

# **Safety Manual for TMS570LS04x, TMS570LS03x, and TMS570LS02x Hercules™ ARM®-Based Safety Critical Microcontrollers**

## **User's Guide**



Literature Number: SPNU552B  
November 2014—Revised October 2015

<b>1</b>	<b>Introduction</b>	<b>7</b>
<b>2</b>	<b>Hercules TMS570LS04x, TMS570LS03x, and TMS570LS02x Product Overview</b>	<b>9</b>
2.1	Targeted Applications	10
2.2	Product Safety Constraints	10
<b>3</b>	<b>Hercules Development Process for Management of Systematic Faults</b>	<b>11</b>
3.1	TI Standard MCU Automotive Development Process	12
3.2	TI MCU Automotive Legacy IEC 61508 Development Process	13
3.3	Yogitech fRMethodology Development Process	13
3.4	Hercules Enhanced Safety Development Process	13
<b>4</b>	<b>Hercules Product Architecture for Management of Random Faults</b>	<b>15</b>
4.1	Safe Island Philosophy and Architecture Partition to Support Safety Analysis (FMEA/FMEDA)	15
4.2	Identification of Parts/Elements	17
4.3	Management of Family Variants	18
4.4	Operating States	18
4.5	Management of Errors	19
<b>5</b>	<b>System Integrator Recommendations</b>	<b>20</b>
5.1	System Integrator Activities	20
5.2	Hints for Performing Dependent/Common Cause Failure Analysis Including the Hercules MCU	22
5.3	Hints for Improving Independence of Function/Co-Existence of Function When Using the Hercules MCU	22
5.4	Support for System Integrator Activities	23
<b>6</b>	<b>Brief Description of Elements</b>	<b>24</b>
6.1	Power Supply	24
6.2	Clocks	24
6.3	Reset	25
6.4	System Control Module	25
6.5	Error Signaling Module (ESM)	26
6.6	CPU Subsystem	26
6.7	Primary Embedded Flash	28
6.8	Flash EEPROM Emulation (FEE)	29
6.9	Primary Embedded SRAM	30
6.10	Level 2 and Level 3 (L2 and L3) Interconnect Subsystem	31
6.11	EFuse Static Configuration	32
6.12	OTP Static Configuration	32
6.13	I/O Multiplexing Module (IOMM)	33
6.14	Vectored Interrupt Module (VIM)	33
6.15	Real Time Interrupt (RTI)	34
6.16	High-End Timer (N2HET), HET Transfer Unit (HTU)	35
6.17	Multi-Buffered Analog-to-Digital Converter (MibADC)	36
6.18	Enhanced Quadrature Encoder Pulse (eQEP)	37
6.19	Multi Buffered Serial Peripheral Interface (MibSPI)	37
6.20	Serial Peripheral Interface (SPI)	38

6.21	Local Interconnect Network (LIN).....	39
6.22	Controller Area Network (DCAN) .....	39
6.23	General-Purpose Input/Output (GIO) .....	40
6.24	JTAG Debug and Test Access.....	41
6.25	Cortex-R4 Central Processing Unit (CPU) Debug.....	41
<b>7</b>	<b>Brief Description of Diagnostics.....</b>	<b>42</b>
7.1	1002 Software Voting Using Secondary Free Running Counter .....	42
7.2	Bit Error Detection .....	42
7.3	Bit Multiplexing in FEE Memory Array .....	42
7.4	Bit Multiplexing in Flash Memory Array .....	42
7.5	Bit Multiplexing in Primary SRAM Memory Array.....	42
7.6	Bit Multiplexing in Peripheral SRAM Memory Array.....	42
7.7	CPU Illegal Operation and Instruction Trapping .....	43
7.8	CPU Logic Built In Self Test (LBIST) .....	43
7.9	CPU Logic Built In Self Test (LBIST) Auto-Coverage.....	43
7.10	CPU Lockstep Compare .....	44
7.11	CPU Lockstep Comparator Self Test .....	44
7.12	CPU Online Profiling Using the Performance Monitoring Unit.....	44
7.13	CPU Memory Protection Unit (MPU).....	44
7.14	CRC Auto-coverage .....	45
7.15	CRC in Message.....	45
7.16	DCAN Acknowledge Error Detection.....	45
7.17	DCAN Form Error Detection .....	45
7.18	DCAN Stuff Error Detection .....	45
7.19	DCAN Protocol CRC in Message.....	45
7.20	Dual Clock Comparator (DCC).....	45
7.21	Efuse Autoload Self Test .....	46
7.22	Efuse Autoload Self Test Auto-Coverage .....	46
7.23	EFuse ECC .....	46
7.24	EFuse ECC Logic Self Test .....	46
7.25	eQEP Quadrature Watchdog.....	46
7.26	eQEP Software Test of Quadrature Watchdog Functionality .....	46
7.27	Error Trapping - IOMM .....	46
7.28	Error Trapping (including Peripheral Slave Error Trapping) - L2/L3 Interconnect.....	46
7.29	External Monitoring of Warm Reset (nRST) .....	47
7.30	External Monitoring via ECLK.....	47
7.31	External Voltage Supervisor.....	47
7.32	External Watchdog .....	47
7.33	FEE Contents Check by Hardware CRC .....	47
7.34	FEE Data ECC .....	47
7.35	FEE Sector Protection .....	48
7.36	Flash ATCM Address and Control Bus Parity .....	48
7.37	Flash Contents Check by Hardware CRC.....	48
7.38	Flash ECC .....	48
7.39	Flash Hard Error Cache and Livelock.....	49
7.40	Flash Sector Protection .....	49
7.41	Flash Wrapper Address ECC .....	49
7.42	Flash Wrapper Diag Mode 1 Test .....	49
7.43	Flash Wrapper Diag Mode 2 Test .....	49

7.44	Flash Wrapper Diag Mode 3 Test .....	49
7.45	Flash Wrapper Diag Mode 4 Test .....	49
7.46	Flash Wrapper Diag Mode 5 Test .....	50
7.47	Flash Wrapper Diag Mode 7 Test .....	50
7.48	Glitch Filtering on nRST and nPORRST .....	50
7.49	Hardware CRC Check of OTP Contents .....	50
7.50	Hardware Disable of JTAG Port .....	50
7.51	Information Redundancy Techniques .....	50
7.52	Information Redundancy Techniques - L2/L3 Interconnect Specific .....	50
7.53	Information Redundancy Techniques - DCAN Specific .....	50
7.54	Information Redundancy Techniques - N2HET Specific .....	51
7.55	Internal Voltage Monitor (VMON) .....	51
7.56	Internal Watchdog .....	51
7.57	IOMM Master ID Filtering .....	52
7.58	LIN Checksum Error Detection .....	52
7.59	LIN No-Response Error Detection .....	52
7.60	LIN Physical Bus Error Detection .....	52
7.61	LIN / SCI Bit Error Detection .....	52
7.62	LIN / SCI Frame Error Detection .....	52
7.63	LIN / SCI Overrun Error Detection .....	52
7.64	Locking Mechanism for Control Registers .....	52
7.65	Lockout of JTAG Access Using AJSM .....	52
7.66	Low Power Oscillator Clock Detector (LPOCLKDET) .....	53
7.67	Memory Protection Unit (MPU) for Non-CPU Bus Masters .....	53
7.68	Memory Protection Unit (MPU) for Cortex R4 CPU .....	53
7.69	MibADC Converter Calibration .....	53
7.70	MibADC Information Redundancy Techniques .....	53
7.71	MibADC Input Self Test .....	53
7.72	MibSPI/SPI Data Length Error Detection .....	54
7.73	MibSPI/SPI Data Overrun Detection .....	54
7.74	MibSPI/SPI Slave Desync Detection .....	54
7.75	MibSPI/SPI Slave Timeout Detection .....	54
7.76	Non-Privileged Bus Master Access .....	54
7.77	OTP Autoload ECC .....	54
7.78	Parity in Message .....	54
7.79	Periodic Hardware CRC Check of OTP Contents .....	55
7.80	Periodic Software Test of VIM Functionality Including Error Tests .....	55
7.81	Peripheral Central Resource (PCR) Access Management .....	55
7.82	Periodic Hardware CRC Check of SRAM Contents .....	55
7.83	Periodic Software Read Back of Static Configuration Registers .....	55
7.84	Peripheral SRAM Parity .....	55
7.85	PLL Slip Detector .....	55
7.86	Primary SRAM BTCM Address and Control Bus Parity .....	56
7.87	Primary SRAM Data and ECC Storage in Multiple Physical Banks per Logical Address .....	56
7.88	Primary SRAM Correctable ECC Profiling .....	56
7.89	Primary SRAM Data ECC .....	56
7.90	Primary SRAM Wrapper Redundant Address Decode .....	57
7.91	Primary SRAM Hard Error Cache and Livelock .....	57

7.92	Privileged Mode Access and Multi-Bit Enable Keys for Control Registers .....	57
7.93	PBIST Check of Primary or Module SRAM .....	58
7.94	PBIST Auto-coverage .....	58
7.95	PBIST Test of Parity Bit Memory .....	58
7.96	Redundant Address Decode Self Test.....	58
7.97	Shadow Registers .....	58
7.98	Software Check of Cause of Last Reset .....	59
7.99	Software Read Back of CPU Registers.....	59
7.100	Software Read Back of Written Configuration .....	59
7.101	Software Test of DCC Functionality .....	59
7.102	Software Test of DWD Functionality .....	59
7.103	Software Test of DWWD Functionality .....	59
7.104	Software Test of ECC Profiler .....	59
7.105	Software Test of Error Path Reporting .....	59
7.106	Software Test of Flash Sector Protection Logic.....	59
7.107	Software Test of Function Including Error Tests .....	59
7.108	Software Test of Function Using I/O Loopback .....	60
7.109	Software Test of Function Using I/O Checking In GIO Mode .....	60
7.110	Software Test of Function Using I/O Loopback in Transceiver / PHY .....	60
7.111	Software Test of Function Using I/O Loopback - IOMM Only .....	60
7.112	Software Test of Hardware CRC .....	60
7.113	Software Test of MPU Functionality .....	60
7.114	Software Test of Parity Logic .....	60
7.115	Software Test of PBIST .....	61
7.116	Software Test of SRAM Wrapper Address Decode Diagnostic and ECC .....	61
7.117	Software Test for Reset .....	61
7.118	Software Warm Reset Generation .....	61
7.119	Transmission Redundancy.....	61
7.120	Use of CoreSight Debug Logic Key Enable Scheme.....	61
7.121	Use of DCC as Program Sequence Watchdog.....	61
7.122	Use of MPUs to Block Access to Memory Mapped Debug .....	61
<b>8</b>	<b>Next Steps in Your Safety Development .....</b>	<b>61</b>
	<b>Appendix A Summary of Recommended Safety Feature Usage.....</b>	<b>63</b>
	<b>Appendix B Development Interface Agreement .....</b>	<b>96</b>
B.1	Appointment of Safety Managers .....	96
B.2	Tailoring of the Safety Lifecycle .....	96
B.3	Activities Performed by TI.....	98
B.4	Information to be Exchanged .....	98
B.5	Parties Responsible for Safety Activities .....	99
B.6	Communication of Target Values .....	99
B.7	Supporting Processes and Tools .....	100
B.8	Supplier Hazard and Risk Assessment .....	100
B.9	Creation of Functional Safety Concept .....	100
	<b>Appendix C Revision History .....</b>	<b>101</b>

## List of Figures

1	Device Revision Code Identification.....	7
2	Hercules Product Architecture Overview .....	9
3	TI Standard MCU Automotive QM Development Process .....	12
4	Hercules Enhanced Functional Safety Development Process .....	14
5	Partition of Hercules MCU for Safety Analysis .....	16
6	Operating States of the Hercules MCU .....	18
7	Diverse CPU Physical Orientation .....	27
8	Lockstep Temporal Diversity.....	28
9	Block Level Implementation of CPU SRAM .....	56
10	Hercules Tailoring of Safety Lifecycle .....	97

## List of Tables

1	Identification of Parts/Elements .....	17
2	Summary of ESM Error Indication .....	19
3	Key to Summary of Safety Features and Diagnostics.....	63
4	Summary of Safety Features and Diagnostics .....	64
5	Activities Performed by TI vs. Performed by SEooC Customer.....	98
6	Product Safety Documentation .....	99
7	Product Functional Documentation to be Considered in Safety-Related Design .....	99
8	Product Safety Documentation Tools and Formats .....	100
9	SPNU552B Revisions .....	101

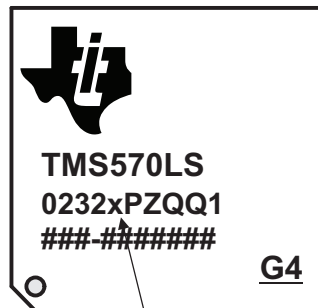
# Safety Manual for TMS570LS04x, TMS570LS03x, and TMS570LS02x Hercules™ ARM®-Based Safety Critical Microcontrollers

## 1 Introduction

You, a system and equipment manufacturer or designer, are responsible to ensure that your systems (and any TI hardware or software components incorporated in your systems) meet all applicable safety, regulatory, and system-level performance requirements. All application and safety related information in this document (including application descriptions, suggested safety measures, suggested TI products, and other materials) is provided for reference only. You understand and agree that your use of TI components in safety critical applications is entirely at your risk, and that you (as buyer) agree to defend, indemnify, and hold harmless TI from any and all damages, claims, suits, or expense resulting from such use.

This document is a safety manual for the Texas Instruments [Hercules](#) safety critical microcontroller product family. The product family utilizes a common safety architecture that is implemented in multiple application focused products. Product configurations supported by this safety manual include silicon revisions 0, A and B of the following products: (Note that the part numbers listed below are for revision B; other revisions are slightly different. The device revision can be determined by the symbols marked on the top of the device as seen in [Figure 1](#) below this list).

- TMS570LS Safety Critical Microcontrollers
  - TMS570LS0232 (Orderable Part #: TMS5700232BPZQQ1)
  - TMS570LS0232 (Orderable Part #: TMS5700232BPZQQ1R)
  - TMS570LS0332 (Orderable Part #: TMS5700332BPZQQ1)
  - TMS570LS0332 (Orderable Part #: TMS5700332BPZQQ1R)
  - TMS570LS0404 (Orderable Part #: TMS5700404BPZQQ1)
  - TMS570LS0404 (Orderable Part #: TMS5700404BPZQQ1R)
  - TMS570LS0432 (Orderable Part #: TMS5700432BPZQQ1)
  - TMS570LS0432 (Orderable Part #: TMS5700432BPZQQ1R)



Device Revision Code

**Figure 1. Device Revision Code Identification**

Hercules, SafeTI are trademarks of Texas Instruments.  
 ARM, Cortex are registered trademarks of ARM Limited.  
 Adobe is a trademark of Adobe Systems Incorporated in the United States, and/or other countries.  
 IBM, DOORS are registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.  
 Microsoft, Excel are registered trademarks of Microsoft Corporation in the United States and/or other countries, or both.  
 All other trademarks are the property of their respective owners.

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

- An overview of the superset product architecture
- An overview of the development process utilized to reduce systematic failures
- An overview of the safety architecture for management of random failures
- The details of architecture partitions, implemented safety mechanisms

The following information is documented in the *Safety Analysis Report Summary for TMS570LS04x, TMS570LS03x, and TMS570LS02x ARM®-Based Safety Critical Microcontrollers* (SPNU556) and is not repeated in this document:

- Summary of failure rates of the MCU estimated at the chip level
- Assumptions of use utilized in calculation of safety metrics
- Summary of targeted standard (IEC 61508, ISO 26262, and so forth) safety metrics at the chip level

The following information is documented in the *Detailed Safety Analysis Report for TMS570LS04x, TMS570LS03x, and TMS570LS02x ARM®-Based Safety Critical Microcontrollers* (SPNU559) and is not repeated in this document:

- Fault model used to estimate device failure rates suitable to enable calculation of customized failure rates
- Quantitative FMEA (also known as FMEDA, Failure Modes, Effects, and Diagnostics Analysis) with detail to the sub-module level of the device, suitable to enable calculation based on customized application of diagnostics

The following information is documented in the *Safety Report*, and will not be repeated in this document:

- Results of assessments of compliance to targeted standards

The user of this document should have a general familiarity with the Hercules product families. For more information, see <http://www.ti.com/hercules>. This document is intended to be used in conjunction with the pertinent data sheets, technical reference manuals, and other documentation for the products under development.

For information which is beyond the scope of the listed deliverables, please contact your TI sales representative or <http://www.ti.com>.



## 2 Hercules TMS570LS04x, TMS570LS03x, and TMS570LS02x Product Overview

The 65 nm Hercules product family is an evolution of the proven TMS570LS 130 nm safety MCU family into a 65 nm manufacturing process. A simplified graphical view of the product superset architecture can be seen in Figure 2. This is a basic representation of the architecture and is not all inclusive. For example, products in the family may scale based on the number of peripherals, number of bus master peripherals, or amount of memory - but the programmer's model remains consistent.

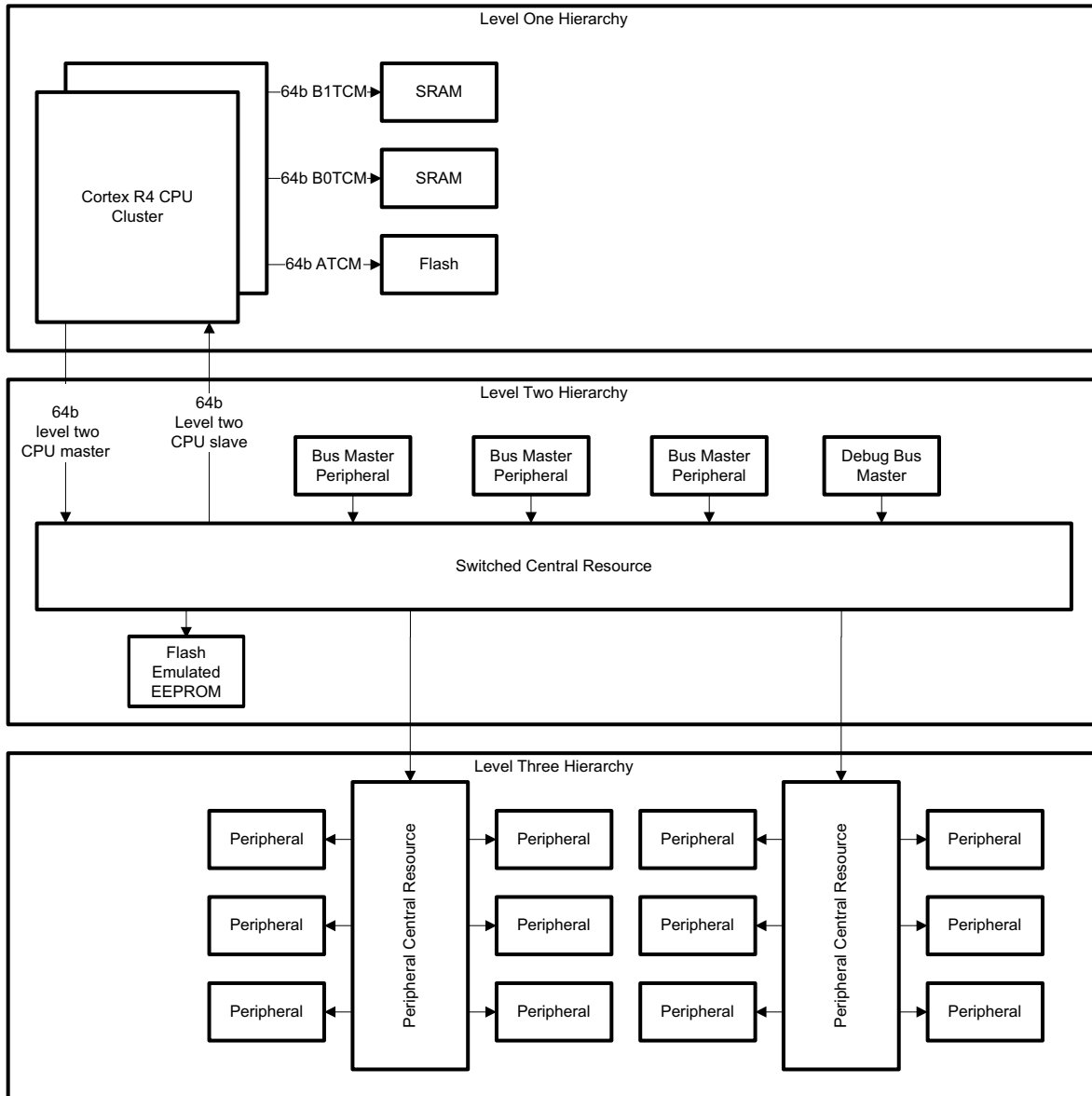


Figure 2. Hercules Product Architecture Overview

The Hercules product architecture utilizes the proven ARM Cortex®-R4 CPU in a tightly coupled memory configuration. The Cortex-R4 CPU is implemented with a checker Cortex-R4 CPU in a lockstep configuration. This provides cycle by cycle checking of correct CPU operation while keeping a simple, easy to use single core programmer's model. Access to primary CPU memory is achieved over three level one 64-bit tightly coupled memory (TCM) interfaces. The TCM interfaces allow up to three parallel accesses to SRAM and Flash in each clock cycle. The architecture is a modified Harvard type; program and data access are not limited to specific memory banks. A separate 64-bit level two bus master interface provides access to the level two memory hierarchy, while a 64-bit level two slave interface allows non-CPU bus masters access to the level one memories.

The level two device hierarchy is dominated by a switched central resource (also known as a bus matrix or crossbar). This is a device level interconnect that allows multiple bus masters to access multiple bus slaves, prioritization, routing, decode, and arbitration functions are provided. Bus masters to the level two device hierarchy include CPUs, bus master peripherals, and debug bus masters. Bus slaves on the level two hierarchy include the Flash EEPROM emulation memory, access to one or more peripheral bus segments, and a Cortex-R4 slave port Flash allows level two bus masters to access the level one tightly coupled memories.

The level three hierarchy is primarily composed of peripherals. Peripherals are grouped into one or more peripheral bus segments, managed by a peripheral central resource. The peripheral central resource provides address decode functionality for bus transactions targeting peripherals.

## 2.1 Targeted Applications

The Hercules MCU family is targeted at general purpose safety applications. Multiple safety applications were analyzed during the concept phase in order to support Safety Element out of Context (SEooC) development according to ISO 26262-10:2012. Example target applications include:

- Automotive braking systems, including anti-lock braking (ABS), anti-lock braking with traction control (ABS+ TC), and electronic stability control (ESC)
- Motor control systems, particularly electronic power steering (EPS) systems and electrical vehicle (EV) power train
- General purpose safety computation, such as integrated sensor cluster processing and vehicle strategy generation in an active safety system
- Industrial automation such as programmable logic controllers (PLCs) and programmable automation controllers (PACs) for safety critical process control

In the case of overlapping requirements between target systems, TI has attempted to design the device respecting the most stringent requirements. For example, the fault tolerant time intervals for timer logic in an ESC application are typically on the order of 100 ms. In an EPS application, the fault tolerant time interval is typically on the order of 10 ms. In such case, TI has performed timer subsystem analysis respecting <10 ms fault tolerant time interval.

While TI considered certain applications during the development of these devices, this should not restrict a customer who wishes to implement other systems. With all safety critical components, rationalization of the component safety concept to the system safety concept should be executed by the system integrator.

## 2.2 Product Safety Constraints

This device is a Type B device, as defined in IEC 61508:2010

This device claims no hardware fault tolerance (HFT = 0), as defined in IEC 61508:2010

For safety components developed according to many safety standards, it is expected that the component safety manual will provide a list of product safety constraints. For a simple component, or more complex components developed for a single application, this is a reasonable response. However, the Hercules 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. The *Detailed Safety Analysis Report for TMS570LS04x, TMS570LS03x, and TMS570LS02x ARM®-Based Safety Critical Microcontrollers* (SPNU559) provides a reference implementation of the Hercules product in a common system with relevant product safety constraints.

### 3 Hercules Development Process for Management of Systematic Faults

For a safety critical development, it is necessary to manage both systematic and random faults. Texas Instruments has created a unique development process for safety critical semiconductors that greatly reduces probability of systematic failure. This process builds on a standard Quality Managed (QM) development process as the foundation for safety critical development. This process is then augmented by a second layer of development activities that are specific to safety critical developments targeting IEC 61508 and ISO 26262.

In 2007, TI first saw the need to augment this standard development process in order to develop products according to IEC 61508. TI engaged with safety industry leader exida consulting to ensure the development was compliant to the standard. During 2008, a process for safety critical development according to IEC 61508 1st edition was implemented. This process has been executed on multiple microcontroller developments that are currently shipping into safety critical systems. The Hercules family product and safety architectures described in this document began development under the IEC 61508 development flow.

By mid 2009, it became clear that the emerging IEC 61508 2nd edition and ISO 26262 functional safety standards would require enhanced process flow capabilities. Due to the lack of maturity of these draft standards, it was not possible to implement a development process that ensured compliance before final drafts were available.

