TMS320C6474 DSP
Frame Synchronization

User's Guide

TEXAS INSTRUMENTS

Literature Number: SPRUG13
October 2008

List of Figures

List of Tables

Read This First

About This Manual

This document describes the frame synchronization module of the TMS320C6474 device.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- 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 below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

Related Documentation From Texas Instruments

The following documents describe the TMS320C6474 DSP. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

SPRU189 — *TMS320C6000 DSP CPU and Instruction Set Reference Guide.* Describes the CPU architecture, pipeline, instruction set, and interrupts for the TMS320C6000 digital signal processors (DSPs).

SPRU198 — *TMS320C6000 Programmer's Guide.* Describes ways to optimize C and assembly code for the TMS320C6000™ DSPs and includes application program examples.

SPRU301 — *TMS320C6000 Code Composer Studio Tutorial.* Introduces the Code Composer Studio™ integrated development environment and software tools.

SPRU321 — *Code Composer Studio Application Programming Interface Reference Guide.* Describes the Code Composer Studio™ application programming interface (API), which allows you to program custom plug-ins for Code Composer.

SPRU871 — *TMS320C64x+ Megamodule Reference Guide.* Describes the TMS320C64x+ digital signal processor (DSP) megamodule. Included is a discussion on the internal direct memory access (IDMA) controller, the interrupt controller, the power-down controller, memory protection, bandwidth management, and the memory and cache.

TMS320C6000, Code Composer Studio are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

TEXAS INSTRUMENTS

User's Guide
SPRUG13–October 2008

C6474 Frame Synchronization

1 Introduction

This section describes the general operation of the frame synchronization (FSYNC) module and how it is synchronized with other modules. Any of the three GEM cores on the C6474 device can program the FSYNC module. The software layers are divided into three categories:

- Application Layer
- FSYNC Driver Layer
- FSYNC Register Layer.

1.1 Purpose of the Peripheral

The frame synchronization module handles all UMTS (and other standard-specific) timing and time alignment for the C6474 device. Timing is provided to all other subsystems via system events and VBUS reads.

1.2 Features

The use of the frame synchronization module is optional for WIMAX, IEE Std 802.16e and depends on a user's specific implementation. The flexible architecture of the FSYNC module may allow for support of other current and future standards. The following features are supported by the frame synchronization module:

- OBSAI RP1 interface
- OBSAI RP1 compliant for RP3 and WCDMA FDD
- OBSAI RP1 compliant time of day
- 30 programmable events based on RP3 or system timer
- Event offset and periodicity is programmable
- WCDMA FDD supported through OBSAI RP1 interface or alternate TRT and UMTS_SYNC interfaces
- TDSCDMA, CDMA2000 supported with alternate TRT and UMTS_SYNC
- Error and alarm events
- Error and alarm condition register
- VBUS CPU interface
- Supports CPRI-type antenna interface.

1.3 Terms and Abbreviations

3GPP — Third-Generation Partnership Project

AIF — Antenna Interface

AP — Applications Processor

ASIC — Application Specific Integrated Circuit

ATPG — Automatic Test Pattern Generation

BBM — Base Band Module (channel card)

BFN — Base station Frame Number

CBA — Common Bus Architecture (SCR, VBUSP, VBUSM)

CCM — Clock and Control Module

CDMA2000 — Code Division Multiple Access 2000

CPRI — Common Public Radio Interface

CRC — Cyclic Redundancy Check

DDR — Dual Data Rate memory

DFT — Design For Test

DL — Downlink

DMA — Direct Memory Access

DSP — Digital Signal Processor

FDD — Frequency Division Duplex

FF — Flip Flop

GEM — A specific DSP core used throughout the device

GPS — Global Positioning System

GSM — Global System for Mobile Communication

HW — Hardware

I/F — Interface

L2 RAM — DSP Core’s Bulk Static Memory

MPW — Minimum Pulse Width

OBSAI — Open Base Station Architecture Initiative

PBIST — Processor Built In Self Test

PLL — Phase Lock Loop

RAM — Random Access Memory

RF — Radio Frequency

RP1 — Reference Point 1 (OBSAI)

RP3 — Reference Point 3 (OBSAI)

RTL — Register Transfer Language

RX — Receive

SCLK — System Clock (FSYNC_CLOCK_N and FSYNC_CLOCK_P)

SCR — Switch Central Resource

SERDES — SERializer / DESerializer

SW — Software

Sync_Burst — Frame Synchronization Burst

TBD — To Be Determined

TC — Terminal Count

TDD — Time Division Duplex

TOD — Time Of Day

TPDMA — Third-Party DMA

TRT — Time Reference Tick

TX — Transmit

UL — Uplink

UMTS — Universal Mobile Telecommunication System

VBUS — Virtual Bus

VBUSM — Virtual Bus Multi-issue

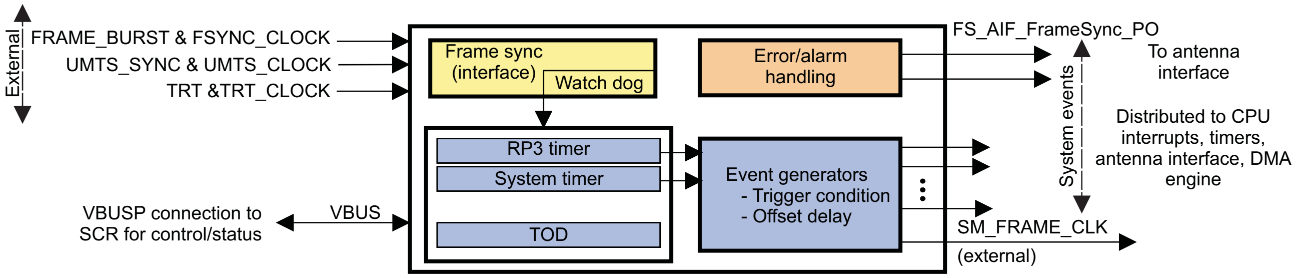
VBUSP — Virtual Bus Pipeline

WCDMA — Wideband Code Division Multiple Access

1.4 Functional Block Diagram

The FSYNC module is intended to generate controllable timed events that are synchronized with system timing. These timed events are then distributed throughout the C6474 device to synchronize C6474 system tasks. The system's clock control module sends synchronization and clock inputs to the FSYNC. The functional block diagram is shown in Figure 1.

Figure 1. Top-Level Frame Synchronization Block Diagram



1.4.1 FSYNC Interface

The FSYNC interface is meant to mark the boundaries of UMTS frames and/or system time so that events can be generated that are synchronized with this time. Frame alignment is exported from the TMS320C6474 device via the SM_FRAME_CLK signal.

There are three sets of synchronization inputs to the FSYNC module. The first set is the FRAME_BURST and FSYNC_CLOCK pair. This pair represents the OBSAI RP1-compliant interface. The second set is the UMTS_SYNC and UMTS_CLOCK pair. This pair is used as an alternative to the RP1 interface for synchronization to UMTS time. The antenna interface depends on using this pair for non-OBSAI synchronization mode. A high true UMTS_SYNC pulse captured on the rising edge of UMTS_CLK marks the beginning of a frame. The third set is the TRT and TRT_CLOCK pair. The TRT synchronization interface is needed because the antenna interface driven by the RP3 timer always needs to run at a derivative of a UMTS clock rate, hence UMTS_SYNC and UMTS_CLOCK. There may be non-UMTS standards that run at a different clock/sync rates, hence TRT and TRT_CLOCK. This pair is also used as an alternative to the RP1 interface for synchronization to system time. This TRT interface has the flexibility to support UMTS and non-UMTS standards. A high true TRT pulse captured on the rising edge of TRT_CLOCK marks the system time boundary.

The watchdog timer is used to detect a failure of the RP1 interface to update synchronization within a programmable time.

1.4.2 FSYNC Timer

There are three main timers. The RP3 timer keeps track of UMTS time that is required to create events for the antenna interface. The time of day (TOD) is the GPS time and is compliant with OBSAI RP1. The system timer is used as the WCDMA FDD time when using the RP1 FRAME_BURST and FSYNC_CLOCK and has flexible capabilities for all supported standards described in Section 1.5, *Industry Standards Compliance*.

1.4.3 Event Generators

System and RP3 timers are separately maintained for system and RP3 frame alignment. Event generators issue events based on these timers. Each event generator is software programmable to select between system timer or RP3 timer values from which to base its event generation. Event generators are used primarily for generation of system events (see Section 2.2.5, *Event Generations*). These system events are routed to all DSP cores, TPDMA controllers, and general-purpose timers as a timing trigger. Usually, event generator trigger conditions are programmed to give periodic strobes which are positioned with a programmed delay. The resulting strobes are driven out on the system event lines.

1.4.4 Error and Alarm Handling

There are two system event wires that are dedicated from the FSYNC module for error and alarm conditions. These system event wires are broadcast to all DSP cores and major C6474 subsystems. For more details, see Section 2.2.5.6, *Error and Alarm Generations*.

1.5 Industry Standards Compliance

The supported standards of the FSYNC module are:

- WCDMA FDD
- CDMA2000
- TDSCDMA
- WIMAX,IEEE Std.802.16e.

2 Peripheral Architecture

This section describes the details of the peripheral architecture and how it is structured. The information describes the architecture and design details that are common to all of the operation modes and how the peripheral works. The FSYNC architecture supports two synchronization modes, namely:

- Non-RP1 frame synchronization
- RP1 frame synchronization.

This section provides detailed steps to be followed when programming the FSYNC for non-RP1 and RP1 modes.

2.1 Memory Map

Table 1. Frame Synchronization (FSYNC) Registers

Hex Address	Acronym	Register Name
0280 0000	PID	Peripheral Identification Register
0280 00A0	ERR_INT_MASK_1	FSYNC Error Interrupt Mask 1 Register
0280 00A4	ERR_INT_SET	FSYNC Error Interrupt Set Register
0280 00A8	ERR_INT_CLEAR	FSYNC Error Interrupt Clear Register
0280 00AC	ERR_END_OF_INT	FSYNC Error End-of-Initialization Register
0280 00B0	SCRATCH	FSYNC Scratch Register
0280 00B4	CTL1	FSYNC Control Register 1
0280 00B8	CTL2	FSYNC Control Register 2
0280 00BC	EMUCTL	FSYNC Emulation Control Register
0280 00C0	EMUMASK	FSYNC Emulation Mask Register
0280 00C4	RP1TS	FSYNC RP1 Type Select Register
0280 00C8	UPDATE	FSYNC Update Register
0280 00CC	RP3INIT	FSYNC RP3 Initialization Register
0280 00D0	SYSINIT	FSYNC System Initialization Register
0280 0080	ERR_INT_SRC_RAW	FSYNC Error Interrupt SRC RAW Register
0280 0084	ERR_MASK_STAT_0	FSYNC Error Mask Status 0 Register
0280 0088	ERR_MASK_STAT_1	FSYNC Error Mask Status 1 Register
0280 008C	ERR_SET_MASK_0	FSYNC Error Set Mask 0 Register
0280 0090	ERR_SET_MASK_1	FSYNC Error Set Mask 1 Register
0280 0094	ERR_CLEAR_MASK_0	FSYNC Error Clear Mask 0 Register
0280 0098	ERR_CLEAR_MASK_1	FSYNC Error Clear Mask 1 Register
0280 009C	ERR_INT_MASK_0	FSYNC Error Interrupt Mask 0 Register
0280 0100	RP3TC	FSYNC RP3 Terminal Count Entry
0280 0128	TOD1	FSYNC Time-of-Day Capture Register 1
0280 012C	FSYN C_TOD2	FSYNC Time-of-Day Capture Register 2
0280 0130	RP31	FSYNC RP3 Capture Register 1
0280 0134	RP32	FSYNC RP3 Capture Register 2
0280 0138	SYS1	FSYNC System Capture Register 1
0280 013C	SYS2	FSYNC System Capture Register 2
0280 0140	SYSTC_RAM	FSYNC System Terminal Count Entry
0280 0168	SYSTC	FSYNC System Terminal Count Register
0280 016C	RP3TC	FSYNC RP3 System Terminal Count Register
0280 0170	TYPE	FSYNC Type Capture Register
0280 0174	TODVAL	FSYNC Time-of-Day Value Register
0280 0178	RP3VAL	FSYNC RP3 Value Register
0280 017C	SYSVAL	FSYNC System Value Register

Table 1. Frame Synchronization (FSYNC) Registers (continued)

Hex Address	Acronym	Register Name
0280 0200	EGM_CTRL	FSYNC Mask Event Generator Control Register
0280 0258	EGC_CTRL	FSYNC Counter Event Generator Control Register
0280 0280	EGM_MASK	FSYNC Mask Event Generator Mask
0280 0300	EGM_OFFSET	FSYNC Mask Event Generator Offset Value
0280 0358	EGM_CTCOUNT	FSYNC Counter Event Generator Control Register
0280 0380	EVT_FORCE	FSYNC Event Force Register

2.2 Signal Descriptions

This section describes each of the signals on the external interface and their functions. The input and output signals are active high, unless otherwise noted. All outputs will be in a 0 state during rst_n = 0, unless otherwise noted.

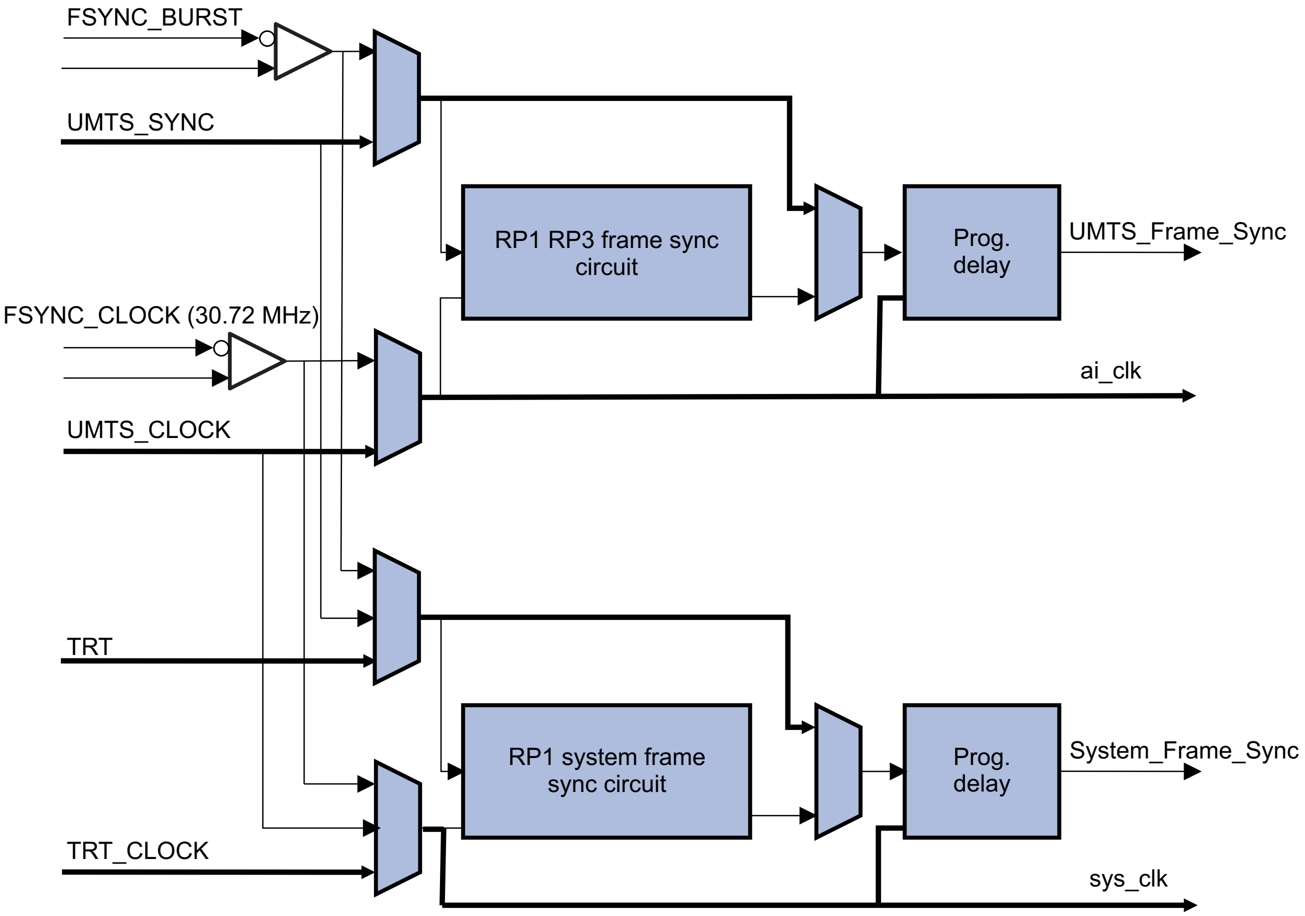
Table 2. External I/O

Signal	Direction	Description
FSYNC_CLK	IN	30.72-MHz input Note: also known as SCLK.
FRAME_BURST	IN	Frame Synchronization Burst. Used to serially clock in synchronization burst packet and to identify the frame boundary. OBSAI RP1 compliant. Note: also known as OBSAI RP1 Sync_Burst.
UMTS_CLK	IN	UMTS Alternate Clock. User-defined frequency, 122.88 MHz max frequency.
UMTS_SYNC	IN	UMTS synchronization. Positive value clocked in with UMTS_Clock indicates sync boundary. One UMTS_CLK duration.
TRT_CLK	IN	Multi-standard Clock. User-defined frequency, 122.88 MHz max frequency
TRT	IN	Time Reference Tick. Positive value clocked in with rising edge of TRT_CLK indicates sync boundary. One TRT_CLK duration.
SM_FRAME_CLK	OUT	Source of external frame clock. External frame clock is used to frame synchronize external devices to the TMS320C6474 device. (This signal is similar to (FSYNC_SYSEVNT0).

2.2.1 Non-RP1 Frame Synchronization

For interfaces other than OBSAI RP1, the C6474 device is required to synchronize to the external UMTS frame or standard-specific alignment. Optionally, the user may choose to use either differential signals (FSYNC_BURST, FSYNC_CLOCK) or single-ended input signals (UMTS_SYNC, UMTS_CLOCK) for the RP3 UMTS timer and the system timer will select between UMTS_SYNC, UMTS_CLOCK and TRT, TRT_CLOCK for the non-OBSAI supported standard. Figure 2 shows the input data path for non-RP1 frame synchronization.

