Functional Safety Information

Functional Safety Manual for TMS320F28P65x Real-Time Microcontrollers

Table of Contents

List of Figures

List of Tables

1 Introduction

The TMS320F28P65x is being offered as a Functional Safety Compliant Safety Element out of Context (SEooC) product. This implies that TMS320F28P65x 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 *Certification for Functional Safety Hardware Process*. As such, this safety manual is intended to be informative only to help explain how to use the features of TMS320F28P65x 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 their system and determining the system-level ASIL or SIL coverage achieved therein.

This document is the Functional Safety Manual for the TMS320F28P65x 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 *Certification for Functional Safety Hardware 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.

Table 1-1. Products Supported by This Safety Manual

Orderable Devices	
Dual-Core Part Numbers	**Single-Core Part Numbers**
F28P650DK9NMR	F28P650SK7NMRR
F28P650DK9NMRR	F28P650SK7PTP
F28P650DK9PTP	F28P650SK7ZEJR
F28P650DK9ZEJ	F28P650SK6PZPR
F28P650DK9ZEJR	F28P650SK6NMRR
F28P650DK7NMRR	F28P650SK6PTP
F28P650DK7PTP	F28P650SK6ZEJR
F28P650DK7ZEJR	F28P659SH6PZPRQ1
F28P650DK8NMRR	F28P659SH6PTPQ1
F28P650DK8PTP	F28P650SH7NMRR
F28P650DK8ZEJR	F28P650SH7PTP
F28P659DK8PZPQ1	F28P650SH7ZEJR
F28P659DK8PZPRQ1	F28P650SH6PZPR
F28P659DK8PTPQ1	F28P650SH6NMRR
F28P659DK8ZEJQ1	F28P650SH6PTP
F28P659DK8ZEJRQ1	F28P650SH6ZEJR
F28P650DK6PZP	
F28P650DK6PZPR	
F28P650DK6NMRR	
F28P650DK6PTP	
F28P650DK6ZEJR	
F28P659DH8PZPRQ1	
F28P650DH6PZPR	
F28P650DH6NMRR	
F28P650DH6PTP	
F28P650DH6ZEJR	

This Functional Safety Manual provides information needed by system developers to assist in the creation of a safety critical system using a supported TMS320F28P65x 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 TMS320F28P65x 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 (viz. IEC 61508:2010 and ISO 26262:2018)
- Quantitative functional safety analysis (also known as FMEDA, Failure Modes, Effects, and Diagnostics Analysis) with detail of the different parts of the component, allowing for customized application of functional safety mechanisms
- Assumptions used in the calculation of functional safety metrics

It is expected that the user of this document should have a general familiarity with the TMS320F28P65x product families. More information can be found at www.ti.com/C2000.

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, please contact your TI sales representative or www.ti.com.

Trademarks

C2000™ is a trademark of Texas Instruments.
EtherCAT® is a registered trademark of and licensed by Beckhoff Automation GmbH.
All trademarks are the property of their respective owners.

2 Hardware Component Functional Safety Capability

This section summarizes the TMS320F28P65x product safety capability. Each TMS320F28P65x 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 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 TMS320F28P65x 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
- For safety components developed according to many safety standards, it is expected that the component functional safety manual will provide a list of product safety constraints. For a simple component or more complex components developed for a single application, this is a reasonable response. However, the TMS320F28P65x 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.

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 B.

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		

4 TMS320F28P65x Component Overview

The TMS320F28P65xD and TMS320F28P65xS are powerful 32-bit floating-point microcontroller units (MCU) designed for advanced closed-loop control in automotive and industrial applications.

The TMS320F28P65x real-time control subsystem is based on TI's 32-bit C28x DSP core, which provides 200 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 key to real-time control systems. Extended instruction sets enable IEEE double-precision 64-bit floating-point math. Users may refer to *Enhancing the Computational Performance of the C2000™ Microcontroller Family* to see 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.

The lockstep dual-CPU comparator option has been added in the secondary C28x CPU along with ePIE and DMA for detection of permanent and transient faults. To allow fast context switching from existing to new firmware, hardware enhancements for Live Firmware Update (LFU) have been added to F28P65x.

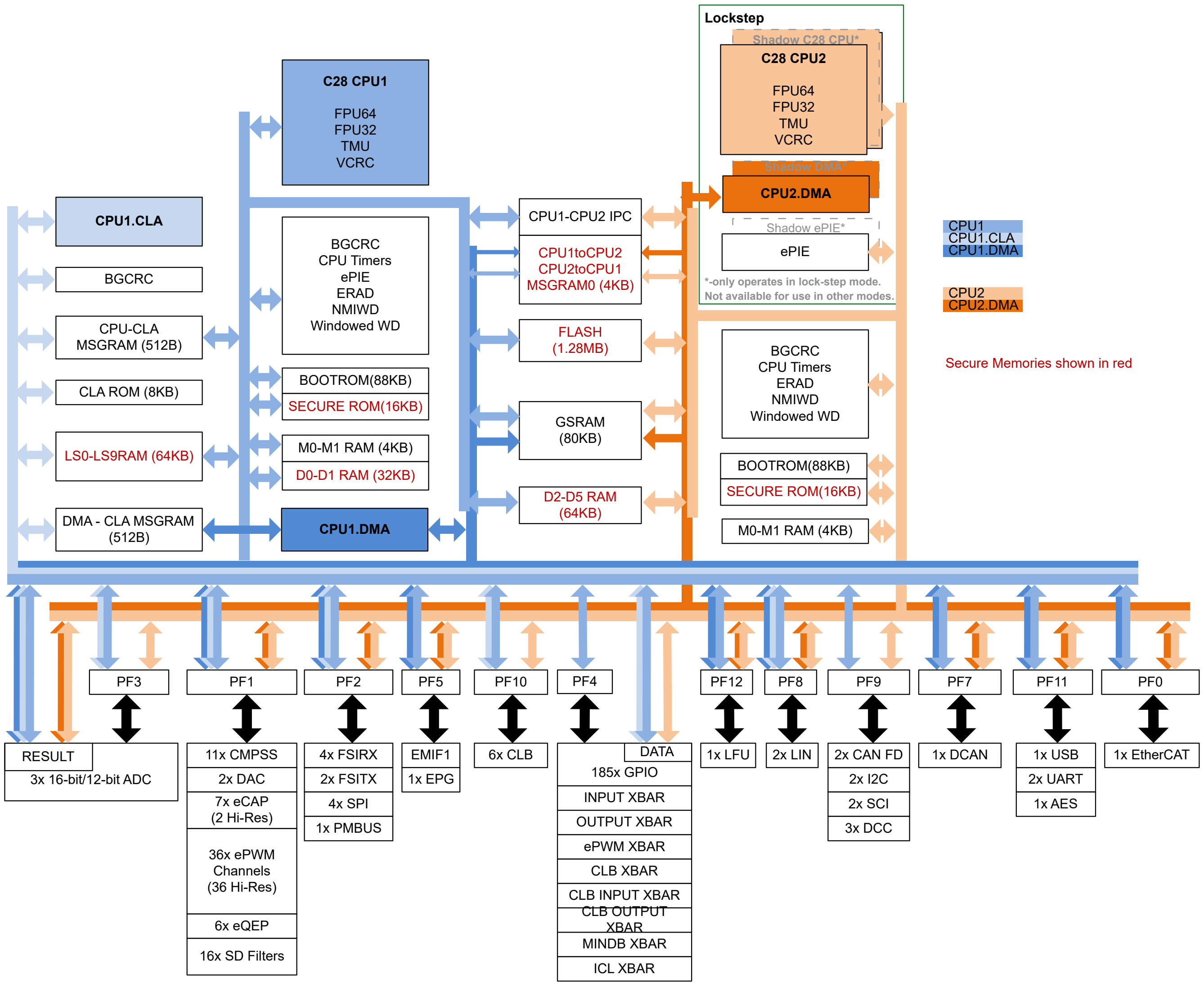


Figure 4-1. Functional Block Diagram of TMS320F28P65x Real-Time MCUs

High-performance analog blocks are tightly integrated with the processing and control units to provide optimal 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 External Memory Interface (EMIF) and 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 TMS320F28P65xD MCUs are outlined in the *TMS320F28P65x 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.

The major safety features of the TMS320F28P65x are shown in Figure 4-2.

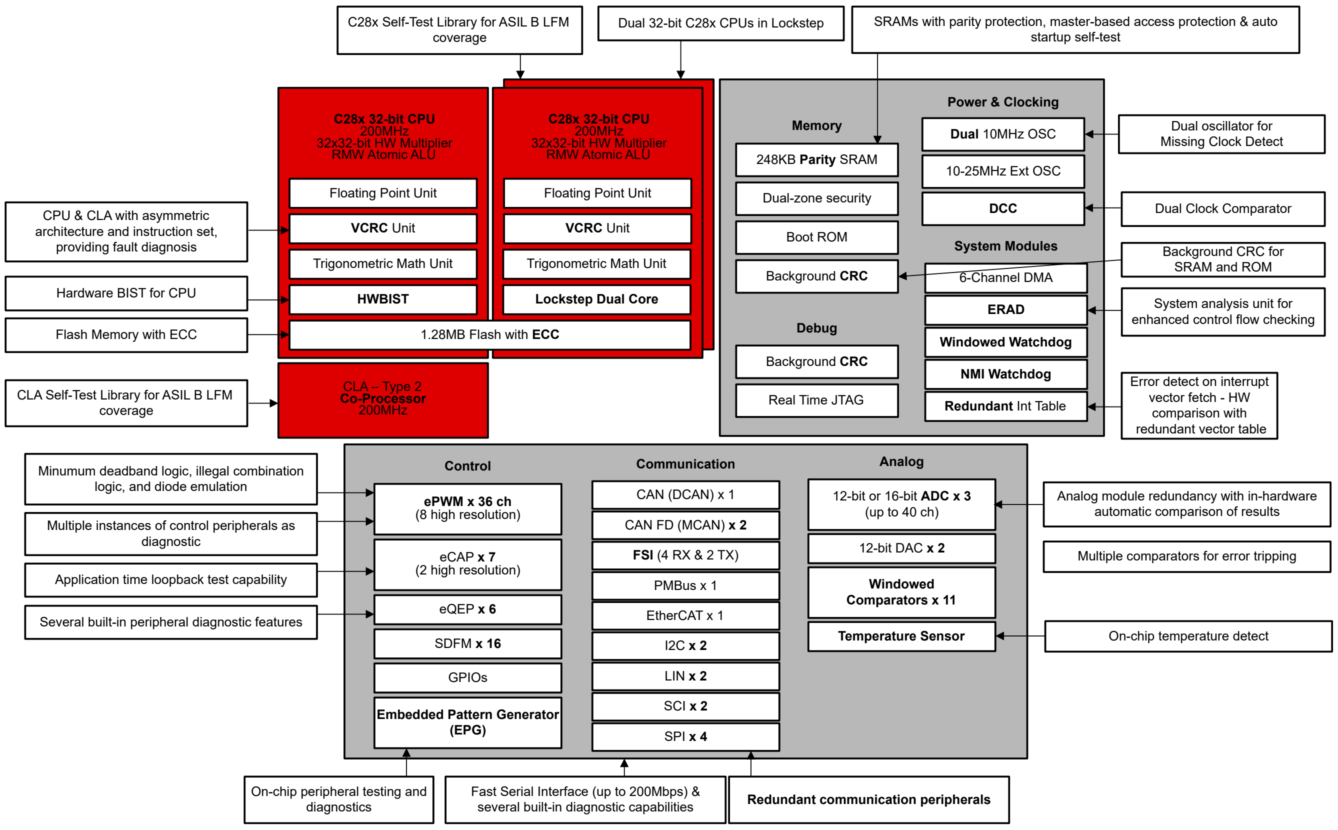

C28x Self-Test Library for ASIL B LFM coverage
Dual 32-bit C28x CPUs in Lockstep
SRAMs with parity protection, master-based access protection & auto startup self-test
C28x 32-bit CPU 200MHz 32x32-bit HW Multiplier RMW Atomic ALU
Floating Point Unit
VCRC Unit
Trigonometric Math Unit
HWBIST
Lockstep Dual Core
1.28MB Flash with ECC
CLA – Type 2 Co-Processor 200MHz
CPU & CLA with asymmetric architecture and instruction set, providing fault diagnosis
Hardware BIST for CPU
Flash Memory with ECC
CLA Self-Test Library for ASIL B LFM coverage
Memory
248KB Parity SRAM
Dual-zone security
Boot ROM
Background CRC
Debug
Background CRC
Real Time JTAG
Power & Clocking
Dual 10MHz OSC
10-25MHz Ext OSC
DCC
System Modules
6-Channel DMA
ERAD
Windowed Watchdog
NMI Watchdog
Redundant Int Table
Dual oscillator for Missing Clock Detect
Dual Clock Comparator
Background CRC for SRAM and ROM
System analysis unit for enhanced control flow checking
Error detect on interrupt vector fetch - HW comparison with redundant vector table
Control
ePWM x 36 ch (8 high resolution)
eCAP x 7 (2 high resolution)
eQEP x 6
SDFM x 16
GPIOs
Embedded Pattern Generator (EPG)
Communication
CAN (DCAN) x 1
CAN FD (MCAN) x 2
FSI (4 RX & 2 TX)
PMBus x 1
EtherCAT x 1
I2C x 2
LIN x 2
SCI x 2
SPI x 4
Analog
12-bit or 16-bit ADC x 3 (up to 40 ch)
12-bit DAC x 2
Windowed Comparators x 11
Temperature Sensor
Minumum deadband logic, illegal combination logic, and diode emulation
Multiple instances of control peripherals as diagnostic
Application time loopback test capability
Several built-in peripheral diagnostic features
Analog module redundancy with in-hardware automatic comparison of results
Multiple comparators for error tripping
On-chip temperature detect
On-chip peripheral testing and diagnostics
Fast Serial Interface (up to 200Mbps) & several built-in diagnostic capabilities
Redundant communication peripherals

Figure 4-2. TMS320F28P65x Real-Time Microcontroller With Safety Features

4.1 Targeted Applications

The TMS320F28P65x 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 & 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 & 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 it) and connected to remote controllers by means of a communication bus as shown in Figure 4-3. The communication bus is directly or indirectly connected to sensors and 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 IEC 61508 series. A system including TMS320F28P65x microcontroller as indicated by Figure 4-3 can be used in a compliant item according to IEC61508.

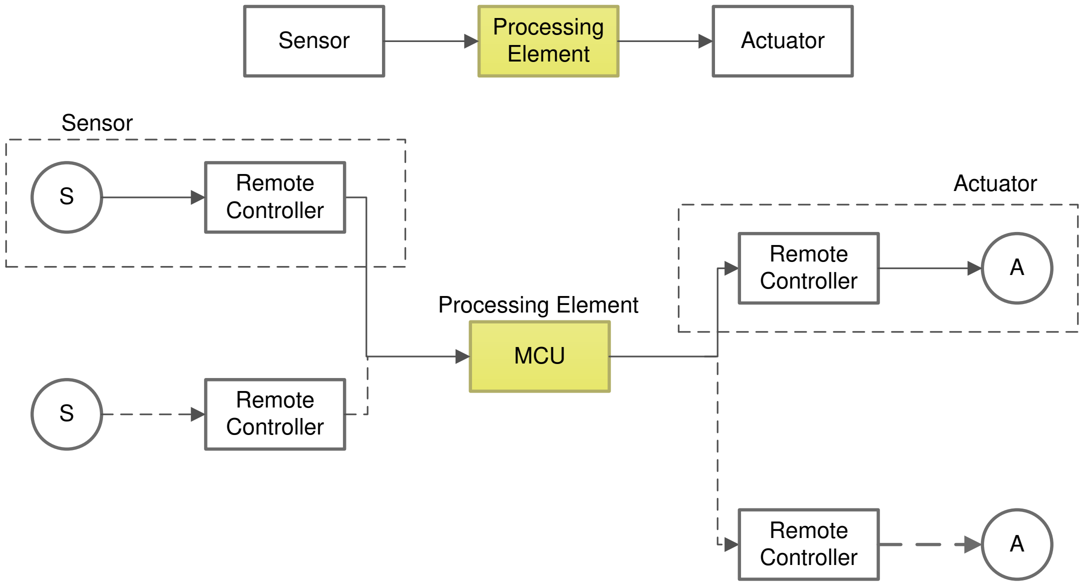


Figure 4-3. Definition of the C2000 MCU Used in a Compliant Item

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 may be used for applications other than engine management systems provided it 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.

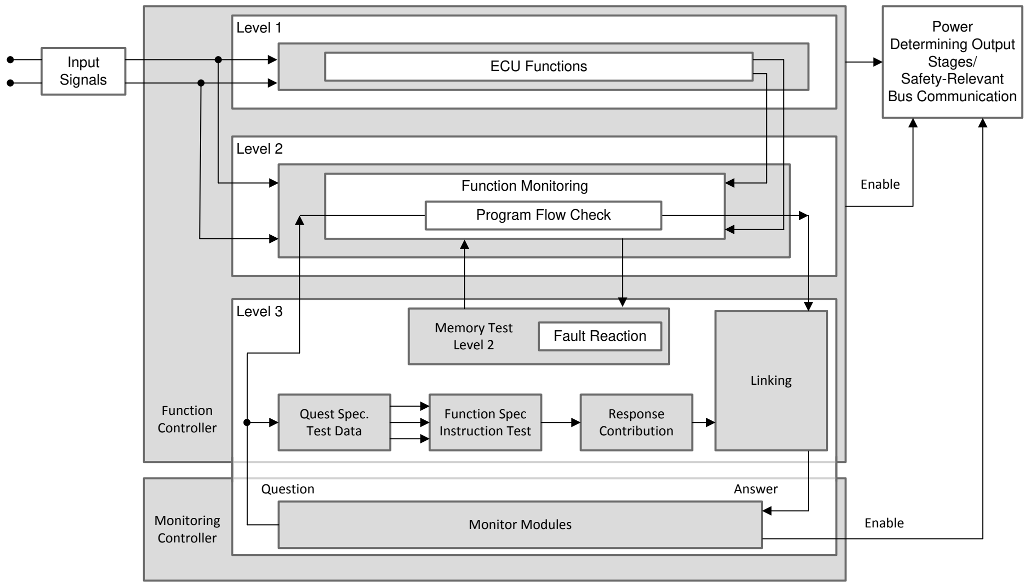


Figure 4-4. E-GAS System Overview From Standard

The TMS320F28P65xD/S MCU device family supports heterogeneous asymmetric architecture and their functional safety features lend themselves 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 selected set of parameters. Third level (Level 3) implements an additional external monitoring element, for the correct carrying out of the mission in the first level and/or monitoring in the second level. The exact functional safety implementation and the modules used for realizing Level 1 and 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 the processing units are capable of implementing either of the levels. The application can determine the partitioning based on the system requirements.

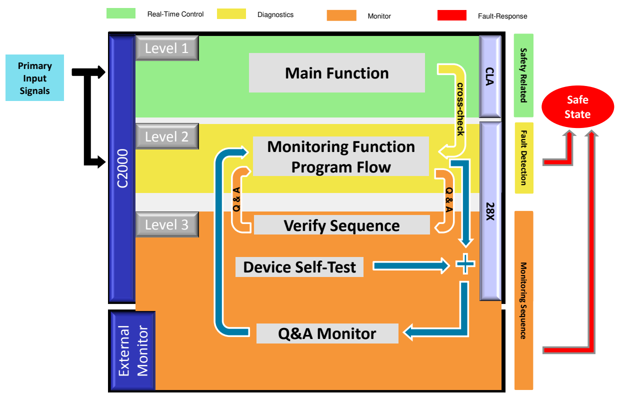


Figure 4-5. VDA E-Gas Monitoring Concept Applied to C2000 MCU

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 CPU Hardware Built-In Self-Test (HWBIST), VCRC 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 will enter into 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.

