Safety Manual for TMS570LS31x and TMS570LS21x Hercules™ ARM® Safety Critical Microcontrollers

User's Guide

Literature Number: SPNU511B
April 2013

Contents

List of Figures

List of Tables

User's Guide
SPNU511B–April 2013

Safety Manual for TMS570LS31x and TMS570LS21x Hercules™ ARM® Safety Critical Microcontrollers

1 Introduction

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 implementations covered by this safety manual include:

- TMS570LS Safety Critical Microcontrollers
  - TMS570LS31x
  - TMS570LS21x

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, and recommended usage

The following information is documented in the *Safety Analysis Report Summary for TMS570LS31x and TMS570LS21x ARM Safety Critical Microcontrollers* (SPNU521) 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, etc.) safety metrics at the chip level

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 harmless TI from any and all damages, claims, suits, or expense resulting from such use.

The following information is documented in the *Detailed Safety Analysis Report for TMS570LS31x and TMS570LS21x ARM Safety Critical Microcontrollers* (SPNU523) 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:

- Summary of evidence showing compliance to targeted safety standards
- 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 more information regarding the *Safety Report* contact your TI sales representative or http://www.ti.com.

2 Hercules TMS570LS31x and TMS570LS21x 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 1. 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.

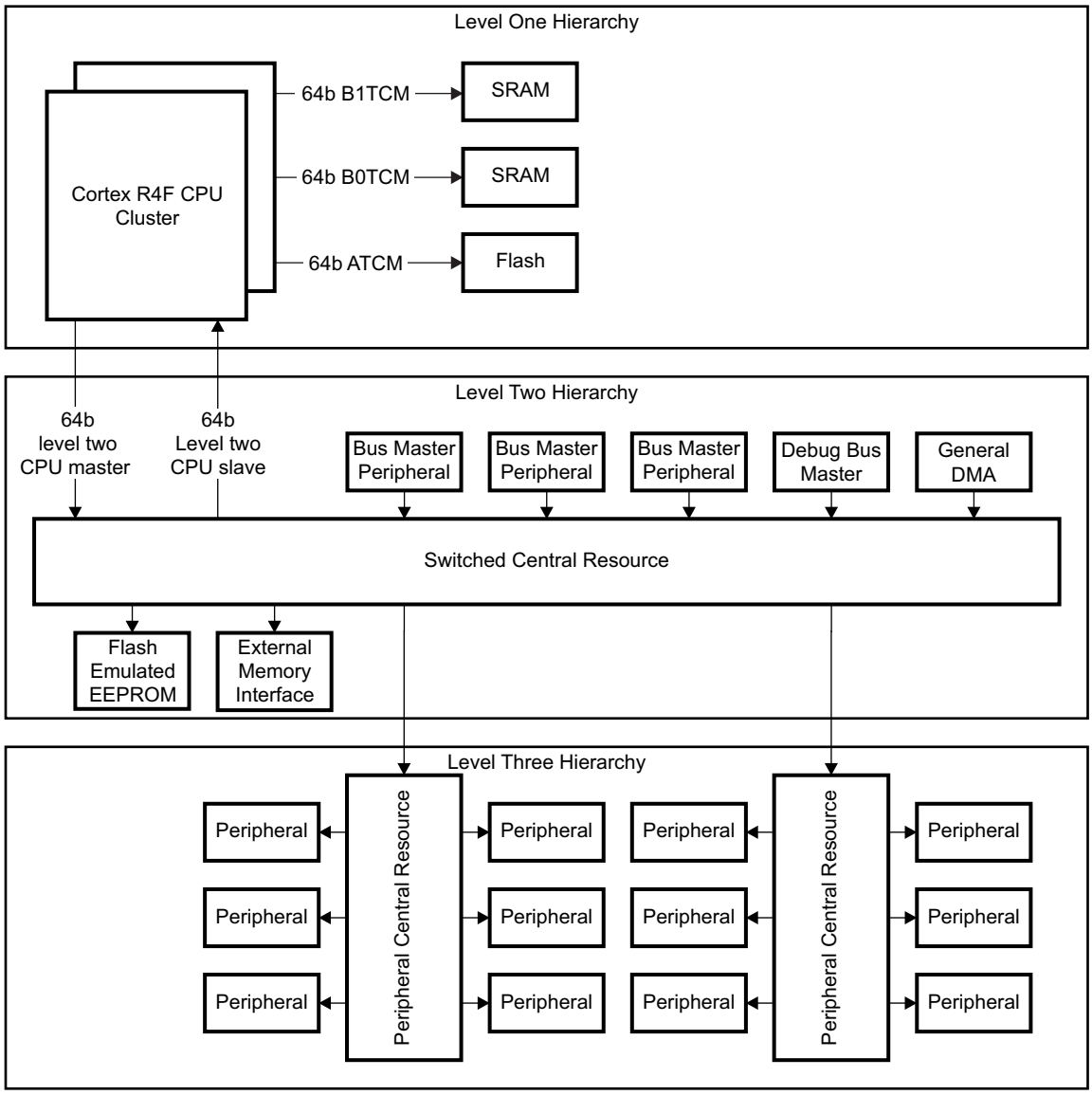


Figure 1. Hercules Product Architecture Overview

The Hercules product architecture utilizes the proven ARM Cortex™-R4F CPU in a tightly coupled memory configuration. The Cortex-R4F CPU is implemented with a checker Cortex-R4F 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 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, debug bus masters, and general purpose direct memory access (DMA) controllers. Bus slaves on the level two hierarchy include the Flash EEPROM emulation memory, external memory interface (EMIF), access to one or more peripheral bus segments, and a Cortex-R4F 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 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 must be executed by the system integrator.

2.2 Product Safety Constraints

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 TMS570LS31x and TMS570LS21x ARM Safety Critical Microcontrollers* (SPNU523) provides an example 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 26262:5-2011, 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 26262:10-2012, 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 26262:10-2012, 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 IEC 61508 2nd edition and ISO 26262. This process has gone through a process of continual improvement as ISO 26262 standards development continues. The process applied to the first 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). The TI standard MCU Automotive development process is certified compliant to ISO TS 16949 as assessed by Det Norske Veritas Certification, Inc. (Katy, Texas) under certificate CERT-07319CC10-2004-AQ-HOU-IATF (IATF certificate No 0113679). The development is also certified compliant to ISO 9001:2008 as assessed by DNV Certification B.V. (Netherlands) under certificate CERT-06185-2003-AQ-HOU-RvA Rev. 2.

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 2.