Figure 2. Non-RP1 Frame Synchronization Input Path



The frame synchronization mode is determined by the SYNC_MODE bit of the CTL1 register. At reset, the FSYNC module is in non-RP1 mode. Table 3 shows the programming of CTL1 register for non-RP1 mode settings. The clock and sync signals of the RP3 timer are selected by programming the CLK_RP3_TIMER and SYNC_RP3_TIMER bits of the CTL1 register. The clock and sync signals of the system timer are selected by programming the CLK_SYSTEM_TIMER and SYNC_SYSTEM_TIMER bits of the CTL1 register.

Table 3. FSYNC Control Register 1 (CTL1) Settings for Non-RP1

Bit	Field	Value	Description
23	SYNC_MODE	0	Non-RP1. Disable RP1 synchronization

The chip, slot, and sub-chip terminal count values are programmed in the SYSTC and RP3TC registers for system and RP3 counters, respectively. The TC registers contains bit fields to program slot, sub-chip TC value, and address pointer to circular buffer which can store a maximum of 10 chip TC values. The DSP loads the RP3 and system timer initialization values in the SYSINIT and RP3INIT registers, respectively. For CDMA2000, the PCG count and chip count are initialized to zero. The sub-chip TC value can be programmed from 0 to 31. For TDSCDMA, the slot TC value is initialized to zero. All the chip TC values can be programmed to different values in order to change the slot size at regular intervals of time. The sub-chip TC value can be programmed from 0 to 31. The timer value can be read from the RP3VAL and SYSVAL registers, respectively. The timer starts counting once the sync pulse is received and the FSYNC_CTL2_ARM_TIMER bit in the CTL2 register is enabled. The timer can be halted by programming the FSYNC_CTL2_TIMER_HALT bit in the CTL2 register and the timer value can be read from the RP3VAL and SYSVAL registers.

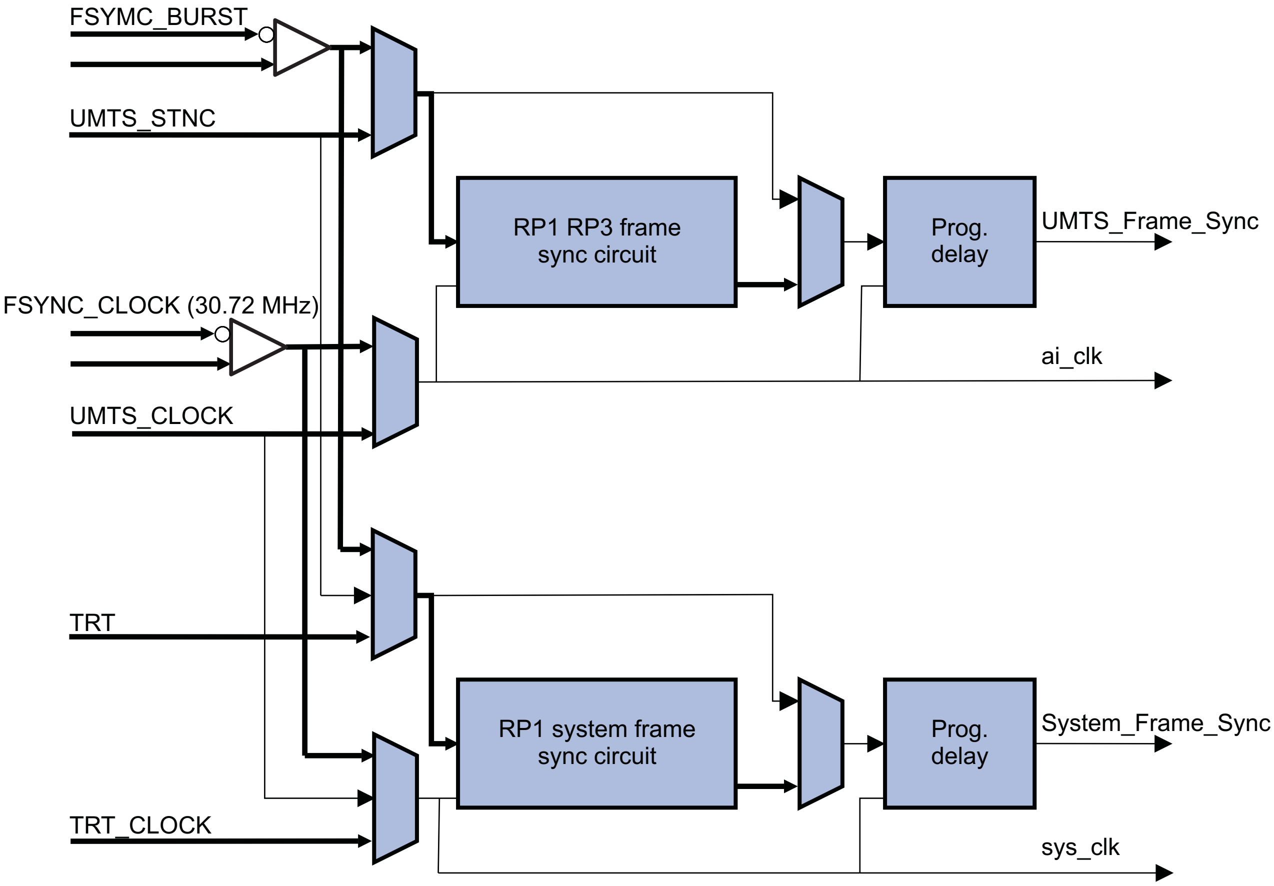
2.2.1.1 Non-RP1 Mode Programming Sequence

```
CSL_FsyncSetup myFsyncCfg;
CSL_FsyncHandle hFsync;
CSL_BitMask16 ctrlArg;
CSL_FsyncTimerCountObj pSysTimerVal, pRp3TimerVal;

/* csl fsync initialization */
 CSL_fsyncInit (NULL);

/* Open fsync handle for */
 hFsync = CSL_fsyncOpen(&myFsyncObj, CSL_FSYNC, NULL, &status);

/* Read the setup values from user defined parameters */
 myFsyncCfg = fsyncSetupValues (fsyncSetupParam[0]);

/*Program the frame sync setup registers namely CTL1 and Init Registers */
 CSL_fsyncHwSetup (hFsync, &myFsyncCfg);

/*Program the frame sync CTL2 register*/
 CSL_fsyncEnableTimer(hFsync);
 CSL_fsyncSetOkStatBit(hFsync);

/* Program Test Bench Register to send UMTS Sync to FSYNC Module */
/*Wait for some time so that UMTS sync is received by FSYNC */
/* Read the type capture register */
 CSL_fsyncGetRp1TypeCapture(hFsync,&fsyncRp1TypeCaptureVal);
/* Read the RP3 and SYSVAL Registers to check whether counter has incremented
 after the arrival of UMTS sync*/
 CSL_fsyncGetSysTimerValue( hFsync, pSysTimerVal);
```

2.2.2 RP1 Frame Synchronization

The OBSAI RP1 interface specifies that the clock and control module (CCM) provides a 30.72-MHz system clock and periodically sends a synchronization bursts to the baseband modules. The synchronization burst is received by TMS320C6474 devices in the baseband module where UMTS timing information is extracted. The RP3 and system frame numbers and time-of-day information is passed in the synchronization bursts. Figure 3 shows the paths selected when using RP1 interface.

Figure 3. RP1 Frame Synchronization Input Path



The OBSAI synchronization burst is serially transmitted over a single differential input which is clocked via the differential system clock (SCLK). Each bit of the serial transfer is held for 8 SCLK periods. For fields with more than a single bit, the least-significant bit (LSB) is sent first. The first field is the *start* bit, which marks the beginning of the synchronization burst. The 8-bit type field follows and identifies the type information, which is contained in the synchronization pay load. The 64-bit information/payload field contains the relevant data (either frame number or time of day). The CRC field is used for data integrity and the end field terminates the synchronization burst packet. Figure 4 and Table 4 shows the synchronization burst format and type field, respectively.

Figure 4. Synchronization Burst Format

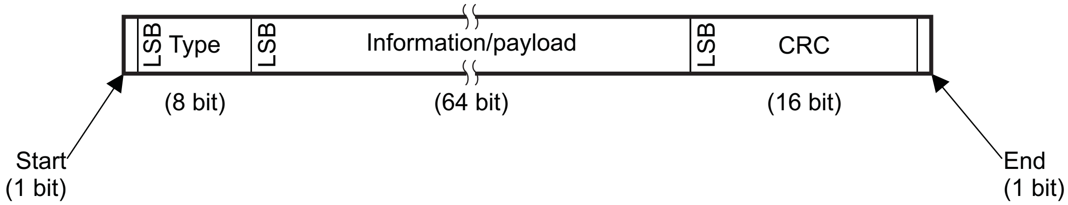


Table 4. Type Field Definition

Type	Supported	Value
Not Used	N/A	00h
RP3 Bus (FDD) Frame Number	Yes	01h
WCDMA/FDD Frame Number	Yes	02h
GSM/Edge1 Frame Number	No	03h
GSM/Edge2 Frame Number	No	04h
GSM/Edge3 Frame Number	No	05h
WCDMA/TDD Frame Number	No	06h
CDMA2000 Frame Number	Yes	07h
Time of Day	Yes	08h
Reserved	N/A	09h – 7Fh
Spare	No	80h – FFh

For the RP1 frame synchronization the SYNC_MODE bit of the CTL1 register should be set to 0x1. The SYNC_RP3_TIMER and SYNC_SYSTEM_TIMER bits of the CTL1 register should be in default value 0x0. The CLK_RP3_TIMER and CLK_System_TIMER bits of the CTL1 register should be 0x2. Table 5 shows the settings of the CTL1 register for OBSAI RP1.

Table 5. FSYNC Control Register 1 (CTL1) Settings for OBSAI RP1

Bit	Field	Value	Description
23	SYNC_MODE	1	OBSAI RP1 synchronization.

The RP1 type field which is used to synchronize the timer is programmed in the RP1TS register. The value to be programmed is shown in Table 6

Table 6. FSYNC RP1 Type Select (RP1TS) Register Settings

Bit	Field	Value	Description
7-0	SYSTEM_TIMER_RP1_SYNC	02h	WCDMA/FDD
		07h	CDMA2000

When RP3, system, or time of day is loaded from the RP1 interface, the loaded values are captured in the LSBs and most-significant bits (MSBs) of the RP31/RP32, SYS1/SYS2, and TOD1/FSYNC_TOD2 registers, respectively. The TYPE register is loaded with the type information from the bursts. The DSP can read these capture registers via the VBUS. The user should specify the expected update rate in the UPDATE register. This register contains bit fields for the TOD frame update rate, RP3 frame update rate, and system frame update rate. The values programmed in this register are used in watchdog counters.

The C6474 device derives the beginning of an RP1 UMTS frame as the end of the synchronization burst. For the sake of flexibility, a programmable offset is used in conjunction with the end of the CRC field when triggering the start of the UMTS frame. The start of frame can be offset from 0 to 15 SCLKs. The RP3_SYNC_DELAY, SYSTEM_SYNC_DELAY, and TOD_SYNC_DELAY bits of the CTL1 register are used to program the offset/delay value for RP3, system, and TOD frame synchronization, respectively.

The RP3 and system timers are maintained separately with identical and full precision. When using the RP1 interface, both system and RP3 timers are set and maintained via an RP1 synchronization burst. Chips are counted in 1/8th-chip increments. This is due to the SCLK running at 8x chip rate. The sub-chip TC must be programmed to count 8 sub-chips (sub-chip TC = 7). The 12-bit frame number is extracted from the SYNC_BURST and written to the frame portion of the RP3 timer. The slot and chip portions are set to zero. At the end of this SYNC_BURST, the RP3 timer starts counting. In WCDMA/FDD, the system timer acts the same as the RP3 timer. Chips are counted in 1/8th-chip increments, 2560 chips per slot, 15 slots per frame. When using the RP1 interface, the frame number is extracted from SYNC_BURST and written into the frame portion, the rest of the counter is set to zero. There is a special feature that allows the RP3 timer to be identically synchronized to the system timer. When using this feature, it is illegal to separately send an additional SYNC_BURST to set up the RP3 timer and would result in erroneous behavior of the RP3 timer. For this feature, the RP3_EQUAL_SYS_TIMER bit of the CTL1 register should be set to 0x1.

2.2.2.1 RP1 Mode Programming Sequence

```
CSL_FsyncSetup myFsyncCfg;
CSL_FsyncHandle hFsync;
CSL_BitMask16 ctrlArg;
CSL_FsyncTimerCountObj pSysTimerVal, pRp3TimerVal;
CSL_FsyncRp1PayloadObj fsyncWcdmaFddCaptureVal;

/* csl fsync initialization */
 CSL_fsyncInit (NULL);

/* Open fsync handle for */
 hFsync = CSL_fsyncOpen(&myFsyncObj, CSL_FSYNC, NULL, &status);

/* Read the setup values from user defined parameters */
 myFsyncCfg = fsyncSetupValues (fsyncSetupParam[0]);

    /*Program the frame sync setup registers namely CTL1 and Init Registers */
 CSL_fsyncHwSetup (hFsync, &myFsyncCfg);

/*Program the frame sync CTL2 register*/
 CSL_fsyncEnableTimer(hFsync);
 CSL_fsyncSetOkStatBit(hFsync);

/* Program Test Bench Register to send WCDMA/FDD type frame burst to FSYNC Module */

/*Wait for some time so that burst data will get updated in Capture Registers */

/* Read the type capture register */
 CSL_fsyncGetRp1TypeCapture(hFsync,&fsyncRp1TypeCaptureVal);

/* Read the WCDMA Type capture register */
 CSL_fsyncGetCapWcdmaFddType(hFsync,&fsyncWcdmaFddCaptureVal);

/* Read the RP3 and SYSVAL Registers to check whether counter has incremented
 after the arrival of UMTS sync*/
 CSL_fsyncGetSysTimerValue( hFsync, pSysTimerVal);
 CSL_fsyncGetRp3TimerValue( hFsync, pRp3TimerVal);
```

2.2.3 Frame Resynchronization

Resynchronization takes place if the RESYNC_MODE bit of the CTL1 register is set. A sync error signal is issued if the sync is out of alignment with the current state of the timer. This error condition causes the system and RP3 timers to issue SYS_RE_SYNC and RP3_RE_SYNC, respectively. The SYS_RE_SYNC and RP3RE_SYNC restart the event offset counter for each event that is connected to the particular timer that has had a resynchronization. This effectively disables the events associated with that timer until the timer reaches the frame boundary. In RP1 mode, the timer will reload the OBSAI frame from payload, 0's for slot, 0's for chip, and 0's for sub-chip. For non-RP1 mode, the RP3INIT and SYSINIT registers are used to load the initial values. From that point, the event is disabled until its offset value has been met. During the resynchronization process, the timer that has not been resynchronized continues to be used for generating events with no break in event generation.

Table 7. FSYNC Control Register 1 (CTL1) Settings for Resynchronization

Bit	Field	Value	Description
24	RESYNC_MODE	1	Automatically resync when new sync is out of alignment.

2.2.4 5.3.4 RP1 CRC Check

The CRC process does a modulo2-polynomial divide of the input burst. The resulting remainder of that division is compared with the sent CRC. The initial value of CRC is 0xFFFF or 0x0. The CRC initial value can be programmed in the CRC_INIT_VALUE bit of the CTL1 register. The default value of the CRC calculator is 0x0. Normally, the MSB of the CRC data is sent first in the frame burst. If the CRC data is flipped, i.e., the LSB is sent first in the frame burst, program the CRC_POSITION bit of the CTL1 register to 1. The CRC data may be inverted in the frame burst such that the CRC_INVERT bit of the CTL1 register must be programmed 0x1. Before programming the CRC_INIT_VALUE, CRC_POSITION, and CRC_INVERT bits, the CRC_USAGE bit of the CTL1 register must be programmed to control whether to discard the frame burst when the CRC check fails.

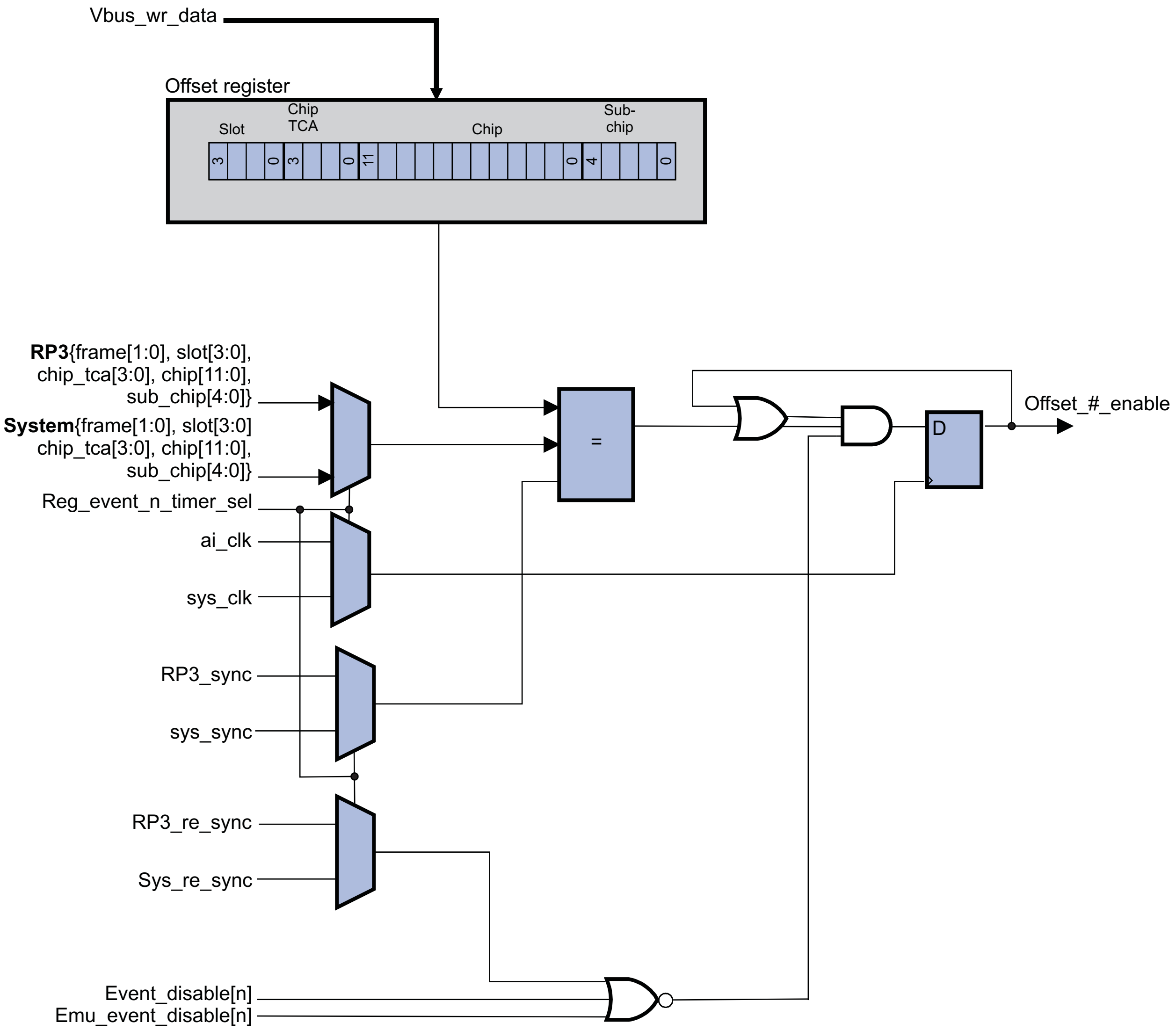
2.2.5 Event Generations

All internal wireless rate system events are generated via the thirty event generators. These custom-built circuits have programmable trigger conditions and programmable offsets that allow the user to create wireless rate system events for C6474 timing control.