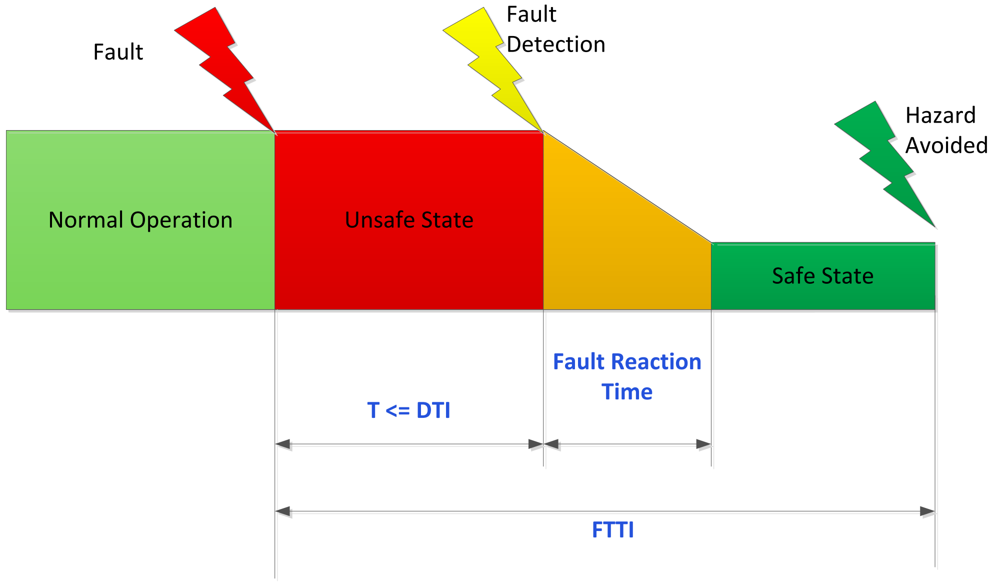


Figure 4-6. Relationship Between DTI, Fault Reaction Time and FTTI

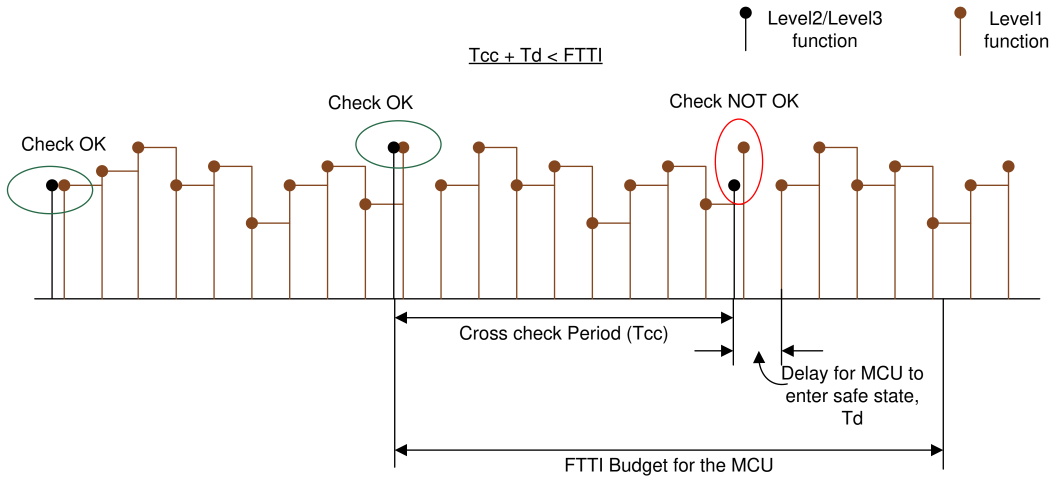


Figure 4-7. Illustration of FTTI

The frequency and extent of each of the Level 2 and Level 3 checks should be consistent with the Fault Tolerant Time Interval (FTTI). Figure 4-7 illustrates the frequency of the required checks. The checks should be such that single point faults of the microcontroller should be detected and responded to, such that the TMS320F28P65xD/S MCU enters a safe state within the FTTI budget. The microcontroller on detection of a fault enters into one of the safe states 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 to ensure 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 TMS320F28P65x MCU Safe State

Referring to Figure 4-8, the safe state of the TMS320F28P65x MCU is defined as the one in which:

- TMS320F28P65x MCU Reset is asserted.
- Power supply to TMS320F28P65x MCU is disabled using an external supervisor as a result of Level 3 check failure. In general, a power supply failure is not considered in detail in this analysis as it is assumed that the system level functionality exists to manage this condition.
- External system is informed using one of TMS320F28P65x MCUs IO pins as a result of Level 2 check failure (for example, ERRORSTS pin is asserted).
- Output of the TMS320F28P65x MCU driving the actuator is forced to inactive mode as a result of Level 2 check failure (for example, GPIO pins corresponding to the mission function are tri-stated).

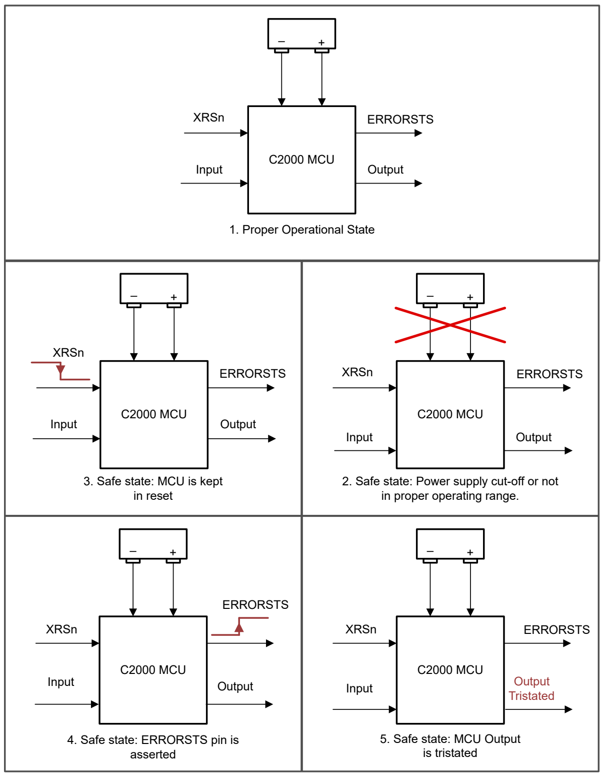

–
+
XRSn
ERRORSTS
C2000 MCU
Input
Output
1. Proper Operational State
3. Safe state: MCU is kept in reset
2. Safe state: Power supply cut-off or not in proper operating range.
4. Safe state: ERRORSTS pin is asserted
Output Tristated
5. Safe state: MCU Output is tristated

Figure 4-8. TMS320F28P65x MCU Safe State Definition

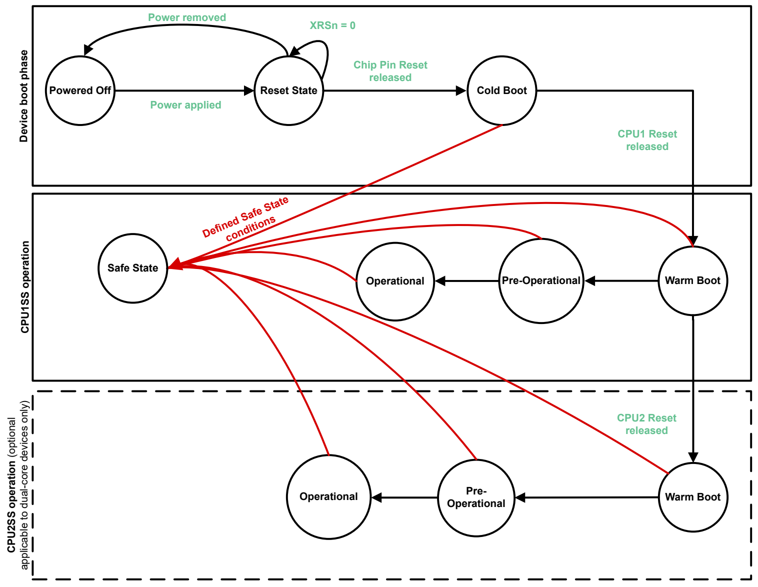


Figure 4-9. TMS320F28P65x MCU Device Operating States

4.2.4 Operating States

The TMS320F28P65x MCU products have a common architectural definition of operating states. These operating states should be observed by the system developer in their 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 CPU1SS operation phase (applicable to all the devices), and CPU2SS operation phase (applicable to TMS320F28P65xD class of devices). CPU2SS operation phase is initiated by CPU1SS operation phase. Any critical errors in either CPU1SS operation phase or CPU2SS operation phase cause the device to enter into safe state.

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

- **Powered Off**: This is the initial operating state of 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 as 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 either using 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 master 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 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/proof-test required to ensure proper device operation is performed during this phase. See Section 6.4.4.8 (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 the CPUs are shown in Figure 4-10.

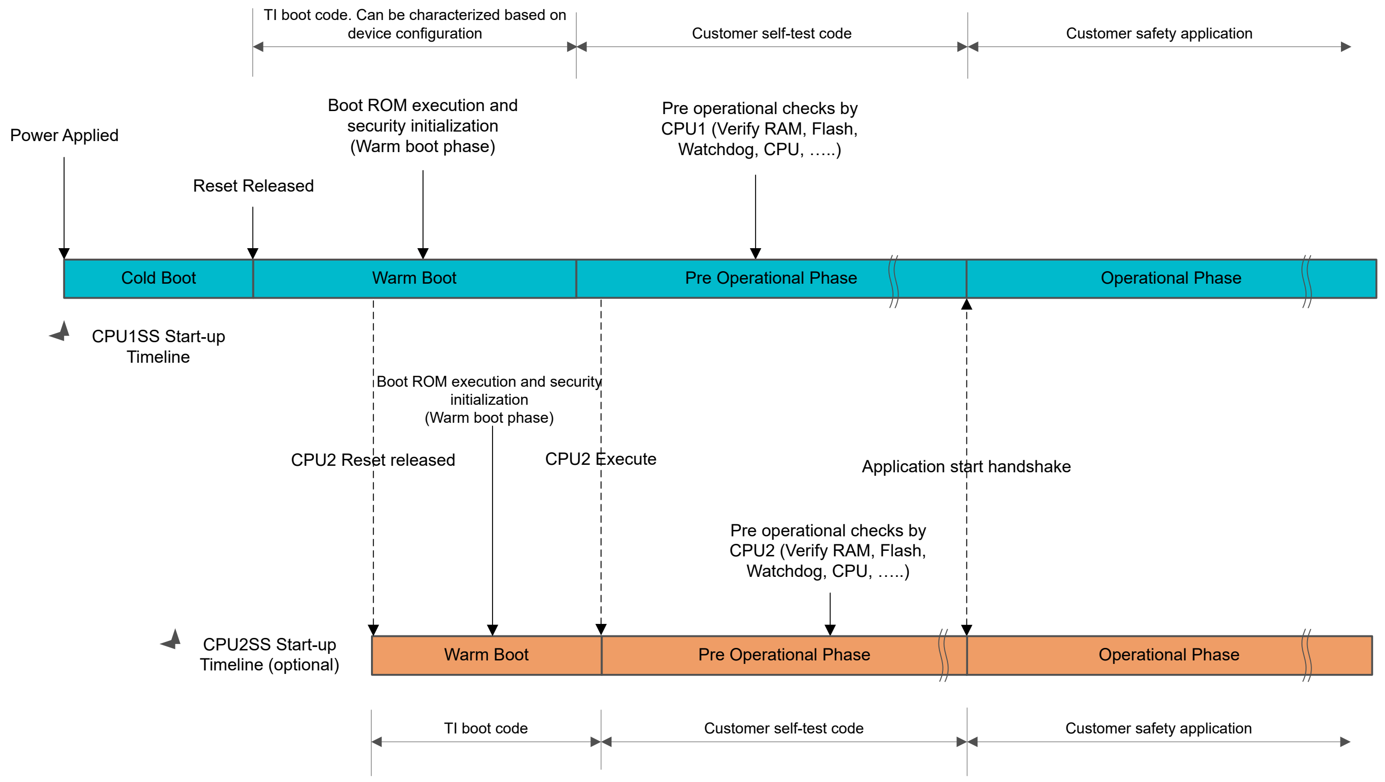

TI boot code. Can be characterized based on device configuration
Customer self-test code
Customer safety application
Power Applied
Reset Released
Boot ROM execution and security initialization (Warm boot phase)
Pre operational checks by CPU1 (Verify RAM, Flash, Watchdog, CPU, …..)
Cold Boot
Warm Boot
Pre Operational Phase
Operational Phase
CPU1SS Start-up Timeline
Boot ROM execution and security initialization (Warm boot phase)
CPU2 Reset released
CPU2 Execute
Application start handshake
Pre operational checks by CPU2 (Verify RAM, Flash, Watchdog, CPU, …..)
CPU2SS Start-up Timeline (optional)
Warm Boot
Pre Operational Phase
Operational Phase
TI boot code
Customer self-test code
Customer safety application

Figure 4-10. TMS320F28P65x MCU CPU Start-Up Timeline

4.2.5 TMS320F28P65x MCU Safety Implementation

4.2.5.1 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 TMS320F28P65x MCU
- External watchdog timer that can be used for diagnostic purposes
- Components required for taking the system to safe state as per the TMS320F28P65x MCU safe state defined in Section 4.2.3.

4.2.5.2 Example Safety Concept Implementation Options on TMS320F28P65x MCU

The CPU1 subsystem in the TMS320F28P65xD class of devices and the sole CPU in TMS320F28P65xS variants supports a pair of diverse processing units (C28x and CLA) with heterogeneous asymmetric architectures, instruction sets and software tools. Either of the processing units can be used to execute the intended function (the main real-time control function). The safety functions, which ensure that each safety goal can be met, can be implemented for diagnostic of random hardware failure 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. CPU Hardware Built-In Self-Test (HWBIST) can be used to implement latent fault coverage of the diagnostic function. Heterogeneous CPU cores minimize possibility of common mode failures while implementing this reciprocal comparison, thereby improving confidence in its Diagnostic Coverage. For common cause failures such as clock, power and reset, an external watchdog should be used.

The CPU2 subsystem in the TMS320F28P65xD class of devices supports a pair of lockstep C28x CPUs. In this case, the safety functions can be implemented via Hardware Redundancy Using Lockstep Compare Module (LCM) for diagnostic of random hardware failure, providing diagnostic coverage for the processing units. 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. Software Test of CPU, Self-test Logic for LCM, and LCM Compare Error Forcing Mode can be used to implement latent fault coverage of the diagnostic function. Again, for common cause failures such as clock, power and reset, an external watchdog should be used.

Here are some definitions relevant to the following implementation options:

- Intended Function: Control application implemented on TMS320F28P65x (PFC, DCDC, traction-inverter etc.)
- Safety Function: Achieves risk reduction and implemented for safety goals identified from HARA
  - Example: prevent over-current, over/under voltage, over temperature, forward/reverse torque etc.)
- Diagnostic Function: Ensures safety-function will operate 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 TMS320F28P65x.

4.2.5.2.1 Safety Concept Implementation: Option 1

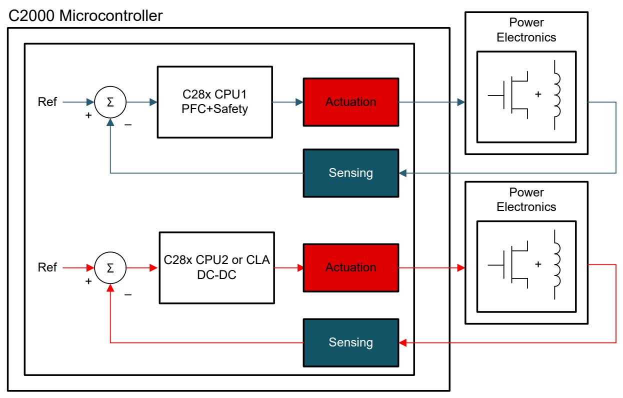


Figure 4-11. Safety Concept Implementation Option 1

- 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 CPU Hardware Built-In Self-Test (HWBIST), Software Test of CPU, Software Test of CLA, and so forth.

4.2.5.2.2 Safety Concept Implementation: Option 2

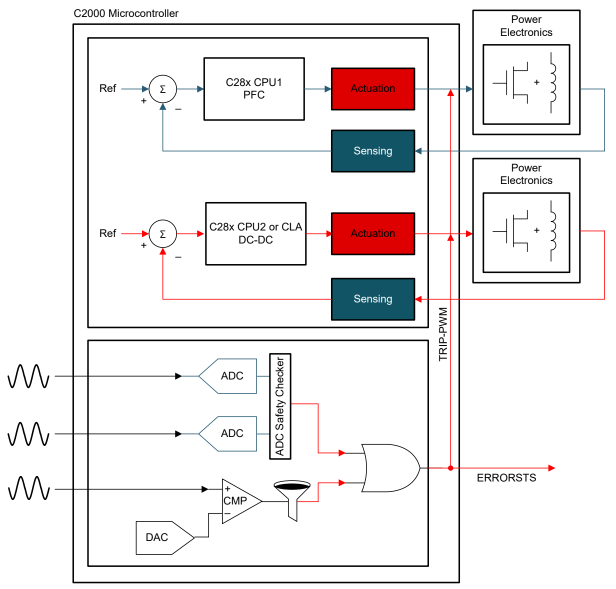


Figure 4-12. Safety Concept Implementation Option 2

- Intended Function: can be implemented on both C28x and CLA.
- Safety Function: Implement using hardware modules such as ADC-PPB, CMPSS, SDFM secondary filter, 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, SDFM secondary filter, CLB, and so forth
  - LFM can be met by Software Test of Function Including Error Tests and so forth.

4.2.5.2.3 Safety Concept Implementation: Option 3

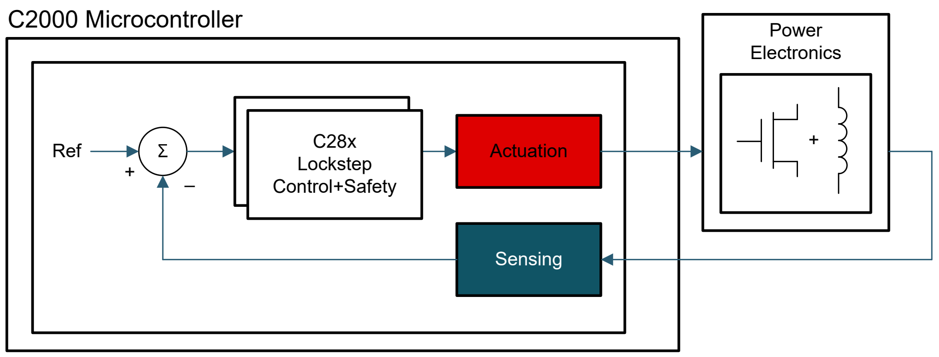


Figure 4-13. Safety Concept Implementation Option 3

- Intended Function: can be implemented on C28x (CPU2).
- Safety Function: Implement on C28x.
  - SPFM can be met by Hardware Redundancy Using Lockstep Compare Module (LCM) and so forth.
- Diagnostic Function: Implement on the other processing unit.
  - LFM can be met by Software Test of CPU, Self-test Logic for LCM, LCM Compare Error Forcing Mode, and so forth.

4.2.5.2.4 Safety Concept Implementation: Option 4

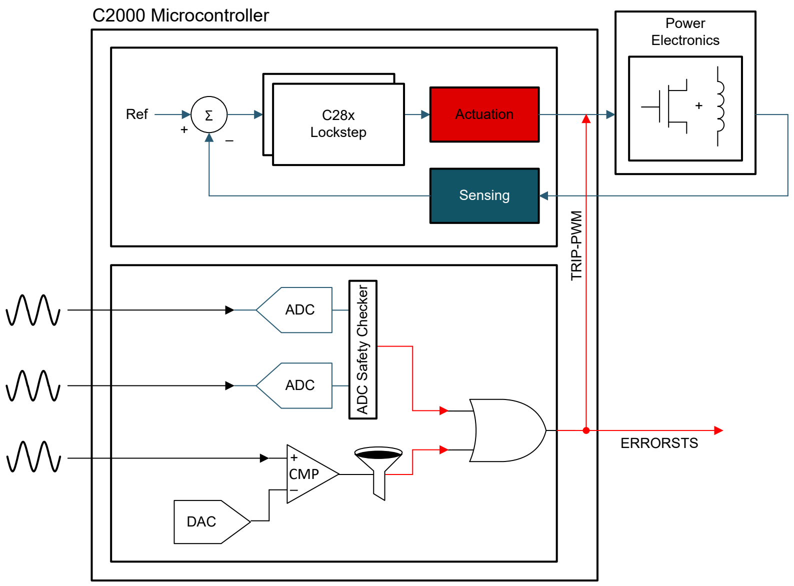


Figure 4-14. Safety Concept Implementation Option 4

- Intended Function: can be implemented on C28x (CPU2).
- Safety Function: Implement using hardware modules such as ADC-PPB, CMPSS, SDFM secondary filter, 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, SDFM secondary filter, CLB, and so forth
  - LFM can be met by Software Test of Function Including Error Tests and so forth.

