Functional Safety Manual for TMS320F28P55x Real-Time Microcontrollers

Functional Safety Information

Literature Number: SFFS779
DECEMBER 2024

Table of Contents

List of Figures

List of Tables

This page intentionally left blank.

Chapter 1
Introduction

CAUTION

The TMS320F28P55x is being offered as a Functional Safety Compliant Safety Element out of Context (SEooC) product. This implies that the TMS320F28P55x was developed in compliance with TI's ISO-9001/IATF-16949 compliant hardware product development process. Subsequently, this product was independently assessed to meet a systematic capability compliance of ASIL D (according to ISO-26262:2018) and SIL 3 (according to IEC-61508:2010), see the *Texas Instrument's Functional Safety Hardware Development Process*. As such, this safety manual is intended to be informative only to help explain how to use the features of theTMS320F28P55x device to assist the system designer in achieving a given ASIL or SIL level. System designers are responsible for evaluating this device in the context of the system and determining the system-level ASIL or SIL coverage achieved therein.

Note

This is only a draft of the TMS320F28P55x Functional Safety Manual and is subject to change between now and the final version. The certification this document references has not yet been completed.

This document is the Functional Safety Manual for the TMS320F28P55x MCU series from Texas Instruments which is part of the high performance C2000™ real-time microcontroller product line. The C2000 product line utilizes a common safety architecture that is implemented for multiple products in automotive and industrial applications.

The products supported by this document have been assessed to be meet a systematic capability compliance of ASIL-D (according to ISO 26262) and SIL-3 (according to IEC 61508). For more information, see the *Texas Instrument's Functional Safety Hardware Development Process*.

This Functional Safety Manual is part of the Functional Safety-Compliant design package to aid customers who are designing systems in compliance with ISO26262 or IEC61508 functional safety standards.

Table 1-1 shows a complete list of the products supported by this safety manual with silicon revision A. The device revision can be determined by the REVID field of the device identification registers.

Table 1-1. Products Supported by This Safety Manual

Orderable Devices	
Automotive Devices	**Industrial Devices**
F28P559SJ9PDTRQ1	F28P550SJ9PDTR
F28P559SJ9PDTQ1	F28P550SJ9PDT
F28P559SJ6PDTRQ1	F28P550SJ6PDTR
F28P559SG9PDTRQ1	F28P550SG9PDTR
F28P559SJ9PZRQ1	F28P550SJ9PZR
F28P559SJ9PZQ1	F28P550SJ9PZ
F28P559SJ6PZRQ1	F28P550SJ6PZR
F28P559SG9PZRQ1	F28P550SG9PZR

Table 1-1. Products Supported by This Safety Manual (continued)

Orderable Devices	
Automotive Devices	**Industrial Devices**
F28P559SG8PZRQ1	F28P550SG8PZR
F28P559SJ9PNARQ1	F28P550SJ9PNAR
F28P559SJ9PNAQ1	F28P550SJ9PNA
F28P559SJ6PNARQ1	F28P550SJ6PNAR
F28P559SG9PNARQ1	F28P550SG9PNAR
F28P559SJ9PMRQ1	F28P550SG8PNAR
F28P559SG8PNARQ1	F28P550SJ9PMR
F28P559SJ9PMQ1	F28P550SJ9PM
F28P559SJ6PMRQ1	F28P550SJ6PMR
F28P559SG9PMRQ1	F28P550SG9PMR
F28P559SG8PMRQ1	F28P550SG8PMR
	F28P550SJ9RSHR
	F28P550SJ6RSHR
	F28P550SG9RSHR
	F28P550SG8RSHR

This Functional Safety Manual provides information needed by system developers to assist in the creation of a safety critical system using a supported TMS320F28P55x MCU. This document contains:

- An overview of the component architecture
- An overview of the development process used to decrease the probability of systematic failures
- An overview of the functional safety architecture for management of random failures
- The details of architecture partitions and implemented functional safety mechanisms

The following information is documented in the *Detailed Safety Analysis Report (SAR)* section of the FMEDA, *for TMS320F28P55x C2000™ Safety Critical Microcontrollers*, which is only available under a Functional Safety NDA and is not repeated in this document:

- Failure rates (FIT) of the component
- Fault model used to estimate device failure rates to enable calculation of customized failure rates
- Functional safety metrics of the hardware component for targeted standards (specifically, IEC 61508:2010 and ISO 26262:2018)
- Quantitative functional safety analysis (also known as FMEDA, Failure Modes, Effects, and Diagnostics Analysis) with details of the different parts of the component, allowing for customized application of functional safety mechanisms
- Assumptions used in the calculation of functional safety metrics

The user of this document should have a general familiarity with the TMS320F28P55x product families. More information can be found at TI's *C2000 Real-time Microcontrollers* webpage.

This document is intended to be used in conjunction with the pertinent data sheets, technical reference manuals, and other documentation for the products being supplied.

For information which is beyond the scope of the listed deliverables, contact your TI sales representative or TI.com.

Trademarks

C2000™ is a trademark of Texas Instruments.
TÜV® is a registered trademark of TÜV SÜD.
All trademarks are the property of their respective owners.

This page intentionally left blank.

Chapter 2

TMS320F28P55x Product Safety Capability and Constraints

This section summarizes the TMS320F28P55x product safety capability. Each TMS320F28P55x product:

- Is offered as a functional Safety Element out of Context (SEooC)
- Was assessed to have met the relevant systematic capability compliance requirements of IEC 61508:2010 and ISO 26262:2018 and
  - Achieves systematic integrity of SIL 3 and ASIL D
- In addition, the device can meet hardware architectural metrics up to ASIL B and SIL 2 by implementing the proper safety concept (for example, *Reciprocal Comparison by Software*).
- Contains multiple features to support freedom from interference (FFI) for mixed-criticality of safety requirements assigned to the different sub-elements
- The TMS320F28P55x MCUs are Type B devices, as defined in IEC 61508-2:2010
- This device claims no hardware fault tolerance, (for example, no claims of HFT > 0), as defined in IEC 61508:2010
- Normally, the component functional safety manual must provide a list of product safety constraints for safety components developed according to many safety standards. For a simple component, or more complex components developed for a single application, this is a reasonable response; however, the TMS320F28P55x MCU product family is both a complex design and is not developed targeting a single, specific application. Therefore, a single set of product safety constraints cannot govern all viable uses of the product.

Chapter 3

TI Development Process for Management of Systematic Faults

For functional safety development, it is necessary to manage both systematic and random faults. Texas Instruments follows a new-product development process for all of its components which helps to decrease the probability of systematic failures. This new-product development process is described in Section 3.2. Components being designed for functional safety applications will additionally follow the requirements of TI's functional safety development process, which is described in Section 3.2.

3.1 TI New-Product Development Process

Texas Instruments has been developing components for automotive and industrial markets since 1996. Automotive markets have strong requirements regarding quality management and product reliability. The TI new-product development process features many elements necessary to manage systematic faults. Additionally, the documentation and reports for these components can be used to assist with compliance to a wide range of standards for customer's end applications including automotive and industrial systems (e.g ISO 26262-4:2018, IEC 61508-2:2010).

This component was developed using TI's new product development process which has been certified as compliant to ISO 9001 / IATF 16949 as assessed by Bureau Veritas (BV).

The standard development process breaks development into phases:

- Assess
- Plan
- Create
- Validate

3.2 TI Functional Safety Development Process

The TI functional safety development flow derives from ISO 26262:2018 and IEC 61508:2010 a set of requirements and methodologies to be applied to semiconductor development. This flow is combined with TI's standard new product development process to develop Functional Safety-Compliant components. The details of this functional safety development flow are described in the TI internal specification - Functional Safety Hardware.

Key elements of the TI functional safety-development flow are as follows:

- Assumptions on system-level design, functional safety concept, and requirements based on TI's experience with components in functional safety applications
- Qualitative and quantitative functional safety analysis techniques including analyses of silicon failure modes and application of functional safety mechanisms
- Base FIT rate estimation based on multiple industry standards and TI manufacturing data
- Documentation of functional safety work products during the component development
- Integration of lessons learned through multiple functional safety component developments, functional safety standard working groups, and the expertise of TI customers

Table 3-1 lists these functional safety development activities that are overlaid atop the standard development flow in Section 3.1.

For more information about which functional safety life-cycle activities TI performs, see Appendix A.

The customer facing work products derived from this Functional Safety-Compliant process are applicable to many other functional safety standards beyond ISO 26262:2018 and IEC 61508:2010.

Table 3-1. Functional Safety Activities Overlaid on Top of TI's Standard Development Process

Assess	Plan	Create	Validate	Sustain and End-of-Life
Determine if functional safety process execution is required	Define component target SIL/ASIL capability	Develop component level functional safety requirements	Validate functional safety design in silicon	Document any reported issues (as needed)
Nominate a functional safety manager	Generate functional safety plan	Include functional safety requirements in design specification	Characterize the functional safety design	Perform incident reporting of sustaining operations (as needed)
End of Phase Audit	Verify the functional safety plan	Verify the design specification	Qualify the functional safety design (per AEC-Q100)	Update work products (as needed)
	Initiate functional safety case	Start functional safety design	Finalize functional safety case	
	Analyze target applications to generate system-level functional safety assumptions	Perform qualitative analysis of design (failure mode analysis)	Perform assessment of project	
	End of Phase Audit	Verify the qualitative analysis	Release functional safety manual	
		Verify the functional safety design	Release functional safety analysis report	
		Perform quantitative analysis of design (FMEDA)	Release functional safety report	
		Verify the quantitative analysis	End of Phase Audit	
		Iterate functional safety design as necessary		
		End of Phase Audit		

Chapter 4

TMS320F28P55x Component Overview

The TMS320F28P55x is a powerful 32-bit floating-point microcontroller unit (MCU) designed for advanced closed-loop control in automotive and industrial applications.

The TMS320F28P55x real-time control subsystem is based on TI's 32-bit C28x DSP core, which provides 150 MIPS of signal-processing performance in each core for floating- or fixed-point code running from either on-chip flash or SRAM. The C28x CPU is further boosted by the trigonometric math unit (TMU) and VCRC (cyclical redundancy check) extended instruction sets, speeding up common algorithms that are essential to real-time control systems. Extended instruction sets enable IEEE single-precision 32-bit floating-point math. Users can refer to *Enhancing the Computational Performance of the C2000™ Microcontroller Family* to learn how the instruction set extensions can be employed to increase the performance of the MCU in many real-time applications.

The CLA is an independent 32-bit floating-point accelerator that runs at the same speed as the main C28x CPU, responding to peripheral triggers with minimum event latency and executing code concurrently with the main CPU.

To allow fast context switching from existing firmware to new firmware, hardware enhancements for live firmware update (LFU) exist on this device.

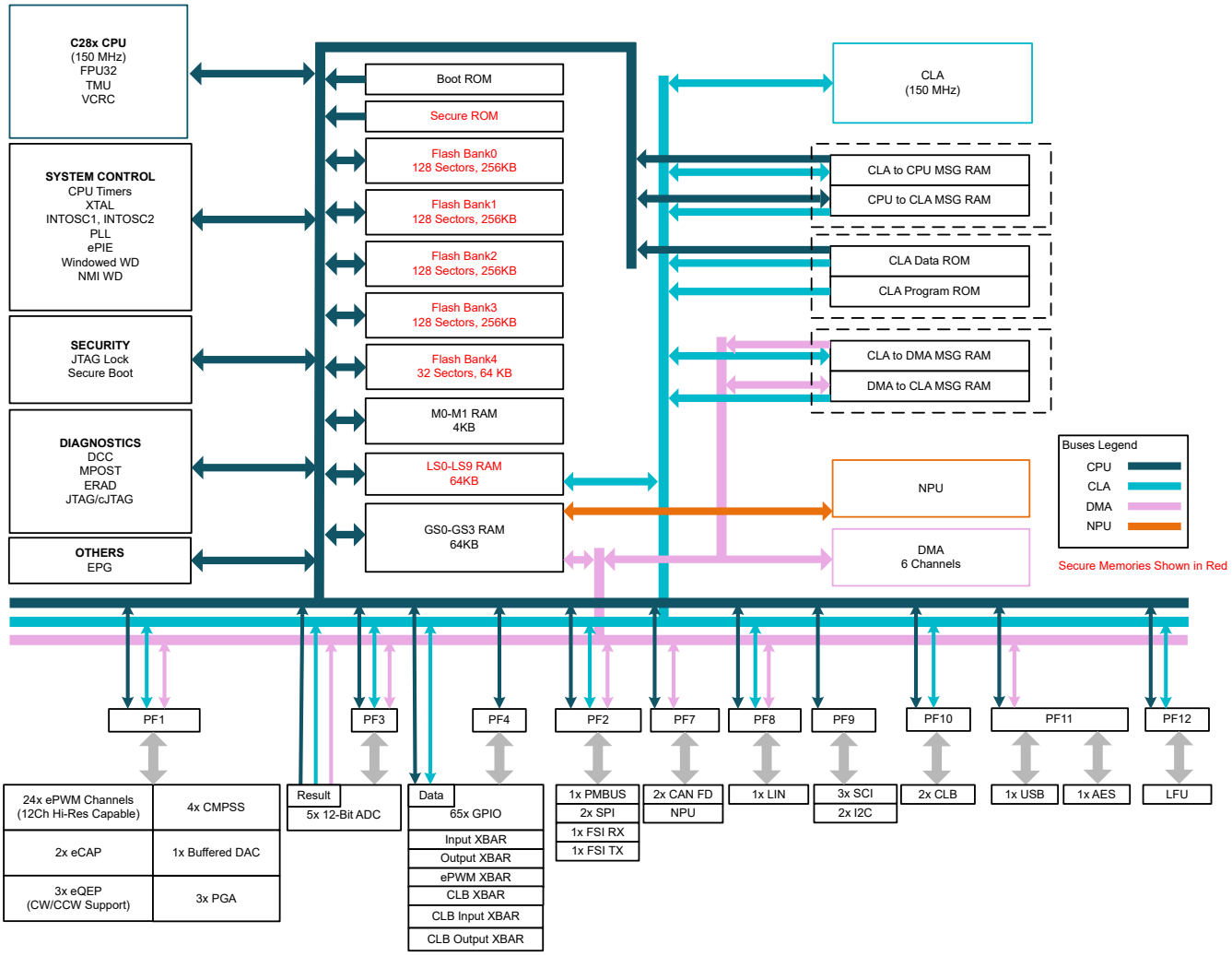

Figure 4-1. Functional Block Diagram of TMS320F28P55x Real-Time MCUs

High-performance analog blocks are tightly integrated with the processing and control units to provide excellent real-time signal chain performance. The analog-to-digital converter (ADC) has been enhanced with up to 40 analog channels. For safety-critical ADC conversions, a hardware redundancy checker has been added that provides the ability to compare ADC conversion results from multiple ADC modules for consistency without additional CPU cycles. The comparator subsystem (CMPSS), with windowed comparators, allows for protection of power stages when current limit conditions are exceeded or not met. Other analog and control peripherals include the digital-to-analog converter (DAC), pulse width modulation (PWM), enhanced capture (eCAP), enhanced quadrature encoder pulse (eQEP) and other peripherals. Peripherals such as controller area network (CAN) modules (ISO11898-1/CAN 2.0B-compliant) extend the connectivity of the C2000 MCUs.

The device configurations supported by this safety manual for TMS320F28P55xMCUs are outlined in the *TMS320F28P55x Real-Time Microcontrollers* data sheet. Not all variants are available in all packages or all temperature grades. To confirm availability, contact your local Texas Instruments sales and marketing representative.

The major safety features of the TMS320F28P55x are shown in Figure 4-2.

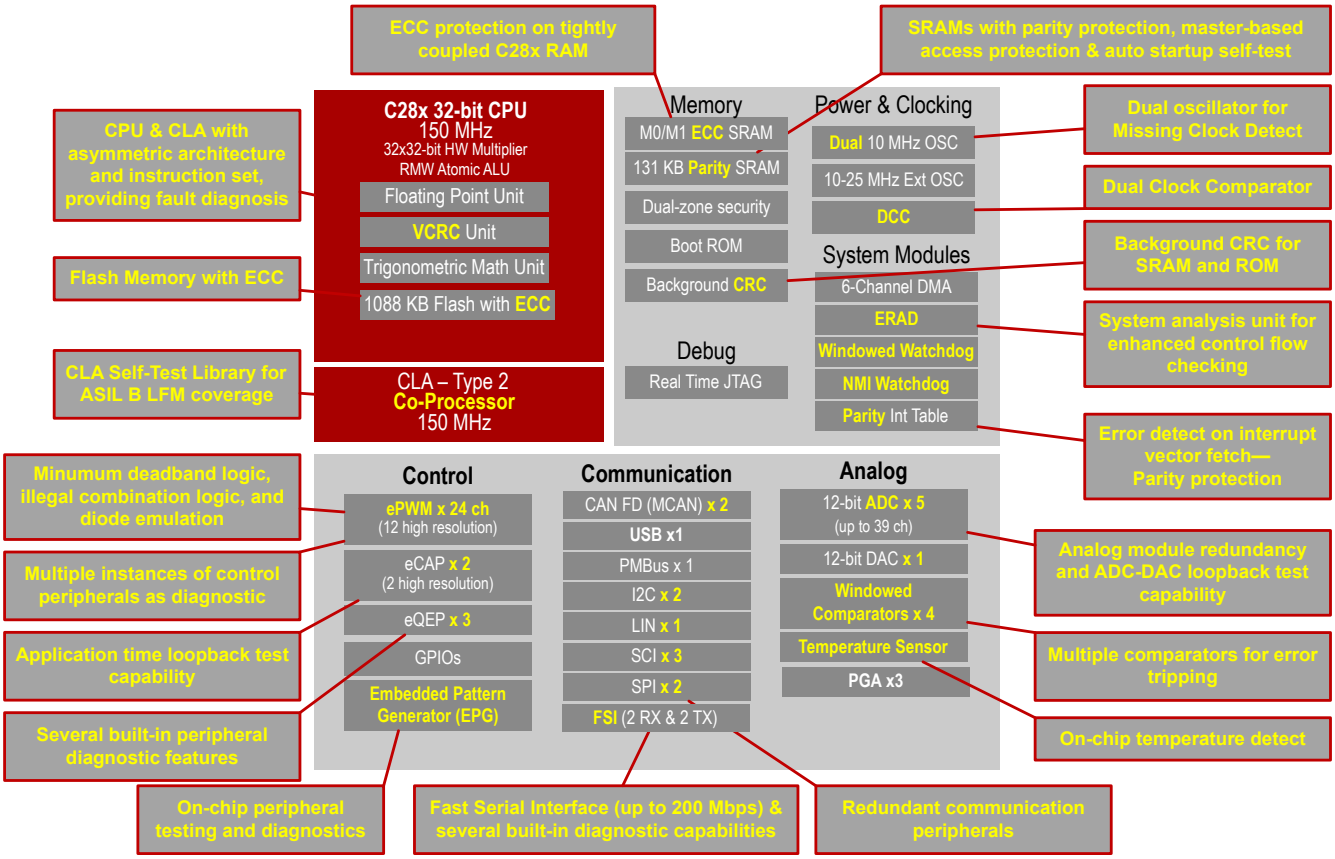

Figure 4-2. TMS320F28P55x Real-Time Microcontroller With Safety Features

4.1 Targeted Applications

The MCU is targeted at general-purpose functional safety applications. This is called Safety Element out of Context (SEooC) development according to ISO 26262-10. In this case, the development is done based on assumptions on the conditions of the semiconductor component usage, and then the assumptions are verified at the system-level. This method is also used to meet the related requirements of IEC 61508 at the semiconductor level. This section describes some of the target applications for this component, the component safety concept, and then describes the assumptions about the systems (also know as Assumptions of Use or AoU) that were made in performing the safety analysis.

Example target applications include, but are not limited to, the following:

- Servo drive control module
- Industrial and collaborative robot servo drive
- Mobile robot motor control
- CNC control
- Linear motor segment controller
- Central inverter
- String inverter
- DC/DC converter system
- Integrated high voltage (OBC and DC/DC)
- Onboard charger

4.2 Hardware Component Functional Safety Concept

To stay as general as possible, the safety concept assumes the MCU playing the role of a processing unit (or part of a processing unit) is connected to a remote controller (or controllers) by means of a communication bus, as shown in Figure 4-3. The communication bus is directly or indirectly connected to a sensor (or sensors) and actuator (or actuators).

IEC 61508:1 clause 8.2.12 defines a compliant item as any item (for example an element) on which a claim is being made with respect to the clauses of the IEC 61508 series. A system including a C2000 microcontroller can be used in a compliant item according to IEC61508, as indicated by Figure 4-3.

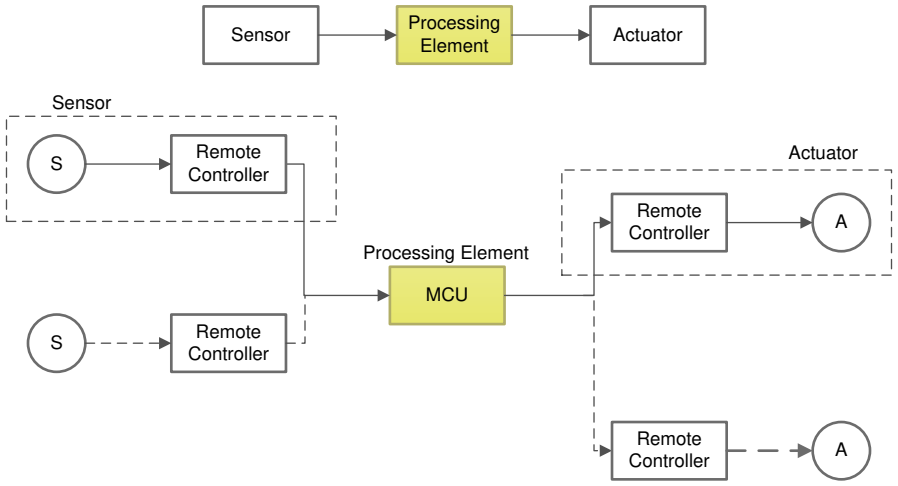

Figure 4-3. Definition of the C2000 MCU Used in a Compliant Item

4.2.1 VDA E-GAS Monitoring Concept

The standardized E-GAS monitoring concept [6] for engine management systems, generated by the German VDA working group *E-Gas-Arbeitskreis*, is an example of a well-trusted safety-architecture that can be used for applications other than engine management systems, provided the architecture fits the purpose of the new application in terms of diagnosis feasibility, environment constraints, time constraints, robustness, and so forth [7]. For more information, see Figure 4-4.

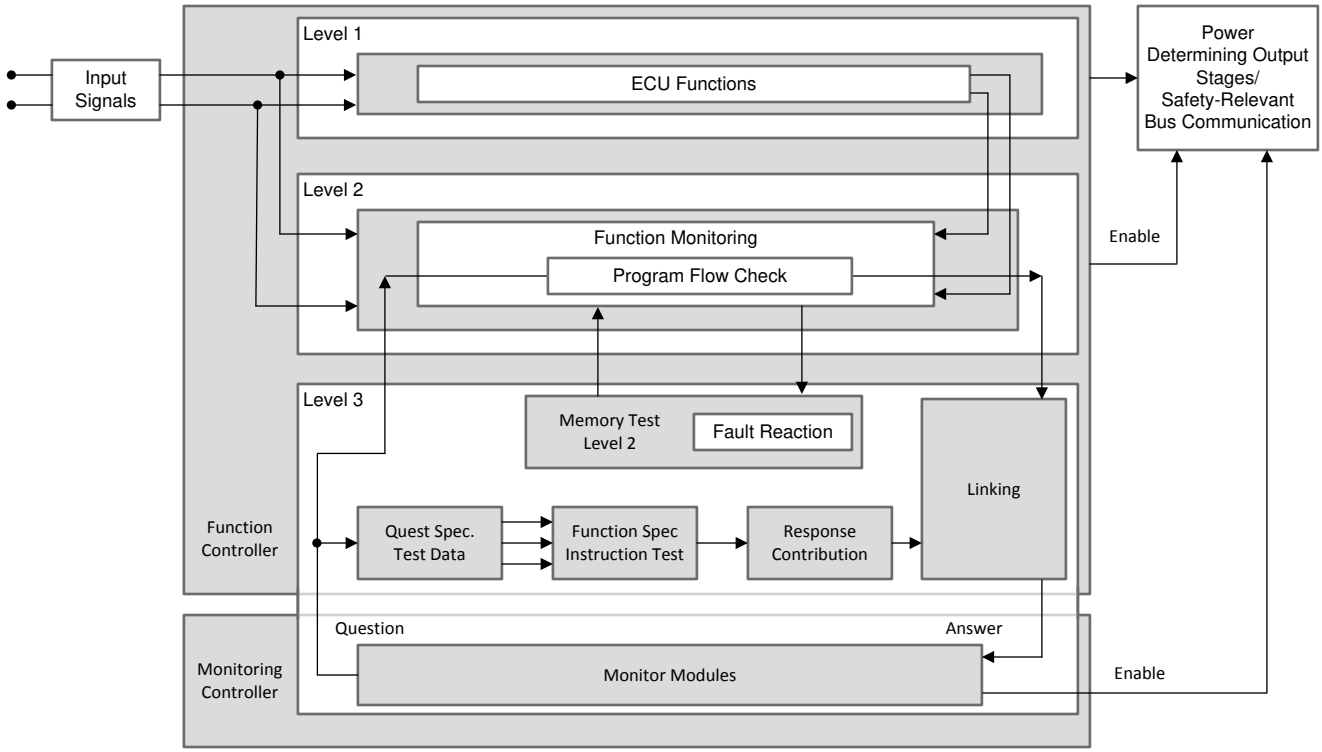

Figure 4-4. E-GAS System Overview From Standard

The TMS320F28P55x MCU device family supports heterogeneous asymmetric architecture and the functional safety features of the device lend to an E-GAS concept implementation at system-level, as indicated in Figure 4-5. In the first level (Level 1), the functions required for the system mission are computed. Second level (Level 2) checks the correct formation in first level based on a selected set of parameters. Third level (Level 3) implements an additional external monitoring element for correctly carrying out the mission in the first level and monitoring in the second level. The exact functional safety implementation and the modules used for realizing Level 1, Level 2, and the external monitoring device for realizing Level 3 are left to the system designer. Though Figure 4-5 indicates CLA implementing Level 1 and CPU(28x) implementing Level 2 of the EGAS monitoring concept, both processing units are capable of implementing either Level 1 or Level 2. The application can determine the partitioning based on the system requirements.

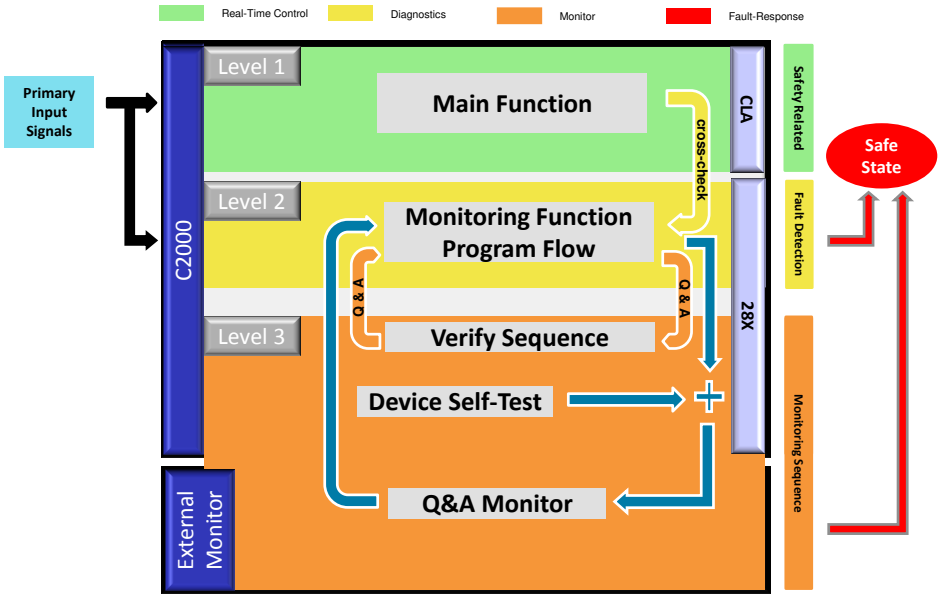

Figure 4-5. VDA E-Gas Monitoring Concept Applied to C2000 MCU

4.2.2 Fault Tolerant Time Interval (FTTI)

Various safety mechanisms in the devices are either always-on (see SRAM ECC, CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping , and so forth) or executed periodically (see VCU CRC Check of Static Memory Contents and so forth) by the application software. The time between the executions of online diagnostic tests by a safety mechanism is termed as diagnostic test interval (DTI). Once the fault is detected, depending on the fault reaction of the associated fault (for example, external system reaction to ERRORSTS pin assertion), the system enters the safe-state. The time-span in which a fault (or faults) can be present in a system before a hazardous event occurs is called fault tolerant time interval (FTTI) as defined in ISO26262. This is similar to process safety time (PST) defined in IEC61508. Figure 4-6 illustrates the relationship between DTI, fault reaction time, and FTTI.

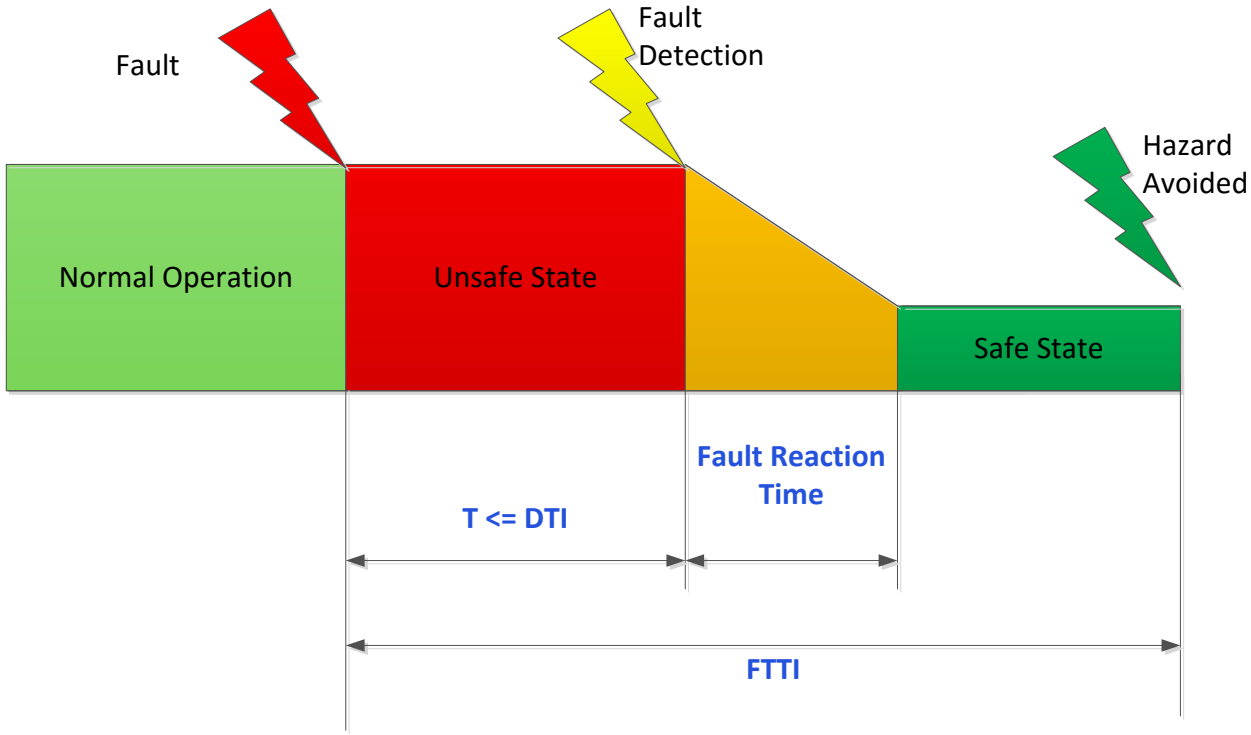

Figure 4-6. Relationship Between DTI, Fault Reaction Time, and FTTI

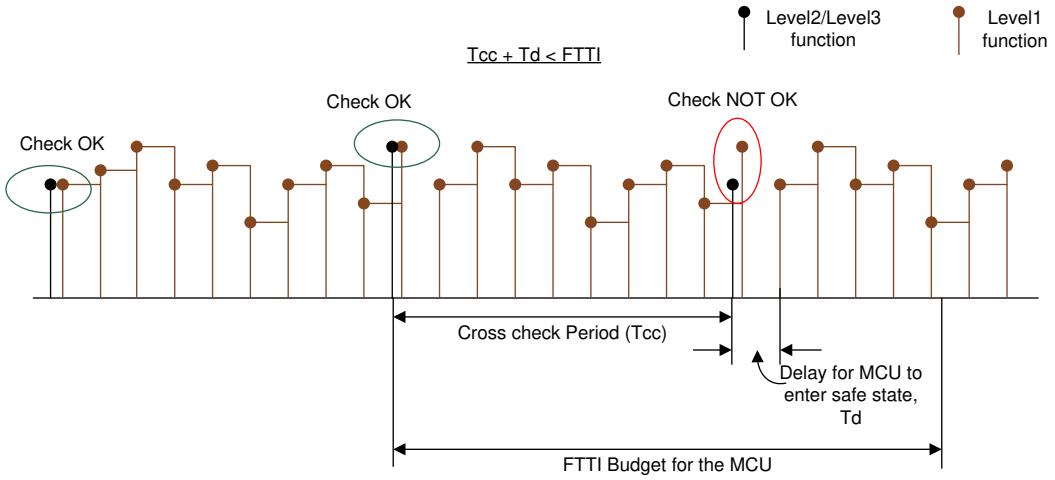

Figure 4-7. Illustration of FTTI

The frequency and extent of each of the Level 2 and Level 3 checks must be consistent with the fault tolerant time interval (FTTI). Figure 4-7 illustrates the frequency of the required checks. The checks must be such that single point faults of the microcontroller are detected and responded to, so that the TMS320F28P55x MCU enters a safe state within the FTTI budget. The microcontroller enters into one of the safe states on detection of a fault, as illustrated in Figure 4-8. An example of a diagnostic for single point faults is ECC/Parity for memories.

The proposed functional safety concept, subsequent functional safety features, and configurations explained in this document are for reference purpose only. The system and equipment designer or manufacturer is responsible for ensuring that the end systems (and any Texas Instruments' hardware or software components incorporated in the systems) meet all applicable safety, regulatory, and system-level performance requirements.

4.2.3 TMS320F28P55x MCU Safe State

Referring to Figure 4-8, the safe state of the TMS320F28P55x MCU is defined as one in which:

- The TMS320F28P55x MCU reset is asserted
- The power supply to the TMS320F28P55x MCU is disabled using an external supervisor as a result of a Level 3 check failure. In general, a power supply failure is not considered in detail in this analysis since TI assumes that the system-level functionality exists to manage this condition.
- The external system is informed using one of the I/O pins of the TMS320F28P55x MCU as a result of a Level 2 check failure (for example, ERRORSTS pin is asserted).
- The output of the TMS320F28P55x MCU driving the actuator is forced to inactive mode as a result of a Level 2 check failure (for example, the GPIO pins corresponding to the mission function is in a tri-state condition).

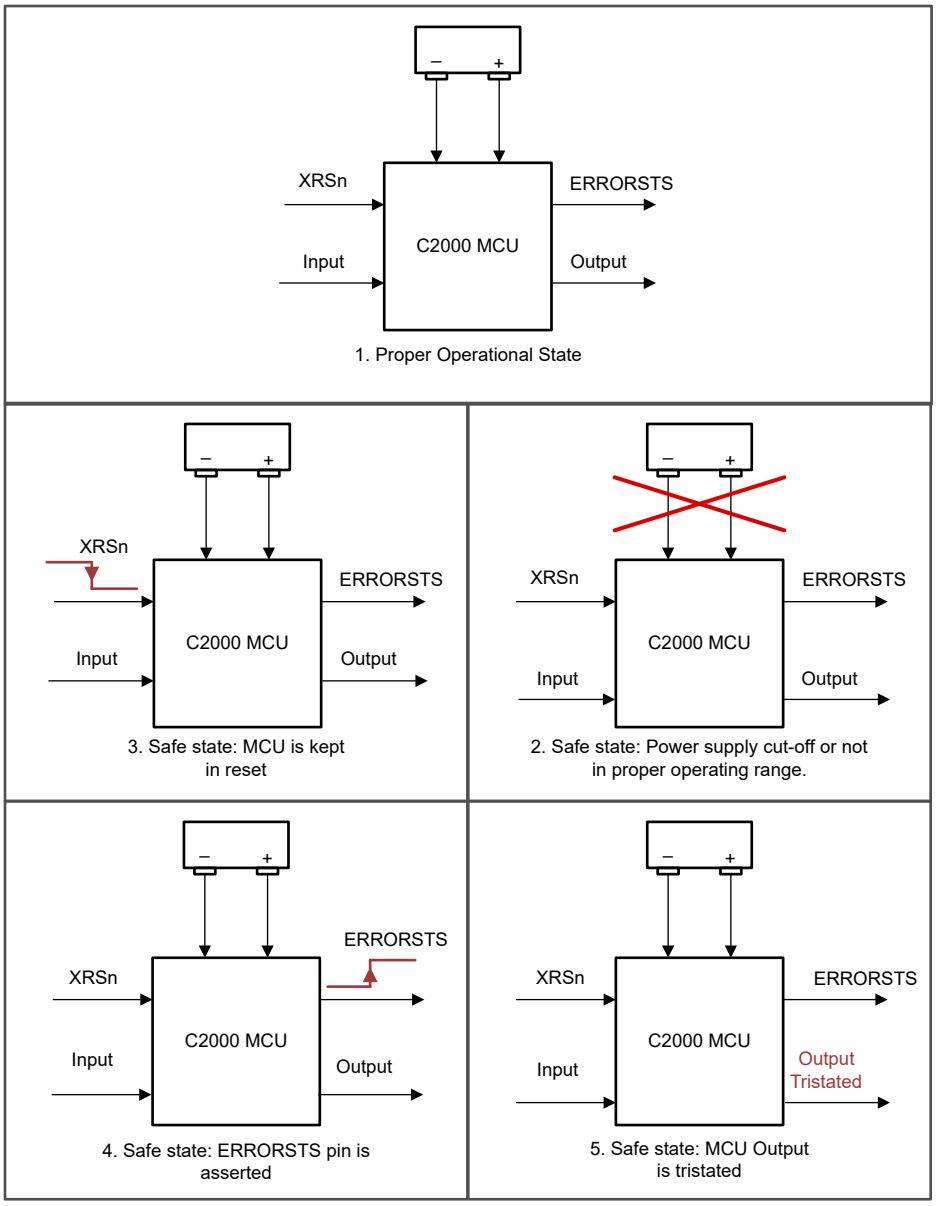

Figure 4-8. TMS320F28P55x MCU Safe State Definition

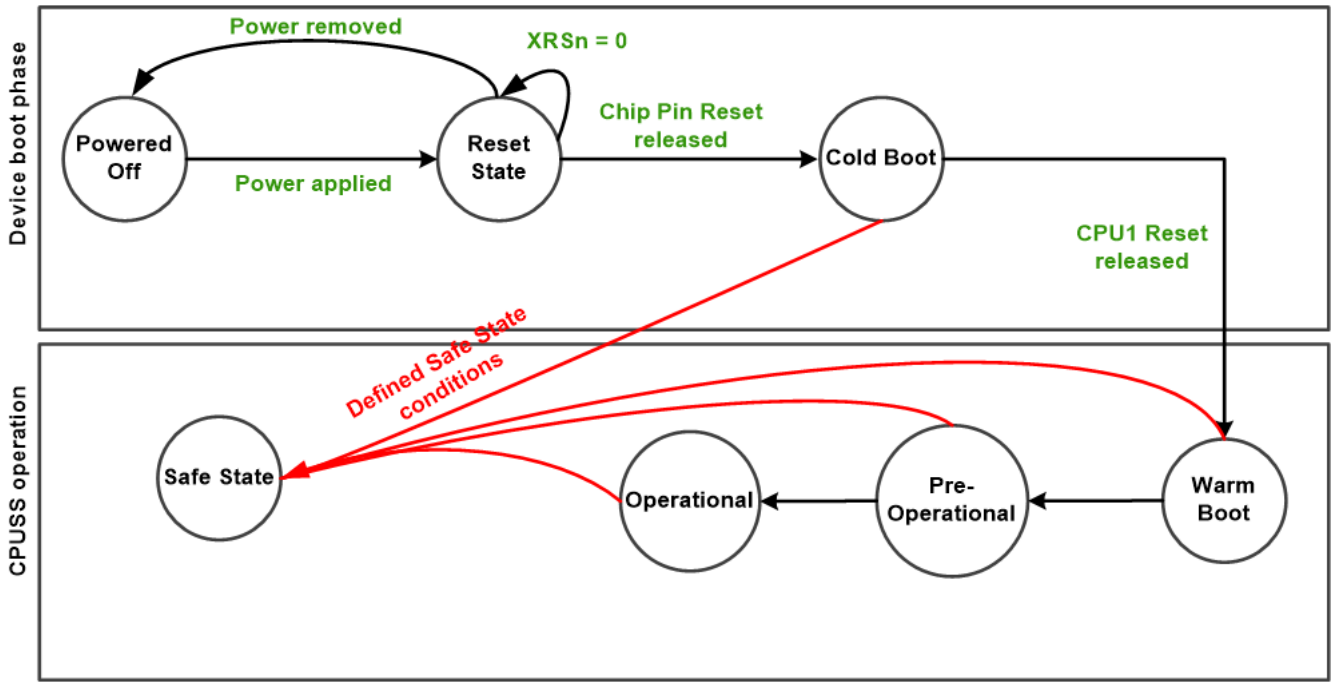

Figure 4-9. TMS320F28P55x MCU Device Operating States

4.2.3.1 Assumed Safety Requirements

The following requirements (assumed safety requirements) (at a minimum) must be implemented by the Level 3 checker (VDA E-gas concept) realized using external components:

- External voltage monitor to supervise the power supply provided to the MCU
- External watchdog timer that can be used for diagnostic purposes
- Components required for taking the system to safe state as per the safe state defined in MCU Safe State.

