User's Guide

Functional Safety Manual for TMS320F28004x

ABSTRACT

This document is a functional safety manual for the Texas Instruments TMS320F28004x safety critical microcontroller product family. The product family utilizes a common safety architecture that is implemented in multiple application-focused products.

Table of Contents

List of Figures

List of Tables

Trademarks

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

1 Introduction

> **WARNING**
> The TMS320F28004x is being offered as a Functional Safety-Compliant Safety Element out of Context (SEooC) product. This implies that TMS320F28004x was developed in compliance with TI's ISO 9001/IATF 16949 compliant hardware product development process. Subsequently, this product was independently assessed to meet a systematic capability compliance of ASIL D (according to ISO 26262:2018) and SIL 3 (according to IEC 61508:2010), see the *Texas Instrument's functional safety hardware development process*. As such, this functional safety manual is intended to be informative only to help explain how to use the features of TMS320F28004x 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.

The products supported by this document have been assessed to be meet a systematic capability compliance of ASIL D (according to ISO 26262) and SIL 3 (according to IEC 61508). For more information, see the *Texas Instrument's functional safety hardware development process*.

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

This document is a functional safety manual for the Texas Instruments TMS320F28004x safety critical microcontroller product family. The product family utilizes a common safety architecture that is implemented in multiple application focused products.

Product configurations supported by this functional safety manual include silicon revision B of the following products listed in Table 1-1. The device revision can be determined by the REVID field of the device identification registers outlined in the product data sheet.

Table 1-1. Products Supported by This Functional Safety Manual

Orderable Devices	Supported Safety Integrity Level
F280048CPMQR	ASIL B
F280048PMQR	ASIL B
F280049CPMS	ASIL B
F280049CPZQR	ASIL B
F280049CPZS	ASIL B
F280049PMS	ASIL B
F280049PMSR	ASIL B
F280049PZQ	ASIL B
F280049PZQR	ASIL B
F280049PZS	ASIL B
F280049PZSR	ASIL B
F280040CPMQR	QM
F280040PMQR	QM
F280041CPMS	QM
F280041CPZQR	QM
F280041CPZS	QM
F280041CRSHSR	QM
F280041PMS	QM
F280041PMSR	QM
F280041PZQR	QM
F280041PZS	QM
F280041PZSR	QM
F280041RSHSR	QM

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

Orderable Devices	Supported Safety Integrity Level
F280045PMS	QM
F280045PMSR	QM
F280045PZS	QM
F280045PZSR	QM
F280045RSHSR	QM
F280049CRSHSR	QM
F280049CRSHS	QM
F280049RSHSR	QM

This Functional Safety Manual provides information needed by system developers to assist in the creation of a safety critical system using a supported TMS320F28004x 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 Functional Safety Analysis Report (SAR) for TMS320F28004x C2000™ MCUs*, which is only available under 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 TMS320F28004x 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.

2 TMS320F28004x Product Safety Capability and Constraints

This section summarizes the TMS320F28004x product safety capability. Each TMS320F28004x 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 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 TMS320F28004x 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 TMS320F28004x 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 3.1. 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

Figure 3-1 shows the standard process.

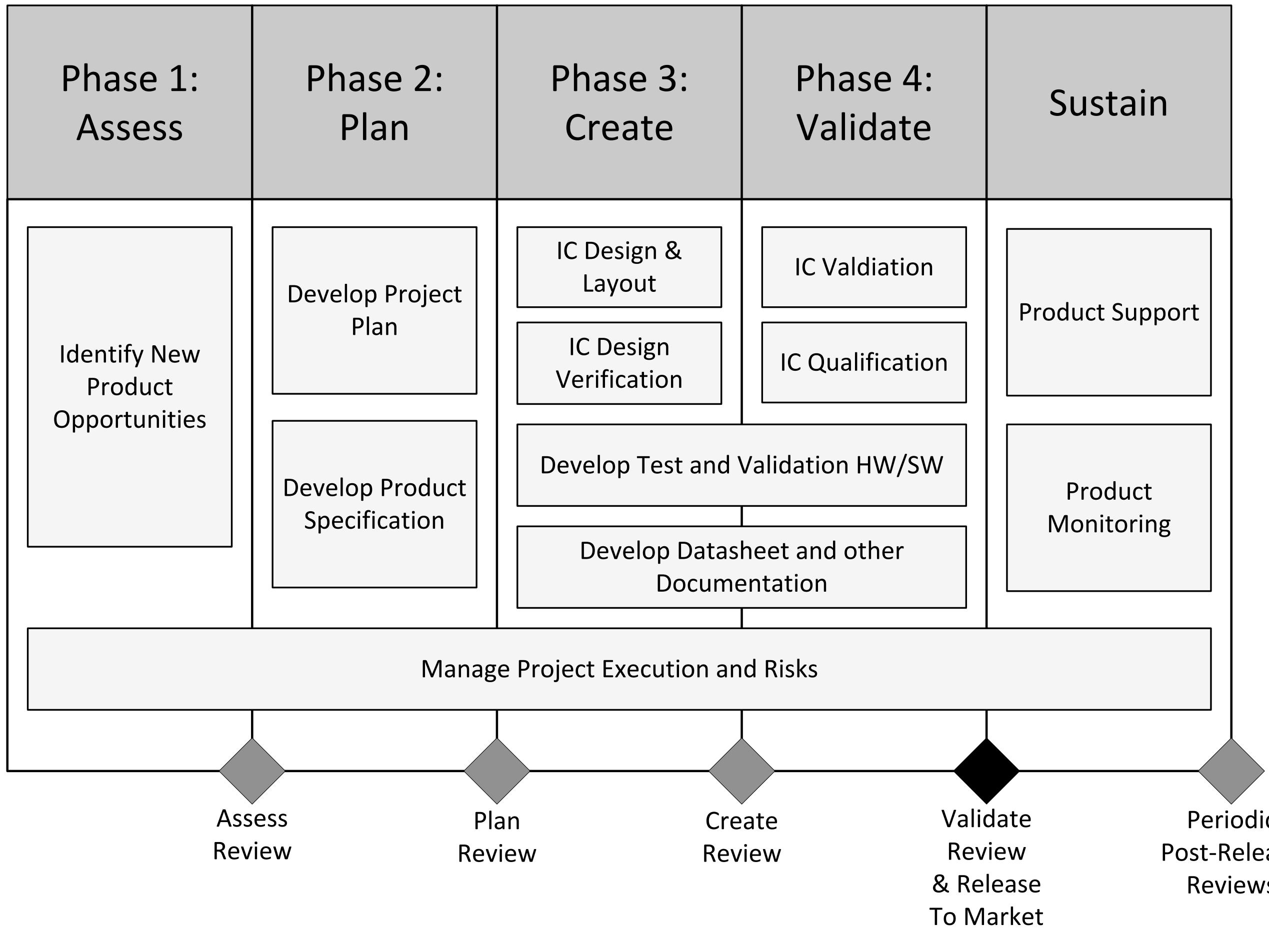


Figure 3-1. TI New-Product Development Process

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 Figure 3-1.

For more information about which functional safety lifecycle 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 (i.e. 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 (i.e. FMEDA)	Release functional safety report	
		Verify the quantitative analysis	End of Phase Audit	
		Iterate functional safety design as necessary		
		End of Phase Audit		

4 TMS320F28004x Product Overview

4.1 C2000 Architecture and Product Overview

The TMS320F28004x devices are powerful 32-bit floating-point microcontroller unit (MCU) designed for advanced closed-loop control applications in automotive and industrial applications.

4.1.1 TMS320F28004x MCU

TMS320F28004x supports C28x and CLA as processing elements that boosts system performance for closed loop control applications. This is a powerful 32-bit floating-point microcontroller unit (MCU) that lets system integrator to access crucial control peripherals, differentiated analog, and nonvolatile memory on a single device.

The C28x CPU is further boosted by the Trigonometric Math Unit (TMU) accelerator that enables fast execution of algorithms with trigonometric operations common in transforms and torque loop calculations. The Viterbi, Complex Math and CRC Unit (VCU) accelerator reduces the time for complex math operations common in encoded applications. Users may refer to *Accelerators: Enhancing the Capabilities of the C2000™ MCU Family* to see how the accelerators 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 frequency as the main C28x CPU, responding to peripheral triggers with minimum event latency and executing code concurrently with the main CPU.

The TMS320F28004x supports up to 256KB (128KW) of on-chip flash memory with error correction code (ECC) and up to 100KB (50KW) of SRAM with parity or ECC.

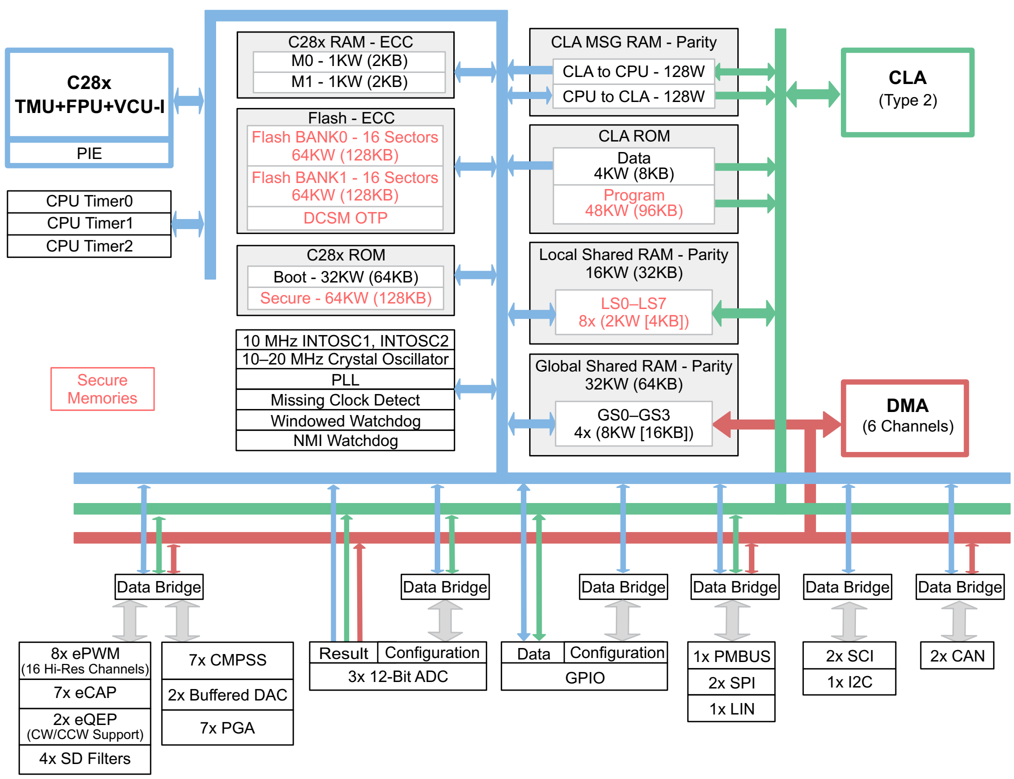


Figure 4-1. Functional Block Diagram of TMS320F28004x MCU

High performance analog and control peripherals are also integrated to further enable system consolidation. Three independent 12-bit ADCs provide precise and efficient management of multiple analog signals, which ultimately boosts system throughput. The new sigma-delta filter module (SDFM) works in conjunction with the sigma-delta modulator to enable isolated current shunt measurements. 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), Enhanced Pulse Width Modulation (ePWM), Enhanced Capture (eCAP), Enhanced Quadrature Encoder Pulse (eQEP) and Programmable Gain Amplifier (PGA). The Programmable Gain Amplifier (PGA) is used to amplify an input voltage for the purpose of increasing the effective resolution of the downstream ADC and CMPSS modules.

Peripherals such as Controller Area Network (CAN) modules (ISO11898-1/CAN 2.0B-compliant), Inter-Integrated Communication (I2C) Bus, Local Interconnect Network (LIN), Serial Communications Interface (SCI), Serial Peripheral Interface (SPI), Power Management Bus (PMBus) Interface, and Fast Serial Interface (FSI) extend connectivity of TMS320F28004x MCU. The Fast Serial Interface (FSI) module is a serial communication peripheral capable of reliable high-speed communication across isolation devices.

The device configurations supported by this functional safety manual for TMS320F28004x MCUs is outlined in the *TMS320F28004x 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.

4.2 Functional Safety Concept

To stay as general as possible, the functional safety concept assumes the MCU playing the role of a processing unit (or part of it) and connected to remote controller(s) by means of a communication bus as shown in Figure 4-2. The communication bus is directly or indirectly connected to sensor(s) and actuator(s).

IEC 61508-1:2010 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 61508:2010 series. A system including TMS320F28004x microcontroller as indicated by Figure 4-2 can be used in a compliant item according to IEC 61508:2010.

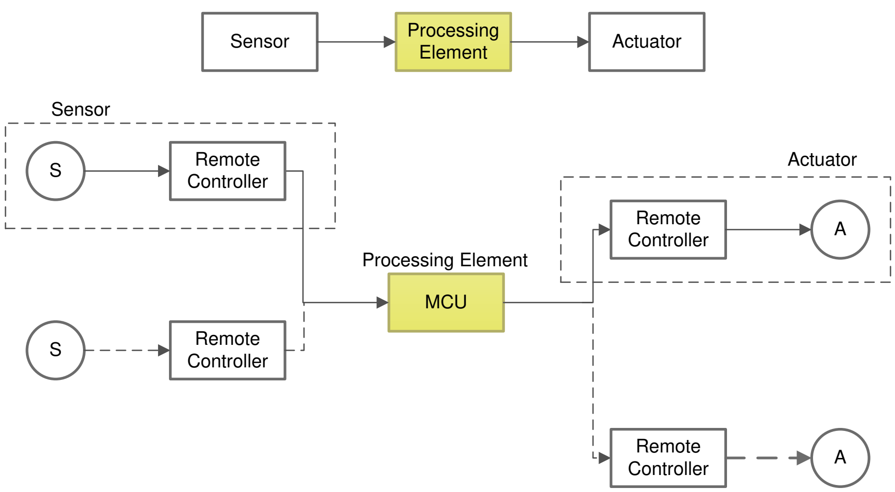


Figure 4-2. Definition of the TMS320F28004x MCU Used in a Compliant Item

4.2.1 VDA E-GAS Monitoring Concept With TMS320F28004x MCU

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 4-3.

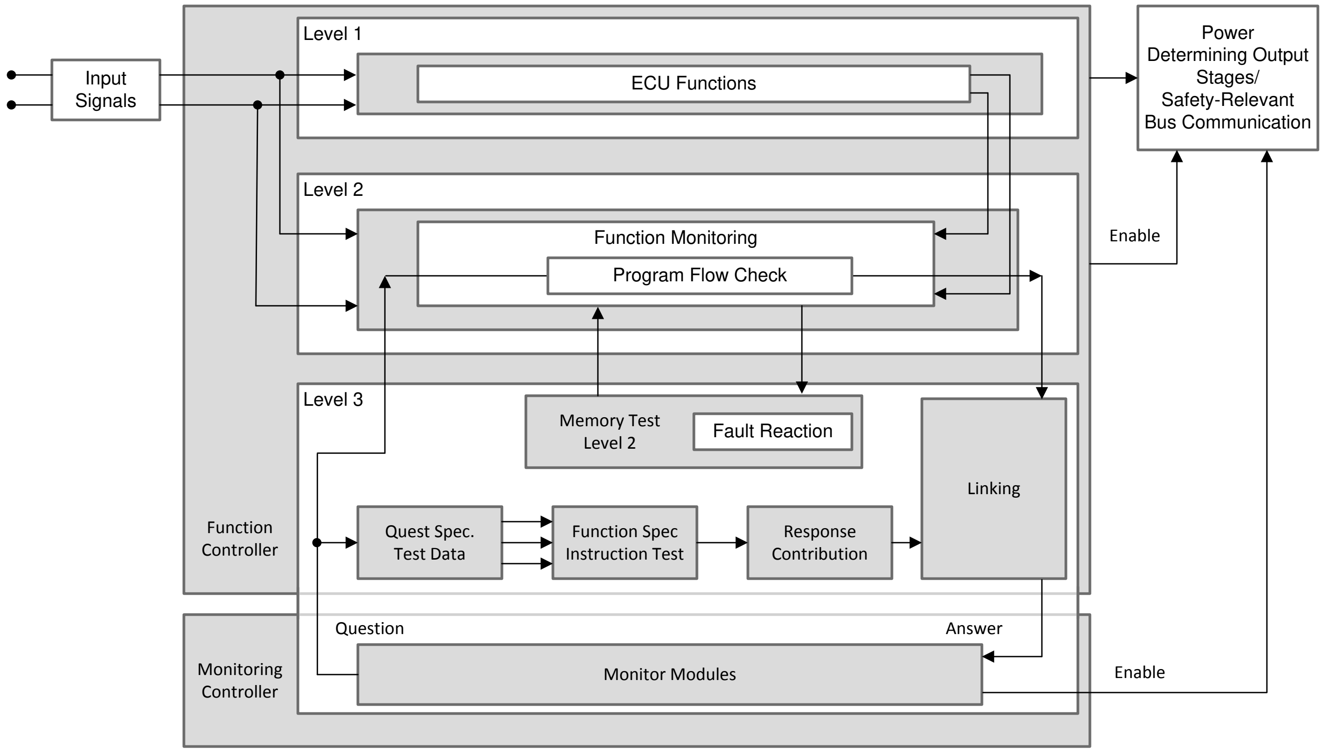


Figure 4-3. E-GAS System Overview From Standard

The 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 4-4. 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.