4.2.5.2.5 Safety Concept Implementation: Option 5

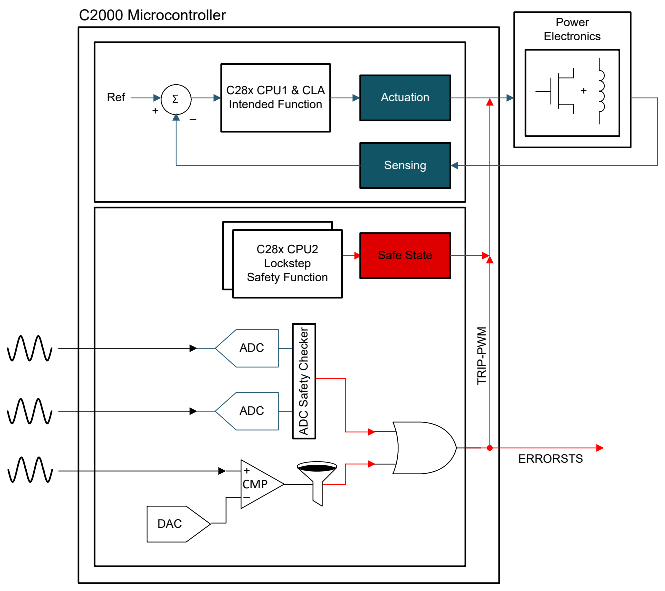


Figure 4-15. Safety Concept Implementation Option 5

- Intended Function: can be implemented on both C28x CPU1 and CLA.
- Safety Function: Implement using C28x CPU2 and other hardware modules such as ADC, CMPSS, SDFM secondary filter, CLB, and so forth.
  - SPFM of the safety goal can be met by hardware redundancy between the modules used in implementing safety function, Hardware Redundancy Using Lockstep Compare Module (LCM), Hardware Redundancy with ADC Safety Checker, Periodic Software Read Back of Static Configuration Registers and so forth.
- Diagnostic Function: Implement with C28x CPU and other hardware modules such as LCM, ADC, CMPSS, SDFM secondary filter, CLB, and so forth
  - LFM can be met by Software Test of CPU, Self-test Logic for LCM, Software Test of Function Including Error Tests and so forth.

4.2.5.2.6 Safety Concept Implementation: Option 6

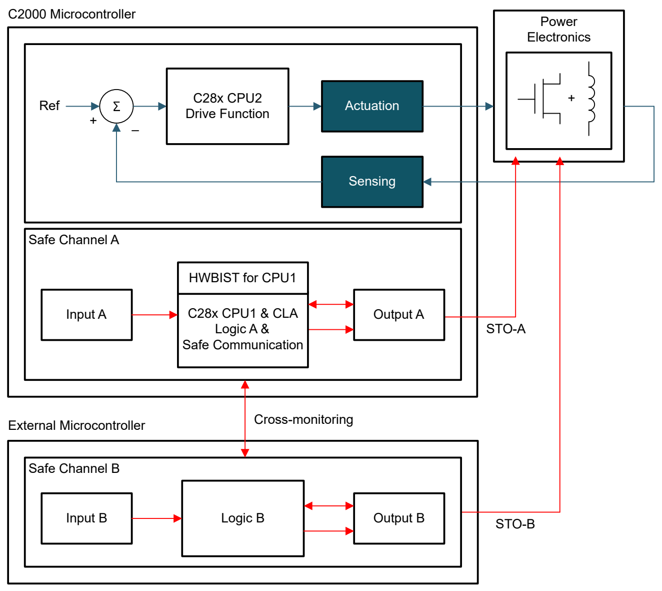


Figure 4-16. Safety Concept Implementation Option 6

Option 6 is the safety concept implementation for machinery safety category 3 and category 4 architecture as defined in ISO 13849-1:2023.

- Drive Function: can be implemented on C28x CPU.
- Safe Channel A and Communication: Implement using CPU1 and CLA. CPU Hardware Built-In Self-Test (HWBIST) for CPU1 can be run periodically for 90% diagnostic coverage to avoid the accumulation of undetected faults required for category 4 architecture.
- Diagnostic Function: Implement with Reciprocal Comparison by Software between the diverse CPUs, CPU1 and CLA.

4.2.6 TMS320F28P65x Diagnostic Libraries

The diagnostic libraries designed for the TMS320F28P65x family of devices comprise 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 TMS320F28P65x, 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.

Table 4-1. DC and SCC Targeted for F28P65x 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 suitable for being integrated 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 to for their own product level third-party certifications.

4.2.6.1 Assumptions of Use: F28P65x Self-Test Libraries

This section provides the high level details related to what a system integrator must consider during the process of defining and building their F28P65x based safety architecture. The software support for the various safety mechanisms in the F28P65x can be divided into the following categories:

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

A safe product built on the F28P65x device hierarchically deploys each of the software solutions provided by TI.

For the CPU1 subsystem, the first in the hierarchy is the HWBIST, supported by the SDL, which verifies the proper operation of the CPU by implementing the CPU2 - CPU Hardware Built-In Self-Test (HWBIST) safety mechanism. The second in the hierarchy is the SDL which provides a series of examples of safety mechanisms 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 it is important to run the HWBIST first to make sure that the CPU is functioning properly and is capable of performing the required safety operations. Then checks of elements such as the clock, internal watchdog, Flash, and RAM relevant in the execution of the CLA_STL should 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.

For the CPU2 subsystem on dual core devices, the CPU2 C28x is not accessible by the CPU1 HWBIST and instead relies on the C28x_STL to provide diagnostic coverage of permanent faults. Since other safety mechanisms make use of and depend on the C28x CPU, it is important to run the C28x_STL first in your pre-operational checks to make sure that the CPU is functioning properly and is capable of performing the required safety operations. Additionally, to detect potential causes of failure of the C28x_STL, the integrator should make sure that the internal watchdog, the LCM, and the Flash and RAM ECC/Parity logic are enabled before the C28x_STL runs.

4.2.6.2 Operational Details: F28P65x Self-Test Libraries

The diagnostic libraries are used on an F28P65x target in the implementation of safety in the host application. Therefore, it is important for a system integrator to fully comprehend all aspects of the associated system constraints imposed by the integration of the STLs to incorporate safety into the overall system.

4.2.6.2.1 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.

In order 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 state. Since the C28x_STL tests and reports on the health of the CPU itself and the system state cannot be meaningfully saved and restored, it must be integrated into the startup portion of the application. System integrator should enable the watchdog to ensure the application is protected against runaway code.

Note that 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.

4.2.6.2.2 Operational Details: CLA Self-Test Library

The CLA_STL implements the CLA2 - Software Test of CLA. Then start-up tests of the CLA_STL are also destructive in nature and should 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 what kind of tests are required to achieve the DC with the optimal distribution and quality of coverage is a fairly thorough and rigorous process. It requires running a fault injection campaign using the right combination of test vectors and confirming the coverage. These test vectors are packaged into the CLA_STL. It is important to note that 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.

4.2.6.2.3 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

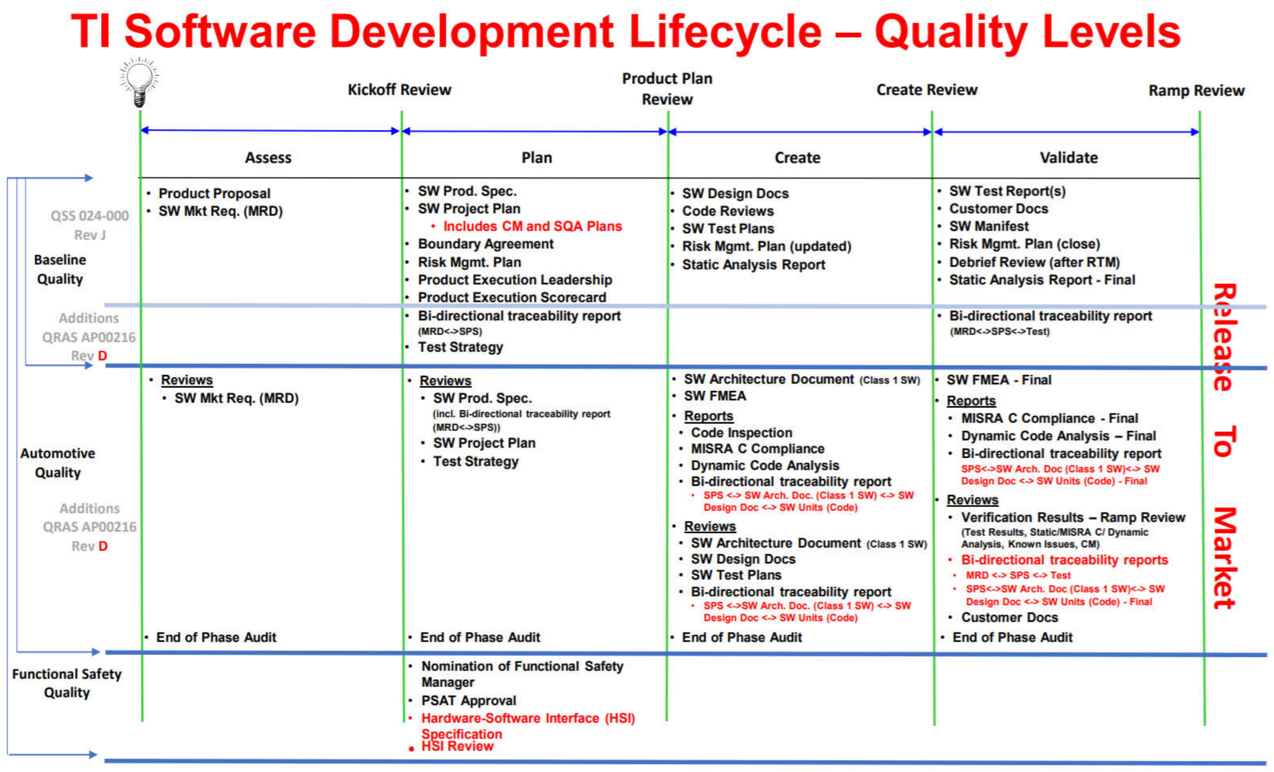
Module Name	Unique Identifier
STL_CAN_RAM	CAN4, CAN15
STL_CPU_REG	No unique identifier, added for IEC 60730
STL_CRC	NWFLASH5
STL_HWBIST	CPU2
STL_LCM	CPU22, CPU23, CPU25, PIE14, PIE15, PIE17
STL_March	SRAM3, ECAT9
STL_MCAN_RAM	MCAN7, MCAN15
STL_OSC_CT	CLK2
STL_OSC_HR	OTTO1, CLK3
STL_PIE_RAM	PIE3, PIE6
sdl_ex_dcsm_ffi	No unique identifier, demo of freedom from interference using DCSM
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, ECAT8
sdl_ex_watchdog	CLK10

4.2.6.3 C2000 Safety STL Software Development Flow

The C28x_STL and CLA_STL are developed using the TUV-SUD 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). TUV-SUD certificate for TI's SW 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), which are comprehended in the work-products, it is ensured that a TI SW/FW development achieves a systematic capability of ASIL D (ISO 26262-6) and SIL 3 (IEC 61508-3).

- Figure 4-17 depicts TI's (TUV-SUD certified) Software Development Life Cycle with respect to the various quality levels supported by the process.
- Detailed supporting procedures are documented to ensure functional safety throughout the project life cycle. Additional tools and techniques respecting the safety integrity levels of the targeted standards are applied at each development phase.
- Functional safety audits and assessments are planned and conducted as per defined procedure. Qualified personnel with adequate independence as required by the targeted standards and safety levels do these audits and assessments.



TI Software Development Lifecycle – Quality Levels

	Assess	Plan	Create	Validate
QSS 024-000 Rev J Baseline Quality	• Product Proposal • SW Mkt Req. (MRD)	• SW Prod. Spec. • SW Project Plan • Includes CM and SQA Plans • Boundary Agreement • Risk Mgmt. Plan • Product Execution Leadership • Product Execution Scorecard	• SW Design Docs • Code Reviews • SW Test Plans • Risk Mgmt. Plan (updated) • Static Analysis Report	• SW Test Report(s) • Customer Docs • SW Manifest • Risk Mgmt. Plan (close) • Debrief Review (after RTM) • Static Analysis Report - Final
Additions QRAS AP00216 Rev D		• Bi-directional traceability report (MRD<->SPS) • Test Strategy		• Bi-directional traceability report (MRD<->SPS<->Test)
Automotive Quality Additions QRAS AP00216 Rev D	• <u>Reviews</u> • SW Mkt Req. (MRD) • End of Phase Audit	• <u>Reviews</u> • SW Prod. Spec. (incl. Bi-directional traceability report (MRD<->SPS)) • SW Project Plan • Test Strategy • End of Phase Audit	• SW Architecture Document (Class 1 SW) • SW FMEA • <u>Reports</u> • Code Inspection • MISRA C Compliance • Dynamic Code Analysis • Bi-directional traceability report • SPS <-> SW Arch. Doc. (Class 1 SW) <-> SW Design Doc <-> SW Units (Code) • <u>Reviews</u> • SW Architecture Document (Class 1 SW) • SW Design Docs • SW Test Plans • Bi-directional traceability report • SPS <->SW Arch. Doc. (Class 1 SW) <-> SW Design Doc <-> SW Units (Code) • End of Phase Audit	• SW FMEA - Final • <u>Reports</u> • MISRA C Compliance - Final • Dynamic Code Analysis – Final • Bi-directional traceability report SPS<->SW Arch. Doc (Class 1 SW)<-> SW Design Doc <-> SW Units (Code) - Final • <u>Reviews</u> • Verification Results – Ramp Review (Test Results, Static/MISRA C/ Dynamic Analysis, Known Issues, CM) • Bi-directional traceability reports • MRD <-> SPS <-> Test • SPS<->SW Arch. Doc (Class 1 SW)<-> SW Design Doc <-> SW Units (Code) - Final • Customer Docs • End of Phase Audit
Functional Safety Quality		• Nomination of Functional Safety Manager • PSAT Approval • Hardware-Software Interface (HSI) Specification • HSI Review		

Figure 4-17. TI Software Development Life Cycle - Quality Level

4.3 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 TMS320F28P65x 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 may 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 datasheet.
- **[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 Multi-Point 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(s) 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 (i.e. device startup) or when device faults have been detected that would compromise the decision making capability of the device (i.e. 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 (i.e. RAM, flash, ADCs, etc.). Examples include, but are not limited to, reciprocal comparison by software, watchdogs, etc.
- **[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, it 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 it into the system level safety analysis, the diagnostics shall be applied as needed with respect to 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 with respect to the safety concept and will implement diagnostics on those components as needed with respect to that safety concept.
- **[SA_16]** The safety function is considered to operate in high/continuous demand mode of operation (per IEC 61508). (Note this 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 F28P65x 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 it after this device has successfully completed its startup 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 may be unresolvable at the system level.

5 Description of Safety Elements

This section contains a brief description of the elements on the TMS320F28P65x MCU device family, organized based on the classification of parts of generic hardware of a system (8) as shown in Figure 5-1. For a full functional description of any of these modules, see 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 of diagnostics. Primary diagnostics of type “Software” and “Hardware/Software” involves execution of the software on the processing units viz. CPU and CLA and also use many of the MCU parts like Interconnect, Memory (Flash, SRAM and ROM) and TMS320F28P65x MCU infrastructure components (Clock, Power, Reset and JTAG). To keep the integrity of the implemented primary diagnostics and the associated diagnostic coverage values, measures to protect execution of primary diagnostics on respective processing units needs to be implemented. Appropriate combination of test of diagnostics is recommended to be implemented for parts of the MCU contributing to the successful operation of the processing units. For diagnostics for these parts, see the respective sections in this safety manual. In this case, separate test of diagnostic measures exist for a primary diagnostic measure, they are mentioned along with the respective hardware part.

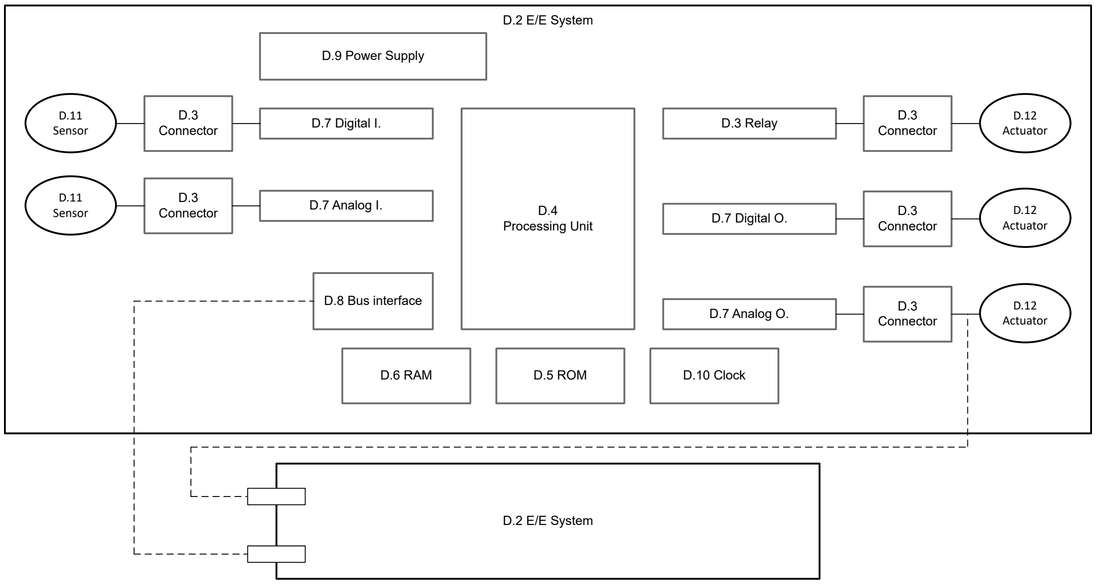


Figure 5-1. Generic Hardware of a System

5.1 C2000 MCU Infrastructure Components

5.1.1 Power Supply

The TMS320F28P65xD/S 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.

- PWR1 - External Voltage Supervisor
- PWR2 - External Watchdog (using GPIO or a serial interface)
- PWR4 - Brownout Reset (BOR)

- 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, it is recommended to implement one voltage supervisor per ganged rail.
- Common mode failure analysis of the external voltage supervisor along with TMS320F28P65xD/S MCU is useful to determine dependencies in the voltage generation and supervision circuitry.
- Customer can consider using TI's TPS6538x power supply and safety companion device for voltage supervision at the system level.

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):

- CLK1 - Missing Clock Detect (MCD)
- CLK5 - External Monitoring of Clock via XCLKOUT
- CLK17 - Dual Clock Comparator (DCC)
- CLK6 - Internal Watchdog (WD)
- CLK7 - External Watchdog
- CLK2 - Clock Integrity Check Using CPU Timer
- CLK3 - Clock Integrity Check Using HRPWM
- 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)

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

- 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 may have EMI implications. The selected clock needs to be scaled suitably before sending out through IO.

5.1.3 APLL

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
- APLL5 - External Watchdog
- APLL7 - External Clock Monitoring 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.

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)
- RST10 - Software Test of Reset – Type 1
- SYS9 - Software Test of ERRORSTS Functionality
- CLK6 - Internal Watchdog
- CLK7 - External Watchdog

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

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

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.

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

- SYS1 - Multibit 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

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

5.1.6 JTAG Debug, Trace, Calibration, and Test Access

The TMS320F28P65xD/S 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 ICEPICK to support usage without preamble scan sequences for easiest manufacturing board test.

JTAG is classified as not safety-related and must not be used during safety-related operation.

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-, 192-, 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 it from plain text to an unintelligible form called cipher text. Decrypting cipher text converts previously encrypted data to the original plain text form.

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
- 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 Unit (FPU), CRC Unit (VCRC) 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/data buses. Its unique architecture makes it amenable to integrate safety features external to CPU but on chip, to provide improved diagnostic coverage.

Note that the two CPU subsystems on this device have different associated hardware diagnostic features. The CPU1SS has the HWBIST and CLA co-processor with which Reciprocal Comparison can be implemented. The CPU2SS contains dual C28x CPUs in lockstep and is supported by the C28x_STL which perform a a software test of the CPU. For more information, see the device data sheet.