TI joined the ISO 26262 working group in mid 2009 as a way to better understand and influence the standard with respect to microcontroller hardware component development. As part of the US Technical Advisory Group (TAG) and international working group for ISO 26262, TI has notable contributions to:

- ISO 26262-5;Annex D - informative section describing failure modes and recommended diagnostics for hardware components, enhanced by TI's detailed knowledge of silicon failure modes and effectiveness of diagnostic methods
- ISO 26262-10;Clause 9 - informative section describing development of safety elements out of context, a technique that legitimizes and enables the use of Commercial Off The Shelf (COTS) safety critical components
- ISO 26262-10;Annex A - informative section describing how to apply ISO 26262 to microcontrollers, influenced by TI's lessons learned in application of IEC 61508 to microcontroller development

In mid 2010, TI started development of a process flow compliant to IEC 61508 2nd edition and ISO 26262 draft baseline 18. TI worked in detail with Yogitech in the ISO 26262 international working group and found that the companies have complementary capabilities. A partnership was established for engineering services and safety consulting services to accelerate new safety-related product development. Yogitech's existing fRMethodology development process and TI's IEC 61508 development process were merged and enhanced to create a new process addressing both ISO 26262 and IEC 61508 2nd edition. This process has gone through a process of continual improvement as ISO 26262 standards development continues. The process applied to the Hercules silicon covered by this document incorporates all changes through ISO 26262:2012 international standard release for part 10 and the ISO 26262:2011 international standard release for parts 1-9.

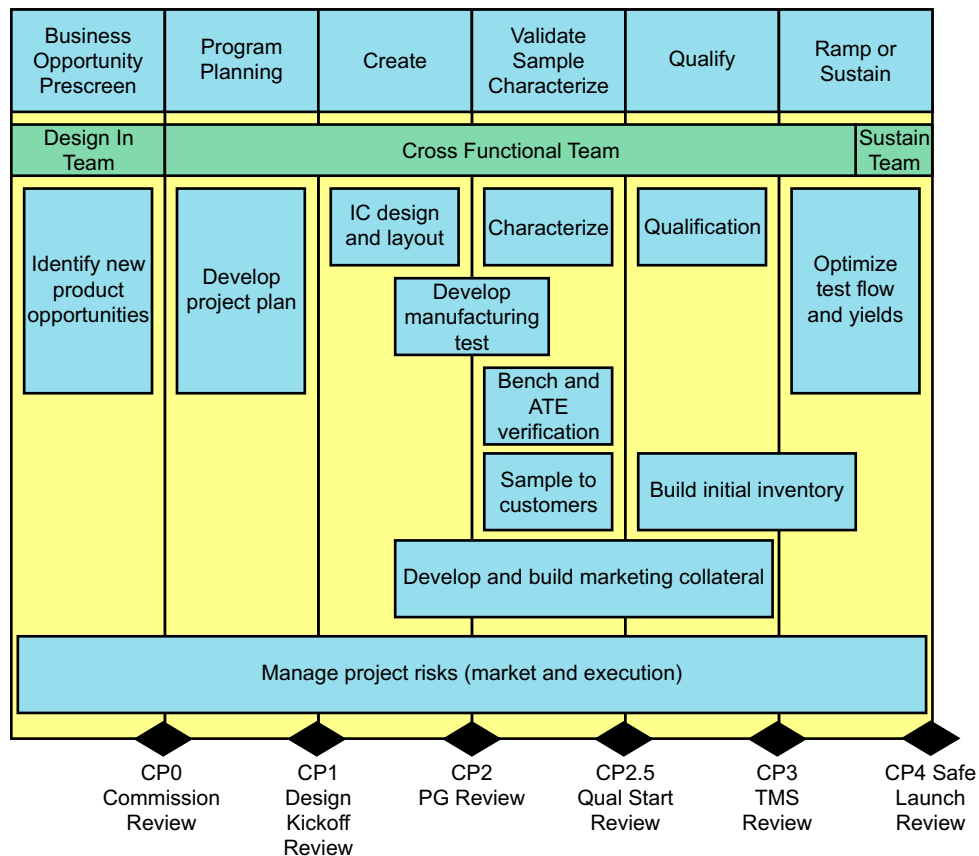
### 3.1 TI Standard MCU Automotive Development Process

Texas Instruments has been developing automotive microcontrollers for safety critical and non-safety critical automotive applications for over twenty years. Automotive markets have strong requirements on quality management and high reliability of product. Though not explicitly developed for compliance to a functional safety standard, the TI standard MCU Automotive development process already features many elements necessary to manage systematic faults. This development process can be considered to be Quality Managed (QM), but does not achieve an IEC 61058 Safety Integrity Level (SIL) or ISO 26262 Automotive Safety Integrity Level (ASIL). For up-to-date information on TI quality process certifications, see <http://www.ti.com/quality>.

The standard process breaks development into phases:

- Business opportunity pre-screen
- Program planning
- Create
- Validate, sample, and characterize
- Qualify
- Ramp to production and sustaining production

The standard process is illustrated in [Figure 3](#).



**Figure 3. TI Standard MCU Automotive QM Development Process**

### 3.2 TI MCU Automotive Legacy IEC 61508 Development Process

Texas Instruments developed an initial process for developing safety critical automotive microcontrollers in 2008. This process was developed targeting the IEC 61508 1st edition standard, as augmented with available committee drafts of the 2nd edition. The process is developed as an additional layer of activities that should be carried out in addition to the standard MCU Automotive QM development process. This process as applied on the TMS570LS20216S product development has been assessed suitable for use in IEC 61508 SIL 3 applications by exida Certification S.A. (certificate TI 071227 C001). In July 2012 the development process and the TMS570LS20x/10x product family was assessed to the IEC 61508:2010 standard and certified suitable for use in IEC 61508 SIL 3 applications by exida Certification Services (certificate TI 1204073 C001).

Key new activities in this process included:

- Nomination of a safety manager with ownership of all safety related activities
- Development of a safety plan to track safety related activities
- Generation, application, and validation of safety requirements
- Execution of qualitative (FMEA) and quantitative (FMEDA) safety analysis
- Authoring of safety manual and safety analysis report to support customer development

### 3.3 Yogitech fRMethodology Development Process

fRMethodology is the “white-box” approach for safety design exploration proprietary of YOGITECH, including:

- fRFMEA, a methodology to perform the FMEA of an integrated circuit in accordance to IEC 61508 and ISO 26262
- fRFI, a tool to perform fault injection of an integrated circuit based on inputs derived from fRFMEA

YOGITECH’s fRMethodology is in line with ISO 26262-10; Annex A. It mainly consists of:

- Splitting the component or system in elementary parts (“sensitive zones”)
- Computing their failure rates
- Using those failure rates to compute safety metrics
- Validating the results with fault injection
- Allowing sensitivity analyses of those metrics by changing architectural or technological parameters
- Delivering to the customer numbers to compare different architectures

### 3.4 Hercules Enhanced Safety Development Process

The Hercules enhanced safety development process is a merger of the existing TI and Yogitech flows for functional safety development. The goal of the process development is to take the best aspects of each flow and collaborate, resulting in the best in class capabilities to reduce systematic faults.

The process flow is targeted for compliance to IEC 61508 and ISO 26262, and is under a process of continuous improvement to incorporate new features of emerging functional safety standards. These functional safety standards are targeted because TI and Yogitech believe they best represent the state of the art in functional safety development for semiconductors. While not directly targeted at other functional safety standards, it is expected that products developed to industry state-of-the-art can be readily utilized in other functional safety systems.

The resulting flow was subsequently assessed by TUEV SUED for compliance to IEC 61508 and ISO 26262 and further enhanced based on technical findings. The development flow is certified by TUEV SUED under certificate Q4B 13 03 84071 001.

Key elements of the combined process flow are:

- Assumptions on system level design, safety concept, and requirements based on TI's expertise in safety critical systems development
- Combined qualitative and quantitative or similar safety analysis techniques comprehending the sum of silicon failure modes and diagnostic techniques known to both TI and Yogitech
- Fault estimation based on multiple industry standards as well as TI manufacturing data
- Application of Yogitech's state-of-the-art fault injection techniques for validation of claimed diagnostic coverage
- Integration of lessons learned by both companies through multiple safety critical developments to IEC 61508 and participation in the ISO 26262 international working group

The Figure 4 is shown below in a simplified graphic.

Phase 0 Business Opportunity Prescreen	Phase 1 Program Planning	Phase 2 Create	Phase 2.5 Validate, Sample, and Characterize	Phase 3 Qualify	Phase 4 Ramp or Sustain
Determine if safety process execution is necessary	Define SIL/ASIL capability	Execute safety design	Validate safety design in silicon	Qualification of safety design	Implement plans to support operation and production
Execute development interface agreement (DIA) with lead customers and suppliers	Generate safety plan	Qualitative analysis of design (FMEA and FTA)	Release safety manual	Release safety case report	Update safety case report (if needed)
	Initiate safety case	Incorporate findings into safety design	Release safety analysis report	Update safety manual (if needed)	Periodic confirmation measure reviews
	Analyze system to generate system level safety assumptions and requirements	Develop safety product preview	Characterization of safety design	Update safety analysis report (if needed)	
	Develop component level safety requirements	Validation of safety design at RTL level	Confirmation measure review	Confirmation measure review	
	Validate component safety requirements meet system safety requirements	Quantitative analysis of design (FMEDA)			
	Implement safety requirements in design specification	Incorporate findings into safety design			
	Validate design specification meets component safety requirements	Validation of safety design at gate/layout level			
	Confirmation measure review	Confirmation measure review			

**Figure 4. Hercules Enhanced Functional Safety Development Process**

## **4 Hercules Product Architecture for Management of Random Faults**

For a safety critical development it is necessary to manage both systematic and random faults. The Hercules product architecture includes many safety mechanisms, which can detect and respond to random faults when used correctly. This section of the document describes the architectural safety concept for the MCU.

### **4.1 Safe Island Philosophy and Architecture Partition to Support Safety Analysis (FMEA/FMEDA)**

The TMS570 Hercules processors share a common safety architecture concept called a “safe island” philosophy. The basic concept involves a balance between application of hardware diagnostics and software diagnostics to manage functional safety, while balancing cost concerns. In the “safe island” approach, a core set of elements are allocated continuously operating hardware safety mechanisms. This core set of elements, including power and clock and reset, CPU, Flash memory, SRAM and associated interconnect to Flash and SRAM, is needed to assure any functionally correct execution of software. Once correct operation of these elements is confirmed, software can be executed on these elements in order to provide software-based diagnostics on other device elements, such as peripherals. This concept has been proven viable through multiple generations of safety-critical products in the automotive passenger vehicle space.

Figure 5 illustrates the safe island approach overlaid to a superset configuration of the Hercules product architecture.

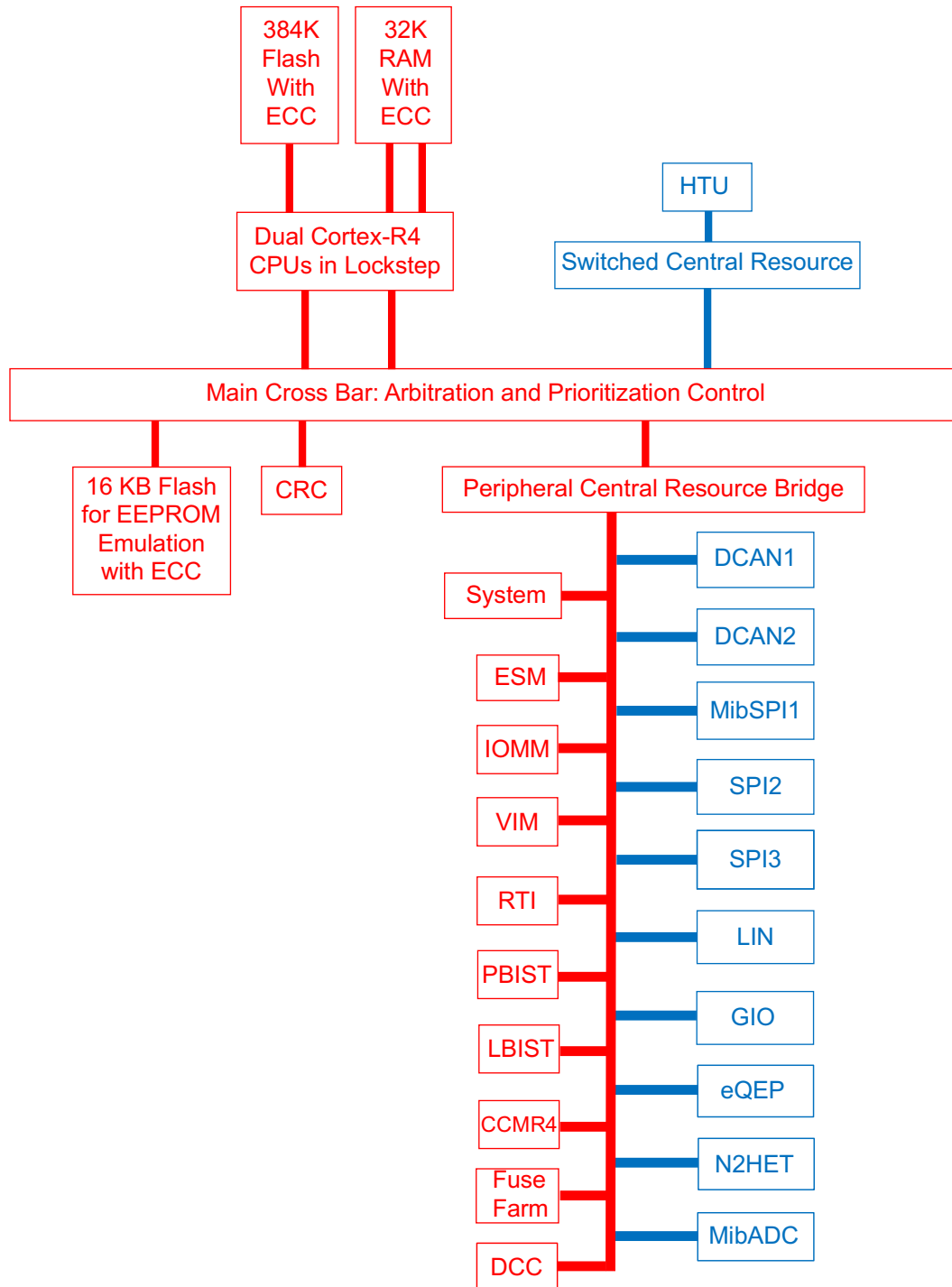


Figure 5. Partition of Hercules MCU for Safety Analysis



Figure 5 illustrates three architectural partitions:

- “Safe Island Layer” (RED) – This is the region of logic that is needed for all processing operations. This logic is protected heavily by on board hardware diagnostics and specific assumptions of use to assure a high level of confidence in safe operation. Once this region is safed, it can be used to provide comprehensive software diagnostics on other design elements.
- “Blended Layer” (BLUE) – This is the region of logic that includes most safety critical peripherals. This region has less reliance on hardware diagnostics. Software diagnostics and application protocols are overlaid to provide the remainder of needed diagnostic coverage.
- “Offline Layer” (BLACK) – This region of logic has minimal or no integrated hardware diagnostics. Many features in this layer are used only for debug, test, and calibration functions; Flash are not active during safety critical operation. Logic in this region could be utilized for safety critical operation assuming appropriate software diagnostics or system level measures are added by the system integrator.

## 4.2 Identification of Parts/Elements

For the purposes of a safety analysis, each module on this device can be considered to be a part or element. Each part or element has been assigned a three letter unique identifier, which is used uniformly in the Safety Manual, Safety Analysis Report, and FMEDA Documents to identify the element and its diagnostics. Table 1 lists each element present on this device and the unique identifier for this element. The overall IEC 61508 systematic capability of the MCU is SC3. TI does not make claims of systematic capability for specific IP modules on the device.

**Table 1. Identification of Parts/Elements**

Part / Element Name	Unique Identifier
Clock	CLK
Cortex-R4 Central Processing Unit (CPU)	CPU
Controller Area Network (DCAN)	CAN
Cortex-R4 Central Processing Unit (CPU) Debug	DBG
EFuse Static Configuration	EFU
Enhanced Quadrature Encoder Pulse (eQEP)	QEP
Error Signaling Module (ESM)	ESM
Flash EEPROM Emulation (FEE)	FEE
Primary Flash and Level 1 (L1) Interconnect	FLA
General Purpose Input/Output (GIO)	GIO
Input/Output (I/O) Multiplexing Module (IOMM)	IOM
Joint Technical Action Group (JTAG) Debug	JTG
Level 2 and Level 3 (L2 and L3) Interconnect	INC
Local Interconnect Network (LIN)	LIN
Multi-Buffered Analog to Digital Converter (MibADC)	ADC
Multi-Buffered Serial Peripheral Interface (MibSPI)	MSP
High-End Timer (N2HET) Including HET Transfer Unit (HTU)	HET
One Time Programmable (OTP) Flash Static Configuration	OTP
Power Supply	PWR
Reset	RST
Real Time Interrupt (RTI) Operating System Timer	RTI
Serial Peripheral Interface (SPI)	SPI
SRAM and Level 1 (L1) Interconnect	RAM
System Control Module	SYS
Vectored Interrupt Module (VIM)	VIM

NOTE: The terms "element" and "part" may have specific meaning and imply specific requirements dependent on the targeted functional safety standard. The terms are used here in a general sense.

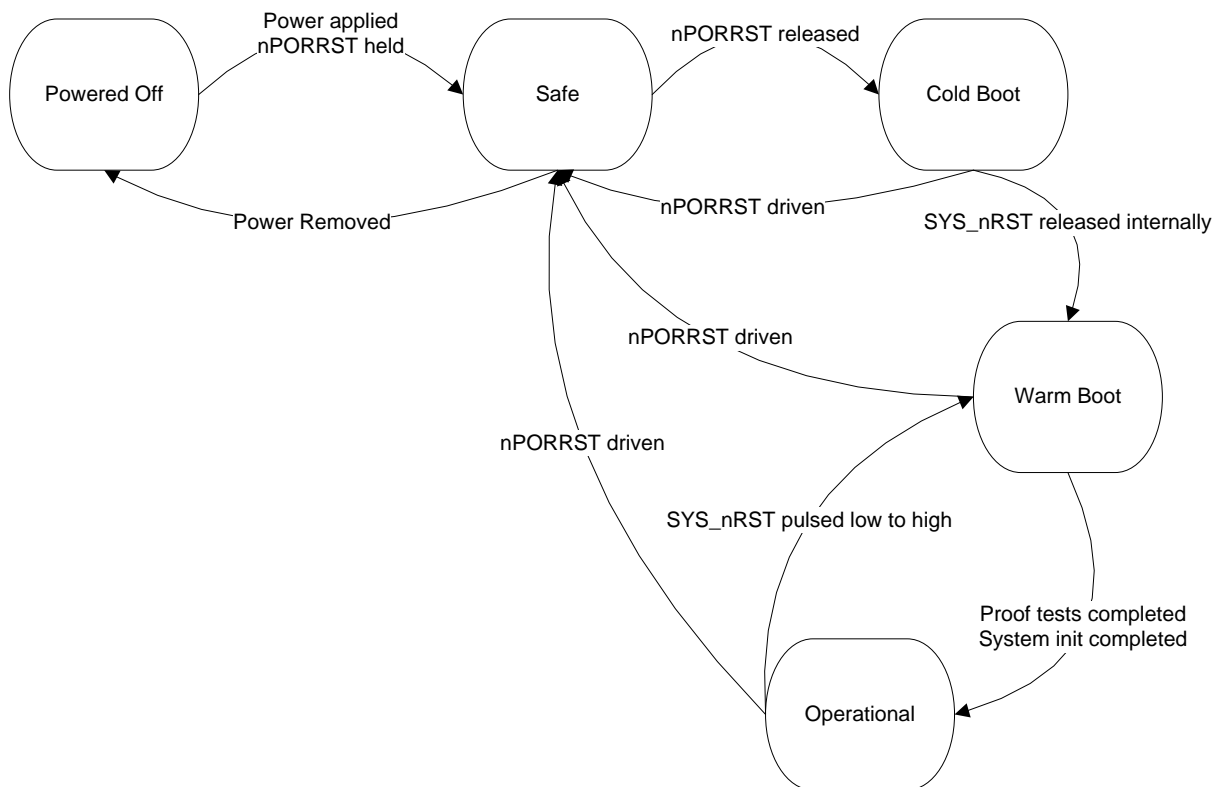
The [Hercules Architecture Brief Description of Elements](#) section contains a brief description of the elements listed above. For a full functional description of any of these modules, see the device-specific technical reference manual.

### 4.3 Management of Family Variants

The Hercules family architecture supports multiple product variants. These products could be implemented as unique silicon designs or they can be shared silicon designs that have elements disabled or not assured by specification, even if present in silicon. Only the elements of the superset architecture that are specifically detailed in the device-specific data sheet and technical reference manual are assured to be present and operate. When developing for the Hercules platform, it is recommended that the safety concept be based on the superset product architecture to enable maximum scalability across family variants. The superset architecture shown in the previous section is valid for all device part numbers noted in the introduction of the safety manual.

### 4.4 Operating States

The Hercules MCU products have a common architectural definition of operating states. These operating states should be observed by the system developer in their software and system level design concepts. The operating states state machine is shown in [Figure 6](#) and described below.



**Figure 6. Operating States of the Hercules MCU**

- "Powered Off" - This is the initial operating state of the Hercules MCU. No power is applied to either core or I/O power supply and the device is non-functional. This state can only transition to the safe state, and can only be reached from the safe state.
- "Safe" - In the safe state, the Hercules MCU is powered but non-operational. The nPORRST (power-on reset, also known as cold reset) is asserted by the system but is not released until power supplies have ramped to a stable state. The internal voltage monitor (VMON) safety mechanism also continues to assert the nPORRST internal to the device if power supplies are not within a minimum operational range. When the product is in the safe state, the CPU and peripherals are non-functional. Output drivers are tri-stated and input/output pins are kept in an input only state.
- "Cold Boot" - In the cold boot state, key analog elements, digital control logic, and debug logic are initialized for future use. The CPU remains powered but non-operational. When the cold boot process is completed, the SYS\_nRST signal is internally released, leading to the warm boot stage. The SYS\_nRST signal transition change can be monitored externally on the SYS\_nRST I/O pin.
- "Warm Boot" - The warm boot mode resets digital logic and enables the CPU. The CPU begins executing software from Flash memory and software initialization of the device can begin. There is no hardware interlock to say that warm boot is completed; this is a software decision.
- "Operational" - During the operational mode, the device is capable of supporting safety critical functionality.

## 4.5 Management of Errors

When a diagnostic detects a fault, the error must be indicated. The Hercules product architecture provides aggregation of fault indication from internal safety mechanisms using a peripheral logic known as the error signaling module (ESM). The ESM provides mechanisms to classify errors by severity and to provide programmable error response. The ESM does not, in and of itself, impact the overall function of the device and serves the limited purpose of fault aggregation and classification. The error classifications in the ESM are summarized in [Table 2](#).

**Table 2. Summary of ESM Error Indication**

Error Group	Interrupt Response	Error Pin Response	Notes
1	Programmable interrupt and programmable interrupt priority	Programmable response	For errors that are generally not of critical severity
2	Non maskable interrupt generated	Error pin activated	For errors that are generally of critical severity
3	No interrupt response	Error pin activated	For critical errors that are seen by diagnostic implemented in CPU

The error response is action that is taken by the MCU or system when an error is indicated. There are multiple potential of error response possible for the Hercules product. The system integrator is responsible to determine what error response should be taken and to ensure that this is consistent with the system safety concept.

- CPU abort - This response is implemented directly in the CPU, for diagnostics implemented in the CPU. During an abort, the program sequence transfers context to an abort handler and software has an opportunity to manage the fault.
- CPU interrupt - This response can be implemented for diagnostics outside the CPU. An 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.
- Generation of SYS\_nRST - This response allows the device to change states to warm boot from operational state. The SYS\_nRST could be generated from an external monitor or internally by the software reset or watchdog. Re-entry to the warm reset state allows possibility for software recovery when recovery in the operational state was not possible.
- Generation of nPORRST - This response allows the device to change state to safe state from cold boot, warm boot, or operational states. From this state, it is possible to re-enter cold boot to attempt recovery when recovery via warm boot is not possible. It is also possible to move to the powered-down state, if desired, to implement a system level safe state. This response can be generated from the internal voltage monitor, but is primarily driven by monitors external to the MCU.

The ESM provides multiple registers that can be read by the CPU to determine the current status of diagnostics and the state of the nERROR pin. For the severe group 2 errors, a shadow register is provided that is not reset by SYS\_nRST. This allows the possibility of warm reset reinitialization to identify that a group 2 error initiated the external reset.

It is possible for the CPU to trigger the nERROR pin response manually to test system behavior or to notify external logic of an internal fault not automatically indicated to ESM. The CPU is responsible to clear indicated errors in the ESM, including clearing of the nERROR pin response.

System level management of the external error response can be simplified through the use of a TI TPS6538x power supply and safety companion device developed for use with the Hercules family.

## 5 System Integrator Recommendations

You, as a system and equipment manufacturer or designer, are responsible to ensure that your systems (and any TI hardware or software components incorporated in your systems) meet all applicable safety, regulatory, and system-level performance requirements. All application and safety related information in this document (including application descriptions, suggested safety measures, suggested TI products, and other materials) is provided for reference only. You understand and agree that your use of TI components in safety critical applications is entirely at your risk, and that you (as buyer) agree to defend, indemnify, and hold TI harmless from any and all damages, claims, suits, or expense resulting from such use.

A brief description of each element on this device and the general assumptions of use are provided in [Section 6](#).

A list of diagnostic mechanisms for this device and a brief description for each diagnostic are provided in [Section 7](#).

The effectiveness of the hardware safety mechanisms is noted in the *Detailed Safety Analysis Report for TMS570LS04x, TMS570LS03x, and TMS570LS02x ARM®-Based Safety Critical Microcontrollers* (SPNU559).