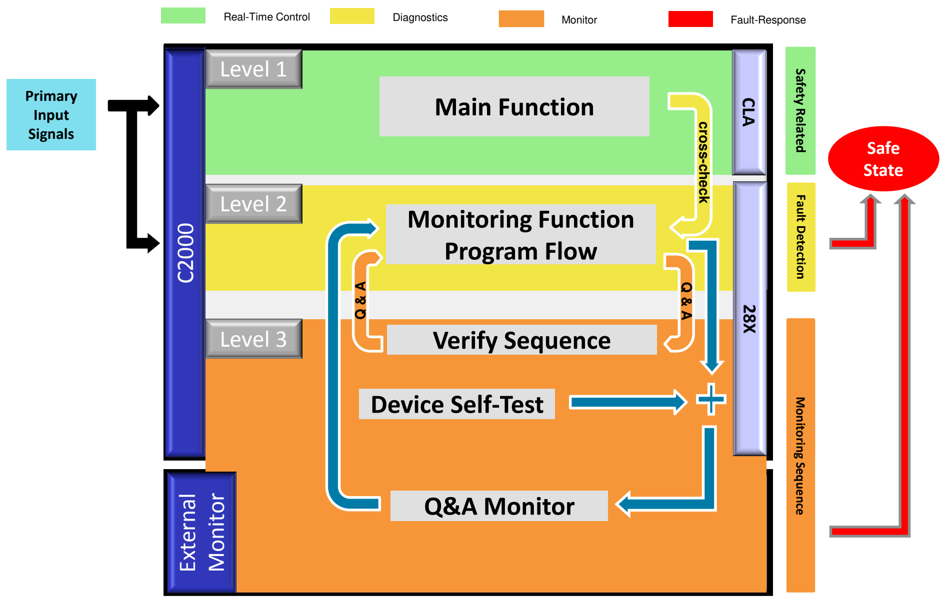


Figure 4-4. VDA E-Gas Monitoring Concept Applied to F28004x MCU

Due to the inherent versatility of the device architecture, several software voting based functional safety configurations are possible. Some of the functional safety configurations possible with TMS320F28004x for improving diagnostic coverage are explained in Table A-1. While implementing these configurations, system integrator needs to consider the potential common mode failures and address them in an appropriate manner. This may suitably be modified to adapt to TMS320F28004x requirements based on the availability of processing units. (As stated earlier, the device claims no hardware fault tolerance, (for example, no claims of HFT > 0), as defined in IEC 61508:2010).

The major safety features of TMS320F28004x are shown in Figure 4-5.

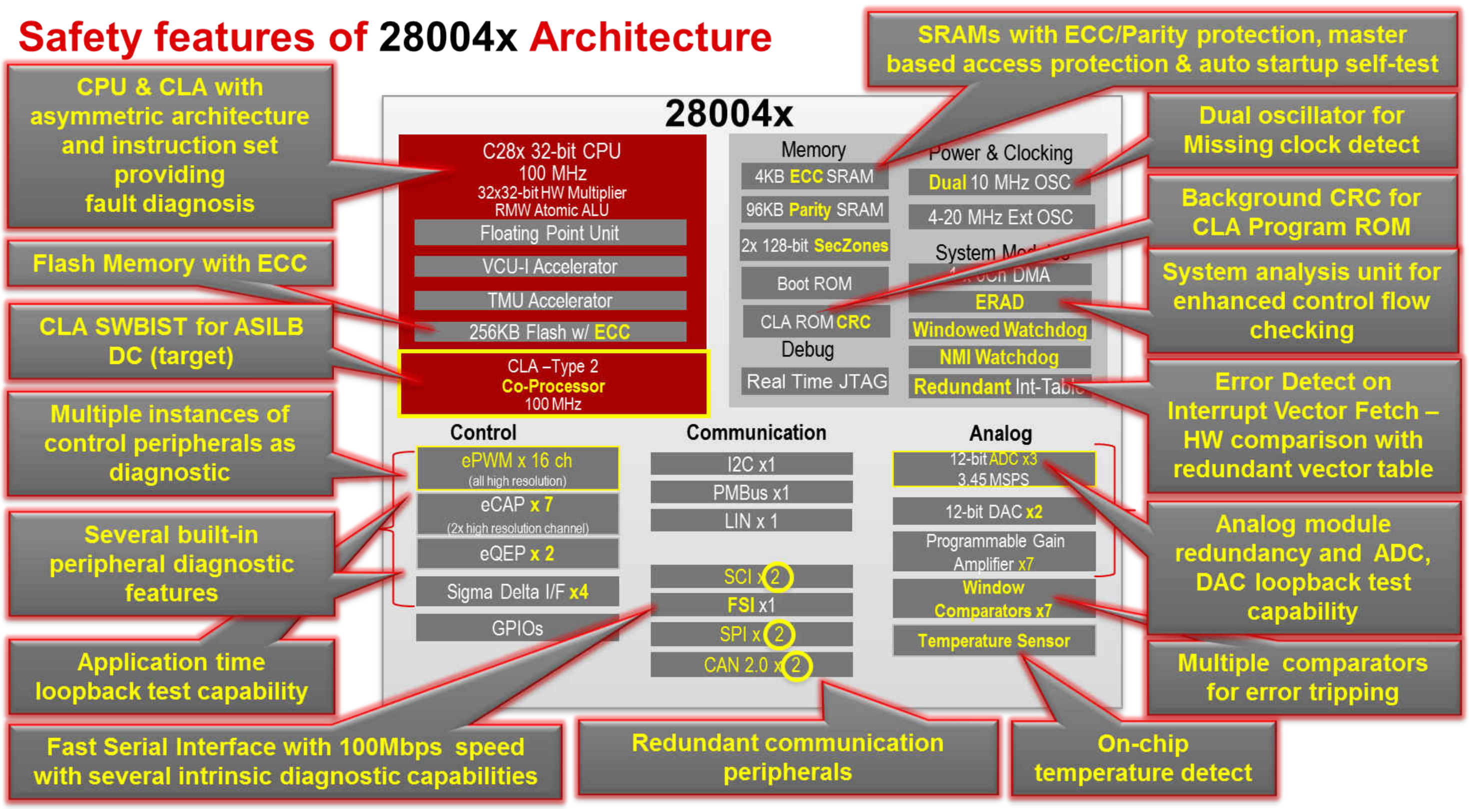


Figure 4-5. TMS320F28004x MCU With Safety Features

4.2.2 Fault Tolerant Time Interval (FTTI)

Various functional safety mechanisms in the devices are either always-on (see CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping, and so forth) or executed periodically (see VCU CRC Check of Static Memory Contents, and so forth) by the application software. The maximum time that a safety mechanism will take to detect a fault is termed as Fault Diagnostic Test Time Interval (FDTI). 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 in 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 ISO 26262. This is similar to Process Safety Time (PST) defined in IEC 61508. Figure 4-6 illustrates the relationship between FDTI, Fault Reaction Time and FTTI.

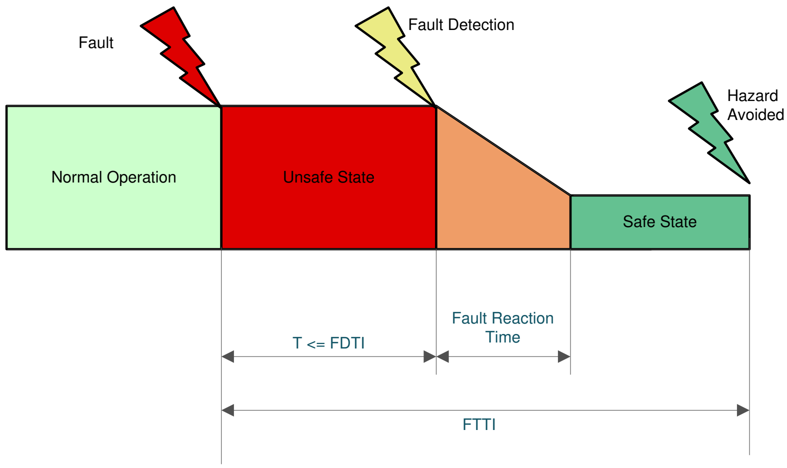


Figure 4-6. Relationship Between FDTI, Fault Reaction Time and FTTI

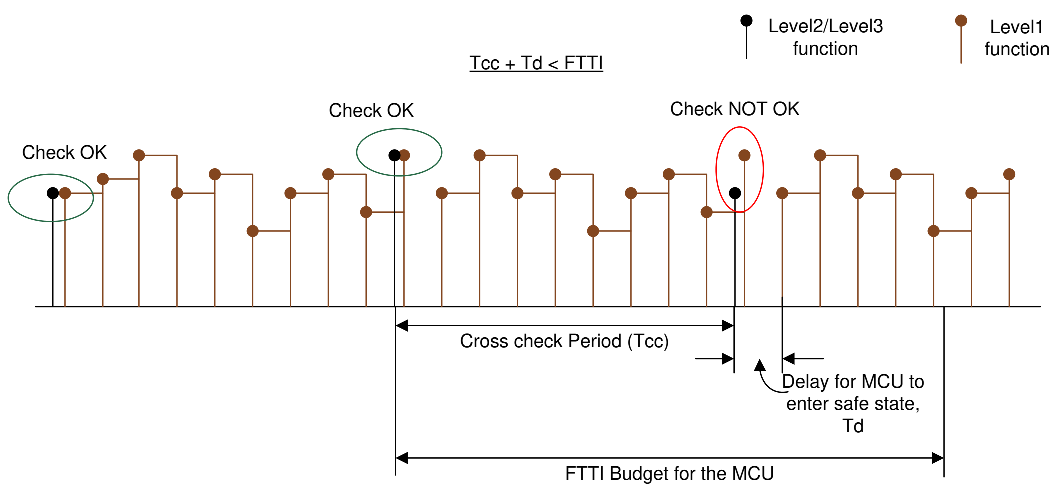


Figure 4-7. Illustration of FTTI

The frequency and extent of each of the Level 2 and Level 3 checks in E-GAS monitoring concept 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 TMS320F28004x 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 TMS320F28004x MCU Safe State

Referring to Figure 4-8, the safe state of the TMS320F28004x MCU is defined as the one in which:

- TMS320F28004x MCU Reset is asserted
- Power supply to TMS320F28004x 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 C2000 MCU's IO pins as a result of Level 2 check failure (for example, ERRORSTS pin is asserted).
- Output of the TMS320F28004x 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 is tri-stated).

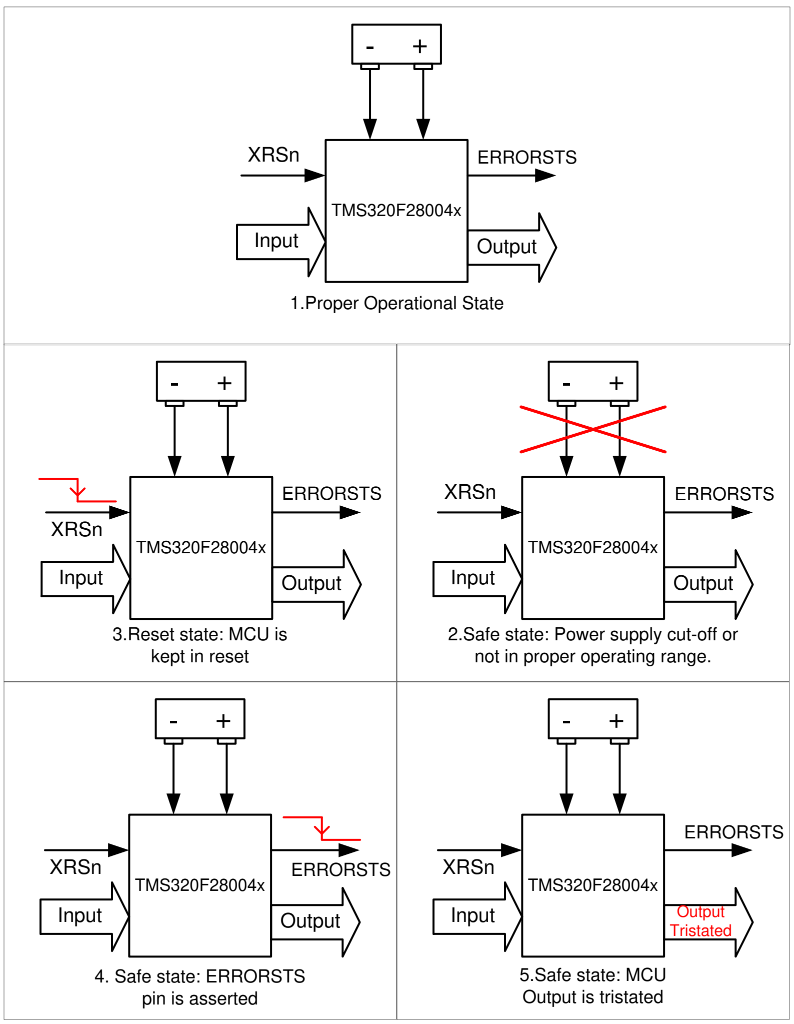


Figure 4-8. TMS320F28004x MCU Safe State Definition

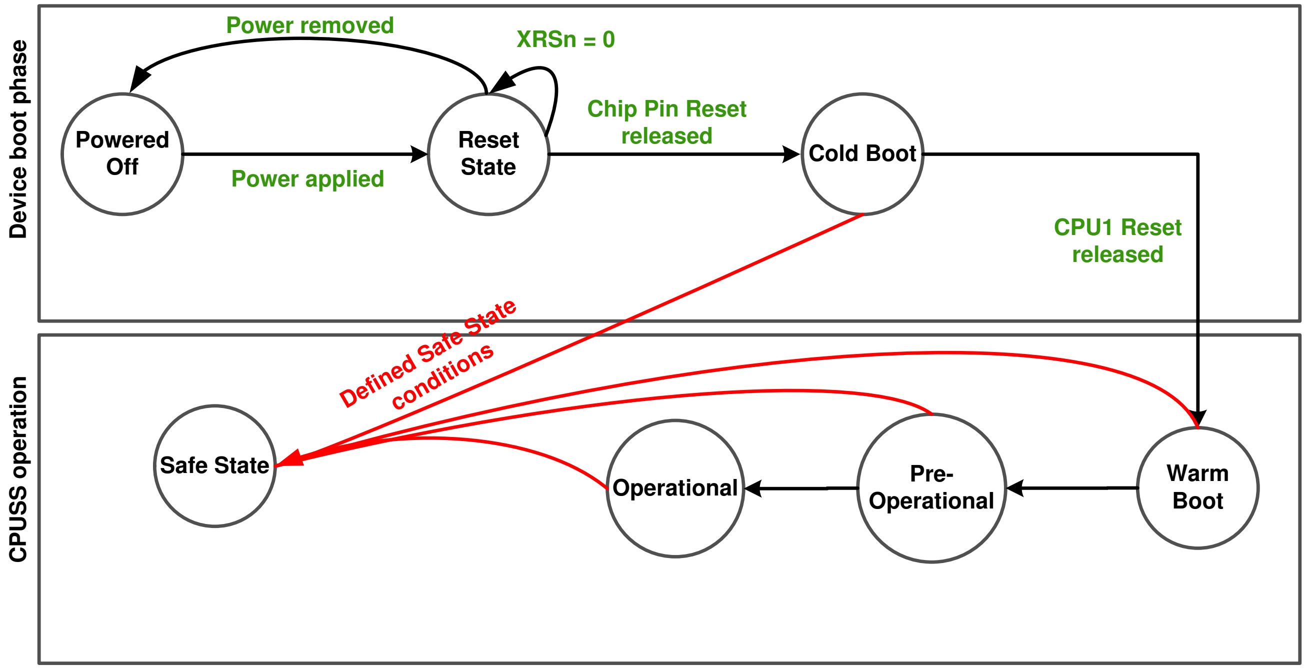


Figure 4-9. TMS320F28004x MCU Device Operating States

4.2.4 Operating States

The C2000 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 CPU Subsystem (CPUSS) operation phase.

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

- Powered Off - This is the initial operating state of TMS320F28004x 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 TMS320F28004x MCU) in any of the TMS320F28004x MCU states as response to a system level fault condition or a fault condition indicated by the TMS320F28004x 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, key analog elements, digital control logic, and debug logic are initialized. The CPU remains powered but in reset. When the cold boot process is completed, the reset of the CPU is internally released, leading to the warm boot stage.
- Warm Boot - The CPU begins execution from Boot ROM during the warm boot stage.
- 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, pinmux, 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. For details, see ROM8-Power-Up Pre-Operational Security.
- 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.