2.2.5.1 Trigger Offset

For clean timing control, the offset condition gates system events from being generated. This offset control is achieved with a simple compare for each event. The compare values specify a count value after which system events generation can begin. Once the compare values match the current timer value, the offset condition is met and normal event generation commences. Each event generator has its own programmable offset control register. The offset value can be up to a full frame. The event generator offset diagram is shown in Figure 5

Figure 5. Event Generator Offset Condition



Under reset or resynchronization conditions, the offset condition must be re-evaluated prior to enabling system events. Different TMS320C6474 subsystems must start up in a precise order. The different offset conditions of each event generator control this start-up order. The events are inhibited until the offset conditions are met. The offset compare value can be programmed for slot, chip TC state, chip, and sub-chip count values for each event. These values can be programmed in the EGM_CTRL register.

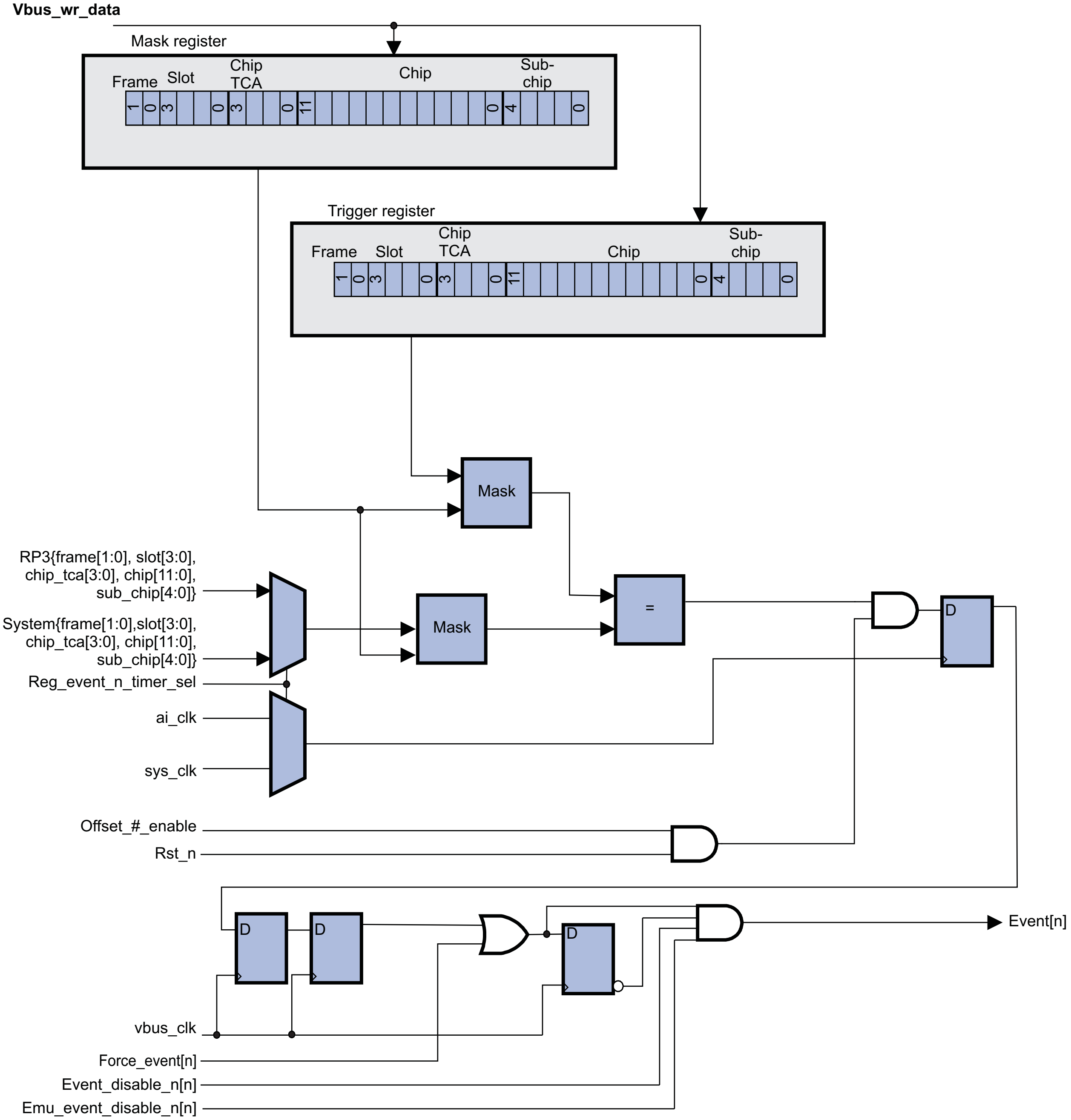
2.2.5.2 Triggering Events

There are two mechanisms for triggering and generating events. The first method is a highly-programmable triggering mechanism giving the ability to generate UMTS trigger conditions of any 2n values of the sub-chip, chip, chip TC address, or frame by using masks and a trigger compare value. The value in the selected system or RP3 timer is compared with the trigger value in order to generate an event. The second set of events is generated using counters to generate an event. The system events FSYNC_SYSEVNT0 to FSYNC_SYSEVENT9 and FSYNC_SYSEVNT18 to FSYNC_SYSEVNT29 are masked-based events. The system events FSYNC_SYSEVENT10 to FSYNC_SYSEVENT17 are counter-based events.

2.2.5.3 Masked Event Generator

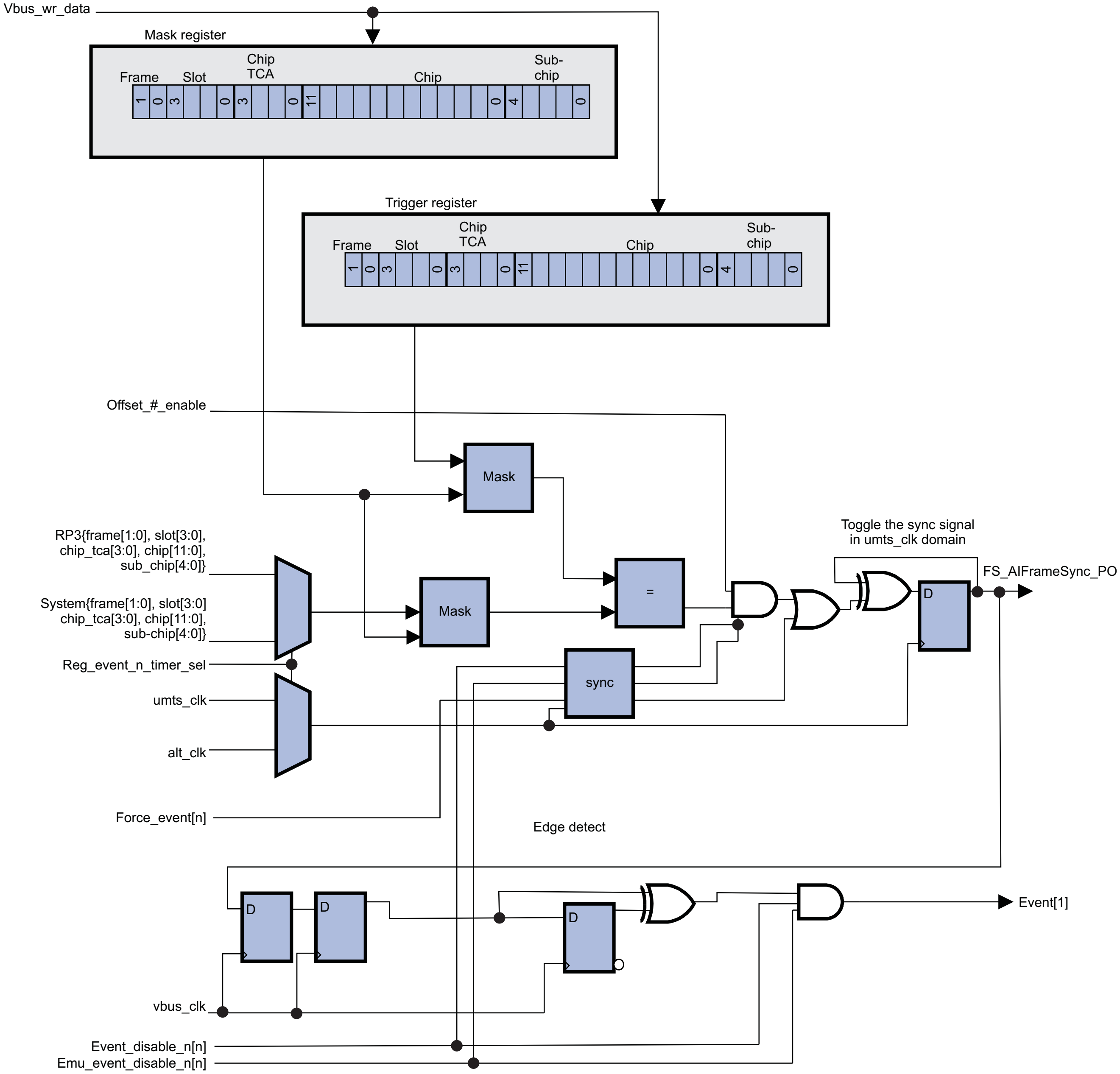
Masked events give the ability to generate UMTS trigger conditions of any 2n values of the sub-chip, chip, chip TC address, or frame. Programmed trigger conditions are compared to either the system or RP3 UMTS timers. The trigger control consists of a trigger value register (EGM_COMPARE) and a trigger mask register (EGM_MASK). The EGM_MASK register enables which bits should be compared. After the offset conditions have been met, a system event is generated when the enabled bits in the EGM_COMPARE register equal the enabled bits, including circular buffer state bits (chip terminal count address), in the selected timer. The resulting trigger is periodic. Placement of these events is on a sub-chip granularity. These events can be disabled by masking all bits with a write of 0 to the EGM_MASK register. System events FSYNC_SYSEVNT0 through FSYNC_SYSEVNT9 and FSYNC_SYSEVNT18 through FSYNC_SYSEVNT29 are mask-/trigger-based. The minimum period for mask-based events is 2 clocks (selected UMTS clocks, *raw* event period), with the exception of FSYNC_AIFRAMESYNC, SM_FRAME_CLK, FSYNC_SYSEVNT1 and FSYNC_SYSEVNT0. The exceptions have a minimum period of 8 clocks — minimum pulse width of 4 cycles high and 4 cycles low. If all of the EGM_MASK register values are 0, then no event is generated, regardless of the trigger value. If all EGM_MASK register values are 1, then all bits of the timer that are in the mask range are compared with the trigger value. If that trigger value is all 1s, then an event occurs every 4th frame or, in other words, whenever the sub-chip, chip, chip TC address, slot, and frame (bits 0 and 1) are all 1s. Figure 6 shows the flow of mask-based event generation.

Figure 6. Mask Based Event Flow



There are two special events, FSYNC_AIFRAMESYNC and SM_FRAME_CLK. The FSYNC_AIFRAMESYNC event is generated directly from the anntenna interface (AI) clock domain so as to avoid latency, and sent directly to the AI. It has an associated VBUS synchronized event, FSYNC_SYSEVNT1, that may be used as a CPU event. The FSYNC_AIFRAMESYNC indicates an event when it pulses high for a minimum of 4 clock cycles and is low for a minimum of 4 clock cycles. The second special event, SM_FRAME_CLK, behaves in the same manner as FSYNC_AIFRAMESYNC, except it is the locally-clocked (selected UMTS domain clock) version of FSYNC_SYSEVNT0. Figure 7 shows the flow of the two special events.

Figure 7. SM_FRAME_CLK and FSYNC_AIFFRAMESYNC Event Generation Flow



The sample code in Section 2.2.5.3.1 shows the programming sequence for mask-based event generation.

2.2.5.3.1 Mask-Based Event Generation Programming Sequence

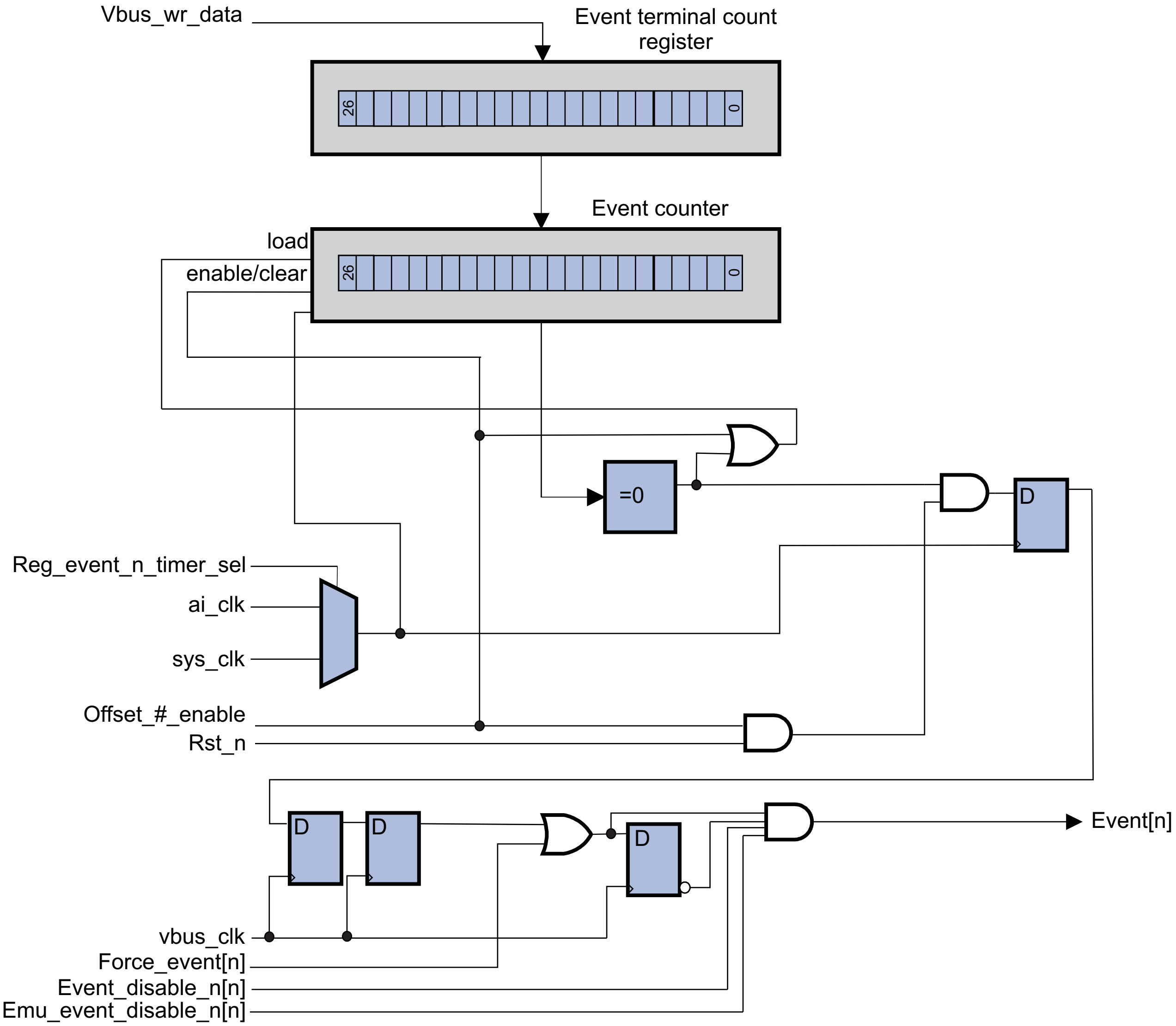
```
CSL_IntcHandle hIntcFsync;
CSL_FsyncSetup myFsyncCfg;
CSL_FsyncHandle hFsync;
CSL_FsyncMaskTriggerGenObj myFsyncMaskEventCfg;
CSL_BitMask16 ctrlArg;

/*Interrupt Setup for Fsync Event Generation */

/* fsync Event FSEVT2 is hooked to int04 */
 hIntcFsync = fsyncIntrInit (CSL_INTC_VECTID_4,CSL_INTC_EVENTID_17);

/* Hook the ISRs for event2 */
 CSL_intcHookIsr (CSL_INTC_VECTID_4,&fsync_event2_ISR);

/* Clear the Event */
 CSL_intcHwControl (hIntcFsync, CSL_INTC_CMD_EVTCLEAR, NULL);

/* Enable the Event */
 CSL_intcHwControl (hIntcFsync, CSL_INTC_CMD_EVTENABLE, NULL);

/* Fsync Setup For Masked Based Event Generation */

/* csl fsync initialization */
 CSL_fsyncInit (NULL);

/* Open fsync handle for */
 hFsync = CSL_fsyncOpen(&myFsyncObj, CSL_FSYNC, NULL, &status);

/* Read the setup values from user defined parameters */ myFsyncCfg = fsyncSetupValues
(fsyncSetupParam[0]);

/*Program the frame sync setup registers namely CTL1 and Init Registers */
 CSL_fsyncHwSetup (hFsync, &myFsyncCfg);

/* Function to fill user defined mask event gen params */
 myFsyncMaskEventCfg = fsyncMaskEventSetupValues(&fsyncMaskEvntSetupParam[0]);

/* configure mask based trigger generator 0 */
 CSL_fsyncHwControl (hFsync, CSL_FSYNC_CMD_CONFIG_MASK_BASED_TRIGGER_GEN, (void
*)&myFsyncMaskEventCfg);

/* Enable trigger generator 0 */
 CSL_fsyncHwControl (hFsync, CSL_FSYNC_CMD_ENABLE_TRIGGER_GEN, (void
*)&(myFsyncMaskEventCfg.eventGenUsed));

/* Enable timer */
 CSL_fsyncHwControl(hFsync, CSL_FSYNC_CMD_ENABLE_TIMER, (void *)&ctrlArg);

/* set RUN bit in CTL2 */
 CSL_fsyncSetOkStatBit(hFsync);

/* ISR for fsync event */
 CSL_FsyncTimerCountObj querySysTimerValue;

/* Disable the Trigger Generation */
 CSL_fsyncHwControl(hFsync, CSL_FSYNC_CMD_DISABLE_TRIGGER_GEN, (void
*)&(myFsyncMaskEventCfg.eventGenUsed));

/* Halt the Timer */
 CSL_fsyncHwControl(hFsync, CSL_FSYNC_CMD_DISABLE_TIMER, NULL);

/* Read the value of SYS timer */
 CSL_fsyncGetHwStatus(hFsync, CSL_FSYNC_QUERY_SYSTEM_TIMER_VALUE, (void *)&querySysTimerValue);
//

/*Close fsync handle */
 CSL_fsyncClose(hFsync);
```

```
/*Close interrupt handle */
 CSL_intcHwControl(hIntcFsync, CSL_INTC_CMD_EVTCLEAR, NULL);
 CSL_intcClose(hIntcFsync);
```