This information should be used to determine the strategy for utilizing safety mechanisms. The details of each safety mechanism can be found in the device-specific technical reference manual.

Depending on the safety standard and end equipment targeted, it may be necessary to manage not only single point faults, but also latent faults. Per ISO 26262, a latent fault occurs when a fault capable of violating a safety goal occurs simultaneously to a fault in the diagnostic mechanism that would normally provide diagnostic coverage for that fault. Latent fault testing does not need to occur during the fault tolerant time interval, but can be performed at boot time, at shut down, or periodically as determined by the system developer. Many of the safety mechanisms described in this section can be used as primary diagnostics, diagnostics for latent fault, or both. When considering system design for management of latent faults, take care to include failure of execution resources when considering latent faults with software diagnostics, such as failure of CPU and memories.

### 5.1 System Integrator Activities

The system integrator is responsible for carrying out a number of product development activities. These activities carried out may include but are not limited to the following:

- Operational and Environmental Constraints
  - Verify that the implementation of the TI component in the system design is compliant to requirements in TI documentation. This includes but is not limited to the requirements found in technical reference manuals, data sheets, errata documents, safety manuals, and safety analysis reports
  - Verify that the system operational lifetime (power-on hours) does not exceed lifetime specifications for the TI component, as specified in the device data sheet. If the operational lifetime (power-on hours) is not specified in the data sheet, the use case does not match published conditions, or there are questions regarding device lifetime, please contact a TI quality/reliability engineering representative or <http://www.ti.com>.
  - Define system maintenance requirements. The Hercules MCU does not require maintenance.
  - Define system repair requirements. The Hercules MCU is non-repairable with respect to permanent faults. A power-on reset of the Hercules MCU may be considered a repair activity for transient

- faults per some definitions of system repair requirements.
- Define system decommissioning requirements. The Hercules MCU has no specific decommissioning requirements.
  - Define system disposal requirements. The Hercules MCU has no specific disposal requirements.
  - Avoidance of Systematic Errors
    - Verify the application of appropriate best practices at all stages of hardware and system development (including development of hardware and system diagnostics external to the MCU) to avoid systematic failure and to control random failures. This may include but is not limited to compliance to the requirements documented in IEC61508-2;Annex A Tables A.15 through A.18, as well as Annex B Tables B.1 through B.6
    - Verify that any software implemented (including software diagnostics) is developed with an appropriate set of measures to avoid systematic errors
  - Safety Concept Definition
    - Define the supported safety functions and verify that the microcontroller behaves properly to support execution of the defined safety function. This microcontroller is a generic product, which is capable of supporting a variety of safety functions but does not have fixed support for any specific safety function.
    - Define the system-level safe state concept considering safe-state entry, maintenance of safe state, and safe-state exit as appropriate to the application and verify correct implementation
    - Define the system-level error-handling concept and verify correct implementation.
    - Define appropriate overall timing requirements for safety metrics to be calculated for the application
    - Define appropriate safety metric targets for the application
  - Safety Concept Implementation
    - Select and implement an appropriate set of diagnostics and safety mechanisms from the MCU safety manual as necessary to satisfy the requirements of the targeted standards and the high level safety concept. Dependent on the results of the system level safety analysis, it may not be necessary to implement all diagnostic measures which TI has identified.
    - For the device diagnostics listed as "system" in [Table 4](#), implement the diagnostic in a manner that meets functional safety requirements of the system, particularly monitoring of the external clock, monitoring of voltage, and MCU state monitoring via external watchdog logic. TI's recommendations are based on analysis of what faults might be detected external to the MCU when considering fault models/failure modes described in IEC 61508 -2 Annex A as to be considered for any claims of high diagnostic coverage, including both permanent and transient failure modes.
    - Implement appropriate mechanisms to detect shorts between pins on the device. Tests may include I/O loopback tests, information redundancy, or system-level mechanisms designed to detect shorts.
    - Any end-to-end communications diagnostics implemented should consider the failure modes and potential mitigating safety measures described in IEC 61784-3:2010 and summarized in IEC 61784-3:2010 in [Table 1](#).
    - Ensure that any additional system level hardware or software diagnostics created or implemented by the system integrator are developed with an appropriate process to avoid systematic errors.
    - Define an appropriate diagnostic test interval per diagnostic to be implemented.
  - Verification of Safety Concept including Safety Metric Calculation
    - Verify the behavior of the MCU outputs in the system when the MCU is in a faulted condition.
    - Evaluate the system design for specific failure modes of functional logic and diagnostic logic which are detectable based on the specific application usage and the specific diagnostics applied. TI's safety analysis for the MCU considers all fault models noted in IEC 61508-2 Annex A as to be considered for any claims of high diagnostic coverage, including both permanent and transient failure modes. Refer to the Safety Analysis Report for more details.
    - Evaluate the system design for specific failure modes of functional logic and diagnostic logic which are not detectable based on the specific application usage and the specific diagnostics applied. TI's safety analysis for the MCU considers all fault models noted in IEC 61508-2 Annex A as to be considered for any claims of high diagnostic coverage, including both permanent and transient failure modes. Refer to the Safety Analysis Report for more details and ensure that the system



design considers system level diagnostics recommended by TI, such as external voltage supervision, external watchdog, and so forth.

- Verify that the implemented diagnostics meet the target diagnostic test interval per diagnostic.
- Estimate failure rates and diagnostic coverage per failure mode with respect to specific application usage. TI provides tools to support this activity in the Safety Analysis Report (SAR).
- Verify that environmental and operational constraints are properly modeled in the FMEDA to provide failure rate estimates.
- Verify that appropriate on-chip design elements are selected in the FMEDA for the specific safety function under analysis.
- Verify that targeted safety metrics are calculated and achieved
- Verify the diagnostic coverage achieved by the implemented system and software based diagnostics.
- Verify that the safety analysis considers MCU elements which are necessary to support the primary function, such as clock, power, OTP configuration, and similar. Many times the focus of analysis is the functional datapath but the elements necessary to support proper operation should also be considered.
- Execute a co-existence/freedom from interference analysis per the targeted standard to confirm that implemented functionality can co-exist without interference.
- Execute a dependent failure/common cause analysis to consider possible dependent/common cause failures on the sub-elements of the MCU, including pin level connections.

## 5.2 Hints for Performing Dependent/Common Cause Failure Analysis Including the Hercules MCU

The following steps may be useful for performing dependent/common cause failure analysis when using the Hercules MCU:

- Consider a relevant list of dependent fault/common cause fault initiators, such as the lists found in the draft ISO/PAS 19451 document, “Application of ISO 26262 to Semiconductors”
- Verify that the dependent failure analysis considers the impact of the software tasks running on the MCU, including hardware and software interactions and task/operating system interactions.
- Verify that the dependent failure analysis considers the impact of pin/ball level interactions on the MCU package, including aspects related to the selected I/O multiplexing

## 5.3 Hints for Improving Independence of Function/Co-Existence of Function When Using the Hercules MCU

The following steps may be useful for improving independence of function when using the Hercules MCU:

- Verify that unused interrupt channels are disabled in the VIM
- Verify that unused interrupt sources are disabled in the source peripherals
- Hold peripheral chip selects in reset with the PCR if the peripherals are unused
- Leave peripherals in default reset state if not controlled via PCR
- Power down power domains if all logic in power domain is not used
- Disable event triggers if unused
- Power down the ADC cores if MibADCs are unused
- Utilize peripheral bus master memory protection units (MPUs) to only allow access to needed transmit and receive buffers
- Utilize the CPU MPU to support isolation of separate software tasks running on the CPU
- Utilize privileged access modes as a secondary level of task isolation
- Utilize bus master ID filtering when available to limit allocation of peripherals to bus masters
- When possible, separate critical I/O functions by using non adjacent I/O pins/balls. Consider using the pin muxing logic to support such separation.
- Power down unused clock sources.

- Disable unused clock domains.
- Power down the flash pump logic if flash memory is not used after boot.

#### **5.4 Support for System Integrator Activities**

If you have any questions regarding usage of the TI documentation for system integration, or if you have questions regarding MCU level functional safety standard work products not provided as part of the TI documentation package, please contact TI support. The preferred and fastest method to contact TI support is via the E2E forum at <http://e2e.ti.com/support/microcontrollers/hercules/default.aspx>.

## 6 Brief Description of Elements

This section contains a brief description of the elements identified on this device in [Table 1](#). For a full functional description of any of these modules, see the device-specific technical reference manual.

### 6.1 Power Supply

The Hercules device family products require an external device to supply the necessary voltages and currents for proper operation. Separate voltage rails are available for core logic and I/O logic (including multi-buffered analog-to-digital converter (MibADC), Flash pump and oscillator).

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

- [Internal Voltage Monitor \(VMON\)](#)
- [External Voltage Supervisor](#)
- [MibADC Converter Calibration](#)
- [External Watchdog](#)

#### 6.1.1 Notes

- Management of voltage supervision at system level can be simplified by using a TI TPS6538x power supply and safety companion device developed for use with the Hercules family.
- Devices can be implemented with multiple power rails that are intended to be ganged together on the system PCB. For proper operation of power diagnostics, it is recommended to implement one voltage supervisor per ganged rail.
- Common mode failure analysis of the external voltage supervisor may be useful to determine dependencies in the voltage generation and supervision circuitry

### 6.2 Clocks

The Hercules 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):

- [Low Power Oscillator Clock Detector](#)
- [PLL Slip Detector](#)
- [Dual Clock Comparator](#)
- [External Monitoring via ECLK](#)
- [Internal Watchdog - DWD](#)
- [Internal Watchdog - DWWD](#)
- [External Watchdog](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Software Test of DCC Functionality](#)
- [Software Test of DWD Functionality](#)
- [Software Test of DWWD Functionality](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)



### 6.2.1 Notes

- Management of the external watchdog functionality at system level can be simplified by using a TI TPS6538x power supply and safety companion device developed for use with the Hercules family.
- User can improve the accuracy of the LPOCLKDET diagnostic via programming the trim values in the HF LPO. This would require the customer to determine the LPO trim value during their manufacturing test via comparison to a calibrated clock source.
- There are many possible implementations of watchdogs for use in providing clock and CPU diagnostics. In general, TI recommends the use of an external watchdog over an internal watchdog for reasons of reduced 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 ECLK pin may have EMI implications.

### 6.3 Reset

The Hercules device family products require an external reset at cold and power-on (nPORRST) to place all asynchronous and synchronous logic into a known state. The power-on reset generates an internal warm reset (nRST) signal to reset the majority of digital logic as part of the boot process. The nRST signal is provided at device level as an I/O pin; it will toggle when asserted internally and can be driven externally to generate 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](#)
- [Software Check of Cause of Last Reset](#)
- [Software Warm Reset Generation](#)
- [Glitch Filtering on Reset Pins](#)
- [Use of Status Shadow Registers](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)
- [Software Test for Reset](#)
- [External Watchdog](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (In combination with other diagnostics on the primary function)

- [Internal Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

### 6.3.1 Notes

- Management of reset at system level can be simplified by using a TI TPS6538x power supply and safety companion device developed for use with the Hercules family.
- Internal watchdogs are not a viable option for reset diagnostics as the monitored reset signals interact with the internal watchdogs.

### 6.4 System Control Module

The system control module contains the memory-mapped registers to interface clock, reset, and other system related control and status logic. The system control module is also responsible for generating the synchronization of system resets and delivering the warm system reset nRST.

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

- [Privileged Mode Access and Multi-Bit Enable Keys for Control Registers](#)
- [Software Readback of Written Configuration](#)
- [Periodic Software Readback of Static Configuration Registers](#)

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

#### 6.4.1 Notes

- Depending on targeted metrics, a user can elect to implement a periodic software test of static configuration registers in the system control module. Such a test can provide additional diagnostic coverage for disruption by soft error.
- Review the clock and reset sections as these features are closely controlled by the system control module.

### 6.5 Error Signaling Module (ESM)

The ESM provides unified aggregation and prioritization of on-board hardware diagnostic errors. For more information, see [Management of Errors](#).

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

- [Periodic Software Readback of Static Configuration Registers](#)
- [Boot Time Software Test of Error Path Reporting](#)
- [Periodic Software Test of Error Path Reporting](#)
- [Use of Status Shadow Registers](#)
- [Software Readback of Written Configuration](#)

The following tests can be applied as test-for-diagnostics on this module to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

#### 6.5.1 Notes

- Software testing of the ESM error path can be combined with boot time latent tests of hardware diagnostics to reduce startup time.
- Testing of ESM error path may result in assertion of the nERROR diagnostic output. System integrator should ensure that the system can manage or recover gracefully from the nERROR event.

### 6.6 CPU Subsystem

The Hercules product family relies on the ARM® Cortex-R4 CPU to provide general-purpose processing. The Cortex-R4 is a high performance CPU with embedded safety diagnostics. The R4 is also designed for easy integration into a 1oo1D lockstep configuration. These aspects make the Cortex-R4 an outstanding CPU for functional safety products.

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

- [CPU Lockstep Compare](#)
- [Boot Time Execution of LBIST STC](#)
- [Periodic Execution of LBIST STC](#)
- [Memory Protection Unit \(MPU\)](#)
- [Online Profiling Using PMU](#)
- [Illegal Operation and Instruction Trapping](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Read Back of CPU Registers](#)
- [Hardware Disable of JTAG Port](#)
- [Internal Watchdog](#)
- [External Watchdog](#)

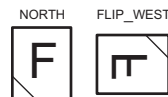
The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [CPU Lockstep Compare](#)
- [LBIST Auto-Coverage](#)

### 6.6.1 Measures to Mitigate Common Mode Failure in CPU Subsystem

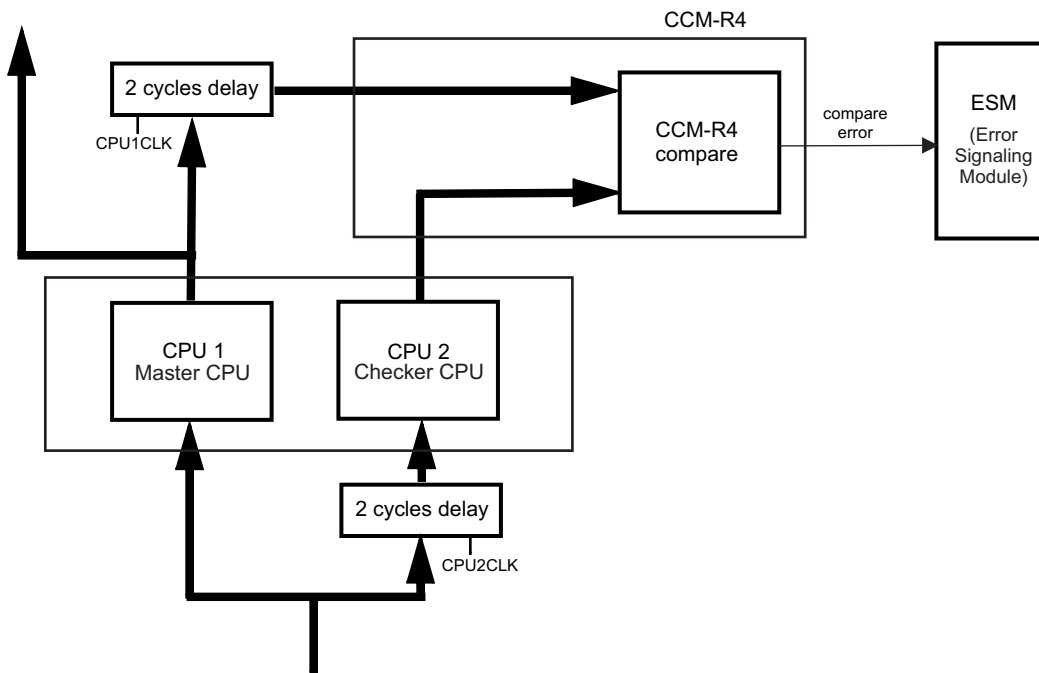
The Hercules lockstep CPU subsystem design includes multiple best practices to mitigate common mode failure:

- Physical diversity:
  - Physical core hard macros are spaced at least 100 µm apart.
  - One core is flipped and rotated in orientation relative to the other core. For example, if one core is considered "North", the second would be "Flip West" as shown in [Figure 7](#).



**Figure 7. Diverse CPU Physical Orientation**

- Temporal diversity:
  - Timing delay blocks are inserted to delay the operation of the CPUs by two cycles as shown in [Figure 8](#).



**Figure 8. Lockstep Temporal Diversity**

- The CPU clock domain is split into two clock trees such that clocks are delivered to the two CPUs by separate paths.
- Power diversity:
  - Each core has a dedicated power ring.

### 6.6.2 Notes

- Many safety critical microcontrollers utilize a software-based test of CPU functionality as opposed to a hardware scheme such as the TI LBIST STC. TI does not recommend such tests for a CPU of medium to high complexity, such as the Cortex-R4. Software-based options have higher memory cost, lower detection capability, and longer execution times than the equivalent LBIST STC solution.
- Due to the lockstep diagnostic, it is not necessary to execute a periodic test of CPU control register configuration. A control register disturb in one CPU should not impact the second CPU.

## 6.7 Primary Embedded Flash

The primary embedded Flash memory is a non-volatile memory that is tightly coupled to the ATCM port of the Cortex-R4 CPU core. The ATCM 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. A Flash wrapper logic provides multiple pipelined read buffers to improve CPU access time on sequential address fetches.

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

- [Flash Data ECC](#)
- [Hard Error Cache and Livelock](#)
- [Flash Wrapper Address ECC](#)
- [Address and Control Parity](#)
- [Boot Time Check of Flash Memory Contents by Hardware CRC](#)

- [Periodic Check of Flash Memory Contents by Hardware CRC](#)
- [Bit Multiplexing in Flash Memory Array](#)
- [Flash Sector Protection](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Flash Wrapper Diag Mode 5 Test](#)
- [Flash Wrapper Diag Mode 7](#)
- [Software Test of Parity Logic](#)
- [Software Test of Flash Sector Protection Logic](#)
- [Software Test of Hardware CRC](#)
- [CRC Auto-Coverage](#)
- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [SRAM Data ECC](#)

### 6.7.1 Notes

- When exporting ECC error events from CPU to Flash wrapper and from the Flash wrapper to ESM, care should be taken that the error is not due to discarded prefetch or other branch discontinuity.
- By executing a CRC read back of Flash contents while ECC is enabled on the Flash, it is possible to perform two diagnostics of the Flash in parallel.
- The Flash module may have unique power supply pins depending on the product configuration. It is highly recommended to implement voltage supervision on these pins as described in the power supply section.
- The Flash module may have unique test signal pins depending on the product configuration. For proper board level management of these signals, see the device-specific data sheet.

### 6.8 Flash EEPROM Emulation (FEE)

The Hercules platform architecture includes the capability for a separate bank of Flash memory to be used as Flash EEPROM Emulation (FEE). FEE is used for data storage only, and cannot be used as a CPU instruction memory. FEE is a level two memory that is accessed by the CPU over the AXI master port as opposed to the tightly coupled memory interface used for the main Flash banks. The emulation of EEPROM in the FEE memory is managed by specific FEE drivers running on the CPU.

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

- [FEE Data ECC](#)
- [Boot Time Check of FEE Memory Contents by Hardware CRC](#)
- [Periodic Check of FEE Memory Contents by Hardware CRC](#)
- [Bit Multiplexing in FEE Array](#)
- [FEE Sector Protection](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)
- [Flash Wrapper Address ECC](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Flash Wrapper Diag Mode 1 Test](#)
- [Flash Wrapper Diag Mode 2 Test](#)
- [Flash Wrapper Diag Mode 3 Test](#)
- [Flash Wrapper Diag Mode 4 Test](#)
- [Software Test of Flash Sector Protection Logic](#)
- [Software Test of Hardware CRC](#)
- [CRC Auto-Coverage](#)
- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

### 6.8.1 Notes

- As all application FEE accesses are run through a software driver to manage EEPROM emulation, any diagnostics must respect the functionality of the FEE driver.
- The FEE module may have unique power supply pins depending on the product configuration. It is highly recommended to implement voltage supervision on these pins as described in the power supply section.
- The FEE module may have unique test signal pins depending on the product configuration. For proper board level management of these signals, see the device-specific data sheet.

## 6.9 Primary Embedded SRAM

The primary embedded SRAM is a volatile memory that is tightly coupled to the BTCM ports of the Cortex-R4 CPU core. The BTCM SRAMs are primarily used for CPU data access, though instruction access is also possible. The SRAM has much faster access time than Flash memory, such that no wait states are necessary at maximum CPU frequency.

Two 64-bit BTCM interfaces are implemented: BTCM0 and BTCM1. It is possible for the CPU to generate two BTCM accesses in one cycle: one on BTCM0 and one on BTCM1. The SRAM addressing is interleaved across the two BTCM interfaces on a 64-bit basis, resulting in significantly reduced arbitration time between multiple BTCM masters (PFU, LSU, and AXI-S)

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

- [Data ECC](#)
- [Hard Error Cache and Livelock](#)
- [Correctable ECC Profiling](#)
- [Address and Control Bus Parity](#)
- [Redundant Address Decode](#)
- [Data and ECC Storage in Multiple Physical Banks](#)
- [Boot Time PBIST Check of Primary SRAM](#)
- [Periodic PBIST Check of Primary SRAM](#)
- [Bit Multiplexing in Primary SRAM Array](#)
- [Periodic Hardware CRC Check of Primary SRAM Contents](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Software Test of SRAM Wrapper Redundant Address Decode and ECC](#)
- [Software Test of Parity Logic](#)
- [Software Test of PBIST](#)
- [PBIST Auto-Coverage](#)
- [Software Test of ECC Profiler](#)
- [Redundant Address Decode Self Test](#)
- [Software Test of Hardware CRC](#)
- [CRC Auto-Coverage](#)
- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)

### 6.9.1 Notes

- There are two SRAM wrappers, one for each BTCM interface. Ensure that you configure both SRAM wrappers if configuration is needed to support a diagnostic.
- By executing a CRC read back of SRAM contents while ECC is enabled on the SRAM, it is possible to perform two diagnostics of the SRAM in parallel.
- The SRAM module may have unique power supply pins depending on the product configuration. It is highly recommended to implement voltage supervision on these pins as described in the power supply section.
- The redundant address decode does not provide hardware fault tolerance. As such, the logic may not be considered fully redundant per the definition of redundancy in some safety standards.

## 6.10 Level 2 and Level 3 (L2 and L3) Interconnect Subsystem

The system interconnect is composed of a number of bridges, gaskets, communications crossbars, and routing logic, which connects the bus masters (CPU, TUs, and so forth) to L2 slaves and L3 peripherals. This logic, while having no explicit safety critical purpose, is an intermediary in the transfer of safety critical communications.

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

- [Error Trapping \(Including Peripheral Slave Error Trapping\)](#)
- [PCR Access Management](#)
- [Information Redundancy](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Boot Time Software Test of Basic Functionality Including Error Tests](#)
- [Periodic Software Test of Basic Functionality Including Error Tests](#)
- [Software Readback of Written Configuration](#)
- [Transmission Redundancy](#)
- [Internal Watchdog](#)
- [External Watchdog](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)



- [SRAM Data ECC](#)

### 6.10.1 Notes

- An end to end communications safety mechanism implemented on a networked peripheral provides an indirect form of information redundancy diagnostic on the L2 and L3 interconnect.
- The only module in the L2 and L3 interconnect subsystem that has memory-mapped registers is the PCR.

## 6.11 *EFuse Static Configuration*

The Hercules platform devices support a manufacturing time configuration of certain functionality (such as trim values for analog macros) via one time programmable (OTP) EFuse structures. The EFuses are read automatically after power-on reset by an autoload function.

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

- [Boot Time Autoload Self Test](#)
- [E-Fuse ECC](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Autoload Self Test Auto-Coverage](#)
- [EFuse ECC Logic Self Test](#)
- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

## 6.12 *OTP Static Configuration*

The Hercules platform devices support a manufacturing time configuration of certain functionality (such as endianness and the initial configuration of power domains after reset) via one time programmable Flash memory. The OTP configuration values are read automatically after warm reset by an autoload function.

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

- [OTP ECC](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)
- [Boot Time Hardware CRC Check of OTP Contents](#)
- [Periodic Hardware CRC Check of OTP Contents](#)
- [Bit Multiplexing in Flash Memory Array](#)
- [Flash Sector Protection](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Flash Wrapper Diag Mode 1 Test](#)
- [Flash Wrapper Diag Mode 2 Test](#)
- [Flash Wrapper Diag Mode 3 Test](#)



- [Flash Wrapper Diag Mode 4 Test](#)
- [Software Test of Flash Sector Protection Logic](#)
- [Software Test of Hardware CRC](#)
- [CRC Auto-Coverage](#)
- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

#### 6.12.1 Notes

The OTP Flash memory used for storage of the configuration words can be read by the CPU.

### 6.13 I/O Multiplexing Module (IOMM)

The I/O multiplexing module (IOMM) provides software configurable mapping of internal module I/O functionality to device pins.

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

- [Locking Mechanism For Control Registers](#)
- [Master ID Filtering](#)
- [Error Trapping](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Boot Time Software Test of Function Using I/O Loopback](#)
- [Periodic Software Test of Function Using I/O Loopback](#)
- [Software Readback of Written Configuration](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

#### 6.13.1 Notes

Software testing of the IOMM can be combined with peripheral loopback testing.