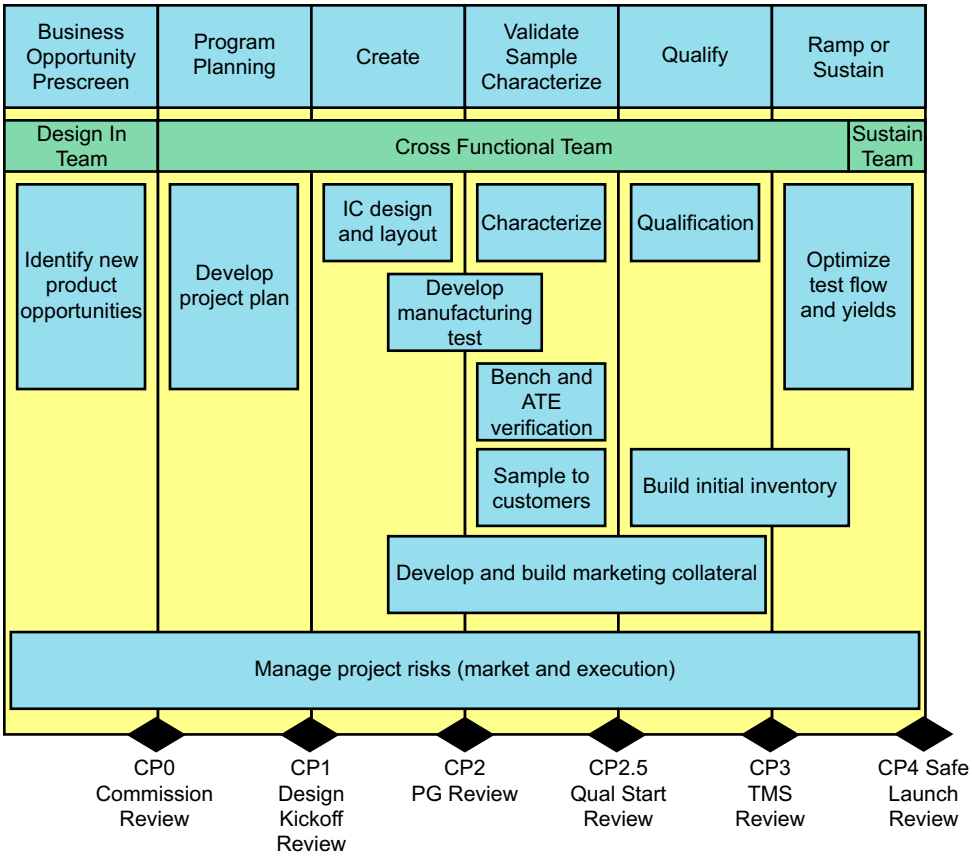


Figure 2. TI Standard MCU Automotive QM Development Process

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

fRMethodology is approved by TÜV SÜD as the flow to assess and validate the safe failure fraction of a given integrated circuit in adherence to IEC 61508 (certificate Z10 06 11 61674 001). It has been extended by YOGITECH to ISO 26262, taking profit of YOGITECH’s active role as member of the ISO-TC22-SC3-WG16 (ISO 26262) Italian and international working group. In the ISO 26262 international working group, YOGITECH is responsible for the Annex A of part 10, for example, how to deal with microcontrollers in the context of an ISO 26262 application.

Moreover, YOGITECH extended both IEC 61508 and ISO 26262 requirements to analogue circuits thanks to YOGITECH’s consolidated experience in analogue design and analogue verification. YOGITECH implemented a tool for verification of analogue circuits, the AMSvkit, that has been used by several companies world-wide (http://www.yogitech.com/#amsvkit).

YOGITECH’s fRMethodology is in line with ISO 26262-10:2012, 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 61508:2010 and ISO 26262:2011, 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.

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

Figure 3 illustrates these activities overlaid atop the standard QM development flow.

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 3. 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 for Safety Analysis

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

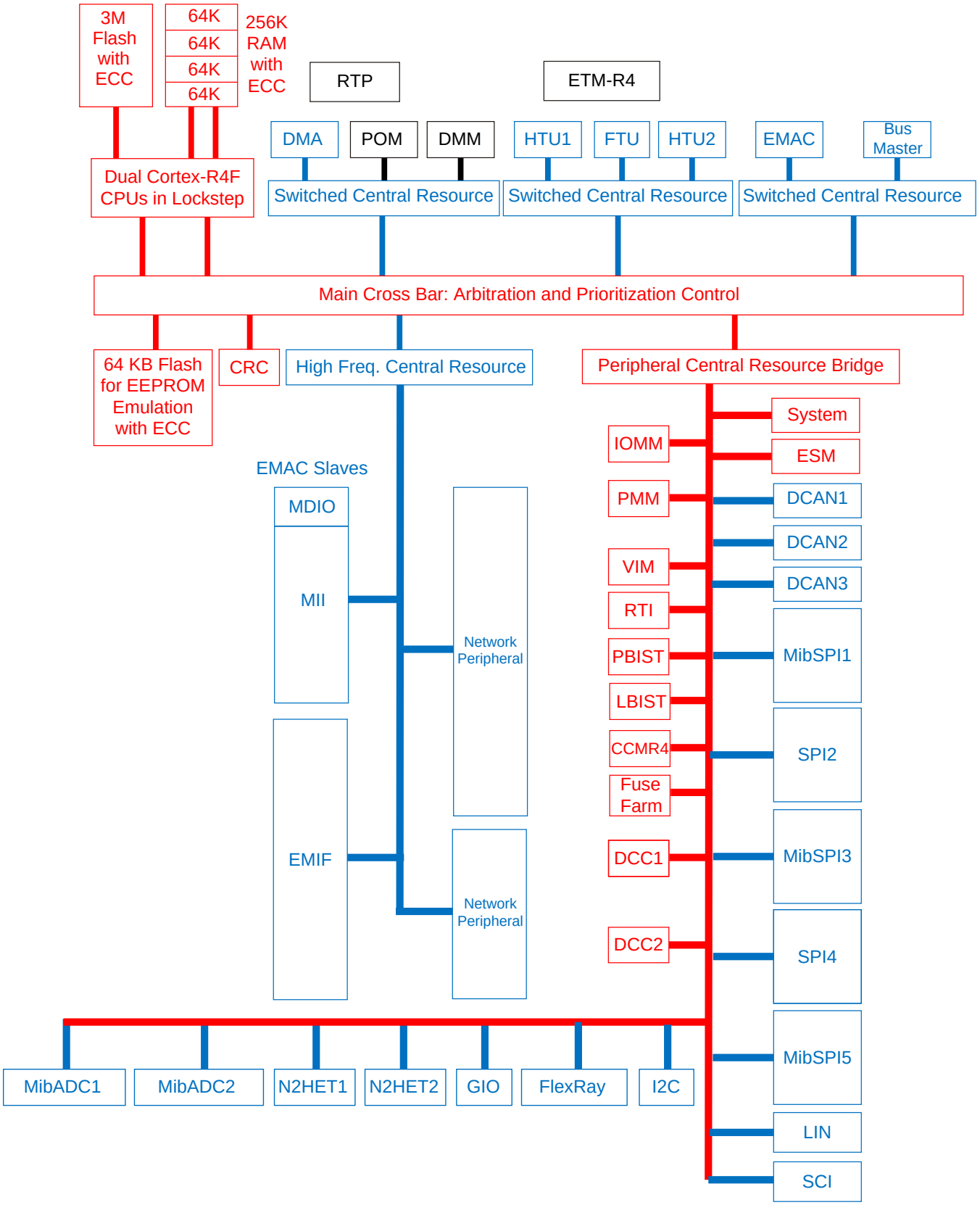


Figure 4. Partition of Hercules MCU for Safety Analysis

Figure 4 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 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.

4.3 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 5 and described below.

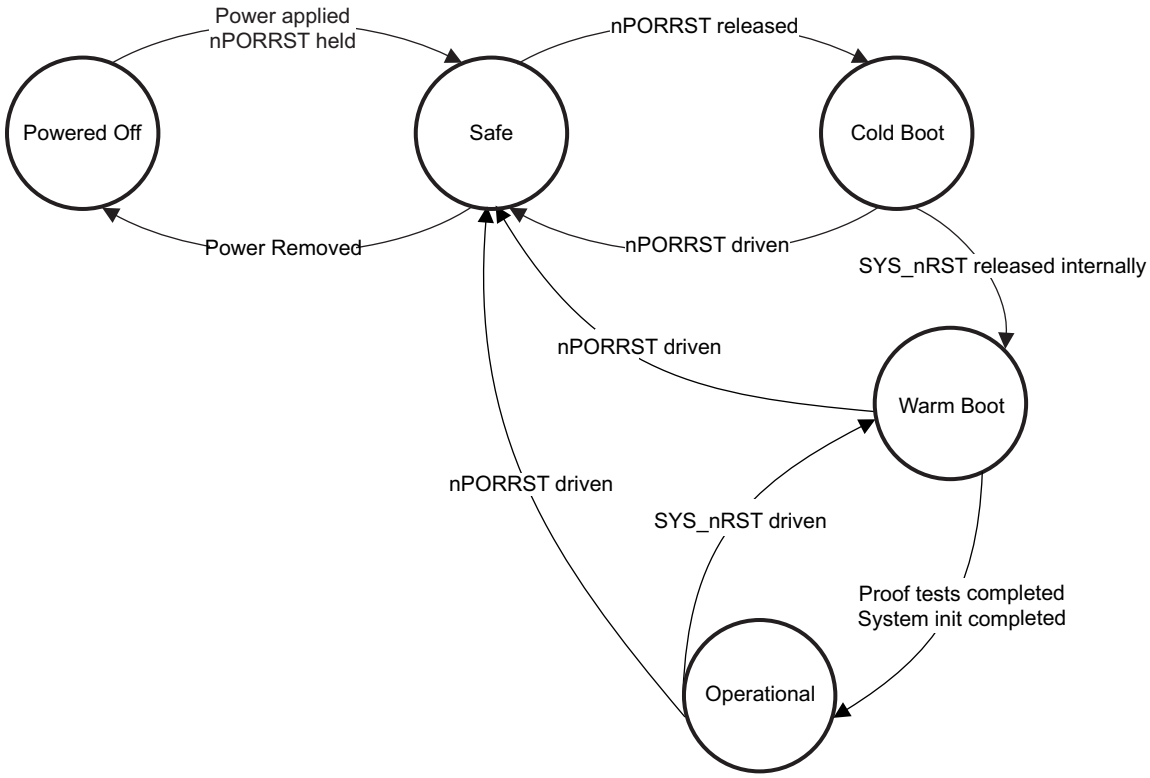


Figure 5. Operating States of the Hercules MCU

- "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.