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

- CPU1 - Reciprocal Comparison by Software
- CPU2 - CPU Hardware Built-In Self-Test (HWBIST)
- CPU3 - Software Test of CPU (C28x_STL)
- CPU21 - Hardware Redundancy Using Lockstep Compare Module (LCM)
- 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)
- CPU19 - Inbuilt Hardware Redundancy in ERAD Bus Comparator Module
- CPU24 - LCM MMR Parity

The following tests can be applied as test-for-diagnostics on this module:

- CPU11 - CPU Hardware Built-In Self-Test (HWBIST) Auto Coverage
- CPU12 - CPU Hardware Built-In Self-Test (HWBIST) Fault Injection Capability
- CPU13 - CPU Hardware Built-In Self-Test (HWBIST) Timeout Feature
- CPU15 - VCRC Auto Coverage
- CPU22 - Self-test Logic for LCM
- CPU23 - LCM Compare Error Forcing Mode
- CPU25 - Test of LCM MMR Parity
- CPU26 - Lockstep Self-test Mux Select Logic Fault Detection
- CPU27 - Redundancy in LCM Comparator

Note

Measures to Mitigate Common Cause Failure in 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. System Integrator should perform a detailed analysis based on the inputs from 26262-11:2018, Section 4.7 and IEC61508 ed2 PART 2 Annex E (BetaIC method).

5.2.2 Control Law Accelerator (CLA)

The Control Law Accelerator (CLA) is an independent, fully-programmable, 32-bit floating-point math accelerator with independent ISA and independent compiler and it helps concurrent control-loop execution. The low interrupt-latency of the CLA allows it 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.

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
- CLA11 - Disabling of Unused CLA Task Trigger Sources

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. The banks of the Flash memory are mappable to either CPUSS . 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.

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 - VCRC 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 Read Back of Written Configuration
- NWFLASH9 - CMDWEPROT* and Program Command Data Buffer Registers Self-Clear After Command Execution
- NWFLASH10 - ECC Generation and Checker Logic is Separate in Hardware
- NWFLASH12 - Bit Multiplexing in Flash Memory Array
- NWFLASH14 - Software Test of Flash Prefetch, Data Cache and Wait-States
- NWFLASH16 - Information Redundancy Techniques
- CPU7 - CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping
- CLK6 - Internal Watchdog (WD)

The following tests can be applied as a test-for-diagnostic on this module:

- NWFLASH13 - Auto ECC Generation Override
- NWFLASH15 - Software Test of ECC Logic
- CPU15 - VCRC Auto Coverage

5.3.2 Embedded SRAM

The TMS320F28P65xD/S MCU device family has the following types of SRAMs with different characteristics.

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

All these RAMs are highly configurable to achieve control for write access and fetch access from different masters. 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 its own controller which implements access protection, security related features and ECC/Parity features for that RAM.

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 - VCRC 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)
- SRAM24 - Background CRC

The following tests can be applied as a test-for-diagnostic on this module:

- SRAM13 - Software Test of ECC Logic
- SRAM14 - Software Test of Parity Logic
- CPU15 - VCRC Auto Coverage
- SRAM25 - Watchdog for Background CRC

5.3.3 Embedded ROM

The TMS320F28P65xD/S MCU device family has the following types of ROMs for each CPU subsystem:

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

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

- ROM1 - VCRC 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)
- ROM13 - Background CRC
- ROM15 - ROM Parity

The following tests can be applied as a test-for-diagnostic on this module:

- CPU15 - VCRC Auto Coverage
- ROM14 - Watchdog for Background CRC

5.4 On-Chip Communication Including Bus-Arbitration

5.4.1 Device Interconnect

The device interconnects links the multiples masters and slaves within the device. The device interconnect logic comprises of static master selection muxes, dynamic arbiters and protocol convertors required for various bus masters (CPU, CLA, DMA) to transact with the peripherals and memories.

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

5.4.2 Direct Memory Access (DMA)

The direct memory access (DMA) module provides a hardware method of transferring data between peripherals and/or 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 it is transferred as well as *ping-pong* data between buffers. These features are useful for structuring data into blocks for optimal CPU processing.

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

- DMA11 - Hardware Redundancy Using Lockstep Compare Module (LCM)
- DMA2 - Information Redundancy Techniques
- DMA3 - Transmission Redundancy
- DMA4 - Software Read Back of Written Configuration
- DMA5 - Periodic Software Read Back of Static Configuration Registers
- DMA6 - Software Test of Function Including Error Tests
- DMA8 - Access Protection Mechanism for Memories
- DMA7 - DMA Overflow Interrupt
- DMA9 - Disabling of Unused DMA Trigger Sources
- DMA14 - LCM MMR Parity

The following tests can be applied as test-for-diagnostics on this module:

- DMA12 - Self-test Logic for LCM
- DMA13 - LCM Compare Error Forcing Mode
- DMA15 - Test of LCM MMR Parity
- DMA16 - Lockstep Self-test Mux Select Logic Fault Detection
- DMA17 - Redundancy in LCM Comparator

5.4.3 Inter Processor Communication (IPC)

The Inter-Processor Communications (IPC) module allows communication between the two CPU subsystems. The module includes message RAMs, IPC flags and interrupts, command registers, flash pump semaphore, clock configuration semaphore and a free running counter that are used to provide reliable communication and synchronization between the two CPUs.

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

- IPC1 - Information Redundancy Techniques Including End-to-End Safing
- IPC2 - Transmission Redundancy
- IPC3 - Software Test of Function Including Error Tests
- IPC4 - Event Timestamping Using IPC Counter
- IPC5 - Periodic Software Read Back of Static Configuration Registers
- IPC6 - Software Read Back of Written Configuration
- IPC7 - IPC 64-Bit Counter Value Plausibility Check

5.4.4 Enhanced Peripheral Interrupt Expander (ePIE) Module

The enhanced Peripheral Interrupt Expander (ePIE) module is used to interface peripheral interrupts to the C28x CPU. It 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.

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

- PIE1 - PIE Double SRAM Hardware Comparison
- PIE13 - Hardware Redundancy Using Lockstep Compare Module (LCM)
- 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
- PIE16 - LCM MMR Parity

The following tests can be applied as a test-for-diagnostic on this module:

- PIE6 - PIE Double SRAM Comparison Check
- PIE14 - Self-test Logic for LCM
- PIE15 - LCM Compare Error Forcing Mode
- PIE17 - Test of LCM MMR Parity
- PIE18 - Lockstep Self-test Mux Select Logic Fault Detection
- PIE19 - Redundancy in LCM Comparator

5.4.5 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 features a dual zone CSM for code protection.

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

- DCSM1 - Multibit Enable Keys for Control Registers
- DCSM2 - Majority Voting and Error Detection of Link Pointer
- DCSM5 - Software Read Back of Written Configuration
- DCSM3 - Periodic Software Read Back of Static Configuration Registers
- DCSM4 - Software Test of Function Including Error Tests
- DCSM6 - CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping
- DCSM8 - VCRC Check of Static Memory Contents
- DCSM9 - External Watchdog
- DCSM11 - Hardware Redundancy

5.4.6 CrossBar (X-BAR)

The crossbars (X-BAR) provide flexibility to connect device inputs, outputs, and internal resources in a variety of configurations. This device contains a several X-BARs:

- Input X-BAR
- CLB Input X-Bar
- Output X-BAR
- CLB Output X-BAR
- CLB X-BAR
- ePWM X-BAR

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

5.4.7 Timer

Each CPU subsystem is provided with three 32-bit CPU-Timers (TIMER0/1/2). 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 operation of safety critical functions. The capabilities of the module enable it to be used for clock monitoring as well.

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

5.4.8 Configurable Logic Block (CLB)

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 would otherwise be 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 SysConfig-based CLB tool. More information on the CLB tool, available examples, application reports and users guide are available in C2000Ware.

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

- CLB1 - Software Test of CLB 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

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.

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.

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
- PWM15 - Online MINMAX Monitoring of TRIP Events
- PWM16 - Fault Avoidance Using Minimum Dead Band
- PWM17 - Fault Avoidance Using Illegal Combo Logic
- PWM18 - Diode Emulation Mode Monitoring

5.5.3 High Resolution PWM (HRPWM)

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 150 ps.

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 via 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.

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
- CAP3 - Monitoring of ePWM by eCAP
- 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

Note

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

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 HW 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.

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

- HRCAP1 - Software Test of Function Including Error Tests
- HRCAP2 - Hardware Redundancy
- HRCAP3 - Monitoring of HRPWM by HRCAP
- HRCAP4 - Periodic Software Read Back of Static Configuration Registers
- HRCAP5 - Software Read Back of Written Configuration
- HRCAP7 - HRCAP Calibration Logic Test Feature

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):

- QEP1 - Software Test of Function Including Error Tests
- QEP2 - eQEP Quadrature Watchdog
- QEP3 - Information Redundancy Techniques
- QEP5 - Software Read Back of Written Configuration
- QEP4 - Periodic Software Read Back of Static Configuration Registers
- QEP6 - eQEP Application Level Safety Mechanisms
- QEP7 - Hardware Redundancy
- QEP8 - QMA error detection logic

The following tests can be applied as a test-for-diagnostic on this module:

- QEP9 - eQEP Software Test of Quadrature Watchdog Functionality

Note

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

5.5.7 Sigma Delta Filter Module (SDFM)

Sigma Delta Filter Module (SDFM) is a four-channel digital filter designed specifically for current measurement and resolver position decoding in motor control applications. Each channel can receive an independent delta-sigma (ΔΣ) modulator bit stream. The bit streams are processed by four individually-programmable digital decimation filters. The filter set includes a fast comparator for immediate digital threshold comparisons for over-current and under-current monitoring.

- SDFM1 - Comparator Filter for Online Monitoring
- SDFM2 - Information Redundancy Techniques
- SDFM3 - SD Modulator Clock Fail Detection Mechanism
- SDFM4 - Periodic Software Read Back of Static Configuration Registers
- SDFM5 - Software Read Back of Written Configuration
- SDFM6 - Software Test of Function Including Error Tests
- SDFM7 - Hardware Redundancy

5.5.8 External Interrupt (XINT)

Interrupts from external sources can be provided to the device using GPIO pins with 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.

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).

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

- ADC16 - Hardware Redundancy with ADC Safety Checker
- 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
- ADC11 - Disabling Unused Sources of SOC Inputs to ADC

The following tests can be applied as a test-for-diagnostic on this module:

- ADC15 - Hardware Redundancy of ADC Safety Checker

Note

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

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 pull-down 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.

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
- DAC7 - DAC to Comparator Loopback Check
- DAC6 - Hardware Redundancy

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 detection of signal exception conditions including High/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.

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
- CMPSS4 - Software Read Back of Written Configuration
- CMPSS5 - Periodic Software Read Back of Static Configuration Registers
- CMPSS6 - Lock Mechanism for Control Registers
- CMPSS7 - VDAC Conversion by ADC
- CMPSS8 - CMPSS Ramp Generator Functionality Check
- CMPSS3 - Hardware Redundancy

5.7 Data Transmission

5.7.1 Controller Area Network (DCAN)

The Controller Area Network (DCAN) interface provides medium throughput networking with event based triggering, compliant to the CAN protocol. The DCAN modules requires 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):

- CAN1 - Software Test of Function Using I/O Loopback
- CAN2 - Information Redundancy Techniques Including End-to-End Safing
- CAN3 - SRAM Parity
- CAN4 - Software Test of SRAM
- CAN5 - Bit Multiplexing in SRAM Memory Array
- CAN7 - Periodic Software Read Back of Static Configuration Registers
- CAN8 - Software Read Back of Written Configuration
- CAN9 - Transmission Redundancy
- CAN10 - DCAN Stuff Error Detection
- CAN11 - DCAN Form Error Detection
- CAN12 - DCAN Acknowledge Error Detection
- CAN13 - Bit Error Detection
- CAN14 - CRC in Message
- CAN16 - Hardware Redundancy
- CAN17 - Software Test of Function Including Error Tests Using EPG

The following tests can be applied as a test-for-diagnostic on this module:

- CAN15 - Software Test of Parity Logic

5.7.2 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 requires 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):

- MCAN1 - Software Test of Function Using I/O Loopback
- MCAN2 - Information Redundancy Techniques Including End-to-End Safing
- MCAN8 - SRAM ECC
- MCAN7 - Software Test of SRAM
- MCAN9 - Bit Multiplexing in SRAM Memory Array
- MCAN3 - Periodic Software Read Back of Static Configuration Registers
- MCAN4 - Software Read Back of Written Configuration
- MCAN5 - Transmission Redundancy
- MCAN6 - PWM Trip by MCAN
- 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 a test-for-diagnostic on this module:

- MCAN15 - Software Test of ECC Logic

5.7.3 Ethernet for Control Automation Technology (EtherCAT)

Ethernet for Control Automation Technology (EtherCAT®) is an Ethernet-based fieldbus system, invented by Beckhoff Automation™ and is standardized in IEC 61158. All the server nodes connected to the bus interpret, process, and modify the data addressed to them "on the fly", without having to buffer the frame inside the node. This real-time behavior, frame processing and forwarding requirements are implemented by the EtherCAT server controller hardware. EtherCAT does not require software interaction for data transmission inside the servers. EtherCAT only defines the MAC layer while the higher layer protocols and stack are implemented in software on the microcontrollers connected to the ESC. See *TMS320F28P65x Real-Time Microcontrollers* for additional details on ECAT. The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- ECAT1 - Software Test of Function Using I/O Loopback
- ECAT2 - Information Redundancy Techniques Including End-to-End Safing
- ECAT3 - Periodic Software Read Back of Static Configuration Registers
- ECAT4 - Software Read Back of Written Configuration
- ECAT5 - Transmission Redundancy
- ECAT6 - SRAM Parity
- ECAT9 - Software Test of SRAM
- ECAT10 - EtherCAT MDIO Command Error Indication
- ECAT11 - EtherCAT Sync-Manager
- ECAT12 - EtherCAT Working Counter Error Indication
- ECAT13 - EtherCAT Frame Error Indication
- ECAT14 - EtherCAT Physical Layer Error Indication
- ECAT15 - CRC in Message
- ECAT16 - PDI Timeout Error Indication
- ECAT17 - EtherCAT EEPROM CRC Error Indication
- ECAT18 - EtherCAT EEPROM Not Done Error Indication
- ECAT19 - EtherCAT Data Link Error Indication
- ECAT20 - EtherCAT Phy Link Error Indication
- ECAT21 - Sync, GPO Monitoring Using External Monitor
- ECAT22 - EtherCAT Enhanced Link Detection With LED
- ECAT23 - HW Redundancy of GPIO, FMMU, Sync Manager and SYNC OUT
- ECAT24 - Bit Multiplexing in SRAM Memory Array

The following tests can be applied as a test-for-diagnostic on this module:

- ECAT7 - Redundant Parity Engine
- ECAT8 - Software Test of Parity Logic

5.7.4 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.

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

5.7.5 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 may be necessary.

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

5.7.6 Universal Asynchronous Receiver/Transmitter (UART)

The UART module performs the functions of parallel-to-serial and serial-to-parallel conversions. The UART module is similar in functionality to a 16C550 UART, but is not register-compatible.

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

- UART1 - Software Test of Function Using I/O Loopback
- UART2 - Information Redundancy Techniques Including End-to-End Safing
- UART3 - Periodic Software Read Back of Static Configuration Registers
- UART4 - Software Read Back of Written Configuration
- UART5 - Transmission Redundancy
- UART6 - Parity in Message
- UART7 - Overrun Error Detection
- UART8 - Break Error Detection
- UART9 - Frame Error Detection

5.7.7 Local Interconnect Network (LIN)

The LIN module supported is compliant to the LIN 2.1 protocol specification. This module can be programmed to work either as an SCI or as a LIN. The SCI's hardware features 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.

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 - Overrun Error Detection
- LIN8 - 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 - Break Error Detection
- LIN16 - Communication Access Latency Profiling Using On-Chip Timer
- LIN17 - Software Test of Function Including Error Tests Using EPG

5.7.8 Inter-Integrated Circuit (I2C)

The Inter-Integrated Circuit (I2C) module provides a multimaster 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):

- I2C1 - Software Test of Function Using I/O Loopback
- I2C3 - Information Redundancy Techniques Including End-to-End Safing
- I2C2 - I2C Data Acknowledge Check
- 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

5.7.9 Fast Serial Interface (FSI)

The Fast Serial Interface (FSI) is a serial peripheral capable of reliable and high-speed communication. The FSI is architected specifically to ensure 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):

- 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

5.7.10 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 master and slave modes.

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

5.7.11 External Memory Interface (EMIF)

The External Memory Interface (EMIF) is used to provide device access to off-chip memories or devices, which support a memory interface. Support is provided for both synchronous (SDRAM) and asynchronous (NOR Flash, SRAM) memories. The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- EMIF1 - Information Redundancy Techniques
- EMIF2 - VCRC Check of Static Memory Contents
- EMIF3 - Periodic Software Read Back of Static Configuration Registers
- EMIF4 - Software Read Back of Written Configuration
- EMIF5 - Transmission Redundancy
- EMIF6 - EMIF Access Protection Mechanism
- EMIF7 - EMIF Asynchronous Memory Timeout Protection Mechanism
- EMIF8 - EMIF Access Latency Profiling Using On-Chip Timer
- EMIF9 - Software Test of Function Including Error Tests

Note

Safety critical data from external memories can be transferred or copied to internal memory for higher integrity operations.

5.8 Not Safety Related Elements

The following elements are not recommended to be used in safety related applications implemented using TMS320F28P65xD/S. If used in the end system, applicable measures listed in section 'Suggestions for Improving Freedom From Interference' should be implemented to avoid a cascading failure from these elements adversely affecting implemented safety functions.

- Universal Serial Bus (USB)

6 Management of Random Faults

For a functional safety critical development, it is necessary to manage both systematic and random faults. The TMS320F28P65x 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 TMS320F28P65x 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 TMS320F28P65x 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 TMS320F28P65x MCU or system when a fault is indicated. Multiple potential fault responses are possible during a fault indication. The system integrator is responsible to determine which fault response should be taken to ensure 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.

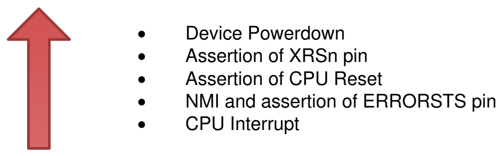


Figure 6-1. Fault Response Severity

- **Device Power Down**: This is the highest priority fault response where the external component (see Section 4.2.5.1) detects malfunctioning of the device or other system components and powers down the TMS320F28P65x MCU. From this state, it is possible to re-enter cold boot to attempt recovery.
- **Assertion of XRSn**: The XRSn reset could be generated from an internal or external monitor that detects a critical fault having potential to violate safety goal. Internal sources generate this fault response when the TMS320F28P65x MCU is not able to handle the internal fault condition by itself (for example, CPU1 (master CPU) is not able to handle NMI by itself). From this state, it is possible to re-enter cold boot and attempt recovery.
- **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 may be the secondary effect due to a fault condition. In addition, CPU2 subsystem generates this fault response when it is not able to handle the internal fault condition by itself (for example, CPU2 is not able to handle NMI by itself). From this state, it is possible to re-enter warm boot phase and attempt recovery.
- **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 C28x CPU. ERRORSTS pin will also be asserted along with NMI. Depending on the system level requirements, the fault can be handled either internal to the TMS320F28P65x 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 may be useful for improving independence of function when using the TMS320F28P65x Real-Time MCU:

1. Hold peripherals 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 components cores if they are not used.
4. When possible, separate critical I/O functions by using non adjacent I/O pins/balls.
5. Partition the memory as per the application requirements to respective processing units and configure the Access Protection Mechanism for Memories, for each memory instance such that only the permitted masters have access to memory.
6. Dual Zone Code Security Module (DCSM) can be used for functional safety as firewall to protect shared memories, where functions with different safety integrity levels can be executed from different security zones (zone1, zone2 and unsecured zone) thus mitigating risk originating due to interference among these.
7. ADC11 - Disabling Unused Sources of SOC Inputs to ADC can help avoid interference from unused peripherals to disturb functionality of ADC.
8. DMA9 - Disabling of Unused DMA Trigger Sources can help minimize interference caused by unintentional DMA transfers.
9. CLA11 - Disabling of Unused CLA Task Trigger Sources can mitigate risk of interference caused by the trigger events.
10. When IPC is used in safety critical application, IPC1 - Information Redundancy Techniques Including End to End Safing can detect failure of interference to CPU due to unintentional interrupts form IPC module.
11. To avoid interference from spurious activity on MCU's debug port, JTAG1 - Hardware Disable of JTAG Port will be helpful in preventing this interference.
12. Safety applications running on the CPU can be interfered by unintentional faulty interrupt events to PIE module. PIE7 - Maintaining Interrupt Handler for unused interrupts and PIE8 - Online Monitoring of Interrupts and Events will detect such interfering failures.
13. MCU resources in supporting CPU execution such as memory, interrupt controller, and so forth could be impacted by resources from lower safety integrity safety function coexisting on same MCU. Safety mechanisms such as CPU9 - External Watchdog, SRAM16 - Information Redundancy Techniques, SRAM17 - CPU handling of Illegal Operation, Illegal Results and Instruction Trapping will be able to detect such interference.
14. Critical configuration registers could be victim of interference from bus masters on MCU which implements lower safety integrity functions. These can be protected by SYS1 - Multibit Enable Keys for Control Registers, SYS2 - Lock Mechanism for Control Registers, SYS8 - EALLOW and MEALLOW Protection for Critical Registers.

