Functional Safety Information

Functional Safety Manual for TMS320F280013x

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

This document is a functional safety manual for the Texas Instruments TMS320F280013x microcontroller product family. The product family utilizes a common safety architecture that is implemented in multiple application focused products.

This functional safety manual provides information needed by system developers to help in the creation of a functional safety system using a TMS320F280013x MCU. This document includes:

- An overview of the component architecture
- The details of architecture partitions and recommended functional safety mechanisms

The following information is documented in the *Functional Safety Analysis Report* and is not repeated in this document:

- Summary of failure rates (FIT) of the component
- Summary of functional safety metrics of the hardware component for any targeted standards, if applicable (for example IEC 61508, ISO 26262, and so forth)
- 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

The user of this document should have a general familiarity with the TMS320F280013x 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 consistency with other published documents, the term 'functional safety' may occasionally be used in this document. These instances are not indicating this product is compliant with any functional safety standards. The F280013x microcontroller product family has not been evaluated for adherence to any functional safety standards.

For information that is beyond the scope of the listed deliverables, contact your TI sales representative or go to www.ti.com.

2 TMS320F280013x Hardware Component Functional Safety Capability

This section summarizes the TMS320F280013x product safety capability. Each TMS320F280013x product:

- Was not developed according to the requirements of any functional safety standard
- FIT rates and failure mode distributions are provided as part of the functional safety analysis report for customers to calculate random fault integrity metrics
- Recommendations are provided in this functional safety manual for external safety mechanisms that may provide coverage for component failure modes

TI recommends that this component is integrated into the system through the strategy of 'evaluation of hardware element' (ISO 26262-8:2018 clause 13).

Additionally,

- The TMS320F280013x 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, the expectation is that the component functional safety manual provides 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 TMS320F280013x 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.

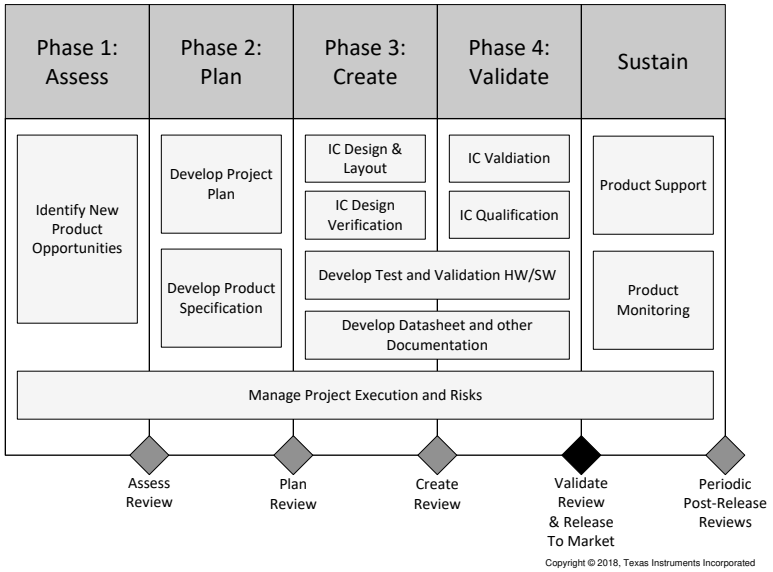

Figure 3-1. TI New-Product Development Process

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 TI functional safety 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 which are overlaid atop the standard development flow in Figure 3-1.

The customer facing work products derived from this TI functional safety 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 Component Overview

4.1 Targeted Applications

The TMS320F280013x devices are powerful, 32-bit floating-point microcontroller units (MCU) designed for advanced, closed-loop control applications in automotive and industrial applications.

4.1.1 TMS320F280013x MCU

TMS320F280013x is a powerful, 32-bit floating-point microcontroller unit (MCU) that allows the 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. 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 TMS320F280013x supports up to 256KB (128KW) of on-chip flash memory with error correction code (ECC) and up to 36KB (18KW) of SRAM with parity or ECC.

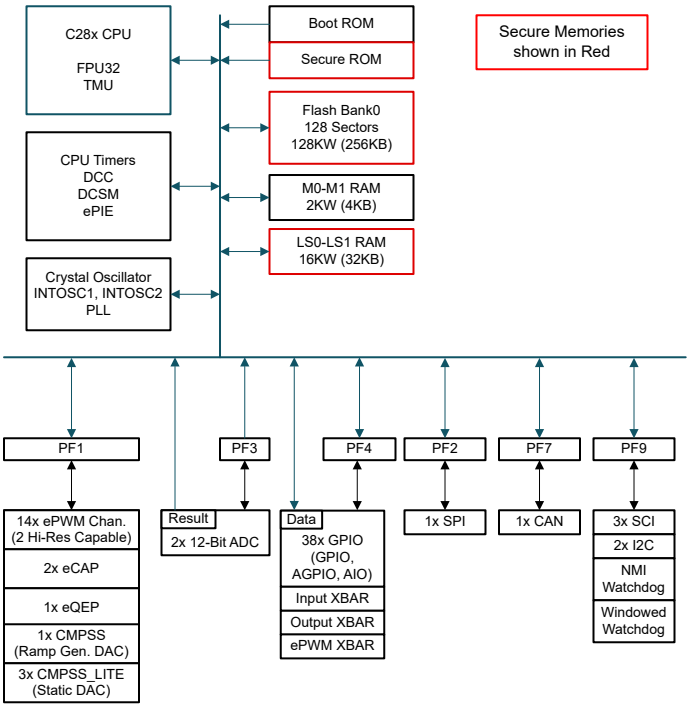

Figure 4-1. Functional Block Diagram of TMS320F280013x MCU

High-performance analog and control peripherals are integrated to further enable system consolidation. Independent 12-bit ADCs provide precise and efficient management of multiple analog signals, which ultimately boosts system throughput. The comparator subsystem (CMPSS) with windowed comparators allows for protection of power stages when current limit conditions are exceeded or not met. Other control peripherals include enhanced pulse width modulation (ePWM), enhanced capture (eCAP), enhanced quadrature encoder pulse (eQEP), and embedded pattern generator (EPG).

Peripherals, such as, controller area network (CAN) (ISO11898-1/CAN 2.0B-compliant), inter-integrated communication (I2C) bus, serial communications interface (SCI), and serial peripheral interface (SPI), extend connectivity of TMS320F280013x MCU.

The device configurations supported by this functional safety manual for TMS320F280013x MCUs is outlined in the *TMS320F280013x Real-Time Microcontrollers* data sheet. Not all variants are available in all packages or all temperature grades. To confirm availability, contact your local Texas Instruments sales and marketing.

4.2 Hardware Component Functional Safety Concept

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

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 the TMS320F280013x microcontroller, as indicated by Figure 4-2, can be used in a compliant item according to IEC 61508:2010.

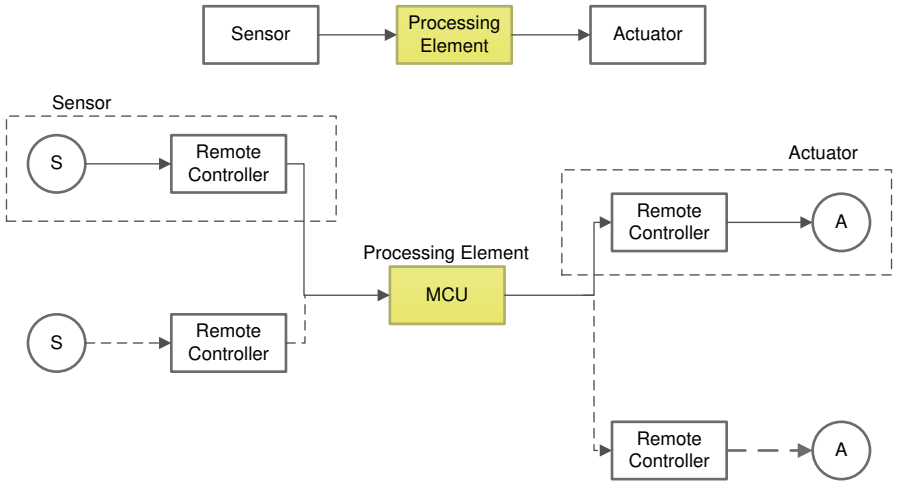

Figure 4-2. Definition of the TMS320F280013x MCU Used in a Compliant Item

4.2.1 TMS320F280013x MCU Safety Features

Due to the inherent versatility of the device architecture, several software voting-based functional safety configurations are possible. While implementing these configurations, the system integrator must consider the potential common-mode failures and address these failures in an appropriate manner. These failures can be addressed by adapting TMS320F280013x 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 TMS320F280013x are shown in Figure 4-3.

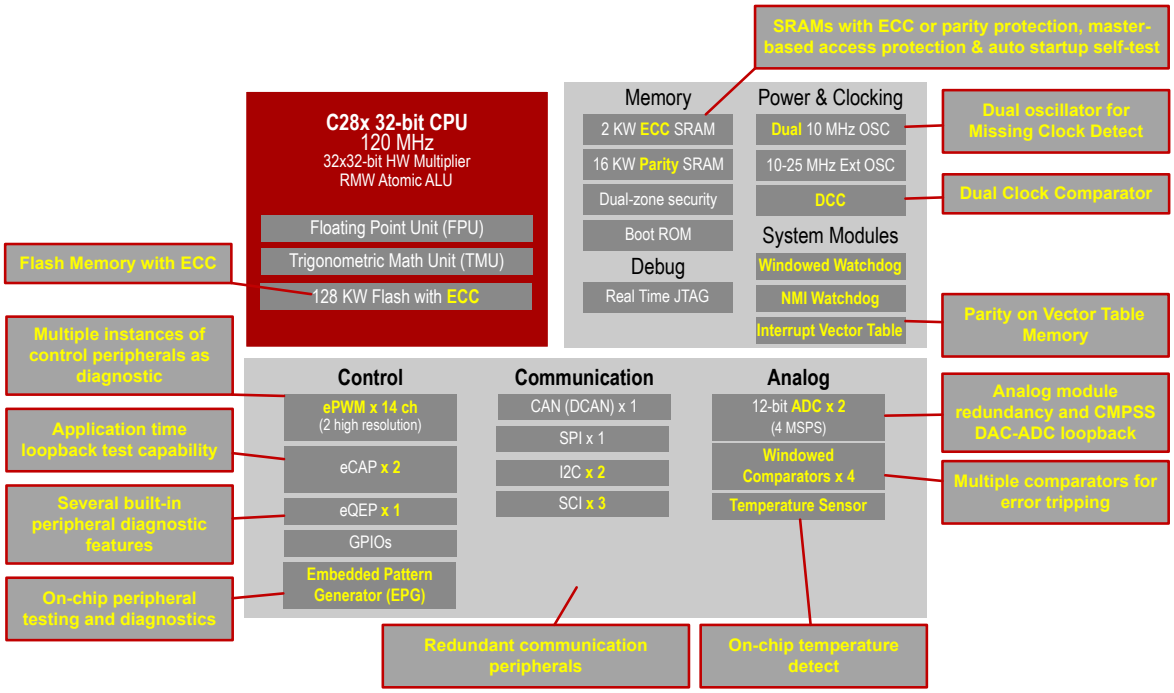

Figure 4-3. TMS320F280013x MCU With Safety Features

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 by the application software. The maximum time that a safety mechanism takes to detect a fault is termed as *fault diagnostic test time interva*l (FDTI). Once the fault is detected, depending on the fault reaction of the associated fault (for example, an external system reaction to an ERRORSTS pin assertion), the system enters the Safe state. The time-span in which a fault, or faults, can be present in a system before a hazardous event occurs is called a fault tolerant time interval (FTTI), as defined in ISO 26262. This is similar to process safety time (PST) defined in IEC 61508. Figure 4-4 illustrates the relationship between FDTI, fault reaction time and FTTI.