2.2.5.4 Counter-Based Event Generator

The second set of events is generated using counters. There is a trigger value that is programmed into the event generator's compare counter register (EGC_COUNTER). There is a mask and sub-chip trigger value that is only used to select sub-chip count values for finer granularity of event placement. When a comparison is made for the sub-chip trigger value, the counter is incremented. This counter has a programmable count value of delay-1. When the counter reaches its full count (counts down to 0), an event is generated. The counter is started when the event is enabled and the first sync is generated from the SYNC sub-module and the OFFSET_#_ENABLE is active from the event's offset counter (EGC_CNTRL). Each of these events can be disabled by clearing a programmable enable/disable bit for each event. System events FSYNC_SYSEVNT10_PO through FSYNC_SYSEVNT17_PO are counter-based. The Figure 8 shows the flow of counter-based event generation.

Figure 8. Counter Based Event Flow



The sample code in Section 2.2.5.4.1 shows the programming sequence for counter-based event generation.

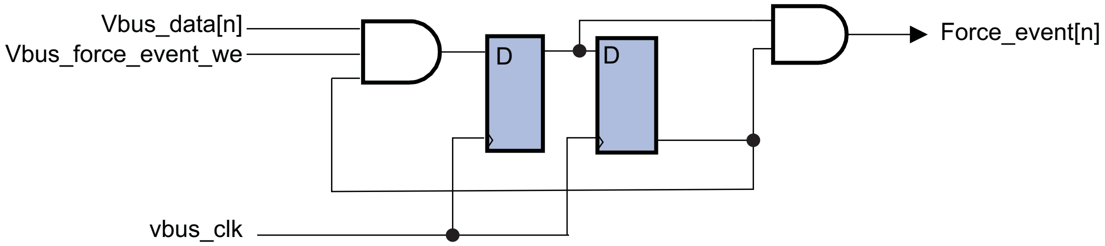
2.2.5.4.1 Counter-Based Event Generation Programming Sequence

```
CSL_IntcHandle hIntcFsync;
CSL_FsyncSetup myFsyncCfg;
CSL_FsyncHandle hFsync;
CSL_FsyncCounterTriggerGenObj myFsyncCounterEventCfg;
CSL_BitMask16 ctrlArg;

/*Interrupt Setup for Fsync Event Generation */

/* fsync Event FSEVT10 is hooked to int04 */
 hIntcFsync = fsyncIntrInit (CSL_INTC_VECTID_4,CSL_INTC_EVENTID_92);

/* Hook the ISRs for event2 */
 CSL_intcHookIsr (CSL_INTC_VECTID_4,&fsync_event2_ISR);

/* Clear the Event */
 CSL_intcHwControl (hIntcFsync, CSL_INTC_CMD_EVTCLEAR, NULL);

/* Enable the Event */
 CSL_intcHwControl (hIntcFsync, CSL_INTC_CMD_EVTENABLE, NULL);

/* Fsync Setup For Masked Based Event Generation */

/* csl fsync initialization */
 CSL_fsyncInit (NULL);

/* Open fsync handle for */
 hFsync = CSL_fsyncOpen(&myFsyncObj, CSL_FSYNC, NULL, &status);

/* Read the setup values from user defined parameters */
 myFsyncCfg = fsyncSetupValues (fsyncSetupParam[0]);

/*Program the frame sync setup registers namely CTL1 and Init Registers */
 CSL_fsyncHwSetup (hFsync, &myFsyncCfg);

/* Function to fill user defined counter event gen params */
 myFsyncCounterEventCfg = fsyncMaskEventSetupValues(&fsyncCounterEvntSetupParam[0]);

/* configure mask based trigger generator 0 */
 CSL_fsyncHwControl (hFsync, CSL_FSYNC_CMD_CONFIG_COUNTER_BASED_TRIGGER_GEN, (void
*)&myFsyncCounterEventCfg);

/* Enable trigger generator 0 */
 CSL_fsyncHwControl (hFsync, CSL_FSYNC_CMD_ENABLE_TRIGGER_GEN, (void
*)&(myFsyncCounterEventCfg.eventGenUsed));

/* Enable timer */
 CSL_fsyncHwControl(hFsync, CSL_FSYNC_CMD_ENABLE_TIMER, (void *)&ctrlArg);

/* set RUN bit in CTL2 */
 CSL_fsyncSetOkStatBit(hFsync);

/* ISR for fsync event */
 CSL_FsyncTimerCountObj querySysTimerValue;

/* Disable the Trigger Generation */
 CSL_fsyncHwControl(hFsync, CSL_FSYNC_CMD_DISABLE_TRIGGER_GEN, (void
*)&(myFsyncCounterEventCfg.eventGenUsed));

/* Halt the Timer */
 CSL_fsyncHwControl(hFsync, CSL_FSYNC_CMD_DISABLE_TIMER, NULL);

/* Read the value of SYS timer */
 CSL_fsyncGetHwStatus(hFsync, CSL_FSYNC_QUERY_SYSTEM_TIMER_VALUE, (void *)&querySysTimerValue);
//

/*Close fsync handle */
 CSL_fsyncClose(hFsync);

/*Close interrupt handle */
 CSL_intcHwControl(hIntcFsync, CSL_INTC_CMD_EVTCLEAR, NULL);
 CSL_intcClose(hIntcFsync);
```

2.2.5.5 Force Events

The frame sync events FSYNC_SYSEVENT0 to FSYNC_SYSEVENT29 are generated by a VBUS register write to the EVT_FORCE register. Force event generation is not used in normal functional operation, mostly it is used in system debug. The event generated is a single shot of one VBUS clock period. Figure 9 shows the flow of force event generation.

Figure 9. Force Event Generation



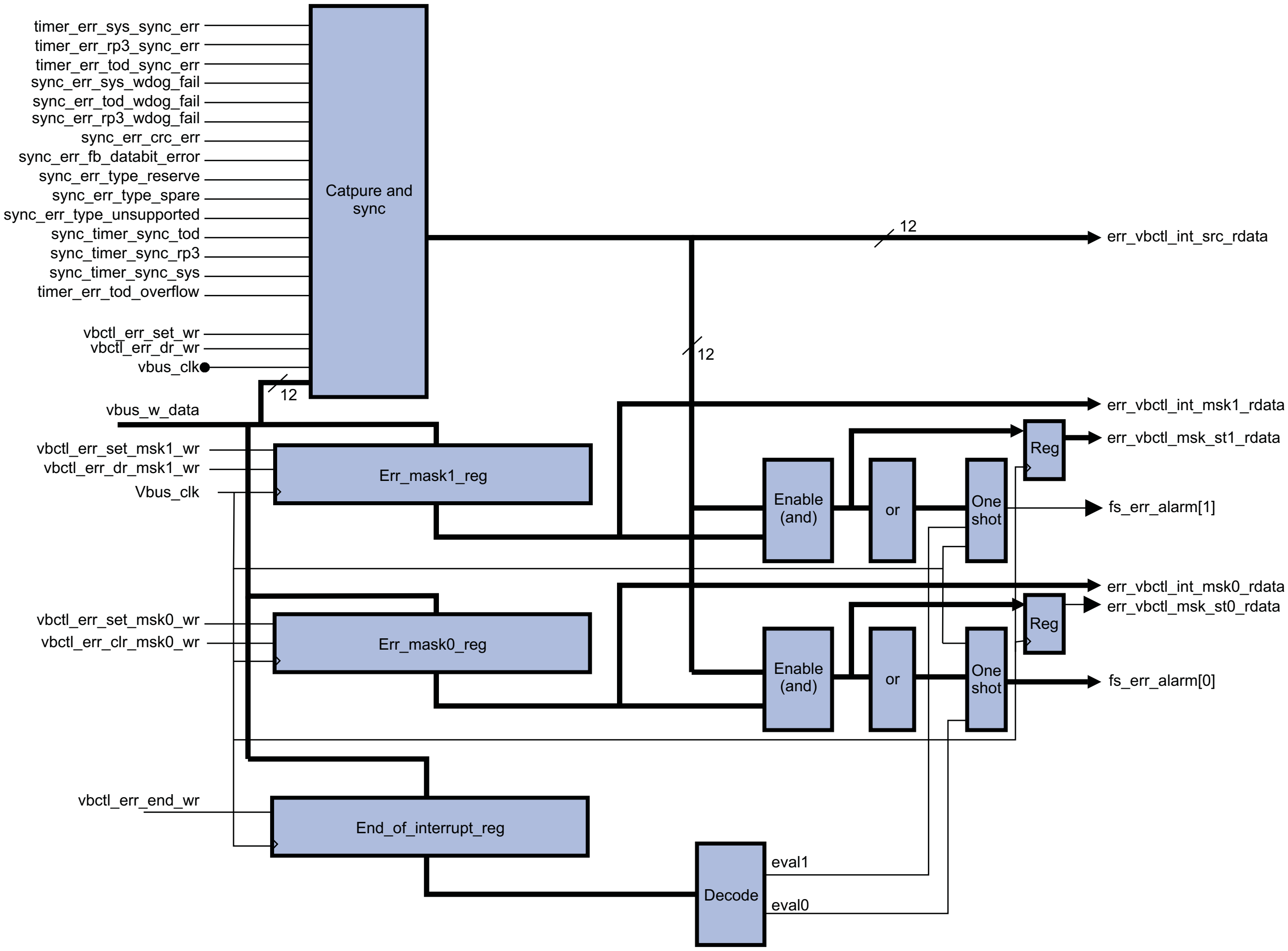
2.2.5.5.1 Force Events Programming Sequence

```
CSL_FsyncObj myFsyncObj;
CSL_FsyncHandle hFsync;
CSL_BitMask16 ctrlArg;
CSL_IntcHandle hIntcFsync;
CSL_FsyncTriggerGenNum myFsyncTriggerEventNum;
CSL_Status status;

/*Interrupt Setup for Fsync Event Generation */
 hIntcFsync = fsyncIntrInit(CSL_INTC_VECTID_4,CSL_INTC_EVENTID_15);

/* Hook the ISRs for event0 */
 CSL_intcHookIsr(CSL_INTC_VECTID_4,&fsync_ForceEvent_ISR);

/* Clear the Event */
 CSL_intcHwControl(hIntcFsync, CSL_INTC_CMD_EVTCLEAR, NULL);

/* Enable the Event */
 CSL_intcHwControl (hIntcFsync, CSL_INTC_CMD_EVTENABLE, NULL);

/* Initialize CSL library, this step is required */
 CSL_fsyncInit(NULL);

/* Open Fsync Handle */
 hFsync = CSL_fsyncOpen(&myFsyncObj,CSL_FSYNC, NULL, &status);

if ((hFsync == NULL) || (status != CSL_SOK))
 {
printf ("\nError opening CSL_FSYNC");
exit(1);
 }
/*Event Number is given as input */
 myFsyncTriggerEventNum = fsyncForceEvntSetupParam[0];
 /* Enable RP3/Sys timer */
  CSL_fsyncEnableTimer(hFsync);
 /* Program FSYNC_Event_Force_Reg to trigger events */
  CSL_fsyncHwControl(hFsync, CSL_FSYNC_CMD_SET_TRIGGER_EVENT, (void *)&myFsyncTriggerEventNum);

/* wait for event */

/* ISR can be written as per user requirements */
```

2.2.5.6 Error and Alarm Generations

There are two system event wires that are dedicated from the frame synchronization module for error and alarm conditions. The user arbitrarily chooses one to be *error* and the other to be *alarm*. Any of the error/alarm states can be ORed to trigger either or both of the two dedicated system event wires. Both these system event wires are broadcast to all DSP cores and major TMS320C6474 subsystems for actual routing and use. An example subsystem is a general-purpose timer. If the user requires an alarm condition to interrupt a given DSP core, the particular wire that is designated as alarm should be enabled as an interrupt.

Figure 10. Error and Alarm Generation



Possible error or alarm conditions are received from various circuits in the FSYNC module. The error conditions are synchronized to the VBUS clock and captured with a rising-edge detect. These raw errors can be set or cleared via VBUS writes. The resultant error flags are enabled or disabled with the two mask registers shown in Figure 10. Each mask register is set or cleared. The raw-error and masked-error flags can be read via VBUS reads. The raw errors are set and cleared under software control for debug purposes. The various fail conditions are listed in Table 8.

Table 8. Error and Alarm Conditions

Bit	Error	Description
0	SYSTEM_FRAME_FAIL	RP1, TRT, or UMTS_SYNC frame synchronization source was resynchronized. The system timer predicted a different frame alignment or RP1 frame number mismatch than the SYNC_BURST, or alternate synchronization pulse, dictated. Bit is not set at the first synchronization after reset.
1	RP3_FRAME_FAIL	RP1 or UMTS_SYNC frame synchronization source was resynchronized. The RP3 timer predicted a different frame alignment or RP1 frame number mismatch than the SYNC_BURST, or alternate synchronization pulse, dictated. Bit is not set at the first synchronization after reset.

Table 8. Error and Alarm Conditions (continued)

Bit	Error	Description
2	TOD_FRAME_FAIL	RP1 frame synchronization source was resynchronized. The timers predicted a different frame alignment or RP1 timestamp number mismatch than the SYNC_BURST dictated. RP1 only. Bit is not set at the first synchronization after reset.
3	FSYNC_WDOG_FAIL	Did not receive a new SYNC_BURST to update the timers within the minimum update time. Minimum update time is dictated by the FSYNC Update Register. RP1 only.
4	RP3_WDOG_FAIL	Did not receive a new SYNC_BURST to update the timers within the minimum update time. Minimum update time is dictated by the FSYNC Update Register. RP1 only.
5	TOD_WDOG_FAIL	Did not receive a new SYNC_BURST to update the timers within the minimum update time. Minimum update time is dictated by the FSYNC Update Register. RP1 only.
6	CRC_FAIL	CRC check failed. RP1 only.
7	BIT_WIDTH_FAIL	Each bit of the SYNC_BURST is supposed to be eight SCLKs in length. This error condition is flagged if a bit is measured to be less than eight SCLKs or if the end bit is not found. RP1 only.
8	TYPE_RESERVE	SYNC_BURST is of type *Reserved* or *Not Used*. The reserved format should never be observed on the frame synchronization interface using the RP1 mechanism. RP1 only. Not set when a frame burst is discarded due to CRC errors.
9	TYPE_SPARE	SYNC_BURST is of type *Spare* using the RP1 mechanism. This is not a true error condition. It is envisioned that this error condition would be masked. RP1 only. Not set when a frame burst is discarded due to CRC errors.
10	TYPE_UNSUPPORTED	SYNC_BURST is of a type that is different than the type programmed in the FSYNC RP1 Type Select Register. RP1 only. Not set when a frame burst is discarded due to CRC errors.
11	TYPE_TOD	SYNC_BURST is of type *Time of Day*. RP1 only. Not set when a frame burst is discarded due to CRC errors. May be used to check resync of timer when the CTL1 register, RESYNC_MODE = 1.

12	TYPE_RP3	RP1_MODE = 1: TYPE_RP3 = 1; SYNC_BURST is of type RP3.
		RP1_MODE = 0: TYPE_RP3 = 0
		Not set when a frame burst is discarded due to CRC errors.
		May be used to check resync of timer when the CTL1 register, RESYNC_MODE = 1.

13	TYPE_SYSTEM	RP1_MODE = 1: TYPE_SYSTEM = 1; SYNC_BURST is of a type that matches the selected system type. RP1_MODE = 0; TYPE_SYSTEM = 0 Not set when a frame burst is discarded due to CRC errors. May be used to check resync of timer when the CTL1 register, RESYNC_MODE = 1.
14	TOD_OVERFLOW	Time-of-day counter has overflowed.