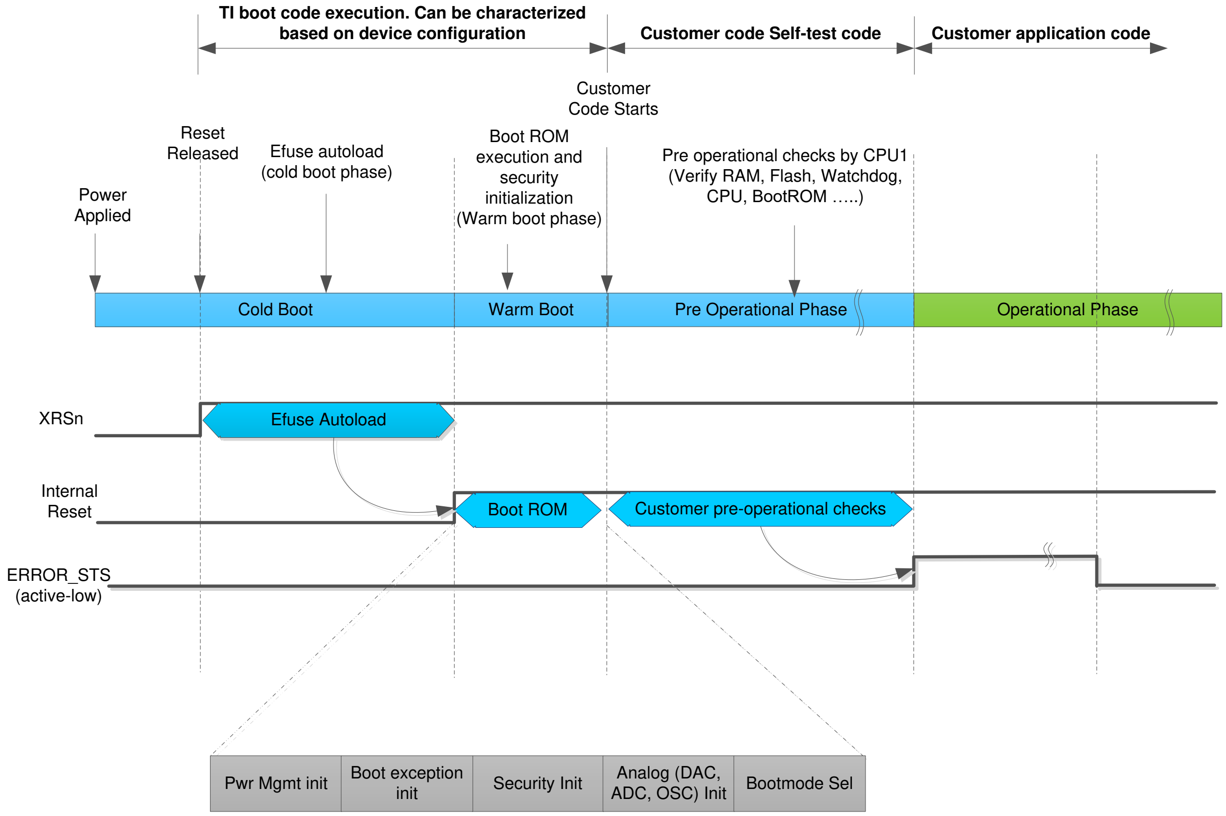
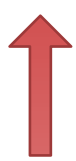


Figure 4-10. TMS320F28004x MCU CPU Start-Up Sequence

4.2.5 Management of Faults

The TMS320F28004x 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 TMS320F28004x 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 4-11.



- Device Powerdown
- Assertion of XRSn pin
- Assertion of CPU Reset
- NMI and assertion of ERRORSTS pin
- CPU Interrupt

Figure 4-11. Fault Response Severity

- Device Powerdown: This is the highest priority fault response where the external component (see Section 4.4.1) detects malfunctioning of the device or other system components and powers down the TMS320F28004x 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 TMS320F28004x 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. Security violations may be the effect of a fault condition.
- 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. The TMS320F28004x MCU 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 TMS320F28004x 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.

4.2.6 Suggestions for Improving Freedom From Interference

The following techniques and safety measures shall be used as applicable for improving independence of function when using the TMS320F28004x 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 (RST9-Peripheral Soft Reset (SOFTPRES)).
3. When possible, separate critical I/O functions by using non adjacent I/O pins/balls.
4. 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.
5. The Dual Code Security Module (DCSM) can be used for functional safety where functions with different safety integrity levels can be executed from different security zones (zone1, zone2, and unsecured zone), acting as firewalls and thus mitigating the risk due to interference from one secure zone to another. For more information, see *Achieving Coexistence of Safety Functions for EV/HEV Using C2000™ MCUs*
6. TMS320F28004x supports SYS10-Peripheral access protection - Type 0. 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 which allow or protect against the access from given master. Each peripheral has two bit qualifier per master to decode the access allowed. For more details, see the PERIPH_AC_REGS Registers in *TMS320F28004x Technical Reference Manual*.
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 will help minimize interference caused by unintentional DMA transfers.
9. CLA11-Disabling of Unused CLA Trigger Sources will mitigate risk of interference caused due to the trigger events.
10. To avoid interference from spurious activity on MCU's debug port, JTAG1-Hardware Disable of JTAG Port can be used.
11. 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.
12. MCU resources in supporting CPU execution such as memory, interrupt controller, and so forth could be impacted by resources from lower safety integrity safety functions coexisting on same MCU. Safety mechanisms such as SRAM11-Access Protection Mechanism for Memories, SRAM16–Information Redundancy Techniques, SRAM17-CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping will be able to detect such interference.
13. 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-Multi-Bit Enable Keys for Control Registers, SYS2-Lock Mechanism for Control Registers, SYS8-EALLOW and MEALLOW Protection for Critical Registers.

4.2.7 Suggestions for Addressing Common Cause Failures

System Integrator needs to execute a common cause failure analysis to consider possible dependent/common cause failures on the sub-elements of the TMS320F28004x MCU, including pin level connections.

1. Consider a relevant list of dependent failure initiators, such as the lists found in ISO 26262-11:2018. Analysis of dependent failures should include common cause failures among functional redundant parts and also between functions and the respective safety mechanisms.
2. Verify that the dependent failure analysis considers the impact of the software tasks running on the TMS320F28004x MCU, including hardware and software interactions.
3. Verify that the dependent failure analysis considers the impact of the pin or ball level interactions on the TMS320F28004x MCU package, including aspects related to the selected I/O multiplexing.

The following should be considered for addressing the common cause failures when using the TMS320F28004x MCU:

1. 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.
2. 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 (MCD), CLK2-Clock Integrity Check Using CPU Timer, CLK5-External Clock Monitoring 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.
3. Failure of common reset signal to redundant functions can be detected by RST1-External Monitoring of Warm Reset (XRSn), RST2-Reset Cause Information.
4. Common cause failure on Interconnect logic could impact both redundant functions and also functional 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.
5. Common cause failure could impact two functions used in a redundant way. In case the of communication peripherals, module specific Information Redundancy Techniques Including End-to-End Safing can be implemented to detect common cause failures, for example, CAN2-Information Redundancy Techniques Including End-to-End Safing, SPI2-Information Redundancy Techniques Including End-to-End Safing, SCI3-Information Redundancy Techniques Including End-to-End Safing, I2C3-Information Redundancy Techniques Including End-to-End Safing.
6. Use different voltage references and SOC trigger sources for ADC (see Section 6.5.8).
7. Use ePWM modules from different sync groups for implementing Hardware Redundancy.
8. Use GPIO pins from different groups when implementing Hardware Redundancy for GPIO pins.
9. It is recommended that two PGA modules used in redundant way to not share same ground pin. Refer to device specific datasheet for details on which PGA's share common ground.

4.3 C2000 Safety Diagnostics Libraries

The diagnostics libraries designed for the F28004x family of devices comprise of three libraries, namely, the CLA_STL, C28x_STL and SDL. These libraries are designed to help TI customers, using the F28004x, develop functionally safe systems that can comply with a wide range of standards for end products in the automotive (ISO 26262), industrial (IEC 61508) and appliance (IEC 60730) markets. The CLA_STL and the C28x_STL implements the CLA2 - Software Test of CLA and CPU3 - Software Test of CPU safety mechanisms and the SDL provides examples for several safety mechanisms provided in the functional safety manual.

Table 4-1. DC and SCC Targeted for F28004x Diagnostic Libraries

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

The CLA_STL and C28x_STL were 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 CLA_STL represents a safety mechanism with the capability to detect permanent faults of the Control Law Accelerator (CLA). The C28x_STL represents a safety mechanism with the capability to detect permanent faults of the C28x CPU. See the SPS delivered with the CSP for the exact DC requirements applicable to each STL product.

The SDL is generally called a Software Diagnostic Library and is an integral part of the overall safety related collateral provided by TI. It 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.

In order to assist customers with getting their own product level certifications, TI has developed an F28004x Compliance Support Package (CSP). The CSP provides documentation, source code, static analysis results, MISRA C compliance results, unit test reports, dynamic analysis results, functional tests and integration examples. The STL (C28x_STL and CLA_STL) libraries and the corresponding source code released in the CSP demonstrate the product of a software development flow that is compliant with ISO 26262 ASIL D systematic capability.

WARNING

In order to maintain the diagnostic coverage, the source code of C28x_STL and CLA_STL must be used as delivered by TI and must not be modified when integrating the libraries into the customer application. Any modification will certainly result in a compromise of the safety goal for the final product, resulting in an unsafe operating environment for the end user. Refer to the Software Delivery Form (SDF) to find the reference MD5 checksums for each of the files corresponding to the STLs. The SDF file is delivered as part of the CSP.

Table 4-2 shows the tools used to develop the F28004x libraries.

Table 4-2. Tools Required for Integration of the F28004x STL

SW/HW/Tool	Version	Dependency
Code Composer Studio	9.2.0.00013	Integrated Development Environment
CGT	20.2.1.LTS	Code Generation Tool Chain (Compiler, Assembler, Linker)
C2000Ware	V3.01.00.00	F28004x Header Files
TMDSCNCD280049C	Rev. A	F280049 controlCARD Information Guide

The system integrator must consult the C28x_STL and CLA_STL user guides for all the details related to installation and development.

The STLs were tested on the F28004x controlCARD.

4.3.1 Assumptions of Use - F28004x 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 F28004x based safety architecture.

The software support for the various safety mechanisms in the F28004x can be divided into the following three categories:

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

A safe product built on the F28004x device hierarchically deploys each of the software solutions provided by TI. The first in the hierarchy is the C28x_STL which detect permanent faults inside the CPU by implementing the CPU3 - Software Test of CPU 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 F28004x device. Lastly, the CLA_STL which implements the CLA2 - Software Test of CLA safety mechanism, 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 C28x_STL first to make sure that the CPU is functioning properly and is capable of performing the required safety operations. The SDL supports safety mechanisms such as: CLK2 - Clock Integrity Check Using CPU Timer, CLK10 - Software Test of Watchdog (WD) Operation, CLK12 - Software Test of Missing Clock Detect Functionality, SRAM14 - Software Test of Parity Logic, SRAM13 - Software Test of ECC Logic, SRAM3 - Software Test of SRAM and several other key processing elements. The system integrator must study all the safety mechanisms supported by the SDL and determine their applicability into the safety system being designed. The safety system must be evaluated with respect to the startup and runtime constraints and whether the software diagnostic tests can be run during POST, PEST or a combination of both.

The successful completion of the software diagnostics, selected by the system integrator, can be used as the qualifier to run the test vectors supported by the CLA_STL.

4.3.2 Operational Details - F28004x Self-Test Libraries

The C28x_STL, SDL and the CLA_STL are co-hosted onto an F28004x target in order to enable the comprehension 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 comprehend safety.

4.3.2.1 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 VCU-CRC functionality. The Viterbi and Complex Math instructions in VCU are not covered by C28x_STL and should not be used for safety related software.

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 a sate. 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.

The system integrator must consult the C28x_STL user guides and understand all aspects of integrating the library into the host application.

4.3.2.2 Operational Details – CLA Self-Test Library

The CLA_STL, implements the CLA2 - Software Test of CLA. Similar to the C28x_STL the CLA_STL's startup tests are also destructive in nature and should be run during startup operations. The CLA_STL's run time tests 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. The CPU must run both the CLA_STL POST and PEST tests for maximum diagnostic coverage. See the SPS delivered with the CSP for the exact DC requirements applicable to the CLA_STL.

The system integrator must consult the CLA_STL user guides and understand all aspects of integrating the library into the host application.

4.3.2.3 Operational Details – SDL

Table 4-3 is a mapping of SDL software modules and APIs to safety features and diagnostic.

Table 4-3. 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	FLASH2
STL_March	SRAM3
STL_OSC_CT	CLK2
STL_OSC_HR	OTTO1, CLK3
STL_PIE_RAM	PIE6
sdl_ex_dcsm_ffi	No unique identifier, demo of freedom from interference using DCSM
sdl_ex_flash_ecc_test	FLASH6
sdl_ex_flash_prefetch_test	FLASH8
sdl_ex_mcd_test	CLK12
sdl_ex_ram_access_protect	SRAM10
sdl_ex_ram_ecc_parity_test	SRAM13, SRAM14
sdl_ex_watchdog	CLK10

4.3.3 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, automotive and functional safety. (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 4-12 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.

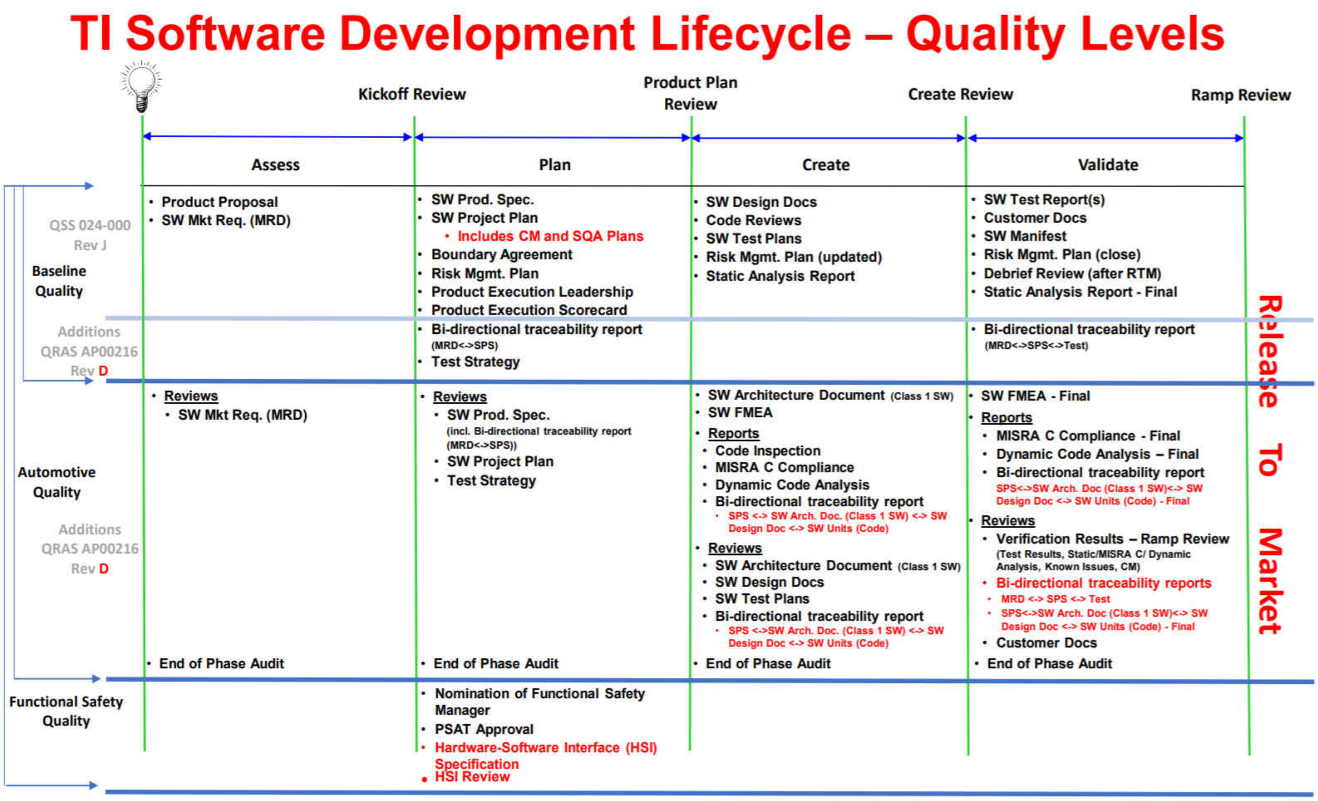


Figure 4-12. TI Software Development Lifecycle - Quality Level

4.3.4 Software Delivery Form (SDF) for STLs

The source code delivered with both C28x_STL and the CLA_STL must be validated with the reference MD5 information provided in SDF file for each STL. A unique MD5 signature applies to each of the source files used to create the Self Test Libraries. Checking the MD5 signature is strongly recommended as a precautionary step to ensure that the source code is exactly the same as what was certified by TÜV SÜD.