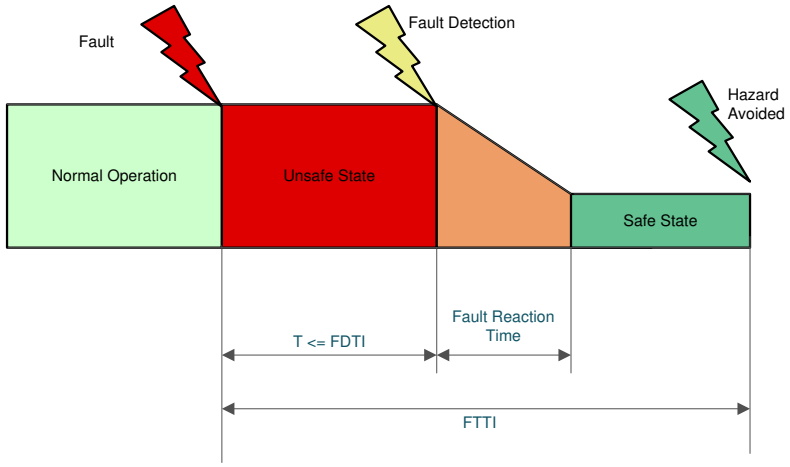

Figure 4-4. Relationship Between FDTI, Fault Reaction Time, and FTTI

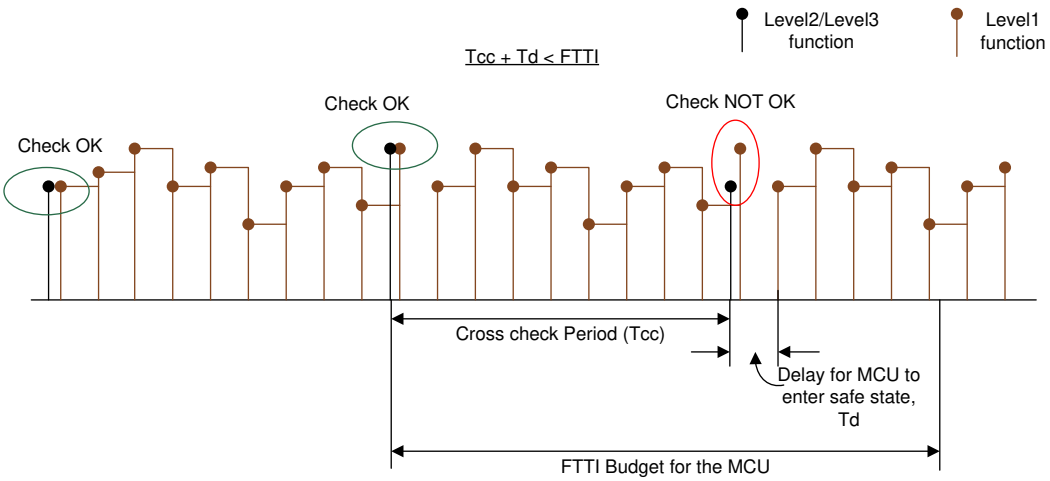

Figure 4-5. Illustration of FTTI

The frequency and extent of each of the checks in the application must be consistent with the fault tolerant time interval (FTTI). Figure 4-5 illustrates the frequency of the required checks. The checks must be such that single point faults of the microcontroller must be detected and responded to, so that the TMS320F280013x 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 4-6. An example of a diagnostic for single point faults is ECC/Parity for memory.

The proposed functional safety concept, subsequent functional safety features, and configurations explained in this document are for reference purpose only. The system and equipment designer, or manufacturer, is responsible for verifying 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 TMS320F280013x MCU Safe State

Referring to Figure 4-6, the Safe state of the TMS320F280013x MCU is defined as one in which:

- The TMS320F280013x MCU reset is asserted.
- The power supply to TMS320F280013x 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 the assumption is that the system-level functionality exists to manage this condition.
- The external system is informed using one of the IO pins of the C2000 MCU as a result of Level 2 check failure (for example, ERRORSTS pin is asserted).
- The output of the TMS320F280013x 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).

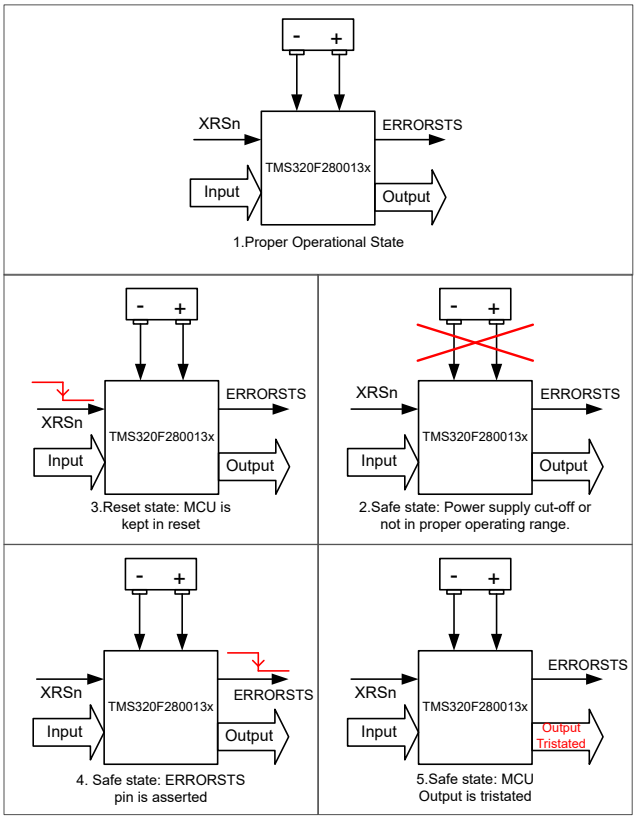

Figure 4-6. TMS320F280013x MCU Safe State Definition

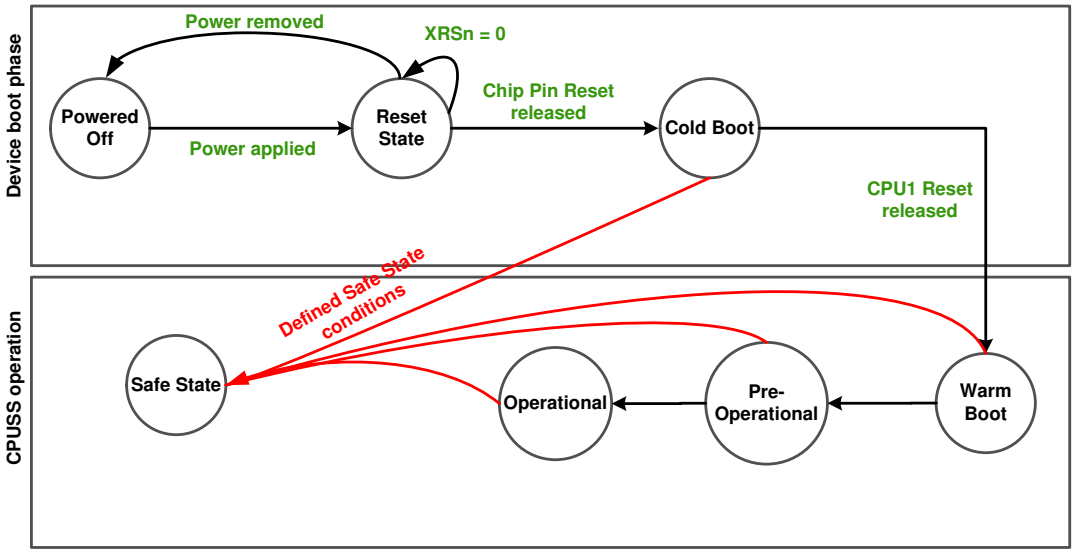

Figure 4-7. TMS320F280013x MCU Device Operating States

4.2.4 Operating States

The C2000 MCU products have a common architectural definition of operating states. These operating states must be observed by the system developer in their software and system-level design concepts. The operating states state machine is shown in Figure 4-7. 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 TMS320F280013x 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 TMS320F280013x MCU) in any of the TMS320F280013x MCU states as response to a system-level fault condition or a fault condition indicated by the TMS320F280013x MCU.
- Reset State – In this state, the device reset is asserted using either the external pins or using any of the internal sources.
- Safe State – In this state, the device is either not performing any functional operations or an internal fault condition is indicated using the device I/O pins.
- Cold Boot - In the cold boot state, the CPU remains powered but in reset. When the cold boot process is completed, the reset of the 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 and proof-test are required to verify 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 the CPU is shown in Figure 4-8.

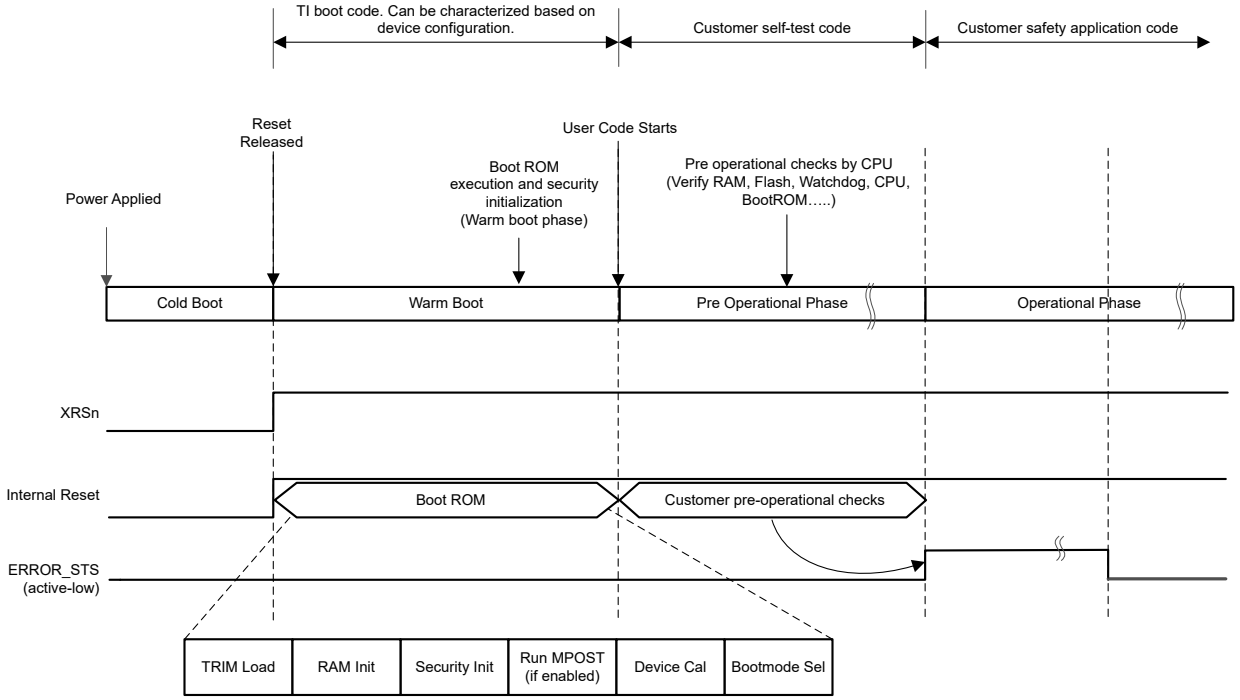

Figure 4-8. TMS320F280013x MCU CPU Start-Up Sequence

4.3 Hardware Component Configuration

The diagnostics libraries designed for the F280013x series of devices is comprised of one library, the SDL. This library is designed to help TI customers, using the F280013x, 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 SDL provides examples for several additional safety mechanisms described in the functional safety manual.