4.2.4 Operating States

The TMS320F28P55xMCU products have a common architectural definition of operating states. These operating states must be observed by the system developer in the software and system-level design concepts. The operating states state machine is shown in Figure 4-9. The operating states can be classified into device boot phase and CPUSS operation phase.

The various states of the device operating states state machine are:

- **Powered Off** - This is the initial operating state of the C2000 MCU. No power is applied to either core or I/O power supply and the device is non-functional. An external supervisor can perform this action (power down the C2000 MCU) in any of the C2000 MCU states in response to a system-level fault condition or a fault condition indicated by the C2000 MCU.
- **Reset State** – In this state, the device reset is asserted using either the external pins or using any of the internal sources.
- **Safe State** – In the Safe state, the device is either not performing any functional operations or an internal fault condition is indicated using the device I/O pins.
- **Cold Boot** - In the cold boot state, the CPU remains powered but in reset. When the cold boot process is completed, the reset of the initiator CPU is internally released, leading to the warm boot stage.
- **Warm Boot** - The CPU begins execution from boot ROM during the warm boot stage. CPU initializes the device security (all memories come up as secure at the beginning of the warm boot and this stage configures the security as needed for the particular system), exception handling and calibration of analog components, and initializes the peripheral boot mode if required. For more details regarding the boot process, see the device-specific boot ROM specification.

- **Pre-operational** - Transfer of control from boot code to customer code takes place during this phase. Application-specific configurations (for example, clock frequency, peripheral enable, pin mux, and so forth) are performed in this phase. Boot time self-test and proof-test are required to verify device operation is performed properly during this phase. See Section 6.4.3.20 (ROM8) for details.
- **Operational** – This marks the system exiting the pre-operational state and entering the functional state. The device is capable of supporting safety-critical functionality during operational mode.

The device start-up timeline for both CPUs are shown in Figure 4-10.

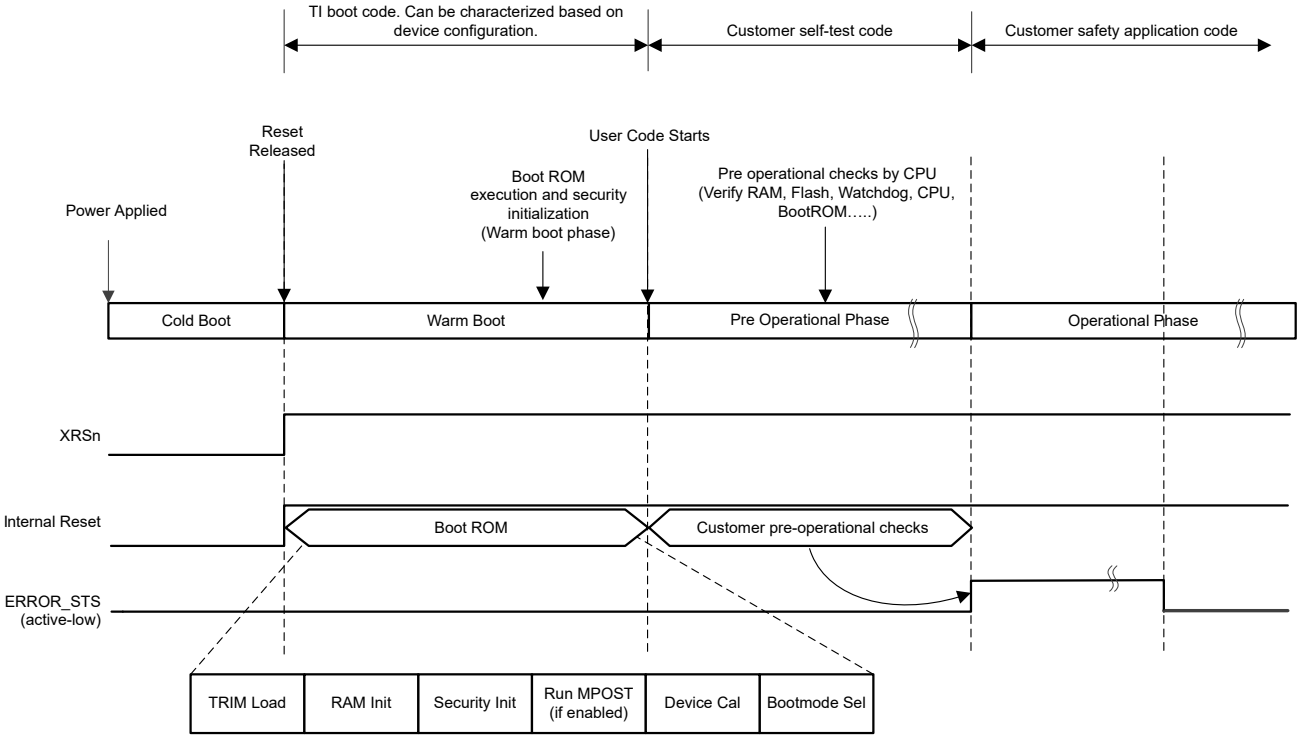

Figure 4-10. TMS320F28P55x MCU CPU Start-Up Timeline

4.3 TMS320F28P55x MCU Safety Implementation

4.3.1 Assumed Safety Requirements

The following assumed safety requirements need to be implemented using external components by the Level 3 checker (VDA E-gas concept).

- External voltage monitor to supervise the power supply provided to the TMS320F28P55x MCU
- External watchdog timer that can be used for diagnostic purposes
- Components required for taking the system to a safe state, as per the TMS320F28P55x MCU safe state defined in Section 4.2.3.

4.3.2 Example Safety Concept Implementation Options on TMS320F28P55x MCU

The CPU in TMS320F28P55x variants support a pair of diverse processing units (C28x and CLA) with heterogeneous asymmetric architectures, instruction sets, and software tools. Either processing units can be used to execute the intended function (the main real-time control function). The safety functions, which allow each safety goal to be met, can be implemented for diagnostics of random hardware failures by running Reciprocal Comparison by Software in separate processing units; providing high diagnostic coverage for the processing units (ISO 26262-5:2018, Table D.4 and IEC 61508-2:2010, Table A.4). Safety mechanisms, such as CPU Handling of Illegal Operation, Illegal Results, and Instruction Trapping, CLA Handling of Illegal Operation and Illegal Results, Internal Watchdog, and so forth, can also be utilized. Heterogeneous CPU cores minimize the possibility of common-mode failures while implementing this reciprocal comparison, thereby improving confidence in the diagnostic coverage. For common-cause failures, such as clock, power, and reset, an external watchdog must be used.

Here are some definitions relevant to the following implementation options:

- Intended Function: Control application implemented on TMS320F28P55x (PFC, DCDC, traction-inverter, and so forth.)
- Safety Function: Achieves risk reduction and is implemented for safety goals identified from HARA
  - Example: Prevent overcurrent, overvoltage, undervoltage, overtemperature, forward or reverse torque, and so forth)
- Diagnostic Function: Verifies safety-function operates correctly when required
  - Shall meet ≥ 60% LFM for ISO 26262:2018 (ASIL B compliance targeted) systems

The following are the safety concept options which can be implemented on TMS320F28P55x.

4.3.2.1 Safety Concept Implementation: Option 1

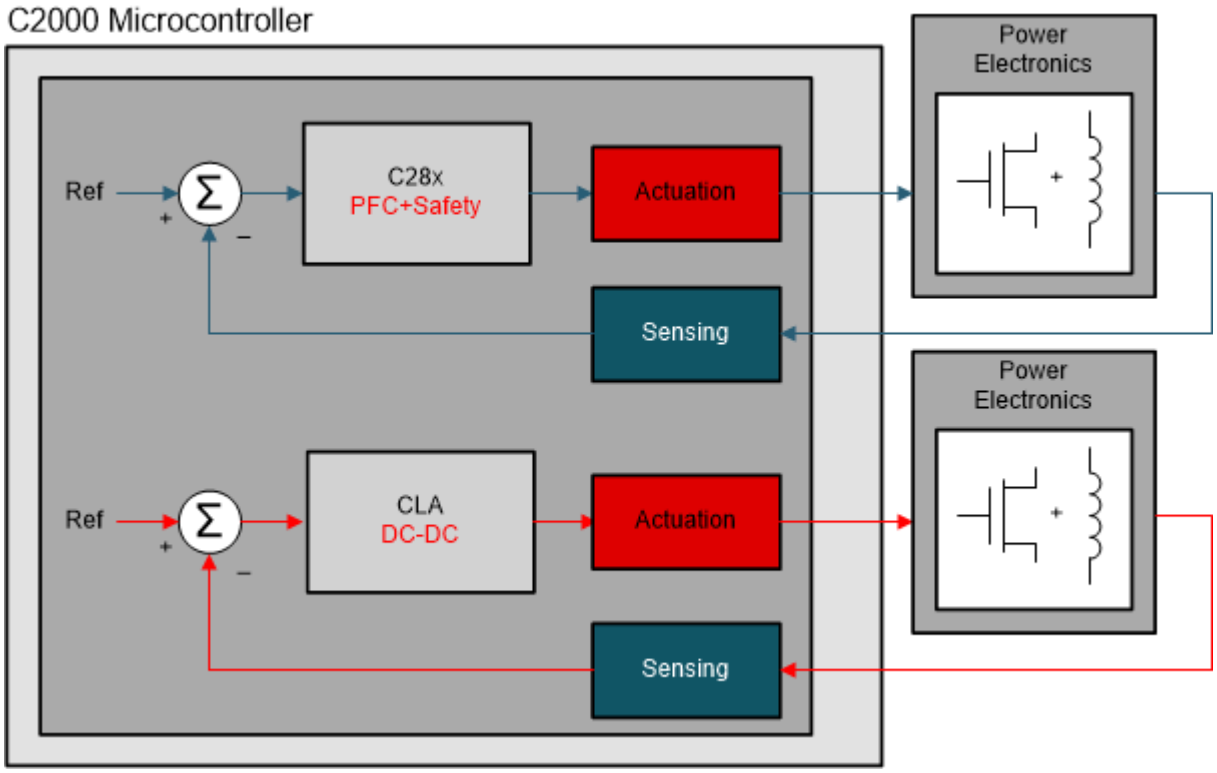

Figure 4-11. Safety Concept Implementation Option 1

- Intended Function: Can be implemented on both C28x and CLA.
- Safety Function: Implement on C28x or CLA.
  - SPFM can be met by Reciprocal Comparison by Software and so forth.
- Diagnostic Function: Implement on the other processing unit.
  - LFM can be met by Software Test of CPU, Software Test of CLA, and so forth.

4.3.2.2 Safety Concept Implementation: Option 2

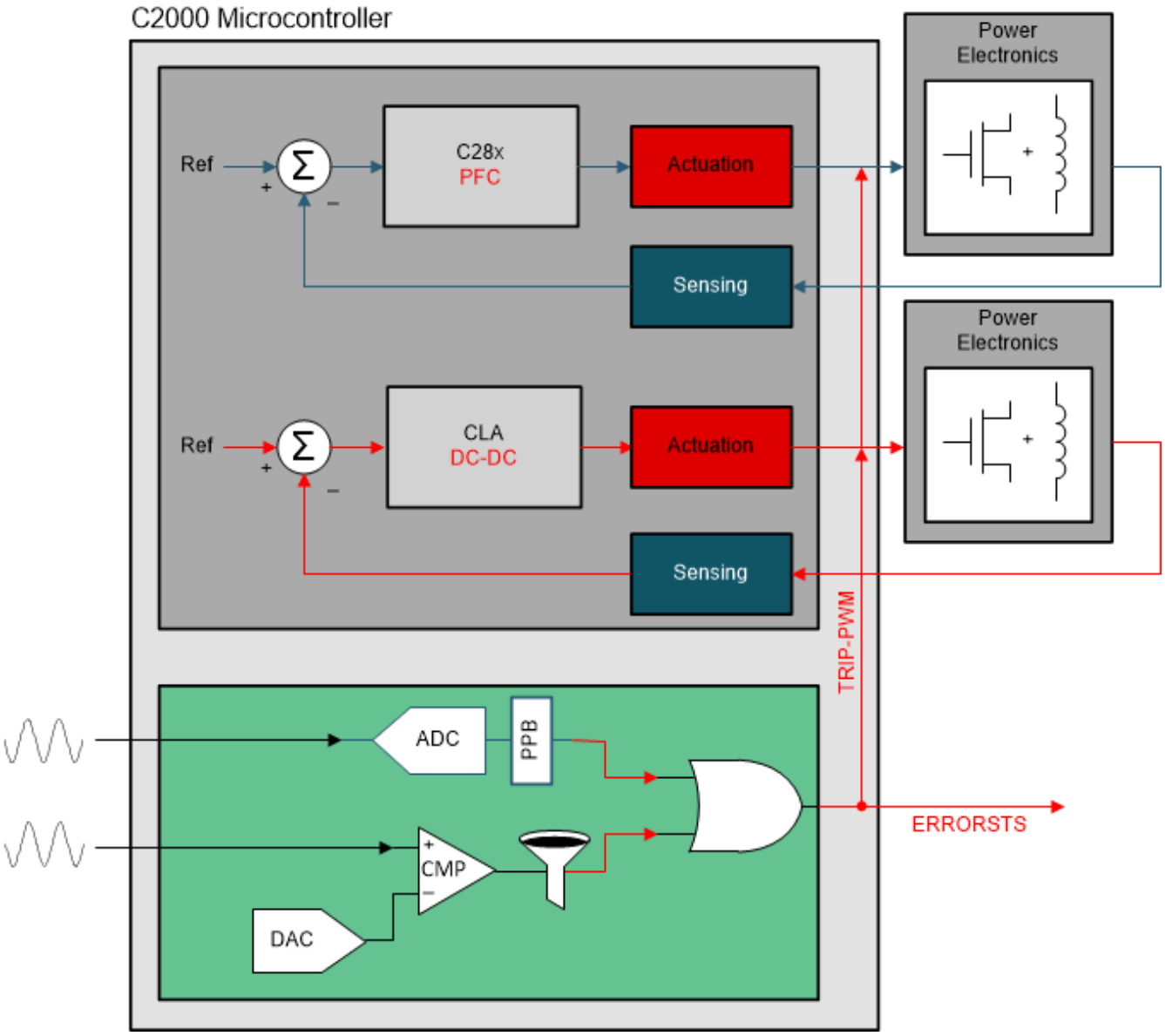

Figure 4-12. Safety Concept Implementation Option 2

- Intended Function: Can be implemented on both C28x and CLA.
- Safety Function: Implement using hardware modules, such as ADC-PPB, CMPSS, CLB, and so forth.
  - SPFM of the safety goal can be met by hardware redundancy between the modules used in implementing safety function, Periodic Software Read Back of Static Configuration Registers and so forth.
- Diagnostic Function: Implement with hardware modules, such as ADC-PPB, CMPSS, CLB, and so forth.
  - LFM can be met by Software Test of Function Including Error Tests and so forth.

4.4 TMS320F28P55x Diagnostic Libraries

The diagnostic libraries designed for the TMS320F28P55x family of devices are comprised of the *C28x CPU Self-Test Library* (C28x_STL), *Control Law Accelerator Self-Test Library* (CLA_STL), and *Software Diagnostic Library* (SDL). These libraries are designed to help TI customers using the TMS320F28P55x to develop

functionally safe systems that can comply with a wide range of standards for end products catering to the automotive (ISO 26262), industrial (IEC 61508), and appliance (IEC 60730) markets.

Note

This is only a draft of the TMS320F28P55x Functional Safety Manual and availability of the diagnostic libraries can differ from what is documented here. Contact your FAE or sales representative for more information about the availability of beta versions of the software. The software certification this document references has not yet been completed.

Table 4-1. DC and SCC Targeted for F28P55x Diagnostic Libraries

Library	Permanent fault Diagnostic Coverage (DC)	Systematic Capability Compliance (SCC)	Description
C28x_STL	≥ 60%	ASIL D | SIL 3	This STL implements CPU3 - Software test of CPU
CLA_STL	≥ 60%	ASIL D | SIL 3	This STL implements CLA2 - Software test of CLA
SDL	Examples Only	N/A	The SDL provides examples of several safety mechanisms described in the safety manual

The C28x_STL and CLA_STL provide an implementation of the CPU3 - Software Test of CPU and CLA2 - Software Test of CLA safety mechanisms, respectively. The C28x_STL and CLA_STL are independently assessed and found to be satisfactory for integration into safety-related systems up to ASIL D and SIL 3, according to ISO 26262:2018 and IEC 61508:2010, respectively.

The software diagnostic library (SDL) comprises general example implementations of several safety mechanisms. The SDL examples are developed using a baseline quality software development flow and are not required to be compliant with any particular standard. As such, the SDL is not certified by TÜV® SÜD. Users are expected to study and adapt the provided examples into their safety-related applications and are responsible for their own product level third-party certifications.

4.4.1 Assumptions of Use - F28P55x Self-Test Libraries

This section provides the high-level details related to what considerations a system integrator must evaluate during the process of defining and building their F28P55x based safety architecture. The software support for the various safety mechanisms in the F28P55x can be divided into the following categories:

- C28x Self-Test Library
- CLA Self-Test Library
- Software Diagnostic Library

A safe product built on the F28P55x device hierarchically deploys each of the software designs provided by TI.

The first in the hierarchy is the C28x_STL which detects permanent faults inside the CPU by implementing a software test of the CPU. The second in the hierarchy is the SDL which provides a series of safety mechanism examples that are designed to detect permanent faults inside several key elements within the device. Lastly, the CLA_STL can be deployed to detect permanent faults inside the CLA.

The CLA_STL makes use of, and depends on, both the C28x CPU and the CLA to test the CLA. Therefore, running the C28x_STL first to make sure that the CPU is functioning properly and is capable of performing the required safety operations is important. To detect potential failure causes of the C28x_STL, the integrator must verify that the internal watchdog and Flash and RAM ECC|Parity logic are enabled before the C28x_STL runs. Checks of elements such as the clock, internal watchdog, Flash, and RAM relevant in the execution of the CLA_STL must be performed. The successful completion of the software diagnostics, selected by the system integrator, can be used as the qualifier to run the test vectors supported by the CLA_STL.

4.4.2 Operational Details - F28P55x Self-Test Libraries

The diagnostic libraries are used on an F28P55x target in the implementation of safety in the host application. Therefore, the system integrator must fully comprehend all aspects of the associated system constraints imposed by the integration of the STLs to incorporate safety into the overall system.

4.4.2.1 Operational Details – C28x Self-Test Library

The C28x_STL implements the CPU3 - Software Test of CPU. This library is certified by TÜV® SÜD to meet LFM for ISO26262:2018 ASIL B. The C28x_STL runs directly on the CPU and effectively tests a subset of CPU registers, CPU instructions, CPU flags, the FPU, TMU, and VCRC functionality.

To run these tests, the C28x_STL occupies program memory storage space and dedicated execution RAM space. All the C28x_STL tests are destructive in nature and do not have a method to restore the system back to the original sate. Since the C28x_STL tests and reports on the health of the CPU and the system state cannot be meaningfully saved and restored, the test must be integrated into the start-up portion of the application. The system integrator must enable the watchdog to protect the application against runaway code.

Note

The C28x_STL source code delivered with the C28x_STL software download can be validated using the MD5 checksum provided for each source file. The MD5 checksums for all the relevant files are available in the SDF file provided in the release package. The system integrator must consult the C28x_STL user guides and understand all aspects of integrating the library into the host application.

4.4.2.2 Operational Details – CLA Self-Test Library

The CLA_STL implements the CLA2 - Software Test of CLA. The start-up tests of the CLA_STL are also destructive in nature and must be run during start-up operations. The run time tests of the CLA_STL comprise the bulk of the tests designed to run in conjunction with the host application. The CLA host application must allocate the time and space for the run time tests. To achieve the required DC, the CPU must run both the CLA_STL POST and PEST tests.

The process of determining the type of tests required to achieve DC with excellent distribution and quality of coverage is a fairly thorough and rigorous process. The process requires running a fault injection campaign using the right combination of test vectors and confirming the coverage. These test vectors are precisely packaged into the CLA_STL.

Note

The CLA_STL source code delivered with the CLA_STL can be validated using the MD5 checksum provided for each source file. The MD5 checksums for all the relevant files are available in the SDF file provided in the release package. The system integrator must consult the CLA_STL user guides and understand all aspects of integrating the library into the host application.

4.4.2.3 Operational Details - Software Diagnostic Libraries

Table 4-2 is a mapping of SDL software modules and APIs to safety features and diagnostic.

Table 4-2. Module to Safety Mechanism Mapping

Module Name	Unique Identifier
STL_CPU_REG	No unique identifier, added for IEC 60730
STL_CRC	NWFLASH5
STL_March	SRAM3,
STL_MCAN_RAM	MCAN7, MCAN15
STL_OSC_CT	CLK2
STL_OSC_HR	OTTO1, CLK3
STL_PIE_RAM	PIE3, PIE6
sdl_ex_dcsm_ffi	No unique identifier, demo of freedom from interference using DCSM

Table 4-2. Module to Safety Mechanism Mapping (continued)

Module Name	Unique Identifier
sdl_ex_flash_ecc_test	NWFLASH15
sdl_ex_flash_prefetch_test	NWFLASH14
sdl_ex_mcd_test	CLK12
sdl_ex_ram_access_protect	SRAM10
sdl_ex_ram_ecc_parity_test	SRAM13, SRAM14,
sdl_ex_watchdog	CLK10

4.4.3 C2000 Safety STL Software Development Flow

The C28x_STL and CLA_STL are developed using the TÜV® SÜD Certified TI internal software development process specification which targets software development flows for baseline quality, automotive quality, and functional safety quality (for functional safety, specifically the target is systematic capability compliance with the IEC 61508 and ISO 26262 standards). The TÜV-SÜD certificate for TI's software development process is available here.

The software development process specification describes the contents of the required deliverables during each of the four phases, namely, *Assess*, *Plan*, *Create*, and *Validate*. By adhering to this specification and complying with the underlying processes, including methods and techniques (IEC 61508-3, ISO 26262-6) that are comprehended in the work-products, TI's software and firmware development process achieves a systematic capability of ASIL D (ISO 26262-6) and SIL 3 (IEC 61508-3).

- Figure 4-13 depicts TI's (TÜV-SÜD certified) software development life cycle regarding the various quality levels supported by the process.
- Detailed supporting procedures are documented to provide functional safety throughout the project life cycle. Additional tools and techniques regarding safety-integrity levels of the targeted standards are applied at each development phase.
- Functional safety audits and assessments are planned and conducted, per the defined procedure. Qualified personnel with adequate independence conduct these audits and assessments, as required by the targeted standards and safety levels.

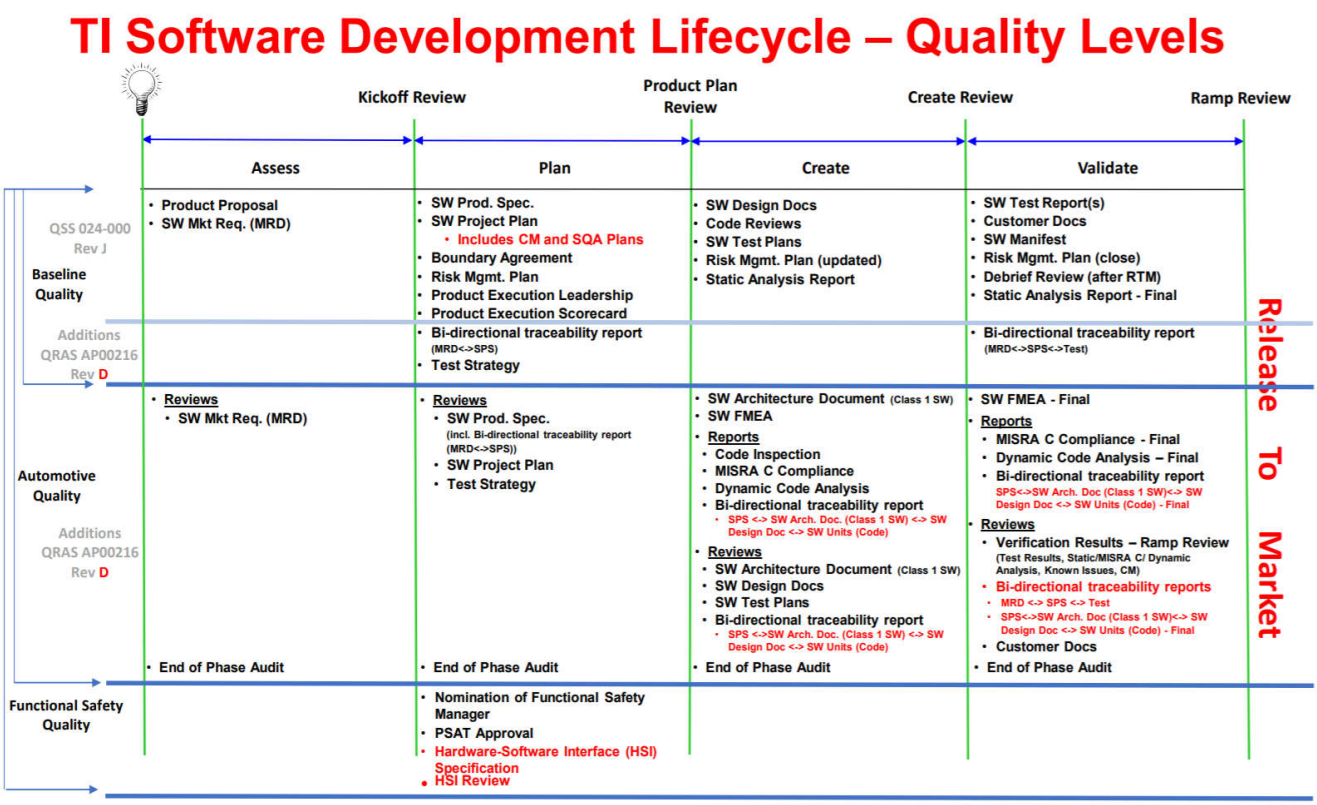

Figure 4-13. TI Software Development Life Cycle - Quality Level

4.5 Functional Safety Constraints and Assumptions

In creating a functional Safety Element out of Context (SEooC) concept and doing the functional safety analysis, TI generates a series of assumptions on system-level design, functional safety concept, and requirements. These assumptions (sometimes called Assumptions of Use) are listed below. Additional assumptions about the detailed implementation of safety mechanisms are separately located in Section 6.4.

The TMS320F28P55x functional safety analysis was done under the following system assumptions:

- **[SA_1]** The system integrator shall follow all requirements in the component data sheet.
- **[SA_2]** The system shall be configured to allow this device to activate or communicate with assigned actuators to maintain proper operating state of the external system based on input from assigned sensors.
- **[SA_3]** If the system is in a fault detected state, software can attempt to recover from the fault before a safety goal is violated. Software shall be configured to put the system into a safe state, if unable to recover from a fault, before the violation of a safety goal occurs.
- **[SA_4]** The system integrator shall review the recommended diagnostics in the Safety Manual and Safety Analysis Report (FMEDA) and determine the appropriate diagnostics to include in their system. These diagnostics shall be implemented according to the device Safety Manual and data sheet.
- **[SA_5]** The software shall initialize the continuous diagnostics and periodically run the test-for-diagnostics in alignment with the system safety concept including fault tolerant time interval (FTTI), process safety time (PST), and multipoint fault detection time interval (MPFDTI).
- **[SA_6]** The power supply to this device shall provide the appropriate power on each of the power inputs. These rails shall be monitored for deviations outside the device specifications.
- **[SA_7]** The power supply or other external monitoring device (or devices) shall monitor the device error pins and transition the system to a safe state after an unrecoverable error is indicated.
- **[SA_8]** The power supply or other external monitoring device shall monitor the device to provide coverage for diagnosing power faults. A watchdog is an example technique to achieve this diagnostic.
- **[SA_9]** The power supply or other external monitoring device shall be used as a parallel path to disable downstream actuators in situations, such as, before the device is capable to perform a safety function (for example, device start-up) or when device faults have been detected that can compromise the decision-making capability of the device (for example, MCU fault conditions).
- **[SA_10]** A system concept appropriate to the targeted ASIL/SIL level shall be chosen to detect faults in the execution of the code running on the CPUs related to the safety function. This concept shall also take into consideration shared modules (for example, RAM, flash, ADCs, and so forth). Examples include, but are not limited to, reciprocal comparison by software, watchdogs, and so forth.
- **[SA_11]** System configuration and implementation checks shall be performed by the system integrator.
- **[SA_12]** Availability of system function is not a safety requirement. When the system is off or in reset, the system shall be in a safe state.
- **[SA_13]** Device power sequencing requirements shall not be considered to be safety critical.
- **[SA_14]** The system integrator shall review the SEooC analysis and integrate the analysis into the system-level safety analysis. The diagnostics shall be applied as needed regarding the system safety goals and requirements and integration testing shall be performed.
- **[SA_15]** The system integrator shall analyze the other components in the system regarding the safety concept and implement diagnostics on those components as needed regarding that safety concept.
- **[SA_16]** The safety function is considered to operate in high and continuous demand mode of operation (per IEC 61508).

Note

This consideration does not exclude the option of operating in low-demand mode. This assumption is made to provide a baseline for judgment of the safety-integrity level targets.

- **[SA_17]** This device is considered to be a type B safety-related element or subsystem (per IEC 61508). Additionally, the F28P55x MCU claims no hardware fault tolerance (HFT = 0), as defined in IEC 61508:2010.
- **[SA_18]** The safety function shall not begin until the software enables the safety function after this device has successfully completed the start-up sequence, run any required integrity checks, and is in a normal mode of operation.

- **[SA_19]** Debug and design for test (DFT) logic shall be disabled during operation of a safety function.

During integration activities these assumptions of use and integration guidelines described for this component shall be considered. Use caution if one of the above functional safety assumptions on this component cannot be met, as some identified gaps can be unresolvable at the system-level.

Chapter 5

Description of Safety Elements

This section contains a brief description of the elements on the TMS320F28P55x MCU device family, organized based on the classification of generic hardware parts of a system [8], as indicated in Figure 5-1. For a full functional description of any of these modules, refer to the device-specific technical reference manual. The brief description of the hardware part is followed by the list of primary safety mechanisms that can be employed to provide diagnostic coverage to the hardware part. Some safety standards have the requirement to provide diagnostic coverage for the primary diagnostic measures (for example, latent fault metric requirement from ISO26262). These measures are called as test for diagnostics. Primary diagnostics, of type *Software* and *Hardware*|*Software*, involve execution of the software on the processing units using the CPU and CLA and also using many of the MCU parts, like Interconnect, memory (Flash, SRAM, and ROM), and TMS320F28P55x MCU infrastructure components (Clock, Power, Reset, and JTAG). To maintain integrity of the implemented primary diagnostics and the associated diagnostic coverage values, measures to protect the execution of primary diagnostics on respective processing units must be implemented. TI recommends an appropriate combination of tests for diagnostics is implemented for parts of the MCU that contribute to the successful operation of the processing units. For diagnostics of these parts, refer to the respective sections in this safety manual. In cases where separate tests for diagnostic measures exist for a primary diagnostic measure, those tests are mentioned along with the respective hardware part.

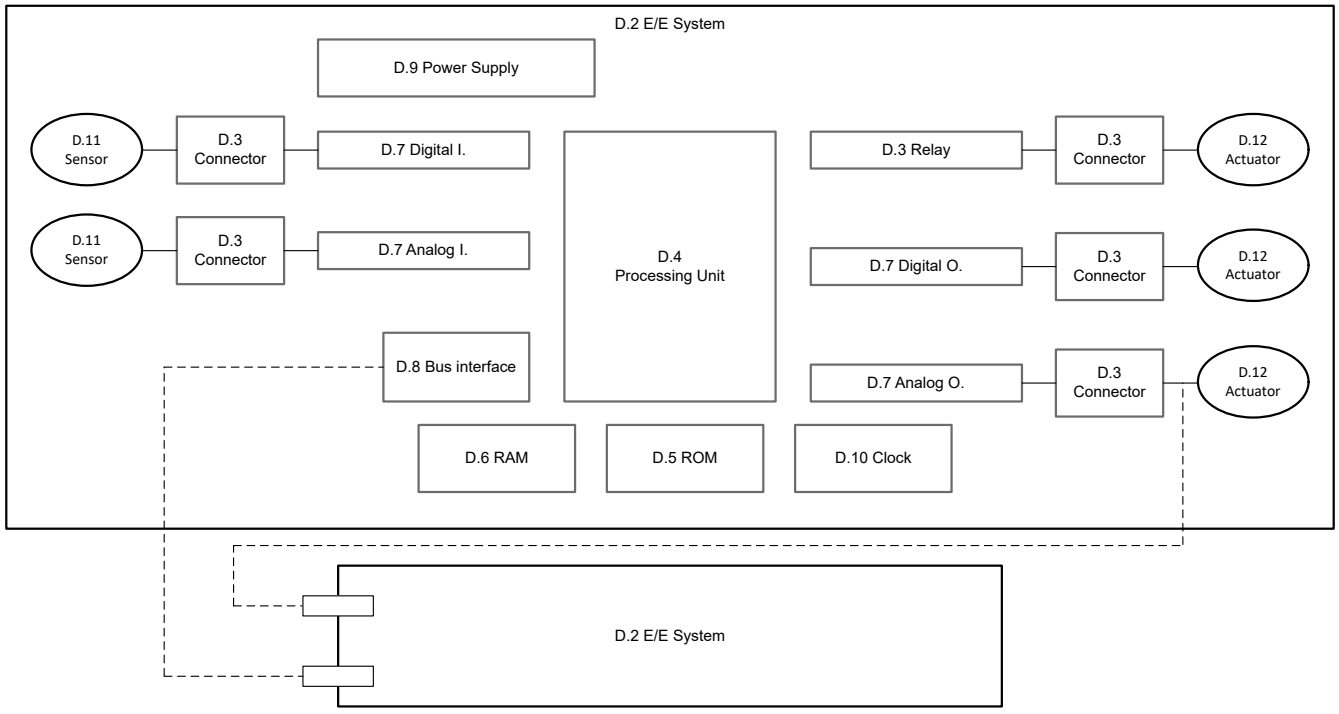

Figure 5-1. Generic Hardware of a System

5.1 C2000 MCU Infrastructure Components

5.1.1 Power Supply

The TMS320F28P55x MCU device family requires an external device to supply the necessary voltage and current for proper operation. Separate voltage rails are available for core (1.2V), analog (3.3V), Flash (3.3V) and I/O logic (3.3V). The following mechanisms can be used to improve the diagnostic coverage of the C2000 MCU power supply.

Note

- Having independent voltage supervision at system-level is an assumption used while performing safety analysis.
- Devices can be implemented with multiple power rails that are intended to be ganged together on the system PCB. For proper operation of power diagnostics, TI recommends implementing one voltage supervisor per ganged rail.
- Common mode failure analysis of the external voltage supervisor, along withTMS320F28P55x MCU, is useful to determine dependencies in the voltage generation and supervision circuitry.
- Customers can consider using TI's TPS6538x power supply and safety-companion device for voltage supervision at system-level.

5.1.1.1 Power Supply (Power) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- PWR1 - External Voltage Supervisor
- PWR2 - External Watchdog
- SYS1 - Multi-Bit Enable Keys For Control Registers
- SYS2 - Lock Mechanism For Control Registers
- SYS3 - Software Read Back Of Written Configuration
- SYS4 - Periodic Software Read Back Of Static Configuration Registers
- SYS5 - Online Monitoring Of Temperature
- SYS8 - Eallow And Meallow Protection For Critical Registers
- CLK6 - Internal Watchdog (Wd)
- PWR4 - Brownout Reset (Bor)

5.1.2 Clock

The C2000 MCU device family products are primarily synchronous logic devices and as such require clock signals for proper operation. The clock management logic includes clock sources, clock generation logic including clock multiplication by phase lock loops (PLLs), clock dividers, and clock distribution logic. The registers that are used to program the clock management logic are located in the system control module. The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

Note

- DCC is the recommended method of clock monitoring over the CPU timer or HRPWM-based methods.
- TI recommends the use of an external watchdog over an internal watchdog for mitigating the risk due to common mode failure. TI also recommends the use of a program sequence, windowed, or question and answer watchdog as opposed to a single-threshold watchdog due to the additional failure modes that can be detected by a more advanced watchdog.
- Driving a high-frequency clock output on the XCLKOUT pin can have EMI implications. The selected clock must be scaled appropriately before sending out through I/O.

