ICEMaker module:

- Run/halt/single-step the target
- Total of ten hardware breakpoints (four from ICEMaker, six from AET)
- Download code
- Access to memory and registers
- Real-time capabilities which include:
  - Access to memory or registers while CPU is running
  - Time-critical interrupts can still be serviced during CPU halt at a debug event
- Trace

26.4.1.3 ARM968

The ARM968E-S processor supports the following native debug features through its EmbeddedICE-RT logic:

- Basic debug functions – run, halt, step, and software breakpoint (SWBP)
- Non-real-time debug – access to register/memory while CPU halt
- Real-time debug (a debug event will trigger a debug interrupt instead of halting CPU)
- Two hardware breakpoints (HWBP) which can used as data watchpoints (WP) as well
- Maskable cycle type and address and data comparison for HWBPs and WPs

For more information about ARM968 native debug features, refer to the ARM968E-S™ Technical Reference Manual.

Additionally, the native debug features of ARM968 are enhanced by an ICECrusher-9 module as follows:

- Two cross-triggering channels
- Bus hang detection and CPU force ready
- Two multi-mode performance/benchmark counters
- JTAG clock synchronization
- Adaptive clocking
- Debug reset support

26.4.1.4 HDVICP Hardware Accelerators

Hardware accelerators offer the following debug capabilities through their embedded SyncBox module:

- Manual halt: Halt occurs at the SYNCBOX task boundary, when requested by the debugger
- Single-step execution at SYNCBOX task level (from 1 to *n* macroblocks depending on the user software)
- Cross-triggering: Hardware accelerator can halt, after the execution of current SYNCBOX task, based on trigger event detection
- Global run

Table 26-5 summarizes the debug capabilities of the hardware accelerators.

Table 26-5. Hardware Accelerators Debug Capability Options

Feature	iME3	iPE3	MC3	CALC3	iLF3	ECD3
Core reset	–	–	–	–	–	–
Execution request	–	–	–	–	–	–
Trigger output	–	–	–	–	–	–
Trigger input	✓	✓	✓	✓	✓	✓
Number of trigger channels	1	1	1	1	1	1
Number of counters	0	0	0	0	0	0
Number of watchpoints	0	0	0	0	0	0
Number of hardware breakpoints	0	0	0	0	0	0

26.4.2 Cross-Triggering

The device supports a cross-triggering feature, which provides a way to propagate debug (trigger) events from one processor subsystem/module to another. For example, *Subsystem A* can be programmed to generate a debug event, which can then be exported as a global trigger across the device. Another *Subsystem B* can be programmed to be sensitive to the trigger line input and to generate an action upon trigger detection.

The device implements two global cross-triggering lines: Trigger0 and Trigger1.

Subsystems cross-triggering is consolidated at the device level by the XTRIGGER module, which is embedded in the debug subsystem.

NOTE: XTRIGGER is not programmatically visible from the JTAG interface or any device processor. Thus, cross-triggering is programmed at the subsystem level.

The Trigger0 and Trigger1 lines can also be configured as external triggers and contribute to cross-triggering.

Table 26-6 summarizes the cross-triggering connections in the device.

Table 26-6. Cross-Triggering Connections in the Device

Module	Channels	Generate/Take Triggers	Comments
Inside DebugSS			
Device to device via pins	Trigger0 / Trigger1	yes / yes	Device to device trigger via EMU[1:0] pins. This is fixed (not affected by configuration)
STM	Trigger0 / Trigger1	no / yes	Trigger generated by STM
Outside DebugSS			
MPUSS	Trigger0 / Trigger1	yes / yes	–
DSPSS	Trigger0 / Trigger1	yes / yes	–
HDVICP2-0 iCONT1	Trigger0 / Trigger1	yes / yes	–
HDVICP2-0 iCONT2	Trigger0 / Trigger1	yes / yes	–
HDVICP2-1 iCONT1	Trigger0 / Trigger1	yes / yes	–
HDVICP2-1 iCONT2	Trigger0 / Trigger1	yes / yes	–
HDVICP2-2 iCONT1	Trigger0 / Trigger1	yes / yes	–
HDVICP2-2 iCONT2	Trigger0 / Trigger1	yes / yes	–
Media Controller (HDVICP control)	Trigger0 / Trigger1	yes / yes	–
Media Controller (HDVPSS control)	Trigger0 / Trigger1	yes / yes	–
OCP_WP	Trigger0 / Trigger1	yes / yes	–
SMSET_HDVICP2-0	Trigger0 / Trigger1	no / yes	Shared with IME3_HDVICP2-0, ILF3_HDVICP2-0, IPE3_HDVICP2-0, ECD3_HDVICP2-0, MC3_HDVICP2-0, CALC3_HDVICP2-0. It can only take triggers.
SMSET_HDVICP2-1	Trigger0 / Trigger1	no / yes	Shared with IME3_HDVICP2-1, ILF3_HDVICP2-1, IPE3_HDVICP2-1 ECD3_HDVICP2-1, MC3_HDVICP2-1, CALC3_HDVICP2-1. It can only take triggers
SMSET_HDVICP2-2	Trigger0 / Trigger1	no / yes	Shared with IME3_HDVICP2-2, ILF3_HDVICP2-2, IPE3_HDVICP2-2 ECD3_HDVICP2-2, MC3_HDVICP2-2, CALC3_HDVICP2-2. It can only take triggers
L3 Debug Port	Trigger0 / Trigger1	no / yes	Entire L3 NoC (including statistics collectors) can only take triggers

26.4.3 Debug Suspend

The device supports a debug suspend feature, which provides a way to stop a "closely coupled" hardware process running on a peripheral-IP when the host processor enters a debug state. The suspend mechanism is important for debug to ensure that peripheral-IPs operate in a lock-step manner with a host controller processor.

An entry is provided for each peripheral-IP that must consider the suspend signals from a number of processors. For each peripheral-IP, sensitivity to the suspend signals is defined within two possibilities (and therefore is coded using 1 bit, usually named EMUFREE or FREEEMU):

- Peripheral-IP is sensitive to the suspend line request.
- Peripheral-IP ignores the suspend line request.

For more information about how to program the sensitivity, see the corresponding peripheral-IP TRM chapter.

Peripherals might be used by multiple host processors. The Debug Resource Manager (DRM) in DebugSS provides a MUX for each peripheral for user to select the suspend source.

Table 26-7 lists the mapping of the device processors to the DRM suspend control input lines.

Table 26-7. Debug Suspend Host Processors Mapping

Suspend Input Line	Host Processor
0	Cortex-A8
1	DSP
2	Media Controller (HDVPSS control)
3	Media Controller (HDVICP control)
4	Reserved
5	HDVICP2-0 iCONT1
6	HDVICP2-0 iCONT2
7	HDVICP2-1 iCONT1
8	HDVICP2-1 iCONT2
9	HDVICP2-2 iCONT1
10	HDVICP2-2 iCONT2
11	Reserved
12	Tied as de-asserted

Table 26-8 lists the mapping of the device peripherals to the DRM suspend control output lines.