Table 4-1. DC and SCC Targeted for F280013x Diagnostic Libraries

Library	Permanent Fault Diagnostic Coverage (DC)	Systematic Capability Compliance (SCC)	Description
SDL	Examples Only	N/A	The SDL provides examples of several safety mechanisms described in the safety manual

The Software Diagnostic Library (SDL) comprises general example implementations of several safety mechanisms. A subset of SDL modules, listed in *Module to Safety Mechanism Mapping*, have been certified for UL 60730-1 Class-B by UL. The end integrator is expected to study and adapt the provided modules, listed below, into their safety related applications and are responsible for obtaining their own product level third-party certifications.

4.3.1 Assumptions of Use - F280013x 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 F280013x-based safety architecture.

The software support for the various safety mechanisms in the F280013x is provided by the Software Diagnostic Library (SDL), which provides a series of examples of safety mechanisms that are designed to detect permanent faults inside several key elements within the F280013x device.

To determine which versions of the SDL software library have been evaluated for UL 60730-1 Class-B compliance, refer to the UL product iQ database file number E352502 or to your TI sales representative.

The SDL supports safety mechanisms, such as: *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 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.

4.3.2 Operational Details - SDL

The SDL modules are co-hosted onto an F280013x target in order to enable the comprehension of safety in the host application. Therefore, system integrators must fully comprehend all aspects of the associated system constraints imposed by the integration of the SDL to comprehend safety.

4.3.2.1 Operational Details – SDL Module Mapping

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

Note

The evaluation of the SDL for UL 60730-1 Class B compliance includes only the SDL library files located in the '...\f280013x\source' or '...\f280013x\include' directories of the SDL software package. Those modules located in the '...\f280013x\examples' directory are intended for reference and example purposes only and are not to be used as-is. As such, these modules have not been evaluated in this context. For ease of use, the directory location of each module has been listed below in the 'Directory' column.

Table 4-2. Module to Safety Mechanism Mapping

Module Name	Unique Identifier	Directory
STL_CAN_RAM	CAN4, CAN15	Source and Include
STL_CPU_REG	No unique identifier, added for IEC 60730	Source and Include
STL_CRC	NWFLASH5	Source and Include
STL_March	SRAM3	Source and Include
STL_OSC_CT	CLK2	Source and Include
STL_OSC_HR	OTTO1, CLK3	Source and Include
STL_PIE_RAM	PIE3, PIE12	Source and Include
STL_SP	CPU_14	Source and Include
sdl_ex_dcsm_ffi	No unique identifier, demo of freedom from interference using DCSM	Examples
sdl_ex_flash_ecc_test	NWFLASH15	Examples
sdl_ex_flash_prefetch_test	NWFLASH14	Examples
sdl_ex_mcd_test	CLK12	Examples
sdl_ex_ram_access_protect	SRAM10	Examples
sdl_ex_ram_ecc_parity_test	SRAM13, SRAM14, PIE12	Examples
sdl_ex_watchdog	CLK10	Examples

4.4 TMS320F280013x MCU Safety Implementation

4.4.1 Assumptions of Use

The following assumed safety requirements need to be implemented using external components.

- External voltage monitor to supervise the power supply provided to the TMS320F280013x MCU
- External Watchdog timer that can be used for diagnostic purposes
- Components required for taking the system to Safe state, as per the TMS320F280013x MCU Safe state defined in Section 4.2.3.

4.4.2 Example Safety Concept Implementation Options on TMS320F280013x MCU

TMS320F280013x class of devices supports a C28x CPU. The safety functions, which verifies that each safety goal can be met, can be implemented through safety mechanisms, such as *CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping*, *Internal Watchdog (WD)*, and so forth. For common cause failures, such as clock, power, and reset, an external watchdog must be used. Here are some definitions, as used in the context of this document:

- Intended function: Control application implemented on TMS320F280013x (PFC, DCDC, and so on).
- Safety function: Aims to achieve risk reduction and implementation of the safety goals identified from HARA. As a reminder, this document makes no claims with respect to safety standards.
  - Example: To prevent overcurrent, overvoltage, undervoltage, overtemperature, forward and reverse torque, and so on.
- Diagnostic function: Verifies the safety-function operates correctly when required.

5 Description of Safety Elements

This section contains a brief description of the elements on the TMS320F280013x MCU device family, organized based on generic classifications of the hardware parts 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, the latent fault metric requirement from ISO 26262:2018). These measures are called test of diagnostics. Primary diagnostics of the type *software* and *hardware/software* involves the execution of the software on the processing units and using many of the MCU parts, like interconnect, memory (flash, SRAM, and ROM), and TMS320F280013x MCU infrastructure components (clock, power, reset, and JTAG). To verify the integrity of the implemented primary diagnostics and their associated diagnostic coverage values, measures to protect the execution of primary diagnostics on respective processing units must be implemented. TI recommends implementing an appropriate combination of diagnostic tests for parts of the MCU contributing to the successful operation of the processing units. See the respective sections in this safety manual for diagnostics of these parts.

In the event that separate diagnostic test measures exist for a primary diagnostic measure, these measures are mentioned with the respective hardware part.

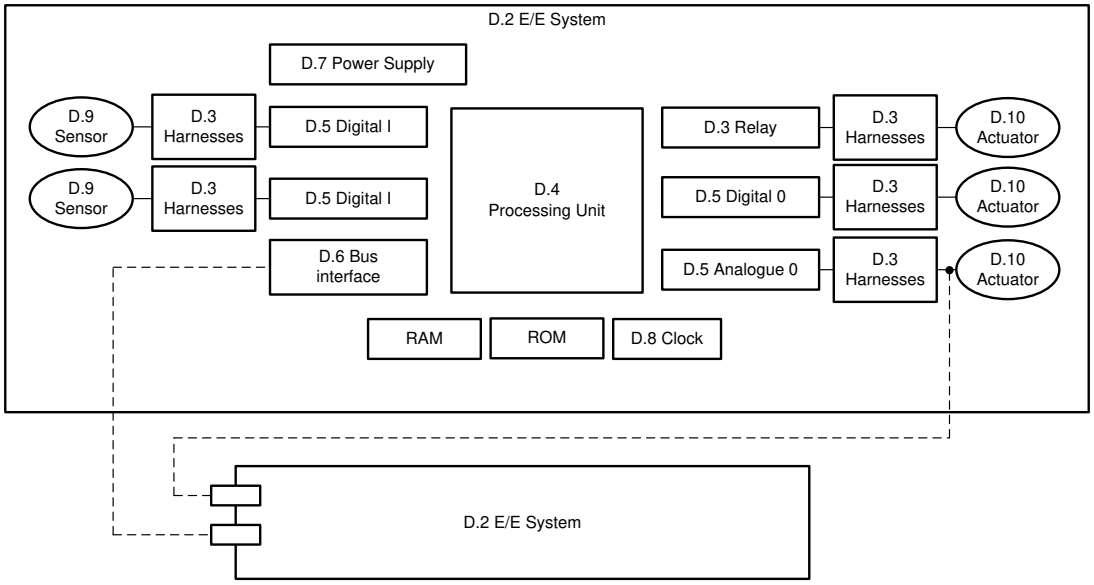

Figure 5-1. Generic Hardware of a System

5.1 TMS320F280013x 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 (1.2V), analog (3.3V), flash (3.3V) and I/O logic (3.3V). 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. TI recommends implementing one voltage supervisor per ganged rail for proper operation of power diagnostics.
- Common mode failure analysis of the external voltage supervisor along with the TMS320F280013x MCU is useful to determine dependencies in the voltage generation and supervision circuitry.
- Customers can consider using TI's TPS6538x power supply and safety companion device for voltage supervision at the system level.

5.1.2 Clock

The TMS320F280013x 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)
- External Monitoring of Clock via XCLKOUT
- Dual-Clock Comparator (DCC)
- Internal Watchdog (WD)
- External Watchdog
- Clock Integrity Check Using CPU Timer
- Clock Integrity Check Using HRPWM
- 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)

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

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

5.1.3 APLL

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

- Clock Integrity Check Using DCC
- PLL Lock Indication
- Internal Watchdog (WD)
- External Watchdog
- External Monitoring of Clock via XCLKOUT

The following tests can be applied as test-for-diagnostics on this module to meet latent fault metric requirements:

- Software Test of DCC Functionality Including Error Tests
- Software Test of PLL Functionality Including Error Tests

5.1.4 Reset

The power-on reset (POR) generates an internal warm reset signal to reset the majority of digital logic as part of the boot process. The warm reset can also be provided at the 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 since the monitored reset signals interact with the internal watchdogs.
- Customers can consider using TI's TPS6538x power supply and safety companion device for reset supervision at the system level.

5.1.5 System Control Module and Configuration Registers

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

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

- 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 Protection for Critical Registers
- Software Test of ERRORSTS Functionality

Note

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

5.1.6 JTAG Debug, Trace, Calibration, and Test Access

The TMS320F280013x 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 easier manufacturing board testing.

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

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

- Hardware Disable of JTAG Port
- Lockout of JTAG Access Using OTP
- 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 Unit (FPU) and Trigonometric Math Unit (TMU) instruction set extensions. This device draws from the best features of digital signal processing, reduced instruction set computing (RISC), microcontroller architectures, firmware, and tool sets. The CPU features include a modified Harvard architecture and circular addressing. The RISC features are single-cycle instruction execution and register-to-register operations. The modified Harvard architecture of the CPU enables instruction and data fetches to be performed in parallel. The CPU does this over six separate address and data buses. The unique architecture of the CPU makes the CPU amenable to integrating safety features external to the CPU but on chip, which provides improved diagnostic coverage.

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

- 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

Note

Measures to mitigate common cause failure in the CPU subsystem: Common-cause failures are one of the important failure modes when a safety-related design is implemented in a silicon device. The contribution of hardware and software dependent failures is estimated on a qualitative basis because no general and sufficiently reliable method exists for quantifying such failures. The system integrator must perform a detailed analysis based on the inputs from ISO 26262-11:2018, Section 4.7 and IEC 61508-2:2010 Annex E (BetaIC method).