5.1.2.1 Clock Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- CLK1 - Missing Clock Detect (Mcd)
- CLK2 - Clock Integrity Check Using Cpu Timer
- CLK3 - Clock Integrity Check Using Hrpwm
- CLK5 - External Monitoring Of Clock Via Xclkout
- CLK6 - Internal Watchdog (Wd)
- CLK7 - External Watchdog
- CLK8 - Periodic Software Read Back Of Static Configuration Registers
- CLK9 - Software Read Back Of Written Configuration
- CLK13 - Pll Lock Profiling Using On-Chip Timer
- CLK14 - Peripheral Clock Gating (Pclkcr)
- CLK17 - Dual Clock Comparator (Dcc) - Type 2

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements:

- CLK10 - Software Test Of Watchdog (Wd) Operation
- CLK12 - Software Test Of Missing Clock Detect Functionality

5.1.3 APLL

The APLL is a phase-locked loop circuit used to multiply the input clock (XTAL, external oscillator or internal zero-pin oscillator) to run up the F28P55x MCU to the full speed allowed in the device data sheet. In this case the *A* prefix indicates the PLL is drawing power from the VDDA supply.

The following tests can be applied as diagnostics for this module to provide diagnostic coverage on a specific function.

5.1.3.1 APLL Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- APLL1 - Clock Integrity Check Using Dcc
- APLL2 - Pll Lock Indication
- APLL4 - Internal Watchdog (Wd)
- APLL5 - External Watchdog
- APLL7 - External Monitoring Of Clock Via Xclkout
- APLL11 - Interleaving Of Fsm States

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements:

- APLL6 - Software Test Of Dcc Functionality Including Error Tests
- APLL10 - Software Test Of Pll Functionality Including Error Tests

5.1.4 Reset

The power-on reset (POR) generates an internal warm reset signal to reset the majority of digital logic as part of the boot process. The warm reset can also be provided at device level as an I/O pin (XRSn) with open drain implementation. Diagnostic capabilities like NMI watchdog and watchdog are capable of issuing a warm reset. For more information on the reset functionality, see the device-specific data sheet.

Note

- Internal watchdogs are not a viable option for reset diagnostics as the monitored reset signals interact with the internal watchdogs.
- Customers can consider using the TI TPS6538x power supply and safety companion device for reset supervision at system-level.

5.1.4.1 Reset Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- RST1 - External Monitoring Of Warm Reset (Xrsn)
- RST2 - Reset Cause Information
- RST4 - Glitch Filtering On Reset Pins
- RST5 - Nmiwd Shadow Registers
- RST6 - Periodic Software Read Back Of Static Configuration Registers
- RST7 - Software Read Back Of Written Configuration
- RST8 - Nmiwd Reset Functionality
- RST9 - Peripheral Soft Reset (Softpres)
- SYS9 - Software Test Of Errorsts Functionality
- CLK6 - Internal Watchdog (Wd)
- CLK7 - External Watchdog
- RST10 - Software Test Of Reset (Type 1)

5.1.5 System Control Module and Configuration Registers

The system control module contains the memory-mapped registers to configure clock, analog peripherals settings, and other system related controls. The system control module is also responsible for generating the synchronization of system resets and delivering the warm reset (XRSn). The configuration registers include the registers within peripherals that are not required to be updated periodically.

Note

- Review the *Clock* and *Reset* sections as these features are closely controlled by the system control module.
- Customers can consider using the TI TPS6538x power supply and safety companion device for ERRORSTS pin supervision at system-level.

5.1.5.1 System Control Module and Configuration Registers (System control) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- SYS1 - Multi-Bit Enable Keys For Control Registers
- SYS2 - Lock Mechanism For Control Registers
- SYS3 - Software Read Back Of Written Configuration
- SYS4 - Periodic Software Read Back Of Static Configuration Registers
- SYS5 - Online Monitoring Of Temperature
- SYS6 - Peripheral Clock Gating (Pclkcr)
- SYS7 - Peripheral Soft Reset (Softpres)
- SYS8 - Eallow And Meallow Protection For Critical Registers
- SYS9 - Software Test Of Errorsts Functionality
- SYS11 - Peripheral Access Protection - Type 1

5.1.6 JTAG Debug, Trace, Calibration, and Test Access

The TMS320F28P55x MCU device family supports debug, test, and calibration implemented over an IEEE 1149.1 JTAG debug port. The physical debug interface is internally connected to a TI debug logic (ICEPICK), which arbitrates access to test, debug, and calibration logic. Boundary scan is connected in parallel to the debug logic to support usage without preamble scan sequences for easier manufacturing board test.

5.1.6.1 Debug Logic (JTAG) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- JTAG1 - Hardware Disable Of Jtag Port
- JTAG2 - Lockout Of Jtag Access Using Otp
- JTAG3 - Internal Watchdog (Wd)
- JTAG4 - External Watchdog

5.1.7 Advanced Encryption Standard (AES) Accelerator

The AES module provides hardware-accelerated data encryption and decryption operations based on a binary key. The AES is a symmetric cipher module that supports a 128-bit, 192-bit, or 256-bit key in hardware for encryption and decryption. The AES module is based on a symmetric algorithm, which means that the encryption and decryption keys are identical. To encrypt data means to convert from plain text to an unintelligible form called cipher text. Decrypting cipher text converts previously encrypted data to the original plain text form.

5.1.7.1 AES Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- AES1 - Decryption Of Encrypted Data Output Using Same Key And Iv (Initialization Vector)
- AES2 - Information Redundancy Techniques Including End-To-End Safing
- AES3 - Periodic Software Read Back Of Static Configuration Registers
- AES4 - Software Read Back Of Written Configuration
- AES5 - Transmission Redundancy
- AES6 - Disabling Of Unused Dma Trigger Sources
- AES7 - Software Test Of Function Including Error Tests
- AES8 - Software Test Of Standalone Ghash Operation

5.2 Processing Elements

5.2.1 C28x Central Processing Unit (CPU)

The CPU is a 32-bit fixed-point processor with floating point, Viterbi, complex math and CRC unit (VCU) and trigonometric math unit (TMU) co-processors. This device draws from the best features of digital signal processing; reduced instruction set computing (RISC); and microcontroller architectures, firmware, and tool sets. The CPU features include a modified Harvard architecture and circular addressing. The RISC features are single-cycle instruction execution and register-to-register operations. The modified Harvard architecture of the CPU enables instruction and data fetches to be performed in parallel. The CPU does this over six separate address and data buses. The unique architecture makes the C28x CPU amenable to integrating safety features external to the CPU, but on-chip, to provide improved diagnostic coverage.

Note

Measures to mitigate common cause failures in the CPU subsystem: Common-cause failures are one of the important failure modes when a safety-related design is implemented in a silicon device. The contribution of hardware- and software-dependent failures is estimated on a qualitative basis because no general and sufficiently reliable method exists for quantifying such failures. The system integrator must perform a detailed analysis based on the inputs from 26262-11:2018, Section 4.7 and IEC61508 ed2 PART 2 Annex E (BetaIC method).

5.2.1.1 C28x Central Processing Unit (C28x) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- CPU1 - Reciprocal Comparison By Software
- CPU3 - Software Test Of Cpu
- CPU4 - Periodic Software Read Back Of Static Configuration Registers
- CPU5 - Access Protection Mechanism For Memories
- JTAG1 - Hardware Disable Of Jtag Port
- CPU7 - Cpu Handling Of Illegal Operation, Illegal Results And Instruction Trapping
- CPU8 - Internal Watchdog (Wd)
- CPU9 - External Watchdog
- CPU10 - Information Redundancy Techniques
- CPU14 - Stack Overflow Detection
- CPU18 - Embedded Real Time Analysis And Diagnostic (Erad)

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements:

- CPU15 - Vcu Crc Auto Coverage
- CPU19 - Inbuilt Hardware Redundancy In Erad Bus Comparator Module

5.2.1.2 FPU_TMU (FPU_TMU) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- CPU1 - Reciprocal Comparison By Software
- CPU5 - Access Protection Mechanism For Memories
- JTAG1 - Hardware Disable Of Jtag Port
- CPU8 - Internal Watchdog (Wd)
- CPU9 - External Watchdog
- CPU10 - Information Redundancy Techniques
- CPU14 - Stack Overflow Detection
- CPU18 - Embedded Real Time Analysis And Diagnostic (Erad)

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements:

- CPU15 - Vcu Crc Auto Coverage
- CPU19 - Inbuilt Hardware Redundancy In Erad Bus Comparator Module

5.2.2 Control Law Accelerator

The control law accelerator (CLA) is an independent, fully-programmable, 32-bit floating-point math accelerator with independent ISA and independent compiler and the accelerator helps concurrent control-loop execution. The low interrupt-latency of the CLA allows the CLA to read ADC samples *just-in-time*. This significantly reduces the ADC sample to output delay to enable faster system response and higher MHz control loops.

5.2.2.1 Control Law Accelerator (MCLA) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- CLA1 - Reciprocal Comparison By Software
- CLA2 - Software Test Of Cla
- CLA3 - Cla Handling Of Illegal Operation And Illegal Results
- CLA4 - Software Read Back Of Written Configuration
- CLA5 - Periodic Software Read Back Of Static Configuration Registers

- CLA7 - Information Redundancy Techniques
- CLA8 - Cla Liveness Check Using Cpu
- CLA9 - Access Protection Mechanism For Memories
- SRAM5 - Periodic Software Read Back Of Static Configuration Registers
- SRAM10 - Software Test Of Function Including Error Tests

5.3 Memory (Flash, SRAM and ROM)

5.3.1 Embedded Flash Memory

The embedded Flash memory is a non-volatile memory that is tightly coupled to the C28x CPU. Each CPUSS has a dedicated flash memory. The Flash memory is not accessible by CLA or DMA. The Flash memory is primarily used for CPU instruction access, though data access is also possible. Access to the Flash memory can take multiple CPU cycles depending upon the device frequency and flash wait state configuration. Flash wrapper logic provides prefetch and data cache to improve performance.

5.3.1.1 NW Embedded Flash Memory (NW Flash) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- NWFLASH1 - Flash Ecc
- NWFLASH2 - Flash Program Verify And Erase Verify Check
- NWFLASH3 - Flash Program/Erase Protection
- NWFLASH4 - Flash Wrapper Error And Status Reporting
- NWFLASH5 - Vcu Crc Check Of Static Memory Contents
- NWFLASH6 - Prevent 0 To 1 Transition Using Program Command
- NWFLASH7 - On-Demand Software Program Verify And Blank Check
- NWFLASH8 - Software Readback Of Written Configuration
- NWFLASH9 - Cmdweprot* And Program Command Data Buffer Registers Self-Clear After Command Execution
- NWFLASH10 - Ecc Generation And Checker Logic Is Seperate In Hardware
- NWFLASH12 - Bit Multiplexing In Flash Memory Array
- NWFLASH14 - Software Test Of Flash Prefetch, Data Cache And Wait-States
- NWFLASH16 - Information Redundancy Techniques

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements:

- NWFLASH13 - Autoecc Generation Override
- NWFLASH15 - Software Test Of Ecc Logic

5.3.2 Embedded SRAM

The TMS320F28P55x MCU device family has the following types of SRAMs with different characteristics.

- Dedicated to each CPU (M0, and M1 RAM)
- Shared between the CPU and the CLA (LSx RAM)
- Shared between the CPU and DMA of both subsystems (GSx RAM)
- Used to send and receive messages between processors (MSGRAM)

All these RAMs are highly configurable to achieve control for write access and fetch access from different initators. All dedicated RAMs are enabled with the ECC feature (both data and address) and shared RAMs are enabled with the parity (both data and address) feature. Each RAM has a controller which implements access protection, security related features, as well as ECC and parity features for that RAM.

5.3.2.1 SRAM Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- SRAM1 - Sram Ecc
- SRAM2 - Sram Parity
- SRAM3 - Software Test Of Sram
- SRAM4 - Bit Multiplexing In Sram Memory Array
- SRAM5 - Periodic Software Read Back Of Static Configuration Registers
- SRAM6 - Software Read Back Of Written Configuration
- SRAM7 - Data Scrubbing To Detect/Correct Memory Errors
- SRAM8 - Vcu Crc Check Of Static Memory Contents
- SRAM10 - Software Test Of Function Including Error Tests
- SRAM11 - Access Protection Mechanism For Memories
- SRAM12 - Lock Mechanism For Control Registers
- SRAM16 - Information Redundancy Techniques
- SRAM17 - Cpu Handling Of Illegal Operation, Illegal Results And Instruction Trapping
- SRAM18 - Internal Watchdog (Wd)
- SRAM19 - External Watchdog
- SRAM20 - Cla Handling Of Illegal Operation And Illegal Results
- SRAM21 - Memory Power-On Self-Test (Mpost)

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements:

- SRAM13 - Software Test Of Ecc Logic
- SRAM14 - Software Test Of Parity Logic

5.3.3 Embedded ROM

The TMS320F28P55x MCU device family has the following types of ROMs for each CPU subsystem :

- Boot ROM helps to boot the device and contains functions for security initialization, device calibration, and supports different boot modes
- Secure ROM functions are not developed to meet any systematic capability compliance (ISO 26262-6/IEC 61508-3) and are not to be used in functional safety applications.
- CLA Data ROM contains math tables for CLA application usage

5.3.3.1 ROM Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- ROM1 - Vcu Crc Check Of Static Memory Contents
- ROM2 - Periodic Software Read Back Of Static Configuration Registers
- ROM3 - Software Read Back Of Written Configuration
- ROM4 - Software Test Of Function Including Error Tests
- ROM5 - Cpu Handling Of Illegal Operation, Illegal Results And Instruction Trapping
- ROM6 - Internal Watchdog (Wd)
- ROM7 - External Watchdog
- ROM8 - Power-Up Pre-Operational Security Checks
- ROM10 - Memory Power-On Self-Test (Mpost)
- ROM15 - Rom Parity

5.4 On-Chip Communication Including Bus-Arbitration

5.4.1 Device Interconnect

The device interconnect links multiple initiators and memory/peripherals within the device. The device interconnect logic is comprised of static initiator-selection muxes and dynamic arbiters and protocol convertors required for various bus initiators (CPU, CLA, DMA) to transact with the peripherals and memories.

5.4.1.1 Device Interconnect (Interconnect_Bridges) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- INC1 - Software Test Of Function Including Error Tests
- INC2 - Internal Watchdog (Wd)
- INC3 - External Watchdog
- INC4 - Periodic Software Read Back Of Static Configuration Registers
- INC5 - Software Read Back Of Written Configuration
- INC6 - Cpu Handling Of Illegal Operation, Illegal Results And Instruction Trapping
- INC7 - Cla Handling Of Illegal Operation And Illegal Results
- INC8 - Transmission Redundancy
- INC9 - Hardware Redundancy
- SYS8 - Eallow And Meallow Protection For Critical Registers

5.4.2 Direct Memory Access (DMA)

The direct memory access (DMA) module provides a hardware method of transferring data between peripherals and memory without intervention from the CPU, thereby freeing up bandwidth for other system functions. Additionally, the DMA has the capability to orthogonally rearrange the data as the data is transferred and *ping-pong* data between buffers. These features are useful for structuring data into blocks for improved CPU processing.

5.4.2.1 DMA Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- DMA2 - Information Redundancy Techniques
- DMA3 - Transmission Redundancy
- DMA4 - Periodic Software Read Back Of Static Configuration Registers
- DMA5 - Software Read Back Of Written Configuration
- DMA6 - Software Test Of Function Including Error Tests
- DMA7 - Dma Overflow Interrupt
- DMA8 - Access Protection Mechanism For Memories
- DMA9 - Disabling Of Unused Dma Trigger Sources

5.4.3 Enhanced Peripheral Interrupt Expander (ePIE) Module

The enhanced peripheral interrupt expander (ePIE) module is used to interface peripheral interrupts to the C28x CPU. The module provides configurable masking on a per interrupt basis. The PIE module includes a local SRAM that is used to hold the address of the interrupt handler per interrupt.

5.4.3.1 Enhanced Peripheral Interrupt Expander (PIE) Module Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- PIE2 - Software Test Of Sram
- PIE3 - Software Test Of Epie Operation Including Error Tests
- PIE4 - Periodic Software Read Back Of Static Configuration Registers
- PIE5 - Software Read Back Of Written Configuration
- PIE7 - Maintaining Interrupt Handler For Unused Interrupts
- PIE8 - Online Monitoring Of Interrupts And Events
- PIE11 - Sram Parity

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements:

- PIE12 - Software Test Of Parity Logic

5.4.4 Dual Zone Code Security Module (DCSM)

The dual code security module (DCSM) is a security feature incorporated in this device. The module prevents access and visibility to on-chip secure memories (and other secure resources) to unauthorized persons. The module also prevents duplication and reverse engineering of proprietary code. Each CPU subsystem has a dual zone CSM for code protection.

5.4.4.1 Dual Zone Code Security Module (DCSM) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- DCSM1 - Multi-Bit Enable Keys For Control Registers
- DCSM2 - Majority Voting And Error Detection Of Link Pointer
- DCSM3 - Periodic Software Read Back Of Static Configuration Registers
- DCSM4 - Software Test Of Function Including Error Tests
- DCSM5 - Software Read Back Of Written Configuration
- DCSM6 - Cpu Handling Of Illegal Operation, Illegal Results And Instruction Trapping
- DCSM8 - Vcu Crc Check Of Static Memory Contents
- DCSM9 - External Watchdog
- DCSM11 - Hardware Redundancy

5.4.5 CrossBar (X-BAR)

The crossbars (X-BAR) provide flexibility to connect device inputs, outputs, and internal resources in a variety of configurations. The device contains a total of three X-BARs: input X-BAR, output X-BAR, and ePWM X-BAR. The input X-BAR has access to every GPIO and can route each signal to any (or multiple) of the IP blocks (for example, ADC, eCAP, ePWM, and so forth). This flexibility relieves some of the constraints on peripheral muxing by simply requiring any GPIO pin to be available. The ePWM X-BAR is connected to the digital compare (DC) submodule of each ePWM module for actions such as trip zones. The GPIO output X-BAR takes signals from inside the device and brings them out to a GPIO.

5.4.5.1 CrossBar (XBAR) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- XBAR1 - Software Test Of Function Including Error Tests
- XBAR2 - Hardware Redundancy
- XBAR3 - Periodic Software Read Back Of Static Configuration Registers
- XBAR4 - Software Read Back Of Written Configuration
- XBAR5 - Software Check Of X-Bar Flag

5.4.6 Timer

Each CPU subsystem is provided with three 32-bit CPU timers (TIMER0, TIMER1, and TIMER2). The module provides the operating system (OS) timer for the device. The OS timer function is used to generate internal event triggers or interrupts, as needed, to provide periodic operations of safety-critical functions. The capabilities of the module enable the module to be used for clock monitoring as well.

5.4.6.1 CPU_Timer Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- TIM1 - 1Oo2 Software Voting Using Secondary Free Running Counter
- TIM2 - Periodic Software Read Back Of Static Configuration Registers
- TIM3 - Software Read Back Of Written Configuration
- TIM4 - Software Test Of Function Including Error Tests

5.4.7 Configurable Logic Block

The configurable logic block (CLB) is a collection of blocks that can be interconnected using software to implement custom digital logic functions or enhance existing on-chip peripherals. The CLB is able to enhance existing peripherals through a set of crossbar interconnections, which provide a high level of connectivity to existing control peripherals such as enhanced pulse width modulators (ePWM), enhanced capture modules (eCAP), and enhanced quadrature encoder pulse modules (eQEP). The crossbars also allow the CLB to be connected to external GPIO pins. In this way, the CLB can be configured to interact with device peripherals to perform small logical functions, such as comparators, or to implement custom serial data exchange protocols. Through the CLB, functions that are otherwise accomplished using external logic devices can now be implemented inside the MCU. CLB can be used to implement absolute or incremental position encoders used for motor control applications.

The CLB peripheral is configured through the CLB tool. More information on the CLB tool, available examples, application reports, and user guides are available in C2000Ware under *\utilities\clb_tool*.

5.4.7.1 Configurable Logic Block (CLB) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- CLB1 - Software Test Of Function Including Error Tests
- CLB2 - Hardware Redundancy
- CLB3 - Monitoring Of Clb By Ecap Or Eqep
- CLB4 - Periodic Software Read Back Of Static Configuration Registers
- CLB5 - Software Read Back Of Written Configuration
- CLB6 - Lock Mechanism For Control Registers
- CLB7 - Internal Watchdog (Wd)
- CLB8 - Periodic Software Read Back Of Spi Buffer Registers / Final Ram Locations Where The Data Gets Transferred Through Dma

5.5 Digital I/O

5.5.1 General-Purpose Input/Output (GPIO) and Pin Muxing

The general-purpose input output (GPIO) module provides software configurable mapping of internal module I/O functionality to device pins. These pins can be individually selected to operate as digital I/O (also called GPIO mode) or connected to one of several peripheral I/O signals.

5.5.1.1 General Pupose I O and Multiplexing (GPIO_Pinmux) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- GPIO1 - Lock Mechanism For Control Registers
- GPIO2 - Periodic Software Read Back Of Static Configuration Registers
- GPIO3 - Software Read Back Of Written Configuration
- GPIO4 - Software Test Of Function Using I/O Loopback
- GPIO5 - Hardware Redundancy

5.5.2 Enhanced Pulse Width Modulators (ePWM)

The enhanced pulse width modulator (ePWM) peripheral is a key element in digital motor control and power electronic systems. Some of the ePWM module instances support a high-resolution pulse width modulator (HRPWM) mode to improve the time resolution. For more information on the ePWM instances supporting the HRPWM mode, see the device-specific data sheet and reference manual.

5.5.2.1 Enhanced Pulse Width Modulators (ePWM) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- PWM1 - Software Test Of Function Including Error Tests
- PWM2 - Hardware Redundancy
- PWM3 - Monitoring Of Epwm By Ecap
- PWM4 - Periodic Software Read Back Of Static Configuration Registers
- PWM5 - Software Read Back Of Written Configuration
- PWM6 - Lock Mechanism For Control Registers
- PWM8 - Epwm Fault Detection Using Xbar
- PWM9 - Epwm Synchronization Check
- PWM11 - Epwm Application Level Safety Mechanism
- PWM12 - Online Monitoring Of Interrupts And Events
- PWM13 - Monitoring Of Epwm By Adc

5.5.3 High Resolution PWM (HRPWM)

The HRPWM module extends the time resolution capabilities of the conventionally derived digital pulse width modulator (PWM). HRPWM is typically used when PWM resolution falls below ≅ 9-10 bits. The HRPWM is based on micro edge positioner (MEP) technology. MEP logic is capable of positioning an edge very finely by sub-dividing one coarse system clock of a conventional PWM generator. The time step accuracy is of the order of 150ps.

5.5.3.1 High Resolution Pulse Width Modulator (OTTO) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- OTTO1 - Hrpwm Built-In Self-Check And Diagnostic Capabilities
- OTTO2 - Hardware Redundancy
- OTTO3 - Monitoring Of Epwm By Ecap
- OTTO4 - Periodic Software Read Back Of Static Configuration Registers
- OTTO5 - Software Read Back Of Written Configuration

5.5.4 Enhanced Capture (eCAP)

The enhanced capture (eCAP) module provides input capture functionality for systems where accurate timing of external events is important. The eCAP module features include speed measurements of rotating machinery (for example, toothed sprockets sensed through Hall sensors), elapsed time measurements between position sensor pulses, period and duty cycle measurements of pulse train signals, and decoding current or voltage amplitude derived from duty cycle encoded current and voltage sensors.

Note

Use of a sensorless positioning algorithm can provide information redundancy through plausibility checking of eCAP results.

5.5.4.1 Enhanced Capture (ECAP) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- CAP1 - Software Test Of Function Including Error Tests
- CAP2 - Information Redundancy Techniques
- CAP4 - Periodic Software Read Back Of Static Configuration Registers
- CAP5 - Software Read Back Of Written Configuration
- CAP6 - Ecap Application Level Safety Mechanism
- CAP7 - Hardware Redundancy

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements:

- CAP3 - Monitoring Of Epwm By Ecap

5.5.5 High Resolution Capture (HRCAP)

The high-resolution capture (HRCAP) peripheral measures the width of external pulses with a typical resolution within hundreds of picoseconds. This module includes capture channel in addition to a hardware calibration block to enable continuous on-line calibration, this drastically reduces software overhead to calibrate. HRCAP input can be connected to HRPWM output using X-BAR to enable periodic testing. The HRCAP enhancement has been added to eCAP 6 and eCAP 7.

5.5.5.1 High Resolution Capture (HRCAP) Safety Features List

5.5.6 Enhanced Quadrature Encoder Pulse (eQEP)

The enhanced quadrature encoder pulse (eQEP) module is used for direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine for use in a high-performance motion and position-control system. The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

Note

Use of a sensorless positioning algorithm can provide information redundancy through plausibility checking of eQEP results.

5.5.6.1 Enhanced Quadrature Encoder Pulse (eQEP) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- QEP1 - Software Test Of Function Including Error Tests
- QEP2 - Eqep Quadrature Watchdog
- QEP3 - Information Redundancy Techniques
- QEP4 - Periodic Software Read Back Of Static Configuration Registers
- QEP5 - Software Read Back Of Written Configuration
- QEP6 - Eqep Application Level Safety Mechanism
- QEP7 - Hardware Redundancy
- QEP8 - Qma Error Detection Logic

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements:

- QEP9 - Eqep Software Test Of Quadrature Watchdog Functionality

5.5.7 External Interrupt (XINT)

Interrupts from external sources can be provided to the device using GPIO pins with the help of XINT module. The module allows configuring the GPIOs to be selected as interrupt sources. The polarity of the interrupts can also be configured with this module.

5.5.7.1 XINT Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- XINT1 - Software Test Of Function Including Error Tests
- XINT2 - Periodic Software Read Back Of Static Configuration Registers
- XINT3 - Software Read Back Of Written Configuration
- XINT4 - Hardware Redundancy

5.6 Analog I/O

5.6.1 Analog-to-Digital Converter (ADC)

The analog-to-digital converter (ADC) module is used to convert analog inputs into digital values. Results are stored in internal registers for later transfer by CLA, DMA or CPU. The C2000 MCU device family products implement up to four modules with shared channels used for fast conversion (ping-pong method).

Note

- ADC module voltages must be supervised as noted in the device-specific data sheet.
- To reduce the probability of common mode failure, consider implementing multiple channels (information redundancy) using non adjacent pins and a different voltage reference.

5.6.1.1 ADC Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- ADC1 - Software Test Of Function Including Error Tests
- ADC2 - Dac To Adc Loopback Check
- ADC3 - Adc Information Redundancy Techniques
- ADC4 - Opens/Shorts Detection Circuit For Adc
- ADC5 - Software Read Back Of Written Configuration
- ADC6 - Periodic Software Read Back Of Static Configuration Registers
- ADC7 - Adc Signal Quality Check By Varying Acquisition Window
- ADC8 - Adc Input Signal Integrity Check
- ADC9 - Monitoring Of Epwm By Adc
- ADC10 - Hardware Redundancy

5.6.2 Buffered Digital-to-Analog Converter (DAC)

The buffered DAC module consists of an internal reference DAC and an analog output buffer that is capable of driving an external load. An integrated pulldown resistor on the DAC output helps to provide a known-pin voltage when the output buffer is disabled. Software writes to the DAC value register can take effect immediately or can be synchronized with PWMSYNC events.

5.6.2.1 BufDAC Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- DAC1 - Software Test Of Function Including Error Tests
- DAC2 - Dac To Adc Loopback Check
- DAC3 - Lock Mechanism For Control Registers
- DAC4 - Software Read Back Of Written Configuration
- DAC5 - Periodic Software Read Back Of Static Configuration Registers
- DAC6 - Hardware Redundancy
- DAC7 - Dac To Comparator Loopback Check

5.6.3 Comparator Subsystem (CMPSS)

The comparator subsystem (CMPSS) consists of analog comparators and supporting components that are combined into a topology that is useful for power applications such as peak-current mode control, switched-mode power, power factor correction, and voltage trip monitoring. The comparator subsystem is built around a pair of analog comparators and helps with the detection of signal exception conditions including high and low thresholds. The positive input of the comparator is always driven from an external pin, but the negative input can be driven by either an external pin or by an internal, programmable, 12-bit DAC. Each comparator output passes

through a programmable digital filter that can remove spurious trip signals. A ramp generator circuit is optionally available to control the internal DAC value for one comparator in the subsystem.

5.6.3.1 Comparator Subsystem (CMPSS) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- CMPSS1 - Software Test Of Function Including Error Tests
- CMPSS3 - Hardware Redundancy
- CMPSS4 - Software Read Back Of Written Configuration
- CMPSS5 - Periodic Software Read Back Of Static Configuration Registers
- CMPSS6 - Lock Mechanism For Control Registers
- CMPSS8 - Cmpss Ramp Generator Functionality Check

5.6.4 Programmable Gain Amplifier (PGA)

The programmable gain amplifier (PGA) is used to amplify an input voltage for the purpose of increasing the dynamic range of the downstream ADC and CMPSS modules. The integrated PGA helps to reduce cost and design effort for many control applications that traditionally require external, standalone amplifiers. Software selectable gain and filter settings make the PGA adaptable to various performance needs.

5.6.4.1 Programmable Gain Amplifier (PGA) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- PGA1 - Pga To Adc Loopback Test
- PGA2 - Hardware Redundancy
- PGA3 - Software Read Back Of Written Configuration
- PGA4 - Periodic Software Read Back Of Static Configuration Registers
- PGA6 - Lock Mechanism For Control Registers

5.7 Data Transmission

5.7.1 Controller Area Network (MCAN, CAN FD)

The controller area network (MCAN) interface provides medium throughput networking with event-based triggering, compliant to the CAN and CAN FD (flexible data-rate) protocols. The MCAN modules require an external transceiver to operate on the CAN network. The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

5.7.1.1 MCAN Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- MCAN1 - Software Test Of Function Using I/O Loopback
- MCAN2 - Information Redundancy Techniques Including End-To-End Safing
- MCAN3 - Periodic Software Read Back Of Static Configuration Registers
- MCAN4 - Software Read Back Of Written Configuration
- MCAN5 - Transmission Redundancy
- MCAN6 - Pwm Trip By Mcan
- MCAN7 - Software Test Of Sram
- MCAN8 - Sram Ecc
- MCAN9 - Bit Multiplexing In Sram Memory Array
- MCAN10 - Mcan Stuff Error Detection
- MCAN11 - Mcan Form Error Detection
- MCAN12 - Mcan Acknowledge Error Detection
- MCAN13 - Bit Error Detection

- MCAN14 - Crc In Message
- MCAN16 - Timeout On Fifo Activity
- MCAN17 - Timestamp Consistency Checks
- MCAN18 - Tx-Event Checks
- MCAN19 - Interrupt On Message Ram Access Failure
- MCAN20 - Software Test Of Function Including Error Tests Using Epg

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements:

- MCAN15 - Software Test Of Ecc Logic

5.7.2 Serial Peripheral Interface (SPI)

The serial peripheral interface (SPI) modules provide serial I/O compliant to the SPI protocol. SPI communications are typically used for communication to smart sensors and actuators, serial memories, and external logic such as a watchdog device.

5.7.2.1 Serial Peripheral Interface (SPI) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- SPI1 - Software Test Of Function Using I/O Loopback
- SPI2 - Information Redundancy Techniques Including End-To-End Safing
- SPI3 - Periodic Software Read Back Of Static Configuration Registers
- SPI4 - Software Read Back Of Written Configuration
- SPI5 - Transmission Redundancy
- SPI6 - Spi Data Overrun Detection
- SPI7 - Hardware Redundancy
- SPI8 - Software Test Of Function Including Error Tests Using Epg

5.7.3 Serial Communication Interface (SCI)

The module provides serial I/O capability for typical asynchronous serial communication interface (SCI) protocols, such as UART. Depending on the serial protocol used, an external transceiver can be necessary.

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

5.7.3.1 Serial Communications Interface (SCI) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- SCI1 - Software Test Of Function Using I/O Loopback
- SCI2 - Parity In Message
- SCI3 - Information Redundancy Techniques Including End-To-End Safing
- SCI4 - Sci Overrun Error Detection
- SCI5 - Sci Break Error Detection
- SCI6 - Sci Frame Error Detection
- SCI7 - Periodic Software Read Back Of Static Configuration Registers
- SCI8 - Software Read Back Of Written Configuration
- SCI9 - Transmission Redundancy
- SCI10 - Hardware Redundancy
- SCI11 - Software Test Of Function Including Error Tests Using Epg

5.7.4 Inter-Integrated Circuit (I2C)

The inter-integrated circuit (I2C) module provides a multi-initiator serial bus compliant to the I2C protocol. The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

5.7.4.1 Inter-Integrated Circuit (I2C) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- I2C1 - Software Test Of Function Using I/O Loopback
- I2C2 - I2C Data Acknowledge Check
- I2C3 - Information Redundancy Techniques Including End-To-End Safing
- I2C4 - Periodic Software Read Back Of Static Configuration Registers
- I2C5 - Software Read Back Of Written Configuration
- I2C6 - Transmission Redundancy
- I2C7 - I2C Access Latency Profiling Using On-Chip Timer
- I2C9 - Software Test Of Function Including Error Tests Using Epg

5.7.5 Fast Serial Interface (FSI)

The fast serial interface (FSI) is a serial peripheral capable of reliable and high-speed communication. The FSI architecture specifically provides reliable and high-speed communication for those system scenarios involving communication across isolation devices. The FSI consists of independent transmitter (FSITX) and receiver (FSIRX) cores. The FSITX and FSIRX cores are configured and operated independently.

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

5.7.5.1 Fast Serial Interface (FSI) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- FSI1 - Software Test Of Function Using I/O Loopback
- FSI2 - Information Redundancy Techniques Including End-To-End Safing
- FSI3 - Periodic Software Read Back Of Static Configuration Registers
- FSI4 - Software Read Back Of Written Configuration
- FSI5 - Transmission Redundancy
- FSI6 - Fsi Data Overrun/Underrun Detection
- FSI7 - Fsi Frame Overrun Detection
- FSI8 - Fsi Crc Framing Checks
- FSI9 - Fsi Ecc Framing Checks
- FSI10 - Fsi Frame Watchdog
- FSI11 - Fsi Rx Ping Watchdog
- FSI12 - Fsi Tag Monitor
- FSI13 - Fsi Frame Type Error Detection
- FSI14 - Fsi End Of Frame Error Detection
- FSI15 - Fsi Register Protection Mechanisms
- JTAG1 - Hardware Disable Of Jtag Port

5.7.6 Power Management Bus Module (PMBus)

The PMBus module provides an interface between the microcontroller and devices compliant with the *SMI Forum PMBus Specification* Part I version 1.0 and Part II version 1.1. PMBus is based on SMBus, which uses a similar physical layer to I2C.This module supports both initiator and target modes.

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a Specific function):

5.7.6.1 Power Management BUS (PMBUS) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- PMBUS2 - I2C Data Acknowledge Check
- PMBUS3 - Information Redundancy Techniques Including End-To-End Safing
- PMBUS4 - Periodic Software Read Back Of Static Configuration Registers
- PMBUS5 - Software Read Back Of Written Configuration
- PMBUS6 - Transmission Redundancy
- PMBUS7 - Pmbus Protocol Crc In Message
- PMBUS8 - Clock Timeout

5.7.7 Local Interconnect Network (LIN)

The supported LIN module is compliant to the LIN 2.1 protocol specification. This module can be programmed to work either as an SCI or as an LIN. The hardware features of the SCI are augmented to achieve LIN functionality. The SCI module is a universal asynchronous receiver-transmitter (UART) that implements the standard non-return to zero format. The SCI can be used to communicate, for example, through an RS-232 port or over a K line.

5.7.7.1 Local Interconnect Network (LIN) Safety Features List

The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- LIN1 - Software Test Of Function Using I/O Loopback
- LIN2 - Information Redundancy Techniques Including End-To-End Safing
- LIN3 - Transmission Redundancy
- LIN4 - Periodic Software Read Back Of Static Configuration Registers
- LIN5 - Software Read Back Of Written Configuration
- LIN6 - Data Parity Error Detection
- LIN7 - Sci Overrun Error Detection
- LIN8 - Sci Frame Error Detection
- LIN9 - Lin Physical Bus Error Detection
- LIN10 - Lin No-Response Error Detection
- LIN11 - Bit Error Detection
- LIN12 - Lin Checksum Error Detection
- LIN13 - Lin Id Parity Error Detection
- LIN15 - Sci Break Error Detection
- LIN16 - Communication Access Latency Profiling Using On-Chip Timer
- LIN17 - Software Test Of Function Including Error Tests Using Epg

5.8 Not Safety Related Elements

The following elements are not recommended for use in safety-related applications using the TMS320F28P55x MCU. If used in the end system, applicable measures listed in section *Suggestions for Improving Freedom From Interference* must be implemented to avoid a cascading failure from these elements adversely affecting implemented safety functions.

- Universal Serial Bus (USB)

Chapter 6

Management of Random Faults

For a functional safety critical development, it is necessary to manage both systematic and random faults. The TMS320F28P55x component architecture includes many functional safety mechanisms, which can detect and respond to random faults when used correctly. This section of the document describes the architectural functional safety concept for each sub-block of the TMS320F28P55x component. The system integrator shall review the recommended functional safety mechanisms in the functional safety analysis report (FMEDA) in addition to this safety manual to determine the appropriate functional safety mechanisms to include in their system. The component data sheet or technical reference manual are useful tools for finding more specific information about the implementation of these features.

6.1 Fault Reporting

The TMS320F28P55x MCU product architecture provides different levels of fault indication from internal safety mechanisms using CPU interrupt, non maskable interrupt (NMI), assertion of ERRORSTS pin, assertion of CPU input reset and assertion of warm reset (XRSn). The fault response is the action that is taken by the TMS320F28P55x MCU or system when a fault is indicated. Multiple potential fault responses are possible during a fault indication. The system integrator is responsible for determining which fault response is taken to verify consistency with the system safety concept. The fault indication, ordered in terms of severity (device power down being the most severe), is shown in Figure 6-1.