Table 26-8. Debug Suspend Peripherals Mapping

Suspend Output Line	Peripheral-IP Module
0	WDTimer0
1	DMTimer1
2	DMTimer2
3	DMTimer3
4	DMTimer4
5	DMTimer5
6	DMTimer6
7	DMTimer7
8	DMTimer8
9	CPGMAC0
10	CPGMAC1
11	USBSS(1)
12	VLYNQ
13	McBSP
14	I2C0
15	I2C1
16-29	Reserved

(1) One common suspension input signal (and one FREEEMU control bit) for the two USB2.0 modules

26.5 Real-Time Debug Events

A few device interrupt channels are dedicated to debug support. Table 26-9 summarizes the debug interrupt events.

Table 26-9. Debug Interrupts

Interrupt Request	Subsystem	Source	Description
EMUINT	Cortex-A8 MPU subsystem	Cortex-A8 ICECrusher-CS	ICECrusher-CS emulation interrupt (EMUINTR)
COMMTX			Cortex-A8 communication transmit channel (COMMTX)
COMMRX			Cortex-A8 communication receive channel (COMMRX)
BENCH			Cortex-A8 PMU (nPMUIRQ)
EMU_DTDMA	DSP subsystem	C674x ICEMaker	ECM interrupt (host scan access, DTDMA)
IC_NINTR1	HDVICP subsystems	ARM968 ICECrusher-9	Combined iCONT1 ICECrusher-9 interrupts (EMUINTR, COMMRX, COMMTX, counters overflow)
IC_NINTR2		ARM968 ICECrusher-9	Combined iCONT2 ICECrusher-9 interrupts (EMUINTR, COMMRX, COMMTX, counters overflow)

NOTE: The debug interrupts are contained within their respective subsystems and do not require additional handling at the SoC level.

26.6 Power, Reset, and Clock Management Debug Support

The global PRCM module implements facilities to support debug across power and clock domain cycles. The debugger can control or get the status of each power and clock domain associated with an ICEPick secondary TAP.

ICEPick provides a set of directives allowing the debugger to:

- Get visibility on the associated power and clock domains state. This includes:
  - Current power setting indicating whether the power domain is on or off
  - Loss of power detected since software last checked the status
  - Current clock setting indicating whether the clock domain is on or off
  - Sleep desired (PM and CM indicate that the debug settings in ICEPick are changing the application state. If it were not for the ICEPick controls, the power or clock would be turned off.)
  - Subsystem reset state
  - Subsystem has entered a debug state that requires the attention of the host debug software.
- Override power/clock control settings to wake up a power or clock domain or to prevent a power or clock domain from going to sleep once it is in ACTIVE state
- Assert/block/extend reset; release from extended reset (WIR)

26.6.1 Power and Clock Management

26.6.1.1 Power and Clock Control Override From Debugger

The debugger can override the application software power and clock management settings through the ICEPick module. It can configure ICEPick to force a domain active or prevent it from going to sleep once it is active. This can be achieved through the FORCEACTIVE and INHIBITSLEEP debugger directives.

26.6.1.1.1 Debugger Directives

26.6.1.1.1.1 FORCEACTIVE Debugger Directive

To ensure that the subsystem debug registers can always be accessed, regardless of the application power-management scenarios, a FORCEACTIVE directive can be issued through the debugger during the entire debug session. From an application standpoint, the system state, status, and timing are preserved, but the subsystem power domain is never shut down. Therefore, the emulation setup is preserved across power transitions, regardless of where the debug hardware is implemented.

If the debugger connects and the subsystems were previously powered down, a FORCEACTIVE directive wakes the system and allows the debugger to take control.

26.6.1.1.1.2 INHIBITSLEEP Debugger Directive

The debugger can use the INHIBITSLEEP directive to keep a subsystem powered and clocked, even if the relevant settings request that this subsystem go to sleep. Unlike the FORCEACTIVE directive, the INHIBITSLEEP command does not wake up a subsystem that is already powered down by the application.

The typical use of the INHIBITSLEEP directive is to prevent power and clock transitions on the subsystem during a debug session. In this situation, when the relevant scenario initiates a transition, the transition does not occur, but from an application standpoint the subsystem is not accessible.

26.6.1.1.2 Intrusive Debug Model

The use of debugger directives is intrusive from the standpoint of the power and clock controls, because they affect the power-management behavior of the application.

26.6.1.2 Debug Across Power Transition

26.6.1.2.1 Nonintrusive Debug Model

To preserve the power-management behavior of the device and allow the subsystem power and clock to be switched off by application software, the subsystem TAP must be disconnected from the ICEPick scan chain. The subsystem is then completely ignored by the debugger and the host-to-target communication is no longer affected by the state of the subsystem power and clocks. The debugger can still be informed that the disconnected subsystem entered the debug state by polling the Debug Attention status bit from ICEPick. The debugger can then insert the TAP, take control of the subsystem power and clock, and examine the system state.

This debug model is nonintrusive, because it can preserve the power-management behavior of the application.

26.6.1.2.2 Debug Context Save and Restore

26.6.1.2.2.1 Debug Context Save

The device partitioning is such that not all the debug components are mapped to an always-on domain. Typically, the programmer wants the debug setup to be preserved along a debug session, including subsystem power cycling. When debug registers are memory-mapped and not implemented within the PD_EMU power domain, the application software must save the state of the debug registers before going to sleep and restore them upon wakeup.

26.6.1.2.2.2 Debug Context Restore

When the application software performs a context restore, it must be able to write to all the debug registers to restore their contents, regardless of the previous ownership. After the subsystem power domain ramps up, the debug resources are in the available state, and ownership is restored. The debug context save and restore sequences are protected. All debug functionality is disabled and debugger accesses are blocked.

26.6.2 Reset Management

The debugger can take control of the system reset for each subsystem through ICEPick. The debugger can configure ICEPick to assert, block, or extend the subsystem reset.

26.6.2.1 Debugger Directives

26.6.2.1.1 Assert Reset

The debugger can program ICEPick to generate a subsystem reset request to the device reset management module. The debugger reset event is then merged with system reset events.

26.6.2.1.2 Block Reset

The debugger can program ICEPick to request the device reset management module to block an unsafe application system or subsystem reset event.

A reset originated by some safe reset sources cannot be blocked by the debugger.