4.4 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 error classifications in the ESM are summarized in Table 1.

Table 1. 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 Hercules Architecture Safety Mechanisms and Assumptions of Use

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.

In this section, the safety mechanisms for each major functional block of the Hercules architecture are summarized and general assumptions of use are provided. 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 for the MCU used. The effectiveness of the hardware safety mechanisms is noted in the *Detailed Safety Analysis Report for TMS570LS31x and TMS570LS21x ARM Safety Critical Microcontrollers* (SPNU523) .

TI classifies technical recommendations for the use of safety mechanisms in this section into a number of categories. The TI recommendations should not be considered infallible. There are many diverse ways to implement safe systems and alternate safety mechanisms may be possible that can provide support to achieve desired safety metrics. The categories of recommendation are as follows:

- Mandatory - A mandatory notation indicates a safety mechanism that is always operable during normal functional operation and cannot be disabled by user action.
- Highly Recommended - A highly recommended notation indicates a safety mechanism that TI believes to provide a high value of diagnostics that are difficult to implement by other means. The user retains the choice whether or not to utilize the safety mechanism in their design, as user action is either needed to enable the safety mechanism or user action can disable the safety mechanism.
- Recommended - A recommended notation indicates a safety mechanism that TI believes to provide a valuable diagnostic that can also be implemented by other means. The user retains the choice whether or not to utilize the safety mechanism in their design, as user action is either needed to enable the safety mechanism or user action can disable the safety mechanism.
- Optional - An optional notation indicates a safety mechanism that TI believe to provide a lower value diagnostic that can also be implemented by other means. The user retains the choice whether or not to utilize the safety mechanism in their design, as user action is either needed to enable the safety mechanism or user action can disable the safety mechanism.

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 26262:2011, the latent faults to be considered are when the faults in a function are both present: the capability to violate a safety goal and to cause a fault in the safety mechanism. 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 CPU and memories in consideration for any primary diagnostic that is executed via software.

5.1 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 ADC, Flash pump and oscillator).

5.1.1 Embedded 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.

5.1.2 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.

5.1.3 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

5.2 Power Management Module (PMM)

The power management module (PMM) is responsible for control of switchable power domains. Dependent on the family variant used, one or more power domains can be implemented. Power domains can be permanently configured at manufacturing time by TI or they can be user programmable. To determine the power domains supported on your MCU, see the device-specific data sheet. For programming information, see the device-specific technical reference manual.

5.2.1 Lockstep of Power State Controller (PSCON)

For each implemented power domain, a power state controller (PSCON) implements the control logic. In the Hercules platform, each PSCON is implemented with a diagnostic PSCON in lockstep to check for correct power state control on a cycle-by-cycle basis. Any error detected by lockstep compare is signaled to the ESM.

The PSCON lockstep compare is enabled by default during the power-on reset state. The PSCON lockstep compare can be disabled by the software in order to test the compare functionality. The test of the lockstep comparison feature is supported by a software triggered hardware self-test. The hardware self test can initiate comparison tests for both match and mismatch inputs. Error forcing is also possible to test system level error response. Use of the PSCON lockstep is mandatory.

5.2.2 Privileged Mode Access and Multi-Bit Keys for Control Registers

The PMM design includes features to support avoidance of unintentional control register programmation. These features include use of multi-bit keys in critical configuration registers and limitation of write commands to privilege bus master transactions. A further bus master filtering protection blocks accesses from any bus master except the CPU. Illegal transactions result in a bus error response to the offending bus master.

The operation of this safety mechanism is continuous and cannot be altered by the software. This mechanism can be tested by the software initiation of incorrect transaction with comparison of resulting error response. Use of this safety mechanism is mandatory.

5.2.3 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.2.4 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the power management module, it is highly recommended that 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 power management module memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit (MPU). This ensures that the register write has completed before the read back is initiated.

5.2.5 PSCON Lockstep Comparator Self-Test

The PSCON lockstep comparator diagnostic includes self test features to check for proper operation of the lockstep comparator. Use of this feature is highly recommended.

5.2.6 Notes

- PSCONs continue to function normally during lockstep compare self-test, but no comparison function is present.
- When the CPU is in a halting debug state, no comparison of PSCON outputs is performed.

5.3 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 module.

5.3.1 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. Use of the LPOCLKDET is highly recommended.

5.3.2 PLL Slip Detection

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 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. Use of the PLL slip detection diagnostic is highly recommended.

5.3.3 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 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. Use of the DCC as a diagnostic for clocks is highly recommended. The cyclical check applied by the DCC module provides an inherent level of self checking (autocoverage), which can be considered for application in latent fault diagnostics.

5.3.4 Monitoring of External Clock Outputs (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 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. Use of the ECLK feature for external monitoring of internal clocks is optional.

5.3.5 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. Use of the DWD functionality is optional.

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. Use of the DWWD functionality is recommended.

5.3.6 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.

5.3.7 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.3.8 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped clock control registers in the system 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 system module memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.3.9 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.

5.4 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.

5.4.1 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. Use of this feature is considered optional.

5.4.2 Software Check of Cause of Last Reset

The system 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. Software use of the SYSESR to check last reset cause is highly recommended.

5.4.3 Software Warm Reset Generation

The system 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. Use of the software warm reset is optional.

5.4.4 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.

5.4.5 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. The system module includes shadow registers that can be used to support failure recovery via software. Use of the shadow register status information by boot software is highly recommended.

5.4.6 External Watchdog

An external watchdog can provide secondary diagnostic. For more information on this diagnostic, see External Watchdog.

5.4.7 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.4.8 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped reset control registers in the system module, it is highly recommended that 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 system module memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.4.9 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.

5.5 System 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.

5.5.1 Privileged Mode Access and Multi-Bit Enable Keys

The system module design includes features to support avoidance of unintentional control register programmation. 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. Use of this safety mechanism is mandatory.

5.5.2 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the system module, it is highly recommended that 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 system module memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.5.3 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.5.4 Notes

- Depending on targeted metrics, a user can elect to implement a periodic software test of static configuration registers in the system 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 module.

5.6 Error Signaling Module (ESM)

The ESM provides unified aggregation and prioritization of on-board hardware diagnostic errors. For more information see Management of Errors

5.6.1 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.6.2 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. The use of a boot time software test of error path reporting is highly recommended. The use of a periodic software test of error path reporting is recommended.

5.6.3 Shadow Registers

The use of a two stage cold and warm reset scheme on the device allows the implementation of shadow registers. Shadow registers 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. The error signaling module includes shadow registers that can be used to support failure recovery via software. Use of the shadow register status information by boot software is highly recommended.

5.6.4 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the ESM, it is highly recommended that the 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 ESM memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.6.5 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

5.7 CPU Subsystem

The Hercules product family relies on the ARM Cortex-R4F CPU to provide general-purpose processing. The Cortex-R4F is a high performance CPU with embedded safety diagnostics. The R4F is also designed for easy integration into a 1oo1D lockstep configuration. These aspects make the Cortex-R4F an outstanding CPU for functional safety products.

5.7.1 Lockstep CPU Diagnostic

The Hercules product family includes a lockstep processor diagnostic. This feature includes the addition of a diagnostic Cortex-R4F 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. Use of the lockstep diagnostic is mandatory.

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.