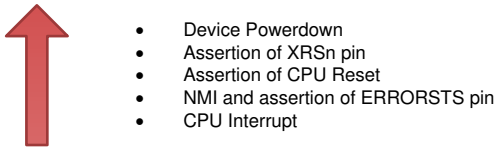

Figure 6-1. Fault Response Severity

- **Device Power Down**: This is the highest priority fault response where the external component (refer to Section 4.2.3.1) detects malfunctioning of the device or other system components and powers down the TMS320F28P55x MCU. From this state, re-entering cold boot to attempting recovery is possible.
- **Assertion of XRSn**: The XRSn reset can be generated from an internal or external monitor that detects a critical fault having the potential to violate a safety goal. Internal sources generate this fault response when the TMS320F28P55x MCU is not able to handle the internal fault condition solely (for example, CPU1 is not able to handle NMI solely). From this state,re-entering cold boot and attempting recovery is possible.
- **Assertion of CPU Reset**: CPU reset changes the state of the CPU from pre-operational or operational state to warm boot phase. The CPU Reset is generated from an internal monitor that detects any security violations. On a properly working system, the security violations can be the secondary effect due to a fault condition. From this state, re-entering warm boot phase and attempting recovery is possible.
- **Non Maskable Interrupt (NMI) and Assertion of ERRORSTS Pin**: C28x CPU supports a non maskable interrupt (NMI), which has a higher priority than all other interrupts. Each CPU subsystem is equipped with a NMIWD module responsible for generating NMI to the CPU. ERRORSTS pin is asserted along with NMI. Depending on the system-level requirements, the fault can be handled either internal to the TMS320F28P55x MCU using software or at the system-level using the ERRORSTS pin information.
- **CPU Interrupt**: CPU interrupt allows events external to the CPU to generate a program sequence context transfer to an interrupt handler where software has an opportunity to manage the fault. The peripheral

interrupt expansion (PIE) block multiplexes multiple interrupt sources into a smaller set of CPU interrupt inputs.

6.2 Suggestions for Improving Freedom From Interference

The following techniques and safety measures can be useful for improving independence of function when using the TMS320F28P55x real-time MCU:

1. Hold peripheral clocks disabled if the available peripherals are unused (CLK14-Peripheral Clock Gating (PCLKCR)).
2. Hold peripherals in reset if the available peripherals are unused (SYS7-Peripheral Soft Reset (SOFTPRES)).
3. Power down the analog component cores if the cores are not used.
4. When possible, separate critical I/O functions by using non adjacent I/O pins and balls.
5. Partition the memory, as per the application requirements, to the respective processing units and configure the Access Protection Mechanism for Memories for each memory instance, so that only the permitted initiators have access to memory.
6. The Dual Zone Code Security Module (DCSM) can be used for functional safety as a firewall to protect shared memories; where functions with different safety integrity levels can be executed from different security zones (zone1, zone2, and unsecured zone), this mitigates risks originating due to interference among memories.
7. Disabling of the SOC inputs to the ADC can help avoid interference from unused peripherals that disturb functionality of the ADC. Disabling of unused DMA trigger sources helps minimize interference caused by unintentional DMA transfers.
8. Disabling unused CLA task trigger sources and disabling unused DMA trigger sources mitigates risks of interference caused due to the trigger events.
9. To avoid interference from unintentional activity on the debug port of the MCU, JTAG1-*Hardware Disable of JTAG Port* is helpful in preventing this interference.
10. Safety applications running on the CPU can be interfered with by unintentional faulty interrupt events to PIE modules. PIE7-*Maintaining Interrupt Handler for unused interrupts* and PIE8- *Online Monitoring of Interrupts and Events* detect such interfering failures.
11. MCU resources in supporting CPU execution, such as memory, interrupt controller, and so forth, can be impacted by resources from lower safety-integrity safety functions coexisting on the same MCU. Safety mechanisms, such as CPU9-*External watchdog*, SRAM16 –*Information Redundancy Techniques* and SRAM17-*CPU handling of Illegal Operation, Illegal Results and Instruction Trapping*, are able to detect such interference.
12. Critical configuration registers can be victim to interference from bus initiators on the MCU which implements lower safety-integrity functions. These registers can be protected by SYS1-*Multibit Enable Keys for Control Registers*, SYS2-*Lock Mechanism for Control Registers*, and SYS8-*EALLOW and MEALLOW Protection for Critical Registers*.

6.3 Suggestions for Addressing Common Cause Failures

The system integrator must execute an independent failure or common cause failure analysis to consider possible dependent or common cause failures on the sub-elements of the TMS320F28P55x real-time MCU, including pin-level connections.

- Consider a relevant list of dependent failure initiators, such as the lists found in ISO 26262-11:2018. Analysis of dependent failures must include common cause failures among functionally redundant parts and also between functions and the respective safety mechanisms.
- Verify that the dependent failure analysis considers the impact of the software tasks running on the TMS320F28P55x MCU, including hardware and software interactions.
- Verify that the dependent failure analysis considers the impact of pin and ball level interactions on the TMS320F28P55x MCU package, including aspects related to the selected I/O multiplexing.

The following can be useful for addressing the common cause failures when using the C2000 MCU:

- Redundant functions and safety mechanism can be impacted by common power failure. A common cause failure on power source can be detected by PWR1-*External voltage supervisor* and PWR2-*External Watchdog*.
- In general, a clock source that is common to redundant functions is monitored and any failures on the same clock source can be detected by safety mechanisms such as CLK1-*Missing Clock Detect*, CLK2-*Clock Integrity Check using CPU Timer*, CLK5-*External monitoring of clock via XCLKOUT*, and CLK8-*Periodic Software Read Back of Static Configuration Registers*. Specifically, to avoid common-clock failures affecting internal watchdog (WD) and CPU, TI recommends using either INTOSC2 or X1/X2 as clock source to PLL.
- Failure of common reset signals to redundant functions can be detected by RST1-*External monitoring of warm reset (XRSn)* and RST2-*Reset Cause Information*.
- Common cause failures on interconnect logic can impact both redundant functions and safety mechanisms in the same way. In addition to other safety mechanisms, INC1-*Software Test of Function Including Error Tests* can be implemented to detect faults on interconnect logic.
- Common cause failures can impact two functions used in a redundant way. In cases of communication peripherals that are module specific, *Information Redundancy Techniques Including End to End Safing* can be implemented to detect common cause failures, for example, CAN2, SPI2, SCI3, I2C3, and MCBSP2.
- Using different voltage references and SOC trigger sources for ADC (refer to Section 6.4.4.9)
- Using PWM modules from different sync groups for implementing hardware redundancy
- Using GPIO pins from different groups when implementing hardware redundancy for GPIO pins

6.4 Descriptions of Functional Safety Mechanisms

This section provides a brief summary of the diagnostic mechanisms available on the TMS320F28P55x MCU device family. The diagnostic mechanisms are arranged as per the device portioning given in Figure 5-1. At places where the safety mechanism is applicable for more than one component, the safety mechanism is placed at an appropriate place based on the applicable use-case scenario. For a detailed description or implementation details for a diagnostic, refer to the device-specific technical reference manual.

6.4.1 C2000 MCU Infrastructure Components

6.4.1.1 External Voltage Supervisor

Texas Instruments highly recommends the use of an external voltage supervisor to monitor all voltage rails (VDDIO, VDDA, and VDD). The voltage supervisor must be configured with overvoltage and undervoltage thresholds within the recommended operating conditions of the target device as noted in the device-specific data sheet. Error response, diagnostic testability, and any necessary software requirements are defined by the external voltage supervisor selected by the system integrator.

6.4.1.2 External Watchdog

External watchdog helps to reduce common mode failures, as external watchdog utilizes clock, reset, and power that are separate from the system being monitored. Error response, diagnostic testability, and any necessary software requirements are defined by the external watchdog selected by the system integrator.

Texas Instruments highly recommends the use of an external watchdog in addition to the internally provided watchdogs. An internal or external watchdog can provide indication of an inadvertent activation of logic that results in an impact to safety-critical execution. Any watchdog added externally must include a combination of temporal and logical monitoring of program sequence [IEC61508-7, clause A.9.3] or other appropriate methods so that high diagnostic effectiveness can be claimed.

6.4.1.3 Multibit Enable Keys for Control Registers

This module includes features to support avoidance of unintentional control register programming. Implementation of multibit keys for critical control registers is one such feature (for example, EPWM_REGS, EPWMLOCK, and so forth). The multibit keys are particularly effective for avoiding unintentional activation. For more details on the registers for which the diagnostic is applicable, refer to the device-specific technical reference manual. The operation of this safety mechanism is continuous and cannot be altered by the software. This mechanism can be tested by generating software transactions with and without correct keys and observing the updated register value.

6.4.1.4 Lock Mechanism for Control Registers

The module contains a lock mechanism for protection of critical control registers. Once the associated LOCK register bits are set, the write access to the registers are blocked. Locked registers cannot be updated by software; once locked, only reset can unlock the registers.

6.4.1.5 Software Readback of Written Configuration

To verify proper configuration of memory-mapped registers in this module, TI recommends that software implements a test to confirm proper configuration of all control registers by reading back the contents. This test also provides diagnostic coverage for the peripheral bus interface and peripheral interconnect bridges.

Since CLA configuration registers are accessible to the C28x CPU only, this safety mechanism for the CLA module must be executed by C28x CPU.

6.4.1.6 Periodic Software Readback of Static Configuration Registers

Configuration registers are typically configured in the beginning and hold the value until the particular task execution. Periodic readback of configuration registers can provide a diagnostic for inadvertent writes or disruption of these registers.

The diagnostic coverage can be improved by extending the test to include readback of the flag registers that are expected to remain constant (for example, PLL lock status, EQEP phase error flag, and so forth) during the device operation as well. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

The diagnostic coverage of some peripherals can be further enhanced by applying some module-specific tests as follows:

- For improving the enhanced peripheral interrupt expander (EPIE) coverage, the PIE flag registers can be periodically checked to verify that all pending interrupts are serviced by reading the PIE flag registers (PIE_CTRL_REGS.PIEIFRx.all) and the peripheral interrupt flag registers.
- While serving the interrupt, the ISR routine can check for flag settings in peripheral and PIE to verify that correct interrupt is being serviced.

Since CLA configuration registers are accessible to C28x CPU only, this safety mechanism for the CLA module has to be executed by C28x CPU.

6.4.1.7 Online Monitoring of Temperature

The internal temperature sensor measures the junction temperature of the device. The output of the sensor can be sampled with the ADC through an internal connection. This internal connection can be enabled on channel ADCIN13 on ADCA by setting the ENABLE bit in the TSNSCTL register.

The micro edge positioning (MEP) block of HRPWM Built-In Self-Check and Diagnostic Capabilities can also be used to detect variations in temperature and voltage.

6.4.1.8 EALLOW and MEALLOW Protection for Critical Registers

EALLOW (CPU, DMA) and MEALLOW (CLA) protection enables write access to emulation and other protected registers. CPU (CLA) can set this bit using EALLOW (MEALLOW) instruction and cleared using EDIS (MEDIS) instruction. The protection can be used to prevent data being written to the wrong place, which happens with conditions like boundary exceeding, incorrect pointers, stack overflow or corruption, and so forth. Reads from the protected registers are always allowed. TI recommends issuing an EDIS (or MEDIS) for protection once write for the protected registers are complete.

6.4.1.9 Internal Watchdog (WD)

The internal watchdog has two modes of operation: normal watchdog (WD) and windowed watchdog (WWD). The system integrator can select to use one mode or the other but not both at the same time. For details of programming the internal watchdogs, refer to the device-specific technical reference manual. The WD is a traditional single threshold watchdog. The user programs a timeout value to the watchdog and must provide a predetermined WDKEY to the watchdog before the timeout counter expires. Expiration of the timeout counter

or an incorrect WDKEY triggers an error response. The WD can issue either a warm system reset or a CPU maskable interrupt upon detection of a failure. The WD is enabled after reset.

In case of WWD, the user programs an upper bound and lower bound to create a time window during which the software must provide a predetermined WDKEY to the watchdog. Failure to receive the correct response within the time window or an incorrect WDKEY triggers an error response. The WWD can issue either a warm system reset or a CPU maskable interrupt upon detection of a failure. Normal WD operation is enabled by default after reset. Additional configuration must be performed to enable the WWD operation. For details of programming the internal watchdogs, refer to the device-specific technical reference manual. The use of the time window allows detection of additional clocking failure modes as compared to the WD implementation.

6.4.1.10 Brownout Reset (BOR)

An internal BOR circuit monitors the VDDIO rail for dips in voltage which result in the supply voltage dropping out of operational range. When the VDDIO voltage drops below the BOR threshold, the device is forced into reset and XRSn is pulled low. XRSn remains in reset until the voltage returns to the operational range. The BOR is enabled by default.

6.4.1.11 Missing Clock Detect (MCD)

The missing clock detector (MCD) is a safety diagnostic that can be used to detect a failure of the PLL reference clock. MCD utilizes the embedded 10MHz internal oscillator (INTOSC1). This circuit only detects complete loss of PLL reference clock and does not do any detection of frequency drift. The MCD circuit is enabled by default during the power-on reset state. The diagnostic can be disabled through software.

6.4.1.12 Clock Integrity Check Using CPU Timer

The CPU timer module can be used to detect incorrect clock frequencies and drift between clock sources. CPU Timer 2 has a programmable counter whose prescale value and clock source can be selected. The frequency relationship between selected clock and system clock can be determined using the system clock as a reference time base. For more information on the clock selection options implemented, refer to the device-specific data sheet. Higher diagnostic coverage can be obtained by setting tighter bounds when checking clock integrity using Timer 2. Common cause failures can be reduced by using different clock sources and different prescale values for the reference clock and measured clock. The timer diagnostic is not enabled by default and must be enabled through software. The cyclical check applied by the timer module provides an inherent level of self-checking (auto-coverage), which can be considered for application in latent fault diagnostics.

6.4.1.13 Clock Integrity Check Using HRPWM

Calibration logic of OTTO (HRPWM) can be used to detect incorrect system clock (SYSCLK) frequencies. The clock whose frequency is being measured is configured as the system clock and the auto-calibration function is executed. The result obtained from the calibration function can be checked against the predetermined range of values to detect incorrect clock frequency or frequency drift. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.1.14 External Clock Monitoring Using XCLKOUT

The TMS320F28P55x MCU device family provides a capability to export select internal clocking signals for external monitoring. This feature can be configured using software by programming registers in the system control module. To determine the number of external clock outputs implemented and the register mapping of internal clocks that can be exported, refer to the device-specific data sheet. Export of internal clocks on the XCLKOUT outputs is not enabled by default and must be enabled using software. Disabling and configuring this diagnostic using software is possible.

6.4.1.15 Software Test of Watchdog (WD) Operation

A basic test of the internal watchdog operation can be performed using software, including checking error responses, by configuring the expected lower and higher threshold values for servicing WDKEY followed by servicing or not servicing the WDKEY during the programmed threshold values. If reset is detrimental to the system operation, the test can be performed by configuring the internal watchdog in Interrupt mode (SCSR.WDENINT) and reverting back to reset mode after completion of the test.

6.4.1.16 Software Test of Missing Clock Detect Functionality

Proper operation of missing clock detect (MCD) functionality can be checked by configuring MCDCR.OSCOFF. The diagnostic test can check for issues of missing clock NMI and setting of missing clock status flag (MCDCR.MCLKSTS).

6.4.1.17 PLL Lock Profiling Using On-Chip Timer

Clock setup for the MCU device family includes selecting the appropriate clock source, configuring the PLL multiplier, waiting for the lock status, and switching the clock to the PLL output once the internal lock status is set. The time required for the PLL lock sequence can be profiled using an on-chip timer to detect faults in the PLL wrapper logic. Once the PLL is locked, the frequency of the output clock can be checked by using the following:

- Dual Clock Comparator (DCC)
- External Clock Monitoring Using XCLKOUT to verify proper clock output
- Clock Integrity Check Using CPU Timer
- Clock Integrity Check Using HRPWM

6.4.1.18 Peripheral Clock Gating (PCLKCR)

Peripherals can be clock gated on a per peripheral basis. This clock gate can be utilized to disable unused features so the features cannot interfere with active safety functions. This safety mechanism is enabled after reset. Software must configure and disable this mechanism to use a particular peripheral. Locking the particular configuration to avoid inadvertent writes is possible.

6.4.1.19 Dual Clock Comparator (DCC) – Type 2

The dual-clock comparator module can be used to validate or monitor the output frequency of the PLL (PLLRAWCLK) over a defined time window. While checking for the PLL clock frequency, DCC uses a known good reference clock to compare with, which is INTOSC1, INTOSC2, or XTAL. If the PLL clock frequency deviates from the targeted frequency more than a predefined threshold, DCC reports an ERROR status flag and sends an interrupt to the PIE.

Proper operation of dual clock comparator (DCC) functionality can be checked by configuring DCC with a wrong ratio between counter 0 (DCCCNTSEED0) and counter 1 (DCCCNTSEED1) to force a failure. The fail flag interrupt can then be checked to verify the functionality of DCC.

6.4.1.20 Clock Integrity Check Using DCC

One or more dual clock comparators (DCCs) are implemented as multipurpose safety diagnostics. The DCC can be used to detect incorrect frequencies and drift between clock sources. The DCC is composed of two counter blocks; one is used as a reference timebase and a second is used for the clock under test. Both reference clock and clock under test can be selected using software, as can the expected ratio of clock frequencies. Deviation from the expected ratio generates an error indication to the ESM. For more information on the clock selection options implemented, refer to the device-specific data sheet. For DCC programming details, refer to the TRM.

6.4.1.21 PLL Lock Indication

PLL lock functionality is implemented by comparing the difference (error) between the feedback clock and reference clock through a phase frequency detector (PFD). When PLL is in lock and generating the correct frequency, the difference is <100pS≅300pS. Once there is any fault causing the PLL output frequency to drift, the difference goes outside of that range. In such a case, PLL lock signal goes from 1 to 0; indicating PLL is out of lock. DCC can be used to detect that drift has occurred.

6.4.1.22 Software Test of DCC Functionality Including Error Tests

A basic test of DCC functionality (including error generation) is possible using software by programming a sequence of good and bad expected clock ratios and executing DCC operations with software confirming expected results.

6.4.1.23 External Clock Monitoring Using XCLKOUT

The TMS320F28P55x MCU device family provides a capability to export select internal clocking signals for external monitoring. This feature can be configured using software by programming registers in the system control module. To determine the number of external clock outputs implemented and the register mapping of internal clocks that can be exported, refer to the device-specific data sheet. Export of internal clocks on the XCLKOUT outputs is not enabled by default and must be enabled using software. Disabling and configuring this diagnostic using software is possible.

6.4.1.24 Software Test of PLL Functionality Including Error Tests

APLL lock indication functionality can be checked by using a CPU timer and the user-defined software. Timer can be configured for a fixed number of cycles, before which, APLL is expected to get locked. In cases where APLL does not get locked before the timer expires (that is, taking more cycles than expected), the timer interrupt is triggered to CPU and further action can be taken by the user-defined software. To verify the correctness of APLL clock, and hence, the system clock generation, TI recommends using the standard clock sources for timer module (like INTOSC, crystal, and so forth) instead of system clock for checking the APLL clock generation correctness.

6.4.1.25 Interleaving of FSM States

Main control FSM includes a hamming distance of 2 to prevent any single bit flip from causing the state machine to transition into another valid state. The FSM defaults to IDLE/INIT state in case of any single bit flip. Since the PLLEN and other control bits from the SYSCTRL remain valid, the FSM (is expected to) relocks and continues. No error is generated for the bit fail scenario.

6.4.1.26 External Monitoring of Warm Reset (XRSn)

The XRSn warm reset signal is implemented as an open drain I/O pin. An external monitor can be utilized to detect expected or unexpected changes to the state of the internal, warm-reset, control signal and maintains proper signaling (for example, low duration) when asserted. Error response, diagnostic testability, and any necessary software requirements are defined by the external monitor selected by the system integrator.

6.4.1.27 Reset Cause Information

The system control module provides a status register (RESC) that latches the cause of the most recent reset event. Application software executed during boot-up can check the status of this register to determine the cause of the last reset event. This information can be used by the software to identify the cause and manage failure recovery if required.

6.4.1.28 Glitch Filtering on Reset Pins

Glitch filters are implemented on functional and JTAG reset of the device. These structures filter out noise and transient signal spikes on the input reset pins to reduce unintended activation of the reset circuitry. The glitch filters are enabled by default and operate continuously. The behavior of the glitch filters cannot be changed by the software.

6.4.1.29 NMIWD Shadow Registers

The use of a two-stage cold and warm reset scheme on the device allows the implementation of NMIWD shadow registers. Shadow registers are reset only by power-on and cold reset. These registers are used to store the NMIFLG information before reset assertion. The NMIFLG information can be used by the application software to provide additional information on the NMI status of the device before the last warm reset operation.

6.4.1.30 NMIWD Reset Functionality

On receiving an NMI, the software can attempt recovery from the NMI condition. Based on the severity and type of the fault condition, recovery is potentially not always successful. In such situations, an additional protection is provided by having an independent watchdog monitoring the NMI recovery. If the attempted recovery is not successful, a reset is issued. The timeout for reset can be configured (using NMIWDPRD) based on the FTTI of the device.

6.4.1.31 Peripheral Soft Reset (SOFTPRES)

Peripherals can be kept in reset on a *per peripheral* basis. This can be utilized to reset the unused features so the features cannot interfere with active safety functions. These safety mechanisms are disabled after reset. Software must configure and enable these mechanisms.

6.4.1.32 Software Test of ERRORSTS Functionality

As indicated in Figure 4-8, the ERRORSTS pin is an integral part of the MCU safety concept; used for indicating to an external system a critical error is occurring within in the MCU. Proper functioning of the ERRORSTS pin, and error handling of the system external to the MCU, can be checked by asserting the ERRORSTS pin. This ERRORSTS pin is asserted by generating an error condition using one of the provided software options (for example, asserting CLOCLKFAIL NMIFLG by updating the NMIFLGFRC.bit.CLOCKFAIL). Error response, diagnostic testability, and any necessary system requirements are defined by the system integrator.

6.4.1.33 Software Test of Reset – Type 1

A software test for detecting basic functionality, errors for reset sources, and reset logic can be implemented. Each of the reset sources (including peripheral resets, DEV_CFG_REGS.SOFTPRESx)—except POR—can be generated internally, and the basic reset functionality can be checked by verifying that the reset-cause register setting is correct and that only the intended logic is reset. Additionally, the SIMRESET configuration for SYSRS or XRS assertion can be used for this test through software write.

To confirm if individual peripherals have received the reset correctly, software can run a peripheral-specific test of functionality and confirm the expected state of the peripheral after reset. Depending on the complexity of the peripheral, this software test of functionality can include a test of the complex features of the peripheral, including error tests necessary to confirm correct propagation of reset. For a peripheral-specific software test of function, including error tests, refer to the device-specific safety mechanism listed for the peripheral.

6.4.1.34 Peripheral Access Protection – Type 1

Peripheral access protection is a fault avoidance measure to block unintended accesses from each initiator. Each module has a configuration to control the type of accesses to be serviced from each initiator (CPU, CLA, and DMA). After programming peripheral-access protection registers, each initiator can exclusively control the peripheral to safeguard usage by a particular application against errant writes or corruption by other initiators in the system. This feature is enabled using the dedicated access control bits, per peripheral, that allow or protect against the access from a given initiator. Each peripheral has two bit qualifiers, per initiator, to decode the access allowed. For more details, refer to the PERIPH_AC_REGS registers in the device technical reference manual.

6.4.1.35 Hardware Disable of JTAG Port

The JTAG debug port can be physically disabled to prevent JTAG access in deployed systems. The recommended scheme is to hold test clock (TCK) to ground and hold test mode select (TMS) high. Disabling of the JTAG port also provides coverage for inadvertent activation of many debug and trace activities; since these are often initiated using an external debug tool that writes commands to the device using the JTAG port.

6.4.1.36 Lockout of JTAG Access Using OTP

JTAGLOCK functionality is implemented as part of the DCSM. JTAG access to the device can be restricted by programming the DCSM OTP. For more information refer to the *DCSM* section of the technical reference manual for a device, or *Enhancing Device Security by Using JTAGLOCK Feature*.

6.4.1.37 Decryption of Encrypted Data Output Using Same KEY and IV

A software test can be utilized to test the basic functionality that encrypts and decrypts with the same configuration and data (decryption of encrypted data output using same KEY and Initialization Vector (IV)). Such a test can be executed at boot or periodically. The necessary software requirements are defined by the software implemented by the system integrator.

6.4.1.38 Information Redundancy Techniques Including End-to-End Safing

Information redundancy techniques can be applied using software as an additional runtime diagnostic. There are many techniques that can be applied, such as a readback of written values and multiple reads of the same target data with a comparison of results.

To provide diagnostic coverage for network elements outside the C2000 MCU (wiring harness, connectors, and transceiver) end-to-end safety mechanisms are applied. These mechanisms can also provide diagnostic coverage inside the C2000 MCU. There are many different schemes applied, such as additional message checksums, redundant transmissions, time diversity in transmissions, and so forth. Most commonly, checksums are added to the payload section of a transmission to verify the correctness of a transmission. The checksums, sequence counter, and timeout expectations (or timestamp) are then included with the original data, in addition to any protocol-level parity and checksums. As these are generated and evaluated by the software at either end of the communication, the whole communication path is safed, resulting in end-to-end safing.

Any end-to-end communication diagnostics implemented must consider the failure modes and potential mitigating safety measures described in IEC 61784-3:2016 and summarized in IEC 61784-3:2016, Table 1.

6.4.1.39 Transmission Redundancy

The information is transferred several times in sequence using the same module instance and then compared to one another. When the same data path is used for duplicate transmissions, transmission redundancy is only useful for detecting transient faults. The diagnostic coverage can be improved by sending inverted data during the redundant transmission.

To provide diagnostic coverage of device interconnects and EMIF, readback of written data (in case of data writes) and multiple readbacks of information (in case of data reads) can be employed.

6.4.1.40 Disabling of Unused DMA Trigger Sources

The unintended trigger of DMA transfers can corrupt critical data and potentially be a source of interference to safety-critical applications. To avoid initiation of the unintended DMA transfers, TI recommends that unused DMA channels and DMA trigger sources are disabled at source or by configuring DMACHSRCSELx registers.

6.4.1.41 Software Test of Function Including Error Tests

A software test can be utilized to test the basic functionality of the module and to inject diagnostic errors and check for proper error response. Such a test can be executed at boot or periodically. The necessary software requirements are defined by the software implemented by the system integrator.

Ideas for creating module-specific functionality tests and error tests are described below:

- DMA functionality can be checked by transferring a known good data from a source memory to the destination memory and checking for data integrity after the transfer. The transfer can be initiated using the software trigger available (CONTROL.PERINTFRC). On-chip timer can be used to profile the time required for such a data transfer.
- A software test of input and output X-BAR module can be performed by creating a loop (output X-BAR can be used as stimulus to input X-BAR) using the input and output X-BAR, sending a known test sequence at the input and observing the sequence at the final output. Integrity of ePWM X-BAR can be checked by sending the test stimulus and observing the response using ePWM trip or sync functionality.
- A software test of XINT functionality can be checked by configuring the input X-BAR and forcing the corresponding GPIO register to generate an interrupt. The diagnostic coverage can be enhanced by performing checks for the polarity (XINTxCR.POLARITY) and enabling (XINTxCR.ENABLE) functionality as well.
- ECAP and EQEP functionality can be checked by looping back the PWM or GPIO outputs to the respective module inputs, providing a known good sequence (as required by the module), and observing the module output. In the case of ECAP, the test can be done internally with the help of input X-BAR.
- ROM prefetch functionality can be checked using similar techniques as given in Section 6.4.3.33.
- The PWM module consists of time-base (TB), counter compare (CC), action qualifier (AQ), dead-band generator (DB), PWM chopper (PC), trip zone (TZ), event trigger (ET), and digital compare (DC) submodules. The individual submodules can be tested by providing appropriate stimulus using PWM and observing

the response using one of the capture (timestamping) modules (eCAP, XINT, eQEP, and so forth). TI recommends covering the various register values associated with application configuration while performing the software test. Due to the regular, linear nature of the various submodules, getting high coverage using a software test is possible.

- A software test of SRAM wrapper logic must provide diagnostic coverage for arbitration between various initiators having access to the particular SRAM and correct functioning of access protection. This is in addition to the test used to provide coverage of SRAM bit cells (refer to Section 6.4.3.3).
- The interconnect (INC) functionality can be tested by writing complementary data-patterns, like 0xA5A5,0x5A5A and so forth, from processing units like CPU and CLA and reading back the data-patterns from registers of the IPs that are connected through different bridges. The readback data can be compared with the expected golden values to verify fault-free interconnect operation. This exercise can be repeated for different data width types of accesses (16/32 bits) and wide address ranges (as applicable) using both CPU and CLA. The CPU accesses can be repeated for different instances of peripherals used in application connected to various bridges, as shown in Figure 4-1.
- DAC has a set of control registers that can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A and so forth, in 16-bit access mode. All the registers can be read back and compared to expected values. Registers can be checked for the reset feature by configuring the registers to 0xA5A5 pattern, asserting soft reset of DAC, reading back the registers, and comparing the readback value with the expected reset value. Lock register can be checked to verify the DAC is set-once. Also, the registers, which are getting locked, must not update when written. To test core functionality of the DAC module, the DAC can be configured using software to provide a set of predetermined voltage levels. These voltage levels can be measured by external or internal ADC and results thus obtained can be cross checked against the expected value to verify proper operation. Extreme corner values of DAC, as per the application, can be programmed and tested to check the successful conversion of a digital to analog module across a valid range.
- Comparator subsystem (CMPSS) has a set of registers which can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A, and so forth, in both 16 and 32-bit access modes. The registers can be read back and compared against expected values. These accesses can be covered by applicable initiators like DMA, CLA, and CPU. Features of the CMPSS module, such as ramp decrement, can be checked for counting down of RAMPDLYA after the DAC module is loaded from RAMPDLYS by a rising PWMSYNC signal. The decrementer must reduce to zero and stay there until next reload from RAMPDLYS. Extreme values of RAMPDLYS can be configured before count down. Digital filter CTRIPHFILCTL/CTRIPLFILCTL registers can be checked by configuring them to a variety of N and T values and then verifying the COMPHSTS/COMPLSTS changes with the change in filter output. The applicable range of filter clock prescaler values (CTRIPLFILCLKCTL) can be exercised to verify that filter samples correctly.
- The general operation of the CPU timers can be tested by a software test by loading 32-bit counter register TIMH from period register PRDH. The PRDH register starts decrementing of the counter on every clock cycle. When counter reaches zero, a timer interrupt output generates an interrupt pulse. While testing the timer functionality, vary the timer prescale counter (TPR) value and input clocks by selecting clock source as SYSCLK, INTOSC1, INTOSC2, XTAL, or AUXPLLCLK. Test interrupts generation capability at the end of the timer counting. Check for the time overflow flag and timer reload (TRB) functions in the TCR register for correct functioning.
- A software test function in DCSM can be implemented independently in zone1, zone2, and unsecured zone to check DCSM functionality. Device security configurations are loaded from OTP to DCSM during the device boot phase. The test function can implement access filtering checks (read-write and execute permissions) to RAMs and flash sectors belonging to the same zone and different zone. An additional check for EXEONLY configuration can also be implemented for the RAMs and flash sectors to verify that all access (other than execute access) is blocked.

6.4.1.42 Software Test of Standalone GHASH Operation

The standalone GHASH must be calculated for the known data and stored. Periodically calculate standalone GHASH for known data and compare with the initially calculated GHASH. When there is a mismatch found, then a fault has occurred in GHASH block. Error response and any necessary software requirements are defined by the system integrator.

6.4.2 Processing Elements

6.4.2.1 Reciprocal Comparison by Software

Each CPU subsystem has a pair of diverse processing units (C28 and CLA) with different architectures and instruction sets. This enables one processing unit to handle the time-critical portion code (control CPU) and the other processing unit (supervisor CPU) to execute the noncritical portion of the code. Perform diagnostic functions and supervise execution of the control CPU as indicated in *Reciprocal Comparison Implementation*.

In case of identification of a fault during diagnostic functions of the supervisor CPU, the supervisor CPU causes the MCU to move to a safe state. The concept, *reciprocal comparison by software in separate processing units*, acts as a 1oo1D structure, providing high diagnostic coverage for the processing units as per ISO26262-5, Table D.4. The comparison must be performed several times during an FTTI. Reciprocal comparison is a software diagnostic feature and hence care must be taken to avoid common-mode failures. The final attained coverage depends on the quality of comparison (determined by extent and frequency of cross checking). The proposed cross-checking mechanism allows for hardware and software diversity since different processors, with different instruction sets and compilers, are used for enabling this mechanism. The diversity is further increased by having separate algorithms executed in both cores. In case failure is identified during reciprocal comparison, NMI can be triggered by software and then assert ERRORSTS.

6.4.2.2 Software Test of CPU

Testing the integrity of various CPU logic (C28x, FPU, TMU, and so forth) is possible using a software-based self-test library (STL). TI provides a C28x_STL start-up test library with 60% diagnostic coverage for C28x, FPU, TMU, and VCRC. For more details, refer to Section 4.4 and the documentation found within TMS320F28P55x C28x_STL software installation. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.2.3 Periodic Software Readback of Static Configuration Registers

Configuration registers are typically configured in the beginning and hold the value until the particular task execution. Periodic readback of configuration registers can provide a diagnostic for inadvertent writes or disruption of these registers.

The diagnostic coverage can be improved by extending the test to include readback of the flag registers that are expected to remain constant (for example, PLL lock status, EQEP phase error flag, and so forth) during the device operation as well. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

The diagnostic coverage of some peripherals can be further enhanced by applying some module-specific tests as follows:

- For improving the enhanced peripheral interrupt expander (EPIE) coverage, the PIE flag registers can be periodically checked to verify that all pending interrupts are serviced by reading the PIE flag registers (PIE_CTRL_REGS.PIEIFRx.all) and the peripheral interrupt flag registers.
- While serving the interrupt, the ISR routine can check for flag settings in peripheral and PIE to verify that correct interrupt is being serviced.

Since CLA configuration registers are accessible to C28x CPU only, this safety mechanism for the CLA module has to be executed by C28x CPU.

6.4.2.4 Access Protection Mechanism for Memories

All volatile memory blocks (including external memories)—except for M0/M1 on both subsystems—have different levels of protection. This capability allows the user to enable or disable specific access (for example, fetch and write) to individual RAM blocks from individual initiators (CPU, CLA, and DMA). There is no protection for read accesses, therefore, reads are always allowed from all the initiators that have access to that RAM block. To identify conditions when an initiator access to an SRAM is blocked, refer to the device-specific technical reference manual. This configuration can be changed during run time and allows memory to block access from specific initiators, or specific application threads, within the same initiator. This capability helps support freedom from interference requirements required by some applications.

6.4.2.5 Hardware Disable of JTAG Port

The JTAG debug port can be physically disabled to prevent JTAG access in deployed systems. The recommended scheme is to hold test clock (TCK) to ground and hold test mode select (TMS) high. Disabling of the JTAG port also provides coverage for inadvertent activation of many debug and trace activities; since these are often initiated using an external debug tool that writes commands to the device using the JTAG port.

6.4.2.6 CPU Handling of Illegal Operation, Illegal Results, and Instruction Trapping

The C28x CPU includes diagnostics for illegal operations, illegal results (underflow and overflow conditions), and instructions trapping (illegal opcode) that can serve as safety mechanisms. Any access to an invalid memory range returns 0x00000000 data. Access to an erased flash (default state for a new device) returns 0xFFFFFFFF. Both 0x00000000 and 0xFFFFFFFF are decoded as invalid instructions so that an erased flash or cleared memory, or an invalid address, forces the CPU to ITRAP. TI highly recommends the installation of software handlers to support the hardware illegal operation and instruction trapping.

Examples of CPU illegal operation, illegal results, and instruction traps include:

- Illegal instruction
- *TMS320C28x FPU Primer* (SPRAAN9)

6.4.2.7 Internal Watchdog (WD)

The internal watchdog has two modes of operation: normal watchdog (WD) and windowed watchdog (WWD). The system integrator can select to use one mode or the other but not both at the same time. For details of programming the internal watchdogs, refer to the device-specific technical reference manual. The WD is a traditional single threshold watchdog. The user programs a timeout value to the watchdog and must provide a predetermined WDKEY to the watchdog before the timeout counter expires. Expiration of the timeout counter or an incorrect WDKEY triggers an error response. The WD can issue either a warm system reset or a CPU maskable interrupt upon detection of a failure. The WD is enabled after reset.

In case of WWD, the user programs an upper bound and lower bound to create a time window during which the software must provide a predetermined WDKEY to the watchdog. Failure to receive the correct response within the time window or an incorrect WDKEY triggers an error response. The WWD can issue either a warm system reset or a CPU maskable interrupt upon detection of a failure. Normal WD operation is enabled by default after reset. Additional configuration must be performed to enable the WWD operation. For details of programming the internal watchdogs, refer to the device-specific technical reference manual. The use of the time window allows detection of additional clocking failure modes as compared to the WD implementation.

6.4.2.8 External Watchdog

External watchdog helps to reduce common mode failures, as external watchdog utilizes clock, reset, and power that are separate from the system being monitored. Error response, diagnostic testability, and any necessary software requirements are defined by the external watchdog selected by the system integrator.

Texas Instruments highly recommends the use of an external watchdog in addition to the internally provided watchdogs. An internal or external watchdog can provide indication of an inadvertent activation of logic that results in an impact to safety-critical execution. Any watchdog added externally must include a combination of temporal and logical monitoring of program sequence [IEC61508-7, clause A.9.3] or other appropriate methods so that high diagnostic effectiveness can be claimed.

6.4.2.9 Information Redundancy Techniques

Information redundancy techniques can be applied using software as an additional runtime diagnostic. To provide diagnostic coverage for network elements outside the C2000 MCU (wiring harness, connectors, and transceiver) end-to-end safety mechanisms are applied. These mechanisms can also provide diagnostic coverage inside the C2000 MCU.

In the case of processing elements (CPU and CLA), this refers to multiple executions of the code and software-based cross checking to verify correctness. The multiple execution and result comparison can be based on

implementations of either the same code executed multiple times or diversified software code. For details regarding the implementation, refer to the ISO26262-5, D.2.3.4.

In the case of the DMA, information redundancy techniques refer to additional information (besides the data payload) that verifies data integrity. For example, SECDED codes, parity codes, CRCs, and so forth enable information redundancy.

Typical control applications involve measuring three phase voltage and current. These values are either sampled directly using the on-chip ADC or sent to the TMS320F28P55x MCU by the sensors, which are captured using ECAP and so forth. In such scenarios, the correlation between input signals can be used to check the integrity (for example, if the three-phase voltage, V1, V2, and V3 is being measured, the function $V_1 + V_2 + V_3 = 0$ can be used to provide diagnostic coverage for input signal integrity).

In the case of SRAM and FLASH memory, the critical data, program, variables, and so forth can be stored redundantly and compared to their active counterparts before being used. Care must be taken to avoid compiler optimizing code containing redundant data or programs.