26.6.2.1.3 Wait-In-Reset

Wait-in-reset (WIR) mode is latched from boot mode (see Section 26.3.2) and lets the user hold a secondary TAP module in a reset state when a reset is applied (and thus, extend the reset). This mode lets the user:

- Gain emulation control of any processor in a power domain at POR
- Capture and extend system-generated functional resets while running (under emulation control or not)
- Hold an entire power domain in reset until emulation control of the subsystem can be established
- Stall the entire system while reset is extended to a power domain
- Reset extensions visible external to the power domain
- Debug execution of code from the first cycle of execution
- Prevent processor execution of random instructions in uninitialized program memory at power up
- Download code before any code execution occurs
- Coordinate debug initialization across multiple cores before code execution begins

WIR mode extends only the processor reset. During reset extension, the debugger can still access modules such as L2 memory and MMU, even if they are embedded in the device subsystem affected by WIR.

When the debugger task is complete, the emulator releases the subsystem reset by programming the corresponding ICEPick TAP control register.

26.7 Performance Monitoring

26.7.1 Cortex-A8 MPU Subsystem Performance Monitoring

The Cortex-A8 processor includes a Performance Monitoring Unit (PMU) that enables events, such as cache misses and instructions executed, to be counted over a period of time. The PMU provides four counters to gather statistics about the operation of the processor and memory system. Each counter can count any of the events available in Cortex-A8. Upon counter overflow, PMU can generate an interrupt on its nPMUIRQ output. This interrupt signal is mapped to the CTI TRIGIN[1] input and also routed to interrupt line #3 (BENCH event) of the Cortex-A8 MPU AINTC.

For Cortex-A8 PMU event mapping, refer to the ARM® Cortex®-A8 Technical Reference Manual.

26.8 Trace Support

The device supports trace at the Cortex-A8, C674x DSP, and system levels. Trace is a debug technology that provides a detailed, historical account of application code execution, timing, and data accesses. Trace collects, compresses, and exports debug information for analysis. The debug information can be exported to the ETB, or to the 5-pin (EMU[4:0]) trace interface (system trace only). Trace works in real-time and does not impact the execution of the system.

26.8.1 Processor Trace

The device supports two CPU traces: ARM Cortex-A8 and C674x DSP.

The Cortex-A8 processor trace is only stored to ETB and is not exported off chip. It is supported by ETM module, which generates a real-time trace that can be configured to include: program (instruction) trace, and data trace (address only). Tracing of data values is not supported by Cortex-A8 ETM. The ETM outputs trace to the ETB via its ATB interface.

The DSP processor trace is also only stored to ETB and is not exported off chip. It is supported by ADTF module, which needs to be programmed to convert the DSP trace format to ATB format (for ETB storage).

The CPU traces are routed to the ETB via a CoreSight Trace Funnel (CS_TF). The CS_TF combines multiple trace streams onto a single ATB bus.

The ETB supports on-chip buffering in two modes: circular buffer and one-shot mode.

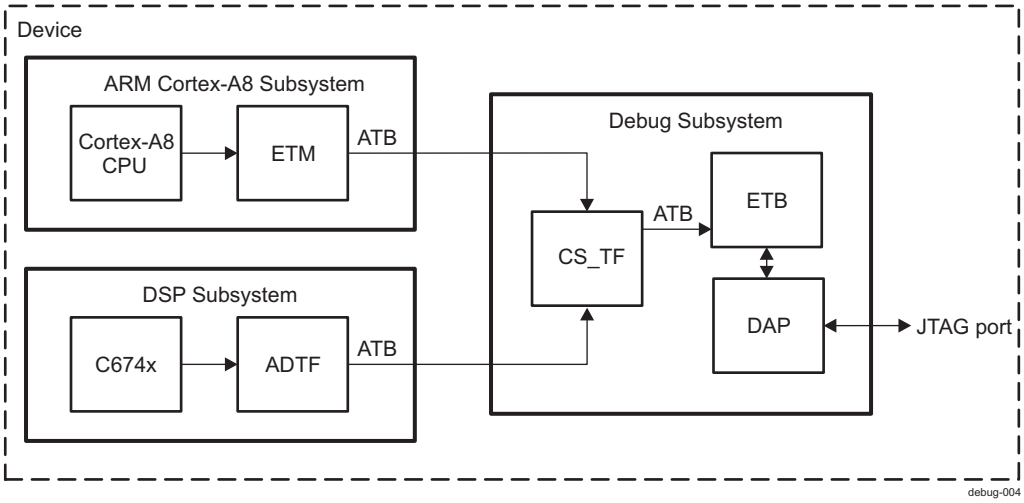
In circular buffer mode, the ETB continuously captures and writes data into memory when trace capture is enabled and its trigger counter is static at 0 (or has not yet decremented to 0). The trace window can be adjusted around a specific trigger spot (trace before, trace after, or trace around) using the ETB trigger counter.

The one-shot mode is not natively supported by the ETB and is actually a TI addition.

For more information about ETB, see the ARM® CoreSight™ Components Technical Reference Manual.

Figure 26-1 shows an overview of the processor traces flow.

Figure 26-1. Processor Traces Flow



26.8.2 System Trace

26.8.2.1 MIPI STM

The STM is a trace module that aids in software debugging. The main features of this module are:

- Implements MIPI STP protocol (rev 1.0) with the following characteristics:
  - Highly optimized for software-generated traces
  - Automatic timestamping of messages
  - Support for 8-, 16-, and 32-bit data types
- Collects the following information:
  - Software messages. See Section 26.9.5.1 for the list of software masters
  - Hardware messages. See Section 26.9.5.2 for the list of hardware masters
- Exports trace data to:
  - External trace receiver
  - On-chip trace buffer
- Available in 1-, 2-, or 4-pin mode with single- or dual-edge clock, depending on the trace bandwidth requirements and characteristics of the trace receiver
- Timestamps:
  - Calculated by the trace receiver if exported off-chip
  - Can use local relative timestamp if exported to the on-chip ETB buffer
- Dedicated 128 × 32-bit FIFO buffer

A maximum of 256 different bus masters can be connected to the STM trace port through a bus arbiter. STP recognizes two distinct modes of tracing (software and hardware types), which use slightly different message combinations to output different types of data. The bus masters can be configured for either type to optimize the system for the different types of trace data.

26.8.2.2 Trace Exported to an External Trace Receiver

System trace data can be exported to an external trace receiver through the STM module. The debugger or application software must take care to program the DRM module according to Table 26-24.

The STM has configurable export width of 1, 2, or 4 data pins (STM_DATA), plus a dedicated export clock (STM_CLK).

26.8.2.3 Trace Captured Into On-Chip Trace Buffer

The user can also configure the STM module to redirect the STP trace stream to the on-chip trace buffer (ETB) and enable local timestamp. This is accomplished by outputting a local timestamp granularity (LTSG) message, which is a TI addition to the MIPI standard messages.