In order to ensure that the required diagnostic coverage based metrics are achieved, the source must not modified in anyway and is expected to be used as is. Violating this condition will result in a potential failure in the operation of the CLA_STL and it may not meet the required safety requirements.

The SDF file is delivered as part of the Compliance Support Package (CSP).

4.4 TMS320F28004x MCU Safety Implementation

4.4.1 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 TMS320F28004x MCU
- External Watchdog timer that can be used for diagnostic purposes
- Components required for taking the system to safe state as per the TMS320F28004x MCU safe state defined in Section 4.2.3.

4.4.2 Example Safety Concept Implementation Options on TMS320F28004x MCU

TMS320F28004x class of devices 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 by one of the processing units, utilizing the other processing unit 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 (WD) and so forth, can also be utilized. Software Test of CLA and Software Test of CPU 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. Here are some definitions:

- Intended Function: Control application implemented on TMS320F28004x (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.)
  - Shall meet >= 60% LFM for both permanent faults
- 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 TMS320F28004x.

4.4.2.1 Safety Concept Implementation: Option 1

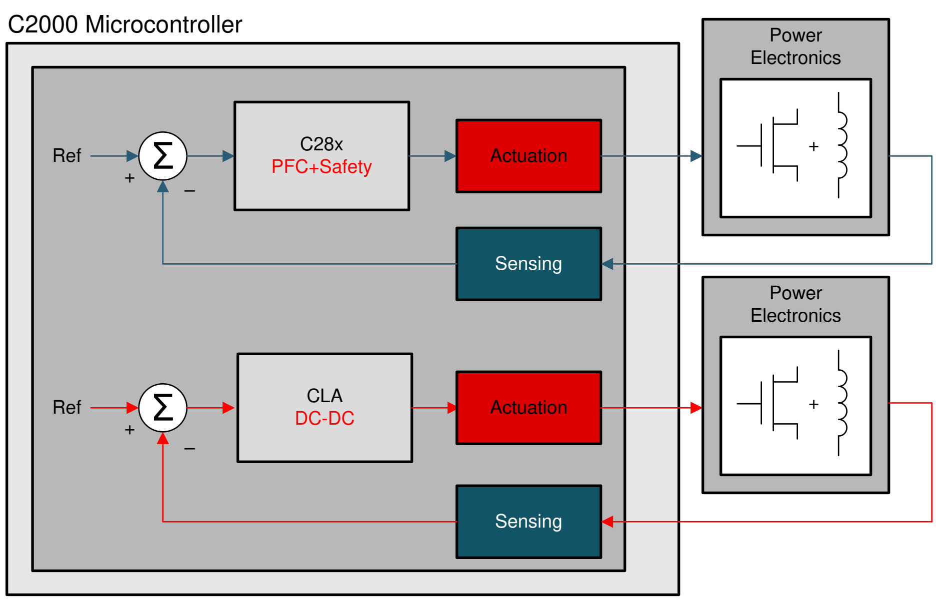


Figure 4-13. Safety Concept Implementation Option 1

- 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
- Diagnostic Function: Implement on the other processing unit.
  - LFM can be met by Software Test of CLA or Software Test of CPU

4.4.2.2 Safety Concept Implementation: Option 2

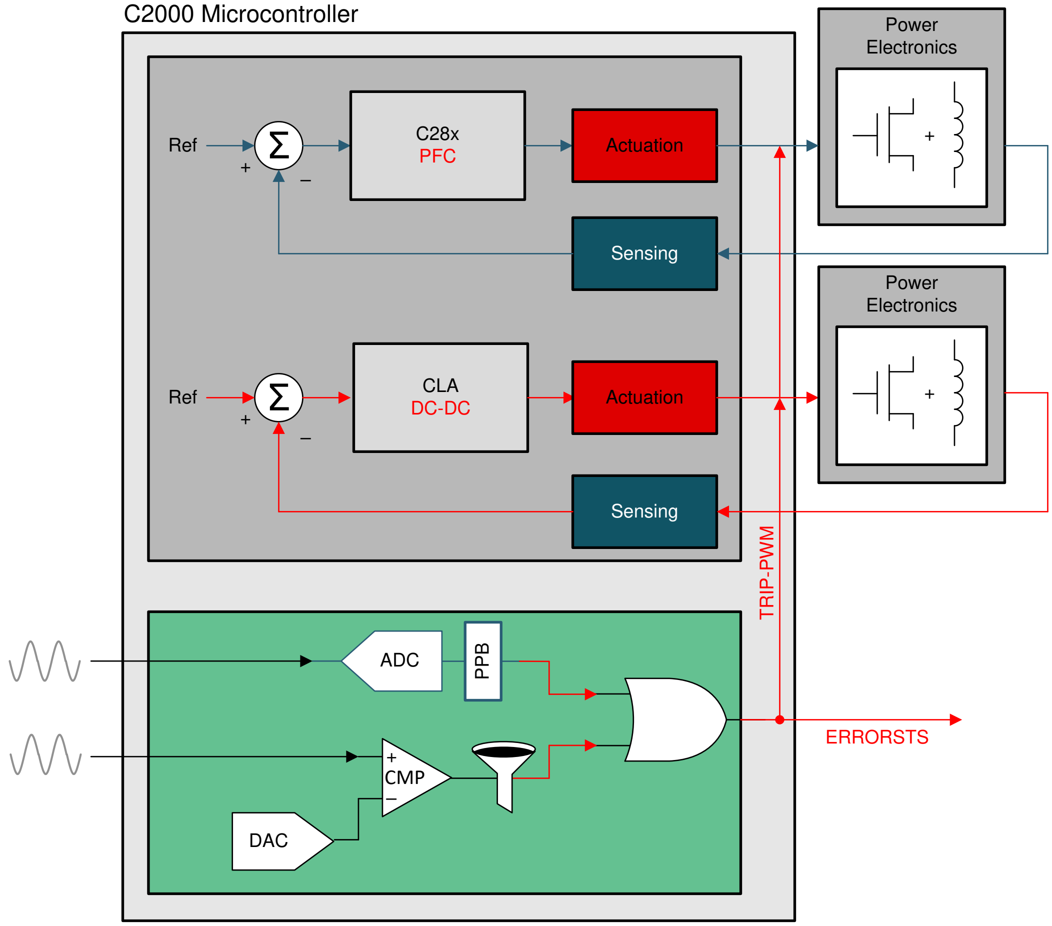


Figure 4-14. Safety Concept Implementation Option 2

- 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.

5 Brief Description of Safety Elements

This section contains a brief description of the elements on the TMS320F28004x MCU device family, organized based on the classification of parts of generic hardware of a system as indicated 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 ISO 26262:2018). 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 and also use many of the MCU parts like Interconnect, Memory (Flash, SRAM and ROM) and TMS320F28004x MCU infrastructure components (Clock, Power, Reset and JTAG). In order to ensure integrity of the implemented primary diagnostics and their 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 the successful operation of the processing units. For diagnostics for these parts, see the respective sections in this safety manual.

In case, separate test of diagnostic measures exist for a primary diagnostic measure, they are mentioned along with the respective hardware part.

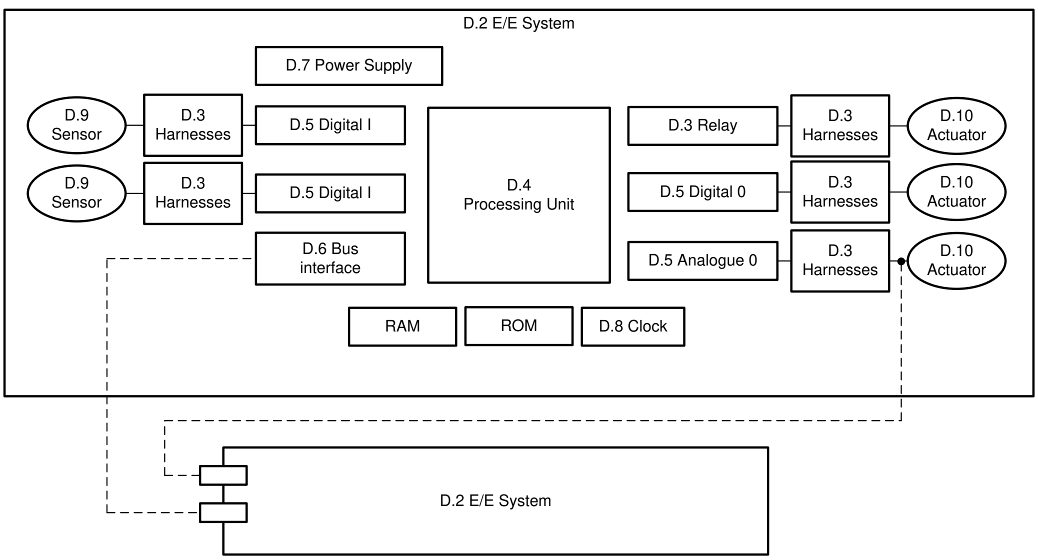


Figure 5-1. Generic Hardware of a System

5.1 TMS320F28004x MCU Infrastructure Components

5.1.1 Power Supply

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

- External Voltage Supervisor
- External Watchdog (using GPIO or a serial interface)
- Internal Watchdog (WD)
- Brownout Reset (BOR)
- Multi-Bit Enable Keys for Control Registers
- Lock Mechanism for Control Registers
- Software Read Back of Written Configuration
- Periodic Software Read Back of Static Configuration Registers
- Online Monitoring of Temperature
- EALLOW and MEALLOW Protection for Critical Registers

Note

- Having independent voltage supervision at system level is an assumption used while performing safety analysis.
- Devices can be implemented with multiple power rails that are intended to be ganged together on the system PCB. For proper operation of power diagnostics, it is recommended to implement one voltage supervisor per ganged rail.
- Common mode failure analysis of the external voltage supervisor along with TMS320F28004x 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 system level.

5.1.2 Clock

The TMS320F28004x 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):

- Missing Clock Detect (MCD)
- Clock Integrity Check Using CPU Timer
- Clock Integrity Check Using HRPWM
- Dual clock comparator (DCC) - Type0
- External Monitoring of Clock via XCLKOUT
- Internal Watchdog (WD)
- External Watchdog
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- PLL Lock Profiling Using On-Chip Timer
- Peripheral Clock Gating (PCLKCR)
- Efuse ECC

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

- Software Test of Watchdog (WD) Operation
- Software Test of Missing Clock Detect Functionality

Note

- Higher diagnostic coverage can be obtained by setting tighter bounds when checking clock integrity using Timer2.
- 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 Reset

The power-on reset (PORn) 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.

- External Monitoring of Warm Reset (XRSn)
- Reset Cause Information
- Software Test of Reset
- Glitch Filtering on Reset Pins
- NMIWD Shadow Registers
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- NMIWD Reset Functionality
- Peripheral Soft Reset (SOFTPRES)
- Internal Watchdog (WD)
- External Watchdog

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

- Software Test of Watchdog (WD) Operation

Note

- 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 TPS6538x power supply and safety companion device for reset supervision at system level.

5.1.4 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):

- Multi-Bit Enable Keys for Control Registers
- Lock Mechanism for Control Registers
- Software Read Back of Written Configuration
- Periodic Software Read Back of Static Configuration Registers
- Online Monitoring of Temperature
- Peripheral Clock Gating (PCLKCR)
- Peripheral Soft Reset (SOFTPRES)
- EALLOW and MEALLOW Protection for Critical Registers
- Software Test of ERRORSTS Functionality
- Peripheral access protection - Type 0

Note

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

5.1.5 Efuse Static Configuration

The TMS320F28004x MCU device family supports a boot time configuration of certain functionality (such as trim values for analog macros) with the help of Efuse structures. The Efuses are read automatically after power-on reset by an autoload function.

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

- Efuse Autoload Self-Test
- Efuse ECC

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

- Efuse ECC Logic Self-Test
- SRAM ECC
- SRAM Parity
- Software Test of SRAM
- VCU CRC Check of Static Memory Contents

5.1.6 JTAG Debug, Trace, Calibration, and Test Access

The TMS320F28004x 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. The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- Hardware Disable of JTAG Port
- Internal Watchdog (WD)
- External Watchdog

5.2 Processing Elements

5.2.1 C28x Central Processing Unit (CPU)

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

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

- Reciprocal Comparison by Software
- Software Test of CPU
- Periodic Software Read Back of Static Configuration Registers
- Access Protection Mechanism for Memories
- CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping
- Internal Watchdog (WD)
- External Watchdog
- Information Redundancy Techniques
- Stack Overflow Detection
- Embedded Real Time Analysis and Diagnostic (ERAD)

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

- VCU CRC Auto Coverage

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 ISO 26262-11:2018, Section 4.7 and IEC 61508-2:2010 Annex E (BetaIC method).

5.2.2 Control Law Accelerator

The Control Law Accelerator (CLA) is an independent, fully-programmable, 32-bit floating-point math accelerator with independent ISA and independent compiler and 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):

- Reciprocal Comparison by Software
- Software Test of CLA
- CLA Handling of Illegal Operation and Illegal Results
- Software Read Back of Written Configuration
- Periodic Software Read Back of Static Configuration Registers
- Information Redundancy Techniques
- CLA Liveness Check Using CPU
- Access Protection Mechanism for Memories
- Disabling of Unused CLA Trigger Sources

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

- Periodic Software Read Back of Static Configuration Registers
- Software Test of Function Including Error Tests

5.3 Memory (Flash, SRAM and ROM)

5.3.1 Embedded Flash Memory

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

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

- Flash ECC
- VCU CRC Check of Static Memory Contents
- Bit Multiplexing in Flash Memory Array
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Flash Program Verify and Erase Verify Check
- Software Test of Flash Prefetch, Data Cache and Wait-States
- Internal Watchdog (WD)
- External Watchdog
- CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping
- Information Redundancy Techniques

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

- Software Test of ECC Logic
- VCU CRC Auto Coverage

5.3.2 Embedded SRAM

The TMS320F28004x MCU device family has the following types of SRAMs with different characteristics.

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

All these RAMs are highly configurable to achieve control for write access and fetch access from different 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):

- SRAM ECC
- SRAM Parity
- Software Test of SRAM
- Bit Multiplexing in SRAM Memory Array
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Data Scrubbing to Detect/Correct Memory Errors
- VCU CRC Check of Static Memory Contents
- Software Test of Function Including Error Tests
- Access Protection Mechanism for Memories
- Lock Mechanism for Control Registers
- Information Redundancy Techniques
- CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping
- Internal Watchdog (WD)
- External Watchdog
- CLA Handling of Illegal Operation and Illegal Results
- Memory Power-On Self-Test (MPOST)

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

- Software Test of ECC Logic
- Software Test of Parity Logic
- VCU CRC Auto Coverage

5.3.3 Embedded ROM

The TMS320F28004x MCU device family has the following types of ROMs:

- 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 26262-6:2018/IEC 61508-3:2010) 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):

- VCU CRC Check of Static Memory Contents
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Software Test of Function Including Error Tests
- CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping
- Internal Watchdog (WD)
- External Watchdog
- Power-Up Pre-Operational Security Checks
- Memory Power-On Self-Test (MPOST)
- Reciprocal Comparison by Software

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

- Background CRC for CLA-ROM (CLAPROMCRC)
- VCU CRC Auto Coverage

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):

- Software Test of Function Including Error Tests
- Internal Watchdog (WD)
- External Watchdog
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping
- CLA Handling of Illegal Operation and Illegal Results
- Transmission Redundancy
- Hardware Redundancy
- EALLOW and MEALLOW Protection for Critical Registers

5.4.2 Direct Memory Access (DMA)

The direct memory access (DMA) module provides a hardware method of transferring data between peripherals and/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):

- Information Redundancy Techniques
- Transmission Redundancy
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Software Test of Function Including Error Tests
- DMA Overflow Interrupt
- Access Protection Mechanism for Memories
- Disabling of Unused DMA Trigger Sources

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

- Periodic Software Read Back of Static Configuration Registers
- Software Test of Function Including Error Tests

5.4.3 Enhanced Peripheral Interrupt Expander (ePIE) Module

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

- PIE Double SRAM Hardware Comparison
- Software Test of SRAM
- Software Test of ePIE Operation Including Error Tests
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Maintaining Interrupt Handler for Unused Interrupts
- Online Monitoring of Interrups and Events
- Hardware Redundancy

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

- PIE Double SRAM Comparison Check

5.4.4 Dual Zone Code Security Module (DCSM)

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

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

- Multi-Bit Enable Keys for Control Registers
- Majority Voting and Error Detection of Link Pointer
- Periodic Software Read Back of Static Configuration Registers
- Software Test of Function Including Error Tests
- Software Read Back of Written Configuration
- CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping
- VCU CRC Check of Static Memory Contents
- External Watchdog
- Hardware Redundancy

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

- VCU CRC Auto Coverage

5.4.5 CrossBar (X-BAR)

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

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

- Software Test of Function Including Error Tests
- Hardware Redundancy
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Software Check of X-BAR Flag

5.4.6 Timer

The 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):

- 1oo2 Software Voting Using Secondary Free Running Counter
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Software Test of Function Including Error Tests

5.5 Digital I/O

5.5.1 General-Purpose Input/Output (GPIO) and Pinmuxing

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):

- Lock Mechanism for Control Registers
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Software Test of Function Using I/O Loopback
- 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):

- Software Test of Function Including Error Tests
- Hardware Redundancy
- Monitoring of ePWM by eCAP
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Lock Mechanism for Control Registers
- ePWM Fault Detection using XBAR
- ePWM Synchronization Check
- ePWM Application Level Safety Mechanism
- Online Monitoring of Interrupts and Events
- Monitoring of ePWM by ADC

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):

- HRPWM Built-In Self-Check and Diagnostic Capabilities
- Hardware Redundancy
- Monitoring of ePWM by eCAP
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Lock Mechanism for Control Registers

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/voltage sensors.