6.4.2.10 Stack Overflow Detection

A stack overflow in a safety application generally produces a catastrophic software crash resulting from data corruption, lost return addresses, or both. Hence, detecting an impending stack overflow is important. When properly configured, the capability exists on the TMS320F28P55x MCU device family for runtime detection of a stack overflow before the overflow occurs. For more information, refer to *Online Stack Overflow Detection on the TMS320C28x DSP*. Detection of an impending stack overflow triggers a maskable interrupt. The programmed error response and any necessary software requirements are defined by the system integrator.

6.4.2.11 VCRC Auto Coverage

The VCRC diagnostic is based on a 32-bit polynomial. For a given test, only one code is valid out of 2^{32} possibilities. Therefore, if there is a fault in the VCRC logic or associated data path, the correct passing code being generated through the fault is extremely unlikely.

6.4.2.12 Embedded Real Time Analysis and Diagnostic (ERAD)

The ERAD module provides system analysis capabilities that can be used to detect faults in the CPU and other logic on the MCU by configuring the bus comparator units that monitor CPU buses and counter units that count events. This module, which is accessible by the application software, consists of the enhanced bus comparator units and benchmark system event counter units.

The enhanced bus comparator units are used to monitor various CPU buses and generate events that can then be further processed or used directly. The activity monitored and detected by these units can be used to generate breakpoints, watch-points, or an interrupt (RTOSINT).

The benchmark system event counter units are used to analyze and profile the system. The counter can count events when setup as event mode and duration between system events when setup as duration mode.

After application code sets up the ERAD module, the module can work independently and generate an RTOSINT interrupt in case an event match occurs. This module can be used as a continuous online monitor of system events on the MCU.

6.4.2.13 Inbuilt Hardware Redundancy in ERAD Bus Comparator Module

The ERAD bus comparator units can be used to monitor CPU buses, and ERAD supports such 8 comparator blocks. The activity monitored and detected by these units can be used to generate breakpoints, watch-points, or an interrupt (RTOSINT). Bus comparator module events from different units can be combined using OR and AND logic to generate new events as required. The faults in the comparison block can be detected by configuring two comparator blocks to monitor the same set of CPU buses continuously and combine them using OR. The RTOS interrupt cause can indicate if the interrupt is set in one of the blocks or both.

6.4.2.14 Software Test of CLA

Testing the integrity of various CLA blocks (register bank, control unit, data path, and so forth) is possible using a software-based self-test library (STL). Based on the safety requirement, this test can be performed at start-up or during application time. For details on implementing the particular test, refer to the safety package delivered with the specific C2000 MCU device. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.2.15 CLA Handling of Illegal Operation and Illegal Results

The CLA co-processor has built in mechanisms to detect execution of an illegal instruction (illegal opcode) and floating point underflow or overflow conditions. CLA interrupts the CPU under such conditions. Any access to an invalid memory range returns 0x00000000 data. Access to an erased flash (default state for a new device) returns 0xFFFFFFFF. Both 0x00000000 and 0xFFFFFFFF are decoded as invalid instructions so that an erased flash, cleared memory, or an invalid address generates an interrupt to the CPU. CPU can decode the cause of interrupt by checking the required CLA flags. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.2.16 Software Readback of Written Configuration

To verify proper configuration of memory-mapped registers in this module, TI recommends that software implements a test to confirm proper configuration of all control registers by reading back the contents. This test also provides diagnostic coverage for the peripheral bus interface and peripheral interconnect bridges.

Since CLA configuration registers are accessible to the C28x CPU only, this safety mechanism for the CLA module must be executed by C28x CPU.

6.4.2.17 CLA Liveness Check Using CPU

CLA does not have an independent watchdog. Hence, TI recommends performing a liveness check periodically by the CPU. Typically, a sequential set of events is used to trigger the watchdog (for example, completion of CPU Task1, CLA1 Task1, CPU1 Task2, and CLA1 Task2). The output of the CLA liveness check can be used as one of the tasks to decide the watchdog triggering as indicated in Figure 6-2. The liveness check can be based on application-specific parameters, as illustrated in the VDA Egas concept [6] to improve the diagnostic coverage.

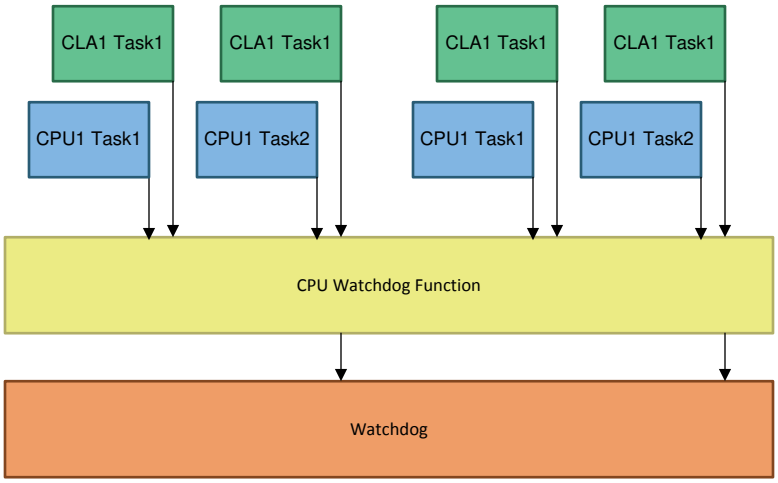

Figure 6-2. CLA Liveness Check

6.4.2.18 Software Test of Function Including Error Tests

A software test can be utilized to test the basic functionality of the module and to inject diagnostic errors and check for proper error response. Such a test can be executed at boot or periodically. The necessary software requirements are defined by the software implemented by the system integrator.

Ideas for creating module-specific functionality tests and error tests are described below:

- DMA functionality can be checked by transferring a known good data from a source memory to the destination memory and checking for data integrity after the transfer. The transfer can be initiated using the software trigger available (CONTROL.PERINTFRC). On-chip timer can be used to profile the time required for such a data transfer.
- A software test of input and output X-BAR module can be performed by creating a loop (output X-BAR can be used as stimulus to input X-BAR) using the input and output X-BAR, sending a known test sequence at the input and observing the sequence at the final output. Integrity of ePWM X-BAR can be checked by sending the test stimulus and observing the response using ePWM trip or sync functionality.
- A software test of XINT functionality can be checked by configuring the input X-BAR and forcing the corresponding GPIO register to generate an interrupt. The diagnostic coverage can be enhanced by performing checks for the polarity (XINTxCR.POLARITY) and enabling (XINTxCR.ENABLE) functionality as well.
- ECAP and EQEP functionality can be checked by looping back the PWM or GPIO outputs to the respective module inputs, providing a known good sequence (as required by the module), and observing the module output. In the case of ECAP, the test can be done internally with the help of input X-BAR.
- ROM prefetch functionality can be checked using similar techniques as given in Section 6.4.3.33.
- The PWM module consists of time-base (TB), counter compare (CC), action qualifier (AQ), dead-band generator (DB), PWM chopper (PC), trip zone (TZ), event trigger (ET), and digital compare (DC) submodules. The individual submodules can be tested by providing appropriate stimulus using PWM and observing the response using one of the capture (timestamping) modules (eCAP, XINT, eQEP, and so forth). TI recommends covering the various register values associated with application configuration while performing the software test. Due to the regular, linear nature of the various submodules, getting high coverage using a software test is possible.
- A software test of SRAM wrapper logic must provide diagnostic coverage for arbitration between various initiators having access to the particular SRAM and correct functioning of access protection. This is in addition to the test used to provide coverage of SRAM bit cells (refer to Section 6.4.3.3).
- The interconnect (INC) functionality can be tested by writing complementary data-patterns, like 0xA5A5,0x5A5A and so forth, from processing units like CPU and CLA and reading back the data-patterns from registers of the IPs that are connected through different bridges. The readback data can be compared with the expected golden values to verify fault-free interconnect operation. This exercise can be repeated for different data width types of accesses (16/32 bits) and wide address ranges (as applicable) using both CPU and CLA. The CPU accesses can be repeated for different instances of peripherals used in application connected to various bridges, as shown in Figure 4-1.
- DAC has a set of control registers that can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A and so forth, in 16-bit access mode. All the registers can be read back and compared to expected values. Registers can be checked for the reset feature by configuring the registers to 0xA5A5 pattern, asserting soft reset of DAC, reading back the registers, and comparing the readback value with the expected reset value. Lock register can be checked to verify the DAC is set-once. Also, the registers, which are getting locked, must not update when written. To test core functionality of the DAC module, the DAC can be configured using software to provide a set of predetermined voltage levels. These voltage levels can be measured by external or internal ADC and results thus obtained can be cross checked against the expected value to verify proper operation. Extreme corner values of DAC, as per the application, can be programmed and tested to check the successful conversion of a digital to analog module across a valid range.
- Comparator subsystem (CMPSS) has a set of registers which can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A, and so forth, in both 16 and 32-bit access modes. The registers can be read back and compared against expected values. These accesses can be covered by applicable initiators like DMA, CLA, and CPU. Features of the CMPSS module, such as ramp decrement, can be checked for counting down of RAMPDLYA after the DAC module is loaded from RAMPDLYS by a rising PWMSYNC signal. The decrementer must reduce to zero and stay there until next reload from RAMPDLYS. Extreme values of RAMPDLYS can be configured before count down. Digital filter CTRIPHFILCTL/CTRIPLFILCTL registers can be checked by configuring them to a variety of N and T values and then verifying the COMPHSTS/COMPLSTS changes with the change in filter output. The applicable range of filter clock prescaler values (CTRIPLFILCLKCTL) can be exercised to verify that filter samples correctly.

- The general operation of the CPU timers can be tested by a software test by loading 32-bit counter register TIMH from period register PRDH. The PRDH register starts decrementing of the counter on every clock cycle. When counter reaches zero, a timer interrupt output generates an interrupt pulse. While testing the timer functionality, vary the timer prescale counter (TPR) value and input clocks by selecting clock source as SYSCLK, INTOSC1, INTOSC2, XTAL, or AUXPLLCLK. Test interrupts generation capability at the end of the timer counting. Check for the time overflow flag and timer reload (TRB) functions in the TCR register for correct functioning.
- A software test function in DCSM can be implemented independently in zone1, zone2, and unsecured zone to check DCSM functionality. Device security configurations are loaded from OTP to DCSM during the device boot phase. The test function can implement access filtering checks (read-write and execute permissions) to RAMs and flash sectors belonging to the same zone and different zone. An additional check for EXEONLY configuration can also be implemented for the RAMs and flash sectors to verify that all access (other than execute access) is blocked.

6.4.3 Memory (Flash, SRAM and ROM)

6.4.3.1 SRAM ECC

Selected on-chip SRAMs support SECDED ECC diagnostic with separate ECC bits for data and address. For the specific address ranges that support ECC, refer to the TMS320F28P55x MCU device-specific data sheet. In SECDED scheme, a 21-bit code word is used to store the ECC data calculated independently for each 16 bit of data and for address. The ECC logic for the SRAM access is located in the SRAM wrapper. The ECC is evaluated directly at the memory output and data is sent to CPU after the data-integrity check. The data and address interconnects from SRAM to the CPU is not protected using ECC. Detected, correctable errors are corrected and monitoring the number of corrected errors is possible. The SRAM wrapper can be configured to trigger an interrupt once the number of corrected errors crosses a threshold. Uncorrectable, SRAM errors trigger an NMI and the ERRORSTS pin is asserted. The ECC logic for the SRAM is enabled at reset. For more information regarding memories supporting ECC, refer to the device-specific data sheet.

6.4.3.2 SRAM Parity

Selected on-chip SRAMs support a parity diagnostic with separate parity bits for data and address. For the specific address ranges that support parity, refer to the device-specific data sheet. In the parity scheme, a 3-bit code word is used to store the parity data; calculated independently for each 16 bit of data and for address. The parity generation and check logic for the SRAM is located in the SRAM wrapper. The parity is checked directly at the memory output and data is sent to the CPU after the data integrity check. The data and address interconnect, from SRAM to the CPU, is not protected using parity. SRAM parity errors trigger an NMI and the ERRORSTS is asserted. The parity logic for the SRAM is enabled at reset. For more information regarding memories supporting parity, refer to the device-specific data sheet.

6.4.3.3 Software Test of SRAM

Testing the integrity of SRAM (bit cells, address decoder, and sense amplifier logic) is possible using the CPU. Based on the safety requirement, this test can be performed at start-up or during application time. If the SRAM contents are static, a CRC check using VCU can be performed in place of destructive test (a test where memory contents must be restored after the test). For details on implementing this particular test, check the safety package delivered with this specific C2000 MCU device.

6.4.3.4 Bit Multiplexing in SRAM Memory Array

The SRAM modules implemented in the TMS320F28P55x MCU device family have a bit multiplexing scheme where the bits accessed to generate a logical (CPU) word are not physically adjacent. This scheme helps to reduce the probability of physical, multibit faults resulting in logical, multibit faults, rather the bits manifest as multiple, single-bit faults. The SECDED SRAM ECC diagnostic can correct a single-bit fault and detect a double-bit fault in a logical word. Similarly, the SRAM parity diagnostic can detect single-bit faults. This scheme improves the usefulness of the SRAM ECC and parity diagnostic. Bit multiplexing is a feature of the SRAM and cannot be modified by the software.

6.4.3.5 Periodic Software Readback of Static Configuration Registers

Configuration registers are typically configured in the beginning and hold the value until the particular task execution. Periodic readback of configuration registers can provide a diagnostic for inadvertent writes or disruption of these registers.

The diagnostic coverage can be improved by extending the test to include readback of the flag registers that are expected to remain constant (for example, PLL lock status, EQEP phase error flag, and so forth) during the device operation as well. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

The diagnostic coverage of some peripherals can be further enhanced by applying some module-specific tests as follows:

- For improving the enhanced peripheral interrupt expander (EPIE) coverage, the PIE flag registers can be periodically checked to verify that all pending interrupts are serviced by reading the PIE flag registers (PIE_CTRL_REGS.PIEIFRx.all) and the peripheral interrupt flag registers.
- While serving the interrupt, the ISR routine can check for flag settings in peripheral and PIE to verify that correct interrupt is being serviced.

Since CLA configuration registers are accessible to C28x CPU only, this safety mechanism for the CLA module has to be executed by C28x CPU.

6.4.3.6 Software Readback of Written Configuration

To verify proper configuration of memory-mapped registers in this module, TI recommends that software implements a test to confirm proper configuration of all control registers by reading back the contents. This test also provides diagnostic coverage for the peripheral bus interface and peripheral interconnect bridges.

Since CLA configuration registers are accessible to the C28x CPU only, this safety mechanism for the CLA module must be executed by C28x CPU.

6.4.3.7 Data Scrubbing to Detect/Correct Memory Errors

Bus initiators (CPU, CLA or DMA) can be configured to provide fake reads to the memory (provided a particular bus initiator has access to the memory) and the read data can be checked by the built-in ECC or parity logic. In the case of SRAMs with ECC protection, single-bit errors are corrected and written back. For both SRAMs and flash, an interrupt is issued once the count exceeds the preset threshold in the case of correctable errors, and NMI is issued in the case of uncorrectable errors.

Since the contents of Flash memory are static, VCU CRC Check of Static Memory Contents provides better diagnostic coverage compared to this diagnostic.

6.4.3.8 VCRC Check of Static Memory Contents

The TMS320F28P55x MCU device family includes a co-processor implementing cyclic redundancy check (CRC) using a standard polynomial. The CRC module can be used to test the integrity of SRAM, Flash, OTP, and external memory contents by calculating a CRC for all memory contents and comparing this value to a previously generated *golden* CRC. The comparison of results, indication of fault, and fault response are the responsibility of the software managing the test. The cyclical check applied by the CRC logic provides an inherent level of self-checking (auto-coverage), which can be considered for application in latent fault diagnostics.

6.4.3.9 Software Test of Function Including Error Tests

A software test can be utilized to test the basic functionality of the module and to inject diagnostic errors and check for proper error response. Such a test can be executed at boot or periodically. The necessary software requirements are defined by the software implemented by the system integrator.

Ideas for creating module-specific functionality tests and error tests are described below:

- DMA functionality can be checked by transferring a known good data from a source memory to the destination memory and checking for data integrity after the transfer. The transfer can be initiated using the

software trigger available (CONTROL.PERINTFRC). On-chip timer can be used to profile the time required for such a data transfer.

- A software test of input and output X-BAR module can be performed by creating a loop (output X-BAR can be used as stimulus to input X-BAR) using the input and output X-BAR, sending a known test sequence at the input and observing the sequence at the final output. Integrity of ePWM X-BAR can be checked by sending the test stimulus and observing the response using ePWM trip or sync functionality.
- A software test of XINT functionality can be checked by configuring the input X-BAR and forcing the corresponding GPIO register to generate an interrupt. The diagnostic coverage can be enhanced by performing checks for the polarity (XINTxCR.POLARITY) and enabling (XINTxCR.ENABLE) functionality as well.
- ECAP and EQEP functionality can be checked by looping back the PWM or GPIO outputs to the respective module inputs, providing a known good sequence (as required by the module), and observing the module output. In the case of ECAP, the test can be done internally with the help of input X-BAR.
- ROM prefetch functionality can be checked using similar techniques as given in Section 6.4.3.33.
- The PWM module consists of time-base (TB), counter compare (CC), action qualifier (AQ), dead-band generator (DB), PWM chopper (PC), trip zone (TZ), event trigger (ET), and digital compare (DC) submodules. The individual submodules can be tested by providing appropriate stimulus using PWM and observing the response using one of the capture (timestamping) modules (eCAP, XINT, eQEP, and so forth). TI recommends covering the various register values associated with application configuration while performing the software test. Due to the regular, linear nature of the various submodules, getting high coverage using a software test is possible.
- A software test of SRAM wrapper logic must provide diagnostic coverage for arbitration between various initiators having access to the particular SRAM and correct functioning of access protection. This is in addition to the test used to provide coverage of SRAM bit cells (refer to Section 6.4.3.3).
- The interconnect (INC) functionality can be tested by writing complementary data-patterns, like 0xA5A5,0x5A5A and so forth, from processing units like CPU and CLA and reading back the data-patterns from registers of the IPs that are connected through different bridges. The readback data can be compared with the expected golden values to verify fault-free interconnect operation. This exercise can be repeated for different data width types of accesses (16/32 bits) and wide address ranges (as applicable) using both CPU and CLA. The CPU accesses can be repeated for different instances of peripherals used in application connected to various bridges, as shown in Figure 4-1.
- DAC has a set of control registers that can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A and so forth, in 16-bit access mode. All the registers can be read back and compared to expected values. Registers can be checked for the reset feature by configuring the registers to 0xA5A5 pattern, asserting soft reset of DAC, reading back the registers, and comparing the readback value with the expected reset value. Lock register can be checked to verify the DAC is set-once. Also, the registers, which are getting locked, must not update when written. To test core functionality of the DAC module, the DAC can be configured using software to provide a set of predetermined voltage levels. These voltage levels can be measured by external or internal ADC and results thus obtained can be cross checked against the expected value to verify proper operation. Extreme corner values of DAC, as per the application, can be programmed and tested to check the successful conversion of a digital to analog module across a valid range.
- Comparator subsystem (CMPSS) has a set of registers which can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A, and so forth, in both 16 and 32-bit access modes. The registers can be read back and compared against expected values. These accesses can be covered by applicable initiators like DMA, CLA, and CPU. Features of the CMPSS module, such as ramp decrement, can be checked for counting down of RAMPDLYA after the DAC module is loaded from RAMPDLYS by a rising PWMSYNC signal. The decrementer must reduce to zero and stay there until next reload from RAMPDLYS. Extreme values of RAMPDLYS can be configured before count down. Digital filter CTRIPHFILCTL/CTRIPLFILCTL registers can be checked by configuring them to a variety of N and T values and then verifying the COMPHSTS/COMPLSTS changes with the change in filter output. The applicable range of filter clock prescaler values (CTRIPLFILCLKCTL) can be exercised to verify that filter samples correctly.
- The general operation of the CPU timers can be tested by a software test by loading 32-bit counter register TIMH from period register PRDH. The PRDH register starts decrementing of the counter on every clock cycle. When counter reaches zero, a timer interrupt output generates an interrupt pulse. While testing the timer functionality, vary the timer prescale counter (TPR) value and input clocks by selecting clock source

as SYSCLK, INTOSC1, INTOSC2, XTAL, or AUXPLLCLK. Test interrupts generation capability at the end of the timer counting. Check for the time overflow flag and timer reload (TRB) functions in the TCR register for correct functioning.

- A software test function in DCSM can be implemented independently in zone1, zone2, and unsecured zone to check DCSM functionality. Device security configurations are loaded from OTP to DCSM during the device boot phase. The test function can implement access filtering checks (read-write and execute permissions) to RAMs and flash sectors belonging to the same zone and different zone. An additional check for EXEONLY configuration can also be implemented for the RAMs and flash sectors to verify that all access (other than execute access) is blocked.

6.4.3.10 Access Protection Mechanism for Memories

All volatile memory blocks (including external memories)—except for M0/M1 on both subsystems—have different levels of protection. This capability allows the user to enable or disable specific access (for example, fetch and write) to individual RAM blocks from individual initiators (CPU, CLA, and DMA). There is no protection for read accesses, therefore, reads are always allowed from all the initiators that have access to that RAM block. To identify conditions when an initiator access to an SRAM is blocked, refer to the device-specific technical reference manual. This configuration can be changed during run time and allows memory to block access from specific initiators, or specific application threads, within the same initiator. This capability helps support freedom from interference requirements required by some applications.

6.4.3.11 Lock Mechanism for Control Registers

The module contains a lock mechanism for protection of critical control registers. Once the associated LOCK register bits are set, the write access to the registers are blocked. Locked registers cannot be updated by software; once locked, only reset can unlock the registers.

6.4.3.12 Software Test of ECC Logic

Testing the functionality of the SRAM ECC is possible by injecting single-bit and double-bit errors in test mode, performing reads on locations with ECC errors, and then checking for the error response.

Flash ECC logic can be checked with the help of ECC test field ECC_TEST_EN in the FECC_CTRL register. This technique causes an output comparison failure between the redundant ECC logic upon a flash read access. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

For additional details on implementing this diagnostic for SRAM and FLASH memory, refer to the *Application Test Hooks for Error Detection and Correction* and *Mechanism to Check the Correctness of ECC Logic* sections in the device-specific technical reference manual.

6.4.3.13 Software Test of Parity Logic

Testing the functionality of parity-error detection logic is possible by forcing a parity error into the data or parity-memory bits and observing whether the parity error detection logic reports an error. Parity can also be calculated manually and compared to the hardware calculated value stored in the parity-memory bits.

6.4.3.14 Information Redundancy Techniques

Information redundancy techniques can be applied using software as an additional runtime diagnostic. To provide diagnostic coverage for network elements outside the C2000 MCU (wiring harness, connectors, and transceiver) end-to-end safety mechanisms are applied. These mechanisms can also provide diagnostic coverage inside the C2000 MCU.

In the case of processing elements (CPU and CLA), this refers to multiple executions of the code and software-based cross checking to verify correctness. The multiple execution and result comparison can be based on implementations of either the same code executed multiple times or diversified software code. For details regarding the implementation, refer to the ISO26262-5, D.2.3.4.

In the case of the DMA, information redundancy techniques refer to additional information (besides the data payload) that verifies data integrity. For example, SECDED codes, parity codes, CRCs, and so forth enable information redundancy.

Typical control applications involve measuring three phase voltage and current. These values are either sampled directly using the on-chip ADC or sent to the TMS320F28P55x MCU by the sensors, which are captured using ECAP and so forth. In such scenarios, the correlation between input signals can be used to check the integrity (for example, if the three-phase voltage, V1, V2, and V3 is being measured, the function $V_1 + V_2 + V_3 = 0$ can be used to provide diagnostic coverage for input signal integrity).

In the case of SRAM and FLASH memory, the critical data, program, variables, and so forth can be stored redundantly and compared to their active counterparts before being used. Care must be taken to avoid compiler optimizing code containing redundant data or programs.

6.4.3.15 CPU Handling of Illegal Operation, Illegal Results, and Instruction Trapping

The C28x CPU includes diagnostics for illegal operations, illegal results (underflow and overflow conditions), and instructions trapping (illegal opcode) that can serve as safety mechanisms. Any access to an invalid memory range returns 0x00000000 data. Access to an erased flash (default state for a new device) returns 0xFFFFFFFF. Both 0x00000000 and 0xFFFFFFFF are decoded as invalid instructions so that an erased flash or cleared memory, or an invalid address, forces the CPU to ITRAP. TI highly recommends the installation of software handlers to support the hardware illegal operation and instruction trapping.

Examples of CPU illegal operation, illegal results, and instruction traps include:

- Illegal instruction
- *TMS320C28x FPU Primer* (SPRAAN9)

6.4.3.16 Internal Watchdog (WD)

The internal watchdog has two modes of operation: normal watchdog (WD) and windowed watchdog (WWD). The system integrator can select to use one mode or the other but not both at the same time. For details of programming the internal watchdogs, refer to the device-specific technical reference manual. The WD is a traditional single threshold watchdog. The user programs a timeout value to the watchdog and must provide a predetermined WDKEY to the watchdog before the timeout counter expires. Expiration of the timeout counter or an incorrect WDKEY triggers an error response. The WD can issue either a warm system reset or a CPU maskable interrupt upon detection of a failure. The WD is enabled after reset.

In case of WWD, the user programs an upper bound and lower bound to create a time window during which the software must provide a predetermined WDKEY to the watchdog. Failure to receive the correct response within the time window or an incorrect WDKEY triggers an error response. The WWD can issue either a warm system reset or a CPU maskable interrupt upon detection of a failure. Normal WD operation is enabled by default after reset. Additional configuration must be performed to enable the WWD operation. For details of programming the internal watchdogs, refer to the device-specific technical reference manual. The use of the time window allows detection of additional clocking failure modes as compared to the WD implementation.

6.4.3.17 External Watchdog

External watchdog helps to reduce common mode failures, as external watchdog utilizes clock, reset, and power that are separate from the system being monitored. Error response, diagnostic testability, and any necessary software requirements are defined by the external watchdog selected by the system integrator.

Texas Instruments highly recommends the use of an external watchdog in addition to the internally provided watchdogs. An internal or external watchdog can provide indication of an inadvertent activation of logic that results in an impact to safety-critical execution. Any watchdog added externally must include a combination of temporal and logical monitoring of program sequence [IEC61508-7, clause A.9.3] or other appropriate methods so that high diagnostic effectiveness can be claimed.

6.4.3.18 CLA Handling of Illegal Operation and Illegal Results

The CLA co-processor has built in mechanisms to detect execution of an illegal instruction (illegal opcode) and floating point underflow or overflow conditions. CLA interrupts the CPU under such conditions. Any access to an invalid memory range returns 0x00000000 data. Access to an erased flash (default state for a new device) returns 0xFFFFFFFF. Both 0x00000000 and 0xFFFFFFFF are decoded as invalid instructions so that an erased

flash, cleared memory, or an invalid address generates an interrupt to the CPU. CPU can decode the cause of interrupt by checking the required CLA flags. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.3.19 Memory Power-On Self-Test (MPOST)

A start-up test of the memories provides detection for permanent faults inside on-chip memories. Some of the C2000 device family products supports the programmable built-in self-test (PBIST), an easy and efficient way of testing the memories by configuring the customer OTP field. PBIST architecture consists of a small co-processor with a dedicated instruction set targeted specifically toward testing memories. This co-processor, when triggered, executes test routines stored in the PBIST ROM and runs them on multiple on-chip memory instances. The on-chip memory configuration information is also stored in the PBIST ROM. PBIST provides very high diagnostic coverage for permanent faults on the implemented SRAMs and ROMs. If PBIST is configured, a test (March13n for SRAMs or triple_read_xor_read for ROMs) is executed on all the memory instances. The PBIST test status is stored in the on-chip memory. The term *memory* covered by PBIST indicates to SRAM and ROM. Flash testing is not covered as part of this specification.

Since the code for testing of the memories resides in boot ROM, testing the boot ROM using PBIST is not possible. Hence, a separate boot ROM checksum test is done prior to PBIST. Prior to performing any test using PBIST, an always-fail test case is executed. This always-fail test case validates the proper functioning of the PBIST controller and the ability of the PBIST controller to indicate a failure. For more details, refer to *C2000 Memory Power-On Self-Test (M-POST)*.

6.4.3.20 Power-Up Pre-Operational Security Checks

During the device boot, the device goes through various phases, as indicated in Figure 4-9. In the pre-operational phase (before starting the application), the application code is expected to perform a set of checks to verify the correct initialization of device security, which includes checks to confirm the correct:

- Link-pointer settings
- CRC lock setting
- Partitioning of secure RAM blocks and Flash sectors (Grab Bits)
- Settings for execute-only protection for secure RAM blocks and Flash sectors
- Partitioning of the CLA and Flash Bank2
- Settings for boot configuration

Before starting the execution of downloaded code, users must check the integrity of the code using a CRC function. Once pre-operational checks are successfully complete with expected results, the device can enter the application phase.

6.4.3.21 ROM Parity

ROMs support a parity diagnostic with separate parity bits for data and address. For the specific address ranges that support parity, refer to the device-specific data sheet. In the parity scheme, a 3-bit code word is used to store the parity data calculated independently for each 16 bit of data and for address. The parity is checked directly at the memory output and data is sent to the CPU after the data-integrity check. ROM parity errors trigger an NMI and the ERRORSTS is asserted. The parity logic for the ROM is enabled at reset.

6.4.3.22 Flash ECC

The on-chip flash memory is supported by single-error correction, double-error detection (SECDED), and an error-correcting code (ECC) diagnostic. In this SECDED scheme, an 8-bit code word is used to store the ECC of 64 bit data and corresponding address. The ECC decoding logic at the flash bank output checks the correctness of memory content. ECC evaluation is done on every data and program read. The data and program interconnects, that connect the CPU and flash memory, are not protected by ECC. Detected, correctable errors can be corrected or not corrected, depending on whether correction functionality is enabled. Single-bit address ECC errors are flagged as uncorrectable errors. Errors that cannot be corrected generate an NMI and the ERRORSTS pin is asserted. A count of the corrected errors (single-bit data errors) is monitored in the memory

error registers and an interrupt is generated once the count exceeds the programmed threshold. The corrupted memory address of the last error location is also logged in the memory error registers.

6.4.3.23 Flash Program Verify and Erase Verify Check

Whenever any program and erase operation is done, the flash controller performs a program and erase verify check. If the program and erase operation fails, FSM status register (STATCMD) indicates the error by setting the corresponding flags into the status register.

6.4.3.24 Flash Program and Erase Protection

There are two types of write and erase protections available. One is static protection (FLPROT) configured at the device level and the other is a CMDWEPROT configuration, which is delivered to the Flash Wrapper through input signals. The FLPROT write and erase protect bit can be configured at boot and locked; providing semi-permanent protection for certain sectors in the Flash. Inadvertent program and erase operations targeting these sectors are blocked. Furthermore, the CMDWEPROT* registers exist in the Flash Wrapper, allowing further protection for critical data. The CMDWEPROT* registers default to protect all sectors at reset and at the completion of every Flash Wrapper command. The CMDWEPROT* must be configured before each erase and program command through the Flash API using the Fapi_setupBankSectorEnable() function.

6.4.3.25 Flash Wrapper Error and Status Reporting

During and after execution of all commands (except ClearStatus), the Flash Wrapper updates the STATCMD register. CMDINPROGRESS and CMDDONE indicators are present to indicate the execution status of a command. A CMDPASS bit indicates whether an operation passed or failed. There are five different fail bits which indicate different failure mechanisms when a command fails. Setting of the CMDDONE status indicator at the end of a command execution also triggers the assertion of an interrupt event to the system.

6.4.3.26 Prevent 0 to 1 Transition Using Program Command

The Flash Wrapper checks the user-provided data during program command execution. Specifically, this data validity check looks for a bit in the targeted flash words that is already programmed to 0 and is configured by the current operation to be programmed to a 1. Programming an existing 0 to a 1 in the Flash is not possible. If such a condition is detected, then the program operation terminates with a FAILINVDATA error in STATCMD without issuing the programming command.

6.4.3.27 On-demand Software Program Verify and Blank Check

Flash API provides CPU-based read verify functions to verify the programmed location (of any length in multiples of 32 bits) and do a blank check of the erased sector or bank. For more information, refer to the Fapi_doVerify() and Fapi_doBlankCheck() functions in the Flash API reference guide.

6.4.3.28 Software Readback of Written Configuration

To verify proper configuration of memory-mapped registers in this module, TI recommends that software implements a test to confirm proper configuration of all control registers by reading back the contents. This test also provides diagnostic coverage for the peripheral bus interface and peripheral interconnect bridges.

Since CLA configuration registers are accessible to the C28x CPU only, this safety mechanism for the CLA module must be executed by C28x CPU.

6.4.3.29 CMDWEPROT* and Program Command Data Buffer Registers Self-Clear After Command Execution

At the end of all command executions in the Flash Wrapper, several registers are forced back to the default states. This default state usually allows protection from program or erase without an update to the MMR. For example, CMDWEPROT* defaults to protect all sectors, which means neither a program nor an erase can be done without clearing the bit (or bits) of the targeted sector. Also, the data buffer registers of the program command inside the Flash wrapper default to all 1s, which means the registers must be updated to do a program operation.

6.4.3.30 ECC Generation and Checker Logic is Separate in Hardware

Hardware for generating ECC data (ECC codec) is included in the flash wrapper design. However, error detection and correction is not done using this hardware. The system provides hardware to decode the ECC data read with data from the flash bank access port and does a detection and correction using that logic. Thus, the hardware used for ECC generation and for ECC detection and correction are separate. This allows an easier diagnostic if there is a fault in either of these sets of ECC logic.

6.4.3.31 Bit Multiplexing in Flash Memory Array

The flash modules implemented in this device family have a bit multiplexing scheme where the bits accessed to generate a logical (CPU) word are not physically adjacent. This scheme helps to reduce the probability of physical multibit faults resulting in logical multibit faults. Rather, the bits manifest as multiple single-bit faults. As the SECDED flash ECC can correct a single-bit fault and detect a double-bit fault in a logical word, this scheme improves the usefulness of the flash ECC diagnostic. Bit multiplexing is a feature of the flash memory and cannot be modified by the software.

6.4.3.32 Auto ECC Generation Override

There is a bit in the Flash Wrapper operation configuration register which allows the ECC generation logic to be bypassed and data written explicitly to special data registers to be used as ECC data. This configuration bit takes effect for program operations. This bit allows ECC data to be controlled for diagnostic purposes.

Refer to the Flash API reference guide for more information on using the Fapi_DataAndEcc programming mode, which allows providing non-auto-generated ECC.

6.4.3.33 Software Test of Flash Prefetch, Data Cache, and Wait States

Once enabled, prefetch logic keeps fetching the next 128-bit row (4 x 32-bit words) from flash bank. On detecting the discontinuity, the prefetch buffer is cleared. A software test can be performed to ascertain the proper behavior of this logic. The following operation sequence can be performed:

1. Disable the prefetch mechanism, enable the timer and watchdog. Execute a particular function which has linear code and code with multiple discontinuities. Store the time taken for executing this function, as time_1 (timer value).
2. Enable the prefetch mechanism and execute the same function again. Store the time taken for executing this function, as time_2 (timer value). This value must be less than the time_1 (time_1 > time_2). We can mark this timer value as a *golden* value and expect the same timer values for each run of the same function.
3. Since each flash bank row has 4 x 32-bit words, number of rows fetched from the flash bank varies, as per the code alignment within the flash bank. Hence, users must verify that the prefetch logic test function is aligned and located in a particular location within flash to maintain the same timing behavior and does not vary compile to compile.

Similar timer-based profiling can be performed to ascertain proper functioning of the data cache and wait states.

6.4.3.34 Software Test of ECC Logic

Testing the functionality of the SRAM ECC is possible by injecting single-bit and double-bit errors in test mode, performing reads on locations with ECC errors, and then checking for the error response.

Flash ECC logic can be checked with the help of ECC test field ECC_TEST_EN in the FECC_CTRL register. This technique causes an output comparison failure between the redundant ECC logic upon a flash read access. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

For additional details on implementing this diagnostic for SRAM and FLASH memory, refer to the *Application Test Hooks for Error Detection and Correction* and *Mechanism to Check the Correctness of ECC Logic* sections in the device-specific technical reference manual.

6.4.4 On-Chip Communication Including Bus-Arbitration

6.4.4.1 Software Test of Function Including Error Tests

A software test can be utilized to test the basic functionality of the module and to inject diagnostic errors and check for proper error response. Such a test can be executed at boot or periodically. The necessary software requirements are defined by the software implemented by the system integrator.

Ideas for creating module-specific functionality tests and error tests are described below:

- DMA functionality can be checked by transferring a known good data from a source memory to the destination memory and checking for data integrity after the transfer. The transfer can be initiated using the software trigger available (CONTROL.PERINTFRC). On-chip timer can be used to profile the time required for such a data transfer.
- A software test of input and output X-BAR module can be performed by creating a loop (output X-BAR can be used as stimulus to input X-BAR) using the input and output X-BAR, sending a known test sequence at the input and observing the sequence at the final output. Integrity of ePWM X-BAR can be checked by sending the test stimulus and observing the response using ePWM trip or sync functionality.
- A software test of XINT functionality can be checked by configuring the input X-BAR and forcing the corresponding GPIO register to generate an interrupt. The diagnostic coverage can be enhanced by performing checks for the polarity (XINTxCR.POLARITY) and enabling (XINTxCR.ENABLE) functionality as well.
- ECAP and EQEP functionality can be checked by looping back the PWM or GPIO outputs to the respective module inputs, providing a known good sequence (as required by the module), and observing the module output. In the case of ECAP, the test can be done internally with the help of input X-BAR.
- ROM prefetch functionality can be checked using similar techniques as given in Section 6.4.3.33.
- The PWM module consists of time-base (TB), counter compare (CC), action qualifier (AQ), dead-band generator (DB), PWM chopper (PC), trip zone (TZ), event trigger (ET), and digital compare (DC) submodules. The individual submodules can be tested by providing appropriate stimulus using PWM and observing the response using one of the capture (timestamping) modules (eCAP, XINT, eQEP, and so forth). TI recommends covering the various register values associated with application configuration while performing the software test. Due to the regular, linear nature of the various submodules, getting high coverage using a software test is possible.
- A software test of SRAM wrapper logic must provide diagnostic coverage for arbitration between various initiators having access to the particular SRAM and correct functioning of access protection. This is in addition to the test used to provide coverage of SRAM bit cells (refer to Section 6.4.3.3).
- The interconnect (INC) functionality can be tested by writing complementary data-patterns, like 0xA5A5,0x5A5A and so forth, from processing units like CPU and CLA and reading back the data-patterns from registers of the IPs that are connected through different bridges. The readback data can be compared with the expected golden values to verify fault-free interconnect operation. This exercise can be repeated for different data width types of accesses (16/32 bits) and wide address ranges (as applicable) using both CPU and CLA. The CPU accesses can be repeated for different instances of peripherals used in application connected to various bridges, as shown in Figure 4-1.
- DAC has a set of control registers that can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A and so forth, in 16-bit access mode. All the registers can be read back and compared to expected values. Registers can be checked for the reset feature by configuring the registers to 0xA5A5 pattern, asserting soft reset of DAC, reading back the registers, and comparing the readback value with the expected reset value. Lock register can be checked to verify the DAC is set-once. Also, the registers, which are getting locked, must not update when written. To test core functionality of the DAC module, the DAC can be configured using software to provide a set of predetermined voltage levels. These voltage levels can be measured by external or internal ADC and results thus obtained can be cross checked against the expected value to verify proper operation. Extreme corner values of DAC, as per the application, can be programmed and tested to check the successful conversion of a digital to analog module across a valid range.
- Comparator subsystem (CMPSS) has a set of registers which can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A, and so forth, in both 16 and 32-bit access modes. The registers can be read back and compared against expected values. These accesses can be covered by applicable initiators like DMA, CLA, and CPU. Features of the CMPSS module, such as ramp decrement, can be checked for

counting down of RAMPDLYA after the DAC module is loaded from RAMPDLYS by a rising PWMSYNC signal. The decrementer must reduce to zero and stay there until next reload from RAMPDLYS. Extreme values of RAMPDLYS can be configured before count down. Digital filter CTRIPHFILCTL/CTRIPLFILCTL registers can be checked by configuring them to a variety of N and T values and then verifying the COMPHSTS/COMPLSTS changes with the change in filter output. The applicable range of filter clock prescaler values (CTRIPLFILCLKCTL) can be exercised to verify that filter samples correctly.