### 6.14 Vectored Interrupt Module (VIM)

The vectored interrupt module (VIM) is used to interface peripheral interrupts to the Cortex-R4 CPU. The VIM provides programmable interrupt prioritization, masking, and sleep mode wake up functionality. The VIM includes a local SRAM that is used to hold the address of the interrupt handler per channel.

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

- [VIM SRAM Data Parity](#)
- [Boot Time PBIST Check of VIM SRAM](#)
- [Periodic PBIST Check of VIM SRAM](#)

- [Bit Multiplexing in VIM SRAM Array](#)
- [Periodic Hardware CRC Check of VIM SRAM Contents](#)
- [Boot Time Software Test of VIM Functionality Including Error Tests](#)
- [Periodic Software Test of VIM Functionality Including Error Tests](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)
- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Memory Protection Unit \(MPU\)](#)
- [Online Profiling Using PMU](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Software test of Parity Logic](#)
- [PBIST Test of Parity Bit Memory](#)
- [Software Test of Hardware CRC](#)
- [CRC Auto-Coverage](#)
- [Software Test of PBIST](#)
- [PBIST Auto-Coverage](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

#### 6.14.1 Notes

Care should be taken to optimize any VIM software diagnostics for the mode of operation - legacy IRQ and FIQ, legacy vectored, or fully hardware vectored.

### 6.15 Real Time Interrupt (RTI)

The real time interrupt (RTI) module provides the operating system 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 following tests can be applied as diagnostics for this module (to provide diagnostic coverage on a specific function):

- [1002 Software Voting Using Secondary Free Running Counter](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

#### 6.15.1 Notes

When using one counter to the operating system timebase counter, a diverse configuration of clock source, scaling factor, and so forth can be used to reduce probability of common mode failure.

## 6.16 High-End Timer (N2HET), HET Transfer Unit (HTU)

The N2HET module is a programmable timer with input/output capabilities. The N2HET is implemented as a simple RISC processor with instruction set dedicated for timing operations. Complex inputs can be captured and pre-processed by the N2HET for later processing by the CPU. Output generation is typically pulse width modulation (PWM), but may also be simple general-purpose input/output (GIO) type signals.

N2HET has a dedicated micro DMA controller called HTU. The HTU provides a high bandwidth connection for data transfer to and from N2HET from and to CPU memories.

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

- [Memory Protection Unit for HTU Bus Master Accesses](#)
- [Information Redundancy Techniques](#)
- [Use of DCC as Program Sequence Watchdog](#)
- [Boot Time Software Test of Function Using I/O Loopback](#)
- [Periodic Software Test of Function Using I/O Loopback](#)
- [N2HET/HTU SRAM Data Parity](#)
- [Boot Time PBIST Check of N2HET/HTU SRAM](#)
- [Periodic PBIST Check of N2HET/HTU SRAM](#)
- [Bit Multiplexing in N2HET/HTU SRAM Array](#)
- [Periodic Hardware CRC Check of N2HET/HTU SRAM Contents](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)
- [Transmission Redundancy for Transfer Unit](#)
- [Boot Time Software test of Function Using I/O Checking In GIO Mode](#)
- [Periodic Software test of Function Using I/O Checking In GIO Mode](#)
- [Information Redundancy Techniques \(while in GIO Mode\)](#)
- [Hardware Disable of JTAG Port](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Software test of Parity Logic](#)
- [PBIST Test of Parity Bit Memory](#)
- [Software Test of Hardware CRC](#)
- [CRC Auto-Coverage](#)
- [Software Test of PBIST](#)
- [PBIST Auto-Coverage](#)
- [Software Test of MPU Functionality](#)
- [Software Test of DCC Functionality](#)
- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

### 6.16.1 Notes

- Information redundancy techniques used on N2HET can be extended to also cover the HTU bus master operation.

- To reduce probability of common mode failure, user should consider implementing multiple channels (information redundancy) using non adjacent pins.

### 6.17 Multi-Buffered Analog-to-Digital Converter (MibADC)

The MibADC module is used to convert analog inputs into digital values. Results are stored in internal SRAM buffers for later transfer by CPU.

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

- [Hardware Disable of JTAG Port](#)
- [Boot Time Input Self Test](#)
- [Boot Time Converter Calibration](#)
- [Periodic Converter Calibration](#)
- [MibADC Information Redundancy Techniques](#)
- [MibADC SRAM Data Parity](#)
- [Boot Time PBIST Check of MibADC SRAM](#)
- [Periodic PBIST Check of MibADC SRAM](#)
- [Bit Multiplexing in MibADC SRAM Array](#)
- [Periodic Hardware CRC Check of MibADC SRAM Contents](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)
- [Boot Time Software Test of Function Using I/O Checking In GIO Mode](#)
- [Periodic Software Test of Function Using I/O Checking In GIO Mode](#)
- [Information Redundancy Techniques \(while in GIO Mode\)](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Software test of Parity Logic](#)
- [PBIST Test of Parity Bit Memory](#)
- [Software Test of Hardware CRC](#)
- [CRC Auto-Coverage](#)
- [Software Test of PBIST](#)
- [PBIST Auto-Coverage](#)
- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

#### 6.17.1 Notes

- The MibADC module may be referred to as MibADC (Multi Buffered ADC) in some documents.
- ADC module voltages should be supervised as noted in [Power Supply](#)
- To reduce probability of common mode failure, user should consider implementing multiple channels (information redundancy) using non adjacent pins.

## 6.18 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):

- [Boot Time Software Test of Function Including Error Tests](#)
- [Periodic Software Test of Function Including Error Tests](#)
- [Information Redundancy Techniques](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)
- [Quadrature Watchdog](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Software Test of Quadrature Watchdog Functionality](#)
- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

### 6.18.1 Notes

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

## 6.19 Multi Buffered Serial Peripheral Interface (MibSPI)

The MibSPI modules provide serial I/O compliant to the MibSPI protocol. MibSPI communications are typically used for communication to smart sensors and actuators, serial memories, and external logic such as a watchdog device. The MibSPI modules contain internal SRAM buffers. If not used for MibSPI communication, the MibSPI modules support the usage of their I/O as general purpose I/O.

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

- [Boot Time Software Test of Function Using I/O Loopback](#)
- [Periodic Software Test of Function Using I/O Loopback](#)
- [Parity in Message](#)
- [Information Redundancy Techniques](#)
- [MibSPI SRAM Data Parity](#)
- [Boot Time PBIST Check of MibSPI SRAM](#)
- [Periodic PBIST Check of MibSPI SRAM](#)
- [Bit Multiplexing in MibSPI SRAM Array](#)
- [Periodic Hardware CRC Check of MibSPI SRAM Contents](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)
- [Transmission Redundancy](#)
- [Data Overrun Error Detection](#)
- [Bit Error Detection](#)

- Slave Desync Detection
- Slave Timeout Detection
- Data Length Error Detection
- Hardware Disable of JTAG Port
- Boot Time Software Test of Function Using I/O Checking In GIO Mode
- Periodic Software Test of Function Using I/O Checking In GIO Mode
- Information Redundancy Techniques (while in GIO Mode)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- Software test of Parity Logic
- PBIST Test of Parity Bit Memory
- Software Test of Hardware CRC
- CRC Auto-Coverage
- Software Test of PBIST
- PBIST Auto-Coverage
- Internal Watchdog
- External Watchdog
- CPU Lockstep Compare
- Flash Data ECC
- SRAM Data ECC

### 6.19.1 Notes

It is possible to use the MIBSPI in standard SPI mode.

## 6.20 Serial Peripheral Interface (SPI)

The SPI modules provide serial I/O compliant to the SPI protocol. SPI communications are typically used for communication to smart sensors and actuators, serial memories, and external logic such as a watchdog device. If not used for SPI communication, the SPI modules support the usage of their I/O as general purpose I/O.

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

- Boot Time Software Test of Function Using I/O Loopback
- Periodic Software Test of Function Using I/O Loopback
- Parity in Message
- Information Redundancy Techniques
- Periodic Software Readback of Static Configuration Registers
- Software Readback of Written Configuration
- Transmission Redundancy
- Data Overrun Error Detection
- Bit Error Detection
- Slave Desync Detection
- Slave Timeout Detection
- Data Length Error Detection
- Hardware Disable of JTAG Port
- Boot Time Software Test of Function Using I/O Checking In GIO Mode
- Periodic Software Test of Function Using I/O Checking In GIO Mode

- [Information Redundancy Techniques \(while in GIO Mode\)](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

## 6.21 **Local Interconnect Network (LIN)**

The LIN module provides serial I/O compliant to the LIN protocol. LIN is a low throughput time triggered protocol. The module can also be configured in SCI mode and used as a general purpose serial port. An external transceiver is used for LIN communications.

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

- [Boot Time Software Test of Function Using I/O Loopback](#)
- [Periodic Software Test of Function Using I/O Loopback](#)
- [Information Redundancy Techniques](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)
- [Transmission Redundancy](#)
- [Overrun Error Detection](#)
- [Frame Error Detection](#)
- [Physical Bus Error Detection](#)
- [No-Response Error Detection](#)
- [Bit Error Detection](#)
- [Checksum Error Detection](#)
- [Parity in Message](#)
- [Hardware Disable of JTAG Port](#)
- [Boot Time Software Test of Function Using I/O Checking In GIO Mode](#)
- [Periodic Software Test of Function Using I/O Checking In GIO Mode](#)
- [Information Redundancy Techniques \(while in GIO Mode\)](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

## 6.22 **Controller Area Network (DCAN)**

The DCAN interface provides medium throughput networking with event based triggering, compliant to the CAN protocol. The DCAN modules requires an external transceiver to operate on the CAN network.

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

- [Boot Time Software Test of Function Using I/O Loopback](#)
- [Periodic Software Test of Function Using I/O Loopback](#)
- [Information Redundancy Techniques](#)
- [DCAN SRAM Data Parity](#)
- [Boot Time PBIST Check of DCAN SRAM](#)
- [Periodic PBIST Check of DCAN SRAM](#)
- [Bit Multiplexing in DCAN SRAM Array](#)
- [Periodic Hardware CRC Check of DCAN SRAM Contents](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)
- [Transmission Redundancy](#)
- [Stuff Error Detection](#)
- [Form Error Detection](#)
- [Acknowledge Error Detection](#)
- [Bit Error Detection](#)
- [CAN Protocol CRC in Message](#)
- [Hardware Disable of JTAG Port](#)
- [Boot Time Software Test of Function Using I/O Checking In GIO Mode](#)
- [Periodic Software Test of Function Using I/O Checking In GIO Mode](#)
- [Information Redundancy Techniques \(while in GIO Mode\)](#)

The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Software test of Parity Logic](#)
- [PBIST Test of Parity Bit Memory](#)
- [Software Test of Hardware CRC](#)
- [CRC Auto-Coverage](#)
- [Software Test of PBIST](#)
- [PBIST Auto-Coverage](#)
- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

### **6.23 General-Purpose Input/Output (GIO)**

The GIO module provides digital input capture and digital input/output. There is no processing function in this block. The GIO is typically used for static or rarely changed outputs, such as transceiver enable signals, warning lights, and so forth. The GIO can also be used to provide external interrupt input capabilities.

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

- [Boot Time Software Test of Function Using I/O Checking In GIO Mode](#)
- [Periodic Software Test of Function Using I/O Checking In GIO Mode](#)
- [Information Redundancy Techniques](#)
- [Periodic Software Readback of Static Configuration Registers](#)
- [Software Readback of Written Configuration](#)



The following tests can be applied as a test-for-diagnostic on this module, and could be used to meet Latent Fault Metric Requirements of ISO26262 (in combination with other diagnostics on the primary function):

- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)
- [Flash Data ECC](#)
- [SRAM Data ECC](#)

### 6.23.1 Notes

To reduce probability of common mode failure, the user should consider implementing multiple channels using non adjacent pins.

## 6.24 JTAG Debug and Test Access

The Hercules platform 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 multiplexor logic (ICEPICK), which arbitrates access to test, debug, and calibration logic. Boundary scan is connected in parallel to the ICEPICK to support usage without preamble scan sequences for easiest manufacturing board test.

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

- [Hardware Disable of JTAG Port](#)
- [Lockout of JTAG Access Using AJSM](#)
- [Internal Watchdog](#)
- [External Watchdog](#)

### 6.24.1 Notes

Boundary scan access remains possible even on systems locked by AJSM.

## 6.25 Cortex-R4 Central Processing Unit (CPU) Debug

The Hercules platform supports CPU debug compliant to the ARM® CoreSight standard. Each CoreSight element is accessible over a memory-mapped debug bus, which can be accessed by the CPU or the JTAG port. The CPU debug logic includes an independent debug bus master (AHB-AP), and the debug unit inside the CPU. These modules are not recommended to be used during safety critical operation and safety mechanisms are in place to disable this logic.

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

- [Use of MPUs to Block Access to Memory-Mapped Debug](#)
- [Use of CoreSight Debug Logic Key Enable Scheme](#)
- [Hardware Disable of JTAG Port](#)
- [Lockout of JTAG Access Using AJSM](#)
- [Internal Watchdog](#)
- [External Watchdog](#)
- [CPU Lockstep Compare](#)

## 7 Brief Description of Diagnostics

This section provides a brief summary of the diagnostic mechanisms available on this device. For a detailed description or implementation details for a diagnostic, see the device-specific technical reference manual. For information on safety-pertinent information such as test execution time, error reporting time, action on detected faults, and diagnostic operation modes, see [Appendix A](#).

### 7.1 1002 Software Voting Using Secondary Free Running Counter

The RTI module contains at minimum two up-counters that can be used to provide an operating system time-tick. While one up-counter is used as the operating system timebase, it is possible to use the second up counter as a diagnostic on the first, via periodic check via software of the counter values in the two timers. The PMU CPU cycle counter inside the Cortex-R4 CPU can also be used to support such a diagnostic. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

### 7.2 Bit Error Detection

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

### 7.3 Bit Multiplexing in FEE Memory Array

The FEE modules implemented in the Hercules architecture have a bit multiplexing scheme implemented such that the bits accessed to generate a logical (CPU) word are not physically adjacent. This scheme helps to reduce the probability of physical multi-bit faults resulting in logical multi-bit faults; rather they manifest as multiple single bit faults. As the SECDED FEE ECC can correct a single bit fault in a logical word, this scheme improves the usefulness of the FEE ECC diagnostic. Bit multiplexing is a feature of the architecture and cannot be modified by the software.

### 7.4 Bit Multiplexing in Flash Memory Array

The Flash modules implemented in the Hercules architecture have a bit multiplexing scheme implemented such that the bits accessed to generate a logical (CPU) word are not physically adjacent. This scheme helps to reduce the probability of physical multi-bit faults resulting in logical multi-bit faults; rather they manifest as multiple single bit faults. As the SECDED Flash ECC can correct a single bit fault in a logical word, this scheme improves the usefulness of the Flash ECC diagnostic. Bit multiplexing is a feature of the architecture and cannot be modified by the software.

### 7.5 Bit Multiplexing in Primary SRAM Memory Array

The SRAM modules implemented in the Hercules architecture have a bit multiplexing scheme implemented such that the bits accessed to generate a logical (CPU) word are not physically adjacent. This scheme helps to reduce the probability of physical multi-bit faults resulting in logical multi-bit faults; rather they manifest as multiple single bit faults. As the SECDED SRAM ECC can correct a single bit fault in a logical word, this scheme improves the usefulness of the SRAM ECC diagnostic. Bit multiplexing is a feature of the architecture and cannot be modified by the software.

### 7.6 Bit Multiplexing in Peripheral SRAM Memory Array

This module's SRAM is implemented with a bit multiplexing scheme such that the bits accessed to generate a logical (CPU) word are not physically adjacent. This scheme helps to reduce the probability of physical multi-bit faults resulting in logical multi-bit faults; rather they manifest as multiple single bit faults.

## 7.7 CPU Illegal Operation and Instruction Trapping

The Cortex-R4 CPU includes diagnostics for illegal operations and instructions that can serve as safety mechanisms. Many of these traps are not enabled after reset and must be configured by the software. Installation of software handlers to support the hardware illegal operation and instruction trapping is highly recommended. For more information on enabling traps, see the *Cortex-R4 and Cortex-R4F Technical Reference Manual* located at <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0363e/index.html>. Examples of CPU illegal operation and instruction traps include:

- Illegal instruction
- Privilege violation

## 7.8 CPU Logic Built In Self Test (LBIST)

The Hercules family architecture supports the use of a hardware logic BIST (LBIST) engine self test controller (STC). This logic is used to provide a very high diagnostic coverage on the lockstep CPUs at a transistor level. This logic utilizes the same design for test (DFT) structures inserted into the device for rapid execution of high quality manufacturing tests, but with an internal test engine rather than external automated test equipment (ATE). This technique has proven to be drastically more effective than software-based tests of logic, particularly for the complex logic structures seen in a modern CPU.

The LBIST tests must be triggered by the software. User may elect to run all tests, or only a subset of the tests based on the execution time, which can be allocated to the LBIST diagnostic. This time sliced test feature enables the LBIST to be used effectively as a runtime diagnostic with execution of test time slices per safety critical loop as well as a comprehensive test for CPU fault during MCU initialization.

Execution of the LBIST STC results in a much higher level of transistor switching per clock cycle than during normal software execution due to the high efficiency of the test. A software control is implemented in the STC that allows the user to reduce the CPU clock for the duration of the test. This feature allows the user to make a compromise between fast execution with higher current consumption or slower execution with reduced current consumption.

The LBIST mechanism requires isolation of the CPU from the remainder of device logic while under test. It is also necessary to perform a complete context save before the LBIST. When test execution is complete, the CPU will be reset. The remainder of device logic continues normal operation. After CPU reset, software should read the system control module SYSESR to identify the reason for the reset and can then restore the CPU context.

LBIST logic includes capabilities for testing proper operation of the diagnostic. As the test time for the diagnostic is deterministic, a timeout counter has been included that can detect a failure to complete the test within expected time. In addition, there is the possibility to force an input error to check error detection and propagation of the error response at system level. This test is performed as follows:

1. Enable the self\_check\_key and fault\_ins bits in the STCSTSCR register.
2. Enable STC test interval zero and execute the test
3. Upon completion of test, the fail bit should be set to 1 in the STC global status register.
4. Disable either or both the self\_check\_key and fault\_ins bits in the STCSTSCR register.
5. Restart the self test by programming bit 0 of the STCGCR, causing self test restart.
6. Upon completion of the test, the fail bit should be set to 0 in the STC global status register.

## 7.9 CPU Logic Built In Self Test (LBIST) Auto-Coverage

The LBIST diagnostic is based on a 128-bit signature capture. For a given test, only one code is valid out of  $2^{128}$  possibilities. Therefore, if there is a fault in the LBIST logic, it is extremely unlikely that the correct passing code will be generated via the fault. The cyclical check applied by the LBIST module provides an inherent level of self checking (auto-coverage), which can be considered for application in latent fault diagnostics.

### 7.10 CPU Lockstep Compare

The Hercules product family includes a lockstep processor diagnostic. This feature includes the addition of a diagnostic Cortex-R4 CPU that is combined into a 1oo1D (single channel with diagnostic channel) configuration with the application CPU. Both the application CPU and the diagnostic CPU are fed the same input signals, which results in both CPUs running the same software. The diagnostic and application CPUs should generate the same output. The core compare module (CCM) compares the CPU outputs and flags all mis-compares to the ESM.

The lockstep diagnostic is continually operating from power-on reset. The lockstep checking is disabled when the CPU is placed into a halting debug state and can only be restored to operation after a subsequent reset. Lockstep functionality can also be disabled temporarily when executing the self test checking functionality of the CCM.

During the first cycles of CPU operation after reset, it is necessary to execute a short initialization code that sets all CPU registers to a known state. This code sequence can be found in the device-specific data sheet. Execution of this code sequence as the first instructions out of reset is mandatory.

The CCM logic provides self test and error forcing capability via software triggered hardware. The self test ensures that the CCM compare logic is working properly. The error forcing capability allows you to test the system level response to a lockstep mis-compare.

### 7.11 CPU Lockstep Comparator Self Test

The CPU lockstep comparator (CCM) diagnostic includes self test features to check for proper operation of the lockstep comparator.

### 7.12 CPU Online Profiling Using the Performance Monitoring Unit

The Cortex-R4 CPU includes a performance monitoring unit (PMU). This logic is intended to be used for debug and code profiling purposes, but it can also be utilized as a safety mechanism. The PMU includes a CPU cycle counter as well as three additional counters, which can be programmed to count a number of different CPU events. For a complete list of CPU events that can be monitored, see the *Cortex-R4 and Cortex-R4F Technical Reference Manual* located at <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0363e/index.html>. Examples of the CPU events that can be monitored include:

- Number of cycles in which an ECC or parity error is detected by diagnostics in CPU
- Number of cycles in which the CPU is in the livelock state
- Number of instructions executed
- Number of cycles in which an exception is taken

With such information available, it is possible to generate a software routine that periodically checks the PMU counter values and compares these values to the profile expected during normal operation. The PMU is not enabled by default and must be configured via software. As the PMU is implemented internal to the CPU, it is checked for proper operation on a cycle by cycle basis by the lockstep diagnostic and can also be checked via execution of the LBIST STC diagnostic.

### 7.13 CPU Memory Protection Unit (MPU)

The Hercules implementation of the Cortex-R4 CPU includes an MPU. The MPU logic can be used to provide spatial separation of software tasks in the device memory. It is expected that the operating system controls the MPU and changes the MPU settings based on the needs of each task. A violation of a configured memory protection policy results in a CPU abort.

The MPU can also be used to configure the memory ordering policies of the memory system. By default, all peripheral accesses are strongly ordered type - meaning that all transactions complete in the sequence they are issued and no write transactions are buffered. If desired, the operating system can configure accesses to be device type - meaning that writes are buffered. This can improve performance over a strongly ordered model, at the cost of some determinism. It is highly recommended that the system control module and other modules deemed to have critical configurations be set to a strongly ordered access model.

As the MPU is internal to the CPU core, proper operation is checked via the lockstep CPU mechanism. In addition, the LBIST STC diagnostic provides a check of the MPU when it performs a test of the CPU.

#### **7.14 CRC Auto-coverage**

The CRC diagnostic is based on a 64-bit polynomial. For a given test, only one code is valid out of  $2^{64}$  possibilities. Therefore, if there is a fault in the CRC logic, it is extremely unlikely that the correct passing code will be generated via the fault.

#### **7.15 CRC in Message**

This module includes a CRC in Message safety mechanism. The CRC values are calculated and transmitted by the transmitter, and then re-calculated by the receiver. If the CRC value calculated by the receiver does not match the transmitted CRC value, a CRC error will be flagged. Error response and any necessary software requirements are defined by the system integrator.

#### **7.16 DCAN Acknowledge Error Detection**

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

#### **7.17 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 violates the protocol, the module will flag a Form Error. Error response and any necessary software requirements are defined by the system integrator.

#### **7.18 DCAN Stuff Error Detection**

In the CAN message protocol, several of the frame segments are coded through bit stuffing. Whenever a transmitter detects five consecutive bits of identical value in the bitstream to be transmitted, it automatically inserts a complementary bit into the actual transmitted bit stream. If a 6th consecutive equal bit is detected in a received segment that should have been coded by bit stuffing, the DCAN module will flag a Stuff Error. Error response and any necessary software requirements are defined by the system integrator.

#### **7.19 DCAN Protocol CRC in Message**

The CAN Protocol includes a CRC in Message. The CRC values are calculated and transmitted by the transmitter, and then re-calculated by the receiver. If the CRC value calculated by the receiver does not match the transmitted CRC value, a CRC error will be flagged. Error response and any necessary software requirements are defined by the system integrator.

#### **7.20 Dual Clock Comparator (DCC)**

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

The DCC diagnostic is not enabled by default and must be enabled via software. It is possible to disable and configure this diagnostic via software. The cyclical check applied by the DCC module provides an inherent level of self checking (auto-coverage), which can be considered for application in latent fault diagnostics.



### **7.21 Efuse Autoload Self Test**

The EFuse controller has a self test logic that executes automatically after autoload completion. This is not a diagnostic that can be run separately, but is a given when Autoload self test executes. The test can be subsequently triggered by the software. Error is indicated via ESM.

### **7.22 Efuse Autoload Self Test Auto-Coverage**

As a test-for-diagnostic the Autoload self test logic implementation provides test coverage for its logic. This is not a diagnostic that can be run separately, but is a given when Autoload self test executes.

### **7.23 EFuse ECC**

The EFuses utilize a SECDED ECC diagnostic to detect (and correct) incorrect configuration values. Errors are indicated via ESM. Use of the EFuse ECC diagnostic is mandatory. The cyclical check applied by the ECC logic provides an inherent level of self checking (auto-coverage), which can be considered for application in latent fault diagnostics.

### **7.24 EFuse ECC Logic Self Test**

As a test-for-diagnostic, it is possible to test the functionality of the Efuse ECC logic by performing a self test supported by hardware and checking for errors

### **7.25 eQEP Quadrature Watchdog**

The 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 VCLK/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 will time out and the watchdog interrupt flag will be set. The time-out value is programmable through the watchdog period register.

### **7.26 eQEP Software Test of Quadrature Watchdog Functionality**

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

### **7.27 Error Trapping - IOMM**

The IOMM can trap address and privilege errors on received transactions. Transactions that attempt to access un-implemented locations in the IOMM chip select result in an ESM response. Transactions that are not in privileged mode also generate an ESM response.

This feature is enabled after reset. Software cannot disable this feature.