26.9 System Instrumentation

The device supports the following system instrumentation features (that can generate system trace messages):

- Real-time software trace (see Section 26.9.1)
- OCP target traffic monitoring (see Section 26.9.2.1)
- System events (see Section 26.9.2.2)
- L3 target load and master latency monitoring (see Section 26.9.4)
- HWA load monitoring (see Section 26.9.3)

26.9.1 Software Instrumentation

The device provides support for real-time software trace through user-defined (that is, by application code) message writes to specific STM memory mapped register (MMR) locations. Software masters can transmit trace data from the operating system (OS) processes or tasks on 256 different channels, with each channel being defined by the STP protocol. The different channels can be used to group different types of data logically so that it is easy to filter out the data irrelevant to the on-going debugging task. The message structures in STP are optimized to provide an efficient transport for software data through the STM module.

The format of a software message can be defined by specific application. On the receiver side, master ID (associated with individual processor) and channel ID (associated with specific memory mapped address of STM) are both inserted in the exported trace data by STM, and are used to decode the received messages.

Moreover, by carefully assigning channel IDs, the use of semaphore or mutex could be avoided, which would dramatically reduce the intrusiveness to an application.

The software masters supported in the device are:

- Cortex-A8 CPU (via L3)
- DAP (via L3); for testing purpose
- C674x DSP (via L3)
- Media Controller Subsystem (via L3)
- ARM968 of each HDVICP subsystem (via L3, and via SMSET)
- EDMA transfer controller write ports 1 and 2 (TPTC_WR1, and TPTC_WR2)

Each software master has a master-ID assigned to it (see Section 26.9.5.1).

Software messages can be interleaved with hardware messages.

26.9.2 OCP Bus Traffic Monitor (OCP_WP)

26.9.2.1 OCP Target Traffic Monitoring

The device instantiates one OCP watchpoint (OCP_WP) module and five OCP traffic probes inside the L3 NoC for OCP traffic monitoring. The probes are attached to the following L3 targets:

- GPMC
- L4_HS
- L4_LS
- OCMC_RAM0
- OCMC_RAM1

The outputs of the probes are muxed together and then sent to the L3 debug port. The OCP_WP is used to collect data from the OCP traffic probes and then transmit captured data to the STM module. The OCP_WP drives a Probe-ID signal to the L3 interconnect for probe selection. The probe selection is exclusive, meaning that interleaving is not possible.

The OCP_WP provides the following main features:

- Monitoring the OCP traffic originated by all initiators that can access the selected target where the probe is attached
- Filtering OCP monitored bus traffic by:
  - Address range
  - Initiator-ID (also named ConnID). For ConnID values of the various L3 initiators, see the *Bus Interconnect* section of the *Chip Level Resources* chapter.
  - Transaction type
  - Transaction qualifier
- Generating a trigger upon WP match
- Starting and stopping OCP traffic monitoring upon:
  - WP address match
  - External trigger
- Generating hardware message upon system event (see Section 26.9.2.2)
- OCP_WP messages can be interleaved with software messages
- Programming from:
  - Debugger
  - Application

NOTE: The OCP_WP is restricted to monitor request flow only.

The user can program the OCP_WP to extract the traffic from a specific set of initiators (maximum four).

Table 26-10 summarizes the OCP targets that can be monitored by the OCP_WP and their respective probe-ID.

Table 26-10. OCP Traffic Probes Mapping

Probe-ID	L3 OCP Target
000	Reserved(1)
001	GPMC
010	L4_HS
011	L4_FS
100	OCMC_RAM0
101	OCMC_RAM1
110	Reserved
111	Reserved

(1) No selection. This is also for power saving.

26.9.2.2 Messages Triggered from System Events

The OCP_WP can be programmed to export a hardware message through the STM upon detection of a system event (interrupt, DMA request). A bus of 16 system events pre-selected at SoC level from a total of 256 system events is routed to the OCP_WP. An event selector controlled by the device Control Module is used to select the system events to be traced.

To simplify the design of the event selector and maintain the flexibility of event selection, the 256 events are grouped into four groups of 64. For each group, a 64-bits configuration register (HW_EVT_SEL_GRPx, where x = 1 to 4) is used to select 4 events out of 64. For the description of the HW_EVT_SEL_GRPx registers, refer to the *Control Module* section of the *Chip Level Resources* chapter.

26.9.3 HDVICP Instrumentation

The device takes advantage of the system trace infrastructure to provide visibility to the user regarding HDVICP micro-task sequencing. This is supported through a SMSET module instantiated in each of the three HDVICP subsystems. The micro-task boundaries are handled as generic events and encapsulated in STP messages with an event-ID and local timestamp and exported through the MIPI-STM module.

The HDVICP instrumentation scheme allows the user to understand micro-task dependencies, hardware accelerators load balancing, and potential bottlenecks. DMA transfer boundaries are reported as HDVICP events. Software messages from ARM968 execution can be interleaved with HDVICP events.

The HDVICP subsystem provides an instrumentation master port that interfaces directly with the debug subsystem. This ensures that the HDVICP instrumentation is not intrusive and does not affect the L3 application traffic.

26.9.4 L3 Load and Latency Monitors

The L3 interconnect supports a built-in performance monitoring feature by implementing a number of performance probes, as well as a statistics collector (SC) component, which computes traffic statistics within a user-defined window and periodically reports to the user through the MIPI_STM interface. Four SC instances are instantiated in the device:

- One statistics collector dedicated to DDR2/3 SDRAM load monitoring – SC_SDRAM (see Section 26.9.4.1). The probes connected to this SC are of OCP type.
- Three statistics collectors dedicated to master latency monitoring – SC_LAT0, SC_LAT1, and SC_LAT2 (see Section 26.9.4.2). The probes connected to this SC are of NTTP type.

Statistics collectors (SDRAM and LAT0/1/2) can report:

- Average burst length in bytes/packet per sampling window
- Average throughput in bytes/cycle
- % Link occupancy on the request link (for store transactions) during a sampling window
- % Link occupancy on the response link (for load transactions) during a sampling window
- % Arbitration conflict cycles on the request link
- % Initiator busy cycles on the response link
- Histogram of payload length in bytes (for example, 0–16 , 16–32, 32–128) each sampling window.
- Histogram of quality of service (QoS) metric for HDVICP initiator (for example, low priority, high priority) each sampling window.

The performance metrics are interleaved with software instrumentation data at the L3 interconnect level.

The performance monitoring probes implement three main functions:

- Events detection
- Transactions filtering
- Aggregation

The probes can be configured to detect the events summarized in Table 26-11.

Table 26-11. Performance Monitoring Events Detection

Link Event	NTTP	OCP	Definition
NONE	✓	✓	No event selected
ANY	✓	✓	Any clock cycles
TRANSFER	✓	✓	Word has been accepted by the receiver.
WAIT	✓	–	Transfer has been initiated but the transmitter currently has no data to send.
BUSY	✓	✓	Receiver applies flow control
PKT	✓	✓	Transfer of a new packet header
DATA	✓	✓	Transfer of a payload word
IDLES	✓	✓	No communication over the link

Table 26-11. Performance Monitoring Events Detection (continued)

Link Event	NTTP	OCP	Definition
LATENCY	✓	–	Debug bit detection

The probes can be configured to filter the traffic based on the criteria summarized in Table 26-12.