5.3 Memory (Flash, SRAM and ROM)

5.3.1 Embedded Flash Memory

The embedded flash memory is a non-volatile memory that is tightly coupled to the C28x CPU. The 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
- Flash Program Verify and Erase Verify Check
- Flash Program/Erase Protection
- Flash Wrapper Error and Status Reporting
- Prevent 0 to 1 Transition Using Program Command
- On-demand Software Program Verify and Blank Check
- Software Read Back of Written Configuration
- CMDWEPROT* and Program Command Data Buffer Registers Self-clear After Command Execution
- ECC Generation and Checker Logic is Separate in Hardware
- Bit Multiplexing in Flash Memory Array
- Software Test of Flash Prefetch, Data Cache and Wait-States
- Internal Watchdog (WD)
- Information Redundancy Techniques

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

- Software Test of ECC Logic
- Auto ECC Generation Override

5.3.2 Embedded SRAM

The TMS320F280013x MCU device family has the following types of SRAMs with different characteristics.

- Dedicated RAM (M0, M1)
- Local shared RAM (LSx RAM)

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 a controller which implements access protection, security related features, and ECC and 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
- 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
- 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

5.3.3 Embedded ROM

The TMS320F280013x MCU device family has the following types of ROMs:

- Boot ROM helps to boot the device and contains functions for security initialization, device calibration, and supporting different boot modes
- Secure ROM functions are not developed to meet any systematic capability compliance (ISO 26262-6:2018/IEC 61508-3:2010) and must not be used in functional safety applications.

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

- 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)

5.4 On-Chip Communication Including Bus Arbitration

5.4.1 Device Interconnect

The device interconnects link the multiple controllers and targets within the device. The device interconnect logic, comprised of static controller selection muxes, dynamic arbiters, and protocol convertors, is required for various bus controllers 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
- Transmission Redundancy
- Hardware Redundancy
- EALLOW and MEALLOW Protection for Critical Registers

5.4.2 Enhanced Peripheral Interrupt Expander (ePIE) Module

The enhanced peripheral interrupt expander (ePIE) module is used to interface peripheral interrupts to the C28x CPU. The ePIE module provides configurable masking on a per interrupt basis. The ePIE 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):

- SRAM Parity
- 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

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

- Software Test of Parity Logic

5.4.3 Dual Zone Code Security Module (DCSM)

The dual code security module (DCSM) is a security feature incorporated in this device. DCSM prevents access and visibility to on-chip secure memories (and other secure resources) to unauthorized persons. DCSM 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
- External Watchdog

5.4.4 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

5.4.5 Timer

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

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

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
- 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 through Hall sensors), elapsed time measurements between position sensor pulses, period and duty cycle measurements of pulse train signals, and decoding current or voltage amplitude derived from duty cycle encoded current and voltage sensors.

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 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

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.

5.5.6 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 Analog I/O

5.6.1 Analog-to-Digital Converter (ADC)

The analog-to-digital converter (ADC) module is used to convert analog inputs into digital values. Results are stored in internal registers for later transfer by CPU. The TMS320F280013x MCU device family products implement up to two 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

Note

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

5.6.2 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 detect signal exception conditions, including high and lLow 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.

This device contains two variants of the CMPSS module: CMPSS and CMPSS_LITE. See the device technical reference manual for details about which features are supported by each variant.

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
- *CMPSS Ramp Generator Functionality Check (Not applicable to CMPSS_LITE)*

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

5.7 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 require an external transceiver to operate on the CAN network. The following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- 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

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, which is 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

5.7.3 Serial Communication Interface (SCI)

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

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

- 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-controller 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

6 Management of Random Faults

For a functional safety critical development it is necessary to manage both systematic and random faults. The TMS320F280013x component architecture includes many functional safety mechanisms, which can detect and respond to random faults when used correctly. This section of the document describes the architectural safety concept for each sub-block of the TMS320F280013x component. The system integrator shall review the recommended functional safety mechanisms in the functional safety analysis report (FMEDA), where available, in addition to this safety manual to determine the appropriate functional safety mechanisms to include in their system. The component data sheet or technical reference manual (if available) are useful tools for finding more specific information about the implementation of these features.

6.1 Fault Reporting

The TMS320F280013x 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 TMS320F280013x 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 must be taken to verify consistency with the system safety concept. The fault indication ordered in terms of severity (device power down being the most severe) is shown in Figure 6-1.

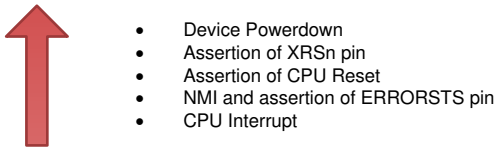

Figure 6-1. Fault Response Severity

- Device 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 TMS320F280013x MCU. From this state, re-entering cold boot to attempt recovery is possible.
- Assertion of XRSn: The XRSn reset can be generated from an internal or external monitor that detects a critical fault having the potential to violate a safety goal. Internal sources generate this fault response when the TMS320F280013x MCU is not able to handle the internal fault condition by itself (for example, CPU1,or controller CPU, is not able to handle NMI by itself). From this state, re-entering cold boot to attempt recovery is possible.
- Assertion of CPU reset: CPU reset changes the state of the CPU from a 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 can 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 TMS320F280013x MCU is equipped with a NMIWD module responsible for generating NMI to the C28x CPU. ERRORSTS pin is also asserted with NMI. Depending on the system level requirements, the fault can be handled either internal to the TMS320F280013x MCU using software or at the system level using the ERRORSTS pin information.
- CPU interrupt: CPU interrupt allows events external to the CPU to generate a program-sequence, context transfer to an interrupt handler where software has an opportunity to manage the fault. The peripheral interrupt expansion (PIE) block multiplexes multiple interrupt sources into a smaller set of CPU interrupt inputs.

6.1.1 Suggestions for Improving Freedom From Interference

The following techniques and safety measures are applicable for improving independence of function when using the TMS320F280013x MCU:

1. Hold peripheral clocks disable 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 and balls.
4. Partition the memory, per the application requirements, to the respective processing units and configure the access protection mechanism for memories for each memory instance so that only the permitted controllers 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 risks resulting from interference from one secure zone to another. For more information, see *Achieving Coexistence of Safety Functions for EV/HEV Using C2000™ MCUs*.
6. Disabling unused sources of SOC inputs to ADC can help avoid interference from unused peripherals to disturb functionality of ADC.
7. To avoid interference from spurious activity on MCU's debug port, JTAG1-Hardware Disable of JTAG Port can be used.
8. 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 detect such interfering failures.
9. MCU resources that support CPU execution, such as memory, interrupt controller, and so forth, can be impacted by resources from lower, safety-integrity, safety functions coexisting on the same MCU. Safety mechanisms such as SRAM11-Access Protection Mechanism for Memories, SRAM16–Information Redundancy Techniques, and SRAM17-CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping are able to detect such interference.
10. Critical configuration registers can be victim to interference from bus controllers on the MCU, which implements lower, safety-integrity functions. These critical configuration registers can be protected by SYS1-Multi-Bit Enable Keys for Control Registers, SYS2-Lock Mechanism for Control Registers, and SYS8-EALLOW Protection for Critical Registers.

6.1.2 Suggestions for Addressing Common Cause Failures

The system Integrator must execute a common cause failure analysis to consider possible dependent and common cause failures on the sub-elements of the TMS320F280013x 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 must 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 TMS320F280013x 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 TMS320F280013x MCU package, including aspects related to the selected I/O multiplexing.

The following must be considered for addressing the common cause failures when using the TMS320F280013x MCU:

1. Redundant functions and safety mechanism can be impacted by common power failure. A common cause failure on a power source can be detected by PWR1-External Voltage Supervisor and PWR2-External Watchdog.
2. In general, a clock source, which is common to redundant functions, must be monitored and any failures on the same clock source can be detected by safety mechanisms. This monitoring is to detect failures and is accomplished by using safety mechanisms, such as CLK1-Missing Clock Detect (MCD), CLK17-Dual-Clock Comparator (DCC), CLK2-Clock Integrity Check Using CPU Timer, CLK5-External Clock Monitoring via XCLKOUT, and CLK8-Periodic Software Read Back of Static Configuration Registers. To specifically avoid common clock failures affecting the Internal Watchdog (WD) and CPU, TI recommends using either INTOSC2 or X1/X2 as the clock source to PLL.
3. Failure of the common reset signal to redundant functions can be detected by RST1-External Monitoring of Warm Reset (XRSn) and RST2-Reset Cause Information.
4. Common cause failures on the interconnect logic can impact both redundant functions and functional safety mechanism in the same way. In addition to other safety mechanisms, INC1-Software Test of Function Including Error Tests can be implemented to detect faults on interconnect logic.
5. Common cause failures can impact two functions used in a redundant way. In the case of communication peripherals, module specific Information Redundancy Techniques Including End-to-End Safing can be implemented to detect common cause failures, for example, CAN2-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, and I2C3-Information Redundancy Techniques Including End-to-End Safing.
6. Use different voltage references and SOC trigger sources for ADC (see Section 6.3.5.8).
7. Use nonadjacent GPIO pins from different groups when implementing hardware redundancy for GPIO pins.

6.2 Functional Safety Mechanism

This section includes a description of the different types of functional safety mechanisms that are applied to the design blocks of the TMS320F280013x component.

The functional safety mechanism categories are defined as follows:

Component Hardware Functional Safety Mechanisms	A safety mechanism that is implemented by TI in silicon which can communicate error status upon the detection of failures. The safety mechanism may require software to enable its functionality, to take action when a failure is detected, or both.
Component Hardware and Software Functional Safety Mechanisms	A test recommended by TI which requires both, safety mechanism hardware which has been implemented in silicon by TI, and which requires software. The failure modes of the hardware used in this safety mechanisms are analyzed or described as part of the functional safety analysis or FMEDA. The system implementer is responsible for analyzing the software aspects for this safety mechanism.
Component Software Functional Safety Mechanisms	A software test recommended by TI. The failure modes of the software used in this safety mechanism are not analyzed or described in the functional safety analysis or FMEDA. For some components, TI may provide example code or supporting code for the software functional safety mechanisms. This code is intended to aid in the development, but the customer shall do integration testing and verification as needed for their system functional safety concept.
System Functional Safety Mechanisms	A safety mechanism implemented externally of this component. For example an external monitoring IC would be considered to be a system functional safety mechanism.
Test for Safety Mechanisms	This test provides coverage for faults on a safety mechanism only. It does not provide coverage for the primary function.
Alternative Safety Mechanisms	An alternative safety mechanism is not capable of detecting a fault of safety mechanism hardware, but instead is capable of recognizing the primary function fault (that another safety mechanism may have failed to detect). Alternate safety mechanisms are typically used when there is no direct test for a safety mechanism.

6.3 Description of Functional Safety Mechanisms

This section provides a brief summary of the diagnostic mechanisms available on the TMS320F280013x MCU device family. The diagnostic mechanisms are arranged as per the device portioning, given in Figure 5-1. At places where the safety mechanism is applicable for more than one component, the safety mechanism is placed at an appropriate place based on the applicable use case scenario. For a detailed description, or implementation details for a diagnostic, see the device-specific technical reference manual.

6.3.1 TMS320F280013x MCU Infrastructure Components

6.3.1.1 Clock Integrity Check Using DCC

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

6.3.1.2 Clock Integrity Check Using CPU Timer

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

6.3.1.3 Clock Integrity Check Using HRPWM

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

6.3.1.4 EALLOW Protection for Critical Registers

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

6.3.1.5 External Monitoring of Clock via XCLKOUT

The TMS320F280013x MCU device family provides the capability to export selected internal clocking signals for external monitoring. This feature can be configured through 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 through software.