- The general operation of the CPU timers can be tested by a software test by loading 32-bit counter register TIMH from period register PRDH. The PRDH register starts decrementing of the counter on every clock cycle. When counter reaches zero, a timer interrupt output generates an interrupt pulse. While testing the timer functionality, vary the timer prescale counter (TPR) value and input clocks by selecting clock source as SYSCLK, INTOSC1, INTOSC2, XTAL, or AUXPLLCLK. Test interrupts generation capability at the end of the timer counting. Check for the time overflow flag and timer reload (TRB) functions in the TCR register for correct functioning.
- A software test function in DCSM can be implemented independently in zone1, zone2, and unsecured zone to check DCSM functionality. Device security configurations are loaded from OTP to DCSM during the device boot phase. The test function can implement access filtering checks (read-write and execute permissions) to RAMs and flash sectors belonging to the same zone and different zone. An additional check for EXEONLY configuration can also be implemented for the RAMs and flash sectors to verify that all access (other than execute access) is blocked.

6.4.4.2 Internal Watchdog (WD)

The internal watchdog has two modes of operation: normal watchdog (WD) and windowed watchdog (WWD). The system integrator can select to use one mode or the other but not both at the same time. For details of programming the internal watchdogs, refer to the device-specific technical reference manual. The WD is a traditional single threshold watchdog. The user programs a timeout value to the watchdog and must provide a predetermined WDKEY to the watchdog before the timeout counter expires. Expiration of the timeout counter or an incorrect WDKEY triggers an error response. The WD can issue either a warm system reset or a CPU maskable interrupt upon detection of a failure. The WD is enabled after reset.

In case of WWD, the user programs an upper bound and lower bound to create a time window during which the software must provide a predetermined WDKEY to the watchdog. Failure to receive the correct response within the time window or an incorrect WDKEY triggers an error response. The WWD can issue either a warm system reset or a CPU maskable interrupt upon detection of a failure. Normal WD operation is enabled by default after reset. Additional configuration must be performed to enable the WWD operation. For details of programming the internal watchdogs, refer to the device-specific technical reference manual. The use of the time window allows detection of additional clocking failure modes as compared to the WD implementation.

6.4.4.3 External Watchdog

External watchdog helps to reduce common mode failures, as external watchdog utilizes clock, reset, and power that are separate from the system being monitored. Error response, diagnostic testability, and any necessary software requirements are defined by the external watchdog selected by the system integrator.

Texas Instruments highly recommends the use of an external watchdog in addition to the internally provided watchdogs. An internal or external watchdog can provide indication of an inadvertent activation of logic that results in an impact to safety-critical execution. Any watchdog added externally must include a combination of temporal and logical monitoring of program sequence [IEC61508-7, clause A.9.3] or other appropriate methods so that high diagnostic effectiveness can be claimed.

6.4.4.4 Periodic Software Readback of Static Configuration Registers

Configuration registers are typically configured in the beginning and hold the value until the particular task execution. Periodic readback of configuration registers can provide a diagnostic for inadvertent writes or disruption of these registers.

The diagnostic coverage can be improved by extending the test to include readback of the flag registers that are expected to remain constant (for example, PLL lock status, EQEP phase error flag, and so forth) during the

device operation as well. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

The diagnostic coverage of some peripherals can be further enhanced by applying some module-specific tests as follows:

- For improving the enhanced peripheral interrupt expander (EPIE) coverage, the PIE flag registers can be periodically checked to verify that all pending interrupts are serviced by reading the PIE flag registers (PIE_CTRL_REGS.PIEIFRx.all) and the peripheral interrupt flag registers.
- While serving the interrupt, the ISR routine can check for flag settings in peripheral and PIE to verify that correct interrupt is being serviced.

Since CLA configuration registers are accessible to C28x CPU only, this safety mechanism for the CLA module has to be executed by C28x CPU.

6.4.4.5 Software Readback of Written Configuration

To verify proper configuration of memory-mapped registers in this module, TI recommends that software implements a test to confirm proper configuration of all control registers by reading back the contents. This test also provides diagnostic coverage for the peripheral bus interface and peripheral interconnect bridges.

Since CLA configuration registers are accessible to the C28x CPU only, this safety mechanism for the CLA module must be executed by C28x CPU.

6.4.4.6 CPU Handling of Illegal Operation, Illegal Results, and Instruction Trapping

The C28x CPU includes diagnostics for illegal operations, illegal results (underflow and overflow conditions), and instructions trapping (illegal opcode) that can serve as safety mechanisms. Any access to an invalid memory range returns 0x00000000 data. Access to an erased flash (default state for a new device) returns 0xFFFFFFFF. Both 0x00000000 and 0xFFFFFFFF are decoded as invalid instructions so that an erased flash or cleared memory, or an invalid address, forces the CPU to ITRAP. TI highly recommends the installation of software handlers to support the hardware illegal operation and instruction trapping.

Examples of CPU illegal operation, illegal results, and instruction traps include:

- Illegal instruction
- *TMS320C28x FPU Primer* (SPRAAN9)

6.4.4.7 CLA Handling of Illegal Operation and Illegal Results

The CLA co-processor has built in mechanisms to detect execution of an illegal instruction (illegal opcode) and floating point underflow or overflow conditions. CLA interrupts the CPU under such conditions. Any access to an invalid memory range returns 0x00000000 data. Access to an erased flash (default state for a new device) returns 0xFFFFFFFF. Both 0x00000000 and 0xFFFFFFFF are decoded as invalid instructions so that an erased flash, cleared memory, or an invalid address generates an interrupt to the CPU. CPU can decode the cause of interrupt by checking the required CLA flags. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.4.8 Transmission Redundancy

The information is transferred several times in sequence using the same module instance and then compared to one another. When the same data path is used for duplicate transmissions, transmission redundancy is only useful for detecting transient faults. The diagnostic coverage can be improved by sending inverted data during the redundant transmission.

To provide diagnostic coverage of device interconnects and EMIF, readback of written data (in case of data writes) and multiple readbacks of information (in case of data reads) can be employed.

6.4.4.9 Hardware Redundancy

Hardware redundancy techniques can be applied using hardware, or as a combination of hardware and software, to provide runtime diagnostic. In this implementation, redundant hardware resources are utilized to

provide diagnostic coverage for elements within and outside (wiring harness, connectors, and transceiver) the TMS320F28P55x MCU.

In case of peripherals like GPIO, XBAR, PWM, OTTO, DAC, CMPSS, and XINT, hardware redundancy can be implemented by having multichannel parallel outputs (where independent outputs are used for transmitting information and failure detection is carried out using internal or external comparators) or input comparison and voting (comparison of independent inputs to comply with a defined tolerance range, time and value). In such scenarios, the system can be designed so the failure of one input or output does not cause the system to go into a dangerous state. While servicing the error conditions (redundancy conditions), as in two redundant sources tripping the PWM, always read back the status flags and verify that both sources are active while tripping and thus providing latent fault coverage for the trip logic.

In case of peripherals like ADC, ECAP, EQEP, and so forth, hardware redundancy can be implemented by having multiple instances of the peripheral sample the same input and simultaneously perform the same operation followed by a cross check of the output values.

In case of communication peripherals like MCAN, SPI, SCI, and so forth, hardware redundancy during signal reception can be implemented by having multiple instances of the peripheral receive the same data followed by a comparison to verify data integrity. Hardware redundancy during transmission can be employed by having a complete redundant signal path (wiring harness, connectors, and transceiver) from the transmitter to the receiver or by sampling the transmitted data by a redundant peripheral instance followed by a data integrity check.

While implementing hardware redundancy for ADC and DAC modules, additional care must be taken to verify common-cause failures do not impact both instances in the same way. Reference voltage sources, configured for redundant module instances, must be independent. Additionally, ADC SOC trigger sources, used for redundant ADC instances, must be configured to different PWM module instances. In case of a DAC module, the comparator can be implemented using an external device.

While implementing hardware redundancy for the PWM module, TI recommends that the PWM module instance used is part of separate sync chains. This requirement is to avoid a common-cause failure on a sync signal that affects both PWM modules in the same way.

While implementing hardware redundancy for the GPIO module, TI recommends using GPIO pins from different GPIO groups to avoid common-cause failures.

6.4.4.10 EALLOW and MEALLOW Protection for Critical Registers

EALLOW (CPU, DMA) and MEALLOW (CLA) protection enables write access to emulation and other protected registers. CPU (CLA) can set this bit using EALLOW (MEALLOW) instruction and cleared using EDIS (MEDIS) instruction. The protection can be used to prevent data being written to the wrong place, which happens with conditions like boundary exceeding, incorrect pointers, stack overflow or corruption, and so forth. Reads from the protected registers are always allowed. TI recommends issuing an EDIS (or MEDIS) for protection once write for the protected registers are complete.

6.4.4.11 Information Redundancy Techniques

Information redundancy techniques can be applied using software as an additional runtime diagnostic. To provide diagnostic coverage for network elements outside the C2000 MCU (wiring harness, connectors, and transceiver) end-to-end safety mechanisms are applied. These mechanisms can also provide diagnostic coverage inside the C2000 MCU.

In the case of processing elements (CPU and CLA), this refers to multiple executions of the code and software-based cross checking to verify correctness. The multiple execution and result comparison can be based on implementations of either the same code executed multiple times or diversified software code. For details regarding the implementation, refer to the ISO26262-5, D.2.3.4.

In the case of the DMA, information redundancy techniques refer to additional information (besides the data payload) that verifies data integrity. For example, SECDED codes, parity codes, CRCs, and so forth enable information redundancy.

Typical control applications involve measuring three phase voltage and current. These values are either sampled directly using the on-chip ADC or sent to the TMS320F28P55x MCU by the sensors, which are captured using ECAP and so forth. In such scenarios, the correlation between input signals can be used to check the integrity (for example, if the three-phase voltage, V1, V2, and V3 is being measured, the function $V_1 + V_2 + V_3 = 0$ can be used to provide diagnostic coverage for input signal integrity).

In the case of SRAM and FLASH memory, the critical data, program, variables, and so forth can be stored redundantly and compared to their active counterparts before being used. Care must be taken to avoid compiler optimizing code containing redundant data or programs.

6.4.4.12 DMA Overflow Interrupt

DMA supports latching one additional trigger event. Before DMA services this latched event, if an additional event occurs, a DMA overflow interrupt is generated so the CONTROL_REG.PERINTFLG is set and another interrupt event occurs. When the CONTROL_REG.PERINTFLG sets, this indicates a previous peripheral event is latched and has not been serviced by the DMA.

6.4.4.13 Access Protection Mechanism for Memories

All volatile memory blocks (including external memories)—except for M0/M1 on both subsystems—have different levels of protection. This capability allows the user to enable or disable specific access (for example, fetch and write) to individual RAM blocks from individual initiators (CPU, CLA, and DMA). There is no protection for read accesses, therefore, reads are always allowed from all the initiators that have access to that RAM block. To identify conditions when an initiator access to an SRAM is blocked, refer to the device-specific technical reference manual. This configuration can be changed during run time and allows memory to block access from specific initiators, or specific application threads, within the same initiator. This capability helps support freedom from interference requirements required by some applications.

6.4.4.14 Disabling of Unused DMA Trigger Sources

The unintended trigger of DMA transfers can corrupt critical data and potentially be a source of interference to safety-critical applications. To avoid initiation of the unintended DMA transfers, TI recommends that unused DMA channels and DMA trigger sources are disabled at source or by configuring DMACHSRCSELx registers.

6.4.4.15 Software Test of SRAM

Testing the integrity of SRAM (bit cells, address decoder, and sense amplifier logic) is possible using the CPU. Based on the safety requirement, this test can be performed at start-up or during application time. If the SRAM contents are static, a CRC check using VCU can be performed in place of destructive test (a test where memory contents must be restored after the test). For details on implementing this particular test, check the safety package delivered with this specific C2000 MCU device.

6.4.4.16 Software Test of ePIE Operation Including Error Tests

A software test for testing the basic functionality and failure modes (such as, continuous interrupts, no interrupts, and crossover interrupts) can be implemented. Such testing can be based on generating the interrupts from the peripherals (using either a software-force capability, for example, ECAP_REGS.ECFRC.CTROVF, or creating the interrupt scenario functionally, for example, creating a counter overflow condition in ECAP) and verifying that the interrupt is serviced and serviced in the correct order. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.4.17 Maintaining Interrupt Handler for Unused Interrupts

The C2000 MCU devices contain a large number of interrupts. A typical application only uses a very small subset of all the available interrupts. Multiple configurations are possible for the unused interrupts. These configurations include disabling the unused interrupts, enabling the unused interrupts, returning to the application in the interrupt service routine (ISR), and so forth. Receiving an unused interrupt in the application can be an early indication of faulty scenarios within the C2000 MCU. Hence, TI highly recommends enabling all interrupts and configuring the ISR to a common routine for logging or error handling.

6.4.4.18 Online Monitoring of Interrupts and Events

For interrupts and events, failure can be detected using information about the time behavior of the system. The monitored signals can be either periodic or aperiodic.

For a typical closed-loop control application, most of the critical events are periodic and these periodic events can be monitored with incoherent events being used for fault detection. A few places where the online monitoring of periodic interrupts and events can be employed include:

- Periodic generation of ADC start of conversion (SoC) (x): An ADC SoC signal can be used to generate an external interrupt (XINT) with the help of X-BAR. The occurrence of periodic interrupts can be monitored.
- Periodic DMA trigger: Some of the DMA events can be periodic in nature (for example, copy of ADC results, updating of CMPA register, and so forth). DMA supports interrupt generation on the completion of the DMA action and this capability can be used for online monitoring.
- Periodic occurrence of ECAP and EQEP interrupts.

Monitoring of interrupts and events that are normally not expected during the correct operation can also be used to improve the diagnostic coverage (for example, ECC correctable error interrupt).

6.4.4.19 SRAM Parity

Selected on-chip SRAMs support a parity diagnostic with separate parity bits for data and address. For the specific address ranges that support parity, refer to the device-specific data sheet. In the parity scheme, a 3-bit code word is used to store the parity data; calculated independently for each 16 bit of data and for address. The parity generation and check logic for the SRAM is located in the SRAM wrapper. The parity is checked directly at the memory output and data is sent to the CPU after the data integrity check. The data and address interconnect, from SRAM to the CPU, is not protected using parity. SRAM parity errors trigger an NMI and the ERRORSTS is asserted. The parity logic for the SRAM is enabled at reset. For more information regarding memories supporting parity, refer to the device-specific data sheet.

6.4.4.20 Software Test of Parity Logic

Testing the functionality of parity-error detection logic is possible by forcing a parity error into the data or parity-memory bits and observing whether the parity error detection logic reports an error. Parity can also be calculated manually and compared to the hardware calculated value stored in the parity-memory bits.

6.4.4.21 Multibit Enable Keys for Control Registers

This module includes features to support avoidance of unintentional control register programming. Implementation of multibit keys for critical control registers is one such feature (for example, EPWM_REGS, EPWMLOCK, and so forth). The multibit keys are particularly effective for avoiding unintentional activation. For more details on the registers for which the diagnostic is applicable, refer to the device-specific technical reference manual. The operation of this safety mechanism is continuous and cannot be altered by the software. This mechanism can be tested by generating software transactions with and without correct keys and observing the updated register value.

6.4.4.22 Majority Voting and Error Detection of Link Pointer

The link pointer OTP location is not protected by ECC. Majority voting and data-consistency-based error detection is implemented to provide better security to the customer code and enable application safety. The location of the zone-select region in OTP is decided based on the value of three 29-bit link pointers (Zx-LINKPOINTERx) programmed in the OTP of each zone of each CPU subsystems. The final value of the link pointer is resolved in hardware when a dummy read is issued to all the link pointers by comparing all the three values (bit-wise voting logic). Any error in the resolution of the final link pointer value sets the Zx_LINKPOINTERERR register.

6.4.4.23 VCRC Check of Static Memory Contents

The TMS320F28P55x MCU device family includes a co-processor implementing cyclic redundancy check (CRC) using a standard polynomial. The CRC module can be used to test the integrity of SRAM, Flash, OTP, and external memory contents by calculating a CRC for all memory contents and comparing this value to

a previously generated *golden* CRC. The comparison of results, indication of fault, and fault response are the responsibility of the software managing the test. The cyclical check applied by the CRC logic provides an inherent level of self-checking (auto-coverage), which can be considered for application in latent fault diagnostics.

6.4.4.24 Software Check of X-BAR Flag

X-BAR flag registers are used to flag the inputs of the ePWM and output X-Bars to provide software knowledge of the input sources that were triggered. The flag registers can be periodically read to ascertain if any ePWM trip zones, ePWM syncing, or GPIO output signaling was missed.

6.4.4.25 1oo2 Software Voting Using Secondary Free Running Counter

The TIMER module contains three counters that can be used to provide an operating system time base. While one counter is used as the operating system time base, using one of the other counters as a diagnostic on the first periodic check is possible using software of the counter values in the two timers. The second counter can be fed with a different clock source and a different prescale configuration can be selected to avoid common-mode errors. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.4.26 Software Test of Function Including Error Tests

A software test can be utilized to test the basic functionality of the module and to inject diagnostic errors and check for proper error response. Such a test can be executed at boot or periodically. The necessary software requirements are defined by the software implemented by the system integrator.

Ideas for creating module-specific functionality tests and error tests are described below:

- DMA functionality can be checked by transferring a known good data from a source memory to the destination memory and checking for data integrity after the transfer. The transfer can be initiated using the software trigger available (CONTROL.PERINTFRC). On-chip timer can be used to profile the time required for such a data transfer.
- A software test of input and output X-BAR module can be performed by creating a loop (output X-BAR can be used as stimulus to input X-BAR) using the input and output X-BAR, sending a known test sequence at the input and observing the sequence at the final output. Integrity of ePWM X-BAR can be checked by sending the test stimulus and observing the response using ePWM trip or sync functionality.
- A software test of XINT functionality can be checked by configuring the input X-BAR and forcing the corresponding GPIO register to generate an interrupt. The diagnostic coverage can be enhanced by performing checks for the polarity (XINTxCR.POLARITY) and enabling (XINTxCR.ENABLE) functionality as well.
- ECAP and EQEP functionality can be checked by looping back the PWM or GPIO outputs to the respective module inputs, providing a known good sequence (as required by the module), and observing the module output. In the case of ECAP, the test can be done internally with the help of input X-BAR.
- ROM prefetch functionality can be checked using similar techniques as given in Section 6.4.3.33.
- The PWM module consists of time-base (TB), counter compare (CC), action qualifier (AQ), dead-band generator (DB), PWM chopper (PC), trip zone (TZ), event trigger (ET), and digital compare (DC) submodules. The individual submodules can be tested by providing appropriate stimulus using PWM and observing the response using one of the capture (timestamping) modules (eCAP, XINT, eQEP, and so forth). TI recommends covering the various register values associated with application configuration while performing the software test. Due to the regular, linear nature of the various submodules, getting high coverage using a software test is possible.
- A software test of SRAM wrapper logic must provide diagnostic coverage for arbitration between various initiators having access to the particular SRAM and correct functioning of access protection. This is in addition to the test used to provide coverage of SRAM bit cells (refer to Section 6.4.3.3).
- The interconnect (INC) functionality can be tested by writing complementary data-patterns, like 0xA5A5,0x5A5A and so forth, from processing units like CPU and CLA and reading back the data-patterns from registers of the IPs that are connected through different bridges. The readback data can be compared with the expected golden values to verify fault-free interconnect operation. This exercise can be repeated

for different data width types of accesses (16/32 bits) and wide address ranges (as applicable) using both CPU and CLA. The CPU accesses can be repeated for different instances of peripherals used in application connected to various bridges, as shown in Figure 4-1.

- DAC has a set of control registers that can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A and so forth, in 16-bit access mode. All the registers can be read back and compared to expected values. Registers can be checked for the reset feature by configuring the registers to 0xA5A5 pattern, asserting soft reset of DAC, reading back the registers, and comparing the readback value with the expected reset value. Lock register can be checked to verify the DAC is set-once. Also, the registers, which are getting locked, must not update when written. To test core functionality of the DAC module, the DAC can be configured using software to provide a set of predetermined voltage levels. These voltage levels can be measured by external or internal ADC and results thus obtained can be cross checked against the expected value to verify proper operation. Extreme corner values of DAC, as per the application, can be programmed and tested to check the successful conversion of a digital to analog module across a valid range.
- Comparator subsystem (CMPSS) has a set of registers which can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A, and so forth, in both 16 and 32-bit access modes. The registers can be read back and compared against expected values. These accesses can be covered by applicable initiators like DMA, CLA, and CPU. Features of the CMPSS module, such as ramp decrement, can be checked for counting down of RAMPDLYA after the DAC module is loaded from RAMPDLYS by a rising PWMSYNC signal. The decrementer must reduce to zero and stay there until next reload from RAMPDLYS. Extreme values of RAMPDLYS can be configured before count down. Digital filter CTRIPHFILCTL/CTRIPLFILCTL registers can be checked by configuring them to a variety of N and T values and then verifying the COMPHSTS/COMPLSTS changes with the change in filter output. The applicable range of filter clock prescaler values (CTRIPLFILCLKCTL) can be exercised to verify that filter samples correctly.
- The general operation of the CPU timers can be tested by a software test by loading 32-bit counter register TIMH from period register PRDH. The PRDH register starts decrementing of the counter on every clock cycle. When counter reaches zero, a timer interrupt output generates an interrupt pulse. While testing the timer functionality, vary the timer prescale counter (TPR) value and input clocks by selecting clock source as SYSCLK, INTOSC1, INTOSC2, XTAL, or AUXPLLCLK. Test interrupts generation capability at the end of the timer counting. Check for the time overflow flag and timer reload (TRB) functions in the TCR register for correct functioning.
- A software test function in DCSM can be implemented independently in zone1, zone2, and unsecured zone to check DCSM functionality. Device security configurations are loaded from OTP to DCSM during the device boot phase. The test function can implement access filtering checks (read-write and execute permissions) to RAMs and flash sectors belonging to the same zone and different zone. An additional check for EXEONLY configuration can also be implemented for the RAMs and flash sectors to verify that all access (other than execute access) is blocked.

6.4.4.27 Monitoring of CLB by eCAP or eQEP

The outputs of the configurable logic block (CLB) can be monitored for proper operation by eCAP or eQEP using internal connections. The user must configure the CLB to generate a known good sequence of pulses (triggers), as required by the modules (eCAP or eQEP), to observe outputs through these blocks. The efficiency of monitoring is based on customer configuration.

6.4.4.28 Lock Mechanism for Control Registers

The module contains a lock mechanism for protection of critical control registers. Once the associated LOCK register bits are set, the write access to the registers are blocked. Locked registers cannot be updated by software; once locked, only reset can unlock the registers.

6.4.4.29 Periodic Software Read Back of SPI Buffer

Using the CLB to SPI, without the CPU intervention, CLB is able to send out a continuous stream of 16-bit data to SPI which can then be saved in the device memory using the FIFO and DMA mechanism of the SPI. TI recommends implementing tests in the software to read back the SPI FIFO contents and the final memory contents and verify the contents are same when sending data from CLB to the SPI and then to the device memory. This provides a check for if the CLB data reaches SPI and the device memory.

6.4.5 Digital I/O

6.4.5.1 Software Test of Function Including Error Tests

A software test can be utilized to test the basic functionality of the module and to inject diagnostic errors and check for proper error response. Such a test can be executed at boot or periodically. The necessary software requirements are defined by the software implemented by the system integrator.

Ideas for creating module-specific functionality tests and error tests are described below:

- DMA functionality can be checked by transferring a known good data from a source memory to the destination memory and checking for data integrity after the transfer. The transfer can be initiated using the software trigger available (CONTROL.PERINTFRC). On-chip timer can be used to profile the time required for such a data transfer.
- A software test of input and output X-BAR module can be performed by creating a loop (output X-BAR can be used as stimulus to input X-BAR) using the input and output X-BAR, sending a known test sequence at the input and observing the sequence at the final output. Integrity of ePWM X-BAR can be checked by sending the test stimulus and observing the response using ePWM trip or sync functionality.
- A software test of XINT functionality can be checked by configuring the input X-BAR and forcing the corresponding GPIO register to generate an interrupt. The diagnostic coverage can be enhanced by performing checks for the polarity (XINTxCR.POLARITY) and enabling (XINTxCR.ENABLE) functionality as well.
- ECAP and EQEP functionality can be checked by looping back the PWM or GPIO outputs to the respective module inputs, providing a known good sequence (as required by the module), and observing the module output. In the case of ECAP, the test can be done internally with the help of input X-BAR.
- ROM prefetch functionality can be checked using similar techniques as given in Section 6.4.3.33.
- The PWM module consists of time-base (TB), counter compare (CC), action qualifier (AQ), dead-band generator (DB), PWM chopper (PC), trip zone (TZ), event trigger (ET), and digital compare (DC) submodules. The individual submodules can be tested by providing appropriate stimulus using PWM and observing the response using one of the capture (timestamping) modules (eCAP, XINT, eQEP, and so forth). TI recommends covering the various register values associated with application configuration while performing the software test. Due to the regular, linear nature of the various submodules, getting high coverage using a software test is possible.
- A software test of SRAM wrapper logic must provide diagnostic coverage for arbitration between various initiators having access to the particular SRAM and correct functioning of access protection. This is in addition to the test used to provide coverage of SRAM bit cells (refer to Section 6.4.3.3).
- The interconnect (INC) functionality can be tested by writing complementary data-patterns, like 0xA5A5,0x5A5A and so forth, from processing units like CPU and CLA and reading back the data-patterns from registers of the IPs that are connected through different bridges. The readback data can be compared with the expected golden values to verify fault-free interconnect operation. This exercise can be repeated for different data width types of accesses (16/32 bits) and wide address ranges (as applicable) using both CPU and CLA. The CPU accesses can be repeated for different instances of peripherals used in application connected to various bridges, as shown in Figure 4-1.
- DAC has a set of control registers that can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A and so forth, in 16-bit access mode. All the registers can be read back and compared to expected values. Registers can be checked for the reset feature by configuring the registers to 0xA5A5 pattern, asserting soft reset of DAC, reading back the registers, and comparing the readback value with the expected reset value. Lock register can be checked to verify the DAC is set-once. Also, the registers, which are getting locked, must not update when written. To test core functionality of the DAC module, the DAC can be configured using software to provide a set of predetermined voltage levels. These voltage levels can be measured by external or internal ADC and results thus obtained can be cross checked against the expected value to verify proper operation. Extreme corner values of DAC, as per the application, can be programmed and tested to check the successful conversion of a digital to analog module across a valid range.
- Comparator subsystem (CMPSS) has a set of registers which can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A, and so forth, in both 16 and 32-bit access modes. The registers can be read back and compared against expected values. These accesses can be covered by applicable initiators like DMA, CLA, and CPU. Features of the CMPSS module, such as ramp decrement, can be checked for

counting down of RAMPDLYA after the DAC module is loaded from RAMPDLYS by a rising PWMSYNC signal. The decrementer must reduce to zero and stay there until next reload from RAMPDLYS. Extreme values of RAMPDLYS can be configured before count down. Digital filter CTRIPHFILCTL/CTRIPLFILCTL registers can be checked by configuring them to a variety of N and T values and then verifying the COMPHSTS/COMPLSTS changes with the change in filter output. The applicable range of filter clock prescaler values (CTRIPLFILCLKCTL) can be exercised to verify that filter samples correctly.

- The general operation of the CPU timers can be tested by a software test by loading 32-bit counter register TIMH from period register PRDH. The PRDH register starts decrementing of the counter on every clock cycle. When counter reaches zero, a timer interrupt output generates an interrupt pulse. While testing the timer functionality, vary the timer prescale counter (TPR) value and input clocks by selecting clock source as SYSCLK, INTOSC1, INTOSC2, XTAL, or AUXPLLCLK. Test interrupts generation capability at the end of the timer counting. Check for the time overflow flag and timer reload (TRB) functions in the TCR register for correct functioning.
- A software test function in DCSM can be implemented independently in zone1, zone2, and unsecured zone to check DCSM functionality. Device security configurations are loaded from OTP to DCSM during the device boot phase. The test function can implement access filtering checks (read-write and execute permissions) to RAMs and flash sectors belonging to the same zone and different zone. An additional check for EXEONLY configuration can also be implemented for the RAMs and flash sectors to verify that all access (other than execute access) is blocked.

6.4.5.2 Hardware Redundancy

Hardware redundancy techniques can be applied using hardware, or as a combination of hardware and software, to provide runtime diagnostic. In this implementation, redundant hardware resources are utilized to provide diagnostic coverage for elements within and outside (wiring harness, connectors, and transceiver) the TMS320F28P55x MCU.

In case of peripherals like GPIO, XBAR, PWM, OTTO, DAC, CMPSS, and XINT, hardware redundancy can be implemented by having multichannel parallel outputs (where independent outputs are used for transmitting information and failure detection is carried out using internal or external comparators) or input comparison and voting (comparison of independent inputs to comply with a defined tolerance range, time and value). In such scenarios, the system can be designed so the failure of one input or output does not cause the system to go into a dangerous state. While servicing the error conditions (redundancy conditions), as in two redundant sources tripping the PWM, always read back the status flags and verify that both sources are active while tripping and thus providing latent fault coverage for the trip logic.

In case of peripherals like ADC, ECAP, EQEP, and so forth, hardware redundancy can be implemented by having multiple instances of the peripheral sample the same input and simultaneously perform the same operation followed by a cross check of the output values.

In case of communication peripherals like MCAN, SPI, SCI, and so forth, hardware redundancy during signal reception can be implemented by having multiple instances of the peripheral receive the same data followed by a comparison to verify data integrity. Hardware redundancy during transmission can be employed by having a complete redundant signal path (wiring harness, connectors, and transceiver) from the transmitter to the receiver or by sampling the transmitted data by a redundant peripheral instance followed by a data integrity check.

While implementing hardware redundancy for ADC and DAC modules, additional care must be taken to verify common-cause failures do not impact both instances in the same way. Reference voltage sources, configured for redundant module instances, must be independent. Additionally, ADC SOC trigger sources, used for redundant ADC instances, must be configured to different PWM module instances. In case of a DAC module, the comparator can be implemented using an external device.

While implementing hardware redundancy for the PWM module, TI recommends that the PWM module instance used is part of separate sync chains. This requirement is to avoid a common-cause failure on a sync signal that affects both PWM modules in the same way.

While implementing hardware redundancy for the GPIO module, TI recommends using GPIO pins from different GPIO groups to avoid common-cause failures.

6.4.5.3 Monitoring of ePWM by eCAP

The ePWM outputs can be monitored for proper operation by an input capture peripheral, such as the eCAP . The connection between ePWM output and eCAP input can be made either externally in the board or internally using XBAR. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator. Similarly, eCAP can be tested by periodically measuring the ePWM pulse width. XINTxCTR (counter of XINT module), capture mode of eQEP, and DCCAP (PWM event filter unit) can also be used to detect rising and falling edges of the PWM and extract the timestamping information. This information can be further used to build additional diagnostics.

6.4.5.4 Periodic Software Readback of Static Configuration Registers

Configuration registers are typically configured in the beginning and hold the value until the particular task execution. Periodic readback of configuration registers can provide a diagnostic for inadvertent writes or disruption of these registers.

The diagnostic coverage can be improved by extending the test to include readback of the flag registers that are expected to remain constant (for example, PLL lock status, EQEP phase error flag, and so forth) during the device operation as well. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

The diagnostic coverage of some peripherals can be further enhanced by applying some module-specific tests as follows:

- For improving the enhanced peripheral interrupt expander (EPIE) coverage, the PIE flag registers can be periodically checked to verify that all pending interrupts are serviced by reading the PIE flag registers (PIE_CTRL_REGS.PIEIFRx.all) and the peripheral interrupt flag registers.
- While serving the interrupt, the ISR routine can check for flag settings in peripheral and PIE to verify that correct interrupt is being serviced.

Since CLA configuration registers are accessible to C28x CPU only, this safety mechanism for the CLA module has to be executed by C28x CPU.

6.4.5.5 Software Readback of Written Configuration

To verify proper configuration of memory-mapped registers in this module, TI recommends that software implements a test to confirm proper configuration of all control registers by reading back the contents. This test also provides diagnostic coverage for the peripheral bus interface and peripheral interconnect bridges.

Since CLA configuration registers are accessible to the C28x CPU only, this safety mechanism for the CLA module must be executed by C28x CPU.

6.4.5.6 Lock Mechanism for Control Registers

The module contains a lock mechanism for protection of critical control registers. Once the associated LOCK register bits are set, the write access to the registers are blocked. Locked registers cannot be updated by software; once locked, only reset can unlock the registers.

6.4.5.7 ePWM Fault Detection Using XBAR

A combination of ePWM outputs feedback to the input X-BAR, GPIO inversion logic, and digital compare (DC) submodules of ePWM and can be used for implementing simple (for example, signal crossover), but effective, anomaly checks on the PWM outputs. The feature can be used to trip the PWM and enter safe state if any anomaly is detected.

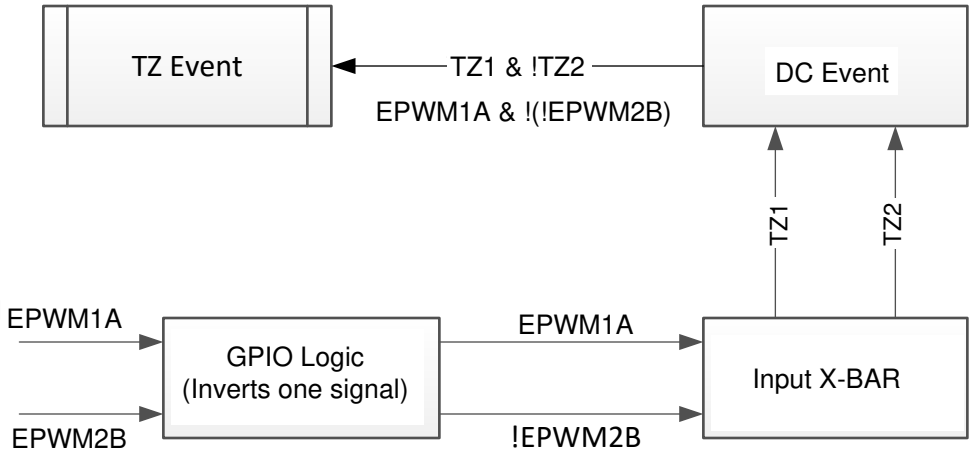

Figure 6-3. ePWM Fault Detection Using X-BAR

6.4.5.8 ePWM Synchronization Check

ePWM modules can be chained together using a clock synchronization scheme that allows the modules to operate as a single system when required. In the synchronous mode of operation, checking for proper synchronization of the various PWM instances is critical to avoid catastrophic conditions. The synchronization of the various PWMs can be checked by reading the TBSTS.SYNCI bit of the ePWM module. The correct phase relationship that is intended as a result of the sync operation can be crosschecked by comparing the TBCTR register value.

6.4.5.9 ePWM Application-Level Safety Mechanism

ePWM is typically used as the output signal in closed-loop control applications, such as EV traction, DC to DC, and industrial drive. In such applications, a failure in the ePWM output (such as, a stuck-at fault or a frequency or duty cycle change) results in a disturbance to control loop parameters or variables; leading to conditions such as overvoltage, overcurrent, or overtemperature. By monitoring the characteristics of these control-loop parameters implemented at application-level, faults in the ePWM module can be detected.

6.4.5.10 Online Monitoring of Interrupts and Events

For interrupts and events, failure can be detected using information about the time behavior of the system. The monitored signals can be either periodic or aperiodic.

For a typical closed-loop control application, most of the critical events are periodic and these periodic events can be monitored with incoherent events being used for fault detection. A few places where the online monitoring of periodic interrupts and events can be employed include:

- Periodic generation of ADC start of conversion (SoC) (x): An ADC SoC signal can be used to generate an external interrupt (XINT) with the help of X-BAR. The occurrence of periodic interrupts can be monitored.
- Periodic DMA trigger: Some of the DMA events can be periodic in nature (for example, copy of ADC results, updating of CMPA register, and so forth). DMA supports interrupt generation on the completion of the DMA action and this capability can be used for online monitoring.
- Periodic occurrence of ECAP and EQEP interrupts.

Monitoring of interrupts and events that are normally not expected during the correct operation can also be used to improve the diagnostic coverage (for example, ECC correctable error interrupt).

6.4.5.11 Monitoring of ePWM by ADC

The ePWM outputs can be monitored for correct operation by the ADC using board-level feedback, as indicated in Figure 6-4. The technical details for implementing such a loopback, like signal resolution and so forth, is provided in the reference link [9]. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

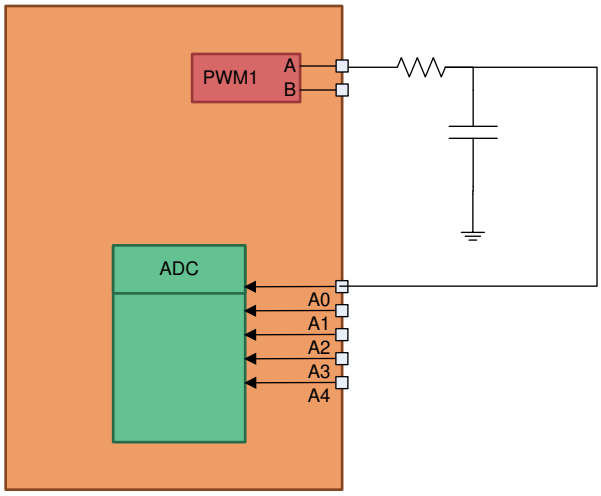

Figure 6-4. Monitoring of ePWM by ADC

6.4.5.12 HRPWM Built-In Self-Check and Diagnostic Capabilities

The microedge positioner (MEP) logic in HRPWM is capable of placing an edge in one of 255 discrete time steps. The size of these steps is 150ps. For typical MEP step size, refer to the device-specific data sheet. The MEP step size varies based on the worst-case process parameters, operating temperature, and voltage. MEP step size increases with decreasing voltage and increasing temperature and decreases with increasing voltage and decreasing temperature. Applications that use the HRPWM feature must use the MEP scale factor optimization (SFO) software function supplied by TI. The SFO function helps to dynamically determine the number of MEP steps per EPWMCLK period while the HRPWM is in operation.