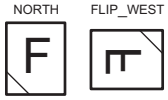
The CCM logic provides self test and error forcing capability via software triggered hardware. The self test ensures that the CCM compare logic is working properly. The error forcing capability allows you to test the system level response to a lockstep miscompare. Use of the self test and error forcing tests at reset is highly recommended.

5.7.1.1 Measures to Mitigate Common Mode Failure

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

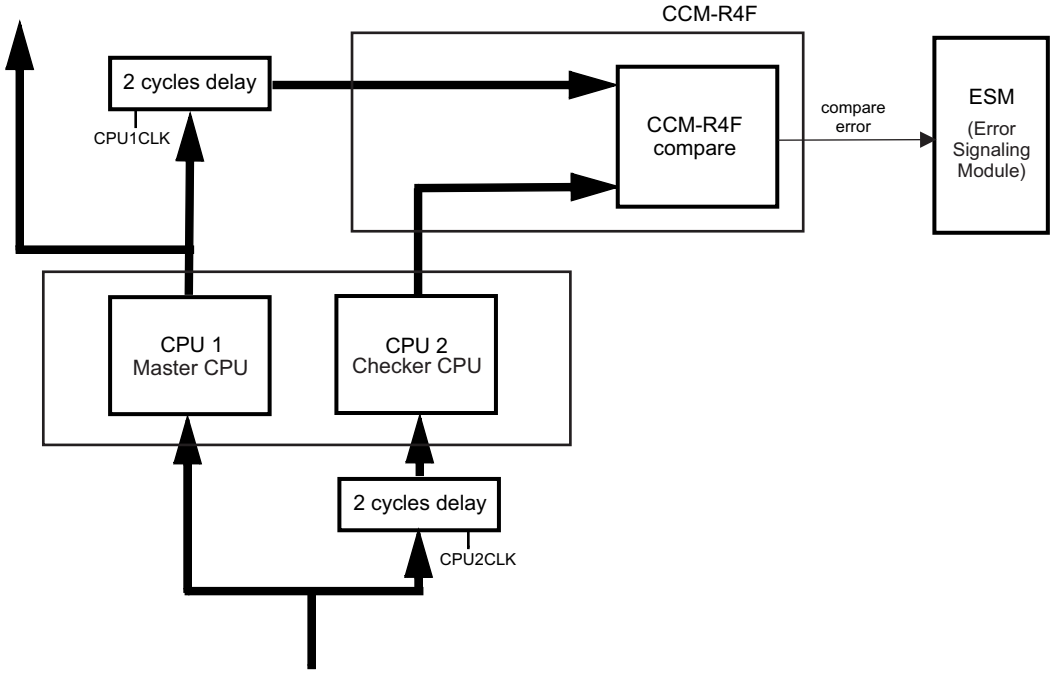
- Physical diversity:
  - Physical core hard macros are spaced at least 100 µm apart.
  - One core is flipped and rotated in orientation relative to the other core. For example, if one core is considered "North", the second would be "Flip West" as shown in Figure 6

Figure 6. 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 7.

Figure 7. 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.

5.7.2 CPU Logic Built In Self Test (LBIST) Self -Test Controller (STC)

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 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:

- Enable the self_check_key and fault_ins bits in the STCSTSCR register.
- Enable STC test interval zero and execute the test
- Upon completion of test, the fail bit should be set to 1 in the STC global status register.
- Disable either or both the self_check_key and fault_ins bits in the STCSTSCR register.
- Restart the self test by programming bit 0 of the STCGCR, causing self-test restart.
- Upon completion of the test, the fail bit should be set to 0 in the STC global status register.

Use of the LBIST logic at boot time is highly recommended. Use of the LBIST logic for periodic execution during normal execution is optional. The cyclical check applied by the LBIST module provides an inherent level of self checking (autocoverage), which can be considered for application in latent fault diagnostics.

5.7.3 CPU Memory Protection Unit (MPU)

The Hercules implementation of the Cortex-R4F 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. Use of the MPU is highly recommended.

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 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. Additional software-based testing of the MPU for proper operation and error response is optional.

5.7.4 Online Profiling Using the Performance Monitoring Unit

The Cortex-R4F 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. Use of the PMU for online diagnostic profiling is optional.

5.7.5 Internal or External Watchdog

An internal or external watchdog can provide secondary diagnostic. For more information on these diagnostics, see Internal Watchdog or External Watchdog.

5.7.6 Illegal Operation and Instruction Trapping

The Cortex-R4F 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
- Floating-point underflow and overflow
- Floating point divide by zero
- Privilege violation

5.7.7 Software Read Back of Written Configuration

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.

5.7.8 CPU Lockstep Comparator (CCM) Self-Test

The CPU lockstep comparator (CCM) diagnostic includes self test features to check for proper operation of the lockstep comparator. Use of this feature is highly recommended.

5.7.9 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-R4F. 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.

5.8 Primary Embedded Flash

The primary embedded Flash memory is a non-volatile memory that is tightly coupled to the ATCM port of the Cortex-R4F 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.

5.8.1 Flash ECC

The on-chip Flash memory is supported by single error correction, dual error detection (SECDED) error-correcting code (ECC) diagnostic. It is connected by a 64-bit-wide data bus interface (ATCM) to the Cortex-R4F 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 ATCM Flash access is located in the Cortex-R4F 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-R4F 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. Use of the Flash ECC is highly recommended. The cyclical check applied by the ECC module provides an inherent level of self checking (autocoverage), which can be considered for application in latent fault diagnostics.

5.8.2 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.

5.8.3 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. Use of this feature is highly recommended. The cyclical check applied by the Flash wrapper address ECC module provides an inherent level of self checking (autocoverage), which can be considered for application in latent fault diagnostics.

5.8.4 ATCM Address Bus Parity

The on-chip ATCM bus connection to Flash memory is supported by a parity diagnostic on the address 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 programmation of the BUS_PAR_DIS key in the FPAR_OVR register of the Flash wrapper. Use of the ATCM address bus parity diagnostic is highly recommended.

5.8.5 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 or the DMA. 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. Periodic execution of the CRC integrity check at runtime is recommended. The cyclical check applied by the hardware CRC module provides an inherent level of self checking (autocoverage), which can be considered for application in latent fault diagnostics.

5.8.6 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 mandatory feature of the architecture and cannot be modified by the software.

5.8.7 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. Use of the Flash sector protection feature is highly recommended.

5.8.8 Periodic Read Back of 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. The use of read back of configuration register mechanism is recommended.

5.8.9 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the Flash wrapper, it is highly recommended that 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 Flash wrapper memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.8.10 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.

5.9 Flash EEPROM Emulation (FEE)

The Hercules platform architecture includes the capability for a separate bank of Flash memory to be used as Flash emulated EEPROM (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.

5.9.1 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. Use of the FEE ECC is highly recommended. The cyclical check applied by the FEE ECC logic provides an inherent level of self checking (autocoverage), which can be considered for application in latent fault diagnostics.

5.9.2 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 or the DMA. 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. Periodic execution of the FEE CRC integrity check at runtime is recommended. The cyclical check applied by the hardware CRC module provides an inherent level of self checking (autocoverage), which can be considered for application in latent fault diagnostics.

5.9.3 Bit Multiplexing in FEE

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 mandatory feature of the architecture and cannot be modified by the software.

5.9.4 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. Use of the FEE sector protection feature is highly recommended.

5.9.5 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.9.6 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the FEE wrapper, it is highly recommended that 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 FEE wrapper memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.9.7 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.

5.10 Primary Embedded SRAM

The primary embedded SRAM is a volatile memory that is tightly coupled to the BTCM ports of the Cortex-R4F 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)

5.10.1 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-R4F 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-R4F 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-R4F 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. Use of the SRAM ECC is highly recommended. The cyclical check applied by the ECC logic provides an inherent level of self checking (autocoverage), which can be considered for application in latent fault diagnostics.

5.10.2 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.

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

5.10.3 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-R4F PMU export of events must also be enabled for this function to operate. Use of the correctable SRAM ECC profiling feature is recommended.

5.10.4 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 programmation of the address parity disable key in the RAMCTRL register of the SRAM wrapper. Use of the BTCM address and control bus parity diagnostic is highly recommended.

5.10.5 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. Use of the duplicated address decode diagnostic is highly recommended.