6.3 Suggestions for Addressing Common Cause Failures

Common cause failures impacting the functions and their safety mechanisms ("diagnostic" or "functional redundancy") have been analyzed at the internal level (sub-elements of the MCU) by Texas Instruments from a generic SEooC point of view. The system integrator shall supplement this analysis by analyzing relevant sub-elements of the MCU based on the intended use case, including pin-level connections, impact of the pin or ball level interactions on the MCU package, and aspects related to the selected I/O multiplexing. Appropriate to the safety concept, the applicable safety measures from the list below shall be implemented for addressing the common cause failures when using the MCU. When diagnostic or functional redundancy, it requires further analysis of the common cause failures.

- 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, PWR2 - External Watchdog.
- In general, a clock source which is common to redundant functions should be monitored and any failures on the same 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 failure affecting Internal Watchdog(WD) and CPU, it is recommended to use either INTOSC2 or X1/X2 as clock source to PLL.
- Failure of common reset signal to redundant functions can be detected by RST1 - External Monitoring of Warm Reset (XRSn) and RST2 - Reset Cause Information.
- Common cause failure on Interconnect logic could impact both redundant functions and also safety mechanism in 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 failure could impact two functions used in redundant way. In case of communication peripherals 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 so on.
- Using different voltage references and SOC trigger sources for ADC (see Hardware Redundancy).
- Using nonadjacent 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 TMS320F28P65x 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, it is placed at an appropriate place based on the applicable use case scenario. For a detailed description or implementation details for a diagnostic, see the device-specific technical reference manual.

6.4.1 C2000 MCU Infrastructure Components

6.4.1.1 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 may be selected via software, as can the expected ratio of clock frequencies. Deviation from the expected ratio can be configured to generate an interrupt. For more information on the clock selection options implemented, see the device-specific data sheet. For DCC programming details, see the TRM.

6.4.1.2 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, see 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 via 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.

6.4.1.3 Clock Integrity Check Using HRPWM

Calibration logic of OTTO (HRPWM) can be used to detect incorrect system clock (SYSCLK) frequencies. The clock whose frequency needs to be 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.

6.4.1.4 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 would happen with conditions like boundary exceeding, incorrect pointers, stack overflow or corruption, and so forth. Reads from the protected registers are always allowed. It is recommended to issue an EDIS (or MEDIS) for protection once write for the protected registers are complete.

6.4.1.5 External Clock Monitoring via XCLKOUT

The TMS320F28P65xD/S MCU device family provides capability to export select internal clocking signals for external monitoring. This feature can be configured via 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, see the device-specific data sheet. Export of internal clocks on the XCLKOUT outputs is not enabled by default and must be enabled via software. It is possible to disable and configure this diagnostic via software.

6.4.1.6 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 ensuring proper signaling (for example, low duration) when it is asserted. Error response, diagnostic testability, and any necessary software requirements are defined by the external monitor selected by the system integrator.

6.4.1.7 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 should be configured with over voltage and under voltage 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.

6.4.1.8 External Watchdog

External watchdog helps to reduce common mode failure, as it 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 an indication of inadvertent activation of logic which results in impact to safety critical execution. Any watchdog added externally should include a combination of temporal and logical monitoring of program sequence [IEC61508-7, clause A.9.3] or other appropriate methods such that high diagnostic effectiveness can be claimed.

6.4.1.9 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 will remain in reset until the voltage returns to the operational range. The BOR is enabled by default.

6.4.1.10 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 in order to reduce unintended activation of the reset circuitry. The glitch filters are enabled by default and operates continuously. Their behavior cannot be changed by the software.

6.4.1.11 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 via an external debug tool that writes commands to the device using the JTAG port.

6.4.1.12 Lockout of JTAG Access Using OTP

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

6.4.1.13 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, see 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, 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 need to be performed to enable the WWD operation. For details of programming the internal watchdogs, see 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.

6.4.1.14 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 accesses to the registers are blocked. Locked registers cannot be updated by software. Once locked, only reset can unlock the registers.

6.4.1.15 Missing Clock Detect (MCD)

The missing clock detector (MCD) is a safety diagnostic that can be used to detect failure of PLL reference clock. MCD utilizes the embedded 10 MHz 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 via software.

6.4.1.16 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 may not always be successful. In such a situation, 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.

6.4.1.17 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/cold reset. These registers are used to store the NMIFLG information before reset assertion. This 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.

6.4.1.18 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, see 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.

6.4.1.19 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 can be enabled on channel ADCIN13 on ADCA by setting the ENABLE bit in the TSNSCTL register.

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.

6.4.1.20 Periodic Software Read Back of Static Configuration Registers

Configuration registers are typically configured in the beginning and hold the value till the particular task execution. Periodic read back of configuration registers can provide a diagnostic for inadvertent writes or disturb of these registers.

The diagnostic coverage can be improved by extending the test to include read back 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 ensure 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 setting in peripheral as well as PIE to ensure that correct interrupt is being serviced.

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

6.4.1.21 Peripheral Clock Gating (PCLKCR)

Peripherals can be clock gated on a per peripheral basis. This can be utilized to disable unused features such that they 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. It is possible to lock the particular configuration to avoid inadvertent writes.

6.4.1.22 Peripheral Soft Reset (SOFTPRES)

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

6.4.1.23 Peripheral Access Protection – Type 1

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

6.4.1.24 Software Test of Reset – Type 1

A software test for detecting basic functionality as well as 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 ensuring the correct setting of reset cause register and making sure only the intended logic is reset. Additionally, the SIMRESET configuration for SYSRS or XRS assertion through software write may be used for this test.

In order 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 testing of complex features of the peripheral including error tests necessary to confirm correct propagation of reset. For peripheral specific software test of function including error tests, see the device-specific safety mechanism listed for the peripheral.

Additionally, secondary CPU (CM/CPU2) reset is also controlled via primary (CPU1) through configuration writes enabling the software test of these resets.

6.4.1.25 PLL Lock Profiling Using On-Chip Timer

Clock setup for the TMS320F28P65xD/S 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 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 via XCLKOUT to ensure proper clock output
- Clock Integrity Check Using CPU Timer
- Clock Integrity Check Using HRPWM

6.4.1.26 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.

6.4.1.27 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped registers in this module, it is recommended for software implement a test to confirm proper configuration of all control register 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 C28x CPU only, this safety mechanism for CLA module has to be executed by C28x CPU.

6.4.1.28 Software Test of ERRORSTS Functionality

As indicated in Figure 4-8, ERRORSTS pin is an integral part of MCU safety concept used for indicating to an external system about a critical error occurring within in the MCU. Proper functioning of ERRORSTS pin and error handling of the system external to MCU can be checked by asserting ERRORSTS pin by generating an error condition using one of the software provided ways (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.

6.4.1.29 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 issue of missing clock NMI and setting of missing clock status flag (MCDCR.MCLKSTS).

6.4.1.30 Software Test of Watchdog (WD) Operation

A basic test of the internal watchdog operation can be performed via software including checking of error response by configuring the expected lower and higher threshold value 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.

6.4.1.31 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 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 will report an ERROR status flag and send an interrupt to the PIE.

Proper operation of Dual clock comparator (DCC) functionality can be checked by configuring DCC with 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.

6.4.1.32 PLL Lock Indication

PLL Lock functionality is implemented by comparing the difference (error) between the feedback clock and reference clock through 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 will go outside of that range. In such a case, PLL Lock signal will go from 1 to 0 indicating PLL is out of lock. DCC can be used to detect that drift has occurred.

6.4.1.33 Software Test of DCC Functionality Including Error Tests

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

6.4.1.34 Software Test of PLL Functionality Including Error Tests

APLL lock indication functionality can be checked by using 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. Case where APLL does not get locked before the Timer expires (that is, taking more cycles than expected), timer interrupt is triggered to CPU and further action can be taken by the user-defined software. In order to ensure the correctness of APLL clock and hence the system clock generation, it is recommended to use the standard clock sources for timer module (like INTOSC, Crystal etc.) instead of system clock for checking the APLL clock generation correctness.

6.4.1.35 Interleaving of FSM States

Main control FSM includes a hamming distance of 2 to ensure that any single bit flip does not cause the state machine to transition into another valid state. The FSM will default 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 will (expected to) relock and continue. No error will be generated for the bit fail scenario.

6.4.1.36 Decryption of Encrypted Data Output Using Same KEY and IV

A software test can be utilized to test basic functionality which 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. Software requirements necessary are defined by the software implemented by the system integrator.

6.4.1.37 Software Test of Standalone GHASH Operation

The standalone GHASH needs to 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 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 will return 0x00000000 data. Access to an erased flash (default state for a new device) will return 0xFFFFFFFF. Both 0x00000000 and 0xFFFFFFFF are decoded as invalid instructions so that an erased flash or cleared memory, or an invalid address will force the CPU to ITRAP. Installation of software handlers to support the hardware illegal operation and instruction trapping is highly recommended

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

- Illegal instruction
- *TMS320C28x FPU Primer*

6.4.2.2 Reciprocal Comparison by Software

Each CPU subsystem has a pair of diverse processing units (C28 and CLA) with different architecture and instruction set. This enables one processing unit to be used for handling the time critical portion code (control CPU) and other processing unit (supervisor CPU) to execute non critical portion of the code, perform diagnostic functions and supervise execution of the control CPU.

In case of identification of fault during diagnostic functions of the supervisor CPU, it can cause the MCU to move to a safe state. This 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 need to be performed several times during a FTTI. Reciprocal comparison is a software diagnostic feature and hence care should be taken to avoid common mode failures. The final attained coverage will depend on 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 set and compiler is used for enabling this. The diversity can be further increased by having separate algorithms being executed in both the cores. In case, failure is identified during reciprocal comparison, NMI can be triggered by software and this in turn will assert ERRORSTS. Also during runtime, the CLA can have access to the GPIO data registers and can indicate an error condition on a GPIO pin independent of the C28x CPU.

6.4.2.3 Software Test of CPU

It is possible to test the integrity of various CPU logic (C28x, FPU, TMU, and so forth) using software-based self-test library (STL). TI provides a C28x_STL startup test library with 60% diagnostic coverage for C28x, FPU, TMU, and VCRC. For more details, see Section 4.2.6 and the documentation found within TMS320F28P65x C28x_STL software installation. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.2.4 CPU Hardware Built-In Self-Test (HWBIST)

The C2000 MCU device family has hardware logic to provide a very high diagnostic coverage on the CPUs at a transistor level during start-up and application time. This logic utilizes Design for Test (DfT) structures inserted into the device for rapid execution of high quality manufacturing tests, but with an internal test engine rather than external automated test equipment (ATE). This technique has proven to be effective in providing high coverage in less time.

The HWBIST tests must be triggered by the software. User may select to run all tests, or only a subset of the tests based on the execution time allocated to the HWBIST diagnostic. This time sliced test feature enables the HWBIST to be used effectively as a runtime diagnostic with execution of test in parallel with the application. Execution of HWBIST results in a much higher level of transistor switching per clock cycle than during normal software execution due to the high efficiency of the test. For more information, see *C2000™ Hardware Built-in Self-Test*. HWBIST execution failure will trigger NMI to the same CPU and other CPUs (if available based on the device configuration). After HWBIST execution, reset is issued to the CPU and the CPU context is restored.

6.4.2.5 CPU Hardware Built-In Self-Test (HWBIST) Auto-Coverage

The HWBIST diagnostic is based on a 512-bit signature capture. For a given test, only one code is valid out of 2^{512} possibilities. Therefore, if there is a fault in the HWBIST logic, it is extremely unlikely that the correct passing code will be generated via the fault. The cyclical check applied by the HWBIST module provides an inherent level of self-checking (auto-coverage), which can be considered for application in latent fault diagnostics.

6.4.2.6 CPU Hardware Built-In Self-Test (HWBIST) Fault Injection Capability

HWBIST diagnostic has capability helps to inject faults and check the correct functioning of the CPU Hardware Built-In Self-Test (HWBIST) Auto-Coverage and CPU Hardware Built-In Self-Test (HWBIST) Timeout feature. This can be used to provide latent fault coverage of the diagnostic logic.

6.4.2.7 CPU Hardware Built-In Self-Test (HWBIST) Timeout Feature

HWBIST diagnostic has capability helps to inject faults. HWBIST module expects the self-test to be completed within a certain time frame. If the test is not completed within this time frame, the test is stopped immediately, CPU is reset and NMI (and hence ERRORSTS) is issued to recover from the indeterminate state. This feature is enabled by default once the HWBIST module enters into self-test mode and cannot be disabled by software. After coming out from reset, CPU can read the HWBIST status registers to understand the reset cause and take the required action.

6.4.2.8 Software Test of CLA

It is possible to test the integrity of various CLA blocks (register bank, control unit, data path, and so forth) using 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, see 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.

6.4.2.9 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), floating point underflow or overflow conditions. CLA will interrupt CPU under such conditions. Any access to an invalid memory range will return 0x00000000 data. Access to an erased flash (default state for a new device) will return 0xFFFFFFFF. Both 0x00000000 and 0xFFFFFFFF are decoded as invalid instructions so that an erased flash, cleared memory, or an invalid address will generate an interrupt to CPU. CPU can decode the interrupt cause 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.

6.4.2.10 CLA Liveness Check Using CPU

CLA does not have an independent watchdog of its own. As a result, it is recommended to perform liveness check periodically by the CPU. Typically, 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 E-gas concept (6) to improve the diagnostic coverage.

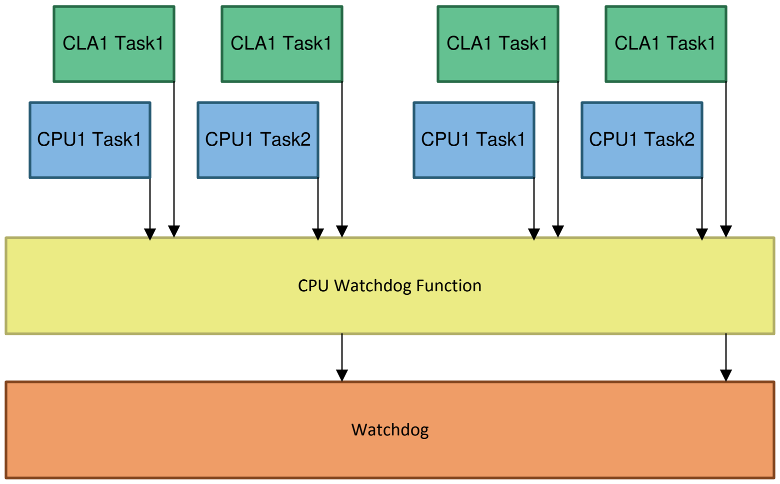


Figure 6-2. CLA Liveness Check

6.4.2.11 Disabling of Unused CLA Task Trigger Sources

The CLA can receive input task triggers from various peripherals and software. To avoid interference from unused trigger sources resulting in disturbance to CLA operation it is recommended to disable these in application.

6.4.2.12 Stack Overflow Detection

A stack overflow in a safety application may result in a catastrophic software crash due to data corruption, lost return addresses, or both. Hence, it is important to detect an impending stack overflow. The enhanced bus comparator (EBC) unit in the ERAD module can monitor the internal address and data buses, and trigger the RTOSINT interrupt when a specified bus and mask matches a specified value. Hence, the basic approach for detecting stack overflow will be to configure the EBC unit to trigger an interrupt when the data write address bus falls within some range prior to the end of a stack. Since this memory is reserved for stack usage only, a data write within the specified address range indicates that the stack usage is approaching its allocated size limit. Detection of an impending stack overflow triggers a maskable interrupt. Programmed error response and any necessary software requirements are defined by the system integrator.

6.4.2.13 VCRC Check of Static Memory Contents

The TMS320F28P65xD/S MCU device family includes co-processor implementing cyclic redundancy check (CRC) using standard polynomial. The CRC module can be used to test the integrity of SRAM/Flash/OTP/ 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.

6.4.2.14 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, it is extremely unlikely that the correct passing code will be generated via the fault.

6.4.2.15 Hardware Redundancy Using Lockstep Compare Module (LCM)

Hardware Redundancy implemented by lockstep architecture (two hardware modules executing the same function and the output of the hardware modules are continuously compared) is a proven method for achieving high diagnostic coverage for both permanent and transient faults. Lockstep Compare Module (LCM) is used to implement lockstep compare functionality and indicate an error.

6.4.2.16 Self-Test Logic for LCM

The LCM self-test logic is implemented to meet the latent fault coverage requirements for the lockstep comparator. The self-test for the comparator has two different modes: match test and mismatch test. When the self-test is initiated, the two different test modes are executed on the two comparators one after the other. A self-test error will trigger error aggregator logic (for example, NMIWD).

6.4.2.17 LCM Compare Error Forcing Mode

This mode will check the error signaling path of a lockstep compare error from the LCM to the error aggregator logic (e.g. NMIWD). This test can be executed at any time during the device operation when the lockstep comparator is enabled. Executing this test will result in assertion of the lockstep comparison error signal. There are separate configurations available to force a compare error to each of the redundant comparators. Once this mode is enabled, it will be active for only one cycle. Enabling the error forcing mode will cause the first functional output from the primary module to be inverted.

It is not possible to execute this test with the debugger connected since the lockstep comparator gets disabled with debugger connection.

6.4.2.18 LCM MMR Parity

The parity scheme provides one parity bit per byte of data in the corresponding registers. Updates to any of the constantly-monitored registers causes an update to the parity bit. A single bit fault can therefore immediately flag an error. If the parity check determines a parity error has occurred, a dedicated error output line from the LCM module will flag an error to the system.

6.4.2.19 Test of LCM MMR Parity

An error can be injected into the register parity error protection that exists for critical LCM registers, in order to test for latent faults. This error is inserted by forcing a particular byte to output a failing parity state to the system. This will then trigger an NMI to the NMIWD and set the corresponding SYS_STATUS_REGS.REGPARITY_ERR_FLG bits.

6.4.2.20 Lockstep Self-test Mux Select Logic Fault Detection

When the LCM is not in self-test mode, the self-test logic should drive a value of {0,1} at the comparator inputs. This will help detect faults in the select logic of the self-test mux which selects between functional input and self-test input.

6.4.2.21 Redundancy in LCM Comparator

The comparator block is instantiated redundantly to enable availability of the lockstep module during self-test execution and provide additional failure detection capability for the comparator logic. A failure in the lockstep compare shall be indicated using the LCM_STATUS.CMP_FAIL status register and generates an NMI.

6.4.2.22 Embedded Real Time Analysis and Diagnostic (ERAD)

The ERAD module provides system analysis capabilities that can be used to detect faults in CPU and other logic on MCU by configuring 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 which 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. It 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, it can work independently and generate RTOSINT interrupt in case of event match occurs. This module can be used as a continuous online monitor of system events on MCU.

6.4.2.23 Inbuilt Hardware Redundancy in ERAD Bus Comparator Module

The ERAD bus comparator units can be used to monitor CPU buses and ERAD supports such eight 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 logics to generate new events as required. The faults in the comparison block can be detected by configuring two comparator blocks to monitor 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.

6.4.3 Memory (Flash, SRAM and ROM)

6.4.3.1 Bit Multiplexing in Flash Memory Array

The flash modules implemented in this device family have a bit multiplexing scheme implemented such that the bits accessed to generate a logical (CPU) word are not physically adjacent. This scheme helps to reduce the probability of physical multi-bit faults resulting in logical multi-bit faults. Rather, they manifest as multiple single bit faults. As the SECDED flash ECC can correct a single bit fault and detect 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.

6.4.3.2 Bit Multiplexing in SRAM Memory Array

The SRAM modules implemented in the TMS320F28P65xD/S MCU device family have a bit multiplexing scheme implemented such that 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 they manifest as multiple single bit faults. The SECDED SRAM ECC diagnostic can correct a single bit fault and detect 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.