6.3.1.6 External Monitoring of Warm Reset (XRSn)

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

6.3.1.7 External Voltage Supervisor

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

6.3.1.8 External Watchdog

External watchdog helps to reduce common mode failure, since the clock, reset, and power utilized 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 must include a combination of temporal and logical monitoring of program sequence [IEC 61508-7:2010, clause A.9.3], or other appropriate methods, so that high diagnostic effectiveness can be claimed.

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

6.3.1.10 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.

6.3.1.11 Lockout of JTAG Access Using OTP

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

6.3.1.12 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. The user programs an upper bound and lower bound to create a time window during which the software must provide a predetermined WDKEY to the watchdog. Failure to receive the correct response within the time window, or an incorrect WDKEY, triggers an error response. The WWD can issue either a warm system reset or a CPU maskable interrupt upon detection of a failure. Normal WD operation is enabled by default after reset. For details of programming the internal watchdogs, see the device-specific technical reference manual.

To avoid a common cause failure of a clock input to both the internal watchdog (WD) and CPU, TI recommends selecting either INTOSC2 or X1/X2 as the clock source to main PLL.

6.3.1.13 Lock Mechanism for Control Registers

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

6.3.1.14 Missing Clock Detect (MCD)

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

6.3.1.15 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 can 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.

6.3.1.16 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.

6.3.1.17 Multi-Bit Enable Keys for Control Registers

Some modules include features to support avoidance of an 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.

6.3.1.18 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. The output connection can be enabled by setting the ENABLE bit in the TSNSCTL register. Refer to the device data sheet to determine the relevant ADC channel.

6.3.1.19 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 verify that all pending interrupts are serviced by reading the PIE flag registers (PIE_CTRL_REGS.PIEIFRx.all) and the peripheral interrupt flag registers.
- While serving the interrupt, the ISR routine can check for interrupt flag in peripherals and PIE module to verify that correct interrupt is being serviced.

6.3.1.20 Peripheral Clock Gating (PCLKCR)

Peripherals can be clock gated on a *per peripheral* basis. This function can be used to disable unused features so 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. Locking a particular configuration to avoid inadvertent writes is possible.

6.3.1.21 Peripheral Soft Reset (SOFTPRES)

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

6.3.1.22 Software Test of Reset - Type 1

A software test for detecting basic functionality, as well as errors for reset sources and reset logic, can be implemented. Each of the reset sources (including peripheral resets, DEV_CFG_REGS.SOFTPRESx), except POR, can be generated internally. The basic functionality of each reset source can be checked by triggering the reset and verifying that only the intended logic is reset. Additionally, the SIMRESET configuration for SYSRS or XRS assertion through software write may be used for this test.

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 a peripheral-specific software test of function, including error tests, see the device-specific safety mechanism listed for the peripheral.

6.3.1.23 PLL Lock Profiling Using On-Chip Timer

Clock setup for the TMS320F280013x MCU device family includes selecting the appropriate clock source, configuring the PLL multiplier, waiting for the lock status, and switching the clock to the PLL output once the internal lock status is set. The time required for the PLL lock sequence can be profiled using on-chip timer to detect faults in the PLL wrapper logic. Once the PLL is locked, the frequency of the output clock can be checked by using the following:

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

6.3.1.24 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.

6.3.1.25 Software Read Back of Written Configuration

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

6.3.1.26 Software Test of ERRORSTS Functionality

As indicated in Figure 4-6, the ERRORSTS pin is an integral part of an MCU safety concept that is used for indicating a critical error occurring in the MCU to an external system. Proper functioning of the ERRORSTS pin and error handling of the system external to the MCU can be checked by asserting the ERRORSTS pin and by generating an error condition using one of the provided software options (for example, asserting CLOCLKFAIL NMIFLG by updating the NMIFLGFRC.bit.CLOCKFAIL). Error response, diagnostic testability, and any necessary system requirements are defined by the system integrator.

6.3.1.27 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 the issue of a missing clock NMI and sets a missing clock status flag (MCDCR.MCLKSTS).

6.3.1.28 Software Test of Watchdog (WD) Operation

A basic test of the internal watchdog operation can be performed through software, including checking the error response by configuring the expected upper and lower threshold values for servicing WDKEY and verifying that the expected servicing of WDKEY occurs. 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.

6.3.1.29 Dual-Clock Comparator (DCC) - Type 2

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

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

6.3.1.30 PLL Lock Indication

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

6.3.1.31 Software Test of DCC Functionality Including Error Tests

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

6.3.1.32 Software Test of PLL Functionality Including Error Tests

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

6.3.1.33 Interleaving of FSM States

Main control FSM includes a hamming distance of two to verify that any single bit flip does not cause the state machine to transition into another valid state. The FSM defaults to the IDLE/INIT state in case of any single bit flip. Since the PLLEN and other control bits from the SYSCTRL remain valid, the FSM (is expected to) relocks and continues. No error is generated for the bit fail scenario.

6.3.1.34 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 the XRSn is pulled low. The XRSn remains in reset until the voltage returns to the operational range. The BOR is enabled by default.

6.3.2 Processing Elements

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

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

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

- Illegal instruction
- *TMS320C28x Extended Instruction Sets. Technical Reference Manual*

6.3.2.2 Stack Overflow Detection

A stack overflow, in a safety application, generally produces a catastrophic software crash due to data corruption, lost return addresses, or both. Hence, detecting an impending stack overflow is important. The capability exists on C2000 devices that, when properly configured, allows for runtime detection of a stack overflow before the stack overflow occurs. For more information, see *Online Stack Overflow Detection on the TMS320C28x DSP*. Detection of an impending stack overflow triggers a maskable interrupt. The programmed error response, and any necessary software requirements, are defined by the system integrator.

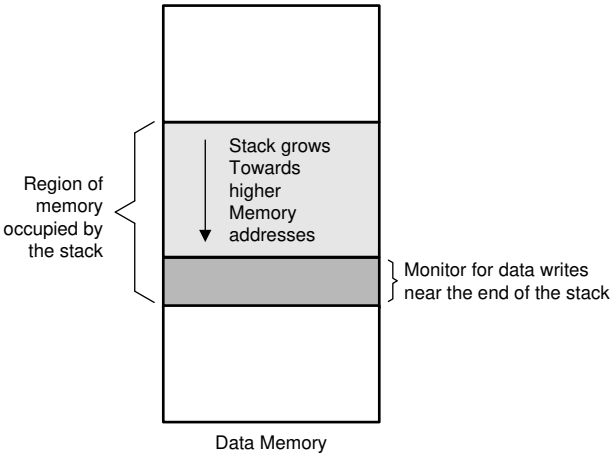

Figure 6-2. Stack Overflow Monitoring

6.3.2.3 CRC Check of Static Memory Contents

A software implementation of cyclic redundancy check (CRC) operations for the F280013x MCU device family is provided in the software diagnostic library (SDL) as module STL_CRC. 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.

6.3.3 Memory (Flash, SRAM and ROM)

6.3.3.1 Bit Multiplexing in Flash Memory Array

The flash modules implemented in the TMS320F280013x MCU device family have a bit multiplexing scheme implemented, so 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, the faults manifest as multiple single-bit faults. Since the SECDED flash ECC can correct a single-bit fault and can detect a double-bit fault in a logical word, this scheme improves the usefulness of the flash ECC diagnostic. Bit multiplexing is a feature of the flash memory and cannot be modified by the software.

6.3.3.2 Bit Multiplexing in SRAM Memory Array

The SRAM modules implemented in the TMS320F280013x 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, the faults manifest as multiple single-bit faults. The SECDED SRAM ECC diagnostic can correct a single-bit fault and detect a double-bit fault in a logical word. Similarly, the SRAM parity diagnostic can detect single-bit faults. This scheme improves the usefulness of the SRAM ECC and parity diagnostic. Bit multiplexing is a feature of the SRAM and cannot be modified by the software.

6.3.3.3 Data Scrubbing to Detect/Correct Memory Errors

Bus owners can be configured to provide fake reads to the memory (provided a particular bus controller 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. In the case of correctable errors, for both SRAMs and flash, an interrupt is issued once the count exceeds the preset threshold. In the case of uncorrectable errors, an NMI is issued.

6.3.3.4 Flash ECC

The on-chip flash memory is supported by a 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 the 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 generate an NMI and the ERRORSTS pin is asserted. A count of the corrected errors (single-bit data errors) is monitored in the memory error registers and an interrupt is generated once the count exceeds the programmed threshold. The corrupted memory address of the last error location is also logged in the memory error registers.

6.3.3.5 Flash Program Verify and Erase Verify Check

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

6.3.3.6 Flash Program/Erase Protection

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

6.3.3.7 Flash Wrapper Error and Status Reporting

During and after execution of all commands (except ClearStatus), the flash wrapper updates the STATCMD register. CMDINPROGRESS and CMDDONE indicators are present to indicate the execution status of a

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

6.3.3.8 Prevent 0 to 1 Transition Using Program Command

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

6.3.3.9 On-Demand Software Program Verify and Blank Check

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

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

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

6.3.3.11 ECC Generation and Checker Logic is Separate in Hardware

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

6.3.3.12 Auto ECC Generation Override

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

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

6.3.3.13 Software Test of ECC Logic

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

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

For additional details on implementing this diagnostic for SRAM and FLASH memory, see the *Application Test Hooks for Error Detection and Correction* and *Mechanism to Check the Correctness of ECC Logic* sections in the *TMS320F280013x Microcontrollers Technical Reference Manual*.

6.3.3.14 Software Test of Flash Prefetch, Data Cache, and Wait-States

Once enabled, prefetch logic keeps fetching the next 128-bit row (4 x 32-bit words) from flash bank. On detecting a discontinuity, the prefetch buffer is cleared. A software test can be performed to ascertain the proper behavior of this logic. The following sequence of operations 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 must be less than 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, the number of rows fetched from the flash bank varies with the code alignment within the flash bank. Hence, users need to verify that the prefetch logic test function is properly aligned and located in a particular location within flash to generate timing behavior that does not vary from compile to compile.

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

6.3.3.15 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 (for example, fetch and write) to individual RAM blocks from the CPU. There is no protection for read accesses, therefore, reads are always allowed. To identify conditions when the 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 application threads within the same CPU. This capability helps support freedom from interference requirements, which are required by some applications.

6.3.3.16 SRAM ECC

Selected on-chip SRAMs support SECDED ECC diagnostic with separate ECC bits for data and address. For the specific address ranges that support ECC, see the TMS320F280013x MCU device-specific data sheet. In the 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 the CPU after the data integrity check. The data and address interconnects from SRAM to the CPU are not protected using ECC. Detected correctable errors are corrected and monitoring the number of corrected errors is possible. The SRAM wrapper can be configured to trigger an interrupt once the number of corrected errors crosses a threshold. Uncorrectable SRAM errors trigger an NMI and the ERRORSTS pin is asserted. The ECC logic for the SRAM is enabled at reset. For more information regarding memories supporting ECC, see the TMS320F280013x MCU device-specific data sheet.