5.10.6 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 8 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.

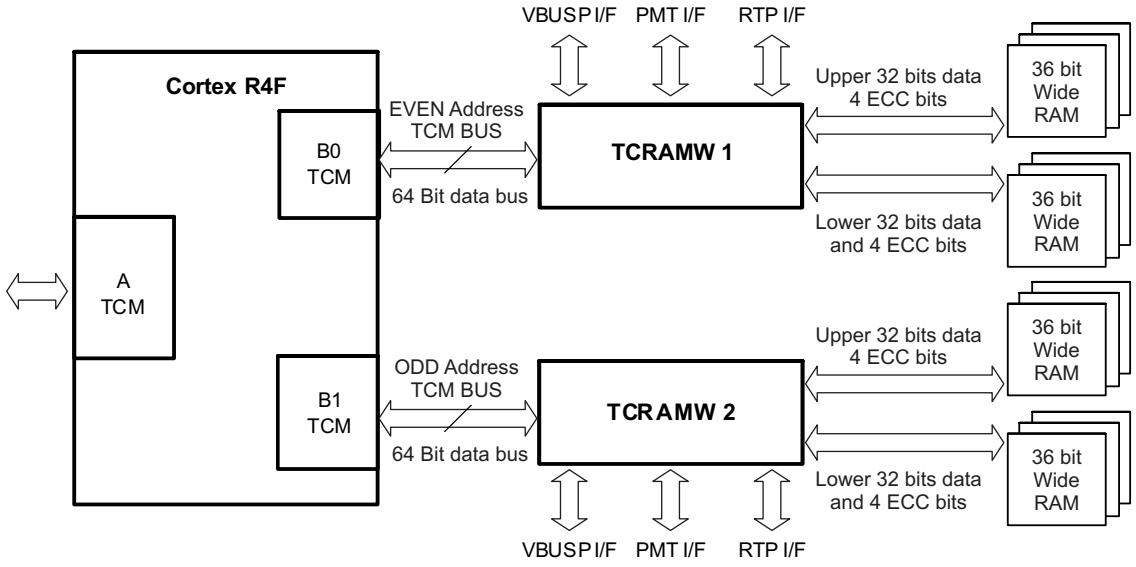


Figure 8. Block Level Implementation of CPU SRAM

5.10.7 Programmable Memory BIST (PBIST)

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 been implemented via software programmation 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.

Use of the PBIST logic at device initialization is highly recommended. Use of the PBIST logic for periodic execution during normal execution is optional. The cyclical check applied by the PBIST logic provides an inherent level of self checking (autocoverage), which can be considered for application in latent fault diagnostics.

5.10.8 SRAM Bit Multiplexing

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 mandatory feature of the architecture and cannot be modified by the software.

5.10.9 SRAM Hardware CRC-64

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 or by DMA. The comparison of results, indication of fault, and fault response are the responsibility of the software managing the test. Because most static values are stored in Flash, execution of a CRC on static contents of the SRAM is optional. The cyclical check applied by the CRC logic provides an inherent level of self checking (autocoverage), which can be considered for application in latent fault diagnostics.

5.10.10 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.10.11 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the SRAM wrapper, it is highly recommended that 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 SRAM wrapper memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.10.12 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 highly recommended at boot time as a latent diagnostic.

5.10.13 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.

5.11 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 (DMA, 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.

5.11.1 Error Trapping (including Peripheral Slave Error Trapping)

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. Use of this safety mechanism is mandatory. These features can be tested via software through the insertion of invalid bus transactions.

5.11.2 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. Use of these mechanisms is highly recommended.

5.11.3 Internal or External Watchdog

An internal or external watchdog can provide secondary indication of a transaction that has timed out due to an issue with interconnect. For more information on these diagnostics, see Internal Watchdog or External Watchdog.

5.11.4 Information Redundancy Techniques

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. The use of information redundancy techniques in L2/L3 interconnect transactions is recommended.

5.11.5 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.11.6 Software Test of Basic Functionality Including Error Tests

A software test can be utilized to test for 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. The use of a boot time software test of basic functionality is highly recommended. The use of a periodic software test of basic fhunctionality reporting is recommended.

5.11.7 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the PCR, it is highly recommended that 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 PCR memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.11.8 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.

5.12 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.

5.12.1 Autoload Self-Test

The EFuse controller has a self-test logic that executes automatically after autoload completion. Error is indicated via ESM. The test can be subsequently triggered by the software. Use of the autoload self-test diagnostic is mandatory.

5.12.2 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 (autocoverage), which can be considered for application in latent fault diagnostics.

5.12.3 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.12.4 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers, it is highly recommended that software implement a test to confirm proper operation of all control register writes. To support this software test, it is recommended to configure the target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.13 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.

5.13.1 Autoload Self-Test

The OTP autoload controller has a self-test logic that executes automatically after autoload completion. Error is indicated via ESM. The test can be subsequently triggered by the software. Use of the autoload self-test diagnostic is mandatory.

5.13.2 OTP Autoload ECC

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

5.13.3 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.13.4 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers, it is highly recommended that software implement a test to confirm proper operation of all control register writes. To support this software test, it is recommended to configure the target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.13.5 Notes

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

5.14 I/O Multiplexing (IOMM)

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

5.14.1 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. Use of the locking mechanism for control registers is highly recommended.

5.14.2 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.

5.14.3 Error Trapping

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. Use of the error trapping functionality is mandatory.

5.14.4 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.14.5 Software Test of Function Using I/O Loopback

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 disabled.

I/O loopback is not enabled at reset. Software is necessary to configure and execute the diagnostic. Error response is managed by the software. Use of the I/O loopback mechanism at boot time is highly recommended. Use of the I/O loopback mechanism periodically is optional.

5.14.6 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the IOMM, it is highly recommended that 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 IOMM memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.14.7 Notes

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

5.15 Vectored Interrupt Module (VIM)

The vectored interrupt module (VIM) is used to interface peripheral interrupts to the Cortex-R4F 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.

5.15.1 VIM SRAM Parity

The VIM 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. Use of the VIM SRAM parity feature is highly recommended.

5.15.2 VIM SRAM PBIST

The VIM SRAM can be tested using the PBIST memory BIST engine. Execution of PBIST tests on VIM SRAM after reset is highly recommended. For more details on this diagnostic, see PBIST.

5.15.3 VIM SRAM Bit Multiplexing

The VIM SRAM is implemented with a memory design such that logically adjacent bits are not physically adjacent. Use of this safety mechanism is mandatory. For more details on this safety mechanism, see SRAM Bit Multiplexing.

5.15.4 VIM SRAM CRC-64 Testing

The VIM SRAM contents can be tested periodically using the hardware CRC-64 diagnostic. As VIM SRAM contents tend to be static, use of this diagnostic is recommended. For more details on this diagnostic, see SRAM Hardware CRC-64.

5.15.5 Periodic Software Test of VIM Operation 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. The use of periodic software test of VIM operation including error tests is highly recommended.

5.15.6 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.15.7 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the VIM, it is highly recommended that 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 VIM memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.15.8 Internal or External Watchdog

An internal or external watchdog can provide secondary indication of a transaction that has timed out due to an issue with interconnect. For more information on these diagnostics, see Internal Watchdog or External Watchdog.

5.15.9 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.

5.16 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.

5.16.1 Use of 2nd Counter as Diagnostic

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-R4F 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. The use of a second counter to diagnose faults in RTI is recommended.

5.16.2 Internal or External Watchdog

A internal or external watchdog can provide indication of a fault in the RTI module. For more information on these diagnostics, see Internal Watchdog or External Watchdog.

5.16.3 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.16.4 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the RTI, it is highly recommended that 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 RTI memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.16.5 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.

The FlexRay Network Time Unit (NTU) can be used to serve as the clock source for the operating system timer. This can be utilized for network synchronization of safety critical tasks. If the NTU detects loss of synch with the network, system can automatically revert to local clock sources.

5.17 Direct Memory Access (DMA)

The direct memory access (DMA) module is used to move data from one location to another inside the system. This is typically used for peripheral configuration (Flash and SRAM to peripheral transfer) and peripheral data update (peripheral buffer memory transfer to CPU SRAM for processing). The DMA is typically used by the operating system to offload bus transactions from the CPU in order to improve overall system performance. The DMA has a local SRAM that is used for channel control information.