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

- Software Test of Function Including Error Tests
- Information Redundancy Techniques
- Monitoring of ePWM by eCAP
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- eCAP Application Level Safety Mechanism
- 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):

- Software Test of Function Including Error Tests
- Hardware Redundancy
- Monitoring of HRPWM by HRCAP
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- 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):

- Software Test of Function Including Error Tests
- eQEP Quadrature Watchdog
- Information Redundancy Techniques
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- eQEP Application Level Safety Mechanisms
- Hardware Redundancy

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

- 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.

- SDFM Comparator Filter for Online Monitoring
- Information Redundancy Techniques
- SD Modulator Clock Fail Detection Mechanism
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Software Test of Function Including Error Tests
- 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):

- Software Test of Function Including Error Tests
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Hardware Redundancy

5.6 Analogue 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 TMS320F28004x MCU device family products implement up to three 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):

- Software Test of Function Including Error Tests
- DAC to ADC Loopback Check
- ADC Information Redundancy Techniques
- Opens/Shorts Detection Circuit for ADC
- Software Read Back of Written Configuration
- Periodic Software Read Back of Static Configuration Registers
- ADC Signal Quality Check by Varying Acquisition Window
- ADC Input Signal Integrity Check
- Monitoring of ePWM by ADC
- Hardware Redundancy
- Disabling Unused Sources of SOC Inputs to ADC

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):

- Software Test of Function Including Error Tests
- DAC to ADC Loopback Check
- Lock Mechanism for Control Registers
- Software Read Back of Written Configuration
- Periodic Software Read Back of Static Configuration Registers
- DAC to Comparator Loopback Check
- Hardware Redundancy

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

- Software Test of Function Including Error Tests
- Periodic Software Read Back of Static Configuration Registers

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):

- Software Test of Function Including Error Tests
- Hardware Redundancy
- Software Read Back of Written Configuration
- Periodic Software Read Back of Static Configuration Registers
- Lock Mechanism for Control Registers
- VDAC Conversion by ADC
- CMPSS Ramp Generator Functionality Check

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

- Software Test of Function Including Error Tests
- Periodic Software Read Back of Static Configuration Registers

5.6.4 Programmable Gain Amplifier (PGA)

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

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

- PGA to ADC Loopback Test
- Hardware Redundancy
- Software Read Back of Written Configuration
- Periodic Software Read Back of Static Configuration Registers
- Lock Mechanism for Control Registers

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):

- Software Test of Function Using I/O Loopback
- Information Redundancy Techniques Including End-to-End Safing
- SRAM Parity
- Software Test of SRAM
- Bit Multiplexing in SRAM Memory Array
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Transmission Redundancy
- DCAN Stuff Error Detection
- DCAN Form Error Detection
- DCAN Acknowledge Error Detection
- Bit Error Detection
- CRC in Message
- Hardware Redundancy

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

- Software Test of Parity Logic

5.7.2 Serial Peripheral Interface (SPI)

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

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

- Software Test of Function Using I/O Loopback
- Information Redundancy Techniques Including End-to-End Safing
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Transmission Redundancy
- SPI Data Overrun Detection
- Hardware Redundancy

5.7.3 Serial Communication Interface (SCI)

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

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

- Software Test of Function Using I/O Loopback
- Parity in Message

- Information Redundancy Techniques Including End-to-End Safing
- Overrun Error Detection
- SCI Break Error Detection
- Frame Error Detection
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Transmission Redundancy
- Hardware Redundancy

5.7.4 Inter-Integrated Circuit (I2C)

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

- Software Test of Function Using I/O Loopback
- I2C Data Acknowledge Check
- Information Redundancy Techniques Including End-to-End Safing
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Transmission Redundancy
- I2C Access Latency Profiling Using On-Chip Timer

5.7.5 Fast Serial Interface (FSI)

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

- Software Test of Function Using I/O Loopback Including Error Tests
- Information Redundancy Techniques Including End-to-End Safing
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Transmission Redundancy
- FSI Data Overrun/Underrun Detection
- FSI Frame Overrun Detection
- FSI CRC Framing Checks
- FSI ECC Framing Checks
- FSI Frame Watchdog
- FSI RX Ping Watchdog
- FSI Tag Monitor
- FSI Frame Type Error Detection
- FSI End of Frame Error Detection
- FSI Register Protection Mechanisms

5.7.6 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):

- Software Test of Function Using I/O Loopback
- Information Redundancy Techniques Including End-to-End Safing
- Transmission Redundancy

- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Data Parity Error Detection
- Overrun Error Detection
- Frame Error Detection
- LIN Physical Bus Error Detection
- LIN No-Response Error Detection
- Bit Error Detection
- Checksum Error Detection
- LIN ID Parity Error Detection
- SCI Break Error Detection
- Communication Access Latency Profiling Using On-Chip Timer

5.7.7 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):

- I2C Data Acknowledge Check
- Information Redundancy Techniques Including End to End Safing
- Periodic Software Read Back of Static Configuration Registers
- Software Read Back of Written Configuration
- Transmission Redundancy
- PMBus Protocol CRC in Message
- Clock Timeout

6 Brief Description of Diagnostics

This section provides a brief summary of the diagnostic mechanisms available on the TMS320F28004x 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.

6.1 TMS320F28004x MCU Infrastructure Components

6.1.1 Clock Integrity Check Using CPU Timer

It is recommended to use the CPU Timer module to detect incorrect clock frequencies and drift between clock sources. CPU Timer2 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 by 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 Timer2. 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.

6.1.2 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.

6.1.3 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 could result from 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 the write of protected registers are complete.

6.1.4 Efuse Autoload Self-Test

Efuse provides a capability to ensure proper loading of the efuse values to all the registers. The capability is enabled by default and configuration cannot be changed by software. Any error in this process will be indicated via ERRORSTS. The device reset is asserted and autoload is re-attempted when the error occurs.

6.1.5 Efuse ECC

The Efuse utilizes a SECDED ECC diagnostic to detect and possibly correct errors in the configuration values fetched from the fuse ROM. Errors are indicated via ERRORSTS. This diagnostic is ON by default and this configuration cannot be changed by software. It covers only data bits of the EFUSE ROM. The device reset is asserted and autoload is re-attempted when the error occurs.

6.1.6 Efuse ECC Logic Self-Test

The Efuse controller has a self-test logic that executes automatically before the efuse operation. Errors are indicated via ERRORSTS and a system control register. The device will remain in a reset state as long as the error occurs.

6.1.7 External Monitoring of Clock via XCLKOUT

The TMS320F28004x MCU device family provides the capability to export selected 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.

6.1.8 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.

6.1.9 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.

6.1.10 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 [IEC 61508-7:2010, clause A.9.3] or other appropriate methods such that high diagnostic effectiveness can be claimed.

6.1.11 Glitch Filtering on Reset Pins

Glitch filters are implemented on XRSn 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.

6.1.12 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 Mode Select (TMS) high. Disabling of the JTAG port also provides coverage for inadvertent activation of many debug and trace activities.

6.1.13 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.

The use of the time window allows detection of additional clocking failure modes as compared to the WD implementation. 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. For details of programming the internal watchdogs, see the device-specific technical reference manual.

In order to avoid common cause failure of clock input to both Internal Watchdog(WD) and CPU, it is recommended to select either INTOSC2 or X1/X2 as clock source to main PLL.

6.1.14 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.

6.1.15 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 doesn't 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.

6.1.16 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.

6.1.17 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 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.

6.1.18 Multi-Bit Enable Keys for Control Registers

Some modules include features to support avoidance of unintentional control register update. Implementation of multi-bit keys for critical control registers is one such feature (for example, EPWM_REGS.EPWMLOCK and so forth). The multi-bit 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.

6.1.19 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 ADCIN14 on ADCB by setting the ENABLE bit in the TSNSCTL register.

6.1.20 Periodic Software Read Back of Static Configuration Registers

Configuration registers are typically configured once in the beginning and hold their value until the particular task execution. Periodic read back of configuration registers can provide a diagnostic for inadvertent writes or disturbances to 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 (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 interrupt flag in peripherals and PIE module to ensure that correct interrupt is being serviced.

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

6.1.21 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.

6.1.22 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.

6.1.23 PLL Lock Profiling Using On-Chip Timer

Clock setup for the TMS320F28004x 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:

- Clock Integrity Check Using CPU Timer
- Clock Integrity Check Using HRPWM
- External Clock Monitoring via XCLKOUT to ensure proper clock output

6.1.24 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.

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

6.1.26 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 (e.g. asserting CLOCLKFAIL NMIFLG by updating the NMIFLGFRC.bit.CLOCKFAIL). Error response, diagnostic testability, and any necessary system requirements are defined by the system integrator.

6.1.27 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).

6.1.28 Software Test of Reset

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 PORn 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.

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.

6.1.29 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 a 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.

6.1.30 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.

6.1.31 Dual clock comparator (DCC) - Type0

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 pre-defined 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.

6.1.32 Peripheral Access Protection - Type 0

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 details, see the PERIPH_AC_REGS Registers in *TMS320F28004x Technical Reference Manual*.

6.2 Processing Elements

6.2.1 CLA Handling of Illegal Operation, 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. 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.

6.2.2 CLA Liveness Check Using CPU

CLA doesn't have an independent watchdog of its own. Hence, 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, CLA Task1, CPU Task2, and CLA 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 6-1. The liveness check can be based on application-specific parameters as illustrated in the VDA Egas concept [6] to improve the diagnostic coverage.

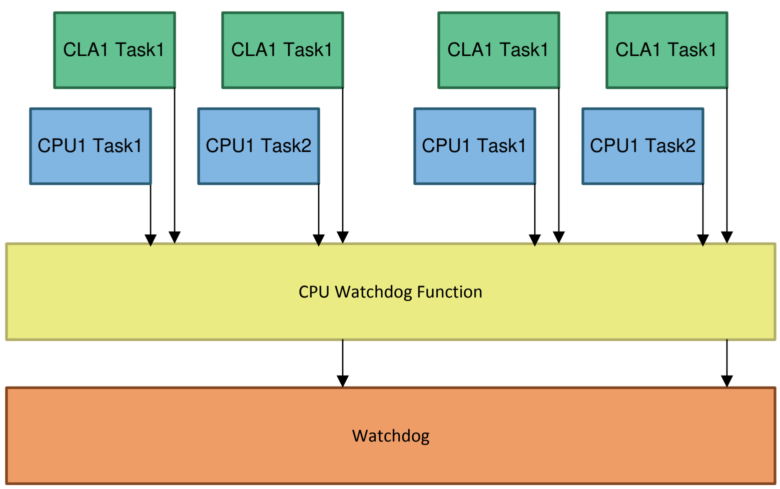


Figure 6-1. CLA Liveness Check

6.2.3 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 Extended Instruction Sets. Technical Reference Manual*

6.2.4 Reciprocal Comparison by Software

The CPU subsystem has a pair of diverse processing units (C28x 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 TMS320F28004x 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 ISO 26262-5:2018, 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 extend 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. During the runtime, CLA has access to GPIO_Data_Regs that can indicate the error condition on a GPIO pin independent of C28x.

6.2.5 Software Test of CLA

It is possible to test the integrity of various CLA blocks including register file, 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 TMS320F28004x MCU device. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.2.6 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 will provide a C28x-STL startup test library with 60% diagnostic coverage for C28x, FPU and TMU. For details on implementing the particular test, check the safety package delivered with the specific TMS320F28004x MCU device. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.2.7 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 ERAD module can monitor the internal address and data buses, and triggers 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. This is illustrated in Figure 6-2. 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.

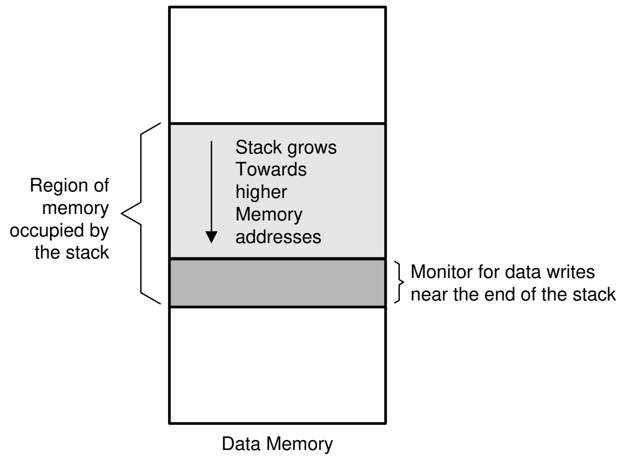


Figure 6-2. Stack Overflow Monitoring

6.2.8 VCU CRC Check of Static Memory Contents

The TMS320F28004x MCU device family includes co-processor implementing cyclic redundancy check (CRC) using standard polynomials. The CRC module can be used to test the integrity of SRAM, Flash, and OTP 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.

6.2.9 VCU CRC Auto Coverage

The VCU CRC diagnostic is based on up to 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 VCU CRC logic or associated data path, it is extremely unlikely that the correct passing code will be generated via the fault.

6.2.10 Disabling of Unused CLA 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.

6.2.11 Embedded Real Time Analysis and Diagnostic (ERAD) - Type 0

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.

6.3 Memory (Flash, SRAM and ROM)

6.3.1 Bit Multiplexing in Flash Memory Array

The flash modules implemented in the TMS320F28004x 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 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.

6.3.2 Bit Multiplexing in SRAM Memory Array

The SRAM modules implemented in the TMS320F28004x 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 multi-bit faults resulting in logical multi-bit 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.

6.3.3 Data Scrubbing to Detect/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, VCU CRC Check of Static Memory Contents provides better diagnostic coverage compared to this diagnostic.

6.3.4 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 by the flash wrapper 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 flash wrapper.

6.3.5 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 (FMSTAT) will indicate the error by setting the corresponding flags into the status register.

6.3.6 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 registers (FECC_CTRL, FADDR_TEST, FECC_TEST, FDATAH_TEST, FDATAL_TEST). Correct functioning of error counter and threshold interrupt associated with single bit errors can also be verified using this technique. 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 *SECDED Logic Correctness Check* sections in the *TMS320F28004x Microcontrollers Technical Reference Manual*.

6.3.7 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.

6.3.8 Access Protection Mechanism for Memories

All volatile memory blocks including external memories except for M0/M1 have different levels of protection. This capability allows the user to enable or disable specific access (e.g. Fetch, Write) to individual RAM blocks from individual masters (viz. CPU,CLA,DMA). 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.

6.3.9 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 TMS320F28004x 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 TMS320F28004x MCU device-specific data sheet.

6.3.10 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 TMS320F28004x MCU device-specific data sheet.

6.3.11 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.

For additional details on implementing this diagnostic for SRAM, see the *Application Test Hooks for Error Detection and Correction* section in *TMS320F28004x Microcontrollers Technical Reference Manual*.

6.3.12 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 VCU 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.

6.3.13 Background CRC for CLA-PROM (CLAPROMCRC)

The CLAPROMCRC is a safety feature performs CRC on a configurable block of memory in the CLA program ROM space. Neither the C28x nor the CLA have access to compute CRC on the CLA program ROM. The CLAPROMCRC solves this problem by calculating a CRC in a non-intrusive manner (that is, without impacting CLA accesses to CLA PROM). It is a hardware CRC-32 module tbat automatically fetches the CLA program ROM in background during idle cycles (when CLA is not accessing the ROM on the CLA program bus), and calculates the CRC-32 in order to perform a code integrity check. It then compares the result with a golden CRC-32 value and registers pass or fail condition. The module issues an interrupt on completion of the test.

Figure 6-3 is a functional diagram of the CLAPROMCRC module.