Table 26-12. Performance Filtering Options

Probe Filtering Options	Description
Master address	Mask and match
Slave address[1]	
UserInfo	
Read	
Write	
Error	
OCP address[2]	

(1) SC_LAT0/1/2 only
(2) SC_SDRAM only

For master address mapping (all statistics collectors) and slave address mapping (SC_LAT0/1/2 only), see the *Bus Interconnect* section of the *Chip Level Resources* chapter.

The probes implement a user-defined set of counters that aggregate the events sampled by the detector and filtered according to the user setup.

NOTE: Statistics collectors counter values are not accessible by application software.

Table 26-13 summarizes the performance probe aggregation modes.

Table 26-13. Aggregation Modes

Aggregation Mode	Description
FILTER_HIT	The counter increments by 1 when the filter hits.
MIN_MAX_HIT	The counter increments by 1 when the filter hits and the selected event information is within range. - Payload length (bytes) - Pressure value - Request/response latency (clock cycles)
EVT_INFO	The selected event information is added to the counter value when the filter hits. - Payload length (bytes) - Pressure value - Request/response latency (clock cycles)
AND_FILTER	The counter increments by 1 when all unit filters hit.
OR_FILTER	The counter increments by 1 when at least one unit filter hits.
SUM_REQ_EVT	The counter sums the events from any request port.
SUM_RSP_EVT	The counter sums the events from any response port.
SUM_ALL_EVT	The counter sums the events from any port.
EXT_EVT	The counter increments by 1 when selected external event input signal is sampled high.

26.9.4.1 L3 Target Load Monitoring

The L3 interconnect implements two performance monitoring probes on DDR2/3 memory channels. The traffic statistics are computed within a user-defined window and periodically reported to the user through the STM interface.

SC_SDRAM supports the following main features:

- Two probe inputs
  - Probe 0 – EMIF1 (DMM-EMIF1 128-bit port)
  - Probe 1 – EMIF2 (DMM-EMIF2 128-bit port)
- Two 32-bit counters/filters shared concurrently
  - Counter 0 with two elements
  - Counter 1 with two elements
- Filtering according to:
  - Initiator of traffic
  - Access priorities
- No latency counter. Only bandwidth measurement on this collector
- 32-bit collecting window counter
- Dump Identifier is 0x0 (tie-off value)
- Dumps frames at L3 interconnect slave address 0x1F (DEBUGSS)

Table 26-14 summarizes the SC_SDRAM configuration.

Table 26-14. SC_SDRAM Configuration

Counters	Min/ Max	Filter Elements	L3 Target	
Counter 0	Yes	2	EMIF1	EMIF2
Counter 1	Yes	2	EMIF1	EMIF2

Table 26-15 shows the SC_SDRAM port mapping.

Table 26-15. SC_SDRAM Port Mapping

Probe #	Description	Link	Port #
0	EMIF1	OCP REQ	0
		OCP RSP	1
1	EMIF2	OCP REQ	2
		OCP RSP	3

26.9.4.2 L3 Master Latency Monitoring

The device L3 interconnect implements a number of performance monitoring probes on the main L3 initiators:

- MPU subsystem
- DSP subsystem (MDMA port)
- HDVICP2-0
- HDVICP2-1
- HDVICP2-2
- MMU

- HDVPSS M0
- HDVPSS M1
- EDMA TPTC1 read channel (TPTC_RD1)
- EDMA TPTC2 read channel (TPTC_RD2)
- EDMA TPTC3 read channel (TPTC_RD3)
- EDMA TPTC4 read channel (TPTC_RD4)
- SGX
- Media Controller
- PCIe
- CPGMAC0
- CPGMAC1
- SATA

The master latency statistics are computed within a user-defined window and periodically reported to the user through the MIPI-STM interface.

The probes can be configured to filter latencies in four classes and report to the user a latency distribution along execution.

Because the performance metrics and the software events are exported through a unified export channel, it is possible to correlate latency trends with on-going execution and system context.

Because computing latency requires maintaining the state between request and response ports, the probe cannot compute latency statistics on 100 percent of the initiator traffic. Hence, latency histograms must be extracted on large execution windows to be accurate.

26.9.4.2.1 SC_LAT0 Configuration

SC_LAT0 supports the following main features:

- Eight probe inputs:
  - Probe 0: MPU subsystem
  - Probe 1: DSP MDMA
  - Probe 2: HDVPSS M0
  - Probe 3: HDVPSS M1
  - Probe 4: MMU
  - Probe 5: HDVICP2-0
  - Probe 6: HDVICP2-1
  - Probe 7: HDVICP2-2
- Eight 32-bit counters shared concurrently:
  - Counter 0 with one filter
  - Counter 1 with one filter
  - Counter 2 with one filter
  - Counter 3 with one filter
  - Counter 4 with one filter
  - Counter 5 with one filter
  - Counter 6 with one filter
  - Counter 7 with one filter
- Filtering according to:
  - L3 target
  - Access priorities
- 12-bit counter for latency measurement

- 32-bit collecting window counter
- Identifier is 0x1 (tie-off value)
- Dumps frames at slave address 0x1F (DEBUGSS)

Table 26-16 summarizes the SC_LAT0 configuration.

Table 26-16. SC_LAT0 Configuration

Counters	Min/ Max	Filter Elements	L3 Master							
Counter 0	Yes	1	MPU	DSP	HDVPSS M0	HDVPSS M1	MMU	HDVICP2-0	HDVICP2-1	HDVICP2-2
Counter 1	Yes	1	MPU	DSP	HDVPSS M0	HDVPSS M1	MMU	HDVICP2-0	HDVICP2-1	HDVICP2-2
Counter 2	Yes	1	MPU	DSP	HDVPSS M0	HDVPSS M1	MMU	HDVICP2-0	HDVICP2-1	HDVICP2-2
Counter 3	Yes	1	MPU	DSP	HDVPSS M0	HDVPSS M1	MMU	HDVICP2-0	HDVICP2-1	HDVICP2-2
Counter 4	Yes	1	MPU	DSP	HDVPSS M0	HDVPSS M1	MMU	HDVICP2-0	HDVICP2-1	HDVICP2-2
Counter 5	Yes	1	MPU	DSP	HDVPSS M0	HDVPSS M1	MMU	HDVICP2-0	HDVICP2-1	HDVICP2-2
Counter 6	Yes	1	MPU	DSP	HDVPSS M0	HDVPSS M1	MMU	HDVICP2-0	HDVICP2-1	HDVICP2-2
Counter 7	Yes	1	MPU	DSP	HDVPSS M0	HDVPSS M1	MMU	HDVICP2-0	HDVICP2-1	HDVICP2-2

Table 26-17 shows the SC_LAT0 port mapping.