Follow this sequence when an error/alarm condition occurs:

1. Set the ERR_SET_MASK_0/ERR_SET_MASK_1 register to enable the selected interrupt flags. The bit positions are shown in Table 8.
2. When a fail condition occurs and its mask bit is set to 0x1, it will produce a one VBUS_CLK width that is wired to interrupt handling logic. Once this FSYNC_ERR_ALARM pulse is generated, subsequent pulses are disabled until further interrupt processing by the CPU.
3. The CPU reads the ERR_INT_SRC_RAW register which shows all the fail conditions that have occurred and it reads the ERR_MASKED_STAT_0/ERR_MASKED_STAT_1 register to find the masked source state of interrupt. The value 0x1 shows it is active.
4. The CPU then clears the ERR_CLEAR_MASK_0/ERR_CLEAR_MASK_1 register, if needed.
5. If the CPU wants to re-evaluate/re-arm the interrupt, then there is no need to clear the ERR_CLEAR_MASK_0/ERR_CLEAR_MASK_1 register, but the ERR_END_OF_INT register needs to be programmed. The interrupt line will be re-evaluated and generate a single pulse if the aggregated masked-interrupt source is high.

The FSYNC module also supports setting the interrupts by programming the ERR_INT_SET register and clearing the interrupt by programming the ERR_INT_CLEAR register.

2.2.5.6.1 Error/Alarm Programming Sequence

```
CSL_FsyncObj myFsyncObj;
CSL_FsyncHandle hFsync;
CSL_FsyncSetup myFsyncCfg;
```

```
CSL_FsyncRp1PayloadObj *pfsyncWcdmaFddCaptureVal;
CSL_FsyncRp1TypeField *pfsyncRp1TypeCaptureVal;
CSL_IntcHandle hIntcFsync;
CSL_FsyncErrEventMaskObj myFsyncErrEventMaskCfg;
CSL_Status status;

/* The below function do the setup for interrupt and hook the isr */
/* fsync Event CICn_EVT12[Pin 92] of GEM Intr Controller is hooked to int04 */

hIntcFsync = fsyncIntrInit(CSL_INTC_VECTID_4,CSL_INTC_EVENTID_92);

/* Hook the ISRs for event0 */
 CSL_intcHookIsr(CSL_INTC_VECTID_4,&fsync_eventErr0_ISR);

/* Clear the Event */
 CSL_intcHwControl(hIntcFsync, CSL_INTC_CMD_EVTCLEAR, NULL);

/* Enable the Event */
 CSL_intcHwControl (hIntcFsync, CSL_INTC_CMD_EVTENABLE, NULL);

/* Initialize CSL library, this step is required */
 CSL_fsyncInit(NULL);

/* Open handle for fsync */
 hFsync = CSL_fsyncOpen(&myFsyncObj,CSL_FSYNC, NULL, &status);

/* Read the setup values from user defined parameters */
 myFsyncCfg = fsyncSetupValues(fsyncSetupParam[0]);

/*Program the frame sync setup registers namely CTL1 and Init Registers */
 CSL_fsyncHwSetup(hFsync, &myFsyncCfg);

/*Fill the user defined values in ErrSetMask0 and ErrSetMask1 Reg */
 myFsyncErrEventMaskCfg = fsyncErrEventSetupValues(fsyncErrEvntSetupParam[0]);

/* Program Err Set Mask 0 and Err Set Mask 1 to enable System Frame Error condition*/
 CSL_fsyncHwControl(hFsync, CSL_FSYNC_CMD_ENABLE_ERR_EVENT, (void *)&myFsyncErrEventMaskCfg);

CSL_fsyncHwControl(hFsync, CSL_FSYNC_CMD_ENABLE_TIMER, (void *)&ctrlArg);
CSL_fsyncSetOkStatBit(hFsync);

/* Program the test bench in such that frame numbers are not incremented for System Frame */
/*This creates SYSTEM FRAME FAIL error condition .The event can be connected to one of the CPU
INT pin*/
/*ISR can be written to read the status of error interrupt as shown below*/

// Read value of Err Masked Status0 and Err Masked Status 1 reg
 CSL_fsyncGetHwStatus(hFsync, CSL_FSYNC_QUERY_ERR_EVENT_0, (void *)&errMaskStat0Value);
 CSL_fsyncGetHwStatus(hFsync, CSL_FSYNC_QUERY_ERR_EVENT_1, (void *)&errMaskStat1Value);

// Read value of Err Int Status0 and Err Int Status 1 reg
 fsync_CSL_fsyncGetErrIntMask0Reg(hFsync,&errIntMask0);
 fsync_CSL_fsyncGetErrIntMask1Reg(hFsync,&errIntMask1);
 fsync_CSL_fsyncGetErrIntSourceRawReg(hFsync,&errIntRaw);
/*Disable Error Event Generation */
 CSL_fsyncHwControl(hFsync, CSL_FSYNC_CMD_DISABLE_ERR_EVENT,(void *)&myFsyncErrEventMaskCfg);
/*Close the Fsync Handle */
 CSL_fsyncClose(hFsync);
 CSL_intcHwControl(hIntcFsync, CSL_INTC_CMD_EVTCLEAR, NULL);
 CSL_intcClose(hIntcFsync);
```

2.2.5.7 Emulation Control

The EMUCTL register has two bits, FREE and SOFT, that determine the FSYNC behavior when the EMU_SUSP input is asserted (= 1). Table 9 shows the behavior of the FSYNC module when the FREE and SOFT bits are programmed.

Table 9. EMUSUSP Control

FREE	SOFT	Halt Some Events (EMU_EVENT_DISABLE)	Halt All Events	Halt Timers
0	0	1	0	0
0	1	0	1	1
1	X	0	0	0

The EMUMASK register is used to control the behavior of the event generation during emulation suspend. There are 30 bits that disable the associated event when they are set.

2.3 Initialization

The FSYNC module should be initialized as follows:

- Perform the interrupt handling setup, if needed.
- Program the FSYNC control registers either for event generation or error/alarm checking.
- Arm the timer by programming the FSYNC_CTL2_ARM_TIMER bit of the CTL2 register.
- To read the timer values, halt the timer by programming the FSYNC_CTL2_TIMER_HALT bit of the CTL2 register and read the RP3VAL/SYSVAL registers.

For RP1 frame synchronization, ensure that the SYNC_BURST is updated by reading the capture register. The timer will start only when the SYNC_BURST is updated and the FSYNC_CTL2_ARM_TIMER bit is set.

For non-RP1 frame synchronization, the timer starts only after the arrival of the sync signal. The FSYNC_CTL2_ARM_TIMER bit of the CTL2 register should be set.

Even though force event logic is not related to timer, the FSYNC_CTL2_ARM_TIMER bit of the CTL2 register should be set after programming the EVT_FORCE register.

2.3.1 Restarting

To reprogram the timers for event generation, use the following sequence:

1. Halt the timer by programming the FSYNC_CTL2_TIMER_HALT bit in the CTL2 register.
2. Clear the FSYNC_CTL2_ARM_TIMER bit of the CTL2 register (disarming).
3. Clear the FSYNC_CTL2_TIMER_HALT bit of the CTL2 register.
4. Perform any set-up changes to timers and event generators.
5. Set the FSYNC_CTL2_ARM_TIMER bit of the CTL2 register (arming) and wait for the sync pulse.

Do not reprogram the timer register before disarming the timer.

3 Registers

All memory-mapped registers allow for 32-bit access only. For all software registers, access is limited to a 32-bit word.

3.1 FSYNC Control and Status Registers

The FSYNC control and status registers are classified into four categories:

- FSYNC Control Registers
- FSYNC Terminal Count Registers
- FSYNC Error/Alarm Registers
- FSYNC Timer Control Registers.

3.1.1 FSYNC Control Registers

The initial bit value after reset for all registers is 0, unless otherwise noted. The address range for FSYNC is base address + 000 to 3FF (hexadecimal). Base address 0XBBBB_BNNN, where NNN = hex address, offset is specified in Table 10.

Table 10. RP1 Control and Status Registers

Offset Address	Acronym	Register Name	See
0B0	SCRATCH	Scratch Register	Section 3.1.1.1
0B4	CTL1	FSYNC Control Register 1	Section 3.1.1.2
0B8	CTL2	FSYNC Control Register 2	Section 3.1.1.3
0BC	EMUCTL	FSYNC Emulation Control Register	Section 3.1.1.4
0C0	EMUMASK	FSYNC Emulation Mask Register	Section 3.1.1.5
0C4	RP1TS	FSYNC RP1 Type Select Register	Section 3.1.1.6
0C8	UPDATE	FSYNC Update Register	Section 3.1.1.7
0CC	RP3INIT	FSYNC RP3 Initialization Register	Section 3.1.1.8
0D0	SYSINIT	FSYNC System Initialization Register	Section 3.1.1.9
128	TOD1	FSYNC Time-of-Day Capture Register 1	Section 3.1.1.10
12C	TOD2	FSYNC Time-of-Day Capture Register 2	Section 3.1.1.11
130	RP31	FSYNC RP3 Capture Register 1	Section 3.1.1.12
134	RP32	FSYNC RP3 Capture Register 2	Section 3.1.1.13
138	SYS1	FSYNC System Capture Register 1	Section 3.1.1.14
13C	SYS2	FSYNC System Capture Register 2	Section 3.1.1.15
168	SYSTC	FSYNC System Terminal Count Register	Section 3.1.1.16
16C	RP3TC	FSYNC RP3 Terminal Count Register	Section 3.1.1.17
170	TYPE	FSYNC Type Capture Register	Section 3.1.1.18
174	TODVAL	FSYNC Time-of-Day Value Register	Section 3.1.1.19
178	RP3VAL	FSYNC RP3 Value Register	Section 3.1.1.20
17C	SYSVAL	FSYNC System Value Register	Section 3.1.1.21

3.1.1.1 Scratch Register

The scratch register (SCRATCH) is mainly used for debugging purpose. It is shown in Figure 11.

Figure 11. Scratch Register

31	0
SCRATCH_REGISTER	
R/W-0	

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

3.1.1.2 FSYNC Control Register 1 (CTL1)

The FSYNC control register 1 (CTL1) sets up the RP1 mode configuration. The initial bit value after reset for all fields is 0, unless otherwise noted. The CTL1 register is shown in Figure 12 and described in Table 11.

Figure 12. FSYNC Control Register 1 (CTL1)

31	30	29	28	27	26	25	24
Reserved		CRC_INVERT	CRC_INIT_VALUE	CRC_POSITION	TOD_LEAP_USAGE	CRC_USAGE	RESYNC_MODE
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

23	22	21			18	17	16
SYNC_MODE	RP3_EQUAL_SYS_TIMER	TOD_SYNC_DELAY				SYSTEM_SYNC_DELAY	
R/W-0	R/W-0	R/W-0				R/W-0	

15	14	13			10	9	8
SYSTEM_SYNC_DELAY		RP3_SYNC_DELAY				Reserved	CLK_SYSTEM_TIMER
R/W-0		R/W-0				R-0	R/W-0

7	6	5	4	3	2	1	0
CLK_SYSTEM_TIMER	Reserved	CLK_RP3_TIMER		SYNC_SYSTEM_TIMER		SYNC_RP3_TIMER	
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 11. FSYNC Control Register 1 (CTL1) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved		Reserved
29	CRC_INVERT		CRC Invert
		0	CRC is not inverted (default)
		1	CRC is inverted
28	CRC_INIT_VALUE		CRC Initialization Value
		0	CRC calculator is initialized to 0x0000 (default)
		1	CRC calculator is initialized to 0xFFFF
27	CRC_POSITION		CRC Position
		0	CRC bit 16 received first (default)
		1	CRC bit 1 received first
26	TOD_LEAP_USAGE		Time-of-Day Leapsecond Usage
		0	Do not add leapseconds to time of day timer (default)
		1	Add leapseconds to the time of day timer
25	CRC_USAGE		CRC Usage
		0	Regardless of CRC, use the sync_burst message (default) CRC error reporting will be disabled
		1	If CRC check fails, discard sync_burst message The CRC will be checked for errors in RP1_mode only
24	RESYNC_MODE		Resynchronization Mode
		0	Do not automatically resynchronize (default)
		1	Automatically resynchronize when a new sync is out of alignment
23	SYNC_MODE		Synchronization Mode
		0	Non-RP1, disable RP1 sync mechanism
		1	OBSAI-RP1 synchronization is used (default)

Table 11. FSYNC Control Register 1 (CTL1) Field Descriptions (continued)

Bit	Field	Value	Description
22	RP3_EQUAL_SYS_TIMER		RP3 Equalization System Timer
		0	Separate RP3 and system timer used (default)
		1	System timer only is used
21-18	TOD_SYNC_DELAY		Time-of-Day Synchronization Delay Select amount of delay for TOD frame synchronization. Delay is in whole selected input clock increments. 0000b (default)
17-14	SYSTEM_SYNC_DELAY		System Synchronization Delay Select amount of delay for system frame synchronization. Delay is in whole selected input clock increments. 0000b (default)
13-10	RP3_SYNC_DELAY		RP3 Synchronization Delay Select amount of delay for RP3 frame synchronization. Delay is in selected input clock increments. 0000b (default)
9	Reserved		Reserved
8-7	CLK_SYSTEM_TIMER		Clock System Timer
		0	VBUS_CLK/3 (default)
		1	TRT_CLOCK
		2	FSYNC_CLOCK differential inputs
		3	UMTS_CLOCK
6	Reserved		Reserved
5-4	CLK_RP3_TIMER		Clock RP3 Timer
		0	VBUS_CLK/3 (default)
		1	TRT_CLOCK (not normally used)
		2	FSYNC_CLOCK differential inputs
		3	UMTS_CLOCK
3-2	SYNC_SYSTEM_TIMER		Synchronization System Timer
		0	OBSAI-RP1 differential FRAME_BURST (default)
		1	UMTS_SYNC
		2	TRT
		3	SYSTEM_TEST_SYNC
1-0	SYNC_RP3_TIMER		Synchronization RP3 Timer
		0	OBSAI-RP1 differential FRAME_BURST (default)
		1	UMTS_SYNC
		2	TRT (not normally used)
		3	SYSTEM_TEST_SYNC

3.1.1.3 FSYNC Control Register 2 (CTL2)

The FSYNC control register 2 (CTL2) controls the arming, halting, SYSTEM_TEST_SYSNC and some run-time control. The FSYNC_CTL2_ARM_TIMER bit in this register should be enabled for generating events. The CTL2 register is shown in Figure 13 and described in Table 12.

Figure 13. FSYNC Control Register 2 (CTL2)

31	8
Reserved	
R-0	

7	6	5	4
FSYNC_CTL2_FSYNC_RUN	FSYNC_CTL2_SYS_TEST_SYNC	Reserved	
R/W-0	R/W-0	R-0	

3	2	1	0
Reserved	FSYNC_CTL2_ARM_TIMER	Reserved	FSYNC_CTL2_TIMER_HALT
R-0	R/W-0	R-0	R/W-0

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

Table 12. FSYNC Control Register 2 (CTL2) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7	FSYNC_CTL2_FSYNC_RUN		FSYNC_CTL2_FSYNC_RUN = 1, FSYNC is running. This bit will be cleared upon a module reset and may also cleared by software .This bit will be set by the CPU after the frame synchronization module detects its first synchronization input and starts the system timer and the CPU has determined that events are being generated as expected. If an error occurs, fs_err_alarm =1, while running, this bit will be cleared. If the FSYNC is auto resynchronized, FSYNC_CTL2_FSYNC_RUN bit will need to be set again by the CPU.
6	FSYNC_CTL2_SYS_TEST_SYNC		Create an internal sync pulse. Only used when the CTL1 Register syncRP3Timer and syncSystemTimer both select this. Once this bit is enabled it effectively forces the start of both RP3 and System timer by forcing a sync when it is selected.
5-3	Reserved		Reserved
2	FSYNC_CTL2_ARM_TIMER		Arms both RP3 and System timers to start when synchronization occurs. Enables event generation when =1.
1	Reserved		Reserved
0	FSYNC_CTL2_TIMER_HALT		Halts both RP3 and System timers.

3.1.1.4 FSYNC Emulation Control Register (EMUCTL)

The FSYNC emulation control register (EMUCTL) is used to control the FSYNC behavior when EMUSUSP input is asserted. The EMUCTL register is shown in Figure 14 and described in Table 13.

Figure 14. FSYNC Emulation Control Register (EMUCTL)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		FREE	SOFT
R-0		R/W-0	R/W-0

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

Table 13. FSYNC Emulation Control Register (EMUCTL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	FREE		Free Ignore EMUSUSP input signal to FSYNC
0	SOFT		Soft
		0	Free-run timers disable some events use, EMUMASK register to control
		1	Halt timer events

3.1.1.5 FSYNC Emulation Mask Register (EMUMASK)

The FSYNC emulation mask register (EMUMASK) is used to disable some events when the SOFT bit in the EMUCTL register is 0. The EMUMASK register is shown in Figure 15 and described in Table 14.

Figure 15. FSYNC Emulation Mask Register (EMUMASK)

31	30	29	0
Rsvd		DISABLE_ASSOCIATED_EVENTS	
R-0		R/W-0	

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

Table 14. FSYNC Emulation Mask Register (EMUMASK) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved		Reserved
29-0	DISABLE_ASSOCIATED_EVENTS		Disable Associated Events Disable events when EMUSUSP is allowed and the SOFT bit of EMUCTL = 0
		0	Do not disable associated event
		1	Disable associated event bit 0 = EVENT0, bit 1 = EVENT1, etc.

3.1.1.6 FSYNC RP1 Type Select Register (RP1TS)

The FSYNC RP1 type select register (RP1TS) is used to select the type field to synchronize with the system timer. This register can be programmed only for RP1-mode synchronization. The RP1TS register is shown in Figure 16 and described in Table 15.