6.3.3.17 SRAM Parity

Selected on-chip SRAMs support parity diagnostic with separate parity bits for data and address. For the specific address ranges that support parity, see the device-specific data sheet. In the parity scheme, a 3-bit code word is used to store the parity data calculated independently for each 16 bit of data and for address. The parity generation and check logic for the SRAM is located in the SRAM wrapper. The parity is checked directly at the memory output and data is sent to CPU after the data integrity check. The data and address interconnect from SRAM to the CPU is not protected using parity. SRAM parity errors trigger an NMI and the ERRORSTS is asserted. The parity logic for the SRAM is enabled at reset. For more information regarding memories supporting parity, see the TMS320F280013x MCU device-specific data sheet.

6.3.3.18 Software Test of Parity Logic

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

For additional details on implementing this diagnostic for SRAM, see the *Application Test Hooks for Error Detection and Correction* section in *TMS320F280013x Microcontrollers Technical Reference Manual*.

6.3.3.19 Software Test of SRAM

Testing the integrity of SRAM (bit cells, address decoder, and sense amplifier logic) is possible using the CPU. Based on the safety requirement, this test can be performed at start-up or during application time.

6.3.3.20 Memory Power-On Self-Test (MPOST)

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

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

6.3.3.21 ROM Parity

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

6.3.4 On-Chip Communication Including Bus-Arbitration

6.3.4.1 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, using one of the other counters as a diagnostic on the first is possible by using periodic check through software of the counter values in the two timers. The second CPU timer 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.

6.3.4.2 Maintaining Interrupt Handler for Unused Interrupts

The TMS320F280013x 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 and enabling the unused interrupts and returning to the application in the interrupt service routine (ISR), and so forth. Receiving an interrupt that is not used in the application can be an early indication of some faulty scenarios within the TMS320F280013x MCU. Hence, TI highly recommends enabling all the interrupts and configuring the ISR with a common routine for logging or error handling.

6.3.4.3 Power-Up Pre-Operational Security Checks

During the device boot, the device goes through various phases, as indicated in Figure 4-7. In the pre-operational phase (before starting the application), the application code is expected to perform a set of checks to verify correct initialization of device security, which includes checks to confirm correct link pointer settings, CRC lock settings, correct partitioning of secure RAM blocks and flash sectors (Grab Bits), settings for execute-only protection for secure RAM blocks and flash sectors, and correct settings for boot configuration. Before starting the execution of downloaded code, users must check the integrity of the code using the CRC function. Once pre-operational checks are successfully completed with expected results, the device can enter the application phase.

6.3.4.4 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 are 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 three values (bit-wise voting logic). Any error in the resolution of the final link pointer value sets the Zx_LINKPOINTERERR register.

6.3.4.5 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 which input sources were triggered. These flag registers can be periodically read to ascertain that no ePWM trips, ePWM syncing, or GPIO output signaling is missed.

6.3.4.6 Software Test of ePIE Operation Including Error Tests

A software test for testing the basic functionality of the ePIE can be implemented, as well as failure modes such as continuous interrupts, no interrupts, and crossover interrupts. Such testing can be based on generating the interrupts from the peripherals and verifying that interrupts are 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.

6.3.5 Digital I/O

6.3.5.1 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.

6.3.5.2 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, failures in the ePWM output, such as stuck-at fault or frequency or duty cycle change, results in disturbance to control loop parameters or variables, leading to conditions such as overvoltage, overcurrent, or overtemperature. By monitoring characteristics of these control loop parameters implemented at the application-level, faults in the ePWM module can be detected.

6.3.5.3 ePWM Fault Detection Using X-BAR

A combination of ePWM output feedback to the input X-BAR, the GPIO inversion logic, and the digital compare (DC) sub-module of ePWM can implement simple, but effective, anomaly checks on the PWM outputs, such as signal cross over. This feature can be used to trip the PWM and enter a Safe state if any anomaly is detected

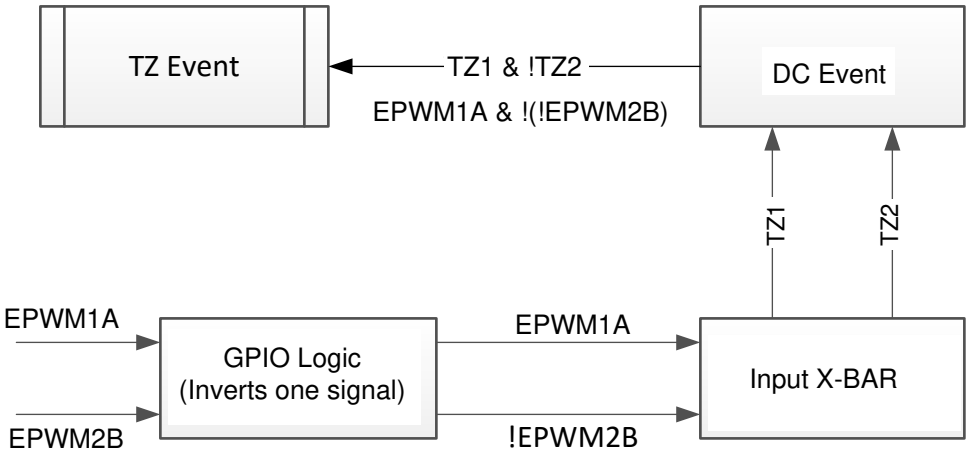

Figure 6-3. ePWM Fault Detection Using X-BAR

6.3.5.4 ePWM Synchronization Check

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

6.3.5.5 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 systems. In such applications, monitoring eQEP outputs for saturation, zero value, or out-of-range is possible based on the application requirement. While estimating the speed and position of rotating machinery, the application can set bounds on the measured speed and position based on the operating profile. Online monitoring of periodic interrupts from eQEP can also be performed for improved diagnostic coverage based on the application profile.

6.3.5.6 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 quadrature clock event (pulse) resets the watchdog timer. If no quadrature clock event is detected until a period match, then the watchdog timer times out and the watchdog interrupt flag is set. The timeout value is programmable through the watchdog period register.

6.3.5.7 eQEP Software Test of Quadrature Watchdog Functionality

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

6.3.5.8 Hardware Redundancy

Hardware redundancy techniques can be applied through 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) TMS320F280013x MCU.

In the case of peripherals like GPIO, X-BAR, ePWM, 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 through internal or external comparators), input comparison, or voting (comparison of independent inputs to verify compliance with a defined tolerance range for time and value). In such scenarios, the system can be designed so that the failure of one input or output does not cause the system to go into a dangerous state. While servicing the error conditions (for example, redundancy conditions), as in two redundant sources tripping the PWM, always read-back the status flags and verify that both sources are active while tripping and thus providing latent fault coverage for the trip logic.

In the case of peripherals like ADC and eCAP, hardware redundancy can be implemented by having multiple instance of the peripheral sample the same input and simultaneously perform the same operation followed by a cross-check of the output values.

In the case of communication peripherals like I2C and SCI, hardware redundancy during signal reception can be implemented by having multiple instances of the peripheral receive the same data followed by comparison to verify data integrity. Hardware redundancy during transmission can be employed by having a completely 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 a data integrity check.

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

While implementing hardware redundancy for ADC modules, additional care must be taken to verify common-cause failures do not impact both instances in same way. Reference voltage sources, which are configured for each redundant module instance, must be independent. Additionally, ADC SOC trigger sources used for redundant ADC instances must be configured to different ePWM module instances.

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

6.3.5.9 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 150ps. 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. The MEP step size increases with decreasing voltage and increasing temperature and decreases with increasing voltage and decreasing temperature. Applications that use the HRPWM feature must use the 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 most appropriate MEP scale-factor value for any operating condition. TI provides a C-callable library containing one SFO function that utilizes this hardware and determines the 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.

6.3.5.10 Information Redundancy Techniques

Information redundancy techniques can be applied through software as an additional runtime diagnostic. To provide diagnostic coverage for network elements outside the TMS320F280013x MCU (wiring harness, connectors, transceiver) end-to-end safety mechanisms are applied. These mechanisms can also provide diagnostic coverage inside the TMS320F280013x MCU.

In the case of processing elements (CPU), this refers to multiple executions of the code and software based cross checking to verify correctness. The multiple execution and result comparison can be based on either the same code executed multiple times or the implementation of diversified software code. For details regarding the implementation, see the ISO 26262-5:2018, D.2.3.4.

Typical control applications involve measuring three-phase voltage and current. These values are either sampled directly using the on chip ADC or sent to the TMS320F280013x MCU by external sensors, which are captured using eCAP, and so forth. In such scenarios, the correlation between input signals can be used to check the integrity (for example, if the three phase voltage, 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 use. Care must be taken to avoid compiler optimizing code containing redundant data and programs. Safety programs in flash can be copied to SRAM and executed after performing a CRC check against a pre-calculated *golden* CRC value.

6.3.5.11 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, on the board, or internally, through the 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. The XINTxCTR (counter of XINT module), capture mode of eQEP and DCCAP (PWM event filter unit), can also be used to detect rising and falling edges of the PWM and extract the time-stamping information. This information can be further used to build additional diagnostics.

6.3.5.12 Monitoring of ePWM by ADC

The ePWM outputs can be monitored for proper operation by ADC using board-level feedback, as indicated in Figure 6-4. The technical details for implementing such a loopback, like signal resolution and so forth, is provided in *Using PWM Output as a Digital-to-Analog Converter on a TMS320F280x Digital Signal Controller* application note. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

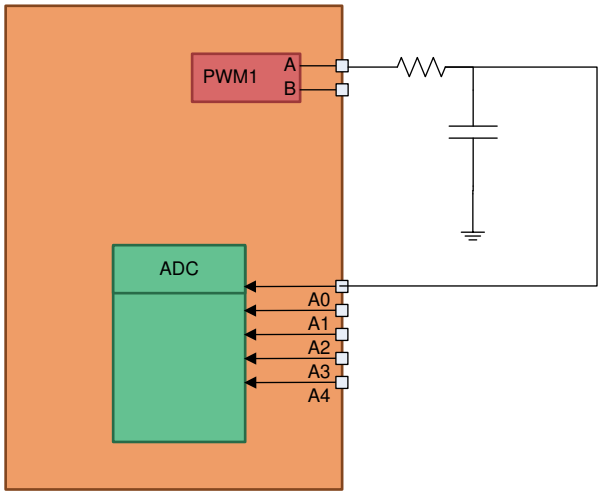

Figure 6-4. Monitoring of ePWM by ADC

6.3.5.13 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 of periodic interrupts and events can be employed 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, such as periodic generation of ADC start-of-conversion (SOC): The 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.

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

6.3.5.14 Software Test of Function Including Error Tests

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

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

- Software tests of the 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 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 the enable (XINTxCR.ENABLE) functionalities as well.