6.4.3.3 Data Scrubbing to Detect and Correct Memory Errors

Bus masters (CPU, CLA or DMA) can be configured to provide dummy reads to the memory (provided a particular bus master 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, interrupt is issued once the count exceeds the preset threshold in the case of correctable errors and NMI will be issued in the case of uncorrectable errors.

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

6.4.3.4 Flash ECC

The on-chip flash memory is supported by single error correction, double error detection (SECDED) 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 is 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 will generate an NMI and ERRORSTS pin is asserted. 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.

6.4.3.5 Flash Program Verify and Erase Verify Check

Whenever any program and erase operation is done, the flash controller will perform program and erase verify check. If the program and erase operation is failed, FSM status register (STATCMD) will indicate the error by setting the corresponding flags into the status register.

6.4.3.6 Flash Program and Erase Protection

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

6.4.3.7 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 execution status of a command. A CMDPASS bit indicates whether an operation passed or failed. There are 5 different fail bits which indicate different failure mechanisms when a command fails. Setting of the CMDDONE status indicator at the end of command execution also triggers the assertion of an interrupt event to the system.

6.4.3.8 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. It is not possible to program an existing 0 to a 1 in the Flash. If such a condition is detected, then the program operation will terminate with a FAILINVDATA error in STATCMD without issuing the programming command.

6.4.3.9 On-Demand Software Program Verify and Blank Check

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

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

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

6.4.3.11 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 the data from the flash bank access port and will do detection and correction using that logic. Thus, the hardware used for ECC generation and for ECC detection and correction are separate. This allows easier diagnostic if there is a fault in either of these sets of ECC logic.

6.4.3.12 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 expicitly 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.

6.4.3.13 Software Test of ECC Logic

It is possible to test the functionality of the SRAM ECC by injecting single bit and double bit errors in test mode and performing reads on locations with ECC errors, and 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, see 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.

6.4.3.14 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 will be cleared. A software test can be performed to ascertain the proper behavior of this logic. The following sequence of operation can be performed.

1. Disable the prefetch mechanism, enable the timer and Watchdog. Execute a particular function which might have linear code and code with multiple discontinuities. Store the time “time_1” (timer value) taken for executing this function.
2. Enable the prefetch mechanism and execute the same function again. Store the time “time_2” (timer value) taken for executing this function. This value should be less than the time_1 (time_1 > time_2). We can mark this timer value as a golden value and should 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, user needs to make sure that the prefetch logic test function should be aligned/located in particular location within flash to guarantee 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.

6.4.3.15 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, Write) to individual RAM blocks from individual masters (CPU1, CPU2, CPU1.CLA1, CPU2.CLA1, CPU1.DMA1, CPU2.DMA1). There is no protection for read accesses. Therefore, reads are always allowed from all the masters which have access to that RAM block. To identify conditions when the master access to an SRAM is blocked, see the device-specific technical reference manual. This configuration can be changed during run-time and allows memory to block access from specific masters or specific application threads within the same master. This capability helps support freedom from interference requirements required by some applications.

6.4.3.16 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, see the TMS320F28P65xD/S 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 it is possible to monitor the number of corrected errors. 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, see the device-specific data sheet.

6.4.3.17 SRAM Parity

Selected on-chip SRAMs support parity diagnostic with separate parity bits for data and address. For the specific address ranges that support parity, see 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 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, see the device-specific data sheet.

6.4.3.18 Software Test of Parity Logic

It is possible to test the functionality of parity error detection logic 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.

6.4.3.19 Software Test of SRAM

It is possible to test the integrity of SRAM (bit cells, address decoder and sense amplifier logic) 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 VCRC can also be performed in place of destructive test (test where memory contents need to be restored after the test). For details on implementing this particular test, check the safety package delivered with this specific C2000 MCU device.

6.4.3.20 Memory Power-On Self-Test (MPOST)

Start-up test of the memories provides detection for permanent faults inside on-chip memories. Some of the C2000 devices 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, 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, it is not be possible to test the boot ROM using PBIST. Hence a separate boot ROM checksum test will be done prior to PBIST. Prior to performing any test using PBIST, an always fail test case is executed. This is to validate the proper functioning of the PBIST controller and its ability to indicate failure. For more details, see *C2000 Memory Power-On Self-Test (M-POST)*.

6.4.3.21 Background CRC

Background CRC is a non-intrusive CRC calculation module. CRC calculation needs to be kicked off by the application and it can continue with the regular application execution. Once kicked off, the module will trigger an interrupt on completion of test or occurrence of an error. The module generates memory access which will be serviced once all pending functional accesses from CPU and DMA are complete. Since the memory accesses happen only during idle cycles, the MIPS impact of performing CRC computation is zero or very minimal. This modules helps to cover the faults in memory bit cells, address decoder logic and sense amplifier (all memory logic involved during read operation).

6.4.3.22 Watchdog for Background CRC

The CRC module has an embedded windowed watchdog as a diagnostic to check memory test completion in the expected time window. This feature protects against hardware defects or rogue code which cause the memory check not to complete in the predetermined duration. Windowing feature helps detect additional failure modes in the watchdog operation, (stuck watchdog). Watchdog counter is a 32-bit counter and BGCRC_WDCNT reflects the watchdog counter value. The lower and upper window settings for the windowed watchdog are set by configuring BGCRC_WD_MIN and BGCRC_WD_MAX registers respectively. Failure to complete the memory check operation in the configured time window will generate an interrupt or NMI as per the configuration.

6.4.3.23 ROM Parity

ROMs support parity diagnostic with separate parity bits for data and address. For the specific address ranges that support parity, see 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 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.

6.4.3.24 Redundant Parity Engine

The parity calculation is done twice at the output of the memory and the two parity outputs are expected to be the same. If either parity engine reports an error, the error is transmitted as NMI to the CPU.

6.4.4 On-Chip Communication Including Bus-Arbitration

6.4.4.1 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, it is possible to use one of the other counters as a diagnostic on the first, using periodic check via 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.

6.4.4.2 DMA Overflow Interrupt

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

6.4.4.3 Event Timestamping Using IPC Counter

IPC has a 64-bit free-running IPCCOUNTERH/L that can be used for timestamping events between the processors. The timestamp can be sent along with the payload and this information can be used by the software running on the receiver CPU to determine the time required for completing the command. If the message is not received within the expected time limit, the receiver CPU can initiate an error response. The round trip delay can be estimated by the transmitter CPU based on the message acknowledge send by receiver CPU.

6.4.4.4 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. This includes disabling of the unused interrupts, enabling the unused interrupts and return to the application in the interrupt service routine (ISR), and so forth. Receiving of an interrupt not used in the application might be an early indication of some faulty scenarios within the C2000 MCU. Hence, it is highly recommended to enable all the interrupts and configure the ISR to a common routine for logging or error handling.

6.4.4.5 Majority Voting and Error Detection of Link Pointer

The link pointer OTP location is not protected by ECC. To provide better security to the customer code and enable application safety, majority voting and data consistency based error detection is implemented. 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 will set the Zx_LINKPOINTERERR register.

6.4.4.6 PIE Double SRAM Comparison Check

In order to check the PIE double SRAM comparison feature and the fault handling, it is possible to inject different data to both the SRAMs. On accessing the particular location, in which there is data mismatch, the CPU will jump to error management routine. For details for implementation of this check, see the *CPU1 and CPU2 PIE Vector Address Validity Check* chapter in the device-specific technical reference manual.

6.4.4.7 PIE Double SRAM Hardware Comparison

PIE SRAM address space is duplicated and data is placed in two memories. During write operations both the SRAMs are simultaneously updated and on reading the values from both the memories are compared. In case of error during comparison, the CPU will branch to a predefined location based on the user configuration. The location will have the routine for error management.

6.4.4.8 Power-Up Pre-Operational Security Checks

During the device boot, it 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 ensure correct initialization of device security which includes checks to confirm correct link-pointer settings, CRC lock setting, correct partitioning of secure RAM blocks and Flash sectors (Grab Bits), setting for execute only protection for secure RAM blocks and Flash sectors, correct partitioning of the CLA and Flash Bank2 and correct settings for boot configuration. Before starting the execution of downloaded code user should check the integrity of the code using CRC function. Once pre-operational checks are successfully completed with expected results, the device can enter the application phase.

6.4.4.9 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 which got triggered. This flag registers can be periodically read to ascertain that no ePWM trip zones, ePWM syncing or GPIO output signaling is missed.

6.4.4.10 Software Test of ePIE Operation Including Error Tests

A software test for testing the basic functionality as well as 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 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 ensuring that the interrupt is serviced and serviced in proper order. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.4.11 Disabling of Unused DMA Trigger Sources

The unintended trigger of DMA transfers could corrupt critical data and that could be a potential source of interference to safety critical applications. In order to avoid initiation of the unintended DMA transfers, it is recommended that unused DMA channels and DMA trigger sources are disabled at source or by configuring DMACHSRCSELx registers.

6.4.4.12 Software Test of CLB Function Including Error Tests

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

6.4.4.13 Monitoring of CLB by eCAP or eQEP

The Configurable Logic Block (CLB) outputs can be monitored for proper operation by eCAP or eQEP using internal connections. User should configure 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 would be based on customer configuration.

6.4.4.14 Periodic Software Read Back of SPI Buffer

Using the CLB to SPI interface, without the CPU intervention, CLB will be able to send out a continuous stream of 16-bit data to SPI which then be saved in the device memory using the FIFO and DMA mechanism of the SPI. In order to ensure that the data from CLB is reaching to SPI and then to device memory, it is recommended for the software to implement tests to read back the SPI FIFO contents as well as to read back the final Memory contents and ensure that they are same.

6.4.4.15 IPC 64-Bit Counter Value Plausibility Check

64-bit counter in IPC is a free-running clocked at PLLSYSCLK and reset by SYSRSn. For reasonable on-time of the device the counter is expected to be always incrementing value. The 64-bit free-running counter can be tested by software for plausibility checks on its values. For example, software can check if the value on the TIMESTAMP is always incrementing, and does not have any unusually High or Low count values, and so forth.

6.4.5 Digital I/O

6.4.5.1 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.

6.4.5.2 ePWM Application Level Safety Mechanism

ePWM is typically used as the output signal in closed loop control applications such as EV traction, DC-DC and industrial drive. In such applications, the failure in ePWM output, such as stuck-at fault or frequency or duty cycle change, will result in disturbance to control loop parameters or variables, leading to conditions such as over voltage, over current or over temperature. By monitoring characteristics of these control loop parameters implemented at application-level, faults in the ePWM module can be detected.

6.4.5.3 ePWM Fault Detection Using XBAR

A combination of ePWM outputs feedback to input X-BAR, GPIO inversion logic and Digital Compare (DC) submodule of ePWM can be used for implementing simple (for example, signal cross over) 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.

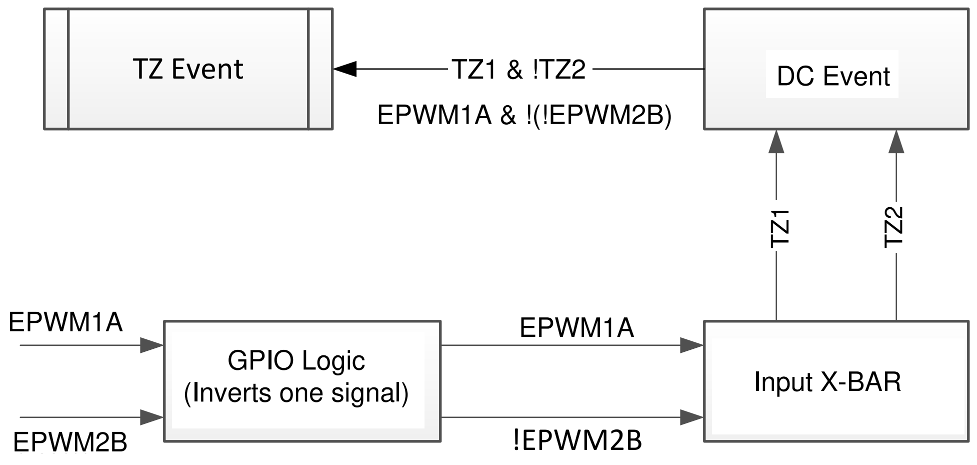


Figure 6-3. ePWM Fault Detection Using X-BAR

6.4.5.4 ePWM Synchronization Check

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

6.4.5.5 Online MINMAX Monitoring of TRIP Events

Critical system parameters can be passed through the CMPSS module and the resultant threshold TRIP signals can be monitored to transition within a predefined TBCNT MINMAX window for every PWM cycle. This event detection feature is part of the ePWM digital compare submodule and is enabled by software.

6.4.5.6 Fault Avoidance Using Minimum Dead Band

The Minimum Dead Band feature can be configured to guarantee a minimum inactive gap (dead band) between the active pulse phases of the two PWM channels and across PWM instances, thereby making the PWM system immune to faults in the preceding digital logic that could result in a cross over.

6.4.5.7 Fault Avoidance Using Illegal Combo Logic

The Illegal Combination Logic submodule can be configured to guarantee a safe value of the PWM outputs for the illegal combinations arising on the input PWM signals. This safety mechanism (applicable also across PWM instances), makes the PWM system immune to faults in the preceding digital logic that could result in undesirable PWM outputs.

6.4.5.8 Diode Emulation Mode Monitoring

Diode Emulation features generation of a TRIP event if the module stays in the Diode Emulation mode for a duration longer than that configured by the user into the Diode Emulation monitor threshold register. This serves as a safety mechanism against faults in the Diode Emulation logic.

6.4.5.9 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 system. In such applications, it is possible to monitor eQEP outputs for saturation, zero value or out of range based on the application requirement. While estimating the speed/position of rotating machinery, the application can set bounds on the measured speed/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.

6.4.5.10 eQEP Quadrature Watchdog

eQEP peripheral contains a 16-bit watchdog timer that monitors the quadrature-clock to indicate proper 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 will time out and the watchdog interrupt flag will be set. The timeout value is programmable through the watchdog period register.

6.4.5.11 eQEP Software Test of Quadrature Watchdog Functionality

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

6.4.5.12 Hardware Redundancy

Hardware redundancy techniques can be applied via 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, transceiver) TMS320F28P65xD/S MCU.

In case of peripherals like GPIO, XBAR, PWM, OTTO (HRPWM), DAC, CMPSS, 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 via internal or external comparators) or input comparison/voting (comparison of independent inputs to ensure compliance with a defined tolerance range (time, value)). In such scenarios, the system can be designed such that the failure of one input/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 ensure that both sources are active while tripping and thus providing latent fault coverage for the trip logic.

In case of peripherals like SDFM, ADC, ECAP, EQEP and so forth, hardware redundancy may be implemented by having multiple instance of the peripheral sample the same input and simultaneously perform the same operation followed by cross check of the output values. The ADC on this device has a built-in hardware-based error checker for doing this comparison. See Hardware Redundancy with ADC Safety Checker for more details.

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

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

While implementing hardware redundancy for GPIO module, it is recommended to use nonadjacent GPIO pins from different GPIO groups to avoid common cause failures.

6.4.5.13 HRPWM Built-In Self-Check and Diagnostic Capabilities

The micro edge positioner (MEP) logic in HRPWM is capable of placing an edge in one of 255 discrete time steps. The size of these steps is of the order of 150 ps. For typical MEP step size, see the device-specific data sheet. The MEP step size varies based on 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 should use the TI-supplied MEP scale factor optimization (SFO) software function. 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 built in self-check and diagnostic capabilities that can be used to determine the optimum 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 optimum 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. Proper System Clock frequency operation is verified by comparing the MEP scale factor value returned with the expected value.

6.4.5.14 Information Redundancy Techniques

Information redundancy techniques can be applied via software as an additional runtime diagnostic. In order to provide diagnostic coverage for network elements outside the C2000 MCU (wiring harness, connectors, 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 ensure correctness. The multiple execution and result comparison may be based on either the same code executed multiple times or diversified software code implemented. For details regarding the implementation, see the ISO26262-5, D.2.3.4.

In the case of the DMA, information redundancy techniques refers to additional information besides the data payload which ensures data integrity. For example, SECDED codes, parity codes, CRCs etc. enable information redundancy.

Typical control applications involve measuring three phase the voltage and current. These values are either sampled directly using the on chip ADC or send to the TMS320F28P65xD/S MCU by the sensors which are captured using ECAP, SDFM, 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, 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, critical data, program, variables, and so forth can be stored redundantly and compared before it is getting used. Care should be taken to avoid compiler optimizing code containing redundant data/programs.

6.4.5.15 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 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.

6.4.5.16 Monitoring of ePWM by ADC

The ePWM outputs can be monitored for proper operation by ADC using a 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 link (9). Error response, diagnostic testability, and any software requirements are defined by the software implemented by the system integrator.

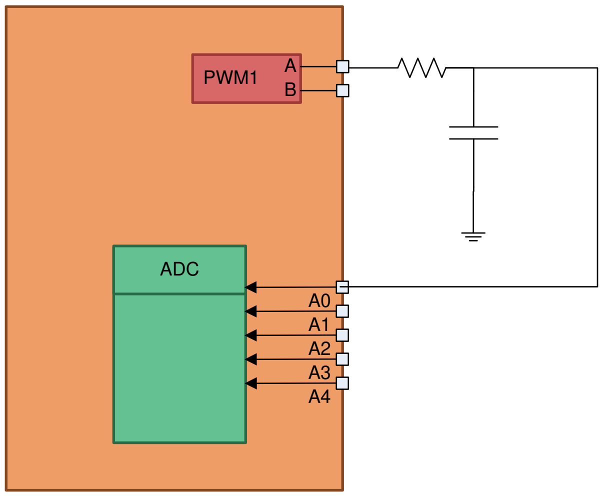


Figure 6-4. Monitoring of ePWM by ADC

6.4.5.17 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 in nature and these periodic events can be monitored and incoherence in the events can be used for fault detection. A few places where online monitoring periodic interrupts and events can be employed include:

- Periodic generation of ADC start of conversion (SoC) (x): 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 may also 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 which are normally not expected during the correct operation can also be used to improve the diagnostic coverage (for example, ECC correctable error interrupt).

6.4.5.18 SDFM Comparator Filter for Online Monitoring

Comparator unit of SDFM can be used for online monitoring of primary filter’s operation. The comparator filter has a configurable Sinc filter whose output is compared with two programmed threshold levels to detect over and under-value conditions. In case comparator filter’s data output crosses low or high threshold limit, it will fire interrupt to the CPU.

6.4.5.19 SD Modulator Clock Fail Detection Mechanism

When SD modulator clock fails or goes missing for 256 continuous system clock cycles, clock fail detection submodule in the input control unit of SD modulator detects the failure and generates an interrupt to CPU. This mechanism can be used to detect missing modulator clock faults or any faults in digital IO connecting modulator clock.

6.4.5.20 Software Test of Function Including Error Tests

A software test can be utilized to test 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. Software requirements necessary are defined by the software implemented by the system integrator.

Ideas for creating some module specific tests functionality and error tests are shown below:

- SDFM functionality can be checked by sending a known input test sequence to the C2000 MCU, process it using the digital decimation filters and cross check the value against a known value. For detecting faults in comparator interrupt generation logic, a test pattern can be created to configure the high/low threshold register values to min/max values respectively. Interrupt should always be generated with such a configuration.
- 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.
- EMIF functionality can be checked by moving a known good data from an external memory to the internal memory and vice versa and checking for data consistency using CRC or other mechanisms. The test should be repeated for all the masters having access to the external memories. In addition, the test should provide coverage to all the interface pins used for connecting external memory to the C2000 MCU.
- Software test of input and output X-BAR module can be performed by having a loop created (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 it 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.
- 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 enable (XINTxCR.ENABLE) functionality as well.
- IPC functionality can be checked by using interrupts or polling method by periodically sending test commands and message as defined by software. Timestamping information using the IPCCOUNTERH/L can be embedded along with the message to estimate the delay in communication.
- 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.
- 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) sub-modules. The individual sub-modules can be tested by providing suitable stimulus using PWM and observing the response using one of the capture (timestamping) modules (eCAP, XINT, eQEP, and so forth). It is recommended to cover the various register values associated with application configuration while performing the software test. Due to the regular linear nature of the various sub-modules, it is possible to get high coverage using a software test.
- A software test of SRAM wrapper logic should provide diagnostic coverage for arbitration between various masters 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 (see Section 6.4.3.19).
- The interconnect (INC) functionality can be tested by writing complementary data-patterns like 0xA5A5,0x5A5A, and so forth from processing units viz CPU and CLA, and reading back it from registers of the IPs connected via different bridges. The read-back data can be compared with expected golden values to ensure 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 reset feature by configuring the registers to 0xA5A5 pattern, asserting soft reset of DAC, reading back the registers and comparing the read back value with the expected reset value. Lock register can be checked to ensure it is set-once. Also, the registers which are getting locked must not update when written. To test core functionality of the DAC module, it 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 ensure proper operation. Extreme corner values of DAC as per application can be programmed and tested to check the successful conversion of digital to analog module across a valid range.
- Comparator sub-system (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. These can be read back and compared against expected values. These accesses can be covered by applicable masters viz. DMA, CLA and CPU. Features of the CMPSS module such as ramp decrement can be checked for counting down of RAMPDLYA after it is loaded from RAMPDLYS by a rising PWMSYNC signal. It should be ensured that the decrementer reduces to zero and stays 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 COMPHSTS/COMPLSTS changes with change in filter output. Applicable range of filter clock prescaler values (CTRIPLFILCLKCTL) can be exercised to ensure 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, 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 also vary 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 ensure that all access other than execute access is blocked.