Table 26-17. SC_LAT0 Port Mapping

Probe #	Description	Link	Port #
0	MPU	NTTP REQ	0
		NTTP RSP	1
1	DSP	NTTP REQ	2
		NTTP RSP	3
2	HDVPSS M0	NTTP REQ	4
		NTTP RSP	5
3	HDVPSS M1	NTTP REQ	6
		NTTP RSP	7
4	MMU	NTTP REQ	8
		NTTP RSP	9
5	HDVICP2-0	NTTP REQ	10
		NTTP RSP	11
6	HDVICP2-1	NTTP REQ	12
		NTTP RSP	13
7	HDVICP2-2	NTTP REQ	14
		NTTP RSP	15

26.9.4.2.2 SC_LAT1 Configuration

SC_LAT1 supports the following main features:

- Eight probe inputs:
  - Probe 0: TPTC_RD1
  - Probe 1: TPTC_RD2
  - Probe 2: TPTC_RD3
  - Probe 3: TPTC_RD4
  - Probe 4: HDVICP2-0 SL2
  - Probe 5: HDVICP2-1 SL2
  - Probe 6: HDVICP2-2 SL2
  - Probe 7: C674x DSP SDMA
- Eight 32-bit counters shared concurrently:
  - Counter 0 with one filter
  - Counter 1 with one filter
  - Counter 2 with one filter
  - Counter 3 with one filter
  - Counter 4 with one filter
  - Counter 5 with one filter
  - Counter 6 with one filter
  - Counter 7 with one filter
- Filtering according to:
  - L3 target
  - Access priorities
- 12-bit counter for latency measurement
- 32-bit collecting window counter
- Identifier is 0x2 (tie-off value)
- Dumps frames at slave address 0x1F (DEBUGSS)

Table 26-18 summarizes the SC_LAT1 configuration.

Table 26-18. SC_LAT1 Configuration

Counters	Min/ Max	Filter Elements	L3 Master							
Counter 0	Yes	1	TPTC_ RD1	TPTC_ RD2	TPTC_ RD3	TPTC_ RD4	HDVICP2-0 SL2	HDVICP2-1 SL2	HDVICP2-2 SL2	C674x DSP SDMA
Counter 1	Yes	1	TPTC_ RD1	TPTC_ RD2	TPTC_ RD3	TPTC_ RD4	HDVICP2-0 SL2	HDVICP2-1 SL2	HDVICP2-2 SL2	C674x DSP SDMA
Counter 2	Yes	1	TPTC_ RD1	TPTC_ RD2	TPTC_ RD3	TPTC_ RD4	HDVICP2-0 SL2	HDVICP2-1 SL2	HDVICP2-2 SL2	C674x DSP SDMA
Counter 3	Yes	1	TPTC_ RD1	TPTC_ RD2	TPTC_ RD3	TPTC_ RD4	HDVICP2-0 SL2	HDVICP2-1 SL2	HDVICP2-2 SL2	C674x DSP SDMA
Counter 4	Yes	1	TPTC_ RD1	TPTC_ RD2	TPTC_ RD3	TPTC_ RD4	HDVICP2-0 SL2	HDVICP2-1 SL2	HDVICP2-2 SL2	C674x DSP SDMA
Counter 5	Yes	1	TPTC_ RD1	TPTC_ RD2	TPTC_ RD3	TPTC_ RD4	HDVICP2-0 SL2	HDVICP2-1 SL2	HDVICP2-2 SL2	C674x DSP SDMA

Table 26-18. SC_LAT1 Configuration (continued)

Counters	Min/ Max	Filter Elements	L3 Master							
Counter 6	Yes	1	TPTC_ RD1	TPTC_ RD2	TPTC_ RD3	TPTC_ RD4	HDVICP2-0 SL2	HDVICP2-1 SL2	HDVICP2-2 SL2	C674x DSP SDMA
Counter 7	Yes	1	TPTC_ RD1	TPTC_ RD2	TPTC_ RD3	TPTC_ RD4	HDVICP2-0 SL2	HDVICP2-1 SL2	HDVICP2-2 SL2	C674x DSP SDMA

Table 26-19 shows the SC_LAT1 port mapping.

Table 26-19. SC_LAT1 Port Mapping

Probe #	Description	Link	Port #
0	TPTC_RD1	NTTP REQ	0
		NTTP RSP	1
1	TPTC_RD2	NTTP REQ	2
		NTTP RSP	3
2	TPTC_RD3	NTTP REQ	4
		NTTP RSP	5
3	TPTC_RD4	NTTP REQ	6
		NTTP RSP	7
4	HDVICP2-0 SL2	NTTP RSP	8
5	HDVICP2-1 SL2	NTTP RSP	9
6	HDVICP2-2 SL2	NTTP RSP	10
7	C674x DSP SDMA	NTTP RSP	11

26.9.4.2.3 SC_LAT2 Configuration

SC_LAT2 supports the following main features:

- Seven probe inputs:
  - Probe 0: SGX530
  - Probe 1: Media Controller
  - Probe 2: Reserved
  - Probe 3: Reserved
  - Probe 4: CPGMAC0
  - Probe 5: CPGMAC1
  - Probe 6: PCIe
- Four 32-bit counters shared concurrently:
  - Counter 0 with one filter
  - Counter 1 with one filter
  - Counter 2 with one filter
  - Counter 3 with one filter
- Filtering according to:
  - L3 target
  - Access priorities
- 12-bit counter for latency measurement
- 32-bit collecting window counter
- Identifier is 0x3 (tie-off value)
- Dumps frames at slave address 0x1F (DEBUGSS)

Table 26-18 summarizes the SC_LAT2 configuration.

Table 26-20. SC_LAT2 Configuration

Counters	Min/ Max	Filter Elements	L3 Master						
Counter 0	Yes	1	SGX530	Media Controller	Reserved	Reserved	CPGMAC0	CPGMAC1	PCIe
Counter 1	Yes	1	SGX530	Media Controller	Reserved	Reserved	CPGMAC0	CPGMAC1	PCIe
Counter 2	Yes	1	SGX530	Media Controller	Reserved	Reserved	CPGMAC0	CPGMAC1	PCIe
Counter 3	Yes	1	SGX530	Media Controller	Reserved	Reserved	CPGMAC0	CPGMAC1	PCIe
Counter 4	Yes	1	SGX530	Media Controller	Reserved	Reserved	CPGMAC0	CPGMAC1	PCIe
Counter 5	Yes	1	SGX530	Media Controller	Reserved	Reserved	CPGMAC0	CPGMAC1	PCIe
Counter 6	Yes	1	SGX530	Media Controller	Reserved	Reserved	CPGMAC0	CPGMAC1	PCIe

Table 26-19 shows the SC_LAT2 port mapping.