5.17.1 Memory Protection Unit (MPU)

The DMA 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 DMA driver controls the MPU and changes the MPU settings based on the needs of the system. A violation of a configured memory protection policy results in an ESM error.

The MPU is not enabled at reset. Software must enable, configure and test the MPU. Use of the MPU is highly recommended.

5.17.2 Non-Privileged Bus Master Access

The DMA is a non-privileged bus master. Since writes to critical configuration registers across the MCU is limited to privileged mode only, this mechanism prevents inadvertent configuring of these registers by the DMA

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.

5.17.3 Information Redundancy Techniques

Information redundancy techniques can be applied using the DMA module. 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. The implementation of information redundancy techniques on DMA transactions is recommended.

5.17.4 DMA SRAM Parity

The DMA 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. Use of the DMA SRAM parity feature is highly recommended.

5.17.5 DMA SRAM PBIST

The DMA SRAM can be tested using the PBIST memory BIST engine. Execution of PBIST tests on DMA SRAM after reset is highly recommended. For more details on this diagnostic, see PBIST.

5.17.6 DMA SRAM Bit Multiplexing

The DMA SRAM is implemented with a memory design such that logically adjacent bits are not physically adjacent. Use of this safety mechanism is mandatory. For more details on this safety mechanism, see SRAM Bit Multiplexing.

5.17.7 DMA SRAM CRC-64 Testing

The DMA SRAM contents can be tested periodically using the hardware CRC-64 diagnostic. As DMA SRAM contents tend to be static, use of this diagnostic is recommended. For more details on this diagnostic, see SRAM Hardware CRC-64.

5.17.8 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.17.9 Software Test of Basic Functionality 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. The use of a boot time software test of basic functionality is highly recommended. The use of a periodic software test of basic functionality reporting is recommended.

5.17.10 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the DMA, it is highly recommended that 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 DMA memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.18 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.

Each 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.

5.18.1 Memory Protection Unit (MPU)

The HTU 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 N2HET driver controls the MPU and changes the MPU settings based on the needs of system implementation. A violation of a configured memory protection policy results in an ESM error.

The MPU is not enabled at reset. Software must enable, configure and test the MPU. Use of the MPU is highly recommended.

5.18.2 Information Redundancy Techniques

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. Splitting functionality between the two implemented N2HET modules can provide reduced probability of common mode failure as opposed to implementing a function and its diagnostic on a single N2HET.

Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator. The use of information redundancy techniques on N2HET operations is highly recommended.

5.18.3 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. Each N2HET has a dedicated channel internally connected to one of the two DCC modules. 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.

The use of DCC as a program sequence watchdog on N2HET operations is highly recommended.

5.18.4 Monitoring by Second N2HET

The N2HET supports an internal channel connection between the two N2HET modules to facilitate easy monitoring of one N2HET by a second N2HET. This feature is supported as a convenience to limit the number of functional channels at device level, which must be utilized for diagnostic purposes. Alternatively external connections can be used. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator.

The use of internal monitoring on N2HET operations is recommended.

5.18.5 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. The use of a boot time software test of basic functionality is highly recommended. The use of a periodic software test of basic functionality reporting is optional.

The N2HET implementation supports both digital and analog loopback capabilities for the I/Os. 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 disabled. For best results any tests of the N2HET functionality should include the I/O loopback.

5.18.6 N2HET and HTU SRAM Parity

The N2HET 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. Use of the N2HET SRAM parity feature is highly recommended.

5.18.7 N2HET and HTU SRAM PBIST

The N2HET SRAM can be tested using the PBIST memory BIST engine. Execution of PBIST tests on N2HET SRAM after reset is highly recommended. For more details on this diagnostic, see PBIST.

5.18.8 N2HET and HTU SRAM Bit Multiplexing

The N2HET SRAM is implemented with a memory design such that logically adjacent bits are not physically adjacent. Use of this safety mechanism is mandatory. For more details on this safety mechanism, see SRAM Bit Multiplexing.

5.18.9 N2HET and HTU SRAM CRC-64 Testing

The N2HET SRAM contents can be tested periodically using the hardware CRC-64 diagnostic. As HTU SRAM contents tend to be static, use of this diagnostic is recommended as opposed to N2HET SRAM where the contents are dynamic and the use of this diagnostic is optional. For more details on this diagnostic, see SRAM Hardware CRC-64.

5.18.10 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.18.11 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the N2HET/HTU, it is highly recommended that 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 target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.18.12 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.

5.19 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 DMA or CPU. The Hercules device family products implement two modules with shared channels used for fast conversion (ping-pong method). Systems implementing two channels using the dual ADC converters may be able to claim fault tolerance in application.

5.19.1 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. The use of the input self-test mechanism is highly recommended.

5.19.2 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. The use of the converter calibration mechanism at boot is highly recommended. The use of the converter calibration mechanism periodically is optional.

5.19.3 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 shared channels and using both converters, 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. The use of information redundancy techniques on ADC conversions is highly recommended.

5.19.4 ADC SRAM Parity

The MibADC 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. Use of the MibADC SRAM parity feature is highly recommended.

5.19.5 ADC SRAM PBIST

The MibADC SRAM can be tested using the PBIST memory BIST engine. Execution of PBIST tests on MibADC SRAM after reset is highly recommended. For more details on this diagnostic, see PBIST.

5.19.6 ADC SRAM Bit Multiplexing

The MibADC SRAM is implemented with a memory design such that logically adjacent bits are not physically adjacent. Use of this safety mechanism is mandatory. For more details on this safety mechanism, see SRAM Bit Multiplexing.

5.19.7 ADC SRAM CRC-64 Testing

The MibADC SRAM contents can be tested periodically using the hardware CRC-64 diagnostic. As MibADC SRAM contents tend to be more dynamic, use of this diagnostic is optional. For more details on this diagnostic, see SRAM Hardware CRC-64.

5.19.8 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.19.9 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the MibADC, it is highly recommended that 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 target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.19.10 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.

5.20 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.

5.20.1 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. The use of a boot time software test of basic functionality is highly recommended. The use of a periodic software test of basic functionality reporting is optional.

The MibSPI implementation supports both digital and analog loopback capabilities for the I/Os. 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 disabled. For best results any tests of the MibSPI functionality should include the I/O loopback.

5.20.2 Message Parity

The Hercules MIBSPI 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. The use of this feature is highly recommended.

5.20.3 Information Redundancy Techniques

Information redundancy techniques can be applied via software as an additional runtime diagnostic for MIBSPI 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. Alternatively redundancy can be achieved by implementation of multiple channels in the system. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator. The use of information redundancy techniques in MIBSPI transactions is highly recommended.

5.20.4 MIBSPI SRAM Parity

The MIBSPI 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. Use of the MIBSPI SRAM parity feature is highly recommended.

5.20.5 MIBSPI SRAM PBIST

The MIBSPI SRAM can be tested using the PBIST memory BIST engine. Execution of PBIST tests on MIBSPI SRAM after reset is highly recommended. For more details on this diagnostic, see PBIST.

5.20.6 MIBSPI SRAM Bit Multiplexing

The MIBSPI SRAM is implemented with a memory design such that logically adjacent bits are not physically adjacent. Use of this safety mechanism is mandatory. For more details on this safety mechanism, see SRAM Bit Multiplexing.

5.20.7 MIBSPI SRAM CRC-64 Testing

The MIBSPI SRAM contents can be tested periodically using the hardware CRC-64 diagnostic. As MIBSPI SRAM contents tend to be more dynamic, use of this diagnostic is optional. For more details on this diagnostic, see SRAM Hardware CRC-64.

5.20.8 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.20.9 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the MibSPI, it is highly recommended that 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 target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.20.10 Notes

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

5.21 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.

5.21.1 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. The use of a boot time software test of basic functionality is highly recommended. The use of a periodic software test of basic functionality reporting is optional.

The SPI implementation supports both digital and analog loopback capabilities for the I/Os. 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 disabled. For best results any tests of the SPI functionality should include the I/O loopback.