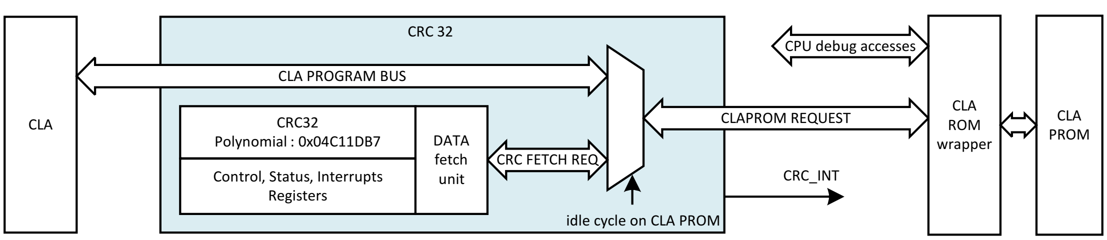


Figure 6-3. CLAPROMCRC Functional Diagram

6.3.14 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 On-Chip Communication Including Bus-Arbitration

6.4.1 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 CPU Timer2 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.

6.4.2 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

6.4.3 Maintaining Interrupt Handler for Unused Interrupts

The TMS320F28004x 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 TMS320F28004x 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 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.

6.4.5 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. 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.

6.4.6 PIE Double SRAM Hardware Comparison

PIE SRAM address space is duplicated and data is placed in two memories. During a vector fetch, the ePIE performs a hardware comparison of both vector table outputs. If there is a mismatch between the two vector tables, the CPU branches to the address in the PIEVERRADDR register and the ePIE sends trip signals to the PWMs. If the PIEVERRADDR register value has not been set, the default boot ROM handler at address 0x003FFFBE is used.

6.4.7 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 by waiting to a redundant vector address. The interrupt corresponding to the mismatched PIE vector in SRAM needs to be triggered by software. Then software needs to verify that CPU branches to the address in the PIEVERRADDR register and the ePIE sends trip signals to the PWMs. For details for implementation of this check, see the *Vector Address Validity Check* section in the *TMS320F28004x Microcontrollers Technical Reference Manual*.

6.4.8 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 tripzones, ePWM syncing or GPIO output signaling is missed.

6.4.9 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 and ensuring that the interrupt is serviced and serviced in proper order. The interrupt can be generated 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. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

6.4.10 Disabling of Unused DMA Trigger Sources

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 the initiation of unintended DMA transfers, it is recommended that unused DMA channels and DMA trigger sources are disabled at source or by configuring DMACHSRCSELx registers.

6.5 Digital I/O

6.5.1 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.

6.5.2 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.

6.5.3 ePWM Fault Detection Using X-BAR

A combination of ePWM outputs feedback to input X-BAR, GPIO inversion logic and Digital Compare (DC) sub-module 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.

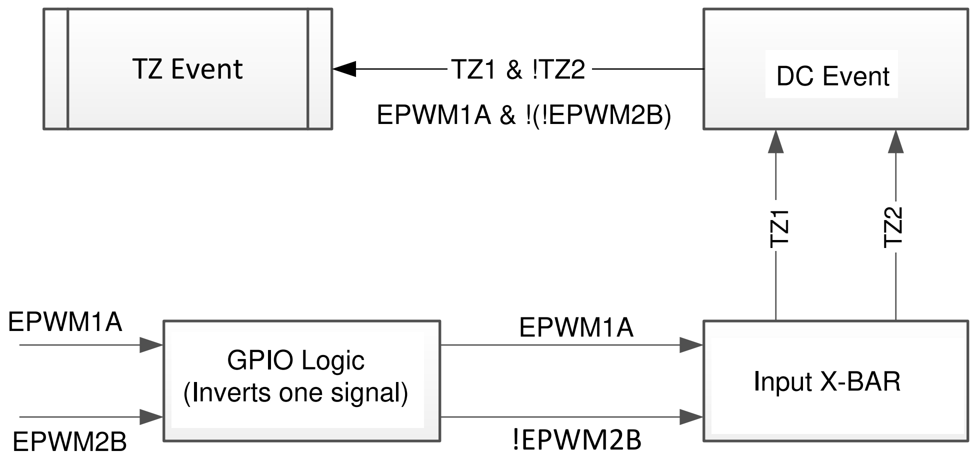


Figure 6-4. ePWM Fault Detection Using X-BAR

6.5.4 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 TBSTS.SYNCI bit of ePWM module. The proper phase relationship intended as a result of the sync operation can be cross-checked by comparing the TBCTR register value.

6.5.5 eQEP Application Level Safety Mechanism

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.

6.5.6 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.

6.5.7 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.

6.5.8 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) TMS320F28004x MCU.

In case of peripherals like GPIO, X-BAR, ePWM, OTTO, DAC, CMPSS and XINT, hardware redundancy can be implemented by having multi-channel parallel outputs (where independent outputs are used for transmitting information, and failure detection is carried out via internal or external comparators) or input comparison or voting (comparison of independent inputs to ensure compliance with a defined tolerance range on time and 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 (e.g. 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 ADC, PGA, eCAP, HRCAP and eQEP, 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.

In case of SDFM, hardware redundancy may be implemented by having multiple channels sample the same input followed by cross check of the output values.

In case of communication peripherals like DCAN, SPI and SCI, 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.

Hardware Redundancy for device interconnect (INC) can be implemented by redundant data storage/ transmission by independent processing units for computation followed by comparison of the computed results.

Hardware Redundancy for peripheral interrupt (PIE) can be implemented by connecting interrupt to CLA in parallel as trigger source. Reciprocal comparison can be implemented between CPU and CLA to enable detection of faulty PIE behavior.

In case of Dual Code Security Module (DCSM), C28x CPU and CLA can be configured to access its resources via independent Zones. Reciprocal comparison can be implemented between CPU and CLA to enable detection of a fault in DCSM module.

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 ePWM module instance. In case of DAC module the comparator can be implemented using an external device.

While implementing hardware redundancy for the ePWM module, it is recommended that ePWM module instance used is part of separate sync chains. This is to avoid common cause failure on sync signal affecting both the ePWM modules in same way.

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

6.5.9 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.

6.5.10 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 TMS320F28004x MCU (wiring harness, connectors, transceiver) end-to-end safety mechanisms are applied. These mechanisms can also provide diagnostic coverage inside the TMS320F28004x 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 ISO 26262-5:2018, 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 TMS320F28004x 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, V_1, V_2, V_3 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. Safety program in flash can be copied to SRAM and execute after performing a CRC check against a pre-calculated golden CRC value.

6.5.11 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 X-BAR. 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 measuring ePWM pulse width as a test for diagnostic. 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.

6.5.12 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 6-5. 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 necessary software requirements are defined by the software implemented by the system integrator.

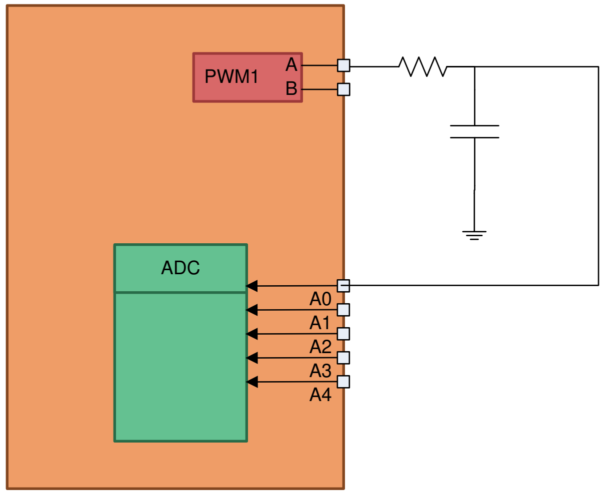


Figure 6-5. Monitoring of ePWM by ADC

6.5.13 Online Monitoring of Periodic Interrupts and Events

For interrupts and events, failures 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 using peripherals such as ERAD include:

- Online monitoring of periodic occurrence of interrupts, for example, ePWM, ADC end-of-conversion (EOC), eCAP and eQEP interrupts
- Online monitoring of periodic events:
  - Periodic generation of ADC start-of-conversion (SOC): 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.

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

6.5.14 SDFM Comparator Filter for Online Monitoring - Type 0

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 three programmed threshold levels to detect over and under-value conditions and zero crossing events. In case comparator filter's data output crosses low or high threshold limit, it will fire interrupt to the CPU. The output of the comparator filter can also be used to verify the primary filter output with proper scaling.

6.5.15 SD Modulator Clock Fail Detection Mechanism

When SD modulator clock fails or goes missing for 256 continuous system clock cycles, clock fail detection sub-module 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 I/O connecting modulator clock.

6.5.16 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 given below:

- SDFM functionality can be checked by sending a known input test sequence to the TMS320F28004x 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 and low threshold register values to minimum and maximum 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.
- 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.
- eCAP, HRCAP and eQEP functionality can be checked by looping back the PWM, HRPWM 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 and HRCAP, the test can be done internally with the help of input X-BAR.
- ROM prefetch functionality can be checked using similar techniques as given in Section 6.3.7.
- The ePWM 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 ePWM and observing the response using one of the capture (time stamping) 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 6.3.12).

- The interconnect (INC) functionality can be tested by writing complementary data-patterns like 0xA5A5,0x5A5A, and so forth from processing units from 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 and 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.
- To test core functionality of the ADC module and post processing block (PPB), a set of predetermined voltage levels can be provided on the ADC input pin by external circuit or internal DAC. The ADC / PPB results thus obtained can be cross checked against the expected value to ensure proper operation. Extreme corner values of ADC being used in application can be applied and tested to check the successful conversion across the operational range. ADC configuration registers can be checked by writing complementary data-patterns, read back and compared to expected values.
- 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 SAMPWIN (Sample window) and THRESH (Majority voting threshold) values, and then verifying COMPHSTS/COMPLSTS changes with change in filter output. Applicable range of filter clock pre-scaler 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, or XTAL. Test interrupts generation capability at the end of the timer counting. Check for the time overflow flag and Timer reload (TRB) functions in 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.

6.5.17 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 measuring HRPWM pulse width as a test for diagnostic. 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.

6.5.18 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 6-6.

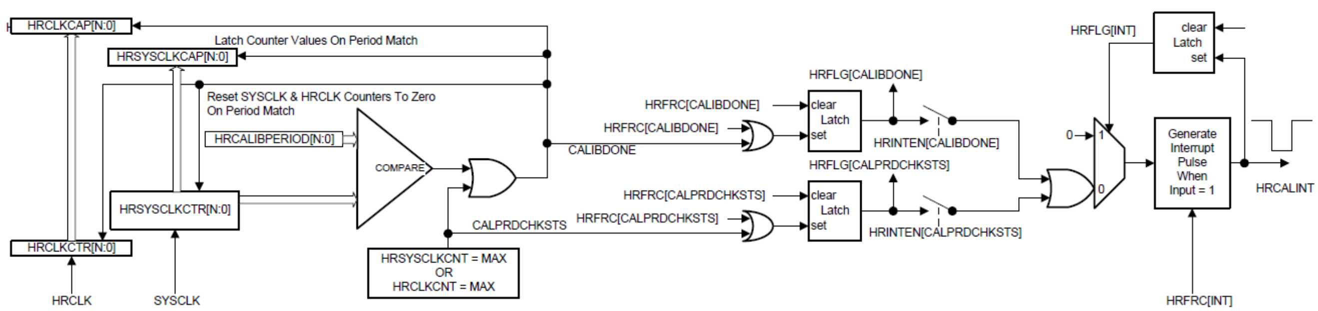


Figure 6-6. HRCAP Calibration

$$ScaleFactor = \frac{HRSYSCLKCAP}{HRCLKCAP} \quad (1)$$

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

Note

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

6.5.19 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 *TMS320F28004x Microcontrollers 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.

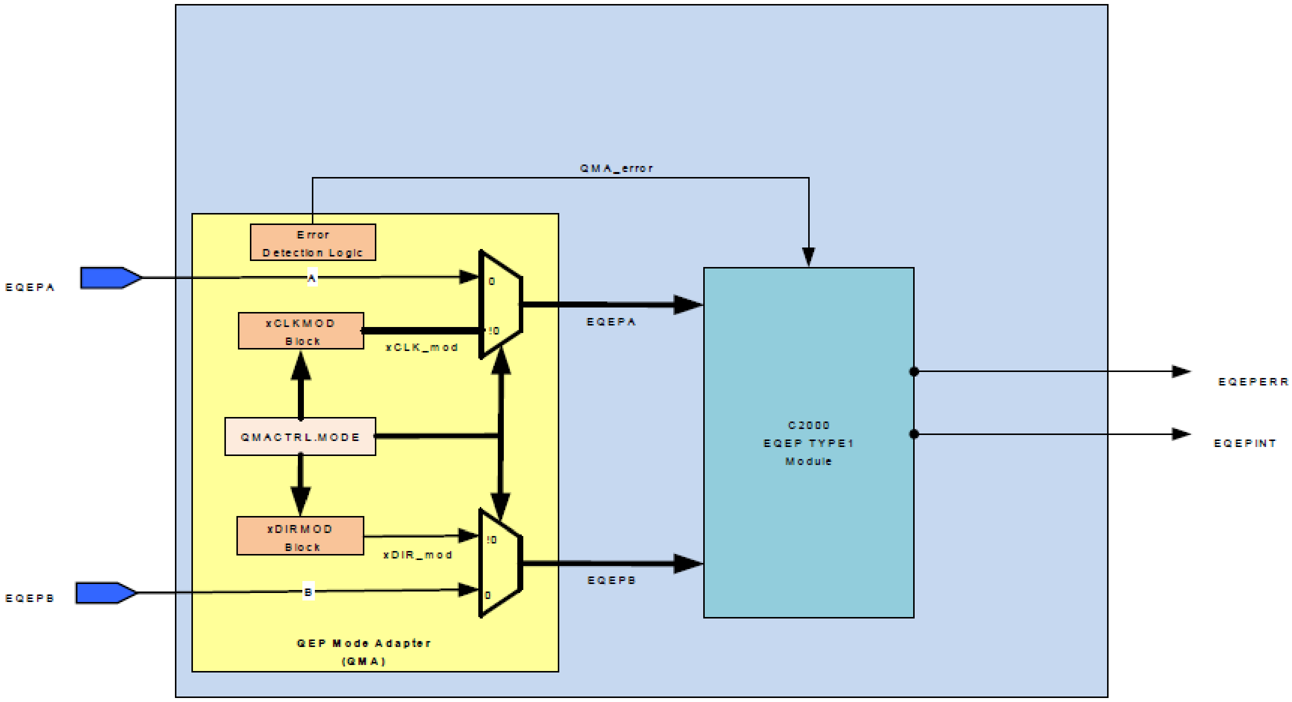


Figure 6-7. QMA Module Block Diagram

6.6 Analogue I/O

6.6.1 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.

6.6.2 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 (e.g. 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.

6.6.3 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 also be 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 pre-defined range determined by the application and ADC specification parameters.

6.6.4 CMPSS Ramp Generator Functionality Check

CMPSS ramp generation functionality is used in certain control applications (e.g. 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.

6.6.5 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 12-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.

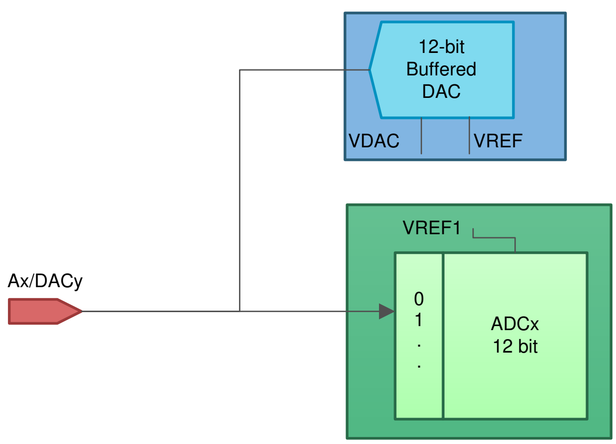


Figure 6-8. DAC to ADC Loopback

6.6.6 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.

6.6.7 PGA to ADC Loopback Test

Integrity of PGA can be checked driving PGAx_IN input with a known set of values and monitoring output after conversion using on-chip ADC. DAC can be configured using software to provide a set of predetermined voltage levels. With an external connection, output of DAC can be connected to PGAx_IN. ADC can convert PGAx_OUT and SW can compare the ADC output to test if PGA amplified input signal to appropriate scale.

Figure 6-9 represents two possible schemes of setting up testing of PGA using DAC and ADC.