- eCAP 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, 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 6.3.3.14.
- 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 appropriate stimulus using ePWM and observing the response using one of the capture (time stamping) modules (eCAP, XINT, eQEP, and so forth). TI recommends covering the various register values associated with application configuration while performing the software test. Due to the regular linear nature of the various sub-modules, getting high coverage is possible using a software test.
- A software test of SRAM wrapper logic must provide diagnostic coverage for arbitration between various controllers 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 6.3.3.19).
- The interconnect (INC) functionality can be tested by writing complimentary data-patterns, such as 0xA5A5, 0x5A5A, or other common test cases, to the processing units in the CPU and reading those patterns back from the registers of the various IPs connected through different interconnect bridges. The read-back data can be compared with expected *golden* values to verify a 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. 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 and PPB results obtained can be cross checked against the expected value to verify proper operation. Extreme corner values of the ADC used in an 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 that are read back and compared to expected values.
- 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 data-patterns are read back and compared against the expected values. Features of the CMPSS module, such as ramp decrement, can be checked for counting down of RAMPDLYA after RAMPDLYA is loaded from RAMPDLYS by a rising PWMSYNC signal. Always verify 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 the registers to a variety of SAMPWIN (sample window) and THRESH (majority voting threshold) values, and then verifying COMPHSTS/COMPLSTS changes with changes in the filter output. An applicable range of filter-clock, pre-scaler values (CTRIPLFILCLKCTL) can be exercised to verify that the filter samples correctly.
- The general operation of the CPU timers can be tested through a software test by loading 32-bit counter register TIMH from period register PRDH. The TIMH counter register then starts decrementing on every clock cycle. When the counter reaches zero, a timer-interrupt output generates an interrupt pulse. While testing the timer functionality, vary the timer prescale counter (TPR) value and input clocks by selecting the clock source as SYSCLK, INTOSC1, INTOSC2, or XTAL. Interrupt generation can be tested at the end of each timer period. 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 verify that all access, other than execute access, is blocked.

6.3.5.15 QMA Error Detection Logic

The QEP Mode Adapter (QMA) is designed to extend the C2000 eQEP module capabilities to support the additional modes described in the *QMA Module* section of the *TMS320F280013x Real-Time Microcontrollers Technical Reference Manual*. The QMA module has error-detection logic to detect illegal transitions on EQEPA and EQEPB input signals. The error and interrupt of the QMA module are integrated inside the eQEP module.

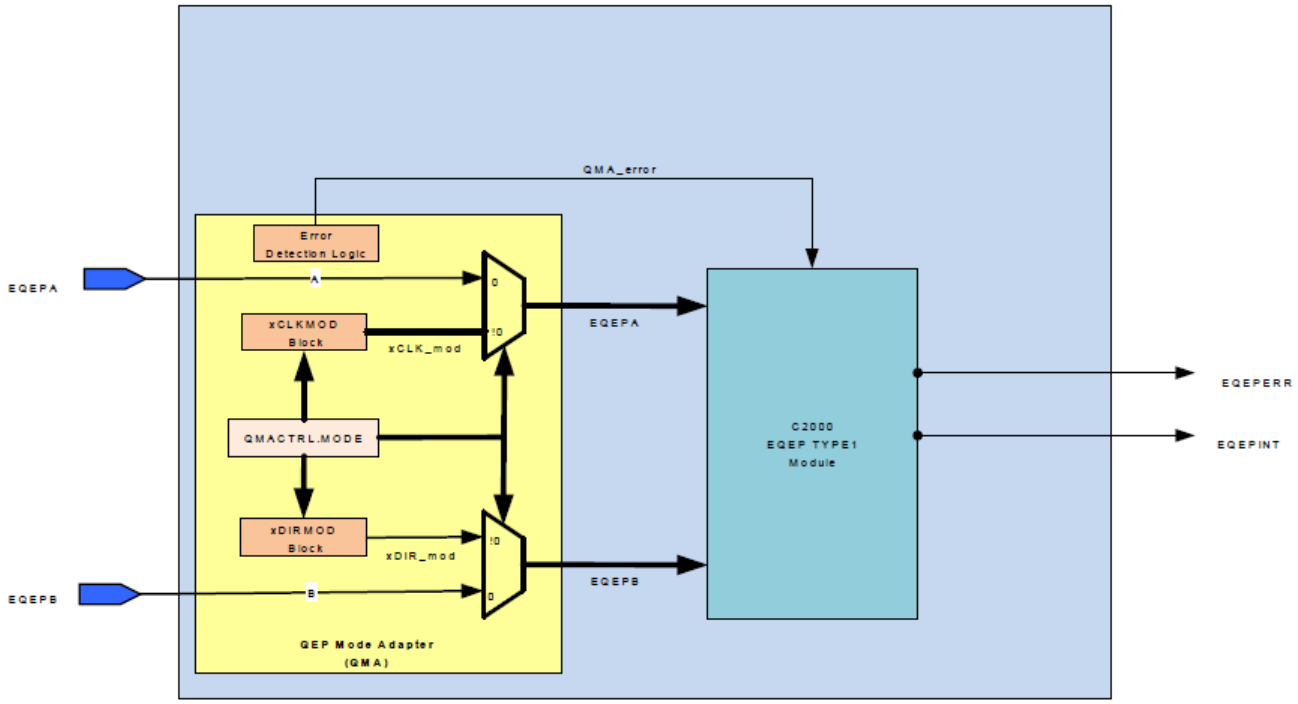

Figure 6-5. QMA Module Block Diagram

6.3.6 Analog I/O

6.3.6.1 ADC Information Redundancy Techniques

Information redundancy techniques can be applied through software for providing runtime diagnostic coverage on ADC conversions. Redundancy techniques can be applied by taking multiple conversions on same ADC followed by a 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.

6.3.6.2 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 a plausibility check (for example, value falls in an expected range) of the converted values, can be performed using some of the built in hardware mechanisms available with the device. A plausibility check of the input signal can be checked with the help of a comparator by setting the proper high and low threshold values. The plausibility check of converted results can be checked using the ADC post processing block.

6.3.6.3 ADC Signal Quality Check by Varying Acquisition Window

External signal sources vary in their ability to drive an analog signal quickly and effectively. To achieve rated resolution, the signal source must charge the sampling capacitor in the ADC core to within 0.5 LSBs of the signal voltage. The acquisition window is the amount of time the sampling capacitor is allowed to charge and is configurable for SOCx by the ADCSOCxCTL.ACQPS register. This configurable parameter can be 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 the ADC using the preset ACQPS configuration and an ACQPS configuration set higher than the preset configuration. The results obtained must be within a pre-defined range determined by the application and the ADC specification parameters.

6.3.6.4 CMPSS Ramp Generator Functionality Check

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

6.3.6.5 DAC to ADC Loopback Check

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

For more information on the CMPSS DAC channels that can be sampled by the ADC without external, board-level connections, see the device-specific data sheet or technical reference manual. Additionally, the input signal to the ADC must not be driven by any other sources while the test is performed.

6.3.6.6 Opens/Shorts Detection Circuit for ADC

The open/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 6-6. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

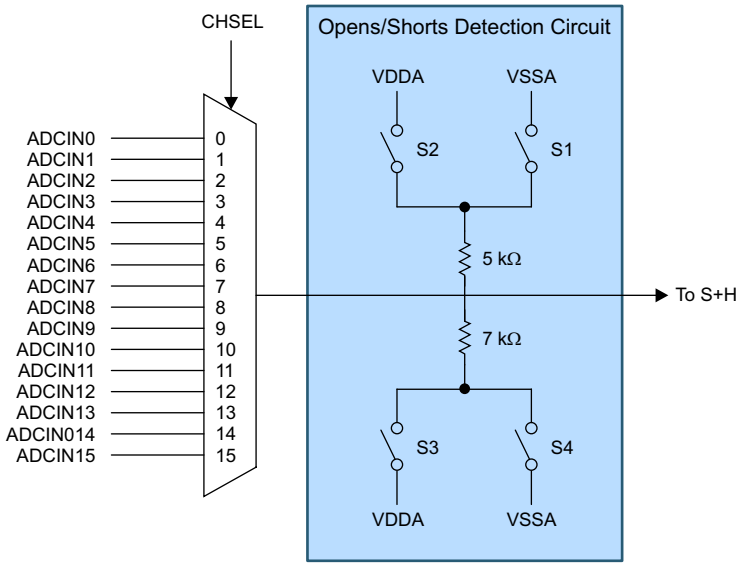

Figure 6-6. ADC Open-Shorts Detection Circuit

The circuit can be operated by writing a value to the DETECTCFG field in the ADCOSDETECT register. This function causes 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 are given in Table 6-1. For additional details on implementing this diagnostic, see the *Opens/Shorts Detection Circuit (OSDETECT)* section in the *TMS320F280013x Real-Time 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

6.3.6.7 Disabling Unused Sources of SOC Inputs to ADC

The ADC SOC (start of conversion) signal input to the ADC module can be triggered by multiple sources, including software, CPU Timers, GPIO, ePWM module instances, and so forth. A fault originating from a peripheral not used in implementing the safety function and cascading into the ADC can cause interference. To achieve freedom from interference, TI recommends that the application only enables the required SOC triggers. This practice avoids faults that originate from an outside source impacting the functionality of the ADC.

6.3.7 Data Transmission

6.3.7.1 Information Redundancy Techniques Including End-to-End Safing

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

To provide diagnostic coverage for network elements outside the TMS320F280013x MCU (wiring harness, connectors, and transceiver), end-to-end safety mechanisms are applied. These mechanisms can provide diagnostic coverage inside the TMS320F280013x MCU. There are many different schemes applied, such as additional message checksums, redundant transmissions, time diversity in transmissions, and so forth. Most commonly, checksums are added to the payload section of a transmission to verify the correctness of a transmission. The checksums, sequence counter, and timeout expectation (or time stamp) are applied with any protocol-level parity and checksums. As these schemes are generated and evaluated by the software, at either end of the communication, the whole communication path is safed, resulting in end-to-end safing.

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

6.3.7.2 Bit Error Detection

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

6.3.7.3 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 is flagged. Error response and any necessary software requirements are defined by the system integrator.

6.3.7.4 DCAN Acknowledge Error Detection

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

6.3.7.5 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 flags a form error. The error response and any necessary software requirements are defined by the system integrator.

6.3.7.6 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 values in the bit stream to transmit, the transmitter automatically inserts a complementary bit into the actual transmitted bit stream. If a sixth consecutive equal bit is detected in a received segment that was to have been coded by bit stuffing, the DCAN module flags a stuff error. The error response and any necessary software requirements are defined by the system integrator.

6.3.7.7 Software Test of Function Including Error Tests Using EPG

The embedded pattern generator (EPG) can be used to check the functionality of DCAN by driving a known pattern on the receive input pin and comparing the received message from CAN. EPG loopback can be used to inject errors on the receive line and check the CAN diagnostic CRC logic operation which is indicated by an error interrupt. Refer to the device-specific technical reference manual for implementing loopback between EPG and DCAN.

6.3.7.8 I2C Access Latency Profiling Using On-Chip Timer

Each I2C message takes a fixed number of system clock cycles to complete a transaction. The controller can detect the completion of a transaction based on a message acknowledge signal from the target. The on-chip timer module can profile the time required for completing each transaction.

6.3.7.9 I2C Data Acknowledge Check

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

6.3.7.10 Parity in Message

This module supports insertion of a parity bit into the data payload of every outgoing message by hardware. The evaluation of incoming message parity is supported by hardware and detected errors generate an interrupt to the CPU.

6.3.7.11 SCI Break Error Detection

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

6.3.7.12 Frame Error Detection

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

6.3.7.13 Overrun Error Detection

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

6.3.7.14 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 better results, any tests of the functionality must include the I/O loopback.