5.21.2 Message Parity

The Hercules MIBSPI supports insertion of a parity bit into the data payload of every outgoing SPI message by hardware. Evaluation of incoming message parity is also supported by hardware. Detected errors generate an interrupt to the CPU. The use of this feature is highly recommended.

5.21.3 Information Redundancy Techniques

Information redundancy techniques can be applied via software as an additional runtime diagnostic for SPI 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. Alternatively redundancy can be achieved by implementation of multiple channels in the system. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator. The use of information redundancy techniques in SPI transactions is highly recommended.

5.21.4 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.21.5 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the SPI, it is highly recommended that 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 target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.22 Inter-Integrated Circuit (I2C)

The I2C module provides a multi-master serial bus compliant to the I2C protocol.

5.22.1 Software Test of Function

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. The use of a boot time software test of basic functionality is highly recommended. The use of a periodic software test of basic functionality reporting is optional.

5.22.2 Information Redundancy Techniques

Information redundancy techniques can be applied via software as an additional runtime diagnostic for I2C 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. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator. The use of information redundancy techniques in I2C transactions is highly recommended.

5.22.3 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.22.4 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the I2C, it is highly recommended that 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 target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.23 Serial Communication Interface (SCI)

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

5.23.1 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. The use of a boot time software test of basic functionality is highly recommended. The use of a periodic software test of basic functionality reporting is optional.

The SCI implementation supports both digital and analog loopback capabilities for the I/Os. 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 disabled. For best results any tests of the SCI functionality should include the I/O loopback.

5.23.2 Information Redundancy Techniques

Information redundancy techniques can be applied via software as an additional runtime diagnostic for SCI 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. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator. The use of information redundancy techniques in SCI transactions is highly recommended.

5.23.3 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.23.4 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the SCI, it is highly recommended that 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 target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.24 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.

5.24.1 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. The use of a boot time software test of basic functionality is highly recommended. The use of a periodic software test of basic functionality reporting is optional.

The LIN implementation supports both digital and analog loopback capabilities for the I/Os. 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 disabled. For best results any tests of the LIN functionality should include the I/O loopback.

5.24.2 Information Redundancy Techniques Including End-to-End Safing

Information redundancy techniques can be applied via software as an additional runtime diagnostic for LIN 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. Use of this mechanism is highly recommended.

5.24.3 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.24.4 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the LIN, it is highly recommended that 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 target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.24.5 Notes

When using the LIN module in SCI mode, see Section 5.23.

5.25 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.

5.25.1 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. The use of a boot time software test of basic functionality is highly recommended. The use of a periodic software test of basic functionality reporting is optional.

The DCAN implementation supports both digital and analog loopback capabilities for the I/Os. 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 disabled. For best results any tests of the DCAN functionality should include the I/O loopback.

5.25.2 Information Redundancy Techniques Including End-to-End Safing

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. Use of this mechanism is highly recommended.

5.25.3 DCAN SRAM Parity

The DCAN 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. Use of the DCAN SRAM parity feature is highly recommended.

5.25.4 DCAN SRAM PBIST

The DCAN SRAM can be tested using the PBIST memory BIST engine. Execution of PBIST tests on DCAN SRAM after reset is highly recommended. For more details on this diagnostic, see PBIST.

5.25.5 DCAN SRAM Bit Multiplexing

The DCAN SRAM is implemented with a memory design such that logically adjacent bits are not physically adjacent. Use of this safety mechanism is mandatory. For more details on this safety mechanism, see SRAM Bit Multiplexing.

5.25.6 DCAN SRAM CRC-64 Testing

The DCAN SRAM contents can be tested periodically using the hardware CRC-64 diagnostic. As DCAN SRAM contents tend to be more dynamic, use of this diagnostic is optional. For more details on this diagnostic, see SRAM Hardware CRC-64.

5.25.7 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.25.8 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the DCAN, it is highly recommended that 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 target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.26 FlexRay, FlexRay Transfer Unit (FTU)

The FlexRay interface provides data interconnect with pre-defined, time triggered (TDMA) slots for each connected node to communicate. The FlexRay network supports higher data throughput than legacy CAN networks and is compliant to the FlexRay protocol. In typical automotive usage, the FlexRay network is the backbone network used for connection to a chassis and safety system such that information can be passed between safety critical systems.

The device provides two FlexRay channels. Depending on configuration, the device could connect two different FlexRay networks and act as a gateway, or the device could act as two nodes on the same FlexRay network if multiple time slots are required in the system network concept. An external transceiver device is needed to provide the physical layer connection to the FlexRay network. A separate PLL is integrated to generate the frequencies required for the FlexRay network.

The Hercules FlexRay module includes dedicated DMA capability in the FlexRay Transfer Unit (FTU).

5.26.1 Memory Protection Unit (MPU)

The FTU 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 FlexRay driver controls the MPU and changes the MPU settings based on the needs of network implementation. A violation of a configured memory protection policy results in an ESM error.

The MPU is not enabled at reset. Software must enable, configure and test the MPU. Use of the MPU is highly recommended.

5.26.2 Non-Privileged Bus Master Access

The FTU is a non-privileged bus master. Since writes to critical configuration registers across the MCU is limited to privileged mode only, this mechanism prevents inadvertent configuring of these registers by the FTU.

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.

5.26.3 Software Test of Function Using I/O Loopback in Transceiver

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. The use of a boot time software test of basic functionality is highly recommended. The use of a periodic software test of basic functionality reporting is optional.

Loopback mode of the FlexRay transceiver allows for testing of the signal path including the physical layer. Error response, diagnostic testability, and any necessary software requirements are defined by the system integrator. For best results any tests of the FlexRay functionality should include I/O loopback.

5.26.4 Information Redundancy Techniques Including End-to-End Safing

Information redundancy techniques can be applied via software as an additional runtime diagnostic for FlexRay 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. Use of this mechanism is highly recommended.

5.26.5 1oo2 Voting Using Both FlexRay Channels

The Hercules FlexRay module has a two channel implementation. The two channels can be used to redundantly receive the same message (dependent on network definition). FlexRay message buffers can be configured to receive messages from one or both channels. In case a single buffer is used for both channels, it stores the first semantically correct message received on either channel. To enhance separation of channels, a buffer per channel can be configured and then a software comparison can be performed. Error response, diagnostic testability, and any necessary software requirements are defined by the system integrator. Use of this feature is recommended.

5.26.6 FlexRay and FTU SRAM Parity

The FlexRay and FTU SRAMs include parity diagnostics 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 for each SRAM individually. Use of the FlexRay and FTU parity feature is highly recommended.

5.26.7 FlexRay and FTU SRAM PBIST

The FlexRay and FTU SRAMs can be tested using the PBIST memory BIST engine. Execution of PBIST tests on FlexRay and FTU SRAMs after reset is highly recommended. For more details on this diagnostic, see PBIST.

5.26.8 FlexRay and FTU SRAM Bit Multiplexing

The FlexRay and FTU SRAMs are implemented with a memory design such that logically adjacent bits are not physically adjacent. Use of this safety mechanism is mandatory. For more details on this safety mechanism, see SRAM Bit Multiplexing.

5.26.9 FlexRay and FTU SRAM CRC-64 Testing

The FlexRay and FTU SRAMs contents can be tested periodically using the hardware CRC-64 diagnostic. As FTU SRAM contents tend to be static, use of this diagnostic is recommended as opposed to FlexRay SRAM where the contents are dynamic and the use of this diagnostic is optional. For more details on this diagnostic, see SRAM Hardware CRC-64.

5.26.10 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.26.11 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the FlexRay/FTU, it is highly recommended that 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 target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.26.12 Notes

- The FlexRay communication controller does not have the capability to calculate the header CRC. The host provides the header CRCs for all transmit buffers.
- Due to the deterministic nature of FlexRay transmissions, this interface is preferred for transmitting timing sensitive sensor data and synchronization of safety critical tasks across multiple network nodes.

5.27 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.

5.27.1 Software Test of Function Using I/O Checking

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. The use of a boot time software test of basic functionality is highly recommended. The use of a periodic software test of basic functionality reporting is optional.