Figure 16. FSYNC RP1 Type Select Register (RP1TS)

31	16
Reserved	
R-0	

15 8	7 0
Reserved	SYSTEM_TIMER_RP1_SYNC
R-0	R/W-0

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

Table 15. FSYNC RP1 Type Select Register (RP1TS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	SYSTEM_TIMER_RP1_SYNC		System Timer RP1 Synchronization The supported types to be programmed for RP1-mode synchronization are:
		02h	WCDMA/FDD
		07h	CDMA2000

3.1.1.7 FSYNC Update Register (UPDATE)

The values programmed in the FSYNC update register (UPDATE) are used in watchdog timers to flag error conditions if the timer times out before its synchronization burst type is found. There are three watchdog timers for the three burst types: RP3, system and time of day. The programmed time in this register is in number of RP3 frames for the RP3 watchdog, number of system frames for the system timer, and time-of-day watchdog. The UPDATE register is shown in Figure 17 and described in Table 16.

Figure 17. FSYNC Update Register (UPDATE)

31 24	23 16
Reserved	TOD_FRAME_UPDATE_RATE
R-0	R/W-0

15 8	7 0
RP3_FRAME_UPDATE_RATE	WCDMA_FRAME_UPDATE_RATE
R/W-0	R/W-0

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

Table 16. FSYNC Update Register (UPDATE) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reserved
23-16	TOD_FRAME_UPDATE_RATE		Time-of-Day Frame Update Rate in system frame boundaries
15-8	RP3_FRAME_UPDATE_RATE		RP3 Frame Update Rate in RP3 frame boundaries
7-0	WCDMA_FRAME_UPDATE_RATE		System Frame Update Rate in system frame boundaries

3.1.1.8 FSYNC RP3 Initialization Register (RP3INIT)

The FSYNC RP3 initialization register (RP3INIT) is used to load the initial value in the RP3 timer. This register is programmed in non-RP1-mode synchronization. The RP3INIT register is shown in Figure 18 and described in Table 17.

Figure 18. FSYNC RP3 Initialization Register (RP3INIT)

31 28	27 16
Reserved	FRAME_NUM
R-0	R/W-0

15 12	11 0
SLOT_NUM	CHIP_NUM
R/W-0	R/W-0

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

Table 17. FSYNC RP3 Initialization Register (RP3INIT) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved		Reserved
27-16	FRAME_NUM		Frame Number Writes the initial frame value of the RP3 timer, non-RP1-mode only
15-12	SLOT_NUM		Slot Number Writes the initial slot value of the RP3 timer, non-RP1-mode only
11-0	CHIP_NUM		Chip Number Writes the initial chip value of the RP3 timer, non-RP1-mode only

3.1.1.9 FSYNC System Initialization Register (SYSINIT)

The FSYNC system initialization register (SYSINIT) loads the initial value in the system timer. This register is programmed in non-RP1-mode synchronization. The SYSINIT register is shown in Table 18 and described in Figure 19.

Figure 19. FSYNC System Initialization Register (SYSINIT)

31 28	27 16
Reserved	FRAME_NUM
R-0	R/W-0

15 12	11 0
SLOT_NUM	CHIP_NUM
R/W-0	R/W-0

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

Table 18. FSYNC System Initialization Register (SYSINIT) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved		Reserved
27-16	FRAME_NUM		Frame Number Writes the initial frame value of the system timer, non-RP1-mode only
15-12	SLOT_NUM		Slot Number Writes the initial slot value of the system timer, non-RP1-mode only
11-0	CHIP_NUM		Chip Number Writes the initial chip value of the system timer, non-RP1-mode only

3.1.1.10 FSYNC Time-of-Day Capture Register 1 (TOD1)

The FSYNC time-of-day capture register 1 is a read-only register used to capture the RP1 information payload LSBs from the TOD type SYNC_BURST. The TOD1 register is shown in Figure 20 and described in Table 19.

Figure 20. FSYNC Time-of-Day Capture Register 1 (TOD1)

31 … 0
FSYNC_TOD1_TOD_LS
R-0

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

Table 19. FSYNC Time-of-Day Capture Register 1 (TOD1) Field Descriptions

Bit	Field	Value	Description
31-0	FSYNC_TOD1_TOD_LS		Time-of-Day Least-Significant bits Captures the RP1 type information payload LSBs [31:0] when a TOD type SYNC_BURST occurs

3.1.1.11 FSYNC Time-of-Day Capture Register 2 (TOD2)

The FSYNC time-of-day capture register 2 (TOD2) is a read-only register used to capture the RP1 information payload MSBs from the TOD type SYNC_BURST. The TOD2 register is shown in Figure 21 and described in Table 20.

Figure 21. FSYNC Time-of-Day Capture Register 2 (TOD2)

31	0
FSYNC_TOD1_TOD_MS	
R-0	

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

Table 20. FSYNC Time-of-Day Capture Register 2 (TOD2) Field Descriptions

Bit	Field	Value	Description
31-0	FSYNC_TOD1_TOD_MS		Time-of-Day Most-Significant bits Captures the RP1 type information payload MSBs [63:32] when a TOD type SYNC_BURST occurs

3.1.1.12 FSYNC RP3 Capture Register 1 (RP31)

The FSYNC RP3 capture register 1 (RP31) is a read-only register used to capture the RP1 information payload LSBs from the RP3 type SYNC_BURST. The RP31 register is shown in Figure 22 and described in Table 21.

Figure 22. FSYNC RP3 Capture Register 1 (RP31)

31 … 0
FSYNC_RP31_RP3_LS
R-0

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

Table 21. FSYNC RP3 Capture Register 1 (RP31) Field Descriptions

Bit	Field	Value	Description
31-0	FSYNC_RP31_RP3_LS		RP3 Least-Significant bits Captures the RP1 type information payload LSBs [31:0] when an RP3 type SYNC_BURST occurs

Submit Documentation Feedback

3.1.1.13 FSYNC RP3 Capture Register 2 (RP32)

The FSYNC RP3 capture register 2 (RP32) is a read-only register used to capture the RP1 information payload MSBs from the RP3 type SYNC_BURST. The RP32 register is shown in Figure 23and described in Table 22.

Figure 23. FSYNC RP3 Capture Register 2 (RP32)

31	0
FSYNC_RP31_RP3_MS	
R-0	

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

Table 22. FSYNC RP3 Capture Register 2 (RP32) Field Descriptions

Bit	Field	Value	Description
31-0	FSYNC_RP31_RP3_MS		RP3 Most-Significant bits Captures the RP1 type information payload MSBs [63:32] when an RP3 type SYNC_BURST occurs

3.1.1.14 FSYNC System Capture Register 1 (SYS1)

The FSYNC system capture register 1 (SYS1) is a read-only register used to capture the RP1 information payload LSBs from the system type SYNC_BURST. The SYS1 register is shown in Figure 24 and described in Table 23.

Figure 24. FSYNC System Capture Register 1 (SYS1)

<table>
<tr><td>31</td><td>0</td></tr>
<tr><td colspan="2">FSYNC_SYS1_SYS_LS</td></tr>
<tr><td colspan="2">R-0</td></tr>
</table>

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

Table 23. FSYNC System Capture Register 1 (SYS1) Field Descriptions

Bit	Field	Value	Description
31-0	FSYNC_SYS1_SYS_LS		System Least-Significant bits Captures the RP1 type information payload LSBs [31:0] when a WCDMA/FDD type SYNC_BURST occurs

3.1.1.15 FSYNC System Capture Register 2 (SYS2)

The FSYNC system capture register 2 (SYS2) is a read-only register used to capture the RP1 information payload MSBs from the system type SYNC_BURST. The SYS2 register is shown in Figure 25 and described in Table 24.

Figure 25. FSYNC System Capture Register 2 (SYS2)

31	0
FSYNC_SYS1_SYS_MS	
R-0	

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

Table 24. FSYNC System Capture Register 2 (SYS2) Field Descriptions

Bit	Field	Value	Description
31-0	FSYNC_SYS1_SYS_MS		System Most-Significant bits Captures the RP1 type information payload MSBs [63:32] when a WCDMA/FDD type SYNC_BURST occurs

3.1.1.16 FSYNC System Terminal Count Register (SYSTC)

The FSYNC system terminal count register (SYSTC) programs the chip, slot, and sub-chip system timer TC values. The chip TC values are programmed in the circular buffer and the chip TC RAM wrap address is programmed in the SYSTC register. The SYSTC register is shown in Figure 26 and described in Table 25.

Figure 26. FSYNC System Terminal Count Register (SYSTC)

31 16
Reserved
R-0

15 13	12 9	8 5	4 0
Reserved	NUM_CHIP_TERMINAL_COUNT-1	LAST_SLOT_NUM	LAST_SAMPLE_NUM
R-0	R/W-0	R/W-0	R/W-0

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

Table 25. FSYNC System Terminal Count Register (SYSTC) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved		Reserved
12-9	NUM_CHIP_TERMINAL_COUNT-1		Chip Terminal Count Number 1 The value programmed defines the circular buffer width. The default value is 0x0, which means the buffer depth is 0x1.
8-5	LAST_SLOT_NUM		Last Slot Number Slot terminal count controls the frame count. For example, if frame count is 0x4, when the slot counter reaches 0x4, the frame counter will be incremented by 0x1.
4-0	LAST_SAMPLE_NUM		Last Sample Number Input clock divider controls the chip rate

3.1.1.16.1 FSYNC System Timer Circular Buffer (SYSTC_RAM)

This circular buffer contains 10 locations allowing a maximum of 10 different CHIP_TC values. The buffer depth is controlled by the CHIP_TC_RAM wrap address bit of the SYSTC_RAM buffer. Table 26 shows the SYSTC_RAM buffer.

The base address is 0xbbbb_bnnn, where nnn = the hex address specified in the table.

Table 26. FSYNC System Timer Circular Buffer (SYSTC_RAM)

Hex Address	Bit [11:0]	Description
100	CHIP_TC0	Terminal count RAM for chip count.
104	CHIP_TC1	
108	CHIP_TC2	
10C	CHIP_TC3	
110	CHIP_TC4	
114	CHIP_TC5	
118	CHIP_TC6	
11C	CHIP_TC7	
120	CHIP_TC8	
124	CHIP_TC9	

3.1.1.17 FSYNC RP3 Terminal Count Register (RP3TC)

The FSYNC RP3 terminal count register (RP3TC) is used to program the chip, slot, and sub-chip RP3 timer TC values. The chip TC values are programmed in the circular buffer and the chip TC RAM wrap address is programmed in the RP3TC register. The RP3TC register is shown in Figure 27 and described in Table 27.

Figure 27. FSYNC RP3 Terminal Count Register (RP3TC)

31 16
Reserved
R-0

15 13	12 9	8 5	4 0
Reserved	NUM_CHIP_TERMINAL_COUNT-1	LAST_SLOT_NUM	LAST_SAMPLE_NUM
R-0	R/W-0	R/W-0	R/W-0

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

Table 27. FSYNC RP3 Terminal Count Register (RP3TC) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved		Reserved
12-9	NUM_CHIP_TERMINAL_COUNT-1		Chip Terminal Count Number 1 The value programmed defines the circular buffer width. The default value is 0x0, which means the buffer depth is 0x1.
8-5	LAST_SLOT_NUM		Last Slot Number Slot terminal count controls the frame count. For example, if frame count is 0x4, when the slot counter reaches 0x4, the frame counter will be incremented by 0x1.
4-0	LAST_SAMPLE_NUM		Last Sample Number Input clock divider controls the chip rate

3.1.1.17.1 RP3TC_ RAM

This circular buffer contains 10 locations allowing a maximum of 10 different CHIP_TC values. The buffer depth is controlled by the CHIP_TC_RAM wrap address bit of the RP3TC register. Table 28 shows the RP3TC_RAM buffer.

Table 28. FSYNC RP3 Timer Circular Buffer (RP3TC_RAM)

Hex Address	Bit [11:0]	Description
100	CHIP_TC0	Terminal count RAM for chip count.
104	CHIP_TC1	
108	CHIP_TC2	
10C	CHIP_TC3	
110	CHIP_TC4	
114	CHIP_TC5	
118	CHIP_TC6	
11C	CHIP_TC7	
120	CHIP_TC8	
124	CHIP_TC9	

3.1.1.18 FSYNC Type Capture Register (TYPE)

The FSYNC type capture register (TYPE) is a read-only register used to capture the type information from the SYNC_BURST. The TYPE register is shown in Figure 28 and described in Table 29.

Figure 28. FSYNC Type Capture Register (TYPE)

31	8
Reserved	
R-0	

7	0
FSYNC_TYPE_TYPE	
R-0	

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

Table 29. FSYNC Type Capture Register (TYPE) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	FSYNC_TYPE_TYPE		FSYNC Type Captures the RP1 type information when a new type is read

3.1.1.19 FSYNC Time-of-Day Value Register (TODVAL)

The FSYNC time-of-day value register (TODVAL) is a read-only register that gives the current value of the TOD timer. To read this register, halt the timer by programming the FSYNC_CTL2_TIMER_HALT bit of the CTL2 register and read the value of the timer. The TODVAL register is shown in Figure 29 and described in Table 30.

Figure 29. FSYNC Time-of-Day Value Register (TODVAL)

31 ... 0
FSYNC_TODVAL_VALUE
R-0

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

Table 30. FSYNC Time-of-Day Value Register (TODVAL) Field Descriptions

Bit	Field	Value	Description
31-0	FSYNC_TODVAL_VALUE		FSYNC Time-of-Day Value Reads the current value of the TOD timer.

3.1.1.20 FSYNC RP3 Value Register (RP3VAL)

The FSYNC RP3 Value Register (RP3VAL) is a read-only register that gives the current value of the RP3 timer. To read this register, halt the timer by programming the FSYNC_CTL2_TIMER_HALT bit of the CTL2 register and read the value of the timer. The RP3VAL register is shown in Figure 30 and described in Table 31.

Figure 30. FSYNC RP3 Value Register (RP3VAL)

31 – 28	27 – 16
Reserved	FSYNC_RP3VAL_FRAME
R-0	R-0

15 – 12	11 – 0
FSYNC_RP3VAL_SLOT	FSYNC_RP3VAL_CHIP
R-0	R-0

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

Table 31. FSYNC RP3 Value Register (RP3VAL) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved		Reserved
27-16	FSYNC_RP3VAL_FRAME		FSYNC RP3 Value Frame Reads the current frame value of the RP3 timer
15-12	FSYNC_RP3VAL_SLOT		FSYNC RP3 Value Slot Reads the current slot value of the RP3 timer
11-0	FSYNC_RP3VAL_CHIP		FSYNC RP3 Value Chip Reads the current chip value of the RP3 timer

3.1.1.21 FSYNC System Value Register (SYSVAL)

The FSYNC System Value Register (SYSVAL) is a read-only register which gives the current value of the system timer. To read this register, halt the timer by programming the FSYNC_CTL2_TIMER_HALT bit of the CTL2 register and read the value of the timer. The SYSVAL register is shown in Figure 31 and described in Table 32.

Figure 31. FSYNC System Value Register (SYSVAL)

31 28	27 16
Reserved	FSYNC_SYSVAL_FRAME
R-0	R-0

15 12	11 0
FSYNC_SYSVAL_SLOT	FSYNC_SYSVAL_CHIP
R-0	R-0

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

Table 32. FSYNC System Value Register (SYSVAL) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved		Reserved
27-16	FSYNC_SYSVAL_FRAME		FSYNC System Value Frame Read the current frame value of the RP3 timer
15-12	FSYNC_SYSVAL_SLOT		FSYNC System Value Slot Read the current slot value of the RP3 timer
11-0	FSYNC_SYSVAL_CHIP		FSYNC System Value Chip Read the current chip value of the RP3 timer

3.1.2 FSYNC Terminal Count Registers

The system and RP3 timers are controlled by a terminal count that is written by the DSP via the VBUS. The terminal count of the chip count is unique in that it has a circular buffer of terminal counts. The DSP must initialize the circular buffer by writing the last terminal count address to the control buffer depth; by default it is zero. For WCDMA/FDD and RP3, the last buffer address would likely be programmed as zero. This circular buffer serves the purpose of allowing for variable chip counts to accommodate variable slot sizes within the frame (in the case of the TPSCDMA sub-frame). Event generators may use the circular buffer state as well as the full timer value to trigger events. System and RP3 timers have their own set of terminal counts. Counters count up until they reach their respective TC values and then wrap to zeros. For RP1 synchronization, chip and slot portions are zeroed and sub-chip TC must be programmed to count 8 sub-chips (sub-chip TC = 7). This ensures that chips are counted in incremnts of 1/8th chip. The initial count values are written by the DSP via the VBUS, as is the case when using UMTS_SYNC or TRT in non-RP1-mode synchronization. By programming the sub-chip TC value, the system and RP3 timers must be able to count at 1x, 2x, 4x, 8x, 16x, and 32x chip-rate clock speeds. The sub-chip terminal count serves the purpose of a divider and must be loaded with the appropriate divisor; for example, 7h for RP1 30.72-MHz clock or 8x. Figure 32 shows the timer configuration concepts.

Figure 32. Timer Terminal Count Configuration

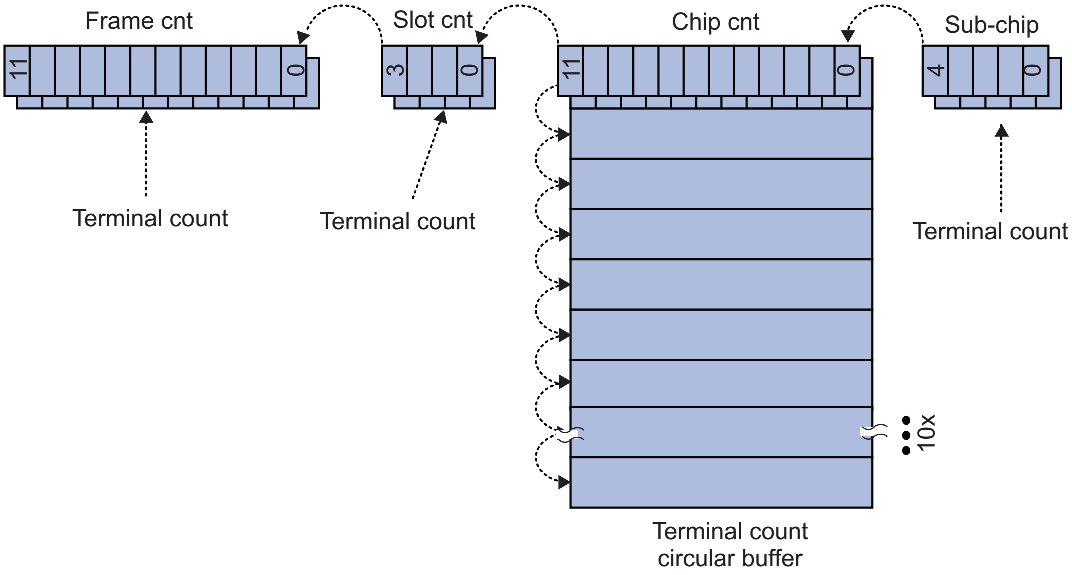


Figure 33 shows the timer TC configuration for RP1 interface.