### **7.28 Error Trapping (including Peripheral Slave Error Trapping) - L2/L3 Interconnect**

The L2 and L3 interconnect subsystem includes a number of mechanisms to detect and trap errors. Address decoders in the diagnostic respond with a bus error to the initiator if a bus transaction does not decode to a valid target. Logic is also present that can detect the timeout of certain transactions and respond with a bus error to the transaction initiator. In addition to testing for slaves directly connected to the L2 interconnect, peripheral slaves which are connected to L3 (PCR) peripheral interconnect can support error trapping.

The interconnect error trapping functionality is enabled by default and cannot be disabled by the software. These features can be tested via software through the insertion of invalid bus transactions.

### 7.29 External Monitoring of Warm Reset (nRST)

The nRST warm reset signal is implemented as an I/O. An external monitor can be utilized to detect expected or unexpected changes to the state of the internal warm reset control signal. Error response, diagnostic testability, and any necessary software requirements are defined by the external monitor selected by the system integrator.

### 7.30 External Monitoring via ECLK

The Hercules platform provides the capability to export select internal clocking signals for external monitoring. This feature can be configured via software by programming registers in the system control module. To determine the number of external clock outputs implemented and the register mapping of internal clocks that can be exported, see the device-specific data sheet.

Export of internal clocks on the ECLK outputs is not enabled by default and must be enabled via software. It is possible to disable and configure this diagnostic via software.

### 7.31 External Voltage Supervisor

The Hercules platform highly recommends the use of an external voltage supervisor to monitor all voltage rails. The voltage supervisor should be configured with overvoltage and undervoltage thresholds matching the voltage ranges supported by 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.

### 7.32 External Watchdog

When using an external watchdog, there is a possibility to reduce common mode failure with the MCU clocking system, as the watchdog can utilize clock, reset, and power that are separate from the system being monitored. Error response, diagnostic testability, and any necessary software requirements are defined by the external watchdog selected by the system integrator. The Hercules platform highly recommends the use of an external watchdog over the internally provided watchdogs.

An internal or external watchdog can provide an indication of inadvertent activation of logic which results in impact to safety critical execution. Any watchdog added externally should include a combination of temporal and logical monitoring of program sequence or other appropriate methods such that high diagnostic effectiveness can be claimed.

### 7.33 FEE Contents Check by Hardware CRC

The platform includes a hardware CRC implementing the ISO CRC-64 standard polynomial. The CRC module can be used to test the integrity of FEE contents by calculating a CRC for all FEE contents and comparing this value to a previously generated "golden" CRC. The read of FEE contents to the CRC can be done by the CPU. The comparison of results, indication of fault, and fault response are the responsibility of the software managing the test. Depending on the software driver scheme used to support FEE, it may be necessary to execute this check driven by the CPU FEE driver. It is highly recommended to perform a CRC integrity check of FEE contents at boot time. The cyclical check applied by the hardware CRC module provides an inherent level of self checking (auto-coverage), which can be considered for application in latent fault diagnostics.

### 7.34 FEE Data ECC

The on-chip FEE memory is supported by SECDED ECC diagnostic. Unlike the main Flash memory, the FEE memory utilizes a local SECDED ECC controller implemented in the Flash wrapper. This allows extra flexibility needed to support EEPROM emulation, but does leave the full transaction path between CPU and FEE memory without end-to-end diagnostics.

The FEE SECDED ECC controller utilizes the same ECC algorithm as used in the main Flash memory; 8-bit of code are implemented per 64-bit of data. All detection of ECC faults is performed in the Flash wrapper. Error response is provided via bus error to the CPU and an error signal to ESM. The error response has added programmability to support multiple EEPROM emulation strategies desired by TI customers. Fault addresses are logged in the Flash wrapper.



The ECC logic for the FEE is disabled at reset and must be configured in the Flash wrapper. The cyclical check applied by the FEE ECC logic provides an inherent level of self checking (auto-coverage), which can be considered for application in latent fault diagnostics.

### 7.35 FEE Sector Protection

It is possible to prevent a write operation on an FEE sector by the software configuration of the Flash wrapper. The sector protection registers, BSE, contain a bit for each sector in the FEE bank, which enables or disables a sector for write operation. The BSE register can only be written in privilege mode while the software PROTLIDIS protection bit is set high. This mechanism can reduce probability of unintended programming of FEE memory.

### 7.36 Flash ATCM Address and Control Bus Parity

The on-chip ATCM bus connection to Flash memory is supported by a parity diagnostic on the address and control signals. The parity is generated by the CPU and evaluated by the Flash wrapper. Detected errors are signaled to the ESM by the Flash wrapper and the error address is captured in the Flash wrapper.

This diagnostic is enabled at reset. The diagnostic can be disabled by programming of the BUS\_PAR\_DIS key in the FPAR\_OVR register of the Flash wrapper.

### 7.37 Flash Contents Check by Hardware CRC

The platform includes a hardware cyclic redundancy check (CRC) implementing the ISO CRC-64 standard polynomial. The CRC module can be used to test the integrity of Flash contents by calculating a CRC for all Flash contents and comparing this value to a previously generated "golden" CRC. The read of Flash contents to the CRC can be done by CPU. The comparison of results, indication of fault, and fault response are the responsibility of the software managing the test. It is highly recommended to perform a CRC integrity check of Flash contents at boot time. The cyclical check applied by the hardware CRC module provides an inherent level of self checking (auto-coverage), which can be considered for application in latent fault diagnostics.

### 7.38 Flash ECC

The on-chip Flash memory is supported by single error correction, dual error detection (SECEDED) error-correcting code (ECC) diagnostic. It is connected by a 64-bit-wide data bus interface (ATCM) to the Cortex-R4 CPU. In this SECEDED scheme, an 8-bit code word is used to store the ECC data as calculated over the 64-bit data bus.

The ECC logic for the ATCM Flash access is located in the Cortex-R4 CPU. All ATCM transactions have ECC on the data payload. ECC evaluation is done by the ECC control logic inside the CPU. This scheme provides end-to-end diagnostics on the transmissions between the CPU and Flash memory. Detected uncorrectable errors result in a processor abort or bus error depending on the requesting master. Detected correctable errors can be corrected or not corrected, depending on whether correction functionality is enabled. The address of the memory, which includes the ECC error, is logged in the CPU. For more details, see the *Cortex-R4 and Cortex-R4F Technical Reference Manual* located at <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0363e/index.html>.

It is possible to export error detection events from the CPU to the Flash wrapper, and then from the Flash wrapper to the ESM. This functionality is not enabled by default and must be configured by the software. The Cortex-R4 PMU must first be set to export events to an external monitor. Then the Flash wrapper must be configured to export the correctable and uncorrectable events to the ESM.

The ECC logic for the Flash is disabled at reset and must be configured in both the CPU and the Flash wrapper. The diagnostic has separate controls for checking, correction, and read and modify and write functionality in the system control coprocessor that must be enabled via software. As the ECC diagnostic is implemented inside the CPU, its behavior is continuously checked via the lockstep functionality and can also be tested via the LBIST STC. The cyclical check applied by the ECC module provides an inherent level of self checking (auto-coverage), which can be considered for application in latent fault diagnostics.

### 7.39 Flash Hard Error Cache and Livelock

If ECC correction is enabled, corrected data values are stored in an internal one entry hard error cache. It is not possible to automatically re-write the corrected data values back to the Flash, as the Flash is a non-volatile memory that has special programming requirements.

A single instruction and its data may not have more than one correctable error. In case more than one correctable error is detected, it is possible to overrun the hard error cache and put the processor into an inoperable livelock state. Cases that can generate a livelock include:

- Two single bit errors in a 64-bit unaligned 32-bit Thumb-2 instruction fetch
- A single bit error in a load instruction (LDR or LDM) followed by a single bit error in the instruction's data payload

Livelock is indicated via the ESM and typically requires a reset for recovery to be attempted. Livelock on a Flash interface transaction can be an indication of severe permanent fault in the Flash memory.

Use of the hard error cache and livelock functionality is mandatory. This feature is enabled at reset and cannot be disabled by the software.

### 7.40 Flash Sector Protection

It is possible to prevent a write operation on a sector by the software configuration of the Flash wrapper. The sector protection registers, Bank Sector Enable Register (BSE), contain a bit for each sector in the Flash bank, which enables or disables a sector for write operation. The BSE register can only be written in privilege mode while the software PROTLIDIS protection bit is set high. This mechanism can reduce probability of unintended programming of Flash memory.

### 7.41 Flash Wrapper Address ECC

In addition to the ECC functionality in the CPU, the Flash wrapper extends the ECC capability to include address. The standard SECDED ECC scheme utilized has 8-bit of code for 64-bit of data. Extensions of the Hamming equations allow additional bits of data beyond 64 bits to be encoded into 8 bits of code with the same Hamming distance. The Flash wrapper design takes advantage of this by incorporating the TCM bus address of the transaction into the Flash memory ECC calculation. All values stored in the Flash memory have the address added into the Flash ECC. Upon read of data from Flash, the Flash wrapper will strip the address component from the ECC and provide the regenerated ECC code to the CPU. Errors in the address result in a multi-bit ECC failure in the CPU.

The address ECC feature is disabled at reset. There are no separate controls for this feature. This feature is enabled whenever ECC is enabled in the Flash wrapper. The cyclical check applied by the Flash wrapper address ECC module provides an inherent level of self checking (auto-coverage), which can be considered for application in latent fault diagnostics.

### 7.42 Flash Wrapper Diag Mode 1 Test

This diagnostic mode provides the capability to test the ECC logic in the Flash Wrapper by corrupting the input data for ECC comparison. This diag mode can be used to test the FEE and OTP interfaces.

### 7.43 Flash Wrapper Diag Mode 2 Test

This diagnostic mode provides the capability to check the generated ECC syndrome. This diag mode can be used to test the FEE and OTP interfaces.

### 7.44 Flash Wrapper Diag Mode 3 Test

This diagnostic mode provides the capability to test the ECC logic in the Flash Wrapper by corrupting the generated ECC syndrome. This diag mode can be used to test the FEE and OTP interfaces.

### 7.45 Flash Wrapper Diag Mode 4 Test

This diagnostic mode provides the capability to test the ECC logic in the Flash Wrapper by corrupting the ECC bits. This diag mode can be used to test the FEE and OTP interfaces.

### **7.46 Flash Wrapper Diag Mode 5 Test**

An error can be injected into the lockstep address tag registers to cause a mismatch between primary and duplicate registers. This diag mode can be used to test the Primary Flash interface.

### **7.47 Flash Wrapper Diag Mode 7 Test**

In order to test the ECC logic of the CPU incorrect ECC can be fed into the CPU. In this diagnostic mode one or more specific bits can be flipped. This diag mode can be used to test the Primary Flash interface.

### **7.48 Glitch Filtering on nRST and nPORRST**

Glitch filters are implemented on the cold and warm reset pins of the device. These structures filter out noise and transient signal spikes on the input reset pins in order to reduce unintended activation of the reset circuitry. The glitch filters are continuously operating and their behavior cannot be changed by the software. Use of the glitch filters is mandatory.

### **7.49 Hardware CRC Check of OTP Contents**

The platform includes a hardware cyclic redundancy check (CRC) implementing the ISO CRC-64 standard polynomial. The CRC module can be used to test the integrity of OTP contents by calculating a CRC for all Flash contents and comparing this value to a previously generated "golden" CRC. The read of Flash contents to the CRC can be done by the CPU. The comparison of results, indication of fault, and fault response are the responsibility of the software managing the test. It is highly recommended to perform a CRC integrity check of OTP contents at boot time.

### **7.50 Hardware Disable of JTAG Port**

The JTAG debug port can be physically disabled to prevent JTAG access in deployed systems. The recommended scheme is to hold test clock (TCK) to ground and hold test mode select (TMS) high, though alternate schemes are possible.

Disabling of the JTAG port also provides coverage for inadvertent activation of many debug activities, since these are often initiated via an external debug tool which writes commands to the device using the JTAG port.

### **7.51 Information Redundancy Techniques**

Information redundancy techniques can be applied via software as an additional runtime diagnostic for this module. There are many techniques that can be applied, such as read back of written values and multiple output of the same data with comparison of results. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

### **7.52 Information Redundancy Techniques - L2/L3 Interconnect Specific**

Information redundancy techniques can be applied via software as an additional runtime diagnostic on the L2 and L3 interconnect. There are many techniques that can be applied, such as read back of written values and multiple reads of the same target data with comparison of results. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

### **7.53 Information Redundancy Techniques - DCAN Specific**

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

In order to provide diagnostic coverage for network elements outside the MCU (wiring harness, connectors, transceiver) end-to-end safing mechanisms are applied. These mechanisms can also provide diagnostic coverage inside the MCU. There are many different schemes applied, such as additional message checksums, redundant transmissions, time diversity in transmissions, and so forth. Most commonly checksums are added to the payload section of a transmission to ensure the correctness of a transmission. These checksums are applied in addition to any protocol level parity and checksums. As the checksum is generated and evaluated by the software at either end of the communication, the whole communication path is safed, resulting in end-to-end safing.

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

#### **7.54 Information Redundancy Techniques - N2HET Specific**

Information redundancy techniques can be applied via software or system as an additional runtime diagnostic on N2HET operations. There are many techniques that can be applied, such as multiple sampling of a single input channel, sampling of two or more input channels, and read back of written output.

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

#### **7.55 Internal Voltage Monitor (VMON)**

The Hercules platform incorporates a simple embedded voltage monitor (VMON) that can detect grossly out of range supply voltages. The VMON operates continuously and requires no software configuration or CPU overhead. VMON monitors the core and I/O supplies. When the supplies are grossly over or under specified voltages (for product specific values, see the device-specific data sheet), the VMON drives the nPORRST (power-on reset) signal internally. This response holds the device in the safe operating state. When power supplies are in range, the VMON will not interfere with the nPORRST signal. For more information on VMON operation, see the device-specific data sheet.

The VMON is a continuously operating diagnostic. It is not possible to disable the VMON diagnostic. In-system test of the VMON diagnostic is generally not feasible as tight control of external voltages is needed to trigger the VMON error response. If improperly applied, such voltage could result in permanent damage to the MCU. Use of the VMON is mandatory.

#### **7.56 Internal Watchdog**

The Hercules platform supports the use of an internal watchdog that is implemented in the real time interrupt (RTI) module. The internal watchdog has two modes of operation: digital watchdog (DWD) and digital windowed watchdog (DWWD). The modes of operation are mutually exclusive, the designer can elect 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 DWD is a traditional single threshold watchdog. The user programs a timeout value to the watchdog and must provide a predetermined "pet" response to the watchdog before the timeout counter expires. Expiration of the timeout counter or an incorrect "pet" response triggers an error response. The DWD can issue either an internal (warm) system reset or a CPU non-maskable interrupt upon detection of a failure. The DWD is not enabled after reset. Once enabled by the software, the DWD cannot be disabled except by system reset or power-on reset. Users should take care not to change the DWD clock source after enabling the module, else behavior is unpredictable.

The DWWD is a traditional windowed watchdog. The user programs an upper bound and lower bound to create a time window during which the software must provide a predetermined "pet" response to the watchdog. Failure to receive the correct response within the time window or an incorrect "pet" response triggers an error response. The DWWD can issue either an internal (warm) system reset or a CPU non-maskable interrupt upon detection of a failure. The DWWD is not enabled after reset. Once enabled by the software, the DWWD cannot be disabled except by system reset or power-on reset. The use of the time window allows detection of additional clocking failure modes as compared to the DWD implementation.

An internal or external watchdog can provide an indication of inadvertent activation of logic which results in impact to safety critical execution, such as an inadvertent activation of JTAG logic.

### **7.57 IOMM Master ID Filtering**

The IOMM checks the master ID of all incoming bus transactions. Only transactions from the CPU are allowed. Illegal transactions result in a bus error response to the offending bus master and the write to IOMM will be discarded. This feature is enabled after reset. Software cannot disable this feature. Use of the master ID filtering is mandatory.

### **7.58 LIN Checksum Error Detection**

After message reception, the LIN performs a CRC check on all received data bytes. This data is then compared to the checksum for the message that was transmitted by the sender. If there is a mismatch, the LIN hardware can flag a Checksum Error and generate an interrupt to the CPU. This feature must be enabled and configured in software.

### **7.59 LIN No-Response Error Detection**

If no response is received within a configurable time period after the LIN master completes a header transmission, the LIN hardware can flag a No-Response Error and generate an interrupt to the CPU. This feature must be enabled and configured in software.

### **7.60 LIN Physical Bus Error Detection**

If no message can be generated on the LIN Bus, the LIN hardware can flag the error and generate a physical bus error and generate an interrupt to the CPU. This feature must be enabled and configured in software.

### **7.61 LIN / SCI Bit Error Detection**

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

### **7.62 LIN / SCI Frame Error Detection**

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

### **7.63 LIN / SCI Overrun Error Detection**

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

### **7.64 Locking Mechanism for Control Registers**

The IOMM contains a two stage lock mechanism for protection of critical control registers. To change the configuration of the pin multiplexing, the user must write two specific 32-bit values to the "kicker" registers in a defined sequence. When complete, the lock function must be reset. Write accesses, when the registers are locked, will not update the registers, nor will they generate an error response. This feature is enabled after reset. Software can disable the lock by unlocking and never re-locking.

### **7.65 Lockout of JTAG Access Using AJSM**

The Hercules platform includes the Advanced JTAG Security Module (AJSM) as support for managing debug access on deployed devices. AJSM can be used to program a unique access key to the OTP Flash memory. Subsequent debug accesses must unlock the AJSM with the correct key sequence in order to gain access to the JTAG-based debug, logic. An error in unlocking the AJSM results in no error response and no access to the debug logic.



### **7.66 Low Power Oscillator Clock Detector (LPOCLKDET)**

The low-power oscillator clock detector (LPOCLKDET) is a safety diagnostic that can be used to detect failure of the primary clock oscillator. LPOCLKDET utilizes the embedded high-frequency, low-power oscillator (HF LPO). The clock detect circuit works by checking for a rising edge on one clock (oscillator or HF LPO) between rising edges of the other clock. The result is that in addition to flagging incorrect, repeating frequencies, the circuit also fails due to transient conditions. The low end of the clock detect window ignores a transient low phase of at least 12 HF LPO cycles. Note that this filtering of the transient response does not change the input frequency range. The LPOCLKDET circuitry is enabled by default during the power-on reset state. The diagnostic can be disabled via software.

### **7.67 Memory Protection Unit (MPU) for Non-CPU Bus Masters**

Many non-CPU bus masters include an MPU. The MPU logic can be used to provide spatial separation of software tasks in the device memory. It is expected that the module's software driver controls the MPU and changes the MPU settings based on the needs of system implementation. A violation of a configured memory protection policy will result in an ESM error. The MPU is not enabled at reset. Software must enable, configure and test the MPU.

### **7.68 Memory Protection Unit (MPU) for Cortex R4 CPU**

The Hercules implementation of the Cortex-R4 CPU includes an MPU. The MPU logic can be used to provide spatial separation of software tasks in the device memory. It is expected that the operating system controls the MPU and changes the MPU settings based on the needs of each task. A violation of a configured memory protection policy results in a CPU abort. The MPU can also be used to configure the memory ordering policies of the memory system. By default, all peripheral accesses are strongly ordered - meaning that all transactions complete in the sequence are issued and no write transactions are buffered. If desired, the operating system can configure accesses to be device - meaning that writes are buffered. This can improve performance over a strongly ordered model, at the cost of some determinism. It is highly recommended that the system control module and other modules deemed to have critical configurations be set to a strongly ordered access model. As the MPU is internal to the CPU core, proper operation is checked via the lockstep CPU mechanism. In addition, the LBIST STC diagnostic provides a check of the MPU when it performs a test of the CPU.

### **7.69 MibADC Converter Calibration**

The Hercules MibADC module implements calibration logic normally used to improve converter accuracy. This logic can also be used as a safety mechanism. Software comparison of the conversion of known reference values from the calibration logic can provide a diagnostic on converter functionality. Repeated execution of the calibration routine can be used to detect drift during application. Software must configure and enable and evaluate the result of this diagnostic. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

### **7.70 MibADC Information Redundancy Techniques**

Information redundancy techniques can be applied via software as an additional runtime diagnostic on ADC conversions. There are many techniques that can be applied, such as multiple conversions using multiple channels on the same converter or by doing multiple conversions on the same channel followed with comparison of results. Filtering or a plausibility check for the converted values are in expected range can also improve diagnostic coverage. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

### **7.71 MibADC Input Self Test**

The Hercules MibADC module implements an input self test engine that can detect short to ADREFLO, ADREFHI or open input. Software must configure and enable and evaluate the result of this diagnostic. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

### **7.72 MibSPI/SPI Data Length Error Detection**

The MibSPI or SPI can detect an error in the length of either a received or a transmitted message. If the transmitted or received message does not match the expected length as specified in the character counter, the MibSPI or SPI module can flag a Data Length error and signal an interrupt to the CPU. This feature must be enabled and configured in software.

### **7.73 MibSPI/SPI Data Overrun Detection**

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

### **7.74 MibSPI/SPI Slave Desync Detection**

When a slave module supports the generation of an enable signal (ENA), slave de-synchronization can be detected. De-synchronization occurs when the slave misses one or more MibSPI or SPI clock edges coming from the master during transmission. In this case the slave would continue to wait for the remaining clock edges, while holding the ENA signal active. The MibSPI can detect this condition through the use of a configurable counter which starts after message transmission is complete. If the ENA signal coming from the slave does not become inactive by the time the counter overflows, the MibSPI or SPI module can flag a slave desynch error and signal an interrupt to the CPU. This feature must be enabled and configured in software.

### **7.75 MibSPI/SPI Slave Timeout Detection**

When a slave supports the generation of an enable signal (ENA), slave timeout can be detection. Slave timeout occurs when the SPI Master generates a handshake signal to indicate the beginning of a transmission, but the slave does not respond. The MibSPI or SPI can detect this condition through the use of a configurable counter which starts after the handshake transmission. If the slave does not respond by activating the ENA signal by the time the counter expires, the MibSPI or SPI module can flag a Slave Timeout error and signal an interrupt to the CPU. This feature must be enabled and configured in software.

### **7.76 Non-Privileged Bus Master Access**

This module is a non-privileged bus master. Since writes to critical configuration registers across the MCU is limited to privileged mode transactions only, this mechanism prevents inadvertent configuring of these registers by this module. The operation of this safety mechanism is continuous and cannot be altered by the software. This mechanism can be tested by generating software transactions and reviewing the device response. Use of this safety mechanism is mandatory.

### **7.77 OTP Autoload ECC**

The OTP autoload controller utilizes a SECDED ECC diagnostic to detect (and correct) incorrect configuration values. The OTP autoload is executed after every power-on reset. Errors are indicated via ESM. The cyclical check applied by the ECC logic provides an inherent level of self checking (auto-coverage), which can be considered for application in latent fault diagnostics.

### **7.78 Parity in Message**

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



### **7.79 Periodic Hardware CRC Check of OTP Contents**

The platform includes a hardware cyclic redundancy check (CRC) implementing the ISO CRC-64 standard polynomial. The CRC module can be used to test the integrity of OTP contents by calculating a CRC for all Flash contents and comparing this value to a previously generated "golden" CRC. The read of Flash contents to the CRC can be done by CPU. 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 hardware CRC module provides an inherent level of self checking (auto-coverage), which can be considered for application in latent fault diagnostics.

### **7.80 Periodic Software Test of VIM Functionality Including Error Tests**

Per guidance found in IEC 61508, a software test for detecting basic functionality as well as for errors such as continuous interrupts, no interrupts, and crossover interrupts can be implemented. Such testing could include PMU profiling of the expected average number of interrupts received over a given time period, software forced interrupts to check VIM and CPU response, or periodic interrupts from the RTI module specifically for the purpose of VIM testing. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

### **7.81 Peripheral Central Resource (PCR) Access Management**

The peripheral central resource (PCR) provides two safety mechanisms that can limit access to peripherals. Peripherals can be clock gated per peripheral chip select in the PCR. This can be utilized to disable unused features such that they cannot interfere with active safety functions. In addition, each peripheral chip select can be programmed to limit access based on privilege level of transaction. This feature can be used to limit access to entire peripherals to privileged operating system code only.

These safety mechanisms are disabled after reset. Software must configure and enable these mechanisms.

### **7.82 Periodic Hardware CRC Check of SRAM Contents**

The platform includes a hardware CRC implementing the ISO CRC-64 standard polynomial. The CRC module can be used to test the integrity of static contents in the SRAM by calculating a CRC for all static contents and comparing this value to a previously generated "golden" CRC. The read of SRAM contents to the CRC can be done by CPU. 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.

### **7.83 Periodic Software Read Back of Static Configuration Registers**

Periodic read back of configuration registers can provide a diagnostic for inadvertent writes or disturb of these registers. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

### **7.84 Peripheral SRAM Parity**

This module's SRAM includes a parity diagnostic that can detect single bit errors in the memory. When a parity error is detected, the ESM is notified. This feature is disabled after reset. Software must configure and enable this feature.

### **7.85 PLL Slip Detector**

The PLL logic includes an embedded diagnostic that can detect a slip of the PLL output clock. The slip is a result of a loss of phase lock between reference clock and feedback clock. Error response and indication is dependent on the programming of the PLL control registers that are located in the system control module. ESM error indication can be generated or masked. In addition, it is possible to generate an internal reset or to revert to operation from the oscillator clock in case of a detected error. For more information on programming this diagnostic, see the device-specific technical reference manual.

The PLL slip detection diagnostic is active whenever the PLL is enabled and has locked on a target frequency. The diagnostic cannot be disabled by the software, but the error indication and error response can be modified by the software.