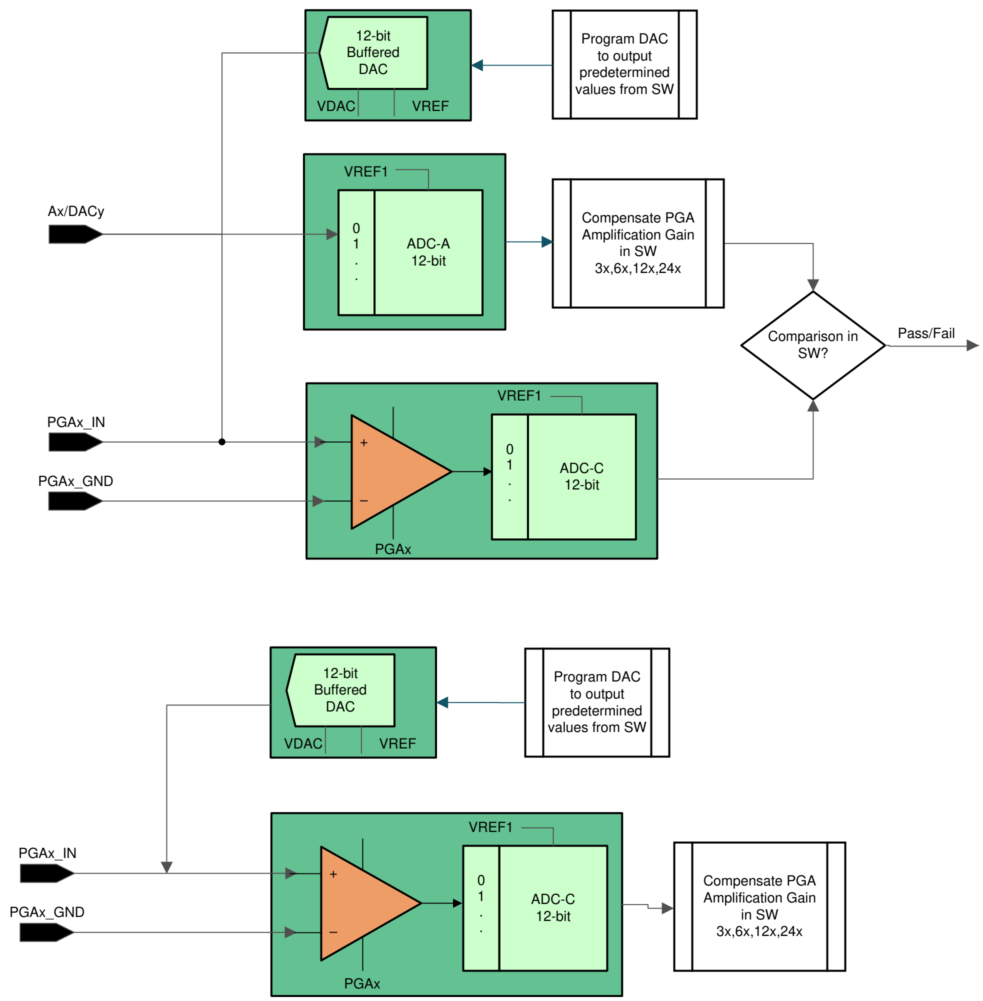


Figure 6-9. Testing PGA Using ADC and DAC

6.6.8 Opens/Shorts Detection Circuit for ADC

The opens/shorts detection circuit (OSDETECT) can be used to detect ADC input channel faults in the system. The circuit connects to the ADC input after the channel select multiplexer but before the S+H circuit as shown in Figure 6-10. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

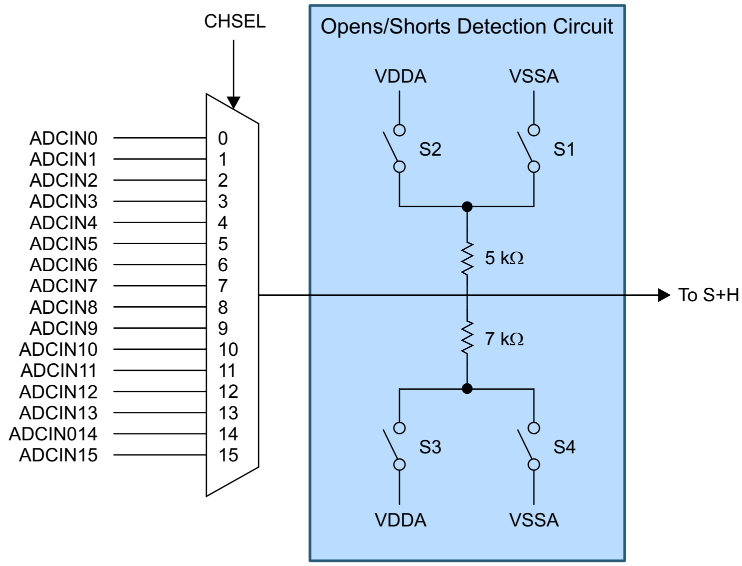


Figure 6-10. ADC Open-Shorts Detection Circuit

The circuit can be operated by writing a value to the DETECTCFG field in the ADCOSDETECT register. This will cause the circuit to source a voltage onto the input during the S+H phase of any conversion. The voltage and drive strength of the OSDETECT circuit for different DETECTCFG settings is given in Table 6-1. For additional details on implementing this diagnostic, see the *Opens/Shorts Detection Circuit (OSDETECT)* section in the *TMS320F28004x Microcontrollers Technical Reference Manual*.

Table 6-1. ADC Open-Shorts Detection Circuit Truth Table

ADCOSDETECT.DETECTCFG	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

6.6.9 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.

6.6.10 Disabling Unused Sources of SOC Inputs to ADC

ADC SOC (start of conversion) signal input to ADC module can be triggered by multiple sources, include software, CPU Timers, GPIO, and ePWM module instances, and so forth. 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 enables only the required SOC triggers. This is a way to avoid faults originating from an outside source to impact functionality of ADC.

6.7 Data Transmission

6.7.1 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 TMS320F28004x MCU (wiring harness, connectors, transceiver), end-to-end safety mechanisms are applied. These mechanisms can also provide diagnostic coverage inside the TMS320F28004x 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 time stamp) 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.

6.7.2 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

6.7.3 CRC in Message

The CAN 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.

6.7.4 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.

6.7.5 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.

6.7.6 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.

6.7.7 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.

6.7.8 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.

PMBus supports detection of errors using acknowledgment handshake, which can be configured to work in either automatic or manual mode(PMBSC.MAN_SLAVE_ACK bit). This acknowledgment handshake can be effectively implemented by firmware to detect communication faults such as masquerading faults by asserting NACK if the received address does not equal the slave address or acknowledging every byte received by PMBus slave receive byte acknowledge, or acknowledging received command byte, and so forth. For more details, see the *UCD3138 Monitoring and Communications Programmer's Manual*.

6.7.9 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.

6.7.10 SCI 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.

This feature is applicable only when LIN 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.

6.7.11 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.

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.

6.7.12 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.

6.7.13 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.

6.7.14 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.

6.7.15 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.

6.7.16 FSI Data Overrun/Underrun Detection

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 data overrun/underrun error occurs and corresponding register bit is enabled.

6.7.17 FSI Frame Overrun Detection

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.

6.7.18 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.

6.7.19 FSI ECC Framing Checks

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 multi-bit errors in received data and invalidate the received data.

Note

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

6.7.20 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 guarantee proper operation. When this condition occurs, a flag is set and an interrupt is generated if enabled.

6.7.21 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.

6.7.22 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.

6.7.23 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 guarantee proper operation. An interrupt is generated if corresponding register bit is enabled.

6.7.24 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.

6.7.25 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.

6.7.26 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 no Synch Break can be generated (for example, because of a bus shortage to VBAT) or if no Synch break Delimiter can be generated (for example, because of a bus shortage to GND).

6.7.27 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.

6.7.28 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.

6.7.29 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.

6.7.30 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

6.7.31 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.

6.7.32 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.

6.7.33 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. On chip timer module can be used for profiling the time required for completing each transaction.

7 References

1. Texas Instruments: *Calculating Useful Lifetimes of Embedded Processors*
2. *Moisture/Reflow Sensitivity Classification for Nonhermetic Solid State Surface Mount Devices*, https://www.jedec.org/standards-documents/docs/jesd-22-a112
3. *Handling, Packing, Shipping and Use of Moisture/Reflow Sensitive Surface Mount Devices*, https://www.jedec.org/sites/default/files/docs/jstd033b01.pdf
4. *IEC 61508 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*, International Electrotechnical Commission, Edition 2.0 2010.
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. Texas Instruments: *Using PWM Output as a Digital-to-Analog Converter on a TMS320F280x Digital Signal Controller*
9. W. M. Goble and H. Cheddie, Safety Instrumented Systems Verification: Practical Probabilistic Calculations. Isa, 2004.
10. Texas Instruments: *TMS320C28x FPU Primer*
11. Texas Instruments: *Online Stack Overflow Detection on the TMS320C28x DSP*
12. IEC-61784 official website. Available online at https://www.iec.ch.
13. Texas Instruments: *TMS320F28004x Microcontrollers Technical Reference Manual*
14. Texas Instruments: *TMS320F28004x Microcontrollers Data Manual*
15. Texas Instruments: *UCD3138 Monitoring and Communications Programmer's Manual*
16. Texas Instruments: *Accelerators: Enhancing the Capabilities of the C2000™ MCU Family*

A Safety Architecture Configurations

The various redundancy architectures possible for the safety instrumented systems are indicated in Table A-1. For more information, see [10].

Table A-1. Safety Architecture Configurations

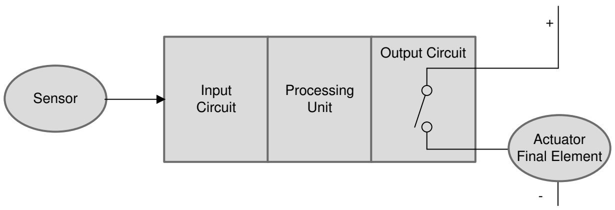
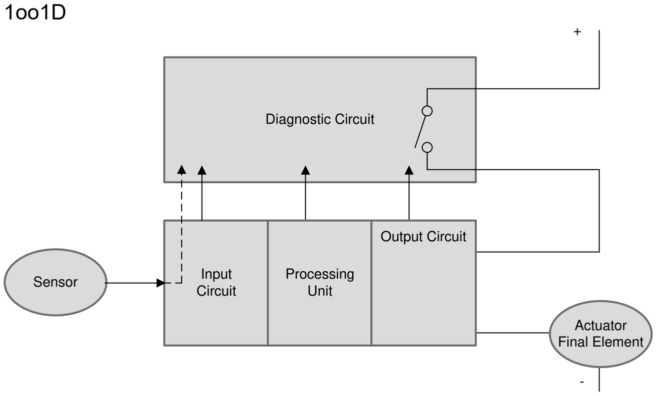
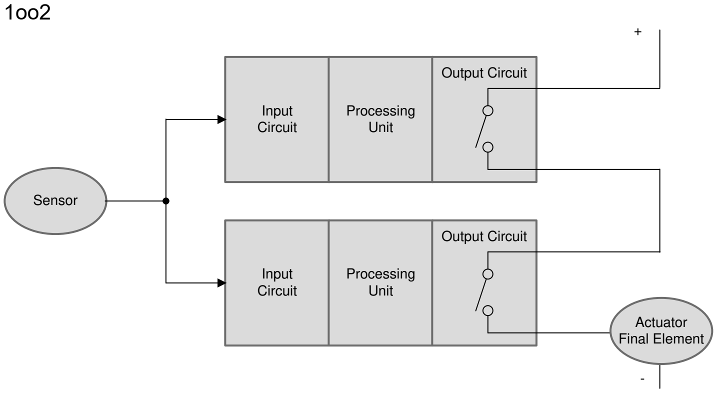
		Diagnostic Implementation
1	1oo1 Architecture 	None.
2	1oo1D 	Diagnostic channel is implemented using various hardware diagnostic features like Watchdog, and so forth.
3	1oo1D Same figure as above.	Diagnostic channel is implemented using reciprocal comparison (uses two processing units for implementing reciprocal comparison) and other hardware diagnostic features.
4	1oo2 	Two different processing units are used to implement one channel.

Table A-1. Safety Architecture Configurations (continued)

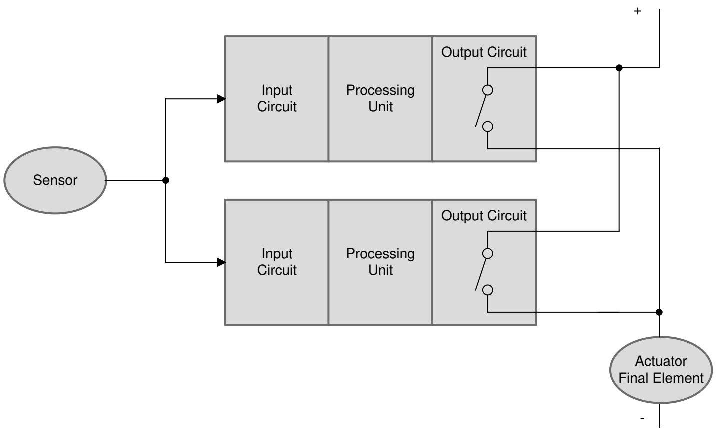
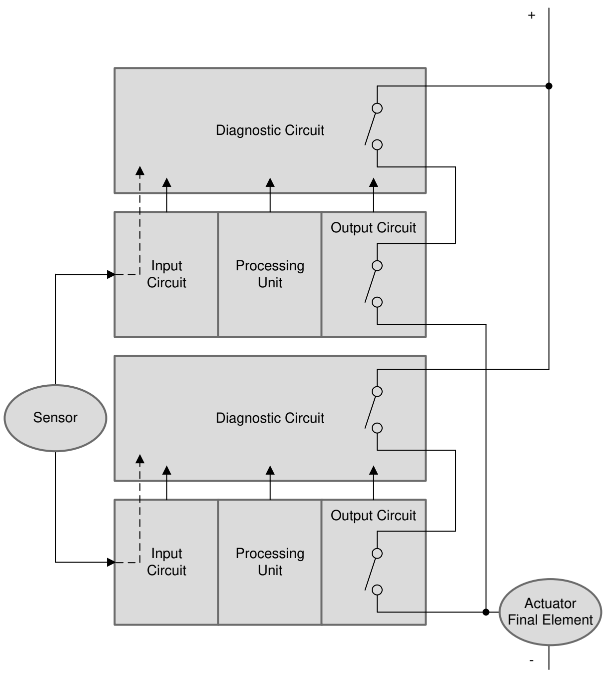
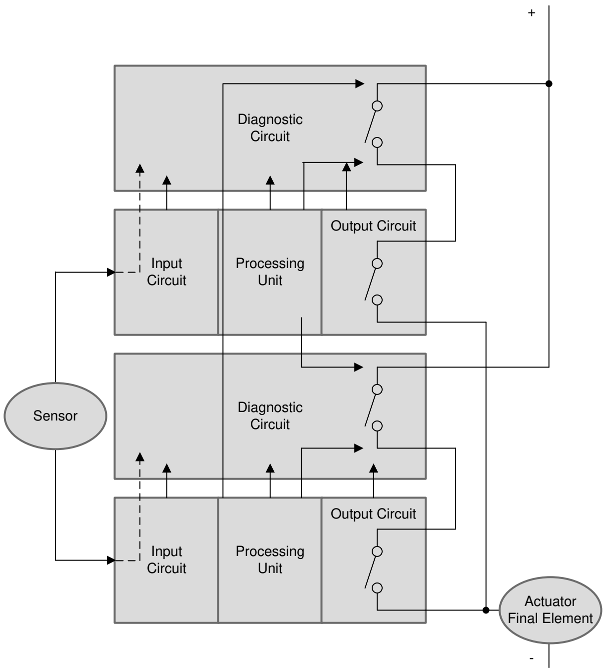
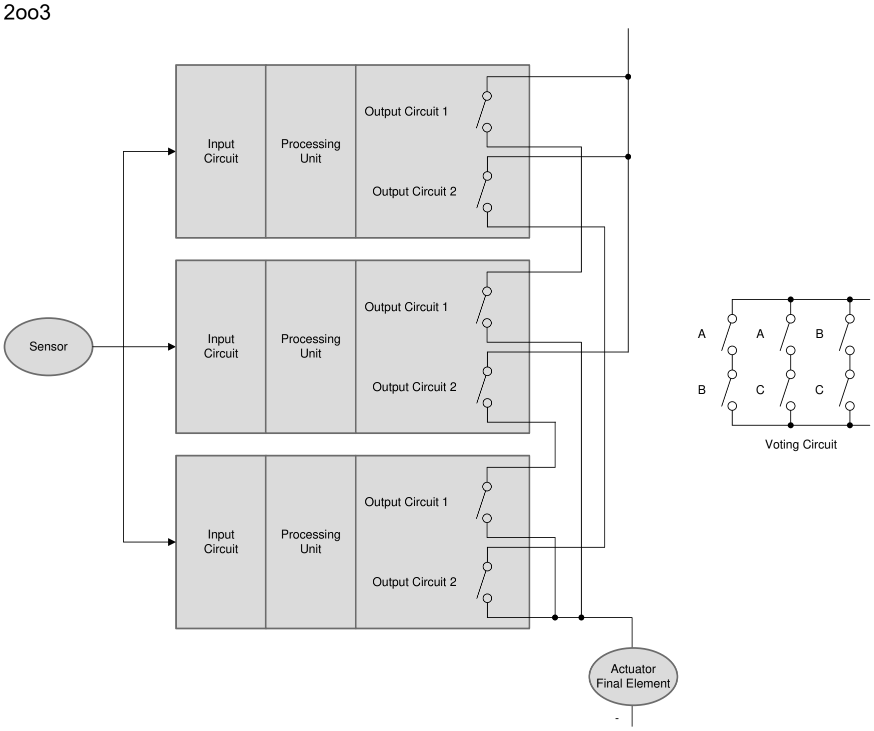
		Diagnostic Implementation
5	2oo2 	Two different processing units are used to implement one channel.
6	2oo2D 	Two 1oo1D structures of #2 wired together to implement a safe channel.
7	2oo2D Same figure as above.	Two 1oo1D structures of #3 wired together.