6.4.5.21 Monitoring of HRPWM by HRCAP

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

6.4.5.22 HRCAP Calibration Logic Test Feature

The calibration logic consists of two free-running counters; one clocked by HRCLK(HRCLKCTR) and the other clocked by SYSCLK(HRSYSCLKCTR). When HRSYSCLKCTR is equal to HRCALIBPERIOD, the calibration block will capture and reset both counter values, then trigger an interrupt indicating a new scale factor is ready to be calculated. The scale factor can be found by dividing HRSYSCLKCAP by HRCLKCAP, see Equation 1. This scale factor computation should be done inside of the calibration interrupt service routine. After computing scale factor, Equation 2 can be applied to get actual measurement of captured value from raw count.

The full details of the calibration block are described in Figure 6-5.

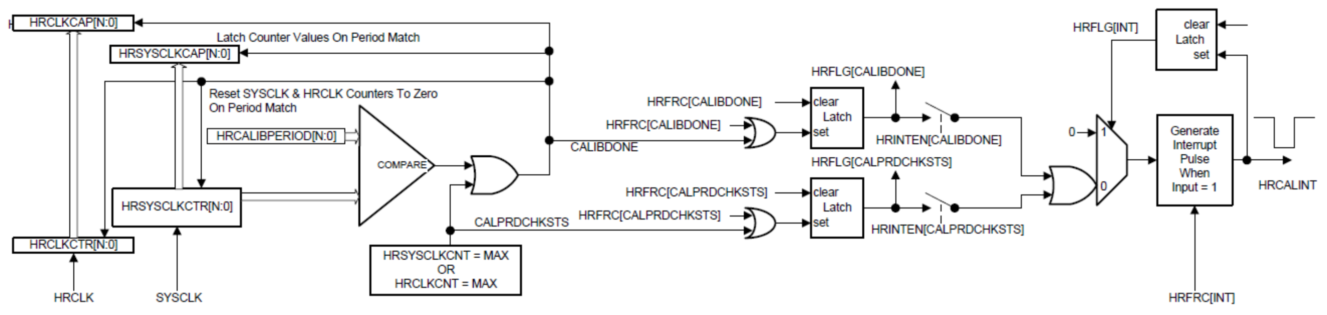


Figure 6-5. HRCAP Calibration

$$ScaleFactor = \frac{HRSYSCLKCAP}{HRCLKCAP} \tag{1}$$

$$Measurement(ns) = \frac{RawCount \times scaleFactor}{128} * SysClkPrd(ns) \tag{2}$$

Note

Even with calibration, noise on the 1.2V VDD supply will negatively affect the standard deviation of the HRCAP sub-module. Care should be taken to ensure that the 1.2V supply is clean, and that noisy internal events such as enabling and disabling clock trees have been minimized while using the HRCAP.

6.4.5.23 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 *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 QMA module's error and interrupt are integrated inside the eQEP module.

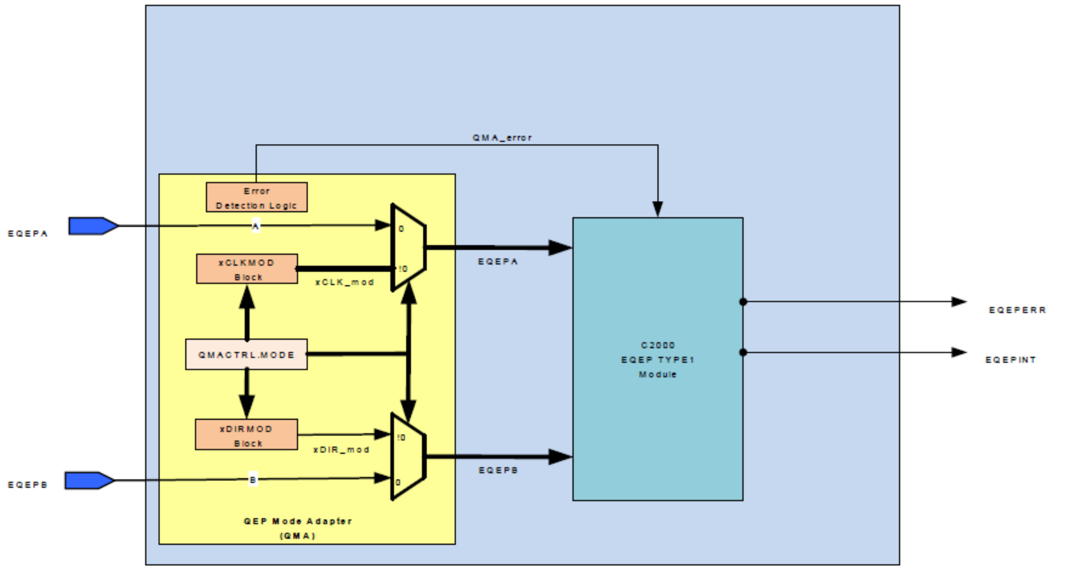


Figure 6-6. QMA Module Block Diagram

6.4.6 Analog I/O

6.4.6.1 ADC Information Redundancy Techniques

Information redundancy techniques can be applied via software for providing runtime diagnostic coverage on ADC conversions. Time redundancy technique can be applied where multiple conversions on same ADC 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.

6.4.6.2 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. Plausibility check of the input signal can be checked with the help of comparator by setting the proper high and low threshold values. The plausibility check of converted results can be checked by using ADC Post Processing Block.

6.4.6.3 ADC Signal Quality Check by Varying Acquisition Window

External signal sources vary in their ability to drive an analog signal quickly and effectively. In order to achieve rated resolution, the signal source needs to charge the sampling capacitor in the ADC core to within 0.5 LSBs 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.

6.4.6.4 Hardware Redundancy with ADC Safety Checker

Hardware redundancy for the ADC may be implemented by having multiple instances of the ADC sample the same input and simultaneously perform the same operation followed by cross check of the output values. The ADC on this device features a hardware-based result safety checker module that automatically compares the results from primary and redundant ADCs once both the results are available. If the computed delta is out of range, the checker can generate a trip event signal that is sent to an ePWM or output crossbar and can trigger a CPU interrupt.

6.4.6.5 Hardware Redundancy of ADC Safety Checker

Hardware redundancy of the ADC result safety checker may be implemented by having multiple instances of the safety checker configured to compare results from same set of ADCs and configured with a similar threshold value for comparison. Out-of-threshold event flags can be compared between the primary and redundant result safety checkers to provide a test of the safety checker diagnostic.

6.4.6.6 Disabling Unused Sources of SOC Inputs to ADC

The start of conversion (SOC) signal input to the ADC module can be triggered by multiple sources, mainly Software, CPU Timers, GPIO, and PWM module instances. In order to achieve freedom from interference due to a fault originating from an peripheral not used in implementing the safety function and cascading into ADC, it is recommended that application configures only the required SOC triggers. This is a way to avoid faults originating from an outside source to impact functionality of ADC.

6.4.6.7 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 ensuring 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.

6.4.6.8 DAC to ADC Loopback Check

Integrity of DAC and ADC can be checked monitoring 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 ensure proper functioning of DAC and ADC. This technique can be applied during run time as well to ensure 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, see the device-specific data sheet or technical reference manual. While performing the loopback checks for 16-bit differential input mode, two DACs should be used to provide input the ADC. To avoid common cause failures, it is recommended to keep the references voltages of the ADC and DAC different while performing the test. In addition, the input signal to ADC should not be driven by any other sources while the test is being performed.

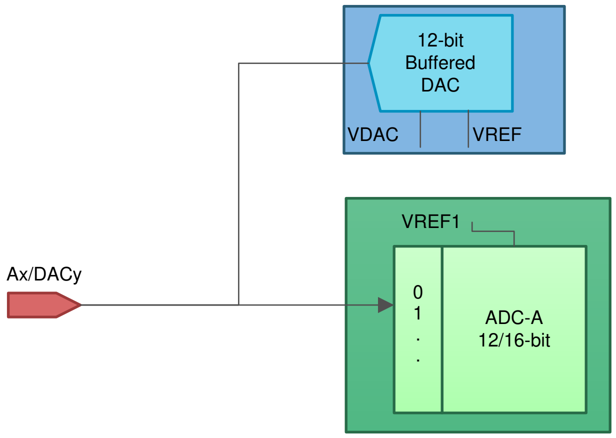


Figure 6-7. DAC to ADC Loopback

6.4.6.9 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.

6.4.6.10 Opens/Shorts Detection Circuit for ADC

An opens/shorts detection circuit is provided to allow customers to detect faults in the ADC input channel. This capability is valid only in single ended mode. For system integrator using differential mode, ADC need to 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 feature availability before using this diagnostic.

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

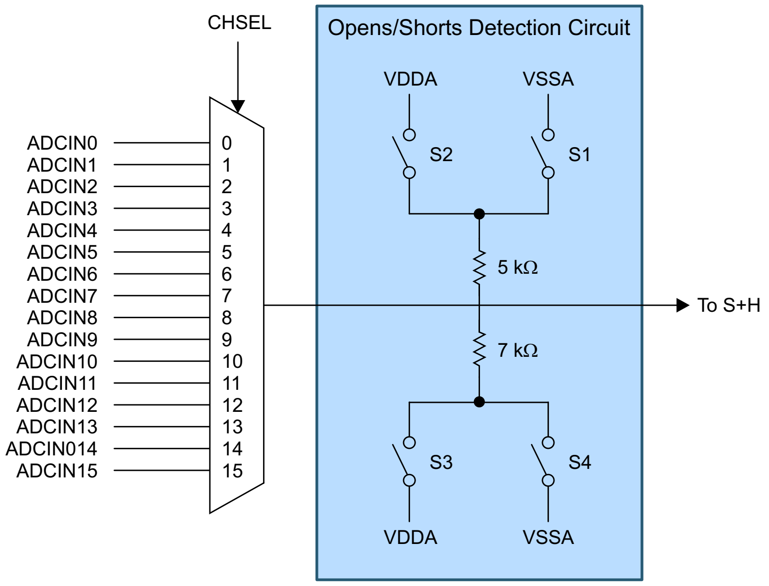


Figure 6-8. Opens/Shorts Detection Circuit

Table 6-1. ADC Open-Shorts 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

6.4.6.11 VDAC Conversion by ADC

Reference voltage input to COMPDACs (VDAC) is double bonded with ADCB input. For detecting faults in VDAC supply and corresponding analog I/O, it can be converted by ADC. The ADC result output can be cross checked against the expected output to identify any faults. Programmed error response and any necessary software requirements are defined by the system integrator.

6.4.7 Data Transmission

6.4.7.1 Bit Error Detection

When the CAN module transmits information onto its bus, it can also monitor the bus to ensure 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.

LIN module supports detection of bit error condition. An error flag bit is set when there has been a bit error detected by the bit monitor in TED (TXRX Error Detector sub-module). A bit error indicates that a collision has happened on the LIN bus, for example, the bit value that is monitored is different from the bit value that is sent. When bit error is detected the transmission is aborted no later than the next byte

6.4.7.2 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 re-calculated by the receiver. If the CRC value calculated by the receiver does not match the transmitted CRC value, a CRC error will be flagged. Error response and any necessary software requirements are defined by the system integrator.

6.4.7.3 DCAN Acknowledge Error Detection

When a node on the CAN network receives a transmitted message, it sends an acknowledgment that it received the message successfully. When a transmitted message is not acknowledged by the recipient node, the transmitting DCAN will flag an Acknowledge Error. Error response and any necessary software requirements are defined by the system integrator.

6.4.7.4 DCAN Form Error Detection

Certain types of frames in the DCAN have a fixed format per the CAN protocol. When a receiver receives a bit in one of these frames that violate the protocol, the module will flag a Form Error. Error response and any necessary software requirements are defined by the system integrator.

6.4.7.5 DCAN 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, it automatically inserts a complementary bit into the actual transmitted bit stream. If a 6th consecutive equal bit is detected in a received segment that should have been coded by bit stuffing, the DCAN module will flag a Stuff Error. Error response and any necessary software requirements are defined by the system integrator.

6.4.7.6 PWM Trip by MCAN

PWM can be tripped by the Rx Events on MCAN. It can be used to stop the PWM generation on receiving the special messages (special identifier for which corresponding Rx Event can be generated). Any message with identifier corresponding to the error message should be 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

6.4.7.7 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 bitstream to be transmitted, it automatically inserts a complementary bit into the actual transmitted bit stream. If a 6th consecutive equal bit is detected in a received segment that should have been coded by bit stuffing, the CAN module will flag a Stuff Error. Error response and any necessary software requirements are defined by the system integrator.

6.4.7.8 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 will flag a Form Error. Error response and any necessary software requirements are defined by the system integrator.

6.4.7.9 MCAN Acknowledge Error Detection

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

6.4.7.10 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 it 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 heart beat.

6.4.7.11 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.

6.4.7.12 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.

6.4.7.13 Interrupt on Message RAM Access Failure

One of the interrupt sources is the MRAF: Message RAM Access Failure in register IR.MRAF. The flag/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 of 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 resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location. The flag is also set when the Tx Handler was 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 has to reset CCCR.ASM.
  - 0 = No Message RAM access failure occurred
  - 1 = Message RAM access failure occurred

6.4.7.14 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 peripheral's diagnostics, such as CAN CRC logic operation which will be indicated by an error interrupt. Refer to the device-specific technical reference manual for more information about EPG.

6.4.7.15 EMIF Access Latency Profiling Using On-Chip Timer

Each EMIF access takes fixed number of cycles for completing an access (read/write) to external memory. Once the access latency is obtained, timer module can be used for profiling data transfers to both asynchronous memories (with and without WAIT/READY handshake) and SDRAM memories.

6.4.7.16 EMIF Access Protection Mechanism

This mechanism provides code protection by preventing unauthorized fetch or writes access thus identifying execution of unauthorized code or unwarranted corruption of external memory contents. The feature enables freedom from interference for the software code and data.

6.4.7.17 EMIF Asynchronous Memory Timeout Protection Mechanism

Asynchronous memories have fixed write and read access timings achieved using wait states. Some memories support handshake in addition to wait states configuration using WAIT/READY signal. Using WAIT/READY signal and timeout counters message delays and hang conditions caused can be detected. An error interrupt will be generated once timeout counters expire and current read/write access will be discarded removing stall to the requested master.

6.4.7.18 I2C Access Latency Profiling Using On-Chip Timer

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

6.4.7.19 Information Redundancy Techniques Including End-to-End Safing

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

In order to provide diagnostic coverage for network elements outside the C2000 MCU (wiring harness, connectors, 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 ensure the correctness of a transmission. These checksums, sequence counter and timeout expectation (or timestamp) are applied 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 communications diagnostics implemented should consider the failure modes and potential mitigating safety measures described in IEC 61784-3:2016 and summarized in IEC 61784-3:2016, Table 1.

6.4.7.20 I2C Data Acknowledge Check

When a node on the I2C network receives a byte (address or data), it 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 will flag NACK. Necessary software requirements are defined by the system integrator. For example a function which needs to transfer 4 bytes of data and can sent CRC as 5th byte. The device software can be designed such that the acknowledge is not provided if the data and CRC doesn’t match.

6.4.7.21 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.

6.4.7.22 Break Error Detection

A 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.

For LIN, this feature is applicable only when the module is working in SCI mode. A SCI break detect condition occurs when the LINRX is low for ten bit periods following a missing stop bit. This action sets the BRKDT flag bit and initiates an interrupt.

UART also can flag break errors. When the receive data input is held low for longer than a full-word transmission time, the BE bit in the UARTDR register is set.

6.4.7.23 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 this, the SCI hardware can flag an error and generate an interrupt to the CPU. This feature must be enabled and configured in software.

The LIN module supports detection of framing error condition. An error flag bit is set when an expected stop bit is not found. In SCI compatible mode, only the first stop bit is checked. The missing stop bit indicates that synchronization with the start bit has been lost and that the character is incorrectly framed. Detection of a framing error generates an error interrupt if the RXERR INT ENA bit is set. LIN module supports feature to verify valid Synch Field. It helps in automatic baud rate adjustment by comparing baud rate and adjust if baud rates differ. If the synch field is not detected within the given tolerances, the inconsistent-synch-field-error (ISFE) flag will be set and an ISFE interrupt will be generated.

UART similarly flags frame errors when received characters are missing a valid stop bit, setting the FE bit in the UARTDR register.

6.4.7.24 Overrun Error Detection

If the SCI RX buffer receives new data before the previous data has been read, the existing data will be 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.

LIN module supports detection of data overrun condition. An error flag bit is set when the transfer of data from receive shift register to receiver data buffer register overwrites unread data already in received data register. Detection of an overrun error also causes the LIN to generate an error interrupt if the SET OE INT bit is one.

For UART, the OE bit of the UARTDR register is set when data is received while the FIFO is full, indicating data loss.

6.4.7.25 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, see 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 best results any tests of the functionality should include the I/O loopback.

6.4.7.26 SPI Data Overrun Detection

If SPI RX buffer receives new data before the previous data has been read, the existing data will be 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.

6.4.7.27 Transmission Redundancy

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

In order to provide diagnostic coverage of device interconnects and EMIF, read back of written data (in case of data writes) and multiple read backs of information (in case of data reads) can be employed.

6.4.7.28 Data Parity Error Detection

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 its 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.

6.4.7.29 LIN Physical Bus Error Detection

LIN module supports detection of Physical Bus Error condition, an error flag is set and interrupt generated. A Physical Bus Error (PBE) is detected by a master 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).

6.4.7.30 LIN No-Response Error Detection

LIN module supports detection of No-Response Error detection. An error flag bit is set and interrupt is generated when there is no response to a master's Header completed within TFRAME_MAX (maximum time length allowed for response). The No-Response Error flag is cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.

6.4.7.31 LIN Checksum Error Detection

LIN module supports detection of checksum error on received data. An error flag bit is set and interrupt is generated when there is checksum error detected by a receiving node. The type of checksum to be used depends on the CIGCR1.CTYPE bit (Classic checksum - compatible with LIN 1.3 slave nodes or Enhanced checksum - compatible with LIN 2.0 and newer slave nodes). The Checksum Error flag is cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.

6.4.7.32 LIN ID Parity Error Detection

LIN module supports detection of parity error on ID field. If parity is enabled, an ID parity error (PE) is detected if any of the two parity bits (even/odd) of the sent ID byte are not equal to the calculated parity on the receiver node. The two parity bits (even/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

6.4.7.33 Communication Access Latency Profiling Using On-Chip Timer

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

6.4.7.34 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 receive shift register to receiver data buffer register overwrites unread data already in received data.
- Receive buffer Underrun – This event indicates that software reads the buffer while it is empty.
- Transmit buffer Overrun – This event occurs if a piece of data is overwritten before it 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 or underrun error occurs and corresponding register bit is enabled.

6.4.7.35 FSI Frame Overrun Detection

The FSI module supports detection of frame overrun event. 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 corresponding register bit is enabled.

6.4.7.36 FSI CRC Framing Checks

FSI module supports detection of CRC framing error condition. A CRC error will be 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.

6.4.7.37 FSI ECC Framing Checks

The FSI module supports detection of ECC framing error condition. It supports 16-bit or 32-bit ECC computation module in both the 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 it to be 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

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

6.4.7.38 FSI Frame Watchdog

FSI module supports detection of Frame Watchdog Timeout event. 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 will start 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 will time out and this event will be generated. If this event occurs, the receiver must undergo a soft reset and subsequent resynchronization in order to ensure proper operation. When this condition occurs, a flag is set and an interrupt is generated if enabled.

6.4.7.39 FSI RX Ping Watchdog

FSI module supports detection of RX Ping Watchdog Timeout event. 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 corresponding register bit is enabled. This bit will be set when the ping counter has timed out. An interrupt is generated if 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.