6.3.7.15 SPI Data Overrun Detection

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

6.3.7.16 Transmission Redundancy

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

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

A Summary of Safety Features and Diagnostics

Table A-1. Summary Table Legend

Unique Identifier	Identifier Used to Reference the Contents
Safety Feature or Diagnostic	Safety feature
Usage	Each test listed in this chart can be one of two types. A 'diagnostic' test or a 'test for diagnostic'. Diagnostic: Provides coverage for faults on a primary function of the device. It can, in addition, provide fault coverage on other diagnostics, and can therefore be 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. Its only purpose is to provide fault coverage on other diagnostics. Fault Avoidance: This is typically a feature used to improve the effectiveness of a related diagnostic.
Diagnostic Type	Hardware: A diagnostic which is implemented by TI in silicon and can communicate error status upon the detection of failures. It can require software to enable the diagnostic and 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 can 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 software that must be created by the software implementer. System: A diagnostic implemented externally on 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 must be enabled by software. (iv) On demand (software defined): Software or hardware-software safety mechanism that gets activated in the diagnostic test interval by the software. (v) System defined: Implemented by the system.
Test Execution Time	This column lists the time required for this diagnostic to complete.
Action on Detected Fault	The response this diagnostic takes when an error is detected. For software-driven tests, this action is often software implementation-dependent.
Error Reporting Time	Typical time required for diagnostic to indicate a detected fault to the system. For safety mechanisms where fault detection time is known, this value is indicated. For software-driven tests, this time is often software implementation-dependent.

Table A-2. Summary of Safety Features and Diagnostic

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
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
	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)	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	CLK17	Dual-Clock Comparator (DCC) - Type 2	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)
APLL	APLL1	Clock Integrity Check Using DCC	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)
	APLL2	PLL Lock Indication	Diagnostic	Hardware	Continuous - Enabled by software	Software defined	Software defined	Software defined
	APLL4	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	APLL5	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	APLL6	Software Test of DCC Functionality Including Error Tests	Test for diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	APLL7	External Monitoring of Clock via XCLKOUT	Diagnostic	System	System defined	System defined	System defined	System defined
	APLL10	Software Test of PLL Functionality Including Error Tests	Test for diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	APLL11	Interleaving of FSM States	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Reset	RST1	External Monitoring of Warm Reset (XRSn)	Diagnostic	System	System defined	System defined	System defined	System defined
	RST2	Reset Cause Information	Fault avoidance	Hardware - Software	On demand (Software defined)	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	RST4	Glitch Filtering on Reset Pins	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	RST5	NMIWD Shadow Registers	Fault avoidance	Hardware - Software	On demand (Software defined)	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	RST6	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	RST7	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	RST8	NMIWD Reset Functionality	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset	Software defined
	RST9	Peripheral Soft Reset (SOFTPRES)	Fault avoidance	Hardware - Software	On demand (Software defined)	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	RST10	Software Test of Reset - Type 1	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
System Control Module and Configuration Registers	SYS1	Multi-Bit Enable Keys for Control Registers	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS2	Lock Mechanism for Control Registers	Fault avoidance	Hardware	Continuous - Enabled by software	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS3	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SYS4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SYS5	Online Monitoring of Temperature	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	SYS6	Peripheral Clock Gating (PCLKCR)	Fault avoidance	Hardware	On demand (Software defined)	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS7	Peripheral Soft Reset (SOFTPRES)	Fault avoidance	Hardware	On demand (Software defined)	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS8	EALLOW and MEALLOW Protection for Critical Registers	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SYS9	Software Test of ERRORSTS Functionality	Diagnostic	Software	On demand (software defined)	Software defined	System defined	System defined
Debug Logic	JTAG1	Hardware Disable of JTAG Port	Fault avoidance	System	Continuous - Enabled at reset	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	JTAG2	Lockout of JTAG Access Using OTP	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	JTAG3	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	JTAG4	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
C28x Central Processing Unit	CPU4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CPU5	Access Protection Mechanism for Memories	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU7	CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CPU8	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	CPU9	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	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)
Flash	NWFLASH1	Flash ECC	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion or interrupt to CPU based on error severity	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	NWFLASH2	Flash Program Verify and Erase Verify Check	Diagnostic	Hardware	Continuous - Enabled at reset	1-2000 µS	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	NWFLASH3	Flash Program/Erase Protection	Fault avoidance	Hardware	Continuous - Enabled by software	Zero or very low overhead	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	NWFLASH4	Flash Wrapper Error and Status Reporting	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	NWFLASH5	CRC Check of Static Memory Contents	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	NWFLASH6	Prevent 0 to 1 Transition Using Program Command	Fault avoidance	Hardware	Continuous - Enabled by software	Zero or very low overhead	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	NWFLASH7	On-demand Software Program Verify and Blank Check	Diagnostic	Hardware - Software	On demand (Software defined)	1-2 µS	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	NWFLASH8	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	NWFLASH9	CMDWEPROT* and Program Command Data Buffer Registers Self-Clear After Command Execution	Fault avoidance	Hardware	Continuous - Enabled at reset	Zero or very low overhead	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	NWFLASH10	ECC Generation and Checker Logic is Separate in Hardware	Fault avoidance	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	NWFLASH12	Bit Multiplexing in Flash Memory Array	Fault avoidance	Hardware	Continuous - Enabled at reset	Zero or very low overhead	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	NWFLASH13	Auto ECC Generation Override	Test for diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	N/A	Software defined
	NWFLASH14	Software Test of Flash Prefetch, Data Cache and Wait-States	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	NWFLASH15	Software Test of ECC Logic	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	NWFLASH16	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	SRAM4	Bit Multiplexing in SRAM Memory Array	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SRAM5	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM6	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM7	Data Scrubbing to Detect/Correct Memory Errors	Fault avoidance	Software	On demand (Software defined)	Software defined	NMI with ERRORSTS assertion or interrupt to CPU based on error severity	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	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	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	SRAM13	Software Test of ECC Logic	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM14	Software Test of Parity Logic	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM16	Information Redundancy Techniques	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SRAM17	CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	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
	SRAM21	Memory Power-On Self-Test (MPOST)	Diagnostic	Hardware	Bootup (enabled by default)	Software defined	Software defined	Software defined
ROM	ROM2	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM3	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM4	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM5	CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	ROM6	Internal Watchdog (WD)	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Device reset or interrupt as per configuration	Software defined
	ROM7	External Watchdog	Diagnostic	System	System defined	System defined	System defined	System defined
	ROM8	Power-Up Pre-Operational Security Checks	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	ROM10	Memory Power-On Self-Test (MPOST)	Diagnostic	Hardware	Bootup (enabled by default)	Zero or very low overhead	Software defined	Software defined
	ROM15	ROM Parity	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	NMI with ERRORSTS assertion	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	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)
	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
Enhanced Peripheral Interrupt Expander (ePIE)	PIE2	Software Test of SRAM	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PIE3	Software Test of ePIE Operation Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PIE4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PIE5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PIE7	Maintaining Interrupt Handler for Unused Interrupts	Diagnostic	Software	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
Dual Zone Code Security Module (DCSM)	PIE11	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)
	PIE12	Software Test of Parity Logic	Test for diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DCSM1	Multi-Bit Enable Keys for Control Registers	Fault avoidance	Hardware	Continuous - Enabled at reset	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	DCSM2	Majority Voting and Error Detection of Link Pointer	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DCSM3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DCSM4	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DCSM5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	DCSM6	CPU Handling of Illegal Operation, Illegal Results and Instruction Trapping	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	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
	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	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	GPIO2	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	PWM4	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM5	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM6	Lock Mechanism for Control Registers	Fault avoidance	Hardware	Continuous - Enabled by software	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	PWM8	ePWM Fault Detection using XBAR	Diagnostic	Software	Continuous - Enabled by software	Zero or very low overhead	Software defined	Software defined
	PWM9	ePWM Synchronization Check	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM11	ePWM Application Level Safety Mechanism	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	PWM12	Online Monitoring of 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
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
	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

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

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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	QEP8	QMA Error Detection Logic	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically < 1µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	QEP9	eQEP Software Test of Quadrature Watchdog Functionality	Test for diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
XINT	XINT1	Software Test of Function Including Error Tests	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	XINT2	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	XINT3	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	XINT4	Hardware Redundancy	Diagnostic	Software	Continuous - Enabled by software	Zero or very low overhead	Software defined	Software defined
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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	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
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
	CMPSS8	CMPSS Ramp Generator Functionality Check	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
CMPSS LITE	CMPSSL1	Software Test of Function Including Error Tests	Test for diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CMPSSL3	Hardware Redundancy	Diagnostic	Software	Continuous - Enabled by software	Software defined	Software defined	Software defined
	CMPSSL4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CMPSSL5	Periodic Software Read Back of Static Configuration Registers	Test for 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

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	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	N/A (Fault Avoidance)	N/A (Fault avoidance technique)	N/A (Fault avoidance technique)
	CAN7	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAN8	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAN9	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAN10	DCAN Stuff Error Detection	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CAN11	DCAN Form Error Detection	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CAN12	DCAN Acknowledge Error Detection	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)

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

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Usage	Diagnostic Type	Diagnostic Operation	Test Execution Time	Action on Detected Fault	Error Reporting Time
	CAN13	Bit Error Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CAN14	CRC in Message	Diagnostic	Hardware	Continuous - Enabled at reset	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
	CAN15	Software Test of Parity Logic	Test for diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	CAN17	Software Test of Function Including Error Tests Using EPG	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
Serial Peripheral Interface (SPI)	SPI1	Software Test of Function Using I/O Loopback	Diagnostic	Hardware - Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SPI2	Information Redundancy Techniques Including End-to-End Safing	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SPI3	Periodic Software Read Back of Static Configuration Registers	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SPI4	Software Read Back of Written Configuration	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SPI5	Transmission Redundancy	Diagnostic	Software	On demand (Software defined)	Software defined	Software defined	Software defined
	SPI6	SPI Data Overrun Detection	Diagnostic	Hardware	Continuous - Enabled by software	Zero or very low overhead	Interrupt to CPU	Typically <1 µS to notify *(Interrupt Handling Time is System Load and Software Dependent)
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

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

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

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

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

8 References

1. Texas Instruments, *Calculating Useful Lifetimes of Embedded Processors*, application note.
2. JEDEC, *Moisture/Reflow Sensitivity Classification for Nonhermetic Solid State Surface Mount Devices*, standards.
3. JEDEC, *Handling, Packing, Shipping and Use of Moisture/Reflow Sensitive Surface Mount Devices*, standards.
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. IAV, *Standardized E-Gas Monitoring Concept for Gasoline and Diesel Engine Control Units*, application report.
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*, application note.
9. W. M. Goble and H. Cheddie, Safety Instrumented Systems Verification: Practical Probabilistic Calculations. Isa, 2004.
10. Texas Instruments: *TMS320C28x FPU Primer*, application note.
11. Texas Instruments: *Online Stack Overflow Detection on the TMS320C28x DSP*, application note.
12. International Electrotechnical Commission, *IEC-61784* , webpage.
13. Texas Instruments, *TMS320F280013x Real-Time Microcontrollers Technical Reference Manual*
14. Texas Instruments, *TMS320F280013x Real-Time Microcontrollers Data Manual*, data manual.
15. Texas Instruments, *UCD3138 Monitoring and Communications Programmer's Manual*, technical reference manual.
16. Texas Instruments, *Accelerators: Enhancing the Capabilities of the C2000™ MCU Family*, application note.
17. Texas Instruments, *Enhancing Device Security by Using JTAGLOCK Feature*, application note.

IMPORTANT NOTICE AND DISCLAIMER

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

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

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

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

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

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