Table A-1. Safety Architecture Configurations (continued)

		Diagnostic Implementation
8	1oo2D  	Similar to 2oo2D implementation of #6 with additional control lines wired to control one set of units using the other unit
9	1oo2D Same figure as above.	Similar to 2oo2D implementation of #7 with additional control lines wired to control one set of units using the other unit.
10	2oo3  	Use three different processing units to implement majority voting. The fourth channel can be used either standalone or with hardware diagnostic features.

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 functional safety manual, functional safety analysis report and functional 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 Lifecycle Applies to Functional Safety-Compliant Products

TI has tailored the functional safety lifecycles 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 Lifecycle 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 Lifecycle 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.

C Terms and Definitions

- IEC 61508: Functional safety standard for E/E/PE safety-related systems. This is intended to be a basic functional safety standard applicable to all kinds of industry. It defines functional safety as: “part of the overall safety relating to the EUC (Equipment Under Control) and the EUC control system which depends on the correct functioning of the E/E/PE safety-related systems, other technology safety-related systems and external risk reduction facilities” [4].
- ISO 13849: provides safety requirements and guidance for the design and integration of safety-related parts of control systems (SRP/CS), including software design.
- M out of N (MooN) architecture: A safety instrumented system where ‘M’ channels out of ‘N’ channels are required for functionally safe operation, (for example, 2oo3, 2 out of 3 architecture, where majority voting is used to implement a safety function).

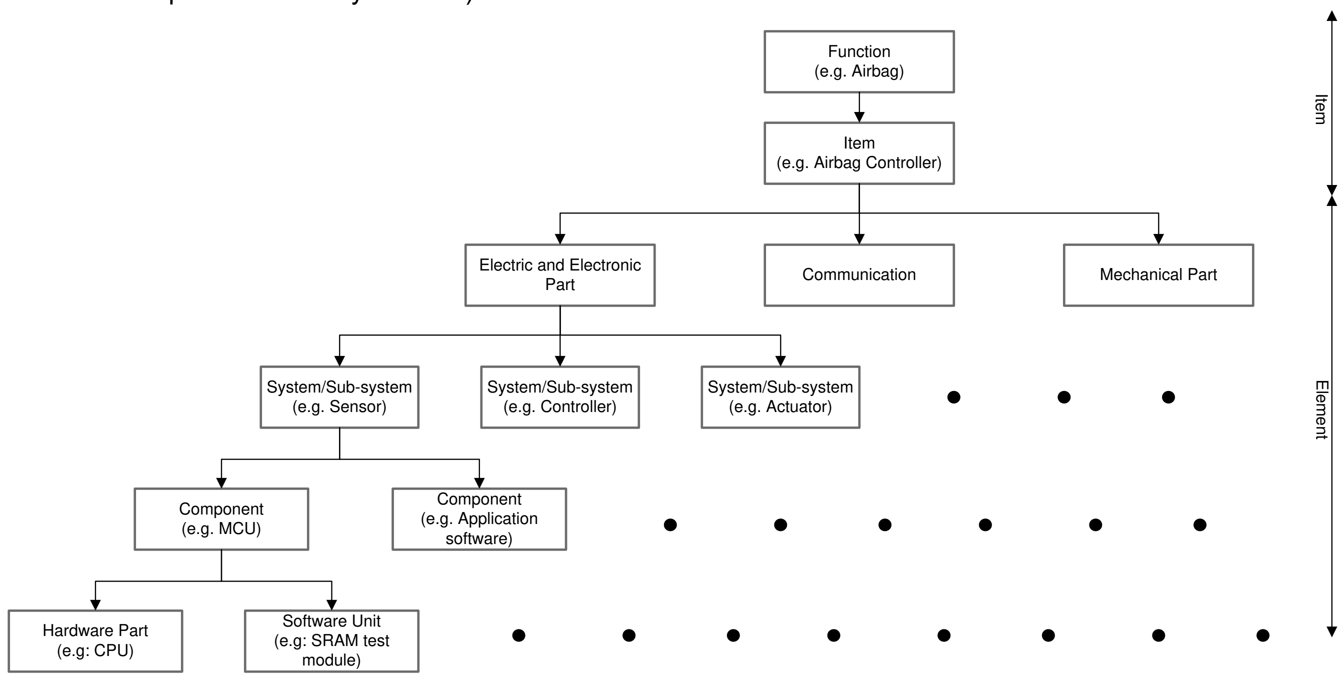


Figure C-1. ISO 26262 Illustration of Item, System, Component, Hardware Part and Software Unit

- M out of N Channel Architecture with diagnostics (MooND).
- Functional Safety: Part of the overall safety relating to the EUC and the EUC control system that depends on the correct functioning of the E/E/PE safety-related systems and other risk reduction measures
- Item: system or array of systems to implement a function at the vehicle level, to which ISO 26262:2018 is applied (for example, power steering of a car).

- Element: System or part of a system including components, hardware, software, hardware parts, and software units.
- System: set of elements that relates at least a sensor, a controller and an actuator with one another
- Component: Non-system level element that is logically and technically separable and is comprised of hardware parts and software units.
- Hardware part: Hardware that cannot be subdivided (for example, CPU).
- Software unit: Atomic level software component of the software architecture that can be subjected to stand-alone testing (for example, SRAM test module).
- Failure: termination of the ability of an element, to perform a function as required.
- Failure mode: manner in which an element or an item fails.
- Single Point Fault: Fault in an element that is not covered by a safety mechanism and that leads directly to the violation of a safety goal.
- Single-point failure: Failure that results from a single-point fault and that leads directly to the violation of a safety goal.
- Multiple-point fault: Individual fault that, in combination with other independent faults, leads to a multiple-point failure.
- Multiple-point failure: Failure resulting from the combination of several independent faults, which leads directly to the violation of a safety goal. For a multiple-point failure to directly violate a safety goal, presence of all independent faults is necessary.
- Multiple-point fault detection interval: time span to detect multiple-point fault before it can contribute to a multiple-point failure.
- Latent fault: multiple-point fault whose presence is not detected by a safety mechanism nor perceived by the driver within the multiple-point fault detection interval.
- Functional Safety Assessment: Investigation, based on evidence, to judge the functional safety achieved by one or more E/E/PE safety-related systems and/or other risk reduction measures.
- Functional Safety Audit: Systematic and independent examination to determine whether the procedures specific to the functional safety requirements to comply with the planned arrangements are implemented effectively and are suitable to achieve the specified objectives.
- Hazard and Risk Analysis (IEC 61508:2010)/Hazard Analysis and Risk Assessment (ISO 26262:2018): An end equipment level functional safety analysis that is used to identify safety functions and/or functional safety goals. This process also establishes the SIL (IEC 61508:2010) or ASIL (ISO 26262:2018), which defines the level of risk reduction necessary per safety function and/or functional safety goal.
- Process Tailoring: The act of changing a development process or functional safety lifecycle to match needs of a business engagement. Requirements can be moved from phase to phase or performed by other developers, but removal of process requirements is not allowed.
- Quality Managed: Describes a design element which is developed compliant to applicable quality standards but is not developed compliant to applicable functional safety standards. It may be possible to use a quality managed design element in a specific functional safety design contingent upon results of a functional safety qualification.
- Safety Requirement Decomposition: Safety requirements decomposition is the process in which safety requirements are split into a series of redundant safety requirements at a lower level of abstraction in order to support tailoring of the SIL (ISO 26262:2018)/ASIL (ISO 26262:2018) compliance requirements of design elements at the lower level of abstraction. For example, a requirement for a peripheral function with high safety integrity might be addressed by redundant instances of a peripheral with lower safety integrity.
- For the full list of applicable terms and their definitions for ISO 26262, see the ISO 26262-1:2018, Road vehicles — Functional safety — Part 1: Vocabulary.
- For the full list of applicable terms and their definitions for IEC 61508, see the IEC 61508:2010, Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 4: Definitions and abbreviations.

D Summary of Safety Features and Diagnostics

Table D-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 two types. A "diagnostic" test or a "test for diagnostic". 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
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 D-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
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 less than 1us
Clock	CLK1	Missing Clock Detect (MCD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion and PLL reference clock switch to INTOSC1	0.82ms
	CLK2	Clock Integrity Check Using CPU Timer	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLK3	Clock Integrity Check Using HRPWM	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CLK4	Dual clock comparator (DCC) - Type0	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Interrupt to CPU	Typically <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CLK5	External Monitoring of Clock 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)

Table D-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)
	RST3	Software Test of Reset	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	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)
System Control Module and Configuration Registers	SYS1	Multi-Bit 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
	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)

Table D-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 (cont.)	SYS9	Software Test of ERRORSTS Functionality	Diagnostic	Software	On demand (software defined)	Software defined	System defined	System defined
	SYS10	Peripheral Access Protection - Type 0	Fault avoidance	Hardware - Software	On demand (software defined)	NA (Fault Avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
EFuse	EFUSE1	Efuse Autoload Self-Test	Diagnostic	Hardware	Bootup (enabled by default)	Zero or very low overhead	Device reset	<400 CPU cycles
	EFUSE2	Efuse ECC	Diagnostic	Hardware	Bootup (enabled by default)	Zero or very low overhead	Device reset	<400 CPU cycles
	EFUSE4	Efuse ECC Logic Self-Test	Test for diagnostic	Hardware	Bootup (enabled by default)	Zero or very low overhead	Device reset	<400 CPU cycles
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)
	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
C28x Central Processing Unit	CPU1	Reciprocal Comparison by Software	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU3	Software Test of CPU	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU5	Access Protection Mechanism for Memories	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <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)

Table D-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 (cont.)	CPU8	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	CPU9	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	CPU10	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU14	Stack Overflow Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU15	VCU CRC Auto Coverage	Test for diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Software defined	Software defined
	CPU18	Embedded Real Time Analysis and Diagnostic (ERAD) - Type 0	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
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	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)

Table D-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) (cont.)	CLA11	Disabling of Unused CLA Trigger Sources	Fault avoidance	Software	Continuous - Enabled by software	Zero or very low overhead	NA (Fault avoidance technique)	NA (Fault avoidance technique)
Flash	FLASH1	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)
	FLASH2	VCU CRC Check of Static Memory Contents	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FLASH3	Bit Multiplexing in Flash Memory Array	Fault avoidance	Hardware	Continuous - Enabled at reset	NA (Fault Avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	FLASH4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FLASH5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FLASH6	Software Test of ECC Logic	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FLASH7	Flash Program Verify and Erase Verify Check	Fault avoidance	Software	On demand (Software defined)	NA (Fault Avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
	FLASH8	Software Test of Flash Prefetch, Data Cache and Wait-States	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FLASH9	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	FLASH10	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	FLASH12	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)
	FLASH14	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined

Table D-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	VCU CRC Check of Static Memory Contents	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM10	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM11	Access Protection Mechanism for Memories	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <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 D-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 (cont.)	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 <1 μS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	SRAM21	Memory Power-On Self-Test (MPOST)	Diagnostic	Hardware	Bootup (enabled by default)	Software defined	Software defined	Software defined
ROM	ROM1	VCU CRC Check of Static Memory Contents	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM2	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM3	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM4	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM5	CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <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

Table D-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 (cont.)	ROM9	Background CRC for CLA-PROM (CLAPROMCRC)	Diagnostic	Hardware - 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
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
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

Table D-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) (cont.)	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	Continuous - Enabled by software	NA (Fault Avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
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

Table D-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	Multi-Bit 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	VCU CRC Check of Static Memory Contents	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DCSM9	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	DCSM11	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
Cross Bar (X-BAR)	XBAR1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	XBAR2	Hardware Redundancy	Diagnostic	Software	Continuous - Enabled by software	Zero or very low overhead	Software defined	Software defined
	XBAR3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	XBAR4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	XBAR5	Software Check of X-BAR Flag	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
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

Table D-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 (cont.)	TIM4	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
General Pupose 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
Enhanced Pulse Width Modulators (ePWM)	PWM1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM2	Hardware Redundancy	Diagnostic	Software	Continuous - Enabled by software	Zero or very low overhead	Software defined	Software defined
	PWM3	Monitoring of ePWM by eCAP	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM6	Lock Mechanism for Control Registers	Fault avoidance	Hardware	Continuous - Enabled by software	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 Periodic Interrupts and Events	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM13	Monitoring of ePWM by ADC	Diagnostic	System	On demand (Software defined)	Software defined	Software defined	Software defined

Table D-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
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
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 D-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 Mechanism	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	QEP7	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	QEP8	QMA Error Detection Logic	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <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
Programmable Gain Amplifier (PGA)	PGA1	PGA to ADC Loopback Test	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PGA2	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PGA3	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PGA4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PGA6	Lock Mechanism for Control Registers	Fault avoidance	Hardware	Continuous - Enabled by software	NA (Fault Avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)

Table D-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
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)
	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)

Table D-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
Local Interconnect Network (LIN) (cont.)	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	SCI 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)
	LIN16	Communication Access Latency Profiling Using On-Chip Timer	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
Fast Serial Interface (FSI)	FSI1	Software Test of Function Using I/O Loopback	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI2	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI5	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI6	FSI Data Overrun/Underrun Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS 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 <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)

Table D-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) (cont.)	FSI8	FSI CRC Framing Checks	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)
	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 <1 µS 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 <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	FSI12	FSI Tag Monitor	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	FSI13	FSI Frame Type Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS 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 <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	FSI15	FSI Register Protection Mechanisms	Fault avoidance	Hardware	Continuous - Enabled by software	NA (Fault Avoidance)	NA (Fault avoidance technique)	NA (Fault avoidance technique)
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

Table D-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) (cont.)	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)
Sigma Delta Filter Module (SDFM)	SDFM1	SDFM Comparator Filter for Online Monitoring - Type 0	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
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

Table D-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
XINT (cont.)	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
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	On demand (Software defined)	Software defined	Software defined	Software defined
	ADC10	Hardware Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	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)
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

Table D-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 (cont.)	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
CMPSS	CMPSS1	Software Test of Function Including Error Tests	Test for 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	Test for 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
Controller Area Network (DCAN)	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

Table D-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 (DCAN) (cont.)	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)
	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
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

Table D-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) (cont.)	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
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	SCI 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)

Table D-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) (cont.)	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
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

E Glossary

Defined terms used in this document are listed in Table E-1.

Table E-1. Glossary

Acronyms	Expansion
ADC	Analog-to-Digital Converter
ASIL	Automotive Safety Integrity Level (ISO 26262:2018)
CLA	Control Law Accelerator
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSP	Compliance Support Package
DAC	Digital-to-Analog Converter
DC	Diagnostic Coverage
DTI	Diagnostic Test Interval
E/E/PE	Electrical/Electronic/Programmable Electronic
E2E	End-to-End Protocol
ePIE	enhanced Peripheral Interrupt Expansion
ePWM	enhanced Pulse Width Modulator
eQEP	enhanced Quadrature Encoder Pulse
ERAD	Embedded Real Time Analysis and Diagnostic
EUC	Equipment Under Control
FMEDA	Failure Mode Effects and Diagnostic Analysis
FPU	Floating Point Unit
FSA	Functional Safety Assessment
FSI	Fast Serial Interface
FTA	Fault Tree Analysis
FTTI	Fault Tolerant Time Interval
HARA	Hazard Analysis and Risk Assessment
HFT	Hardware Fault Tolerance
HRCAP	High Resolution Capture
IEC	International Electro Technical Commission
ISO	International Organization for Standardization
LIN	Local Interconnect Network
MCU	Microcontroller Unit
MTBF	Mean Time Between Failure
OTP	One Time Configurable
PGA	Programmable Gain Amplifier
PMBus	Power Management Bus Module
PWM	Pulse Width Modulator
SECDED	Single Error Correction, Double Error Detection
SIL	Safety Integrity Level
SOC	Start of Conversion
SPS	Software Product Specification
TI	Texas Instruments Inc.
TMU	Trigonometric Math Unit
VCU	Viterbi, Complex Math and CRC Unit

Revision History

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

Changes from Revision C (September 2020) to Revision D (January 2022)	Page
• Updated the numbering format for tables, figures and cross-references throughout the document	3
• Update was made in Section 4.3	19
• Update was made in Section 4.3.4	22

IMPORTANT NOTICE AND DISCLAIMER