### 7.86 Primary SRAM BTCM Address and Control Bus Parity

The on-chip BTCM bus connections to SRAM are supported by a parity diagnostic on the address and control signals. The parity is generated by the CPU and evaluated by the SRAM. Detected errors are signaled to the ESM by the SRAM and the error address is captured in the SRAM wrapper.

This diagnostic is enabled at reset. The diagnostic can be disabled by programming of the address parity disable key in the RAMCTRL register of the SRAM wrapper.

### 7.87 Primary SRAM Data and ECC Storage in Multiple Physical Banks per Logical Address

Each logical SRAM word and its associated ECC code is split and stored in two physical SRAM banks. Each access comprises 72 total bits - 64 bits of data and 8 bits of ECC code. This is split in half, with each physical SRAM storing 32 bits of data and 4 bits of ECC code. Figure 9 provides a graphical view of this partition.

This scheme provides an inherent safety mechanism for address decode failures in the physical SRAM banks. Faults in the bank addressing are detected by the CPU as an ECC fault. Use of the diverse data and ECC storage in SRAM is mandatory. This feature is enabled at reset and cannot be disabled by the software.

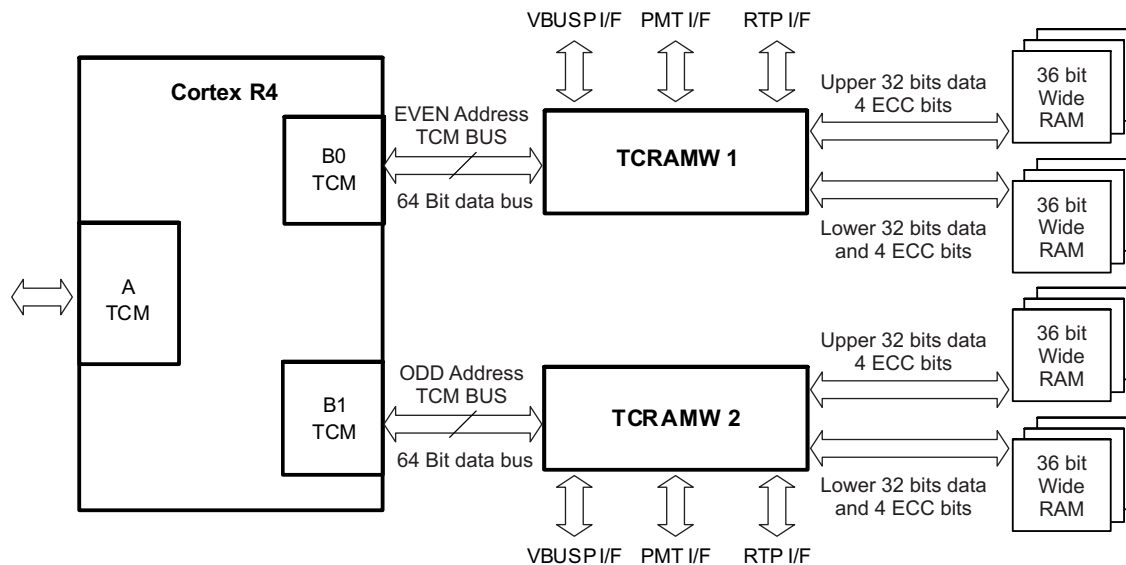


Figure 9. Block Level Implementation of CPU SRAM

### 7.88 Primary SRAM Correctable ECC Profiling

The SRAM wrapper includes a capability to count the number of correctable ECC errors detected. When the error count exceeds a user programmed threshold, an error event is signaled to the ESM. This mechanism is disabled by default and must be enabled by the software in the SRAM wrapper. Cortex-R4 PMU export of events must also be enabled for this function to operate.

### 7.89 Primary SRAM Data ECC

The on-chip SRAM is supported by SECDED ECC diagnostic. It is connected by a 64-bit-wide data bus interface (BTCM0 or BTCM1) to the Cortex-R4 CPU. In this SECDED scheme, an 8-bit code word is used to store the ECC data as calculated over the 64-bit data bus.

The ECC logic for the BTCM SRAM access is located in the Cortex-R4 CPU. All BTCM transactions have ECC on the data payload. ECC evaluation is done by the ECC control logic inside the CPU. This scheme provides end-to-end diagnostics on the transmissions between CPU and SRAM. Detected uncorrectable errors result in a processor abort or bus error depending on the requesting master. Detected correctable errors can be corrected or not corrected, depending on whether correction functionality is enabled. The address of the memory that includes the ECC error will be logged in the CPU. For more details, see the *Cortex-R4 and Cortex-R4F Technical Reference Manual* located at <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0363e/index.html>.

It is possible to export error detection events from the CPU to the SRAM wrapper, and then from the SRAM wrapper to the ESM. This functionality is not enabled by default and must be configured by the software. The Cortex-R4 PMU must first be set to export events to an external monitor. Then, the SRAM wrapper must be configured to export the correctable and uncorrectable events to the ESM.

The ECC logic for the SRAM is disabled at reset and must be configured in the CPU. The diagnostic has separate controls for checking, correction, and read and modify and write functionality in the system control coprocessor that must be enabled via software. As the ECC diagnostic is implemented inside the CPU, its behavior is continuously checked via the lockstep functionality and can also be tested via the LBIST STC. The cyclical check applied by the ECC logic provides an inherent level of self checking (auto-coverage), which can be considered for application in latent fault diagnostics.

### 7.90 Primary SRAM Wrapper Redundant Address Decode

The SRAM wrapper includes a diagnostic to check for errors in the address decode logic. The diagnostic implements a checker copy of the address decode logic in lockstep to the functional address decode logic. The SRAM wrapper includes comparator logic to detect differences in the output from the two address decoders. Any detected mismatch will be signaled to the ESM and the address will be captured to a local register.

The redundant address decode logic diagnostic is active after reset. The diagnostic is supported by a hardware self test that is triggered via software. The compare function is disabled during self test operation.

### 7.91 Primary SRAM Hard Error Cache and Livelock

If correction is enabled, corrected data values are stored in an internal one entry hard error cache, rewritten to the SRAM, and re-fetched from the SRAM.

A single instruction and its data may not have more than one correctable error. In case more than one correctable error is detected, it is possible to overrun the hard error cache and put the processor into an inoperable livelock state. Cases that can generate a livelock include:

- Two single bit errors in a 64-bit unaligned 32-bit Thumb-2 instruction fetch
- A single bit error in a load instruction (LDR or LDM) followed by a single bit error in the instruction's data payload

Livelock is indicated via the ESM and typically requires a reset for recovery to be attempted. Livelock on a SRAM interface transaction can be an indication of severe permanent fault in the SRAM. This feature is enabled at reset and cannot be disabled by the software.

### 7.92 Privileged Mode Access and Multi-Bit Enable Keys for Control Registers

This module includes features to support avoidance of unintentional control register programming. These features include limitation of write commands to privilege bus master transactions and the implementation of multi-bit keys for critical controls. The multi-bit keys are particularly effective for avoiding unintentional activation. For more details on the register safety mechanisms and error response, 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 and reviewing the device response.

### 7.93 PBIST Check of Primary or Module SRAM

The Hercules family architecture supports the use of a hardware programmable memory BIST (PBIST) engine. This logic is used to provide a very high diagnostic coverage on the implemented SRAMs at a transistor level. This logic utilizes the same design for test (DFT) logic and algorithms used by TI to provide rapid execution of high quality manufacturing tests. This technique has proven to be drastically more effective than software-based tests of SRAM, particularly for the devices with complex CPUs whose addressing modes do not enable an optimal software-based test.

The PBIST tests must be triggered by the software. User can elect to run all algorithms, or only a subset of the algorithms based on the execution time that can be allocated to the PBIST diagnostic. Similarly, the user can elect to run the PBIST on one SRAM or on groups of SRAMs based on the execution time, which can be allocated to the PBIST diagnostic. The PBIST tests are destructive to memory contents, and as such are typically run only at MCU initialization. However, the user has the freedom to initiate the tests at any time when the CPU is operable.

Execution of the PBIST results in a much higher level of transistor switching per clock cycle than during normal software execution due to the high efficiency of the test. A software control is implemented in the PBIST that allows the user to reduce the SRAM clock for the duration of the test. This feature allows the user to make a compromise between fast execution with higher current consumption or slower execution with reduced current consumption.

The most effective SRAM tests known to TI require test across a full physical memory module and are destructive to previous memory contents. If PBIST is to be implemented during operation, it is recommended to copy the data from the SRAM to be tested to a non-tested memory before test execution and to restore the data once the test is complete. When test execution is complete, the SRAM can be utilized for normal operation. The remainder of device logic continues normal operation during SRAM test. Any fault detected by the PBIST results in an error indicated in PBIST status registers. It is also possible to log faults in the PBIST logic.

PBIST logic includes capabilities for testing proper operation of the diagnostic. As the test time for the diagnostic is deterministic, a timeout counter can be implemented via software programming of RTI that can detect a failure to complete the test within expected time. In addition, there is the possibility to test the contents of the ROM that contains the PBIST algorithms via running a checksum ROM test using the PBIST engine. Error forcing can be accomplished by executing a test not intended for a targeted memory, such as executing an SRAM read and write test on the PBIST ROM.

### 7.94 PBIST Auto-coverage

The PBIST diagnostic is based on a 32-bit signature capture. For a given test, only one code is valid out of  $2^{32}$  possibilities. Therefore, if there is a fault in the PBIST logic, it is extremely unlikely that the correct passing code will be generated via the fault.

### 7.95 PBIST Test of Parity Bit Memory

When a Programmable Memory Bist (PBIST) test is run on a selected memory, the corresponding parity bit memory will be tested as well. This provides latent coverage for the parity memory as part of the parity diagnostic.

### 7.96 Redundant Address Decode Self Test

The redundant address decode diagnostic includes hardware to support self test. It is possible to force a pass and a fail to help diagnose proper operation of the redundant address decode diagnostic.

### 7.97 Shadow Registers

The use of a two stage cold and warm reset scheme on the device allows the implementation of shadow registers. Shadow registers are reset only by power-on reset. These registers can be used to store device status or other critical information that remain persistent after a warm reset changes the system state. This module includes shadow registers which can be used by software to provide additional information on the state of the device before the last warm reset operation.

### **7.98 Software Check of Cause of Last Reset**

The system control module provides a status register (SYSESR) that latches the cause of the most recent reset event. A boot software that checks the status of this register to determine the cause of the last reset event is typically implemented by software developers. This information can be used by the software to manage failure recovery.

### **7.99 Software Read Back of CPU Registers**

In order to ensure proper configuration of CPU coprocessor control registers, it is highly recommended that software implement a test to confirm proper operation of all control register writes. The CPU control registers are not memory mapped and must be accessed via the CPU coprocessor read and write commands.

### **7.100 Software Read Back of Written Configuration**

In order to ensure proper configuration of memory-mapped registers in this system control module, it is highly recommended to software implement a test to confirm proper operation of all control register writes. To support this software test, it is highly recommended to configure the register memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4 memory protection unit. This ensures that the register write by CPU will complete before additional CPU instructions are executed.

### **7.101 Software Test of DCC Functionality**

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

### **7.102 Software Test of DWD Functionality**

A basic test of DWD operation can be performed via software including checking of error response by programming expected "pet" thresholds and servicing or not servicing the "pet" requests.

### **7.103 Software Test of DWWD Functionality**

A basic test of DWWD operation can be performed via software including checking of error response by programming expected "pet" windows and servicing or not servicing the "pet" requests.

### **7.104 Software Test of ECC Profiler**

It is possible to test the functionality of the ECC profiler by performing reads on locations with ECC errors, and confirming proper operation of the error counters and threshold interrupt

### **7.105 Software Test of Error Path Reporting**

A software test can be utilized to inject diagnostic errors and check for proper reporting. Such a test can be executed at boot or periodically. Necessary software requirements are defined by the software implemented by the system integrator.

### **7.106 Software Test of Flash Sector Protection Logic**

By attempting to write to a protected flash sector, the operation of flash sector protection logic can be tested

### **7.107 Software Test of Function Including Error Tests**

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



### **7.108 Software Test of Function Using I/O Loopback**

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

Most modules support digital and analog loopback capabilities for the I/Os. Refer to the device technical reference manual to confirm the implemented loopback capabilities of the module. Digital loopback tests the signal path to the module boundary. Analog loopback tests the signal path from the module to the I/O cell with output driver enabled. For best results any tests of the functionality should include the I/O loopback.

### **7.109 Software Test of Function Using I/O Checking In GIO Mode**

A software test can be utilized to test basic functionality of the I/O in GIO Mode to perform an analog loopback diagnostic on an I/O pin. Such a test can be executed at boot or periodically. Software requirements necessary are defined by the software implemented by the system integrator.

The GIO buffer architecture supports an analog loopback between the input and output buffers of a given I/O structure. The content of the data in and data out registers will directly reflect the state of the pin whether the I/O direction is configured as an input or an output. This configuration allows the software test to detect inconsistencies between known input states or known output states with the pin state.

### **7.110 Software Test of Function Using I/O Loopback in Transceiver / PHY**

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

Most PHYs include a loopback mode that allows testing of the signal path including the PHY. Error response, diagnostic testability, and any necessary software requirements are defined by the system integrator. For best results any tests of the networking functionality should include I/O loopback.

### **7.111 Software Test of Function Using I/O Loopback - IOMM Only**

Analog loopback testing from peripherals results in signals traversing the I/O pin mux logic to the I/O pad and can provide diagnostic coverage on the I/O pin mux. Analog loopback tests the signal path from the module to the I/O cell with the output driver enabled.

I/O loopback is not enabled at reset. Software is necessary to configure and execute the diagnostic. Error response is managed by the software.

### **7.112 Software Test of Hardware CRC**

It is possible to test the functionality of hardware CRC by checking for correct calculation of expected CRC values as well as intentionally forcing a mismatch between the data and expected CRC signature, and seeing if the CRC Logic reports an error

### **7.113 Software Test of MPU Functionality**

It is possible to test the functionality of the non-CPU bus master MPUs by checking for correct memory access to a programmed region as well as by marking a memory region as invalid and then attempting to write to it in order to generate an error. If the MPU features discrimination by read/write then the testing should also consider check for correct operation and errors related to read/write permissions.

### **7.114 Software Test of Parity Logic**

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

### 7.115 Software Test of PBIST

It is possible to configure the PBIST logic by selecting an algorithm that should fail, and seeing if the PBIST Logic reports an error under this condition. For example, a read-write test could be performed on a read-only memory to ensure that a failure is reported. An alternative scheme is to run a test for memory with more bits than are actually present on the device memory.

### 7.116 Software Test of SRAM Wrapper Address Decode Diagnostic and ECC

A software based test of the redundant address decode diagnostic and ECC error reporting logic in the SRAM wrapper is can be performed at boot time as a latent diagnostic.

### 7.117 Software Test for Reset

A software test for detecting basic functionality as well as errors for reset sources and reset logic can be implemented.

### 7.118 Software Warm Reset Generation

The system control module provides the ability to the software to generate an internal warm reset (nRST). This is accomplished by writing appropriate control bits in the SYSECR control register. Software can utilize this feature to attempt failure recovery.

### 7.119 Transmission Redundancy

The information is transferred several times in sequence and compared. If the same data path is used for the duplicate transmissions, then transmission redundancy will only be useful for detecting transient faults. Error response, diagnostic testability, and any necessary software requirements are defined by the system integrator.

### 7.120 Use of CoreSight Debug Logic Key Enable Scheme

To enable operation of the memory-mapped CoreSight debug components, it is necessary to write a defined 32-bit key to an unlock register in each debug module. This debug lock protection provides an additional safety mechanism to limit undesired activation.

### 7.121 Use of DCC as Program Sequence Watchdog

The design of the N2HET is such that the instruction memory is contiguously executed in a circular loop with deterministic duration. A clocked output of known frequency can be generated on the N2HET with minimum software overhead. The N2HET has a dedicated channel internally connected to the DCC module. Comparison of the N2HET clocking output to a known clock reference by DCC can effectively implement a program sequence watchdog on the N2HET. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

### 7.122 Use of MPUs to Block Access to Memory Mapped Debug

The CoreSight debug peripherals are accessible over a memory-mapped debug bus. Access to this region can be blocked via the use of bus master based memory protection. For more information on memory protection, see [CPU Memory Protection Unit \(MPU\)](#).

## 8 Next Steps in Your Safety Development

TI's support for your safety development does not stop with the delivery of this safety document. Customers have a wide range of support options, such as:

- Access safety collateral for Hercules and other SafeTI™ products at: <http://www.ti.com/safeti>
- Access Hercules documentation including safety docs and app notes any time on the web: <http://www.ti.com/hercules>
- Discuss questions and concerns with TI experts and other Hercules developers using the TI Connected Community (E2E forums): <http://www.ti.com/hercules-support>



- Discuss questions and concerns regarding TI's safety documents that are provided under NDA with TI safety experts at: [https://e2eprivate.ti.com/safeti\\_functional\\_safety\\_support/default.aspx](https://e2eprivate.ti.com/safeti_functional_safety_support/default.aspx). By requesting a copy of this document, access will be granted to the support forum as well here: <http://www.ti.com/safetyanalysis>
- The Hercules Wiki page provides answers to many commonly asked questions: <http://www.ti.com/hercules-wiki>
- Attend a training class delivered by TI's Hercules experts: <http://www.ti.com/herculestraining>

Feedback and concerns about the safety manual are always welcome and can be submitted via the web by clicking on the feedback link at the bottom of each page of this document.

## Summary of Recommended Safety Feature Usage

Table 4 provides a summary of the safety features and diagnostics present in hardware or recommend for implementation in software or at system level.

**Table 3. Key to Summary of Safety Features and Diagnostics**

Device Partition	This field provides the element (device partition) which has safety feature and diagnostic recommendations
Unique Identifier	This field provides a short unique identifier to aid in traceability of each safety feature and diagnostic recommendation
Safety Feature or Diagnostic	This field provides the name of the safety feature or diagnostic and a link to more information.
Possible Tests for Diagnostics	This column lists mechanisms that can be applied to verify that the diagnostic itself is functioning correctly.
Usage	Each test listed in this chart can be one of two types. A "diagnostic" test, or a "test-for-diagnostic only" test Diagnostic: Provides coverage for faults on a primary function of the device. It may, in addition, provide fault coverage on other diagnostics, and can therefore be also used as a test-for-diagnostic in certain cases Test-for-Diagnostic Only: Does NOT provide coverage for faults on a primary function of the device. It's only purpose is to provide fault coverage on other diagnostics
Diagnostic Type	Hardware: A diagnostic or test-for-diagnostic which is implemented by TI in silicon. It may require software activation Software: A diagnostic or test-for-diagnostic, recommended by TI, which is performed in software and which must be created by the software implementor Software-Hardware: A diagnostic or test-for-diagnostic, recommended by TI, which requires both hardware in the device and software created by the software implementor System: A diagnostic or test-for-diagnostic, recommended by TI, which must be implemented at the system level by the system implementor
Diagnostic Operation	Continuous or Periodic/On-Demand
Text Execution Time	This column lists the time that is required for this diagnostic to complete
Action on Detected Fault	The response that 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 software-driven tests, this time is often software implementation-dependent

**Table 4. Summary of Safety Features and Diagnostics**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Power Supply	PWR1	<a href="#">Internal Voltage Monitor</a>	PWR2: <a href="#">External Voltage Supervisor</a> CLK5C: <a href="#">External Watchdog</a>	Diagnostic	Hardware	Continuous	1 $\mu$ s (Glitch filter logic filters transient glitches of <1 $\mu$ s)	Assert component power-on reset until fault condition not detection system control module Reset Request Register will log the cause of reset	Reset action <1 $\mu$ s after detection Software indication of reset cause available on first CPU active clock
	PWR2	<a href="#">External Voltage Supervisor</a>	PWR1: <a href="#">Internal Voltage Monitor</a> CLK5C: <a href="#">External Watchdog</a>	Diagnostic	System	System Defined (Typically Continuous)	System Defined	System Defined (Typically the powered device is disabled via reset or power cut)	System Defined

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Clock	CLK1	Low Power Oscillator Clock Detector	CLK5C: External Watchdog CLK4: External Monitoring via ECLK	Diagnostic	Hardware	Continuous	No Hardware Overhead	Error event provided to ESM Optional MCU soft (warm) reset generation Optional clock source switch to LPO	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	CLK2	PLL Slip Detector	CLK5C: External Watchdog CLK4: External Monitoring via ECLK	Diagnostic	Hardware	Continuous	No Hardware Overhead	Error Event Provided to ESM Optional MCU Soft (warm) Reset Generation Optional Clock Source Switch to LPO	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	CLK3	Dual Clock Comparator	CLK6: Periodic Software Readback of Static Configuration Registers CLK5C: External Watchdog CLK4: External Monitoring via ECLK	Diagnostic	Software-Hardware	Continuous	Software Dependent	Error Event Provided to ESM	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	CLK4	External Monitoring via ECLK	System Dependent	Diagnostic	System	Continuous	No Hardware Overhead	System Defined	System Defined
	CLK5A	Internal Watchdog - DWD	Software Test of DWD Operation External Watchdog External Monitoring via ECLK	Diagnostic	Software-Hardware	Continuous	No Hardware Overhead - Software Dependent	Internal MCU Soft (warm) Reset	Reset reason available in SYS module on first clock after reset
	CLK5B	Internal Watchdog - DWWD	CLK9: Software Test of DWD Operation CLK5C: External Watchdog CLK4: External Monitoring via ECLK	Diagnostic	Software-Hardware	Continuous	No Hardware Overhead - Software Dependent	Internal MCU Soft (warm) Reset or Non-Maskable Interrupt	Reset reason available in SYS module on first clock after reset Typically <1 μS to notify CPU of NMI* *(Interrupt handling time is system load and software dependent)
	CLK5C	External Watchdog	System Dependent	Diagnostic	Software-Hardware	Continuous	System Dependent	System Defined	System Defined
	CLK6	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU Lockstep Comparator Flash Data ECC Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	CLK7	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	CLK8	Software Test of DCC Operation	NA - This is a test of the DCC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	CLK9	Software Test of DWD Operation	NA - This is a test of the DWD diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
CLK10	Software Test of DWWD Operation	NA - This is a test of the DWWD diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent	

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Reset	RST1	<a href="#">External Monitoring of Warm Reset</a>	System Dependent	Diagnostic	System	System Defined (Typically Continuous)	System Defined	System Defined (Typically the powered device is disabled via reset or power cut)	System Defined
	RST2	<a href="#">Software Check of Cause of Last Reset</a>	N/A	Diagnostic	Software - Hardware	Periodic / On-Demand	Software Dependent	N/A (Fault Avoidance)	N/A (Fault Avoidance)
	RST3	<a href="#">Software Warm Reset Generation</a>	N/A	Diagnostic	Software - Hardware	Periodic / On-Demand	Software Dependent	N/A (Fault Avoidance)	N/A (Fault Avoidance)
	RST4	<a href="#">Glitch Filtering on Reset Pins</a>	N/A	Diagnostic	Hardware	Continuous	1 $\mu$ s (Glitch filter logic filters transient glitches of <1 $\mu$ s)	N/A (Fault Avoidance)	N/A (Fault Avoidance)
	RST5	<a href="#">Use of Status Shadow Registers</a>	N/A	Diagnostic	Software - Hardware	Periodic / On-Demand	Software Dependent	N/A (Fault Avoidance)	N/A (Fault Avoidance)
	RST6	<a href="#">Periodic Software Readback of Static Configuration Registers</a>	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	RST7	<a href="#">Software Readback of Written Configuration</a>	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	RST8	<a href="#">Software Test for Reset</a>	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software-Dependent Hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
System Control Module	SYS1	<a href="#">Privileged Mode Access and Multi-Bit Enable Keys for Control Registers</a>	N/A	Diagnostic	Hardware	Continuous	No Hardware Overhead	Control Register not Updated	N/A (Fault Avoidance)
	SYS2	<a href="#">Software Readback of Written Configuration</a>	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	SYS3	<a href="#">Periodic Software Readback of Static Configuration Registers</a>	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Error Signaling Module (ESM)	ESM1	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	ESM2A	Boot Time Software Test of Error Path Reporting	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software-Dependent Hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	ESM2B	Periodic Software Test of Error Path Reporting	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software-Dependent Hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	ESM3	Use of Status Shadow Registers	N/A	Diagnostic	Hardware	Continuous	N/A - Fault Avoidance	N/A - Fault Avoidance	N/A - Fault Avoidance
	ESM4	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Cortex-R4 Central Processing Unit (CPU)	CPU1	CPU Lockstep Comparator	Lockstep Comparator (CCM) Self Test	Diagnostic	Hardware	Continuous	2 CPU Cycles	ESM Error	Typically <1 $\mu$ s (implementation dependent)
	CPU2A	Boot Time Execution of LBIST STC	CPU1: CPU Lockstep Comparator, CLK4: External Watchdog, CPU9: LBIST Auto-Coverage	Diagnostic	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Error Flag Asserted	Typically <1 $\mu$ s (implementation dependent)
	CPU2B	Periodic Execution of LBIST STC	CPU1: CPU Lockstep Comparator, CLK4: External Watchdog, CPU9: LBIST Auto-Coverage	Diagnostic	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Error Flag Asserted	Typically <1 $\mu$ s (implementation dependent)
	CPU3	Memory Protection Unit (MPU)	CPU1: CPU Lockstep Comparator, CPU2A: Boot Time Execution of LBIST STC, CPU2B: Periodic Execution of LBIST STC	Diagnostic	Hardware	Continuous	No Hardware Overhead	Instruction Abort/Data Abort	1 CPU Cycle After Detection
	CPU4	Online Profiling Using PMU	CPU1: CPU Lockstep Comparator, CPU2A: Boot Time Execution of LBIST STC, CPU2B: Periodic Execution of LBIST STC	Diagnostic	Software-Hardware	Continuous	Software Dependent	Software Defined	Software-Dependent
	CPU5	CPU Illegal Operation and Instruction Trapping	CPU1: CPU Lockstep Comparator, CPU2A: Boot Time Execution of LBIST STC, CPU2B: Periodic Execution of LBIST STC	Diagnostic	Hardware	Continuous	No Hardware Overhead	Instruction Abort/Data Abort	1 CPU Cycle After Detection
	CPU6	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	CPU7	Software Read Back of CPU Registers	CPU1: CPU Lockstep Comparator, CPU2A: Boot Time Execution of LBIST STC, CPU2B: Periodic Execution of LBIST STC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	CPU8	Lockstep Comparator (CCM) Self Test	NA - This is a test of the CCM diagnostic, It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Error Flag Asserted	Typically <1 $\mu$ s (implementation dependent)
CPU9	LBIST Auto-Coverage	NA - This is a test of the LBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in LBIST operation	N/A	N/A	