The chip TC value or slot TC value is programmed for short frames. The values programmed for actual UMTS frames are shown in Figure 33. In this case, there is only one value in the circular buffer, so the chip TC RAM wrap address is zero and CHIP_TC0 = 2559.

Figure 33. Timer Configuration for RP1 Interface

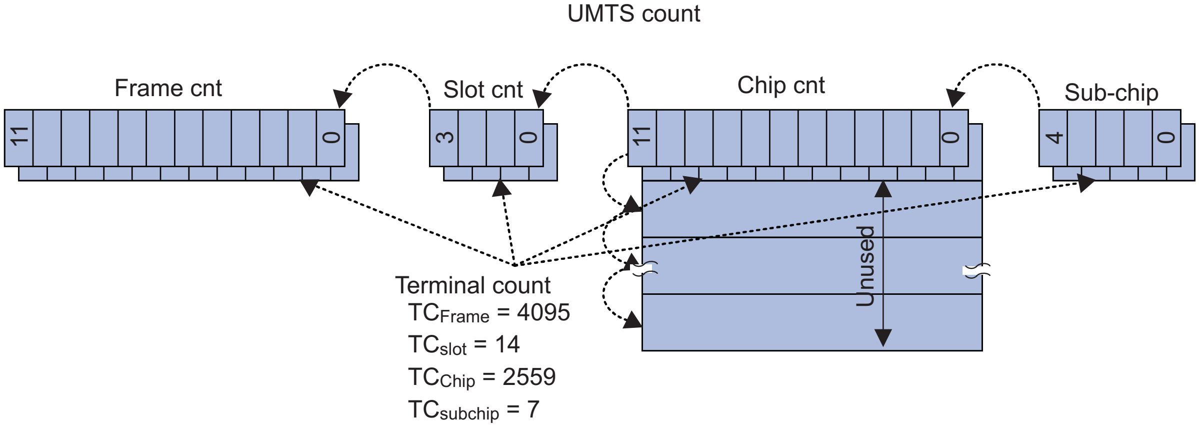
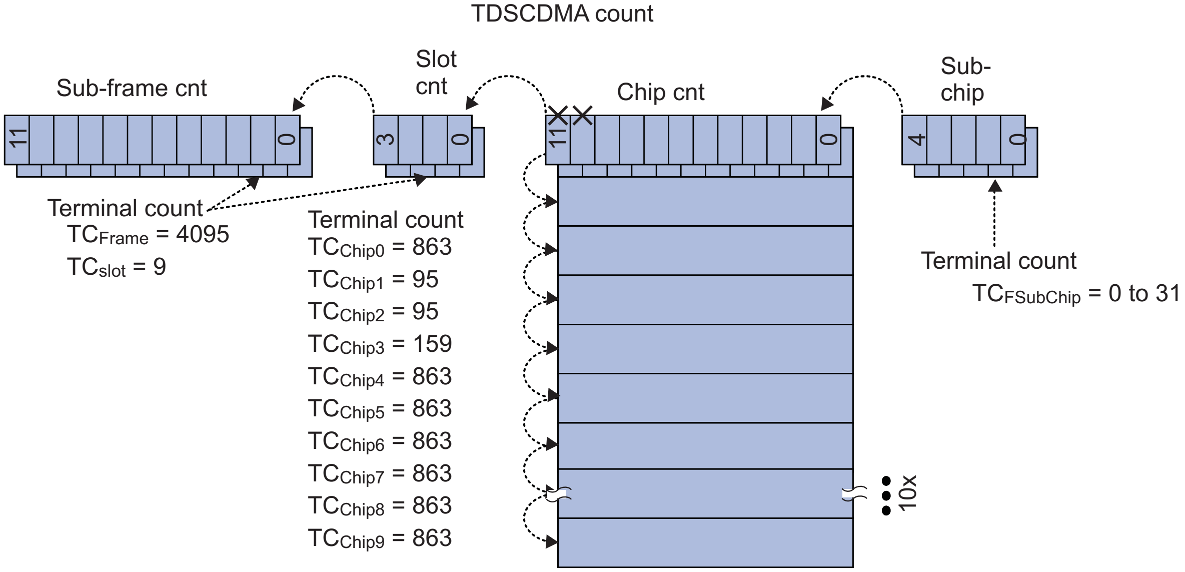


Figure 34 shows the timer configuration for non-RP1 interface.

Figure 34. Timer Configuration for Non-RP1 Interface



For non-RP1 interface, there is flexibility to program different chip TC values in the circular buffer. The corresponding chip TC RAM wrap address should be programmed in the FSYNC system TC; in this case, it is 0x9.

3.1.3 FSYNC Error/Alarm Registers

The FSYNC module may trigger 14 error/alarm conditions and broadcast them to all the DSP cores. The module contains two dedicated wires, one is designated as *error* and other is designated as *alarm*. The particular error/alarm is set in the FSYNC module by programming the ERR_SET_MASK_0 and ERR_SET_MASK_1 registers. The values written to these registers is verified by reading the ERR_INT_MASK_0 and ERR_INT_MASK_1 registers. The status of enabled interrupts is read from the ERR_MASK_STAT_0 and ERR_MASK_STAT_1 registers. All the error/alarm interrupts that have occurred in the FSYNC mdoule are read in the ERR_INT_SRC_RAW register. The software also sets/clears the interrupt by programming the ERR_INT_SET and ERR_INT_CLEAR registers, which are mainly used for debug purposes. The interrupts are re-evaluated by programming the ERR_END_OF_INT register.

3.1.3.1 FSYNC Error Interrupt SRC RAW Register (ERR_INT_SRC_RAW)

The FSYNC error interrupt SRC RAW register (ERR_INT_SRC_RAW) shows the state of all interrupts irrespective of the interrupts enabled in the ERR_SET_MASK_0 and ERR_SET_MASK_1 registers. These values are read via the VBUS. The ERR_INT_SRC_ RAW register is shown in Figure 35 and described in Table 33.

Figure 35. FSYNC Error Interrupt SRC RAW Register (ERR_INT_SRC_RAW)

31-16
Reserved
R-0

15	14-0
Rsvd	FSYNC_ERR_INT_SRC_RAW_ERR
R-0	R-0

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

Table 33. FSYNC Error Interrupt SRC RAW Register (ERR_INT_SRC_RAW) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14-0	FSYNC_ERR_INT_SRC_RAW_ERR		FSYNC Error Shows the status of all the error/alarm conditions occurring in the FSYNC module; for example, if bit 0 = 1, then the SYSTEM_FRAME_FAIL condition has occurred. (For a list of error/alarm conditions, see Table 8.)
		0	Inactive
		1	Active

3.1.3.2 FSYNC Error Mask Status Register 0 (ERR_MASK_STAT_0)

The FSYNC error mask status register 0 (ERR_MASK_STAT_0) shows the masked source state of interrupts for the FSYNC_ERROR_ALARM[0] output. The state of interrupt is active only when the mask bit in the ERR_SET_MASK_0 register for a particular error/alarm is enabled and that particular error/alarm has occurred. The ERR_MASK_STAT_0 register is shown in Figure 36 and described in Table 34.

Figure 36. FSYNC Error Mask Status Register 0 (ERR_MASK_STAT_0)

31	16
Reserved	
R-0	

15	14	0
Rsvd	FSYNC_ERR_MASK_STAT_0_ERR	
R-0	R-0	

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

Table 34. FSYNC Error Mask Status Register 0 (ERR_MASK_STAT_0) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14-0	FSYNC_ERR_MASK_STAT_0_ERR		FSYNC Error Mask Status 0 Shows the masked source state of interrupts occurring in the FSYNC module; for example, if bit 0 = 1, then the SYSTEM_FRAME_FAIL condition has occurred. (For a list of error/alarm conditions, see Table 8.)

3.1.3.3 FSYNC Error Mask Status Register 1 (ERR_MASK_STAT_1)

The FSYNC error mask status register 1 (ERR_MASK_STAT_1) shows the masked source state of interrupts for the FSYNC_ERROR_ALARM[1] output. The state of interrupt is active only when the mask bit in the ERR_SET_MASK_0 register for a particular error/alarm is enabled and that particular error/alarm has occurred. The ERR_MASK_STAT_1 register is shown in Figure 37 and described in Table 35.

Figure 37. FSYNC Error Mask Status Register 1 (ERR_MASK_STAT_1)

31	16
Reserved	
R-0	

15	14	0
Rsvd	FSYNC_ERR_MASK_STAT_1_ERR	
R-0	R-0	

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

Table 35. FSYNC Error Mask Status Register 1 (ERR_MASK_STAT_1) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14-0	FSYNC_ERR_MASK_STAT_1_ERR		FSYNC Error Mask Status 1 Shows the masked source state of interrupts occurring in the FSYNC module; for example, if bit 0 = 1, then the SYSTEM_FRAME_FAIL condition has occurred. (For a list of error/alarm conditions, see Table 8.)

3.1.3.4 FSYNC Error Set Mask 0 Register (ERR_SET_MASK_0)

TheFSYNC error set mask 0 register (ERR_SET_MASK_0) sets/enables the interrupts for the FSYNC_ERROR_ALARM[0] output. The ERR_SET_MASK_0 register is shown in Figure 38 and described in Table 36.

Figure 38. FSYNC Error Set Mask 0 Register (ERR_SET_MASK_0)

31	16
Reserved	
R-0	

15	14	0
Rsvd	ERR_EVENT_MASK_0	
R-0	W	

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

Table 36. FSYNC Error Set Mask 0 Register (ERR_SET_MASK_0) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14-0	ERR_EVENT_MASK_0		Error Event Mask 0 Enables the interrupts. The FSYNC_ERROR_ALARM[0] output signal goes high only if a particular interrupt is enabled and that interrupt has occurred in the FSYNC module; for eample, if bit 0 = 1, then the SYSTEM_FRAME_FAIL condition is enabled. (For a list of error/alarm conditions, see Table 8.)

3.1.3.5 FSYNC Error Set Mask 1 Register (ERR_SET_MASK_1)

The FSYNC error set mask 1 register (ERR_SET_MASK_1) sets/enables the interrupts for the FSYNC_ERROR_ALARM[1] output. The ERR_SET_MASK_1 register is shown in Figure 39 and described in Table 37.

Figure 39. FSYNC Error Set Mask 1 Register (ERR_SET_MASK_1)

31 16
Reserved
R-0

15	14 0
Rsvd	ERR_EVENT_MASK_1
R-0	W

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

Table 37. FSYNC Error Set Mask 1 Register (ERR_SET_MASK_1) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14-0	ERR_EVENT_MASK_1		Error Event Mask 1 Enables the interrupts. The FSYNC_ERROR_ALARM[1] output signal goes high only if a particular interrupt is enabled and that interrupt has occurred in the FSYNC module; for eample, if bit 0 = 1, then the SYSTEM_FRAME_FAIL condition is enabled. (For a list of error/alarm conditions, see Table 8.)

3.1.3.6 FSYNC Error Clear Mask 0 Register (ERR_CLEAR_MASK_0)

The FSYNC error clear mask 0 register (ERR_CLEAR_MASK_0) clears the interrupts for the FSYNC_ERROR_ALARM[0] output. The interrupt enable bits, which are set in the ERR_SET_MASK_0 register, are cleared by using this register. The ERR_CLEAR_MASK_0 register is shown in Figure 40 and described in Table 38.

Figure 40. FSYNC Error Clear Mask 0 Register (ERR_CLEAR_MASK_0)

31 16
Reserved
R-0

15	14 0
Rsvd	ERR_EVENT_MASK_0
R-0	W

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

Table 38. FSYNC Error Clear Mask 0 Register (ERR_CLEAR_MASK_0) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14-0	ERR_EVENT_MASK_0		Error Event Mask 0 Clears the interrupts; for example, if bit 0 = 1, then the SYSTEM_FRAME_FAIL condition is disabled. (For a list of error/alarm conditions, see Table 8.)

3.1.3.7 FSYNC Error Clear Mask 1 Register (ERR_CLEAR_MASK_1)

The FSYNC error clear mask 1 register (ERR_CLEAR_MASK_1) clears the interrupts for the FSYNC_ERROR_ALARM[1] output. The interrupt enable bits, which are set in the ERR_SET_MASK_1 register, are cleared by using this register. The ERR_CLEAR_MASK_1 register is shown in Figure 41 and described in Table 39.

Figure 41. FSYNC Error Clear Mask 1 Register (ERR_CLEAR_MASK_1)

31	16
Reserved	
R-0	

15	14	0
Rsvd	ERR_EVENT_MASK_1	
R-0	W	

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

Table 39. FSYNC Error Clear Mask 1 Register (ERR_CLEAR_MASK_1) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14-0	ERR_EVENT_MASK_1		Error Event Mask 1 Clears the interrupts; for example, if bit 0 = 1, then the SYSTEM_FRAME_FAIL condition is disabled. (For a list of error/alarm conditions, see Table 8.)

3.1.3.8 FSYNC Error Interrupt Mask 0 Register (ERR_INT_MASK_0)

The FSYNC error interrupt mask 0 register (ERR_INT_MASK_0) reads the state of the interrupt sources that are enabled. It reflects the bits programmed in the ERR_SET_MASK_0 register. The ERR_INT_MASK_0 register is shown in Figure 42 and described in Table 40.

Figure 42. FSYNC Error Interrupt Mask 0 Register (ERR_INT_MASK_0)

31 - 16
Reserved
R-0

15	14 - 0
Rsvd	FSYNC_ERR_INT_MASK_0_ERR
R-0	R-0

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

Table 40. FSYNC Error Interrupt Mask 0 Register (ERR_INT_MASK_0) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14-0	FSYNC_ERR_INT_MASK_0_ERR		Error Interrupt Mask 0 Error Reflects the mirror image of the ERR_SET_MASK_0 register.
		0	Disabled
		1	Enabled

3.1.3.9 FSYNC Error Interrupt Mask 1 Register (ERR_INT_MASK_1)

The FSYNC error interrupt mask 1 register (ERR_INT_MASK_1) reads the state of the interrupt sources that are enabled. It reflects the bits programmed in the ERR_SET_MASK_1 register. The ERR_INT_MASK_1 register is shown in Figure 43 and described in Table 41.

Figure 43. FSYNC Error Interrupt Mask 1Register (ERR_INT_MASK_1)

31	16
Reserved	
R-0	

15	14	0
Rsvd	FSYNC_ERR_INT_MASK_1_ERR	
R-0	R-0	

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

Table 41. FSYNC Error Interrupt Mask 1 Register (ERR_INT_MASK_1) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14-0	FSYNC_ERR_INT_MASK_1_ERR		Error Interrupt Mask 1 Error Reflects the mirror image of the ERR_SET_MASK_1 register.
		0	Disabled
		1	Enabled

3.1.3.10 FSYNC Error Interrupt Set Register (ERR_INT_SET)

The FSYNC error interrupt set register (ERR_INT_SET) is mainly used for debugging purpose. The interrupts can be set by software. The ERR_INT_SET register is shown in Figure 44 and described in Table 42.

Figure 44. FSYNC Error Interrupt Set Register (ERR_INT_SET)

31–16
Reserved
R-0

15	14–0
Rsvd	SET_ERR_EVENT
R-0	W

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

Table 42. FSYNC Error Interrupt Set Register (ERR_INT_SET) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14-0	SET_ERR_EVENT		Set Error Event Sets the error/alarm conditions. (For a list of error/alarm conditions, see Table 8.)
		0	No Change
		1	Set

3.1.3.11 FSYNC Error Interrupt Clear Register (ERR_INT_CLEAR)

The FSYNC error interrupt clear register (ERR_INT_CLEAR) is mainly used for debugging purpose. The interrupts, which are set by programming the ERR_INT_SET register, are cleared by programming the corresponding bit in this register. The ERR_INT_CLEAR register is shown in Figure 45 and described in Table 43.

Figure 45. FSYNC Error Interrupt Clear Register (ERR_INT_CLEAR)

31	16
Reserved	
R-0	

15	14	0
Rsvd	CLR_ERR_EVENT	
R-0	W	

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

Table 43. FSYNC Error Interrupt Clear Register (ERR_INT_CLEAR) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14-0	CLR_ERR_EVENT		Clear Error Event Clears the error/alarm conditions set by the ERR_INT_SET register. (For a list of error/alarm conditions, see Table 8.)
		0	No Change
		1	Clear

3.1.3.12 FSYNC Error End-of-Initialization Register (ERR_END_OF_INT)

The FSYNC error end-of-initialization register (ERR_END_OF_INT) is mainly used to re-evaluate the FSYNC_ERR_ALARM[0] or FSYNC_ERR_ALARM[1] interrupt lines.The condition to be re-evaluated is, the ERR_SET_MASK_0/ERR_SET_MASK_1 of the particular interrupts should be enabled and the error/alarm condition should occur. After setting the bits the register it can be read back to check whether the fs_err_alarm[0]/fs_err_alarm[1] is enabled. The ERR_END_OF_INT register is shown in Figure 46 and described in Table 44.

Figure 46. FSYNC Error End-of-Initialization Register (ERR_END_OF_INT)

31–16
Reserved
R-0

15–2	1–0
Reserved	ERR_ALARM_ INDEX
R-0	R/W-0

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

Table 44. FSYNC Error End-of-Initialization Register (ERR_END_OF_INT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1-0	ERR_ALARM_INDEX		Re-evaluation of error/alarm can be done by programming this bit field.
			Write:
		00	Select FSYNC_ERR_ALARM[0]
		01	Select FSYNC_ERR_ALARM[1]
			Read:
			Bit 0=1: FSYNC_ERR_ALARM[0] is enabled
			Bit 1=1: FSYNC_ERR_ALARM[1] is enabled.