The HRPWM module has a built-in self-check and diagnostic capabilities that can be used to determine the appropriate MEP scale factor value for any operating condition. TI provides a C-callable library containing one SFO function that utilizes this hardware and determines the appropriate MEP scale factor. For a given system-clock frequency at a given temperature, a known MEP scale factor value is returned by the SFO determination function. The correct system-clock frequency operation is verified by comparing the MEP scale factor value returned with the expected value.

6.4.5.13 Information Redundancy Techniques

Information redundancy techniques can be applied using software as an additional runtime diagnostic. To provide diagnostic coverage for network elements outside the C2000 MCU (wiring harness, connectors, and transceiver) end-to-end safety mechanisms are applied. These mechanisms can also provide diagnostic coverage inside the C2000 MCU.

In the case of processing elements (CPU and CLA), this refers to multiple executions of the code and software-based cross checking to verify correctness. The multiple execution and result comparison can be based on implementations of either the same code executed multiple times or diversified software code. For details regarding the implementation, refer to the ISO26262-5, D.2.3.4.

In the case of the DMA, information redundancy techniques refer to additional information (besides the data payload) that verifies data integrity. For example, SECDED codes, parity codes, CRCs, and so forth enable information redundancy.

Typical control applications involve measuring three phase voltage and current. These values are either sampled directly using the on-chip ADC or sent to the TMS320F28P55x MCU by the sensors, which are captured using ECAP and so forth. In such scenarios, the correlation between input signals can be used to check the integrity (for example, if the three-phase voltage, V1, V2, and V3 is being measured, the function $V_1 + V_2 + V_3 = 0$ can be used to provide diagnostic coverage for input signal integrity).

In the case of SRAM and FLASH memory, the critical data, program, variables, and so forth can be stored redundantly and compared to their active counterparts before being used. Care must be taken to avoid compiler optimizing code containing redundant data or programs.

6.4.5.14 ECAP Application-Level Safety Mechanism

ECAP module outputs can be checked for saturation, zero width, or out-of-range based on the application requirement. While measuring the speed of rotating machinery, the application can set bounds on the measured speed based on the operating profile. Similar bound settings are possible for other application scenarios, like period and duty cycle measurement, decoding current or voltage from the duty cycle of the encoded current, or voltage sensors, and so forth. Online monitoring of periodic interrupts can also be performed for improved diagnostic coverage based on the application profile.

6.4.5.15 eQEP Quadrature Watchdog

eQEP peripheral contains a 16-bit watchdog timer that monitors the quadrature clock to indicate correct operation of the motion-control system. The eQEP watchdog timer is clocked from SYSCLKOUT/64 and the quadrate-clock event (pulse) resets the watchdog timer. If no quadrature-clock event is detected until a period match, then the watchdog timer times out and the watchdog interrupt flag is set. The timeout value is programmable through the watchdog period register.

6.4.5.16 eQEP Application-Level Safety Mechanisms

eQEP is typically used in closed-loop control applications to have direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine for use in high-performance motion and position-control systems. In such applications, monitoring eQEP outputs for saturation, zero value, or out-of-range is possible based on the application requirement. While estimating the speed and position of rotating machinery, the application can set bounds on the measured speed and position based on the operating profile. Online monitoring of periodic interrupts from eQEP can also be performed for improved diagnostic coverage based on the application profile.

6.4.5.17 QMA Error Detection Logic

The QEP mode adapter (QMA) is designed to extend the C2000 eQEP module capabilities to support the additional modes described in the *QMA Module* section in the device-specific technical reference manual. The QMA module has error-detection logic to detect illegal transitions on EQEPA and EQEPB input signals. The error and interrupt of the QMA module are integrated inside the eQEP module.

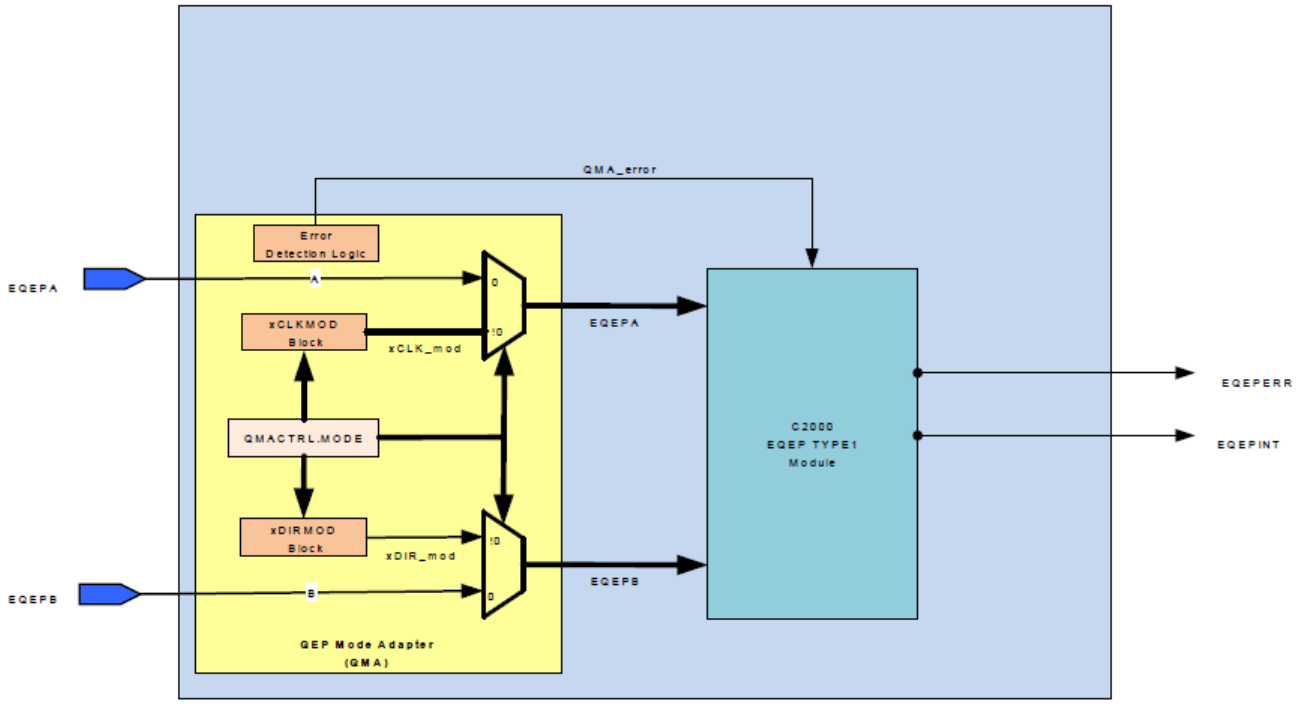

Figure 6-5. QMA Module Block Diagram

6.4.5.18 eQEP Software Test of Quadrature Watchdog Functionality

A software test can be used to test for basic functionality of the quadrature watchdog and to inject diagnostic errors and check for proper error response. Such a test can be executed at boot or periodically. The necessary software requirements are defined by the software implemented by the system integrator.

6.4.6 Analog I/O

6.4.6.1 Software Test of Function Including Error Tests

A software test can be utilized to test the basic functionality of the module and to inject diagnostic errors and check for proper error response. Such a test can be executed at boot or periodically. The necessary software requirements are defined by the software implemented by the system integrator.

Ideas for creating module-specific functionality tests and error tests are described below:

- DMA functionality can be checked by transferring a known good data from a source memory to the destination memory and checking for data integrity after the transfer. The transfer can be initiated using the software trigger available (CONTROL.PERINTFRC). On-chip timer can be used to profile the time required for such a data transfer.
- A software test of input and output X-BAR module can be performed by creating a loop (output X-BAR can be used as stimulus to input X-BAR) using the input and output X-BAR, sending a known test sequence at the input and observing the sequence at the final output. Integrity of ePWM X-BAR can be checked by sending the test stimulus and observing the response using ePWM trip or sync functionality.
- A software test of XINT functionality can be checked by configuring the input X-BAR and forcing the corresponding GPIO register to generate an interrupt. The diagnostic coverage can be enhanced by performing checks for the polarity (XINTxCR.POLARITY) and enabling (XINTxCR.ENABLE) functionality as well.
- ECAP and EQEP functionality can be checked by looping back the PWM or GPIO outputs to the respective module inputs, providing a known good sequence (as required by the module), and observing the module output. In the case of ECAP, the test can be done internally with the help of input X-BAR.
- ROM prefetch functionality can be checked using similar techniques as given in Section 6.4.3.33.
- The PWM module consists of time-base (TB), counter compare (CC), action qualifier (AQ), dead-band generator (DB), PWM chopper (PC), trip zone (TZ), event trigger (ET), and digital compare (DC) submodules. The individual submodules can be tested by providing appropriate stimulus using PWM and observing the response using one of the capture (timestamping) modules (eCAP, XINT, eQEP, and so forth). TI recommends covering the various register values associated with application configuration while performing the software test. Due to the regular, linear nature of the various submodules, getting high coverage using a software test is possible.
- A software test of SRAM wrapper logic must provide diagnostic coverage for arbitration between various initiators having access to the particular SRAM and correct functioning of access protection. This is in addition to the test used to provide coverage of SRAM bit cells (refer to Section 6.4.3.3).
- The interconnect (INC) functionality can be tested by writing complementary data-patterns, like 0xA5A5,0x5A5A and so forth, from processing units like CPU and CLA and reading back the data-patterns from registers of the IPs that are connected through different bridges. The readback data can be compared with the expected golden values to verify fault-free interconnect operation. This exercise can be repeated for different data width types of accesses (16/32 bits) and wide address ranges (as applicable) using both CPU and CLA. The CPU accesses can be repeated for different instances of peripherals used in application connected to various bridges, as shown in Figure 4-1.
- DAC has a set of control registers that can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A and so forth, in 16-bit access mode. All the registers can be read back and compared to expected values. Registers can be checked for the reset feature by configuring the registers to 0xA5A5 pattern, asserting soft reset of DAC, reading back the registers, and comparing the readback value with the expected reset value. Lock register can be checked to verify the DAC is set-once. Also, the registers, which are getting locked, must not update when written. To test core functionality of the DAC module, the DAC can be configured using software to provide a set of predetermined voltage levels. These voltage levels can be measured by external or internal ADC and results thus obtained can be cross checked against the expected value to verify proper operation. Extreme corner values of DAC, as per the application, can be programmed and tested to check the successful conversion of a digital to analog module across a valid range.
- Comparator subsystem (CMPSS) has a set of registers which can be checked by writing complementary data-patterns, like 0xA5A5, 0x5A5A, and so forth, in both 16 and 32-bit access modes. The registers can be read back and compared against expected values. These accesses can be covered by applicable initiators like DMA, CLA, and CPU. Features of the CMPSS module, such as ramp decrement, can be checked for

counting down of RAMPDLYA after the DAC module is loaded from RAMPDLYS by a rising PWMSYNC signal. The decrementer must reduce to zero and stay there until next reload from RAMPDLYS. Extreme values of RAMPDLYS can be configured before count down. Digital filter CTRIPHFILCTL/CTRIPLFILCTL registers can be checked by configuring them to a variety of N and T values and then verifying the COMPHSTS/COMPLSTS changes with the change in filter output. The applicable range of filter clock prescaler values (CTRIPLFILCLKCTL) can be exercised to verify that filter samples correctly.

- The general operation of the CPU timers can be tested by a software test by loading 32-bit counter register TIMH from period register PRDH. The PRDH register starts decrementing of the counter on every clock cycle. When counter reaches zero, a timer interrupt output generates an interrupt pulse. While testing the timer functionality, vary the timer prescale counter (TPR) value and input clocks by selecting clock source as SYSCLK, INTOSC1, INTOSC2, XTAL, or AUXPLLCLK. Test interrupts generation capability at the end of the timer counting. Check for the time overflow flag and timer reload (TRB) functions in the TCR register for correct functioning.
- A software test function in DCSM can be implemented independently in zone1, zone2, and unsecured zone to check DCSM functionality. Device security configurations are loaded from OTP to DCSM during the device boot phase. The test function can implement access filtering checks (read-write and execute permissions) to RAMs and flash sectors belonging to the same zone and different zone. An additional check for EXEONLY configuration can also be implemented for the RAMs and flash sectors to verify that all access (other than execute access) is blocked.

6.4.6.2 DAC to ADC Loopback Check

Integrity of DAC and ADC can be checked by monitoring the DAC output using ADC. DAC can be configured using software to provide a set of predetermined voltage levels. These voltage levels can be measured by the ADC and results thus obtained can be cross checked against the expected value to verify the DAC and ADC are functioning properly. This technique can be applied during run time to verify that proper voltage levels are being driven from DAC.

For more information on the DAC channels that can be sampled by ADC without external board level connections, refer to the device-specific data sheet or technical reference manual. While performing the loopback checks for 16-bit differential input mode, two DACs must be used to provide input to the ADC. To avoid common cause failures, TI recommends keeping the reference voltages of the ADC and DAC different while performing the test. In addition, the input signal to ADC must not be driven by any other sources while the test is being performed.

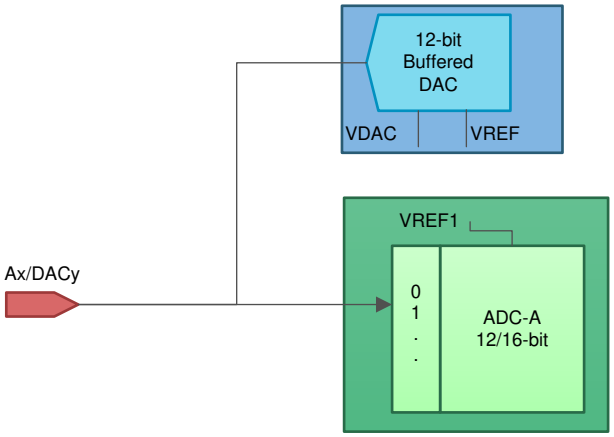

Figure 6-6. DAC to ADC Loopback

6.4.6.3 ADC Information Redundancy Techniques

Information redundancy techniques can be applied using software for providing runtime diagnostic coverage on ADC conversions. Time redundancy techniques can be applied where multiple conversions on same ADC are followed by comparison of results done in software. In addition, the correlation between input signals can be used to check the integrity (for example, if the three phase voltage, V_1, V_2, V_3 is being measured using ADC,

the function $V_1 + V_2 + V_3 = 0$ can be used to provide diagnostic coverage for input signal integrity and ADC conversion).

Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.6.4 Open and Short Detection Circuit for ADC

The open and short detection circuit allows customers to detect faults in the ADC input channel. This capability is valid only in single-ended mode. For system integrators using differential mode, the ADC must be configured in 12-bit single-ended mode to perform this test. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator. This capability is deprecated in few part numbers. Confirm the availability of the feature before using this diagnostic.

The following circuit and configuration, selected by programmable register bits to control switches S1, S2, S3, and S4, allows controlling input ADC channel to test for open and short conditions.

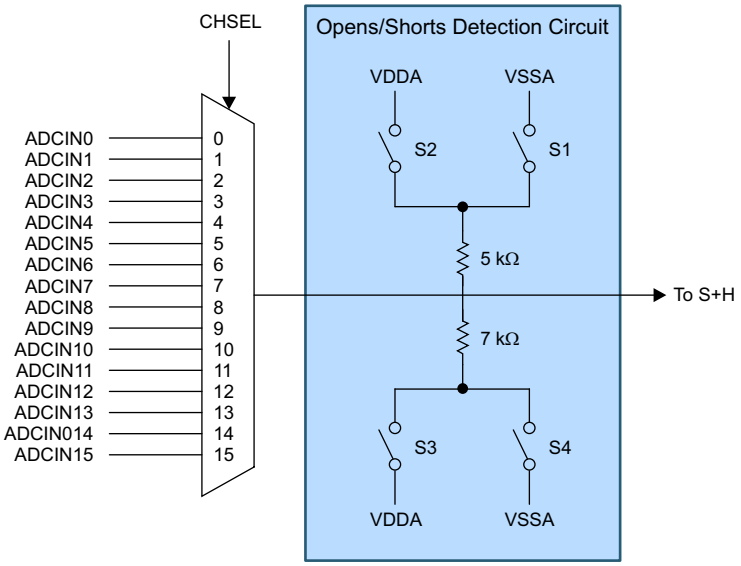

Figure 6-7. Open and Short Detection Circuit

Table 6-1. ADC Open and Short Detection Circuit Truth Table

ADCOSDETECT.DETECT CFG	Source Voltage	S4	S3	S2	S1	Drive Impedance
0	Off	Open	Open	Open	Open	Open
1	Zero Scale	Closed	Open	Open	Closed	5K || 7K
2	Full Scale	Open	Closed	Closed	Open	5K || 7K
3	5/12 VDDA	Open	Closed	Open	Closed	5K || 7K
4	7/12 VDDA	Closed	Open	Closed	Open	5K || 7K
5	Zero Scale	Open	Open	Open	Closed	5K
6	Full Scale	Open	Open	Closed	Open	5K
7	Zero Scale	Closed	Open	Open	Open	7K

6.4.6.5 Software Readback of Written Configuration

To verify proper configuration of memory-mapped registers in this module, TI recommends that software implements a test to confirm proper configuration of all control registers by reading back the contents. This test also provides diagnostic coverage for the peripheral bus interface and peripheral interconnect bridges.

Since CLA configuration registers are accessible to the C28x CPU only, this safety mechanism for the CLA module must be executed by C28x CPU.

6.4.6.6 Periodic Software Readback of Static Configuration Registers

Configuration registers are typically configured in the beginning and hold the value until the particular task execution. Periodic readback of configuration registers can provide a diagnostic for inadvertent writes or disruption of these registers.

The diagnostic coverage can be improved by extending the test to include readback of the flag registers that are expected to remain constant (for example, PLL lock status, EQEP phase error flag, and so forth) during the device operation as well. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

The diagnostic coverage of some peripherals can be further enhanced by applying some module-specific tests as follows:

- For improving the enhanced peripheral interrupt expander (EPIE) coverage, the PIE flag registers can be periodically checked to verify that all pending interrupts are serviced by reading the PIE flag registers (PIE_CTRL_REGS.PIEIFRx.all) and the peripheral interrupt flag registers.
- While serving the interrupt, the ISR routine can check for flag settings in peripheral and PIE to verify that correct interrupt is being serviced.

Since CLA configuration registers are accessible to C28x CPU only, this safety mechanism for the CLA module has to be executed by C28x CPU.

6.4.6.7 ADC Signal Quality Check by Varying Acquisition Window

External signal sources vary in the ability to drive an analog signal quickly and effectively. To achieve rated resolution, the signal source must charge the sampling capacitor in the ADC core to within 0.5LSBs of the signal voltage. The acquisition window is the amount of time the sampling capacitor is allowed to charge and is configurable for SOCx by the ADCSOCxCTL.ACQPS register. This configurable parameter can be also used to provide diagnostic coverage for the input signal path and ADC sampling capacitor logic. The test can be done by redundant conversion of the same input signal by ADC using the preset ACQPS configuration and an ACQPS configuration higher than the preset configuration. The results thus obtained have to be within a predefined range determined by the application and ADC specification parameters.

6.4.6.8 ADC Input Signal Integrity Check

ADC input signal integrity can be checked using a mix of hardware and software runtime diagnostic on ADC conversions. Filtering or plausibility check (for example, value fall in an expected range) of the converted values can be performed using some of the built-in hardware mechanisms available within the device. A plausibility check of the input signal can be checked with the help of a comparator by setting the proper high and low threshold values. The plausibility check of converted results can be checked by using an ADC post processing block.

6.4.6.9 Monitoring of ePWM by ADC

The ePWM outputs can be monitored for correct operation by the ADC using board-level feedback, as indicated in Figure 6-8. The technical details for implementing such a loopback, like signal resolution and so forth, is provided in the reference link [9]. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

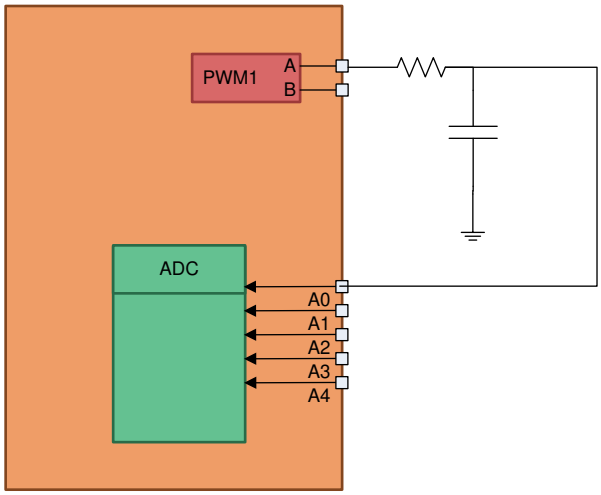

Figure 6-8. Monitoring of ePWM by ADC

6.4.6.10 Hardware Redundancy

Hardware redundancy techniques can be applied using hardware, or as a combination of hardware and software, to provide runtime diagnostic. In this implementation, redundant hardware resources are utilized to provide diagnostic coverage for elements within and outside (wiring harness, connectors, and transceiver) the TMS320F28P55x MCU.

In case of peripherals like GPIO, XBAR, PWM, OTTO, DAC, CMPSS, and XINT, hardware redundancy can be implemented by having multichannel parallel outputs (where independent outputs are used for transmitting information and failure detection is carried out using internal or external comparators) or input comparison and voting (comparison of independent inputs to comply with a defined tolerance range, time and value). In such scenarios, the system can be designed so the failure of one input or output does not cause the system to go into a dangerous state. While servicing the error conditions (redundancy conditions), as in two redundant sources tripping the PWM, always read back the status flags and verify that both sources are active while tripping and thus providing latent fault coverage for the trip logic.

In case of peripherals like ADC, ECAP, EQEP, and so forth, hardware redundancy can be implemented by having multiple instances of the peripheral sample the same input and simultaneously perform the same operation followed by a cross check of the output values.

In case of communication peripherals like MCAN, SPI, SCI, and so forth, hardware redundancy during signal reception can be implemented by having multiple instances of the peripheral receive the same data followed by a comparison to verify data integrity. Hardware redundancy during transmission can be employed by having a complete redundant signal path (wiring harness, connectors, and transceiver) from the transmitter to the receiver or by sampling the transmitted data by a redundant peripheral instance followed by a data integrity check.

While implementing hardware redundancy for ADC and DAC modules, additional care must be taken to verify common-cause failures do not impact both instances in the same way. Reference voltage sources, configured for redundant module instances, must be independent. Additionally, ADC SOC trigger sources, used for redundant ADC instances, must be configured to different PWM module instances. In case of a DAC module, the comparator can be implemented using an external device.

While implementing hardware redundancy for the PWM module, TI recommends that the PWM module instance used is part of separate sync chains. This requirement is to avoid a common-cause failure on a sync signal that affects both PWM modules in the same way.

While implementing hardware redundancy for the GPIO module, TI recommends using GPIO pins from different GPIO groups to avoid common-cause failures.

6.4.6.11 Lock Mechanism for Control Registers

The module contains a lock mechanism for protection of critical control registers. Once the associated LOCK register bits are set, the write access to the registers are blocked. Locked registers cannot be updated by software; once locked, only reset can unlock the registers.

6.4.6.12 DAC to Comparator Loopback Check

The DAC outputs can be looped back to comparator inputs to check whether the outputs being driven are at proper voltage levels. The connections need to be provided externally on the board to enable this check. Higher diagnostic coverage can be obtained by configuring tighter limits to the comparator. This technique can also be used to detect control flow errors which cause the DAC output to be set at a value outside the applications safe operating range.

6.4.6.13 Lock Mechanism for Control Registers

The module contains a lock mechanism for protection of critical control registers. Once the associated LOCK register bits are set, the write access to the registers are blocked. Locked registers cannot be updated by software; once locked, only reset can unlock the registers.

6.4.6.14 CMPSS Ramp Generator Functionality Check

CMPSS ramp generation functionality is used in certain control applications (for example, peak current mode control). The functionality of ramp generator can be checked by reading back the contents of DACHVALA register and verifying that the register value is periodically updated based on the RAMPDLY, RAMPDECVAL, and RAMPMAXREF. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.6.15 PGA to ADC Loopback Test

Integrity of PGA can be checked driving PGAx_IN input with a known set of values and monitoring output after conversion using on-chip ADC. DAC can be configured using software to provide a set of predetermined voltage levels. With an external connection, output of DAC can be connected to the PGAx_INP pin. This pin can be routed to both the PGA and ADC in parallel. ADC can convert both PGAx_OUT and the signal on the pin, and with SW scaling, can compare the ADC output to test if PGA amplified the input signal to an appropriate scale. PGAx_INM can be connected internally, or driven, as the system requires.

While using different ADCs to convert the native signal from the pin and the PGA output is possible, using the same ADC to eliminate any other error sources from the diagnostic is preferred.

Figure 6-9 shows the conceptual setup for testing the PGA using DAC and ADC.

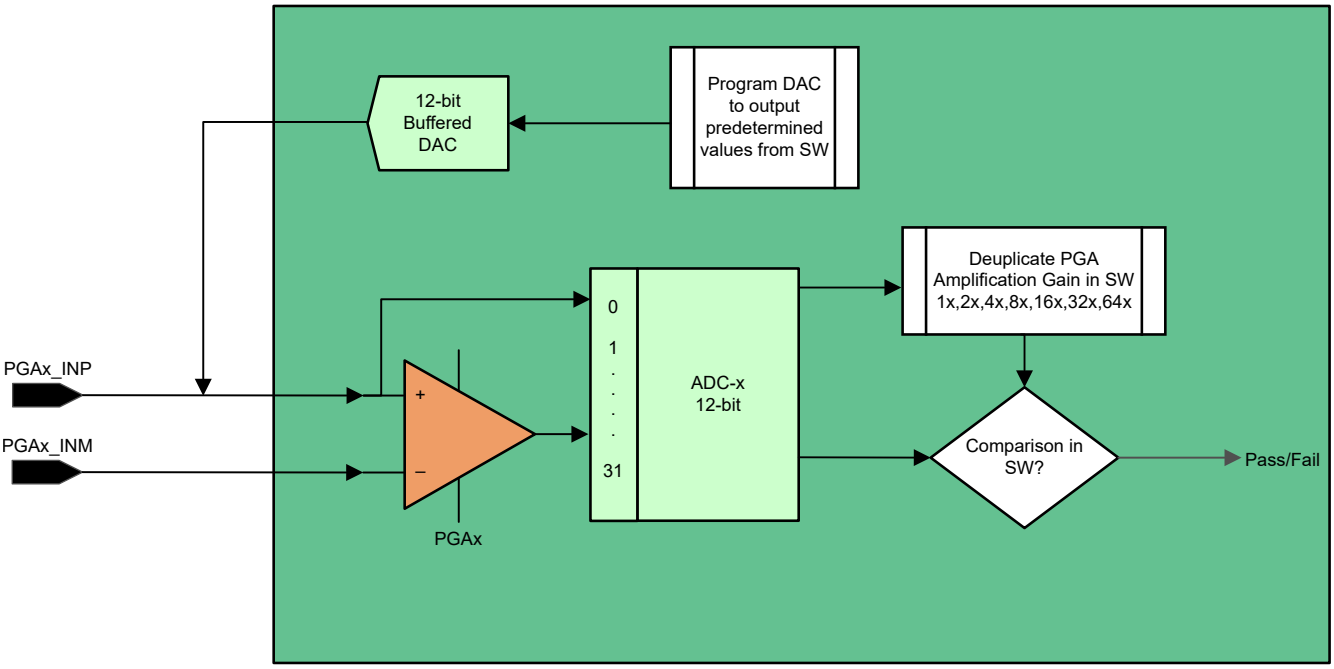

Figure 6-9. Testing PGA Using ADC and DAC

6.4.7 Data Transmission

6.4.7.1 Software Test of Function Using I/O Loopback

Most communication modules support digital or analog loopback capabilities for the I/Os. To confirm the implemented loopback capabilities of the module, refer to the device-specific technical reference manual. Digital loopback tests the signal path to the module boundary. Analog loopback tests the signal path from the module to the I/O cell with output driver enabled. For better results, include the I/O loopback in any tests of functionality.

6.4.7.2 Information Redundancy Techniques Including End-to-End Safing

Information redundancy techniques can be applied using software as an additional runtime diagnostic. There are many techniques that can be applied, such as a readback of written values and multiple reads of the same target data with a comparison of results.

To provide diagnostic coverage for network elements outside the C2000 MCU (wiring harness, connectors, and transceiver) end-to-end safety mechanisms are applied. These mechanisms can also provide diagnostic coverage inside the C2000 MCU. There are many different schemes applied, such as additional message checksums, redundant transmissions, time diversity in transmissions, and so forth. Most commonly, checksums are added to the payload section of a transmission to verify the correctness of a transmission. The checksums, sequence counter, and timeout expectations (or timestamp) are then included with the original data, in addition to any protocol-level parity and checksums. As these are generated and evaluated by the software at either end of the communication, the whole communication path is safed, resulting in end-to-end safing.

Any end-to-end communication diagnostics implemented must consider the failure modes and potential mitigating safety measures described in IEC 61784-3:2016 and summarized in IEC 61784-3:2016, Table 1.

6.4.7.3 Transmission Redundancy

The information is transferred several times in sequence using the same module instance and then compared to one another. When the same data path is used for duplicate transmissions, transmission redundancy is only useful for detecting transient faults. The diagnostic coverage can be improved by sending inverted data during the redundant transmission.

To provide diagnostic coverage of device interconnects and EMIF, readback of written data (in case of data writes) and multiple readbacks of information (in case of data reads) can be employed.

6.4.7.4 Periodic Software Readback of Static Configuration Registers

Configuration registers are typically configured in the beginning and hold the value until the particular task execution. Periodic readback of configuration registers can provide a diagnostic for inadvertent writes or disruption of these registers.

The diagnostic coverage can be improved by extending the test to include readback of the flag registers that are expected to remain constant (for example, PLL lock status, EQEP phase error flag, and so forth) during the device operation as well. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

The diagnostic coverage of some peripherals can be further enhanced by applying some module-specific tests as follows:

- For improving the enhanced peripheral interrupt expander (EPIE) coverage, the PIE flag registers can be periodically checked to verify that all pending interrupts are serviced by reading the PIE flag registers (PIE_CTRL_REGS.PIEIFRx.all) and the peripheral interrupt flag registers.
- While serving the interrupt, the ISR routine can check for flag settings in peripheral and PIE to verify that correct interrupt is being serviced.

Since CLA configuration registers are accessible to C28x CPU only, this safety mechanism for the CLA module has to be executed by C28x CPU.

6.4.7.5 Software Readback of Written Configuration

To verify proper configuration of memory-mapped registers in this module, TI recommends that software implements a test to confirm proper configuration of all control registers by reading back the contents. This test also provides diagnostic coverage for the peripheral bus interface and peripheral interconnect bridges.

Since CLA configuration registers are accessible to the C28x CPU only, this safety mechanism for the CLA module must be executed by C28x CPU.

6.4.7.6 Data Parity Error Detection

The LIN module supports detection of parity error on received data. An error flag bit is set when a parity error is detected in the received data. In address-bit mode, the parity is calculated on the data and address bit fields of the received frame. In idle-line mode, only the data is used to calculate parity. An error is generated when a character is received with a mismatch between the number of 1s and the parity bit. If the parity function is disabled (that is, SCIGCR1.2 = 0), the PE flag is disabled and read as 0. Detection of a parity error causes the LIN to generate an error interrupt if the SCISETINT.SETPEINT bit = 1.

6.4.7.7 SCI Overrun Error Detection

If the SCI RX buffer receives new data before the previous data has been read, the existing data is overwritten and lost. If this occurs, the SCI hardware can flag the error and generate an interrupt to the CPU. This feature must be enabled and configured in software.

6.4.7.8 SCI Frame Error Detection

When receiving serial data, each byte of information on the SCI has an expected format. If the received message does not match the expected format, the SCI hardware can flag an error and generate an interrupt to the CPU. This feature must be enabled and configured in software.

6.4.7.9 LIN Physical Bus Error Detection

The LIN module supports detection of physical bus error conditions, an error flag is set and an interrupt is generated. A physical bus error (PBE) is detected by an initiator if no valid message can be generated on the bus (bus shorted to GND or VBAT). The bit monitor detects a PBE during the header transmission if a synch break cannot be generated (for example, because of a bus shortage to VBAT) or if a synch break delimiter cannot be generated (for example, because of a bus shortage to GND).

6.4.7.10 LIN No-Response Error Detection

The LIN module supports detection of no-response error detection. An error flag bit is set and an interrupt is generated when the header of the initiator does not complete a response within TFRAME_MAX (the maximum time length allowed for response). The no-response error flag is cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.

6.4.7.11 Bit Error Detection

When this module transmits information onto the bus, the module can monitor the bus to verify that the transmitted information is appearing as expected on the bus. If the expected values are not read back from the bus, the hardware can flag the error and signal an interrupt to the CPU. This feature must be enabled and configured in software.

6.4.7.12 LIN Checksum Error Detection

The LIN module supports the detection of checksum errors on received data. An error flag bit is set and interrupt is generated when there is a checksum error detected by a receiving node. The type of checksum to use depends on the CIGCR1.CTYPE bit (for example, classic checksum is compatible with LIN 1.3 target nodes and enhanced checksum is compatible with LIN 2.0 and newer target nodes). The checksum error flag is cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.

6.4.7.13 LIN ID Parity Error Detection

The LIN module supports detection of parity errors on the ID field. If parity is enabled, an ID parity error (PE) is detected if any of the two parity bits (even or odd) of the sent ID byte are not equal to the calculated parity on the receiver node. The two parity bits (even or odd) are generated using the mixed parity algorithm. If an ID parity error is detected, the ID parity error is flagged and the received ID is not valid.

6.4.7.14 SCI Break Error Detection

An SCI break detect condition occurs when the SCIRXD is low for ten bit periods following a missing stop bit. This action sets the BRKDT flag bit (SCIRXST, bit 5) and initiates an interrupt.

6.4.7.15 Communication Access Latency Profiling Using On-Chip Timer

Each communication message takes a fixed number of system-clock cycles for completing the transaction. The initiator can detect the transaction completion based on message acknowledge signaling from the target. The on-chip timer module can be used for profiling the time required for completing each transaction.

6.4.7.16 Software Test of Function Including Error Tests Using EPG

Embedded pattern generator (EPG) can be used to check the functionality of several communications on the device by driving a known pattern on the receive input pin and comparing the received message. EPG loopback can also be used to inject errors on the receive line and to check the diagnostics of the peripherals, such as the CAN CRC logic operation, which is indicated by an error interrupt. Refer to the device-specific technical reference manual for more information about EPG.

6.4.7.17 Software Test of Function Using I/O Loopback

Most communication modules support digital or analog loopback capabilities for the I/Os. To confirm the implemented loopback capabilities of the module, refer to the device-specific technical reference manual. Digital loopback tests the signal path to the module boundary. Analog loopback tests the signal path from the module to the I/O cell with output driver enabled. For better results, include the I/O loopback in any tests of functionality.

6.4.7.18 SPI Data Overrun Detection

If the SPI RX buffer receives new data before the previous data has been read, the existing data is overwritten and lost. If this occurs, SPI hardware can flag the error and generate an interrupt to the CPU. This feature must be enabled and configured in software.

6.4.7.19 Hardware Redundancy

Hardware redundancy techniques can be applied using hardware, or as a combination of hardware and software, to provide runtime diagnostic. In this implementation, redundant hardware resources are utilized to provide diagnostic coverage for elements within and outside (wiring harness, connectors, and transceiver) the TMS320F28P55x MCU.

In case of peripherals like GPIO, XBAR, PWM, OTTO, DAC, CMPSS, and XINT, hardware redundancy can be implemented by having multichannel parallel outputs (where independent outputs are used for transmitting information and failure detection is carried out using internal or external comparators) or input comparison and voting (comparison of independent inputs to comply with a defined tolerance range, time and value). In such scenarios, the system can be designed so the failure of one input or output does not cause the system to go into a dangerous state. While servicing the error conditions (redundancy conditions), as in two redundant sources tripping the PWM, always read back the status flags and verify that both sources are active while tripping and thus providing latent fault coverage for the trip logic.

In case of peripherals like ADC, ECAP, EQEP, and so forth, hardware redundancy can be implemented by having multiple instances of the peripheral sample the same input and simultaneously perform the same operation followed by a cross check of the output values.

In case of communication peripherals like MCAN, SPI, SCI, and so forth, hardware redundancy during signal reception can be implemented by having multiple instances of the peripheral receive the same data followed by a comparison to verify data integrity. Hardware redundancy during transmission can be employed by having a complete redundant signal path (wiring harness, connectors, and transceiver) from the transmitter to the receiver or by sampling the transmitted data by a redundant peripheral instance followed by a data integrity check.

While implementing hardware redundancy for ADC and DAC modules, additional care must be taken to verify common-cause failures do not impact both instances in the same way. Reference voltage sources, configured for redundant module instances, must be independent. Additionally, ADC SOC trigger sources, used for redundant ADC instances, must be configured to different PWM module instances. In case of a DAC module, the comparator can be implemented using an external device.

While implementing hardware redundancy for the PWM module, TI recommends that the PWM module instance used is part of separate sync chains. This requirement is to avoid a common-cause failure on a sync signal that affects both PWM modules in the same way.

While implementing hardware redundancy for the GPIO module, TI recommends using GPIO pins from different GPIO groups to avoid common-cause failures.

6.4.7.20 FSI Data Overrun and Underrun Detection

The FSI module supports detection of data overrun or underrun conditions.

- Receive buffer overrun - This event indicates that the transfer of data from the receive shift register to the receiver data buffer register overwrites unread data already in received data.
- Receive buffer underrun – This event indicates that software reads the buffer while the buffer is empty.
- Transmit buffer overrun – This event occurs if a piece of data is overwritten before the data has been transmitted.
- Transmit buffer underrun – This event occurs when the transmitter tries to read data from a location which has not yet been written.

A flag bit is set and an interrupt is generated when a data overrun and underrun error occurs and a corresponding register bit is enabled.

6.4.7.21 FSI Frame Overrun Detection

The FSI module supports detection of frame overrun events. This event indicates that a new frame has been received while the FRAME_DONE flag was still set. A flag is set and an interrupt is generated if a corresponding register bit is enabled.