**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Primary Flash and Level 1 (L1) Interconnect	FLA1	Flash Data ECC	CPU1: CPU Lockstep Comparator FLA10: Flash Wrapper Diag Mode 5 Test FLA11: Flash Wrapper Diag Mode 7	Diagnostic	Hardware	Continuous	No Hardware Overhead	CPU Abort / ESM Event	Same CPU clock cycles as data used
	FLA2	Hard Error Cache and Livelock	CPU Lockstep Comparator	Diagnostic	Hardware	Continuous	No Hardware Overhead	Live lock / ESM Event	Typically <1 µs
	FLA3	Flash Wrapper Address ECC	FLA10: Flash Wrapper Diag Mode 5 Test FLA11: Flash Wrapper Diag Mode 7	Diagnostic	Hardware	Continuous	No Hardware Overhead	CPU Abort / ESM Event	Same CPU clock cycles as data used
	FLA4	Address and Control Parity	Software Readback of Written Configuration Software Test of Parity Logic	Diagnostic	Hardware	Continuous	No Hardware Overhead	CPU Bus Error	Same CPU clock cycles as data used
	FLA5A	Boot Time Check of Flash Memory Contents by Hardware CRC	External Watchdog Software Readback of Written Configuration Software Test of Hardware CRC CRC Auto-Coverage	Diagnostic	Software - Hardware	On Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Defined	Software Dependent
	FLA5B	Periodic Check of Flash Memory Contents by Hardware CRC	External Watchdog Software Readback of Written Configuration Software Test of Hardware CRC CRC Auto-Coverage	Diagnostic	Software - Hardware	On Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Defined	Software Dependent
	FLA6	Bit Multiplexing in Flash Memory Array	N/A - Fault Avoidance	Diagnostic	Hardware	Continuous	N/A - Fault Avoidance	N/A - Fault Avoidance	N/A - Fault Avoidance
	FLA7	Flash Sector Protection	Software Test of Flash Sector Protection Logic	Diagnostic	Hardware	Continuous	No Hardware Overhead	Flash Programming Blocked	Software dependent based on readback of content
	FLA8	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	FLA9	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	FLA10	Flash Wrapper Diag Mode 5 Test	NA - this is a test of a diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software - Hardware	On Demand	Software Dependent Typically <1 µs overhead due to hardware	Software Dependent	Software Dependent
FLA11	Flash Wrapper Diag Mode 7	NA - this is a test of a diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software - Hardware	On Demand	Software Dependent Typically <1 µs overhead due to hardware	Software Dependent	Software Dependent	

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Primary Flash and Level 1 (L1) Interconnect	FLA12	<a href="#">Software Test of Parity Logic</a>	NA - This is a test of the parity diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	On Demand	Software Dependent	Software Dependent	Software Dependent
	FLA13	<a href="#">Software Test of Flash Sector Protection Logic</a>	NA - This is a test of the flash sector protection logic diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	On Demand	Software Dependent	Software Dependent	Software Dependent
	FLA14	<a href="#">Software Test of Hardware CRC</a>	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software - Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent	Software Dependent	Software Dependent
	FLA15	<a href="#">CRC Auto-Coverage</a>	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in CRC operation	N/A	N/A

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Flash EEPROM Emulation (FEE)	FEE1	FEE Data ECC	CPU Lockstep Comparator Flash Wrapper Diag Mode 1 Test Flash Wrapper Diag Mode 2 Test Flash Wrapper Diag Mode 3 Test Flash Wrapper Diag Mode 4 Test	Diagnostic	Hardware	Continuous	No Overhead (Same clock cycle)	ESM Event	Same CPU clock cycles as data used
	FEE2A	Boot Time Check of FEE Memory Contents by Hardware CRC	Software Readback of Written Configuration Software Test of Hardware CRC CRC Auto-Coverage External Watchdog	Diagnostic	Software - Hardware	On Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Defined	Software Dependent
	FEE2B	Periodic Check of FEE Memory Contents by Hardware CRC	Software Readback of Written Configuration Software Test of Hardware CRC CRC Auto-Coverage External Watchdog	Diagnostic	Software - Hardware	On Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Defined	Software Dependent
	FEE3	Bit Multiplexing in FEE Array	N/A - Fault Avoidance	Diagnostic	Hardware	Continuous	N/A - Fault Avoidance	N/A - Fault Avoidance	N/A - Fault Avoidance
	FEE4	FEE Sector Protection	Periodic Check of FEE Memory Contents by Hardware CRC Software Test of Flash Sector Protection Logic	Diagnostic	Hardware	Continuous	No Overhead (Same clock cycle)	Flash Programming Blocked	Software dependent based on readback of content
	FEE5	Periodic Software Readback of Static Configuration Registers	Internal Watchdog - DWD Internal Watchdog - DWWD External Watchdog CPU Lockstep Comparator Flash Data ECC Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	FEE6	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	FEE7	Flash Wrapper Address ECC	Periodic Check of FEE Memory Contents by Hardware CRC Flash Wrapper Diag Mode 1 Test Flash Wrapper Diag Mode 2 Test Flash Wrapper Diag Mode 3 Test Flash Wrapper Diag Mode 4 Test	Diagnostic	Hardware	Continuous	No Hardware Overhead	ESM Event	Same CPU clock cycles as data used

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Flash Emulated EEPROM (FEE)	FEE8	<a href="#">Flash Wrapper Diag Mode 1 Test</a>	NA - this is a test of a diagnostic. It is not a diagnostic itself	Test for Diagnostic	Software - Hardware	On Demand	Software Dependent Typically <1 $\mu$ s overhead due to hardware	Software Dependent	Software Dependent
	FEE9	<a href="#">Flash Wrapper Diag Mode 2 Test</a>	NA - this is a test of a diagnostic. It is not a diagnostic itself	Test for Diagnostic	Software - Hardware	On Demand	Software Dependent Typically <1 $\mu$ s overhead due to hardware	Software Dependent	Software Dependent
	FEE10	<a href="#">Flash Wrapper Diag Mode 3 Test</a>	NA - this is a test of a diagnostic. It is not a diagnostic itself	Test for Diagnostic	Software - Hardware	On Demand	Software Dependent Typically <1 $\mu$ s overhead due to hardware	Software Dependent	Software Dependent
	FEE11	<a href="#">Flash Wrapper Diag Mode 4 Test</a>	NA - this is a test of a diagnostic. It is not a diagnostic itself	Test for Diagnostic	Software - Hardware	On Demand	Software Dependent Typically <1 $\mu$ s overhead due to hardware	Software Dependent	Software Dependent
	FEE12	<a href="#">Software Test of Flash Sector Protection Logic</a>	NA -This is a test of the flash sector protection logic diagnostic. It is not a diagnostic itself	Test for Diagnostic	Software	On Demand	Software-Dependent	Software Dependent	Software Dependent
	FEE13	<a href="#">Software Test of Hardware CRC</a>	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic	Software - Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	FEE14	<a href="#">CRC Auto-Coverage</a>	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in CRC operation	N/A	N/A

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
SRAM and Level 1 (L1) Interconnect	RAM1	Data ECC	CPU Lockstep Comparator	Diagnostic	Hardware	Continuous	No Hardware Overhead	CPU Abort / ESM Event	Same CPU clock cycles as data used
	RAM2	Hard Error Cache and Livelock	CPU Lockstep Comparator	Diagnostic	Hardware	Continuous	No Hardware Overhead	Live lock / ESM Event	Typically <1us
	RAM3	Correctable ECC Profiling	Software Test of ECC Profiler	Diagnostic	Software / Hardware	Continuous	Software Dependent	Software Dependent	Software Dependent
	RAM4	Address and Control Bus Parity	Software Readback of Written Configuration Software Test of Parity Logic	Diagnostic	Hardware	Continuous	No Hardware Overhead	CPU Bus Error	Same CPU clock cycles as data used
	RAM5	Redundant Address Decode	Redundant Address Decode Self Test	Diagnostic	Hardware	Continuous	No Hardware Overhead	CPU Bus Error	Same CPU clock cycles as data used
	RAM6	Data and ECC Storage in Multiple Physical Banks	N/A - Fault Avoidance	Diagnostic	Hardware	Continuous	N/A (Fault Avoidance)	N/A (Fault Avoidance)	N/A (Fault Avoidance)
	RAM7A	Boot Time PBIST Check of Primary SRAM	Data ECC Software Readback of Written Configuration Software Test of PBIST PBIST Auto-Coverage CPU Lockstep Comparator External Watchdog Flash Data ECC	Diagnostic	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	RAM7B	Periodic PBIST Check of Primary SRAM	Data ECC Software Readback of Written Configuration Software Test of PBIST PBIST Auto-Coverage CPU Lockstep Comparator External Watchdog Flash Data ECC	Diagnostic	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	RAM8	Bit Multiplexing in Primary SRAM Array	N/A - Fault Avoidance	Diagnostic	Hardware	Continuous	N/A (Fault Avoidance)	N/A (Fault Avoidance)	N/A (Fault Avoidance)
	RAM9	Periodic Hardware CRC Check of Primary SRAM Contents	Software Readback of Written Configuration, Software Test of Hardware CRC, CRC Auto-Coverage, External Watchdog	Diagnostic	Software / Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	RAM10	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
RAM11	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent	

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
SRAM and Level 1 (L1) Interconnect	RAM12	<a href="#">Software Test of SRAM Wrapper Redundant Address Decode and ECC</a>	N/A - This is a test of Redundant Address Decode and ECC. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	RAM13	<a href="#">Software Test of Parity Logic</a>	N/A - This is a test of the parity logic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	RAM14	<a href="#">Software Test of PBIST</a>	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	RAM15	<a href="#">PBIST Auto-Coverage</a>	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in PBIST operation	N/A	N/A
	RAM16	<a href="#">Software Test of ECC Profiler</a>	NA - This is a test of the ECC profiler diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	RAM17	<a href="#">Redundant Address Decode Self Test</a>	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	No Overhead (Same clock cycle)	Software Dependent	Software Dependent
	RAM18	<a href="#">Software Test of Hardware CRC</a>	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software / Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	RAM19	<a href="#">CRC Auto-Coverage</a>	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in CRC operation	N/A	N/A

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Level 2 and Level 3 (L2 and L3) Interconnect	INC1	Error Trapping (Including Peripheral Slave Error Trapping)	Boot Time Software Test of Basic Functionality Including Error Tests Periodic Software Test of Basic Functionality Including Error Tests	Diagnostic	Hardware	Continuous	No Hardware Overhead	Bus Error	Bus master notification typically <10 clock cycles after fault detection
	INC2	PCR Access Management	Boot Time Software Test of Basic Functionality Including Error Tests Periodic Software Test of Basic Functionality Including Error Tests	Diagnostic	Hardware	Continuous	No Hardware Overhead	Bus Error	Bus master notification typically <10 clock cycles after fault detection
	INC3	Information Redundancy	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	INC4	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	INC5A	Boot Time Software Test of Basic Functionality Including Error Tests	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software-Dependent Hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	INC5B	Periodic Software Test of Basic Functionality Including Error Tests	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software-Dependent Hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	INC6	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	INC7	Transmission Redundancy	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	System Dependent	System Dependent	System Dependent



**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
EFuse Static Configuration	EFU1	<a href="#">Boot Time Autoload Self Test</a>	<a href="#">Autoload Self Test Auto-Coverage</a>	Diagnostic	Hardware	Continuous (Every power-on reset operation)	Device Dependent (see the device-specific data sheet)	ESM Event	No Overhead (Available on 1st CPU clock cycle after release of reset)
	EFU2	<a href="#">E-Fuse ECC</a>	<a href="#">EFuse ECC Logic Self Test</a>	Diagnostic	Hardware	Continuous	No Hardware Overhead	ESM Event	Same CPU clock cycle as data is used
	EFU3	<a href="#">Periodic Software Readback of Static Configuration Registers</a>	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	EFU4	<a href="#">Software Readback of Written Configuration</a>	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	EFU5	<a href="#">Autoload Self Test Auto-Coverage</a>	NA - This is a test of the Efuse autoload diagnostic. It is not a diagnostic itself	Test- for-Diagnostic	Hardware	At Boot Time	Device Dependent (see the device-specific data sheet)	ESM Event	No Overhead (Available on 1st CPU clock cycle after release of reset)
	EFU6	<a href="#">EFuse ECC Logic Self Test</a>	NA - This is a test of the Efuse ECC logic through self test diagnostic. It is not a diagnostic itself	Test-for-Diagnostic	Software - Hardware	On Demand	No Hardware Overhead	ESM Event	No Overhead

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
One Time Programmable (OTP) Flash Static Configuration	OTP1	OTP ECC	CPU Lockstep Comparator, Flash Wrapper Diag Mode 1 Test, Flash Wrapper Diag Mode 2 Test, Flash Wrapper Diag Mode 3 Test, Flash Wrapper Diag Mode 4 Test	Diagnostic	Hardware	Continuous	No Hardware Overhead	ESM Event	Same CPU clock cycle as data used
	OTP2	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	OTP3	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	OTP4A	Boot Time Hardware CRC Check of OTP Contents	Software Readback of Written Configuration, Software Test of Hardware CRC, CRC Auto-Coverage, External Watchdog	Diagnostic	Software - Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	OTP4B	Periodic Hardware CRC Check of OTP Contents	Software Readback of Written Configuration, Software Test of Hardware CRC, CRC Auto-Coverage, External Watchdog	Diagnostic	Software - Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	OTP5	Bit Multiplexing in Flash Memory Array	N/A - Fault Avoidance	Diagnostic	Hardware	Continuous	N/A - Fault Avoidance	N/A - Fault Avoidance	N/A - Fault Avoidance
	OTP6	Flash Sector Protection	Software Test of Flash Sector Protection Logic	Diagnostic	Hardware	Continuous	No Hardware Overhead	Flash Programming Blocked	Software Dependent
	OTP7	Flash Wrapper Diag Mode 1 Test	N/A - This is a test of a diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software - Hardware	Periodic / On-Demand	Software Dependent Typically <1 μs overhead due to hardware	Software Dependent	Software Dependent
	OTP8	Flash Wrapper Diag Mode 2 Test	N/A - This is a test of a diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software - Hardware	Periodic / On-Demand	Software Dependent Typically <1 μs overhead due to hardware	Software Dependent	Software Dependent
OTP9	Flash Wrapper Diag Mode 3 Test	N/A - This is a test of a diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software - Hardware	Periodic / On-Demand	Software Dependent Typically <1 μs overhead due to hardware	Software Dependent	Software Dependent	

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
One Time Programmable (OTP) Flash Static Configuration	OTP10	Flash Wrapper Diag Mode 4 Test	N/A - This is a test of a diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software - Hardware	Periodic / On-Demand	Software Dependent Typically <1 $\mu$ s overhead due to hardware	Software Dependent	Software Dependent
	OTP11	Software Test of Flash Sector Protection Logic	NA - This is a test of the flash sector protection logic diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	OTP12	Software Test of Hardware CRC	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software - Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	OTP13	CRC Auto-Coverage	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in CRC operation	N/A	N/A
Input/Output (I/O) Multiplexing Module (IOMM)	IOM1	Locking Mechanism For Control Registers	N/A	Diagnostic	Hardware (Fault Avoidance)	Continuous	No Hardware Overhead	N/A (Fault Avoidance)	N/A (Fault Avoidance)
	IOM2	Master ID Filtering	N/A	Diagnostic	Hardware (Fault Avoidance)	Periodic / On-Demand	No Hardware Overhead	N/A (Fault Avoidance)	N/A (Fault Avoidance)
	IOM3	Error Trapping	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Hardware	Periodic / On-Demand	No Hardware Overhead	Bus Error	Bus master notification typically <10 clock cycles after fault detection
	IOM4	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	IOM5A	Boot Time Software Test of Function Using I/O Loopback	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	IOM5B	Periodic Software Test of Function Using I/O Loopback	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	IOM6	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Vectored Interrupt Module (VIM)	VIM1	VIM SRAM Data Parity	VIM7: Software Readback of Written Configuration VIM8: Software test of Parity Logic VIM9: PBIST Test of Parity Bit Memory	Diagnostic	Hardware	Continuous	No Overhead (Same clock cycle)	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	VIM2A	Boot Time PBIST Check of VIM SRAM	VIM7: Software Readback of Written Configuration VIM12: Software Test of PBIST VIM13: PBIST Auto-Coverage CPU1: CPU Lockstep Comparator CLK5C: External Watchdog FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	VIM2B	Periodic PBIST Check of VIM SRAM	VIM7: Software Readback of Written Configuration VIM12: Software Test of PBIST VIM13: PBIST Auto-Coverage CPU1: CPU Lockstep Comparator CLK5C: External Watchdog FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	VIM3	Bit Multiplexing in VIM SRAM Array	N/A	Diagnostic	Hardware	Continuous	N/A (Fault Avoidance)	N/A (Fault Avoidance)	N/A (Fault Avoidance)
	VIM4	Periodic Hardware CRC Check of VIM SRAM Contents	VIM7: Software Readback of Written Configuration VIM10: Software Test of Hardware CRC VIM11: CRC Auto-Coverage	Diagnostic	Software - Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	VIM5A	Boot Time Software Test of VIM Functionality Including Error Tests	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software-Dependent; hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	VIM5B	Periodic Software Test of VIM Functionality Including Error Tests	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software-Dependent; hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	VIM6	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	VIM7	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Vectored Interrupt Module (VIM)	VIM8	<a href="#">Software test of Parity Logic</a>	NA - This is a test of the parity logic diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	VIM9	<a href="#">PBIST Test of Parity Bit Memory</a>	NA - This is a test of the parity bit memory diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	VIM10	<a href="#">Software Test of Hardware CRC</a>	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software - Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	VIM11	<a href="#">CRC Auto-Coverage</a>	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in CRC operation	N/A	N/A
	VIM12	<a href="#">Software Test of PBIST</a>	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	VIM13	<a href="#">PBIST Auto-Coverage</a>	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in PBIST operation	N/A	N/A
Real Time Interrupt (RTI) Operating System Timer	RTI1	<a href="#">1002 Software Voting Using Secondary Free Running Counter</a>	<a href="#">CLK5A: Internal Watchdog - DWD</a> <a href="#">CLK5B: Internal Watchdog - DWWD</a> <a href="#">CLK5C: External Watchdog</a> <a href="#">CPU1: CPU Lockstep Comparator</a> <a href="#">FLA1: Flash Data ECC</a> <a href="#">RAM1: RAM Data ECC</a>	Diagnostic	Software - Hardware	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	RTI2	<a href="#">Periodic Software Readback of Static Configuration Registers</a>	<a href="#">CLK5A: Internal Watchdog - DWD</a> <a href="#">CLK5B: Internal Watchdog - DWWD</a> <a href="#">CLK5C: External Watchdog</a> <a href="#">CPU1: CPU Lockstep Comparator</a> <a href="#">FLA1: Flash Data ECC</a> <a href="#">RAM1: RAM Data ECC</a>	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	RTI3	<a href="#">Software Readback of Written Configuration</a>	<a href="#">CLK5A: Internal Watchdog - DWD</a> <a href="#">CLK5B: Internal Watchdog - DWWD</a> <a href="#">CLK5C: External Watchdog</a> <a href="#">CPU1: CPU Lockstep Comparator</a> <a href="#">FLA1: Flash Data ECC</a> <a href="#">RAM1: RAM Data ECC</a>	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Multi-Buffered Analog to Digital Converter (MibADC)	ADC1	Boot Time Input Self Test	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Two conversions per channel under test required Typical sample time is 1 µs/conversion but dependent on device configuration Software overhead must also be considered	Software Dependent	Software Dependent
	ADC2A	Boot Time MibADC Converter Calibration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Four conversions per channel under test required Typical sample time is 1 µs/conversion but dependent on device configuration Software overhead must also be considered	Software Dependent	Software Dependent
	ADC2B	Periodic MibADC Converter Calibration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Four conversions per channel under test required Typical sample time is 1 µs/conversion but dependent on device configuration Software overhead must also be considered	Software Dependent	Software Dependent
	ADC3	MibADC Information Redundancy Techniques	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software-Dependent	Software Dependent	Software Dependent
	ADC4	MibADC SRAM Data Parity	ADC9: Software Readback of Written Configuration ADC11: PBIST Test of Parity Bit Memory	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 µS to notify CPU* *(Interrupt handling time is system load and software dependent)
	ADC5A	Boot Time PBIST Check of MibADC SRAM	ADC9: Software Readback of Written Configuration ADC14: Software Test of PBIST ADC15: PBIST Auto-Coverage CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	ADC5B	Periodic PBIST Check of MibADC SRAM	ADC9: Software Readback of Written Configuration ADC14: Software Test of PBIST ADC15: PBIST Auto-Coverage CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Multi-Buffered Analog-to-Digital Converter (MibADC)	ADC6	Bit Multiplexing in MibADC SRAM Array	N/A	Diagnostic	Hardware	Continuous	N/A (Fault Avoidance)	N/A (Fault Avoidance)	N/A (Fault Avoidance)
	ADC7	Periodic Hardware CRC Check of MibADC SRAM Contents	ADC9: Software Readback of Written Configuration ADC12: Software Test of Hardware CRC ADC13: CRC Auto-Coverage	Diagnostic	Software - Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	ADC8	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	ADC9	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software-Dependent	Software Dependent	Software Dependent
	ADC10	Software test of Parity Logic	NA - This is a test of the parity logic diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	ADC11	PBIST Test of Parity Bit Memory	NA - This is a test of the parity bit memory diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	ADC12	Software Test of Hardware CRC	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software - Hardware	Periodic / On-Demand	Depends on Size of Memory Under Test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	ADC13	CRC Auto-Coverage	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in CRC operation	N/A	N/A
	ADC14	Software Test of PBIST	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	ADC15	PBIST Auto-Coverage	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in PBIST operation	N/A	N/A



**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
High-End Timer (N2HET) Including HET Transfer Unit (HTU)	HET1	Memory Protection Unit for HTU Bus Master Accesses	HET11: Software Readback of Written Configuration HET19: Software Test of MPU Functionality	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	HET2	Information Redundancy Techniques	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWW CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	HET3	Use of DCC as Program Sequence Watchdog	HET11: Software Readback of Written Configuration HET20: Software Test of DCC Operation	Diagnostic	Software - Hardware	Continuous	Software Dependent	Software Dependent	Software Dependent
	HET5A	Boot Time Software Test of Function Using I/O Loopback	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWW CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software-Dependent Hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	HET5B	Periodic Software Test of Function Using I/O Loopback	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWW CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software-Dependent Hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	HET6	N2HET/HTU SRAM Data Parity	HET11: Software Readback of Written Configuration HET13: Software test of Parity Logic HET14: PBIST Test of Parity Bit Memory	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	HET7A	Boot Time PBIST Check of N2HET/HTU SRAM	HET11: Software Readback of Written Configuration HET17: Software Test of PBIST HET18: PBIST Auto-Coverage	Diagnostic	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	HET7B	Periodic PBIST Check of N2HET/HTU SRAM	HET11: Software Readback of Written Configuration HET17: Software Test of PBIST HET18: PBIST Auto-Coverage	Diagnostic	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	HET8	Bit Multiplexing in N2HET/HTU SRAM Array	N/A - Fault Avoidance	Diagnostic	Hardware	Continuous	N/A (Fault Avoidance)	N/A (Fault Avoidance)	N/A (Fault Avoidance)
	HET9	Periodic Hardware CRC Check of N2HET/HTU SRAM Contents	HET11: Software Readback of Written Configuration HET15: Software Test of Hardware CRC HET16: CRC Auto-Coverage	Diagnostic	Software - Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
High-End Timer (N2HET) Including HET Transfer Unit (HTU)	HET10	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software-Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	HET11	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	HET12	Transmission Redundancy (for Transfer Unit)	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	System Dependent	System Dependent	System Dependent
	HET13	Software test of Parity Logic	NA - This is a test of the parity logic diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	HET14	PBIST Test of Parity Bit Memory	NA - This is a test of the parity bit memory diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	HET15	Software Test of Hardware CRC	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software - Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Defined	Software Dependent
	HET16	CRC Auto-Coverage	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in CRC operation	N/A	N/A
	HET17	Software Test of PBIST	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	HET18	PBIST Auto-Coverage	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in PBIST operation	N/A	N/A
	HET19	Software Test of MPU Functionality	NA - This is a test of the MPU functionality diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
HET20	Software Test of DCC Functionality	NA - This is a test of the DCC functionality diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent	