6.4.7.40 FSI Tag Monitor

FSI module supports Tag field in the transfer frame. It contains 4-bit FRAME_TAG field of the last successfully received frame. FSI Tag Monitor checks has to be implemented in software. Tag field for each of the frame on the receive side can be monitored through software and verified against expected values. In addition to FRAME_TAG, FSI module supports the user data as 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.

6.4.7.41 FSI Frame Type Error Detection

FSI module supports detection of Frame Type Error. 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 in order to ensure proper operation. An interrupt is generated if corresponding register bit is enabled.

6.4.7.42 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 in order to ensure proper operation. An interrupt is generated if corresponding register bit is enabled.

6.4.7.43 FSI Register Protection Mechanisms

As a fault avoidance safety measure for key registers of FSI module, registers are protected by EALLOW privilege, register keys, and a master register lock. These protections ensure that no spurious writes or unintentional modifications to these registers are avoided. Some bits in the FSI registers are protected by a key. In order to write to these bits, the key must be written at the same time.

The control register lock will prevent 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.

6.4.7.44 PMBus Protocol CRC in Message

PMBus module supports detection of data corruption during transfer using Packet Error Check (PEC) value feature. When this feature is enabled, it 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.

6.4.7.45 PMBus Clock Timeout

PMBus module support detection of stuck fault on clock (SCL) pin. If the SCL pin is stuck during communication to either High or Low value for duration more than programmed value (in PMBTIMHIGHTIMOUT and PMBTIMLOWTIMOUT Registers), an interrupt is generated and respective Flags are set in PMBSTS status register.

6.4.7.46 EtherCAT MDIO Command Error Indication

An incorrect MDIO access on the Management Interface is flagged as an error in the status register. This can be used to catch faults in the MDIO access filter.

6.4.7.47 EtherCAT Sync-Manager

Sync-manager is used to protect memory data from being mismanaged while being shared between host and External Master by using either a buffer or a mailbox scheme. This makes sure that each master can know whether the other master is sharing the latest information, and also prevent corruption by assigning a single write master. This SM can be used to protect against faults causing inadvertent access.

6.4.7.48 EtherCAT Working Counter Error Indication

Any successful access commanded on the EtherCAT frame increments the working counter which is added to the frame and passed down the network. In a system, this working counter ensures that the access has been successfully completed by the previous slave. This can detect faults in access filter as well as memory/register interfaces.

6.4.7.49 EtherCAT Frame Error Indication

Any illegal frame on the ECAT can be logged as a frame error in a status register.

6.4.7.50 EtherCAT Physical Layer Error Indication

Any Rx errors in the Physical layer can be detected and logged in a status register.

6.4.7.51 PDI Timeout Error Indication

A PDI access can be limited for time using the TIMEOUT error which generates an interrupt, if the PDI access doesn’t complete in stipulated time. This can detect faults in the PDI interface.

6.4.7.52 EtherCAT EEPROM CRC Error Indication

EEPROM accesses using I2C are qualified with a CRC which can detect faults in Tx/Rx of the EEPROM message. This can detect faults in the EEPROM management and loopback controls.

6.4.7.53 EtherCAT EEPROM Not Done Error Indication

Initial EEPROM loading is mandatory for access to RAM. If due to fault, this loading is not successful, the EEPROM not done status is captured. This can detect faults in the EEPROM management and loopback controls.

6.4.7.54 EtherCAT Data Link Error Indication

ECAT communication establishment with PHY is marked by Data Link status, which is captured in a status register. This can detect faults in the PHY management interface.

6.4.7.55 EtherCAT Phy Link Error Indication

PHY LINK errors which is detected by link signal is marked using the PHY link status, which is captured in a status register. This can detect faults in the PHY management interface.

6.4.7.56 Sync, GPO Monitoring Using External Monitor

SYNC signals are generated using two outputs 0/1, and 32 GPO signals are available. These can be used redundantly or can be monitored for precision using an external monitor.

6.4.7.57 EtherCAT Enhanced Link Detection With LED

A monitor checks for PHY link errors, and if there are more than 32 errors in 10us, the link is automatically pulled down and reflected in a status register and in LED output signals. This can be used to protect against faults in the PHY link interface.

6.4.7.58 HW Redundancy of GPIO, FMMU, Sync Manager and SYNC OUT

Multiple GPIO lines, 2 Sync signals, 8FMMUs and 8 Sync Managers are available in ECAT. These can be used in half capacity, and the other half can be configured redundantly by the host. Any transaction via these signals should be compared each time they are consumed.

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*, webpage.
3. JEDEC, *Handling, Packing, Shipping and Use of Moisture/Reflow Sensitive Surface Mount Devices*, webpage.
4. *IEC61508 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*, International Electrotechnical Commission, 1998.
5. ISO 26262–Road Vehicles-Functional Safety, International Standard ISO, vol. 26262, 2018.
6. Standardized E-Gas Monitoring Concept for Gasoline and Diesel Engine Control Units
7. J. Astruc and N. Becker, *Toward the Application of ISO 26262 for Real-Life Embedded Mechatronic Systems*, in International Conference on Embedded Real Time Software and Systems. ERTS2, 2010.
8. *ISO26262–Road Vehicles-Functional Safety, Part 5: Product development at the hardware level, Appendix D*, International Standard ISO, 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, *TMS320F28P65x Real-Time Microcontrollers*, data sheet.
14. Texas Instruments, *TMS320F28P65x Real-Time Microcontrollers Technical Reference Manual*, technical reference manual.
15. Texas Instruments, *C2000™ Hardware Built-In Self-Test*, application note.
16. IEC, IEC-60730, webpage.
17. IEC, IEC-61784, webpage.
18. Texas Instruments, *Certification for Functional Safety Hardware Process*, certificate.

A Summary of Safety Features and Diagnostics

Table A-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.
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 A-2. Summary of Safety Features and Diagnostic

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Analog-to-Digital Converter (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
	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	System defined	On demand (Software defined)	Software defined	Software defined
	ADC10	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ADC11	Disabling Unused Sources of SOC Inputs to ADC	Fault avoidance	Software	Continuous - Enabled by software	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	ADC15	Hardware Redundancy of ADC Safety Checker	Test for diagnostic	Hardware	On demand (Software defined)	Software defined	Software defined	Software defined
	ADC16	Hardware Redundancy with ADC Safety Checker	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Software defined	Software defined
Advanced Encryption Standard (AES)	AES1	Decryption of Encrypted Data Output Using Same KEY and IV	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	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	NA (Fault avoidance technique)	NA (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
APLLController Area Network (DCAN)	APLL1	Clock Integrity Check Using DCC	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Interrupt to CPU	Typically less than 1us 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 Clock Monitoring via XCLKOUT	Diagnostic	System	System defined	System defined	System defined	System defined
	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	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	CAN1	Software Test of Function Using I/O Loopback	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAN2	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAN3	SRAM Parity	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CAN4	Software Test of SRAM	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAN5	Bit Multiplexing in SRAM Memory Array	Fault avoidance	Hardware	Continuous - Enabled at reset	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	CAN7	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAN8	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAN9	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAN10	DCAN Stuff Error Detection	Diagnostic	Hardware	Continuous - Enabled at reset	zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CAN11	DCAN Form Error Detection	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CAN12	DCAN Acknowledge Error Detection	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	CAN13	Bit Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CAN14	CRC in Message	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CAN15	Software Test of Parity Logic	Test for diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAN16	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAN17	Software Test of Function Including Error Tests Using EPG	Diagnostic	Hardware - 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	Software	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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Control Law Accelerator (CLA)	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 <1 μS 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 (multiple execution)	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
	CLA9	Access Protection Mechanism for Memories	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CLA11	Disabling of Unused CLA Task Trigger Sources	Fault avoidance	Software	Continuous - Enabled by software	Zero or very low overhead	NA (Fault avoidance technique)	NA (Fault avoidance technique)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Configurable Logic Block (CLB)	CLB1	Software Test of Function Including Error Tests - CLB Specific	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	NA (Fault avoidance)	NA (Fault avoidance Technique)	NA (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	Diagnostic	Hardware	Continuous - Enabled At reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Clock	CLK1	Missing Clock Detect (MCD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion Clock switch to internal oscillator	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 Clock Monitoring via XCLKOUT	Diagnostic	System	System defined	System 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
	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)	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	CLK17	Dual Clock Comparator (DCC) – Type 2	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Interrupt to CPU	Typically less than 1us to notify *(interrupt Handling Time is System Load and Software Dependent.

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
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	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	CMPSS7	VDAC Conversion by ADC	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CMPSS8	CMPSS Ramp Generator Functionality Check	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
C28x Central Processing Unit (CPU)	CPU1	Reciprocal Comparison by Software	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU2	CPU Hardware Built-In Self-Test (HWBIST)	Diagnostic	Hardware-Software	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	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 <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	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 <1 µS 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
	CPU11	CPU Hardware Built-In Self-Test (HWBIST) Auto Coverage	Test for diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU12	CPU Hardware Built-In Self-Test (HWBIST) Fault Injection Capability	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU13	CPU Hardware Built-In Self-Test (HWBIST) Timeout Feature	Test for diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1µS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	CPU14	Stack Overflow Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU15	VCRC Auto Coverage	Test for diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Software defined	Software defined
	CPU16	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	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
	CPU21	Hardware Redundancy Using Lockstep Compare Module (LCM)	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU22	Self-test Logic for LCM	Test for diagnostic	Hardware	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion	Typically < 1µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU23	LCM Compare Error Forcing Mode	Test for diagnostic	Hardware	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion	Typically < 1µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU24	LCM MMR Parity	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU25	Test of LCM MMR Parity	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	CPU26	Lockstep Self-test Mux Select Logic Fault Detection	Test for diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically < 1µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU27	Redundancy in LCM Comparator	Test for diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
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	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (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 A-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Dual-Zone Code Security Module (DCSM)	DCSM1	Multibit Enable Keys for Control Registers	Fault avoidance	Hardware	Continuous - Enabled at reset	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (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 <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	DCSM8	VCRC 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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Direct Memory Access (DMA)	DMA2	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software Defined	Software defined
	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 <1 µS 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 <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	DMA9	Disabling of Unused DMA Trigger Sources	Fault avoidance	Software	Software defined	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	DMA11	Hardware Redundancy Using Lockstep Compare Module (LCM)	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	DMA12	Self-test Logic for LCM	Test for diagnostic	Hardware	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion	Typically <1µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	DMA13	LCM Compare Error Forcing Mode	Test for diagnostic	Hardware	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	DMA14	LCM MMR Parity	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	DMA15	Test of LCM MMR Parity	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	DMA16	Lockstep Self-test Mux Select Logic Fault Detection	Test for diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	DMA17	Redundancy in LCM Comparator	Test for diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Ethernet for Control Automation Technology (ECAT)	ECAT1	Software Test of Function Using I/O Loopback	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ECAT2	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ECAT3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ECAT4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ECAT5	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ECAT6	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 1us to notify *(Interrupt Handling Time is System Load and Software Dependent)
	ECAT7	Redundant Parity Engine	Test for 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 1us to notify *(Interrupt Handling Time is System Load and Software Dependent)
	ECAT8	Software Test of Parity Logic	Test for diagnostic	Hardware-Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ECAT9	Software Test of SRAM	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ECAT10	EtherCAT MDIO Command Error Indication	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Setting of status flag	Software defined
	ECAT11	EtherCAT Sync-Manager	Fault avoidance	Hardware	Continuous - Enabled by software	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	ECAT12	EtherCAT Working Counter Error Indication	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	System defined	System defined
	ECAT13	EtherCAT Frame Error Indication	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Setting of status flag	Software defined
	ECAT14	EtherCAT Physical Layer Error Indication	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Setting of status flag	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	ECAT15	CRC in Message	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Setting of status flag	Software defined
	ECAT16	PDI Timeout Error Indication	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1us to notify *(Interrupt Handling Time is System Load and Software Dependent)
	ECAT17	EtherCAT EEPROM CRC Error Indication	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Setting of status flag	Software defined
	ECAT18	EtherCAT EEPROM Not Done Error Indication	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Setting of status flag	Software defined
	ECAT19	EtherCAT Data Link Error Indication	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Setting of status flag	Software defined
	ECAT20	EtherCAT Phy Link Error Indication	Diagnostic	Hardware	Continuous - Enabled at reset	System defined	System defined	System defined
	ECAT21	Sync, GPO Monitoring Using External Monitor	Diagnostic	System	System defined	System defined	System defined	System defined
	ECAT22	EtherCAT Enhanced Link Detection With LED	Diagnostic	Hardware	Continuous - Enabled at reset	System defined	System defined	System defined
	ECAT23	HW Redundancy of GPIO, FMMU, Sync Manager and SYNC OUT	Diagnostic	Hardware	Continuous - Enabled at reset	System defined	System defined	System defined
	ECAT24	Bit Multiplexing in SRAM Memory Array	Fault avoidance	Hardware	Continuous - Enabled at reset	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
External Memory Interface (EMIF)	EMIF1	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	EMIF2	VCRC Check of Static Memory Contents	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	EMIF3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	EMIF4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	EMIF5	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	EMIF6	EMIF Access Protection Mechanism	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically < 1µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	EMIF7	EMIF Asynchronous Memory Timeout Protection Mechanism	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	EMIF8	EMIF Access Latency Profiling Using On-Chip Timer	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	EMIF9	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Flash	NWFLASH1	Flash 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 <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	NWFLASH2	Flash Program Verify and Erase Verify Check	Diagnostic	Hardware	Continuous - Enabled at reset	1-2000 µS	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	NWFLASH3	Flash Program/Erase Protection	Fault avoidance	Hardware	Continuous - Enabled by software	Zero or very low overhead	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	NWFLASH4	Flash Wrapper Error and Status Reporting	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	NWFLASH5	VCRC 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	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	NWFLASH7	On-demand Software Program Verify and Blank Check	Diagnostic	Hardware - Software	On demand (Software defined)	1-2µS	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	NWFLASH8	Software Read Back 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	NA (Fault avoidance technique)	NA (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	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	NWFLASH12	Bit Multiplexing in Flash Memory Array	Fault avoidance	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	NWFLASH13	Auto ECC Generation Override	Test for diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	NA	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)	Software defined	Software defined	Software defined
	NWFLASH16	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
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 1us 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 1us to notify *(Interrupt Handling Time is System Load and Software Dependent)
	FSI8	FSI CRC Framing Checks	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically less than 1us 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 1us 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 1us to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	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 1us 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 1us 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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
General-Purpose I/O and Multiplexing (GPIO and PINMUX)	GPIO1	Lock Mechanism for Control Registers	Fault avoidance	Hardware	Continuous - Enabled by software	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	GPIO2	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
High-Resolution Capture (HRCAP)	HRCAP1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	HRCAP2	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	HRCAP3	Monitoring of HRPWM by HRCAP	Test for diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	HRCAP4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	HRCAP5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	HRCAP7	HRCAP calibration logic test feature	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
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 A-2. Summary of Safety Features and Diagnostic (continued)

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Device Interconnect	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 <1 μS 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 <1 μS 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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Inter-Processor Communication (IPC)	IPC1	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	IPC2	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	IPC3	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	IPC4	Event Timestamping Using IPC Counter	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	IPC5	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	IPC6	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	IPC7	IPC 64-Bit Counter Value Plausibility Check	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Debug logic	JTAG1	Hardware Disable of JTAG Port	Fault avoidance	System	Continuous - Enabled at reset	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	JTAG2	Lockout of JTAG Access Using OTP	Fault avoidance	Hardware	Continuous - Enabled at reset	NA (Fault Avoidance)	NA (Fault avoidance technique)	NA (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
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
	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	Overrun Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	LIN8	Frame Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 μS 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 <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	LIN10	LIN No-Response Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS 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 <1 µS 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 <1 µS 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 <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	LIN15	Break Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Controller Area Network (MCAN, CAN-FD)	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
	MCAN8	SRAM ECC	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically less than 1us 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	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (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 1us 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 1us to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	MCAN12	MCAN Acknowledge Error Detection	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically less than 1us 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 1us 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 1us 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 1us 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	Interrupt to CPU	Typically less than 1us 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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
High Resolution Pulse Width Modulator (HRPWM)	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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Enhanced Peripheral Interrupt Expander (ePIE)	PIE1	PIE Double SRAM Hardware Comparison	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	CPU exception for single core device, NMI with ERRORSTS assertion for dual core device	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	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
	PIE6	PIE Double SRAM Comparison Check	Test for 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 <1 μS 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
	PIE9	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PIE13	Hardware Redundancy Using Lockstep Compare Module (LCM)	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	PIE14	Self-test Logic for LCM	Test for diagnostic	Hardware	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	PIE15	LCM Compare Error Forcing Mode	Test for diagnostic	Hardware	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	PIE16	LCM MMR Parity	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	PIE17	Test of LCM MMR Parity	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	PIE18	Lockstep Self-test Mux Select Logic Fault Detection	Test for diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	PIE19	Redundancy in LCM Comparator	Test for diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Power Management Bus Module (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 <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
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	Diagnostic	Hardware	Continuous - Enabled by software	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (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
	PWM15	Online MINMAX Monitoring of TRIP Events	Diagnostic	Hardware - Software	On demand (Software defined)	Zero or very low overhead	Software defined	Software defined
	PWM16	Fault Avoidance Using Minimum Dead Band	Fault avoidance	Hardware - Software	On demand (Software defined)	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	PWM17	Fault Avoidance Using Illegal Combo Logic	Fault avoidance	Hardware - Software	On demand (Software defined)	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	PWM18	Diode Emulation Mode Monitoring	Diagnostic	Hardware - Software	On demand (Software defined)	Zero or very low overhead	Software defined	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Power Supply	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
	PWR4	Brownout Reset (BOR)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset	Typically <1 μS

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
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 <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	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 Mechanisms	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 <1 µS 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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
ROM	ROM1	VCRC 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 <1 μS 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
	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)	Zero or very low overhead	Software defined	Software defined
	ROM13	Background CRC	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM14	Watchdog for Background CRC	Test for diagnostic	Hardware - Software	On demand (Software defined)	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 <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
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)	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	RST4	Glitch Filtering on Reset Pins	Fault avoidance	Hardware	Continuous - Enabled at reset	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	RST5	NMIWD Shadow Registers	Fault avoidance	Hardware - Software	On demand (Software defined)	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (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)	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	RST10	Software Test of Reset – Type 1	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
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 <1 µS 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	Overrun Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SCI5	Break Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SCI6	Frame Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS 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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Sigma Delta Filter Module (SDFM)	SDFM1	SDFM Comparator Filter for Online Monitoring	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SDFM2	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SDFM3	SD Modulator Clock Fail Detection Mechanism	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SDFM4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SDFM5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SDFM6	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SDFM7	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
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
	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 <1 µS 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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
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 <1 µS 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 <1 µS 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	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (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 <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SRAM8	VCRC 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 <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SRAM12	Lock Mechanism for Control Registers	Fault avoidance	Hardware	Continuous - Enabled by software	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	SRAM13	Software Test of ECC Logic	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	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 <1 µS 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 <1µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SRAM21	Memory Power-On Self-Test (MPOST)	Diagnostic	Hardware	Boot up (enabled by default)	Software defined	Software defined	Software defined
	SRAM24	Background CRC	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM25	Watchdog for Background CRC	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
System Control Module and Configuration Registers	SYS1	Multibit Enable Keys for Control Registers	Fault avoidance	Hardware	Continuous - Enabled at reset	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	SYS2	Lock Mechanism for Control Registers	Fault avoidance	Hardware	Continuous - Enabled by software	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (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	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SYS6	Peripheral Clock Gating (PCLKCR)	Fault avoidance	Hardware - Software	On demand (Software defined)	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	SYS7	Peripheral Soft Reset (SOFTPRES)	Fault avoidance	Hardware - Software	On demand (Software defined)	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	SYS8	EALLOW and MEALLOW Protection for Critical Registers	Fault avoidance	Hardware	Continuous - Enabled at reset	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (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	Fault avoidance	Hardware-Software	On demand (Software defined)	NA (Fault avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Cross Bar (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 XBAR Flag	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
Universal Asynchronous Receiver/ Transmitter (UART)	UART1	Software Test of Function Using I/O Loopback	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	UART2	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	UART3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	UART4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	UART5	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	UART6	Parity in Message	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	UART7	Overrun Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	UART8	Break Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	UART9	Frame Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
External Interrupt (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

B 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.

B.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 here.

B.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 B-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
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

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

Functional Safety Life Cycle Activity	TI Execution	System Integrator Execution
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

B.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 B-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.