3.1.4 FSYNC Timer Control Registers

The initial bit value after reset for all registers is 0, unless otherwise noted. The base address is 0XBBBB_BNNN where NNN = the hex address specified in Table 45. The event registers 0 to 9 and 18 to 29 are mask-based event generators and the event registers 10 to 17 are counter-based event generators.

Table 45. FSYNC Timer Control Register

Offset Address	Acronym	Description
380	EVT_FORCE	FSYNC Force Event Register
300	EGM_COMPARE [0]	FSYNC Trigger Value Register for Event0 [Mask-Based Evnt Gen]
280	EGM_MASK [0]	FSYNC Trigger Mask Register for Event0 [Mask-Based Evnt Gen]
200	EGM_CTRL [0]	FSYNC Event Generation Control Register for Event0[Mask-Based Evnt Gen]
304	EGM_COMPARE [1]	FSYNC Trigger Value Register for Event1 [Mask-Based Evnt Gen]
284	EGM_MASK [1]	FSYNC Trigger Mask Register for Event1 [Mask-Based Evnt Gen]
204	EGM_CTRL [1]	FSYNC Event Generation Control Register Event1 [Mask-Based Evnt Gen]
308	EGM_COMPARE [2]	FSYNC Trigger Value Register for Event2 [Mask-Based Evnt Gen]
288	EGM_MASK [2]	FSYNC Trigger Mask Register for Event2 [Mask-Based Evnt Gen]
208	EGM_CTRL [2]	FSYNC Event Generation Control Register Event2 [Mask-Based Evnt Gen]
30C	EGM_COMPARE [3]	FSYNC Trigger Value Register for Event3 [Mask-Based Evnt Gen]
28C	EGM_MASK [3]	FSYNC Trigger Mask Register for Event3 [Mask-Based Evnt Gen]
20C	EGM_CTRL [3]	FSYNC Event Generation Control Register Event3 [Mask-Based Evnt Gen]
310	EGM_COMPARE [4]	FSYNC Trigger Value Register for Event4 [Mask-Based Evnt Gen]
290	EGM_MASK [4]	FSYNC Trigger Mask Register for Event4 [Mask-Based Evnt Gen]
210	EGM_CTRL [4]	FSYNC Event Generation Control Register Event4 [Mask-Based Evnt Gen]
314	EGM_COMPARE [5]	FSYNC Trigger Value Register for Event5 [Mask-Based Evnt Gen]
294	EGM_MASK [5]	FSYNC Trigger Mask Register for Event5 [Mask-Based Evnt Gen]
214	EGM_CTRL [5]	FSYNC Event Generation Control Register Event5 [Mask-Based Evnt Gen]
318	EGM_COMPARE [6]	FSYNC Trigger Value Register for Event6 [Mask-Based Evnt Gen]
298	EGM_MASK [6]	FSYNC Trigger Mask Register for Event6 [Mask-Based Evnt Gen]
218	EGM_CTRL [6]	FSYNC Event Generation Control Register Event6 [Mask-Based Evnt Gen]
31C	EGM_COMPARE [7]	FSYNC Trigger Value Register for Event7 [Mask-Based Evnt Gen]
29C	EGM_MASK [7]	FSYNC Trigger Mask Register for Event7 [Mask-Based Evnt Gen]
21C	EGM_CTRL [7]	FSYNC Event Generation Control Register Event7 [Mask-Based Evnt Gen]
320	EGM_COMPARE [8]	FSYNC Trigger Value Register for Event8 [Mask-Based Evnt Gen]
2A0	EGM_MASK [8]	FSYNC Trigger Mask Register for Event 8[Mask-Based Evnt Gen]
220	EGM_CTRL [8]	FSYNC Event Generation Control Register Event8 [Mask-Based Evnt Gen]
324	EGM_COMPARE [9]	FSYNC Trigger Value Register for Event9 [Mask-Based Evnt Gen]
2A4	EGM_MASK [9]	FSYNC Trigger Mask Register for Event9 [Mask-Based Evnt Gen]
224	EGM_CTRL [9]	FSYNC Event Generation Control Register Event9 [Mask-Based Evnt Gen]
328	EGM_COMPARE [10]	FSYNC Trigger Value Register for Event18 [Mask-Based Evnt Gen]
2A8	EGM_MASK [10]	FSYNC Trigger Mask Register for Event18 [Mask-Based Evnt Gen]
228	EGM_CTRL [10]	FSYNC Event Generation Control Register for Event18 [Mask-Based Evnt Gen]
32C	EGM_COMPARE [11]	FSYNC Trigger Value Register for Event19 [Mask-Based Evnt Gen]
2AC	EGM_MASK [11]	FSYNC Trigger Mask Register for Event19 [Mask-Based Evnt Gen]
22C	EGM_CTRL [11]	FSYNC Event Generation Control Register for Event19 [Mask-Based Evnt Gen]
330	EGM_COMPARE [12]	FSYNC Trigger Value Register for Event20 [Mask-Based Evnt Gen]
2B0	EGM_MASK [12]	FSYNC Trigger Mask Register for Event20 [Mask-Based Evnt Gen]
230	EGM_CTRL [12]	FSYNC Event Generation Control Register for Event20 [Mask-Based Evnt Gen]

Table 45. FSYNC Timer Control Register (continued)

Offset Address	Acronym	Description
334	EGM_COMPARE [13]	FSYNC Trigger Value Register for Event21 [Mask-Based Evnt Gen]`
2B4	EGM_MASK [13]	FSYNC Trigger Mask Register for Event21 [Mask-Based Evnt Gen]
234	EGM_CTRL [13]	FSYNC Event Generation Control Register for Event21 [Mask-Based Evnt Gen]
338	EGM_COMPARE [14]	FSYNC Trigger Value Register for Event22 [Mask-Based Evnt Gen]
2B8	EGM_MASK [14]	FSYNC Trigger Mask Register for Event22 [Mask-Based Evnt Gen]
238	EGM_CTRL [14]	FSYNC Event Generation Control Register for Event22 [Mask-Based Evnt Gen]
33C	EGM_COMPARE [15]	FSYNC Trigger Value Register for Event23 [Mask-Based Evnt Gen]
2BC	EGM_MASK [15]	FSYNC Trigger Mask Register for Event23 [Mask-Based Evnt Gen]
23C	EGM_CTRL [15]	FSYNC Event Generation Control Register for Event23 [Mask-Based Evnt Gen]
340	EGM_COMPARE [16]	FSYNC Trigger Value Register for Event24 [Mask-Based Evnt Gen]
2C0	EGM_MASK [16]	FSYNC Trigger Mask Register for Event24 [Mask-Based Evnt Gen]
240	EGM_CTRL [16]	FSYNC Event Generation Control Register for Event24 [Mask-Based Evnt Gen]
344	EGM_COMPARE [17]	FSYNC Trigger Value Register for Event25 [Mask-Based Evnt Gen]
2C4	EGM_MASK [17]	FSYNC Trigger Mask Register for Event25 [Mask-Based Evnt Gen]
244	EGM_CTRL [18]	FSYNC Event Generation Control Register for Event25 [Mask-Based Evnt Gen]
348	EGM_COMPARE [19]	FSYNC Trigger Value Register for Event26 [Mask-Based Evnt Gen]
2C8	EGM_MASK [19]	FSYNC Trigger Mask Register for Event26 [Mask-Based Evnt Gen]
248	EGM_CTRL [19]	FSYNC Event Generation Control Register for Event26 [Mask-Based Evnt Gen]
34C	EGM_COMPARE [20]	FSYNC Trigger Value Register for Event27 [Mask-Based Evnt Gen]
2CC	EGM_MASK [20]	FSYNC Trigger Mask Register for Event27 [Mask-Based Evnt Gen]
24C	EGM_CTRL [20]	FSYNC Event Generation Control Register for Event27 [Mask-Based Evnt Gen]
350	EGM_COMPARE [21]	FSYNC Trigger Value Register for Event28 [Mask-Based Evnt Gen]
2D0	EGM_MASK [21]	FSYNC Trigger Mask Register for Event2 8[Mask-Based Evnt Gen]
250	EGM_CTRL [21]	FSYNC Event Generation Control Register for Event28 [Mask-Based Evnt Gen]
354	EGM_COMPARE [22]	FSYNC Trigger Value Register for Event29 [Mask-Based Evnt Gen]
2D4	EGM_MASK [22]	FSYNC Trigger Mask Register for Event29 [Mask-Based Evnt Gen]
254	EGM_CTRL [22]	FSYNC Event Generation Control Register for Event29 [Mask-Based Evnt Gen]
358	EGC_COUNTER [0]	FSYNC Event Generation Trigger Value Register for event10 [Counter Based]
258	EGC_CTRL [0]	FSYNC Event Generation Control Register for event10 [Counter Based]
35C	EGC_COUNTER [1]	FSYNC Event Generation Trigger Value Register for event11 [Counter Based]
25C	EGC_CTRL [1]	FSYNC Event Generation Control Register for event11 [Counter Based]
360	EGC_COUNTER [2]	FSYNC Event Generation Trigger Value Register for event12 [Counter Based]
260	EGC_CTRL [2]	FSYNC Event Generation Control Register for event12 [Counter Based]
364	EGC_COUNTER [3]	FSYNC Event Generation Trigger Value Register for event13 [Counter Based]
264	EGC_CTRL [3]	FSYNC Event Generation Control Register for event13 [Counter Based]
368	EGC_COUNTER [4]	FSYNC Event Generation Trigger Value Register for event14 [Counter Based]
268	EGC_CTRL [4]	FSYNC Event Generation Control Register for event14 [Counter Based]
36C	EGC_COUNTER [5]	FSYNC Event Generation Trigger Value Register for event15 [Counter Based]
26C	EGC_CTRL [5]	FSYNC Event Generation Control Register for event15 [Counter Based]
370	EGC_COUNTER [6]	FSYNC Event Generation Trigger Value Register for event16 [Counter Based]
270	EGC_CTRL [6]	FSYNC Event Generation Control Register for event16 [Counter Based]

Table 45. FSYNC Timer Control Register (continued)

Offset Address	Acronym	Description
374	EGC_COUNTER [6]	FSYNC Event Generation Trigger Value Register for event17 [Counter Based]
274	EGC_CTRL [6]	FSYNC Event Generation Control Register for event17 [Counter Based]

3.1.4.1 EVNT_FORCE Register

The FSYNC module has the feature of forcing events by programming EVNT_FORCE register. This register is a write-only register where the user can generate all 30 events by programming the corresponding bits. After programming this register, the FSYNC_CTL2_ARM_TIMER bit of the CTL1 register should be enabled. The EVNT_FORCE register is shown in Figure 47 and described in Table 46. This feature is only used in system debug and not used in normal operations.

Figure 47. EVNT_FORCE Register

31 30	29 0
Rsvd	EVENT_GEN
R-0	W

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

Table 46. EVNT_FORCE Register Field Descriptions

Bit	Field	Value	Description
31-30	Reserved		Reserved
29-0	EVENT_GEN		Programming bit fields 29-0 generates events 0 to 29. Bit 0 corresponds to event 0, bit 1 corresponds to event 1, and so on.
		0	No events
		1	Generates events

3.1.4.2 EGM_COMPARE [0] Register

The trigger value register is used as a comparator to generate events. The compared value can be FRAME_VALUE, SLOT_VALUE, CHIP_TERMINAL_COUNT_INDEX_VALUE, CHIP_VALUE, and SAMPLE_VALUE. The FSYNC module generates event, once the offset condition is met and the trigger value is met. The event generation is periodical until the timer stops. The EGM_COMPARE [0] register is shown in Figure 48 and described in Table 47.

Figure 48. EGM_COMPARE [0] Register

31 27	26 25	24
Reserved	FRAME_VALUE	SLOT_VALUE
R-0	R/W-0	R/W-0

23 21	20 17	16
SLOT_VALUE	CHIP_TERMINAL_COUNT_INDEX_VALUE	CHIP_VALUE
R/W-0	R/W-0	R/W-0

15 5	4 0
CHIP_VALUE	SAMPLE_VALUE
R/W-0	R/W-0

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

Table 47. EGM_COMPARE [0] Register Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved
26-25	FRAME_VALUE		The frame value in the counter is compared with this bit field to generate event 0
24-21	SLOT_VALUE		The slot value in the counter is compared with this bit field to generate event 0
20-17	CHIP_TERMINAL_COUNT_INDEX_VALUE		The chip TC state value in the counter is compared with this bit field to generate event 0
16-5	CHIP_VALUE		The chip value in the counter is compared with this bit field to generate event 0
4-0	SAMPLE_VALUE		The sub-chip value in the counter is compared with this bit field to generate event 0

3.1.4.3 EGM_MASK [0] Register

The trigger mask register is used to validate the bit field programmed in the EGM_COMPARE [0] register.The mask value is ANDed with the trigger value and the resultant field is used for the comparison for generating events. The frame, slot, chip TC state, chip, and sub-chip can be masked. Here, masking means programming the bit field to 1. The EGM_MASK [0] register is shown in Figure 49 and described in Table 48.

Figure 49. EGM_MASK [0] Register

31 27	26 25	24
Reserved	FRAME_MASK	SLOT_MASK
R-0	R/W-0	R/W-0

23 21	20 17	16
SLOT_MASK	CHIP_TERMINAL_COUNT_INDEX_MASK	CHIP_MASK
R/W-0	R/W-0	R/W-0

15 5	4 0
CHIP_MASK	SAMPLE_MASK
R/W-0	R/W-0

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

Table 48. EGM_MASK [0] Register Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved
26-25	FRAME_MASK		The masked frame value, which is ANDed with the frame value programmed in the EVGEN_TRIG_VALUE_REG0 register to get the comparison value to generate event 0. Masking: Setting the field to 1.
24-21	SLOT_MASK		The masked slot value, which is ANDed with the slot value programmed in the EVGEN_TRIG_VALUE_REG0 register to get the comparison value to generate event 0. Masking: Setting the field to 1.
20-17	CHIP_TERMINAL_COUNT_INDEX_MASK		The masked chip TC state value, which is ANDed with the chip TC state value programmed in the EVGEN_TRIG_VALUE_REG0 register to get the comparison value to generate event 0. Masking: Setting the field to 1.
16-5	CHIP_MASK		The masked chip value, which is ANDed with the chip value programmed in the EVGEN_TRIG_VALUE_REG0 register to get the comparison value to generate event 0. Masking: Setting the field to 1.
4-0	SAMPLE_MASK		The masked sub-chip value, which is ANDed with the sub-chip value programmed in the EVGEN_TRIG_VALUE_REG0 register to get the comparison value to generate event 0. Masking: Setting the field to 1.

3.1.4.4 EGM_CTRL [0] Register

The event generation control register is used to select the timer for generating events. Bit field 26 is used for the selection. The default selection is system timer. This register also has the control of enabling and disabling the event generation. The offset values are programmed in this register. The EGM_CTRL [0] register is shown in Figure 50 and described in Table 49.

Figure 50. EGM_CTRL [0] Register

31 27	26	25	24
Reserved	TIMER_USED	FSYNC_EGM_CTRL_ENABLE	SLOT_OFFSET
R-0	R/W-0		R/W-0

23 21	20 17	16
SLOT_OFFSET	CHIP_TERMINAL_COUNT_INDEX	CHIP_OFFSET
R/W-0	R/W-0	R/W-0

15 5	4 0
CHIP_OFFSET	SAMPLE_OFFSET
R/W-0	R/W-0

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

Table 49. EGM_CTRL [0] Register Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved
26	TIMER_USED		Timer source to generate events
		0	System timer
		1	RP3 timer
25	FSYNC_EGM_CTRL_ENABLE		Enable/disable timer to generate events
		0	Enable event
		1	Disable event
24-21	SLOT_OFFSET		Slot offset value
20-17	CHIP_TERMINAL_COUNT_INDEX		Chip TC state offset value
16-5	CHIP_OFFSET		Chip offset value
4-0	SAMPLE_OFFSET		Sub-chip offset value

3.1.4.5 EGC_COUNTER [0] Register

The EGC_COUNTER [0] register is used to generate counter-based events. These registers are used to generate events in the sub-chip level. The trigger value programmed in this register is decremented for every clock once the offset condition is met. It generates the event once the counter reaches 0 and is automatically reloaded. The EGC_COUNTER [0] register is shown in Figure 51 and described in Table 50.

Figure 51. EGC_COUNTER [0] Register

31 27	26 0
Reserved	EGC_TCOUNT
R-0	R/W-0

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

Table 50. EGC_COUNTER [0] Register Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved
26-0	EGC_TCOUNT		The counter value loaded is decremented with the clock once the offset condition is met. The minimum value to be programmed is 1.

3.1.4.6 EGC_CTRL [0] Register

The EGC_CTRL [0] register is used to select the timer for generating events. Bit field 26 is used for the selection. The default selection is system timer. This register also has the control of enabling and disabling the event generation. The offset values are programmed in this register. The EGC_CTRL [0] register is shown in Figure 52 and described in Table 51.

Figure 52. EGC_CTRL [0] Register

31 27	26	25	24
Reserved	TIMER_USED	FSYNC_EGC_CTRL_ENABLE	SLOT_OFFSET
R-0	R/W-0	R/W	W

23 21	20 17	16
SLOT_OFFSET	CHIP_TERMINAL_COUNT_INDEX	CHIP_OFFSET
W	R/W-0	R/W-0

15 5	4 0
CHIP_OFFSET	SAMPLE_OFFSET
R/W-0	R/W-0

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

Table 51. EGC_CTRL [0] Register Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved
26	TIMER_USED		Timer source to generate events
		0	System timer
		1	RP3 timer
25	FSYNC_EGC_CTRL_ENABLE		Enable/disable timer to generate events
		0	Enable event
		1	Disable event
24-21	SLOT_OFFSET		Slot offset value
20-17	CHIP_TERMINAL_COUNT_INDEX		Chip TC state offset value
16-5	CHIP_OFFSET		Chip offset value
4-0	SAMPLE_OFFSET		Sub-chip offset value

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI’s standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265