Table 26-21. SC_LAT1 Port Mapping

Probe #	Description	Link	Port #
0	SGX530	NTTP REQ	0
		NTTP RSP	1
1	Media Controller	NTTP REQ	2
		NTTP RSP	3
2	Reserved	NTTP REQ	4
		NTTP RSP	5
3	Reserved	NTTP REQ	6
		NTTP RSP	7
4	CPGMAC0	NTTP REQ	8
		NTTP RSP	9
5	CPGMAC1	NTTP REQ	10
		NTTP RSP	11
6	PCIe	NTTP REQ	12
		NTTP RSP	13

26.9.5 Master-ID Encoding

A master-ID (MReqMstID[7:0]) field is used by the STM to encode a MASTER type of message:

- MReqMstID[7] – Differentiates software versus hardware masters (0 for software masters; 1 for hardware masters)
- MReqMstID[6:2] – Master address exported by the L3 interconnect debug subsystem target
- MReqMstID[1:0] – Additional qualifier for multicore masters

26.9.5.1 Software Masters

The STM module allows:

- Enabling a maximum of four SoC software masters
- Masking MReqMstID[1:0]
- Differentiating multicore software masters through MReqMstID[1:0]

Table 26-22 summarizes the software initiators that can export trace messages through the STM.

Table 26-22. STM Message Software Masters

Initiator	MReqMstID			Description
	[7]	[6:2]	[1:0]	
Cortex-A8 MPU subsystem	0	00000	00	SW messages routed through L3 DebugSS target port
DebugSS (DAP)	0	00100	–	STP link testing; SW messages routed through L3 DebugSS target port
DSP subsystem	0	01000	–	SW messages routed through L3 DebugSS target port
HDVICP2-0 iCONT1	0	00010	00	SW messages routed through L3 DebugSS target port
HDVICP2-0 iCONT2 (via SMSET)	0	00011	00	SW messages routed through HDVICP2-0 instrumentation port
HDVICP2-1 iCONT1	0	00110	00	SW messages routed through L3 DebugSS target port
HDVICP2-1 iCONT2 (via SMSET)	0	00111	00	SW messages routed through HDVICP2-1 instrumentation port
HDVICP2-2 iCONT1	0	01100	00	SW messages routed through L3 DebugSS target port
HDVICP2-2 iCONT2 (via SMSET)	0	01101	00	SW messages routed through HDVICP2-2 instrumentation port
Reserved	0	10000	00	Reserved
Media Controller (HDVPSS and HDVICP control)	0	10001	00	HDVPSS control; SW messages routed through L3 DebugSS target port
	0	10001	01	HDVICP control; SW messages routed through L3 DebugSS target port
Reserved	0	10100	00	Reserved
TPTC_WR1	0	11001	00	Data logging; SW messages routed through L3 DebugSS target port
TPTC_WR2	0	11011	–	Data logging; SW messages routed through L3 DebugSS target port

26.9.5.2 Hardware Masters

The STM module allows enabling a subset of SoC hardware masters (maximum = 4).

Table 26-23 summarizes the hardware initiators that can export trace messages through the STM.

Table 26-23. STM Message Hardware Masters

Initiator	MReqMstID		
	[7]	[6:2]	[1:0]
OCP_WP Traffic Probe	1	00001	00
OCP_WP System Events	1	00010	00
Statistics collector 0	1	11100	00
Statistics collector 1	1	11101	00
Statistics collector 2	1	11110	00
Statistics collector 3	1	11111	00
HDVICP2-0 (via SMSET)	1	01001	00
HDVICP2-1 (via SMSET)	1	01010	00
HDVICP2-2 (via SMSET)	1	01011	00

26.10 Concurrent Debug Modes

The debugger or application software can program the DRM to route a specific debug function to each debug interface pin.

Because of the limited number of pins allocated to debug and trace, a muxing of signals is implemented (at DRM level).

Table 26-24 summarizes the trace port configuration.

Table 26-24. Trace Port Configuration

Device Pin Name	Triggers	STM (System Trace)
EMU4	–	STM_DATA[3]
EMU3	–	STM_DATA[2]
EMU2	–	STM_CLK
EMU1	Trigger1	STM_DATA[1]
EMU0	Trigger0	STM_DATA[0]

Table 26-25 summarizes the concurrent debug and trace in the device.

Table 26-25. Concurrent Debug and Trace

Debug Use Case	Concurrent Debug Flows	Debug Pins	Trace Pins		
		Triggers	Data	Control	Clock
0	STM		4	–	1
	Triggers	–			
1	STM		2	–	1
	Triggers	2			

26.11 Debug Components Memory Mapping

Table 26-26 summarizes the base addresses for the debug components.

Table 26-26. Debug Modules Memory Mapping

Region Name	Block Name	Start Address	End Address	Size	Description
L3 Target Space	Instrumentation	0x4B00 0000	0x4BFF FFFF	16MB	Debug Instrumentation Interconnect region
	OCP_WP	0x4818 C000	0x4818 CFFF	4KB	OCP_WP register space

26.11.1 Debug Instrumentation Interconnect Address Space

The instrumentation address space is driven by the L3 or HW master access ports and targets the STM module. The address space is detailed in Table 26-27.

Table 26-27. Debug Instrumentation Interconnect Address Space

Region Name	Module Name	Start Address Offset(1)	End Address Offset(1)	Size	Description
Debug Instrumentation Interconnect	MIPI_STM [256 × 4K channels] (OCP – AddrSpace 0)	0x000000	0x0FFFFF	1MB	STM [256 × 4K channels]
	MIPI_STM [256 × 1K channels] (OCP – AddrSpace 1)	0x100000	0x13FFFF	256KB	STM [256 × 1K channels]
	Debug Configuration Interconnect	0x140000	0x17FFFF	256KB	Debug Configuration Interconnect
	Reserved	0x181000	0x1FFFFF	508KB	If accessed, returns error response

(1) Offset from the base address of the region

26.11.2 Debug Configuration Interconnect Address Space

The following modules may be accessible in the configuration address space of an application initiator to the configuration bus or from the DAP APB port:

Table 26-28. Debug Configuration Interconnect Address Space

Region Name	Module Name / Interface Type	Start Address Offset(1)	End Address Offset(1)	Size	Description
Debug Configuration Interconnect	External port (OCP) for Cortex-A8 MPU Subsystem	0x00000	0x1FFFF	128KB	MPU subsystem register space
	DRM (OCP)	0x20000	0x20FFF	4KB	DRM register space
	MIPI_STM (OCP)	0x21000	0x21FFF	4KB	MIPI_STM register space
	ETB (APBv3)	0x22000	0x22FFF	4KB	ETB register space
	Reserved	0x23000	0x23FFF	4KB	Reserved
	CS_TF (APBv3)	0x24000	0x24FFF	4KB	CS_TF register space
	Reserved	0x25000	0x25FFF	4KB	Reserved
	External port (OCP) for ADTF	0x26000	0x26FFF	4KB	ADTF register space
	Technology Specific Interconnect Registers	0x27000	0x27FFF	4KB	Note: Not all space may be used
	Reserved	0x28000	0x3FFFF	96K	If accessed, returns error response

(1) Offset from the base address of the region

26.11.3 Cortex-A8 MPU Subsystem Address Space

The base address for the debug components at the MPUSS (Cortex-A8) are described in Table 26-29.