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Enhanced Quadrature Encoder Pulse (eQEP)	QEP1A	Boot Time Software Test of Function Including Error Tests	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software Dependent Hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	QEP1B	Periodic Software Test of Function Including Error Tests	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software Dependent Hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	QEP2	Information Redundancy Techniques	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	QEP3	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	QEP4	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	QEP5	Quadrature Watchdog	QEP6: Software Test of Quadrature Watchdog Functionality	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	QEP6	Software Test of Quadrature Watchdog Functionality	NA - Test of quadrature watchdog functionality diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Multi-Buffered Serial Peripheral Interface (MibSPI)	MSP1A	Boot Time Software Test of Function Using I/O Loopback	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software-Dependent Hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	MSP1B	Periodic Software Test of Function Using I/O Loopback	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software / Hardware	Periodic / On-Demand	Software-Dependent Hardware contribution to test execution time is negligible	Software Defined	Software-Dependent
	MSP2	Parity in Message	MSP9: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 $\mu$ S to notify CPU* *(Interrupt handling time is system load and software dependent)
	MSP3	Information Redundancy Techniques	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	MSP4	MibSPI SRAM Data Parity	MSP9: Software Readback of Written Configuration MSP16: Software test of Parity Logic MSP17: PBIST Test of Parity Bit Memory	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 $\mu$ S to notify CPU* *(Interrupt handling time is system load and software dependent)
	MSP5A	Boot Time PBIST Check of MibSPI SRAM	MSP9: Software Readback of Written Configuration MSP20: Software Test of PBIST MSP21: PBIST Auto-Coverage CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	MSP5B	Periodic PBIST Check of MibSPI SRAM	MSP9: Software Readback of Written Configuration MSP20: Software Test of PBIST MSP21: PBIST Auto-Coverage CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
MSP6	Bit Multiplexing in MibSPI SRAM Array	N/A	Diagnostic	Hardware	Continuous	N/A (Fault Avoidance)	N/A (Fault Avoidance)	N/A (Fault Avoidance)	

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Multi-Buffered Serial Peripheral Interface (MibSPI)	MSP7	Periodic Hardware CRC Check of MibSPI SRAM Contents	MSP9: Software Readback of Written Configuration MSP18: Software Test of Hardware CRC MSP19: CRC Auto-Coverage	Diagnostic	Software - Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	MSP8	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	MSP9	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	MSP10	Transmission Redundancy	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	System Dependent	System Defined	System Dependent
	MSP11	Data Overrun Error Detection	MSP9: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	MSP12	Bit Error Detection	MSP9: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	MSP13	Slave Desync Detection	MSP9: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	MSP14	Slave Timeout Detection	MSP9: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	MSP15	Data Length Error Detection	MSP9: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	MSP16	Software test of Parity Logic	NA - This is a test of the parity logic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Multi-Buffered Serial Peripheral Interface (MibSPI)	MSP17	<a href="#">PBIST Test of Parity Bit Memory</a>	NA - This is a test of the parity bit memory. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	MSP18	<a href="#">Software Test of Hardware CRC</a>	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software - Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	MSP19	<a href="#">CRC Auto-Coverage</a>	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in CRC operation	N/A	N/A
	MSP20	<a href="#">Software Test of PBIST</a>	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	MSP21	<a href="#">PBIST Auto-Coverage</a>	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; implicit in PBIST operation	N/A	N/A

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Serial Peripheral Interface (SPI)	SPI1A	Boot Time Software Test of Function Using I/O Loopback	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	SPI1B	Periodic Software Test of Function Using I/O Loopback	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	SPI2	Parity in Message	SPI5: Software Readback of Written Configuration	Diagnostic	Hardware	Periodic / On-Demand	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	SPI3	Information Redundancy Techniques	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	SPI4	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	SPI5	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	SPI6	Transmission Redundancy	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	System Dependent	System Dependent	System Dependent
	SPI7	Data Overrun Error Detection	SPI5: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	SPI8	Bit Error Detection	SPI5: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	SPI9	Slave Desync Detection	SPI5: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)



**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Serial Peripheral Interface (SPI)	SPI10	Slave Timeout Detection	SPI5: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 $\mu$ S to notify CPU* *(Interrupt handling time is system load and software dependent)
	SPI11	Data Length Error Detection	SPI5: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 $\mu$ S to notify CPU* *(Interrupt handling time is system load and software dependent)

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Local Interconnect Network (LIN)	LIN1A	Boot Time Software Test of Function Using I/O Loopback	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	On Demand	Software-Dependent	Software Defined	Software-Dependent
	LIN1B	Periodic Software Test of Function Using I/O Loopback	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	On Demand	Software-Dependent	Software Defined	Software-Dependent
	LIN2	Information Redundancy Techniques	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	On Demand	Software-Dependent	Software Defined	Software-Dependent
	LIN3	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	On Demand	Software-Dependent. Typically <25 CPU clocks per 32-bit Register Read	Software Defined	Software-Dependent
	LIN4	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	On Demand	Software Dependent	Software Defined	Software-Dependent
	LIN5	Transmission Redundancy	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	On Demand	System Dependent	System Defined	System Dependent
	LIN6	Overrun Error Detection	LIN4: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Overhead (Same clock cycle)	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	LIN7	Frame Error Detection	LIN4: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Overhead (Same clock cycle)	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	LIN8	Physical Bus Error Detection	LIN4: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Overhead (Same clock cycle)	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
LIN9	No-Response Error Detection	LIN4: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Overhead (Same clock cycle)	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)	

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Local Interconnect Network (LIN)	LIN10	Bit Error Detection	LIN4: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Overhead (Same clock cycle)	Interrupt Provided to CPU	Typically <1 $\mu$ S to notify CPU* *(Interrupt handling time is system load and software dependent)
	LIN11	Checksum Error Detection	LIN4: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Overhead (Same clock cycle)	Interrupt Provided to CPU	Typically <1 $\mu$ S to notify CPU* *(Interrupt handling time is system load and software dependent)
	LIN12	Parity in Message	LIN4: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Overhead (Same clock cycle)	Interrupt Provided to CPU	Typically <1 $\mu$ S to notify CPU* *(Interrupt handling time is system load and software dependent)

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Controller Area Network (DCAN)	CAN1A	Boot Time Software Test of Function Using I/O Loopback	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software-Hardware	On Demand	Software Dependent	Software Dependent	Software Dependent
	CAN1B	Periodic Software Test of Function Using I/O Loopback	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software-Hardware	On Demand	Software Dependent	Software Dependent	Software Dependent
	CAN2	Information Redundancy Techniques	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	On Demand	Software Dependent	Software Dependent	Software Dependent
	CAN3	DCAN SRAM Data Parity	CAN8: Software Readback of Written Configuration CAN15: Software test of Parity Logic CAN16: PBIST Test of Parity Bit Memory	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 μS to notify CPU* *(Interrupt handling time is system load and software dependent)
	CAN4A	Boot Time PBIST Check of DCAN SRAM	CAN8: Software Readback of Written Configuration CAN19: Software Test of PBIST CAN20: PBIST Auto-Coverage CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Hardware	On Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	CAN4B	Periodic PBIST Check of DCAN SRAM	CAN8: Software Readback of Written Configuration CAN19: Software Test of PBIST CAN20: PBIST Auto-Coverage CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Hardware	On Demand	Device Dependent (see the device-specific data sheet)	Flag Set in Error Register	Software Dependent
	CAN5	Bit Multiplexing in DCAN SRAM Array	N/A - Fault Avoidance	Diagnostic	Hardware	Continuous	N/A - Fault Avoidance	N/A - Fault Avoidance	N/A - Fault Avoidance
	CAN6	Periodic Hardware CRC Check of DCAN SRAM Contents	CAN8: Software Readback of Written Configuration CAN17: Software Test of Hardware CRC CAN18: CRC Auto-Coverage	Diagnostic	Software-Hardware	On Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
	CAN7	Periodic Software Readback of Static Configuration Registers	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Controller Area Network (DCAN)	CAN8	Software Readback of Written Configuration	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	CAN9	Transmission Redundancy	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	On Demand	System Dependent	System Dependent	System Dependent
	CAN10	Stuff Error Detection	CAN8: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 $\mu$ S to notify CPU* *(Interrupt handling time is system load and software dependent)
	CAN11	Form Error Detection	CAN8: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 $\mu$ S to notify CPU* *(Interrupt handling time is system load and software dependent)
	CAN12	Acknowledge Error Detection	CAN8: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 $\mu$ S to notify CPU* *(Interrupt handling time is system load and software dependent)
	CAN13	Bit Error Detection	CAN8: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 $\mu$ S to notify CPU* *(Interrupt handling time is system load and software dependent)
	CAN14	CAN Protocol CRC in Message	CAN8: Software Readback of Written Configuration	Diagnostic	Hardware	Continuous	No Hardware Overhead	Interrupt Provided to CPU	Typically <1 $\mu$ S to notify CPU* *(Interrupt handling time is system load and software dependent)
	CAN15	Software test of Parity Logic	NA - This is a test of the parity logic diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	CAN16	PBIST Test of Parity Bit Memory	NA - This is a test of the parity bit memory diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	Device Dependent (see the device-specific data sheet)	Flag set in Error Register	Software Dependent
	CAN17	Software Test of Hardware CRC	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software-Hardware	Periodic / On-Demand	Depends on size of memory under test. One 64-bit word per clock cycle plus overhead. Overhead is software dependent.	Software Dependent	Software Dependent
CAN18	CRC Auto-Coverage	NA - This is a test of the hardware CRC diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; this is an implicit part of the CRC diagnostic	N/A	N/A	

**Table 4. Summary of Safety Features and Diagnostics (continued)**

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Possible Tests for Diagnostics	Usage	Diagnostic Type	Diagnostic Operation	Text Execution Time	Action on Detected Fault	Error Reporting Time
Controller Area Network (DCAN)	CAN19	<a href="#">Software Test of PBIST</a>	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	CAN20	<a href="#">PBIST Auto-Coverage</a>	NA - This is a test of the PBIST diagnostic. It is not a diagnostic itself	Test-for-Diagnostic Only	Hardware	Periodic / On-Demand	N/A; this is an implicit part of the PBIST diagnostic	N/A	N/A
General Purpose Input/Output (GIO)	GIO1A	<a href="#">Boot Time Software Test of Function Using I/O Checking In GIO Mode</a>	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software Dependent Hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	GIO1B	<a href="#">Periodic Software Test of Function Using I/O Checking In GIO Mode</a>	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software - Hardware	Periodic / On-Demand	Software Dependent Hardware contribution to test execution time is negligible	Software Dependent	Software Dependent
	GIO2	<a href="#">Information Redundancy Techniques</a>	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent	Software Dependent	Software Dependent
	GIO3	<a href="#">Periodic Software Readback of Static Configuration Registers</a>	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
	GIO4	<a href="#">Software Readback of Written Configuration</a>	CLK5A: Internal Watchdog - DWD CLK5B: Internal Watchdog - DWWD CLK5C: External Watchdog CPU1: CPU Lockstep Comparator FLA1: Flash Data ECC RAM1: RAM Data ECC	Diagnostic	Software	Periodic / On-Demand	Software Dependent Typically <25 CPU clocks per 32-bit Register Read	Software Dependent	Software Dependent
Joint Technical Action Group (JTAG) Debug	JTG1	<a href="#">Hardware Disable of JTAG Port</a>	N/A	Diagnostic	Hardware	Continuous	No Hardware Overhead	N/A (Fault Avoidance)	N/A (Fault Avoidance)
	JTG2	<a href="#">Lockout of JTAG Access Using AJSM</a>	N/A	Diagnostic	Hardware	Continuous	No Hardware Overhead	JTAG Access Blocked	N/A (Fault Avoidance)
Cortex-R4 Central Processing Unit (CPU) Debug	DBG1	<a href="#">Use of MPUs to Block Access to Memory-Mapped Debug</a>	N/A	Diagnostic	Hardware	Continuous	No Hardware Overhead	Data abort	Same CPU cycle as fault detected
	DBG2	<a href="#">Use of CoreSight Debug Logic Key Enable Scheme</a>	N/A	Diagnostic	Hardware	Continuous	No Hardware Overhead	Debug Access Locked	N/A (Fault Avoidance)

## ***Development Interface Agreement***

---

---

---

A Development Interface Agreement (DIA) is intended to capture an agreement between a customer and supplier towards the management of shared responsibilities in developing a functional safety system. In custom developments, the DIA is a key document executed between customer and supplier early in the development process. As the Hercules family is a commercial, off the shelf (COTS) product, TI has prepared a standard DIA within this section that describes the support that TI can provide for customer developments. Requests for custom DIAs should be referred to your local TI sales office for disposition.

### ***B.1 Appointment of Safety Managers***

Texas Instruments has developed the Hercules MCUs with one or more development specialist safety managers in place throughout the silicon design, release to market, and release to production. Safety management after release to production is maintained by separate safety managers who specialize in production and operation issues. Safety management responsibilities are continued through product end-of-life.

### ***B.2 Tailoring of the Safety Lifecycle***

TI has tailored the safety lifecycles of IEC 61508 and ISO 26262 to best match the needs of a safety element out of context (SEooC). The tailoring activity has been executed in conjunction with input from exida and Yogitech to ensure application of state of the art techniques and measures.

Key elements of the tailored safety lifecycle are:

- Assumptions on system level design, safety concept, and requirements
- Combined qualitative and quantitative or similar safety analysis techniques comprehending the sum of silicon failure modes and diagnostic techniques known to both TI and Yogitech
- Fault estimation based on multiple industry standards as well as TI "real-world" reliability data
- Application of Yogitech's state-of-the-art fault injection methodology for validation of claimed diagnostic coverage
- Integration of lessons learned by through multiple safety critical developments to IEC 61508 and participation in the ISO 26262 international working group



Figure 10 illustrates these activities overlaid atop TI's standard QM development flow for microcontrollers.

Phase 0 Business Opportunity Prescreen	Phase 1 Program Planning	Phase 2 Create	Phase 2.5 Validate, Sample, and Characterize	Phase 3 Qualify	Phase 4 Ramp or Sustain
Determine if safety process execution is necessary	Define SIL/ASIL capability	Execute safety design	Validate safety design in silicon	Qualification of safety design	Implement plans to support operation and production
Execute development interface agreement (DIA) with lead customers and suppliers	Generate safety plan	Qualitative analysis of design (FMEA and FTA)	Release safety manual	Release safety case report	Update safety case report (if needed)
	Initiate safety case	Incorporate findings into safety design	Release safety analysis report	Update safety manual (if needed)	Periodic confirmation measure reviews
	Analyze system to generate system level safety assumptions and requirements	Develop safety product preview	Characterization of safety design	Update safety analysis report (if needed)	
	Develop component level safety requirements	Validation of safety design at RTL level	Confirmation measure review	Confirmation measure review	
	Validate component safety requirements meet system safety requirements	Quantitative analysis of design (FMEDA)			
	Implement safety requirements in design specification	Incorporate findings into safety design			
	Validate design specification meets component safety requirements	Validation of safety design at gate/layout level			
	Confirmation measure review	Confirmation measure review			

**Figure 10. Hercules Tailoring of Safety Lifecycle**

### **B.3 Activities Performed by TI**

The TI microcontroller products covered by this DIA are hardware components developed as safety elements out of context. As such TI's safety activities focus on those related to management of functional safety and hardware component development. System level architecture, design, and safety analysis are not in scope of TI activities and are the responsibility of the TI customer.

**Table 5. Activities Performed by TI vs. Performed by SEooC Customer**

<b>Safety Lifecycle Activity</b>	<b>TI Execution</b>	<b>SEooC Customer Execution</b>
Management of functional safety	Yes	Yes
Definition of end equipment and item	No	Yes
Hazard and risk analysis of end equipment/item	No	Yes
Development of end equipment safety concept	Assumptions made to support internal development. Contact TI if support is needed	Yes
Allocation of end equipment requirements to sub-systems, hardware components, and software components	Assumptions made to support internal development. Contact TI if support is needed	Yes
Definition of MCU safety requirements	Yes	No
MCU architecture and design execution	Yes	Possible for diagnostics implemented by SEooC customer. Contact TI if support is needed
MCU level safety analysis	Yes	Possible for diagnostics implemented by SEooC customer. Contact TI if support is needed
MCU level verification and validation	V&V executed to support internal development. Contact TI if support is needed.	Possible for diagnostics implemented by SEooC customer. Contact TI if support is needed
Integration of MCU into end equipment	Support provided	Yes
Verification of MCU performance in end equipment	Support provided	Yes
Selection of Diagnostics to be Applied to MCU	Support provided	Yes
End equipment level safety analysis	No	Yes
End equipment level verification and validation	No	Yes
End equipment level safety assessment	Support provided	Yes
End equipment release to production	No	Yes
Management of safety issues in production	Support provided	Yes

### **B.4 Information to be Exchanged**

In a custom development, there is an expectation under IEC 61508 and ISO 26262 that all development documents related to work products are made available to the customer. In a COTS product, this approach is not sustainable. TI has summarized the most critical development items into a series of documents that can be made available to customers either publicly or under a non-disclosure agreement (NDA). NDAs are required in order to protect proprietary and sensitive information disclosed in certain safety documents.

**Table 6** summarizes the product safety documentation that TI can provide to customers to assist in development of safety systems.

**Table 6. Product Safety Documentation**

Deliverable Name	Contents	Confidentiality	Availability
Safety Product Preview	Overview of safety considerations in product development and product architecture. Delivered ahead of public product announcement.	NDA required	Removed from circulation after release to market due to availability of Safety Manual
Safety Manual	User guide for the safety features of the product	Public, no NDA required	Available
<i>Safety Analysis Report Summary for TMS570LS04x, TMS570LS03x, and TMS570LS02x ARM®-Based Safety Critical Microcontrollers (SPNU556)</i>	Summary of FIT rates and device safety metrics according to ISO 26262 and/or IEC 61508 at device level.	NDA required	Available
<i>Detailed Safety Analysis Report for TMS570LS04x, TMS570LS03x, and TMS570LS02x ARM®-Based Safety Critical Microcontrollers (SPNU559)</i>	Full results of all available safety analysis documented in a format that allows computation of custom metrics	NDA required	Available
Safety Report (for more information, contact your TI sales representative)	Summary of the conformance of the product to the ISO 26262 and/or IEC 61508 standards	NDA required	In development

**Table 7. Product Functional Documentation to be Considered in Safety-Related Design**

Deliverable Name	Contents	Confidentiality	Availability
Technical Reference Manual (TRM)	User Guide for the device	Public; no NDA required	Available
Datasheet	Device-specific features and operating constraints	Public; no NDA required	Available
Errata	Lists known discrepancies between device documentation and device functionality / performance	Public; no NDA required	Available

Users may subscribe to be notified of documentation changes for any Hercules device by performing the following steps:

- Visit [www.ti.com/hercules](http://www.ti.com/hercules)
- Select "Products"
- Choose the link for the desired device
- Click the "Product Alert" link

## **B.5 Parties Responsible for Safety Activities**

TI applies a cross functional approach to safety related development. Safety related activities are carried out by a variety of program managers, safety managers, applications engineers, design engineers, and other development and production engineering functions.

## **B.6 Communication of Target Values**

As the Hercules MCU product is developed as a safety element out of context, there is no system developer involved during the MCU design who has provided target metrics for the MCU development. At start of development TI makes assumptions of plausible MCU level safety target values and designs the products to meet such targets. The *Safety Analysis Report Summary for TMS570LS04x, TMS570LS03x, and TMS570LS02x ARM®-Based Safety Critical Microcontrollers (SPNU556)* and the *Detailed Safety Analysis Report for TMS570LS04x, TMS570LS03x, and TMS570LS02x ARM®-Based Safety Critical Microcontrollers (SPNU559)* can be utilized to evaluate achieved safety metrics. The ultimate responsibility to determine if the TI component is suitable for use in the system rests on the system integrator.

## B.7 Supporting Processes and Tools

TI uses a variety of tools and corresponding data formats for internal and external documents. The tools and data formats that are relevant to the safety related documents shared with SEooC customers are noted in [Table 8](#).

**Table 8. Product Safety Documentation Tools and Formats**

Deliverable Name	Creation Tool(s)	Output Format(s)
Safety Product Preview	Microsoft® Word	Adobe™ PDF
Safety Manual	XML	Adobe PDF
<i>Safety Analysis Report Summary for TMS570LS04x, TMS570LS03x, and TMS570LS02x ARM®-Based Safety Critical Microcontrollers (SPNU556)</i>	XML, Microsoft Excel®	Adobe PDF
<i>Detailed Safety Analysis Report for TMS570LS04x, TMS570LS03x, and TMS570LS02x ARM®-Based Safety Critical Microcontrollers (SPNU559)</i>	XML, Microsoft Excel	Adobe PDF, Microsoft Excel
Safety Case Report (for more information, contact your TI sales representative)	IBM® DOORS®, XML	Adobe PDF

## B.8 Supplier Hazard and Risk Assessment

Hazard and risk assessments under IEC 61508 and ISO 26262 are targeted at the system level of abstraction. When developing a hardware component out of context, the system implementation is not known. Therefore TI has not executed a system hazard and risk analysis. Instead, TI has made assumptions on the results of hazard and risk analysis that are fed into the component design. The ultimate responsibility to determine if the TI component is suitable for use in the system rests on the system integrator.

## B.9 Creation of Functional Safety Concept

The functional safety concept under IEC 61508 and ISO 26262 is targeted at the system level of abstraction. When developing a hardware component out of context, the system implementation is not known. Therefore TI cannot generate a system functional safety concept. Instead, TI has made assumptions on the output of a system functional safety concept and this data has been fed into the component design. The ultimate responsibility to determine if the TI component is suitable for use in the system rests on the system integrator.

## Revision History

This document has been revised from SPNU552A to SPNU552B because of the following technical change(s).

**Table 9. SPNU552B Revisions**

Location	Additions, Deletes, and Edits
Section 1	Added text to include silicon revision B and to indicate the part number list is for revision B
	Added TMS570LS0232 configuration to the part number list
	Added Tape and Reel part numbers for all configurations
Figure 1	Updated revision identification graphic to be revision independent.
Section 2	Updated title of the section to include TMS570LS02x device part number
Section 4.5	Added a statement to the introductory paragraph to clarify the role of the ESM as a non-functional element.
Section 6	Renamed "Boot Time Software Test of Function Including I/O Checking (in GIO Mode) to "Boot Time Software Test of Function Including I/O In GIO Mode" and updated links to new section "Software Test of Function Including I/O In GIO Mode" in Section 7.
	Renamed "Periodic Software Test of Function Including I/O Checking (in GIO Mode) to "Periodic Software Test of Function Including I/O In GIO Mode" and updated links to new section "Software Test of Function Including I/O In GIO Mode" in Section 7.
	Removed redundant cross references/hyperlinks "Hardware Disable of JTAG Port" in section titled "Multi-Buffered Analog-to-Digital Converter (MibADC)"
	Removed redundant cross references/hyperlinks "Periodic Software Test of Function Including I/O Checking (while in GIO Mode)" in section titled "Multi-Buffered Analog-to-Digital Converter (MibADC)"
	Several instances of interchangeable use of Operation in place of Functionality such as in Software Test of MPU Operation, Software Test of DWD Operation, Software Test of DWWD Operation, Periodic Test of VIM Operation Including Errors, etc. All instances of the use of "Operation" in the diagnostic titles and associated cross references/hyperlinks updated to use "Functionality" for consistency
	Corrected cross references/hyperlinks in section titled "Inter-Integrated Circuit (I2C)" labeled "Boot Time Software Test of Function" and "Periodic Software Test of Function" to point to paragraph in section 7 titled "Software Test of Function Using I/O Loopback"
	Updated all cross references to "CPU Lockstep Comparator Self Test" (with exception of the "CPU Subsystem" section) to reference "CPU Lockstep Compare"
	Updated all references/hyperlinks titled "CPU Lockstep Comparator" to "CPU Lockstep Compare" for consistency
	Replaced all instances of links to section 7 "Hardware Disable of JTAG Port to Limit Functionality of This Module" and replaced them with links to "Hardware Disable of JTAG Port" and updated associated label of the hyperlink with the revised title reference.
Section 7	Removed section titled "Hardware Disable of JTAG Port to Limit Functionality of This Module" as it indirectly referenced "Hardware Disable of JTAG Port" and decision was to directly reference this section.
	Updated title of "CPU Lockstep Comparator" to CPU Lockstep Compare" for consistency
	Updated description of section titled "Hardware CRC Check of OTP Contents" to remove references to boot time or cyclic operation.
	Renamed "Hardware CRC Check of SRAM Contents" diagnostic description to "Periodic Hardware CRC Check of SRAM Contents"
	Updated paragraph text in section "Hardware CRC Check of OTP Contents" back to original content to correct erroneous content in prior version

**Table 9. SPNU552B Revisions (continued)**

<b>Location</b>	<b>Additions, Deletes, and Edits</b>
Appendix A	Added Label "CRC In Message" to Cross reference/Hyperlink in column 2
	Misc. updates to column two hyperlink/cross references to support changes to Section 6 and Section 7
Throughout document	Added clarifications and updated formatting in various sections
	Updated document title and document references within the content to include TMS570LS02x

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

### Products

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>

### Applications

Automotive and Transportation	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Energy and Lighting	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Space, Avionics and Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>

### TI E2E Community

[e2e.ti.com](http://e2e.ti.com)