6.4.7.22 FSI CRC Framing Checks

The FSI module supports detection of CRC framing error conditions. A CRC error is generated when the received expected CRC value and the computed CRC value do not match. A flag is set and an interrupt is generated if enabled.

6.4.7.23 FSI ECC Framing Checks

The FSI module supports detection of ECC framing error conditions. The module supports a 16-bit or 32-bit ECC computation module in both transmitter and receiver mode. In transmit mode, software can configure the FSI registers to compute the ECC value on the data in transmit buffer and include the data as part of the transmit frame in receive mode. Software can feed the ECC module with received data and ECC value to detect and autocorrect single bit errors in data or detect multibit errors in received data and invalidate the received data.

Note

The ECC check supported in the FSI module requires software assistance. The hardware in the FSI module supports ECC computation, but the task of writing data and checking the ECC error must be handled in software.

6.4.7.24 FSI Frame Watchdog

The FSI module supports detection of frame watchdog timeout events. This event indicates that the frame watchdog timer has timed out. The conditions of this timeout are set using the RX_FRAME_WD_CTRL register. As soon as the start of frame phase is detected, the frame watchdog counter starts counting from 0. The end-of-frame phase must complete by the time the watchdog counter reaches the reference value. If this does not happen, the watchdog times out and this event is generated. If this event occurs, the receiver must undergo a soft reset and subsequent resynchronization to properly operate. When this condition occurs, a flag is set and an interrupt is generated if enabled.

6.4.7.25 FSI RX Ping Watchdog

The FSI module supports detection of RX ping watchdog timeout events. This event indicates that the ping watchdog timer has timed out. The receiver has not received a valid frame within the time period specified in the RX_PING_WD_REF register. The ping frame triggered interrupt is generated when the ping frame has been triggered and a corresponding register bit is enabled. This bit is set when the ping counter has timed out. An interrupt is generated if a corresponding register bit is enabled. On the transmitter, the ping frame can be set up and transmitted without any further software or DMA intervention. Ping frames can be transmitted by automatic ping timer, software, or external triggers.

6.4.7.26 FSI Tag Monitor

The FSI module supports tag field in the transfer frame. The module contains a 4-bit FRAME_TAG field of the last successfully received frame. FSI tag monitor checks must be implemented in software. The tag field for each frame on the receive side can be monitored through software and verified against expected values. In addition to FRAME_TAG, the FSI module supports the user data as a fully user-configurable data field, available in data frames. The user data to be transmitted is set by writing to TX_FRAME_TAG_UDATA.USER_DATA. The received user data is stored in RX_FRAME_TAG_UDATA.USER_DATA.

6.4.7.27 FSI Frame Type Error Detection

The FSI module supports detection of frame type errors. This error indicates that an invalid frame type has been received. If this error occurs, the receiver must undergo a soft reset and subsequent resynchronization to properly operate. An interrupt is generated if a corresponding register bit is enabled.

6.4.7.28 FSI End-of-Frame Error Detection

This error indicates that an invalid end-of-frame bit pattern has been received. If this error occurs, the receiver must undergo a soft reset and subsequent resynchronization to properly operate. An interrupt is generated if corresponding register bit is enabled.

6.4.7.29 FSI Register Protection Mechanisms

As a fault avoidance safety measure for key registers of the FSI module, registers are protected by EALLOW privilege, register keys, and an initiator register lock. These protections avoid spurious writes or unintentional modifications to these registers. Some bits in the FSI registers are protected by a key. To write to these bits, the key must be written at the same time.

The control register lock prevents any writes to the control registers until the lock is released. To set the control register lock, write 0xA501 to RX_LOCK_CTRL and TX_LOCK_CTRL for the receiver and transmitter, respectively.

6.4.7.30 Hardware Disable of JTAG Port

The JTAG debug port can be physically disabled to prevent JTAG access in deployed systems. The recommended scheme is to hold test clock (TCK) to ground and hold test mode select (TMS) high. Disabling of the JTAG port also provides coverage for inadvertent activation of many debug and trace activities; since these are often initiated using an external debug tool that writes commands to the device using the JTAG port.

6.4.7.31 Parity in Message

This module supports insertion of a parity bit into the data payload of every outgoing message by hardware. Evaluation of incoming message parity is also supported by hardware. Detected errors generate an interrupt to the CPU.

6.4.7.32 I2C Data Acknowledge Check

When a node on the I2C network receives a byte (address or data), I2C sends an acknowledgment that the address is acknowledged or the data byte is received successfully. When a transmitted message is not acknowledged by the recipient I2C, the transmitting I2C flags NACK. The necessary software requirements are defined by the system integrator. A good example of a function that requires software implementation is the requirement to transfer 4 bytes of data and also send CRC as a 5th byte. The device software can be designed so the acknowledge is not provided if the data and CRC do not match.

6.4.7.33 I2C Access Latency Profiling Using On-Chip Timer

Each I2C message takes a fixed number of system clock cycles for completing the transaction. The initiator can detect the transaction completion based on message acknowledge signaling from the target. The on-chip timer module can be used for profiling the time required for completing each transaction.

6.4.7.34 PMBus Protocol CRC in Message

The PMBus module supports detection of data corruption during transfer using a packet error check (PEC) value feature. When this feature is enabled, the feature forces the PMBus transmitter interface to append a PEC byte onto the end of the message. Receiver hardware checks the last byte in a message for a valid packet error check value corresponding to the number of bytes in the message.

6.4.7.35 PMBus Clock Timeout

The PMBus module supports detection of a stuck fault on clock (SCL) pin. If the SCL pin is stuck during communication to either the high or low value for a duration more than the programmed value (in PMBTIMHIGHTIMOUT and PMBTIMLOWTIMOUT registers), an interrupt is generated and respective flags are set in PMBSTS status register.

6.4.7.36 PWM Trip by MCAN

PWM can be tripped by the Rx events on MCAN. PWM can be used to stop the PWM generation from receiving the special messages (special identifier for which the corresponding Rx event can be generated). Any message with an identifier corresponding to the error message is sent to indicate an error condition and stop the PWM generation.

This SM can be used to check the filter event output of the MCAN.

6.4.7.37 Software Test of SRAM

Testing the integrity of SRAM (bit cells, address decoder, and sense amplifier logic) is possible using the CPU. Based on the safety requirement, this test can be performed at start-up or during application time. If the SRAM contents are static, a CRC check using VCU can be performed in place of destructive test (a test where memory contents must be restored after the test). For details on implementing this particular test, check the safety package delivered with this specific C2000 MCU device.

6.4.7.38 SRAM ECC

Selected on-chip SRAMs support SECDED ECC diagnostic with separate ECC bits for data and address. For the specific address ranges that support ECC, refer to the TMS320F28P55x MCU device-specific data sheet. In SECDED scheme, a 21-bit code word is used to store the ECC data calculated independently for each 16 bit of data and for address. The ECC logic for the SRAM access is located in the SRAM wrapper. The ECC is evaluated directly at the memory output and data is sent to CPU after the data-integrity check. The data and address interconnects from SRAM to the CPU is not protected using ECC. Detected, correctable errors are corrected and monitoring the number of corrected errors is possible. The SRAM wrapper can be configured to trigger an interrupt once the number of corrected errors crosses a threshold. Uncorrectable, SRAM errors trigger an NMI and the ERRORSTS pin is asserted. The ECC logic for the SRAM is enabled at reset. For more information regarding memories supporting ECC, refer to the device-specific data sheet.

6.4.7.39 Bit Multiplexing in SRAM Memory Array

The SRAM modules implemented in the TMS320F28P55x MCU device family have a bit multiplexing scheme where the bits accessed to generate a logical (CPU) word are not physically adjacent. This scheme helps to reduce the probability of physical, multibit faults resulting in logical, multibit faults, rather the bits manifest as multiple, single-bit faults. The SECDED SRAM ECC diagnostic can correct a single-bit fault and detect a double-bit fault in a logical word. Similarly, the SRAM parity diagnostic can detect single-bit faults. This scheme improves the usefulness of the SRAM ECC and parity diagnostic. Bit multiplexing is a feature of the SRAM and cannot be modified by the software.

6.4.7.40 MCAN Stuff Error Detection

In the CAN message protocol, several of the frame segments are coded through bit stuffing. Whenever a transmitter detects five consecutive bits of identical value in the bit stream to be transmitted, the CAN message protocol automatically inserts a complementary bit into the actual transmitted bit stream. If a sixth consecutive equal bit is detected in a received segment that should have been coded by bit stuffing, the CAN module flags a stuff error. Error response and any necessary software requirements are defined by the system integrator.

6.4.7.41 MCAN Form Error Detection

Certain types of frames have a fixed format per the CAN protocol. When a receiver receives a bit in one of these frames that violates the protocol, the module flags a form error. Error response and any necessary software requirements are defined by the system integrator.

6.4.7.42 MCAN Acknowledge Error Detection

When a node on the CAN network receives a transmitted message, the CAN network sends an acknowledgment that the message was successfully received. When a transmitted message is not acknowledged by the recipient node, the transmitting CAN flags an acknowledge error. Error response and any necessary software requirements are defined by the system integrator.

6.4.7.43 CRC in Message

This module appends a CRC word along with the message. The CRC values are calculated and transmitted by the transmitter and then recalculated by the receiver. If the CRC value calculated by the receiver does not match the transmitted CRC value, a CRC error is flagged. Error response and any necessary software requirements are defined by the system integrator.

6.4.7.44 Software Test of ECC Logic

Testing the functionality of the SRAM ECC is possible by injecting single-bit and double-bit errors in test mode, performing reads on locations with ECC errors, and then checking for the error response.

Flash ECC logic can be checked with the help of ECC test field ECC_TEST_EN in the FECC_CTRL register. This technique causes an output comparison failure between the redundant ECC logic upon a flash read access. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

For additional details on implementing this diagnostic for SRAM and FLASH memory, refer to the *Application Test Hooks for Error Detection and Correction* and *Mechanism to Check the Correctness of ECC Logic* sections in the device-specific technical reference manual.

6.4.7.45 Timeout on FIFO Activity

MCAN implements a timeout counter that is programmed during the INIT phase for the module. The timeout function can be continuous (preset by writing to TOCC.TOP on a periodic basis before the timeout function expires) or associated with Tx, Rx0, or Rx1 FIFOs (a FIFO empty presets the counter and first-push starts the down counting). A CAN system implementing a periodic messaging can use the timeout diagnostic to ascertain the presence of a system heartbeat.

6.4.7.46 Timestamp Consistency Checks

MCAN implements a timestamp for received and transmitted messages. An external timestamp counter is required for CAN FD messages. The timestamp counter provided by MCAN is equipped with a prescaler for tradeoff between resolution and wraparound period. The timestamp counter value is stored in the message buffer for each transmitted or received message. Software can perform sanity checks on messages to determine if the messages have been sent in the order expected by the system as a diagnostic. For example, multiple messages with the same timestamp (taking into consideration the wraparound time) are not expected as the CAN protocol can carry one message at a time. End-to-end safing that includes numbering the messages can be used to indicate linear incrementing timestamps that software can verify.

6.4.7.47 Tx-Event Checks

Tx handler controls the message transfer from the external message RAM to the CAN core. A Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported. Each element stores information about transmitted messages. By reading the Tx event FIFO the host CPU gets this information in the order the messages were transmitted.

6.4.7.48 Interrupt on Message RAM Access Failure

One of the interrupt sources is the MRAF (Message RAM access failure in register IR.MRAF). The flag and interrupt is set, when the Rx handler:

- Has not completed acceptance filtering, or storage, of an accepted message until the arbitration field of the following message has been received. In this case, acceptance filtering or message storage is aborted and the Rx handler starts processing the following message.
- Was not able to write a message to the message RAM. In this case, message storage is aborted. In both cases, the FIFO put index is not updated respectively, the new data flag for a dedicated Rx buffer is not set, and a partly stored message is overwritten when the next message is stored to this location. The flag is also set when the Tx handler is not able to read a message from the message RAM in time. In this case, message transmission is aborted. In case of a Tx handler access failure, the M_CAN is switched into restricted operation mode. To leave restricted operation mode, the host CPU must reset CCCR.ASM.
  - 0 = No message RAM access failure occurred
  - 1 = Message RAM access failure occurred

6.4.7.49 Software Test of Function Including Error Tests Using EPG

Embedded pattern generator (EPG) can be used to check the functionality of several communications on the device by driving a known pattern on the receive input pin and comparing the received message. EPG loopback

can also be used to inject errors on the receive line and to check the diagnostics of the peripherals, such as the CAN CRC logic operation, which is indicated by an error interrupt. Refer to the device-specific technical reference manual for more information about EPG.

Chapter 7
References

1. Texas Instruments, *Calculating Useful Lifetimes of Embedded Processors*, application note.
2. JEDEC, *Moisture/Reflow Sensitivity Classification for Nonhermetic Solid State Surface Mount Devices*.
3. JEDEC, *Handling, Packing, Shipping and Use of Moisture/Reflow Sensitive Surface Mount Devices*.
4. International Electrotechnical Commission, *IEC61508 Functional Safety of Electrical/Electronic/ Programmable Electronic Safety-Related Systems*, 1998.
5. International Standard ISO, *ISO 26262–Road Vehicles-Functional Safety*, vol. 26262, 2018.
6. STUDYLIB, Standardized E-Gas Monitoring Concept for Gasoline and Diesel Engine Control Units, 2024.
7. J. Astruc and N. Becker, *Toward the Application of ISO 26262 for Real-Life Embedded Mechatronic Systems*, International Conference on Embedded Real Time Software and Systems. ERTS2, 2010.
8. International Standard ISO, *ISO26262–Road Vehicles-Functional Safety, Part 5: Product development at the hardware level, Appendix D*, vol. 26262, 2018.
9. Texas Instruments, *Using PWM Output as a Digital-to-Analog Converter on a TMS320F280x Digital Signal Controller*, application note.
10. W. M. Goble and H. Cheddie, *Safety Instrumented Systems Verification: Practical Probabilistic Calculations*. Isa, 2004.
11. Texas Instruments, *TMS320C28x FPU Primer*, application note.
12. Texas Instruments, *Online Stack Overflow Detection on the TMS320C28x DSP*, application note.
13. Texas Instruments, *TMS320F28P55x Real-Time Microcontrollers Data Sheet*, data sheet.
14. Texas Instruments, *TMS320F28P55x Real-Time Microcontrollers Technical Reference Manual*, technical reference manual.
15. Texas Instruments, *C2000™ Hardware Built-In Self-Test*, application note.
16. IEC-60730 official website, Available online at http://www.iec.ch.
17. IEC-61784 official website, Available online at http://www.iec.ch.
18. Texas Instruments, *Texas Instrument's Functional Safety Hardware Development Process*, functional safety certificate.

Chapter 8

f28p55x Summary of Safety Features and Diagnostic

Table 8-1. Summary Table Legend

Unique Identifier	Identifier used to reference the contents.
Safety Feature or Diagnostic	Safety feature
Usage	Each test listed in this chart can be one of three types: a "diagnostic" test, a "test for diagnostic", or a "fault avoidance" measure. Diagnostic: Provides coverage for faults on a primary function of the device. It may, in addition, provide fault coverage on other diagnostics, and can therefore be also used as a test-for-diagnostic in certain cases Test-for-Diagnostic Only: Does not provide coverage for faults on a primary function of the device. It's only purpose is to provide fault coverage on other diagnostics Fault Avoidance: This is typically a feature used to improve the effectiveness of a related diagnostic.
Diagnostic Type	Hardware - A diagnostic which is implemented by TI in silicon and can communicate error status upon the detection of failures. It may require software to enable the diagnostic and/or to take action upon the detection of a failure. Software - A test recommended by TI which must be created by the software implementer. This test may use additional hardware implemented on the device by TI. Hardware / Software - A test recommended by TI which requires both, diagnostic hardware which has been implemented in silicon by TI, and which requires software that must be created by the software implementer. System - A diagnostic implemented externally of the microcontroller
Diagnostic Operation	This can be one among the following: (i) Bootup (enabled by default) (ii) Continuous - Enabled at reset: Hardware safety mechanism that is enabled by default at reset. (iii) Continuous - Enabled by software: Hardware safety mechanism that needs to be enabled by software. (iv) On demand (Software defined): Software or Hardware-software safety mechanism that gets activated in the diagnostic test interval by the software (v) System defined: Implemented by the system.
Test Execution Time	This column lists the time required for this diagnostic to complete.
Action on Detected Fault	The response this diagnostic takes when an error is detected. For software-driven tests, this action is often software implementation-dependent.

Table 8-1. Summary Table Legend (continued)

Error Reporting Time	Typical time required for diagnostic to indicate a detected fault to the system. For safety mechanisms where fault detection time is known, this value is indicated. For software-driven tests, this time is often software implementation-dependent.

Table 8-2. Summary of Safety Features and Diagnostic

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
Power Supply (Power)	PWR1	External Voltage Supervisor	Diagnostic	System	System defined	System defined	System defined	System defined
	PWR2	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	SYS1	Multi-Bit Enable Keys for Control Registers	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS2	Lock Mechanism for Control Registers	Fault avoidance	Hardware	Continuous - Enabled by software	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS3	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SYS4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SYS5	Online Monitoring of Temperature	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SYS8	EALLOW and MEALLOW Protection for Critical Registers	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	NA (Fault avoidance technique)
	CLK6	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	PWR4	Brownout Reset (BOR)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset	Typically less than 1µs
Clock	CLK1	Missing Clock Detect (MCD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion and PLL reference clock switch to INTOSC1	0.82ms
	CLK2	Clock Integrity Check using CPU Timer	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLK3	Clock Integrity check using HRPWM	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLK5	External Monitoring of Clock via XCLKOUT	Diagnostic	System	System defined	System defined	System defined	System defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	CLK6	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	CLK7	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	CLK8	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLK9	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLK10	Software Test of Watchdog (WD) Operation	Test for diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLK12	Software Test of Missing Clock Detect Functionality	Test for diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLK13	PLL Lock Profiling using On-Chip Timer	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLK14	Peripheral Clock Gating (PCLKCR)	Fault avoidance	Hardware - Software	On demand (Software defined)	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	CLK17	Dual clock comparator (DCC) - Type 2	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion or interrupt to CPU based on error severity	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
APLL	APLL1	Clock integrity check using DCC	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	APLL2	PLL lock indication	Diagnostic	Hardware	Continuous - Enabled by software	Software defined	Software defined	Software defined
	APLL4	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	APLL5	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	APLL6	Software test of DCC functionality including error tests	Test for diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	APLL7	External Monitoring Of Clock Via Xclkout	Diagnostic	System	System defined	System defined	System defined	System defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	APLL10	Software test of PLL functionality including error tests	Test for diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	APLL11	Interleaving of FSM states	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
Reset	RST1	External monitoring of warm reset (XRSn)	Diagnostic	System	System defined	System defined	System defined	System defined
	RST2	Reset Cause Information	Fault avoidance	Hardware - Software	On demand (Software defined)	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	RST4	Glitch filtering on reset pins	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	RST5	NMIWD Shadow Registers	Fault avoidance	Hardware - Software	On demand (Software defined)	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	RST6	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	RST7	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	RST8	NMIWD Reset functionality	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset	Software defined
	RST9	Peripheral Soft Reset (SOFTPRES)	Fault avoidance	Hardware - Software	On demand (Software defined)	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS9	Software Test of ERRORSTS functionality	Diagnostic	Software	On demand (Software defined)	Software defined	System defined	System defined
	CLK6	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	CLK7	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	RST10	Software Test of Reset (Type 1)	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
System Control Module and Configuration Registers (System control)	SYS1	Multi-Bit Enable Keys for Control Registers	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS2	Lock Mechanism for Control Registers	Fault avoidance	Hardware	Continuous - Enabled by software	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS3	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SYS4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SYS5	Online Monitoring of Temperature	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SYS6	Peripheral Clock Gating (PCLKCR)	Fault avoidance	Hardware - Software	On demand (Software defined)	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS7	Peripheral Soft Reset (SOFTPRES)	Fault avoidance	Hardware - Software	On demand (Software defined)	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS8	EALLOW and MEALLOW Protection for Critical Registers	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS9	Software Test of ERRORSTS functionality	Diagnostic	Software	On demand (Software defined)	Software defined	System defined	System defined
	SYS11	Peripheral access protection - Type 1	Diagnostic	Hardware - Software	On demand (Software defined)	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
Debug Logic (JTAG)	JTAG1	Hardware Disable of JTAG Port	Fault avoidance	System	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	JTAG2	Lockout of JTAG Access using OTP	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	JTAG3	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	JTAG4	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
C28x Central Processing Unit (C28x)	CPU1	Reciprocal Comparison by Software	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU3	Software Test of CPU	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU5	Access Protection Mechanism for Memories	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	JTAG1	Hardware Disable of JTAG Port	Fault avoidance	System	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	CPU7	CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping	Diagnostic	Hardware	Continuous - Enabled at reset	zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU8	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	CPU9	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	CPU10	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU14	Stack overflow detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU15	VCU CRC Auto Coverage	Test for diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Software defined	Software defined
	CPU18	Embedded Real Time Analysis and Diagnostic (ERAD)	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU19	Inbuilt hardware redundancy in ERAD bus comparator module	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
FPU_TMU (FPU_TMU)	CPU1	Reciprocal Comparison by Software	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU5	Access Protection Mechanism for Memories	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	JTAG1	Hardware Disable of JTAG Port	Fault avoidance	System	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	CPU8	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	CPU9	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	CPU10	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU14	Stack overflow detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU15	VCU CRC Auto Coverage	Test for diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Software defined	Software defined
	CPU18	Embedded Real Time Analysis and Diagnostic (ERAD)	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU19	Inbuilt hardware redundancy in ERAD bus comparator module	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
Control Law Accelerator (MCLA)	CLA1	Reciprocal Comparison by Software	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLA2	Software Test of CLA	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLA3	CLA Handling of Illegal Operation and Illegal Results	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CLA4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLA5	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLA7	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLA8	CLA liveness check using CPU	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	CLA9	Access Protection Mechanism for Memories	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SRAM5	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM10	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
SRAM	SRAM1	SRAM ECC	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion or interrupt to CPU based on error severity	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SRAM2	SRAM Parity	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SRAM3	Software Test of SRAM	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM4	Bit Multiplexing in SRAM Memory Array	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SRAM5	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM6	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM7	Data scrubbing to detect/correct Memory errors	Fault avoidance	Software	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion or interrupt to CPU based on error severity	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SRAM8	VCU CRC check of static memory contents	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM10	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM11	Access Protection Mechanism for Memories	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SRAM12	Lock mechanism for control registers	Fault avoidance	Hardware	Continuous - Enabled by software	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	SRAM13	Software Test of ECC Logic	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM14	Software Test of Parity Logic	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM16	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM17	CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SRAM18	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	SRAM19	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	SRAM20	CLA Handling of Illegal Operation and Illegal Results	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SRAM21	Memory Power-On Self-Test (MPOST)	Diagnostic	Hardware	Bootup (enabled by default)	Software defined	Software defined	Software defined
ROM	ROM1	VCU CRC check of static memory contents	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM2	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM3	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM4	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM5	CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	ROM6	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	ROM7	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	ROM8	Power-Up Pre-Operational Security Checks	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM10	Memory Power-On Self-Test (MPOST)	Diagnostic	Hardware	Bootup (enabled by default)	Software defined	Software defined	Software defined
	ROM15	ROM Parity	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
Device Interconnect (Interconnect_Bridges)	INC1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	INC2	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	INC3	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	INC4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	INC5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	INC6	CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	INC7	CLA Handling of Illegal Operation and Illegal Results	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	INC8	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	INC9	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SYS8	EALLOW and MEALLOW Protection for Critical Registers	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
DMA	DMA2	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	DMA3	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	System defined	Software defined
	DMA4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DMA5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DMA6	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DMA7	DMA Overflow interrupt	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	DMA8	Access Protection Mechanism for Memories	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	DMA9	Disabling of unused DMA trigger sources	Fault avoidance	Software	Continuous - Enabled by software	Zero or very low overhead	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
Enhanced Peripheral Interrupt Expander (PIE) Module	PIE2	Software Test of SRAM	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PIE3	Software Test of ePIE Operation Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PIE4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PIE5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PIE7	Maintaining Interrupt Handler for unused interrupts	Diagnostic	Software	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	PIE8	Online Monitoring of Interrupts and Events	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PIE11	SRAM Parity	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion or interrupt to CPU based on error severity	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	PIE12	Software Test of Parity Logic	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
Dual Zone Code Security Module (DCSM)	DCSM1	Multi-Bit Enable Keys for Control Registers	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	DCSM2	Majority Voting and Error Detection of Link Pointer	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DCSM3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DCSM4	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DCSM5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DCSM6	CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	DCSM8	VCU CRC check of static memory contents	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DCSM9	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	DCSM11	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
CrossBar (XBAR)	XBAR1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	XBAR2	Hardware Redundancy	Diagnostic	Software	Continuous - Enabled by software	Zero or very low overhead	Software defined	Software defined
	XBAR3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	XBAR4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	XBAR5	Software Check of X-BAR Flag	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
CPU_Timer	TIM1	1oo2 Software Voting Using Secondary Free Running Counter	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	TIM2	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	TIM3	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	TIM4	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
Configurable Logic Block (CLB)	CLB1	Software test of function including error tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLB2	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLB3	Monitoring of CLB by eCAP or eQEP	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLB4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLB5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLB6	Lock Mechanism for control registers	Fault avoidance	Hardware	Continuous - Enabled by software	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	CLB7	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	CLB8	Periodic Software Read Back of SPI buffer Registers / final RAM locations where the data gets transferred through DMA	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
General Pupose I/O and Multiplexing (GPIO_Pinmux)	GPIO1	Lock Mechanism for Control Registers	Fault avoidance	Hardware	Continuous - Enabled by software	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	GPIO2	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	GPIO3	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	GPIO4	Software test of function using I/O loopback	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	GPIO5	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
Enhanced Pulse Width Modulators (ePWM)	PWM1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM2	Hardware Redundancy	Diagnostic	Software	Continuous - Enabled by software	Zero or very low overhead	Software defined	Software defined
	PWM3	Monitoring of ePWM by eCAP	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM6	Lock Mechanism for Control Registers	Fault avoidance	Hardware	Continuous - Enabled by software	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	PWM8	ePWM fault detection using XBAR	Diagnostic	Software	Continuous - Enabled by software	Zero or very low overhead	Software defined	Software defined
	PWM9	ePWM synchronization check	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM11	ePWM Application Level Safety Mechanism	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM12	Online Monitoring of Interrupts and Events	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM13	Monitoring of ePWM by ADC	Diagnostic	System	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
High Resolution Pulse Width Modulator (OTTO)	OTTO1	HRPWM Built-In Self-Check and Diagnostic Capabilities	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	OTTO2	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	OTTO3	Monitoring of ePWM by eCAP	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	OTTO4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	OTTO5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
Enhanced Capture (ECAP)	CAP1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAP2	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAP3	Monitoring of ePWM by eCAP	Test for diagnostic	Hardware	On demand (Software defined)	Software defined	Software defined	Software defined
	CAP4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAP5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAP6	eCAP Application Level Safety Mechanism	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAP7	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
Enhanced Quadrature Encoder Pulse (eQEP)	QEP1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	QEP2	eQEP Quadrature Watchdog	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	QEP3	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	QEP4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	QEP5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	QEP6	eQEP Application Level Safety Mechanism	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	QEP7	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	QEP8	QMA error detection logic	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	QEP9	eQEP Software Test of Quadrature Watchdog Functionality	Test for diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
XINT	XINT1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	XINT2	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	XINT3	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	XINT4	Hardware Redundancy	Diagnostic	Software	Continuous - Enabled by software	Zero or very low overhead	Software defined	Software defined
ADC	ADC1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ADC2	DAC to ADC Loopback Check	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ADC3	ADC Information redundancy techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	ADC4	Opens/Shorts Detection Circuit for ADC	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ADC5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ADC6	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ADC7	ADC signal quality check by varying acquisition window	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ADC8	ADC Input Signal Integrity Check	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Software defined	Software defined
	ADC9	Monitoring of ePWM by ADC	Diagnostic	System	On demand (Software defined)	Software defined	Software defined	Software defined
	ADC10	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
BufDAC	DAC1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DAC2	DAC to ADC Loopback Check	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DAC3	Lock mechanism for control registers	Fault avoidance	Hardware	Continuous - Enabled by software	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	DAC4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DAC5	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DAC6	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DAC7	DAC to Comparator Loopback Check	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
Comparator Subsystem (CMPSS)	CMPSS1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CMPSS3	Hardware Redundancy	Diagnostic	Software	Continuous - Enabled by software	Software defined	Software defined	Software defined
	CMPSS4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CMPSS5	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CMPSS6	Lock Mechanism for Control Registers	Fault avoidance	Hardware	Continuous - Enabled by software	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	CMPSS8	CMPSS Ramp generator functionality check	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
Programmable Gain Amplifier (PGA)	PGA1	PGA to ADC Loopback Test	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PGA2	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PGA3	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PGA4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PGA6	Lock mechanism for control registers	Fault avoidance	Hardware	Continuous - Enabled by software	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
Local Interconnect Network (LIN)	LIN1	Software Test of Function Using I/O Loopback	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	LIN2	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	LIN3	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	LIN4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	LIN5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	LIN6	Data Parity Error Detection	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	LIN7	SCI Overrun Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	LIN8	SCI Frame Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	LIN9	LIN Physical Bus Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	LIN10	LIN No-Response Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	LIN11	Bit Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	LIN12	LIN Checksum Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	LIN13	LIN ID Parity Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	LIN15	SCI Break Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	LIN16	Communication Access Latency Profiling Using On-Chip Timer	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	LIN17	Software Test of Function Including error tests using EPG	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
Serial Peripheral Interface (SPI)	SPI1	Software test of function using I/O loopback	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	SPI2	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SPI3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SPI4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SPI5	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SPI6	SPI Data Overrun Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SPI7	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SPI8	Software Test of Function Including error tests using EPG	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
Fast Serial Interface (FSI)	FSI1	Software test of function using I/O loopback	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI2	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI5	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI6	FSI Data Overrun/Underrun Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	FSI7	FSI Frame Overrun Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	FSI8	FSI CRC Framing Checks	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	FSI9	FSI ECC Framing Checks	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI10	FSI Frame Watchdog	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	FSI11	FSI RX Ping Watchdog	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	FSI12	FSI Tag Monitor	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI13	FSI Frame Type Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	FSI14	FSI End of Frame Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	FSI15	FSI Register Protection Mechanisms	Fault avoidance	Hardware	Continuous - Enabled by software	Zero or very low overhead	Ping Trigger to Receiver	Ping Frame Duration
	JTAG1	Hardware Disable of JTAG Port	Fault avoidance	System	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
Serial Communications Interface (SCI)	SCI1	Software test of function using I/O loopback	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SCI2	Parity in Message	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SCI3	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SCI4	SCI Overrun Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SCI5	SCI Break Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	SCI6	SCI Frame Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SCI7	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SCI8	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SCI9	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SCI10	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SCI11	Software Test of Function Including error tests using EPG	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
Inter-Integrated Circuit (I2C)	I2C1	Software test of function using I/O loopback	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	I2C2	I2C Data Acknowledge Check	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	I2C3	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	I2C4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	I2C5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	I2C6	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	I2C7	I2C Access Latency Profiling Using On-Chip Timer	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	I2C9	Software Test of Function Including error tests using EPG	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
Power Management BUS (PMBUS)	PMBUS2	I2C Data Acknowledge Check	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PMBUS3	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PMBUS4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PMBUS5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PMBUS6	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PMBUS7	PMBus Protocol CRC in Message	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PMBUS8	Clock Timeout	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
MCAN	MCAN1	Software test of function using I/O loopback	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	MCAN2	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	MCAN3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	MCAN4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	MCAN5	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	MCAN6	PWM Trip by MCAN	Diagnostic	Hardware	Continuous - Enabled by software	Software defined	Software defined	Software defined
	MCAN7	Software Test of SRAM	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	MCAN8	SRAM ECC	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	MCAN9	Bit Multiplexing in SRAM Memory Array	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	MCAN10	MCAN Stuff Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	MCAN11	MCAN Form Error Detection	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	MCAN12	MCAN Acknowledge Error Detection	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	MCAN13	Bit Error Detection	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	MCAN14	CRC in Message	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	MCAN15	Software Test of ECC Logic	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	MCAN16	Timeout on FIFO activity	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Software defined	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	MCAN17	Timestamp Consistency checks	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	MCAN18	Tx-Event Checks	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	MCAN19	Interrupt on Message RAM Access Failure	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Software defined	Typically less than 1µs to notify *(Interrupt Handling Time is System Load and Software Dependent)
	MCAN20	Software Test of Function Including Error Tests using EPG	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
AES	AES1	Decryption of encrypted data output using same KEY and IV (Initialization Vector)	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	AES2	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	AES3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	AES4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	AES5	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	AES6	Disabling of unused DMA trigger sources	Fault avoidance	Software	Continuous - Enabled by software	Zero or very low overhead	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	AES7	Software test of function including error tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	AES8	Software test of standalone GHASH operation	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
NW Embedded Flash Memory (NW Flash)	NWFLASH1	Flash ECC	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Software defined
	NWFLASH2	Flash Program Verify and Erase Verify Check	Diagnostic	Hardware	Continuous - Enabled at reset	Software defined	Software defined	Software defined
	NWFLASH3	Flash Program/Erase Protection	Fault avoidance	Hardware	Continuous - Enabled by software	Zero or very low overhead	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	NWFLASH4	Flash Wrapper Error and Status Reporting	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Software defined	Software defined
	NWFLASH5	VCU CRC check of static memory contents	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	NWFLASH6	Prevent 0 to 1 Transition Using Program Command	Fault avoidance	Hardware	Continuous - Enabled by software	Zero or very low overhead	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	NWFLASH7	On-demand Software Program Verify and Blank Check	Diagnostic	Hardware-Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table 8-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action On Detected Fault	Error Reporting Time
	NWFLASH8	Software readback of written configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	NWFLASH9	CMDWEPROT* and Program Command Data Buffer Registers Self-Clear After Command Execution	Fault avoidance	Hardware	Continuous - Enabled at reset	Zero or very low overhead	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	NWFLASH10	ECC generation and checker logic is separate in hardware	Fault avoidance	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Software defined	Software defined
	NWFLASH12	Bit Multiplexing in Flash Memory Array	Fault avoidance	Hardware	Continuous - Enabled at reset	Zero or very low overhead	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	NWFLASH13	AUTOECC Generation Override	Test for diagnostic	Hardware	On demand (Software defined)	Zero or very low overhead	Software defined	Software defined
	NWFLASH14	Software Test of Flash Prefetch, Data Cache and Wait-States	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	NWFLASH15	Software Test of ECC logic	Test for diagnostic	Hardware-Software	On demand (Software defined)	Zero or very low overhead	NMI with ERRORSTS assertion	Software defined
	NWFLASH16	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

Appendix A

Distributed Developments

A Development Interface Agreement (DIA) is intended to capture the agreement between two parties towards the management of each party's responsibilities related to the development of a functional safety system. Functional Safety-Compliant components are typically designed for many different systems and are considered to be Safety Elements out of Context (SEooC) hardware components. The system integrator is then responsible for taking the information provided in the hardware component safety manual, safety analysis report and safety report to perform system integration activities. Because there is no distribution of development activities, TI does not accept DIAs with system integrators.

"Functional Safety-Compliant" components are products that TI represents, promotes or markets as helping customers mitigate functional safety related risks in an end application and/or as compliant with an industry functional safety standard. For more information about Functional Safety-Compliant components, go to here.

A.1 How the Functional Safety Life Cycle Applies to Functional Safety-Compliant Products

TI has tailored the functional safety life cycles of ISO 26262:2018 and IEC 61508:2010 to best match the needs of a functional Safety Element out of Context (SEooC) development. The functional safety standards are written in the context of the functional safety systems, which means that some requirements only apply at the system-level. Since Functional Safety-Compliant components are hardware or software components, TI has tailored the functional safety activities to create new product development processes for hardware and for software that makes sure state-of-the-art techniques and measures are applied as appropriate. These new product development processes have been certified by third-party functional safety experts. To find these certifications, go to ti.com.

A.2 Activities Performed by Texas Instruments

The functional safety compliant Integrated Circuit (IC) products are hardware components developed as functional Safety Elements out of Context. As such, TI's functional safety activities focus on those related to management of functional safety around hardware component development. System-level architecture, design, and functional safety analysis are not within the scope of TI activities and are the responsibility of the system integrator. Some techniques for integrating the SEooC safety analysis of this hardware component into the system-level can be found in ISO 26262-11:2018.

Table A-1. Activities Performed by Texas Instruments versus Performed by the Customer

Functional Safety Life Cycle Activity	TI Execution	System Integrator Execution
Management of functional safety	Yes	Yes
Definition of end equipment and item	No	Yes
Hazard analysis and risk assessment (of end equipment/ item)	No	Yes
Creation of end equipment functional safety concept	No. Assumptions made for internal development.	Yes
Allocation of end equipment requirements to sub-systems, hardware components, and software components	No. Assumptions made for internal development.	Yes

Table A-1. Activities Performed by Texas Instruments versus Performed by the Customer (continued)

Functional Safety Life Cycle Activity	TI Execution	System Integrator Execution
Definition of hardware component safety requirements	Yes	No
Hardware component architecture and design execution	Yes	No
Hardware component functional safety analysis	Yes	No
Hardware component verification and validation (V&V)	V&V executed to support internal development.	Yes
Integration of hardware component into end equipment	No	Yes
Verification of IC performance in end equipment	No	Yes
Selection of safety mechanisms to be applied to IC	No	Yes
End equipment level verification and validation	No	Yes
End equipment level functional safety analysis	No	Yes
End equipment level functional safety assessment	No	Yes
End equipment release to production	No	Yes
Management of functional safety issues in production	Support provided as needed	Yes

A.3 Information Provided

Texas instruments has summarized what it considers the most critical functional safety work products that are available to the customer either publicly or under a nondisclosure agreement (NDA). NDAs are required to protect proprietary and sensitive information disclosed in certain functional safety documents.

Table A-2. Product Functional Safety Documentation

Deliverable Name	Contents
Functional Safety Product Preview	Overview of functional safety considerations in product development and product architecture. Delivered ahead of public product announcement.
Functional Safety Manual	User guide for the functional safety features of the product, including system-level assumptions of use.
Functional Safety Analysis Report	Results of all available functional safety analysis documented in a format that allows computation of custom metrics.
Functional Safety Report(1)	Summary of arguments and evidence of compliance to functional safety standards. References a specific component, component family, or TI process that was analyzed.
Assessment Certificate(1)	Evidence of compliance to functional safety standards. References a specific component, component family, or TI process that was analyzed. Provided by a 3rd party functional safety assessor.

(1) When an Assessment Certificate is available for a functional safety compliant product, the Functional Safety Report may not be provided. When a Functional Safety Report is provided, an Assessment Certificate may not be available. These two documents fulfill the same functional safety requirements and will be used interchangeably depending on the functional safety compliant product.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

DATE	REVISION	NOTES
December 2024	*	Initial Release

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

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