Table 26-29. Cortex-A8 MPU Subsystem Address Space

Region Name	Module Name	Start Address Offset(1)	End Address Offset(1)	Size	Description
DAP-APB	Cortrex-A8 ETM unit	0x0000	0x0FFF	4KB	Cortrex-A8 ETM register space
	Cortex-A8 Debug unit	0x1000	0x1FFF	4KB	Cortex-A8 Debug unit register space
	Cortex-A8 CTI unit	0x2000	0x2FFF	4KB	Cortex-A8 CTI register space
	ICECrusher-CS APB	0x3000	0x3FFF	4KB	ICECrusher-CS register space
	Timeout Register for the CortexA8 APB Port	0x4000	0x43FF	1KB	–
	AP registers	0x4400	0x47FF	1KB	Reserved
	LA registers	0x4800	0x4BFF	1KB	Reserved. Writing to this range may cause unpredictable behavior in the bridge
	Reserved	0x4C00	0x4FFF	1KB	If accessed, returns error response
	Timeout Register for the ICECrusher-CS APB Port	0x5000	0x53FF	1KB	–
	Reserved	0x5400	0x7FFF	8KB	If accessed, returns error response

(1) Offset from the base address of the region

The MPUSS debug components are accessible via the external subsystem port (for Cortex-A8) of the configuration interconnect in DebugSS.

The physical address of a certain MPU debug module (for example, Cortex-A8 debug unit) can be calculated as follows:

- The SoC level address for DebugSS is 0x4B000000 (see Table 26-26).
- The offset of the configuration interconnect in DebugSS is 0x140000 (see Table 26-27).
- The offset of the external subsystem port (for Cortex-A8) from the configuration interconnect is 0x0 (see Table 26-28).
- Therefore, the SoC level address for Cortex-A8 debug unit is 0x4B141000 (0x4B000000 + 0x140000 + 0x1000).

26.11.4 DAP-APB Address Space

The DAP-APB port has a 32-bit address space.

The address map as seen by the APB interface of the CoreSight DAP is described in Table 26-30.

Any accesses made to the "no mapping" region return implementation defined data. In fact, the current implementation allows the address of such accesses to wrap around a 256KB address boundary.

MSB of the address allows the access on the APB interface to be qualified as an application (when not set) or debugger one (when set).

Table 26-30. DAP-APB Address Map

Region Name	Module Name	Start Address (hex)	End Address (hex)	Size	Description
DAP-APB	Debug Configuration Interconnect	0x00000000	0x0003FFFF	256KB	Debug Configuration Interconnect [application access]
	DAP ROM Table	0x00040000	0x00040FFF	4KB	DAP ROM Table [application access]
	No Mapping	0x00041000	0x7FFFFFFF	2MB	–
	Debug Configuration Interconnect	0x80000000	0x8003FFFF	256KB	Debug Configuration Interconnect [debugger access]
	DAP ROM Table	0x80040000	0x80040FFF	4KB	DAP ROM Table [debugger access]
	No Mapping	0x80041000	0x7FFFFFFF	2MB	–

Revision History

Changes from March 26, 2013 to March 6, 2015 (from B Revision (March 2013) to C Revision) **Page**

- **Chapter 1: Chip Level Resources** 109
- Figure 1-1: Removed all frequencies 111
- Table 1-1: Deleted frequency from Comments column 115
- Table 1-2: Deleted Maximum Frequency column 115
- Section 1.2.6.2: Deleted first bulleted item 118
- Section 1.7.2.1.1: Deleted last sentence in fourth paragraph 178
- Figure 1-67: Changed 432 MHz to Audio reference clock 205
- Table 1-73: Deleted Frequency column 206
- Table 1-73: Changed DDRPLL to MAINPLL for DDR2/DDR3 SYSCLK4 206
- Section 1.10.3: Changed $_{fvco}$ to f_{vco} in f_s frequency 210
- Section 1.10.3: Changed fractional range to FF FFFFh in FREQ 210
- Section 1.10.3: Added NOTE 210
- Section 1.10.3.1.1: Changed third paragraph 211
- Figure 1-70: Deleted 432 MHz 211
- Table 1-76: Changed table 212
- Table 1-77: Changed table 213
- Table 1-78: Added Main PLL Frequencies for Speed Grade 2 table. Subsequent tables renumbered 213
- Table 1-79: Added Main PLL Frequencies for Speed Grade 4 table. Subsequent tables renumbered 214
- Table 1-80: Added paragraph before table 214
- Table 1-80: Added SYSCLK5 Frequency with Integer FREQ vs Fractional FREQ table. Subsequent tables renumbered 214
- Section 1.10.3.1.2: Changed frequency for DMM to 364 MHz in first paragraph 217
- Table 1-82: Changed table 218
- Table 1-83: Changed table 218
- Table 1-84: Added DDR PLL Frequencies for Speed Grade 4 table. Subsequent tables renumbered 219
- Table 1-85: Added DDR PLL Frequencies for All Speed Grades table. Subsequent tables renumbered 219
- Table 1-87: Changed table 223
- Table 1-88: Changed table 223
- Figure 1-73: Changed 432 MHz to Audio reference clock 226
- Table 1-90: Changed table 227
- Table 1-91: Changed table 227
- Figure 1-75: Deleted all frequencies 232
- Section 1.16.1.2.20: Changed 380 MHz to 364 MHz. Deleted the last sentence 324
- Section 1.16.1.2.34: Changed 432 MHz to the audio reference clock 332
- Section 1.16.1.3: Deleted USB Control Register 1 (USB_CTRL1) subsection. Subsequent subsections renumbered 338
- Table 1-212: Changed USB_CTRL0 to USB_CTRL. 338
- Table 1-212: Changed address offset 628h to Reserved. Deleted USB Control Register 1 (USB_CTRL1) 338
- Figure 1-170: Changed figure 346
- Table 1-220: Changed table 346
- Figure 1-171: Changed figure 347
- Table 1-221: Changed table 347
- Figure 1-172: Changed figure 348
- Table 1-222: Changed table 348
- **Chapter 4: DMM/TILER** 415
- Figure 4-1: Deleted frequencies 416
- Figure 4-1: Connected DMM to EMIF and DDR Bank 416
- **Chapter 9: General-Purpose Memory Controller (GPMC)** 859
- Section 9.2.4.9.6: Changed first bullet in second paragraph (RDCYCLETIME - CLKACTIVATIONTIME) 885
- Table 9-80: Changed Description of ECCTOPSECTOR bit 996

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265