The GIO module does not support a distinct I/O loopback mode. However it is possible to support I/O checking using normal functionality. To do this software generates output and reads back and checks for the same value in the input registers. This implements functionality similar to analog loopback in other modules. For best results, any tests of the GIO functionality should include the I/O loopback.

5.27.2 Information Redundancy Techniques

Information redundancy techniques can be applied via software as an additional runtime diagnostic on GIO function. There are many techniques that can be applied, such as multiple inputs and read back of output with an input channel. Signals from many other peripherals can be used as GIO if not used for primary function. Use of a GIO module signal and a non GIO module signal for multi-channel implementation can reduce probability of common mode failures.

Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator. The use of information redundancy techniques on GIO functions is highly recommended.

5.27.3 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.27.4 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the GIO, it is highly recommended that 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 target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.27.5 Notes

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

5.28 Ethernet

The Hercules platform includes an Ethernet media access controller (EMAC) and physical layer (PHY) device management data input/output (MDIO) module. These modules allow connection of the Hercules MCU to a 10 and 100 Mb Ethernet network. The Ethernet network provides higher network throughput than LIN, CAN, or FlexRay.

5.28.1 Non-Privileged Bus Master Access

The Ethernet module is a non-privileged bus master. Since writes to critical configuration registers across the MCU are limited to privileged mode 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.

5.28.2 Software Test of Function Using I/O Loopback in 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. The use of a boot time software test of basic functionality is highly recommended. The use of a periodic software test of basic functionality reporting is optional.

Most Ethernet 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 Ethernet functionality should include I/O loopback.

5.28.3 Information Redundancy Techniques Including End-to-End Safing

Information redundancy techniques can be applied via software as an additional runtime diagnostic for Ethernet 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. Use of this mechanism is highly recommended.

5.28.4 EMAC SRAM PBIST

The EMAC SRAM can be tested using the PBIST memory BIST engine. Execution of PBIST tests on EMAC SRAM after reset is highly recommended. For more details on this diagnostic, see PBIST.

5.28.5 EMAC SRAM Bit Multiplexing

The EMAC SRAM is implemented with a memory design such that logically adjacent bits are not physically adjacent. Use of this safety mechanism is mandatory. For more details on this safety mechanism, see SRAM Bit Multiplexing.

5.28.6 EMAC SRAM CRC-64 Testing

The EMAC SRAM contents can be tested periodically using the hardware CRC-64 diagnostic. As EMAC SRAM contents tend to be dynamic in application, use of this diagnostic is optional. For more details on this diagnostic, see SRAM Hardware CRC-64.

5.28.7 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.28.8 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the EMAC, it is highly recommended that 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 target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.29 External Memory Interface (EMIF)

The external memory interface is used to provide device access to off-chip memories or devices, which support a memory interface. Support is provided for both synchronous (SDRAM) and asynchronous (NOR Flash, SRAM) memories.

5.29.1 Information Redundancy Techniques

Information redundancy techniques can be applied via software as an additional runtime diagnostic for EMIF transactions. 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. The use of information redundancy techniques in EMIF transactions is highly recommended.

5.29.2 EMIF Memory CRC-64 Testing

The contents of external memories connected to the EMIF can be tested periodically using the hardware CRC-64 diagnostic. This diagnostic is especially useful for static memory contents, but is less useful if the memory contents change dynamically in application. For more details on this diagnostic, see SRAM Hardware CRC-64. For non-volatile memories use of this diagnostic at boot time is highly recommended and periodic usage is recommended. For volatile memories the same recommendations as for internal SRAMs apply.

5.29.3 Periodic Read Back of 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. The use of read back of configuration registers mechanism is recommended.

5.29.4 Software Read Back of Written Configuration

In order to ensure proper configuration of memory-mapped control registers in the EMIF, it is highly recommended that 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 target memory space as a strongly ordered, non-bufferable memory region using the Cortex-R4F memory protection unit. This ensures that the register write has completed before the read back is initiated.

5.29.5 Notes

Safety critical data from external memories can be transferred or copied to internal memory in the safe island region for higher integrity operations.

5.30 JTAG Debug, Trace, Calibration, 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.

5.30.1 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. Hardware disable of the JTAG port is recommended.

5.30.2 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, trace, and calibration logic. An error in unlocking the AJSM results in no error response and no access to the debug logic. Use of the AJSM to lock out debug access is highly recommended.

5.30.3 Internal or External Watchdog

An internal or external watchdog can provide an indication of inadvertent activation of the JTAG logic which results in impact to safety critical execution. For more information on these diagnostics, see Internal Watchdog or External Watchdog.

5.30.4 Notes

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

5.31 Cortex-R4F Central Processing Unit (CPU) Debug and Trace

The Hercules platform supports CPU debug and trace 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 and trace logic includes an independent debug bus master (AHB-AP), the debug unit inside the CPU, and the embedded trace macrocell (ETM-R4). These modules are not recommended to be used during safety critical operation and safety mechanisms are in place to disable this logic.

5.31.1 Disable JTAG Port to Limit Functional Access

Most debug and trace activities are initiated via an external debug tool, which writes commands to the device using the JTAG port. The JTAG port can be disabled on production hardware as described in JTAG Debug/Trace/Calibration/Test Access. Disabling the JTAG port to limit debug module access is highly recommended.

5.31.2 Lockout of JTAG Access using AJSM

The Hercules platform includes the 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, trace, and calibration logic. An error in unlocking the AJSM results in no error response and no access to the debug logic. Use of the AJSM to lock out debug access is highly recommended.

5.31.3 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). Blocking of access to memory-mapped debug components is highly recommended.

5.31.4 CoreSight Debug Logic Key Enables

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. Use of the debug module unlock key is highly recommended.

5.31.5 Internal or External Watchdog

An internal or external watchdog can provide an indication of inadvertent activation of the debug logic which results in impact to safety critical execution. For more information on these diagnostics, see Internal Watchdog or External Watchdog.

5.32 Data Modification Module (DMM)

The Data Modification Module (DMM) provides a calibration bus master functionality. During calibration the DMM is used as a minimally intrusive DMA to copy calibration data to the device. Commands to the DMM are made via JTAG or via memory-mapped registers.

5.32.1 Disable JTAG Port to Limit Functional Access

Most DMM activities are initiated via an external debug tool that writes commands to the device using the JTAG port. The JTAG port can be disabled on production hardware as described in JTAG Debug/Trace/Calibration/Test Access. Disabling the JTAG port to limit DMM module access is highly recommended.

5.32.2 Lockout of JTAG Access using AJSM

The Hercules platform includes the 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, trace, and calibration logic. An error in unlocking the AJSM results in no error response and no access to the debug logic. Use of the AJSM to lock out debug access is highly recommended.

5.32.3 Block Access to Memory Mapped Debug

The DMM peripheral is accessible over the peripheral 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). Blocking of access to DMM control registers is highly recommended.

5.32.4 Disable the DMM Pin Interface

The DMM peripheral has a device level parallel input port, which is typically driven by an external tool or the ram trace port (RTP). For production usage, the DMM pin interface can be disabled to block data input. One possible method is to drive the DMM clock input low and to drive the active low DMM enable input high. Disabling the DMM pin interface in a production environment is highly recommended.

5.32.5 Internal or External Watchdog

An internal or external watchdog can provide an indication of inadvertent activation of the DMM logic which results in impact to safety critical execution. For more information on these diagnostics, see Internal Watchdog or External Watchdog.

5.33 RAM Trace Port (RTP)

The RAM trace port (RTP) is used to log data writes to internal SRAM. This functionality is used in calibration to keep a remote mirror copy of device memory.

5.33.1 Disable JTAG Port to Limit Functional Access

Most RTP activities are initiated via an external debug tool that writes commands to the device using the JTAG port. The JTAG port can be disabled on production hardware as described in JTAG Debug/Trace/Calibration/Test Access. Disabling the JTAG port to limit RTP module access is highly recommended.

5.33.2 Lockout of JTAG Access using AJSM

The Hercules platform includes the 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, trace, and calibration logic. An error in unlocking the AJSM results in no error response and no access to the debug logic. Use of the AJSM to lock out debug access is highly recommended.

5.33.3 Block Access to Memory Mapped Debug

The RTP peripheral is accessible over the peripheral 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). Blocking of access to RTP control registers is highly recommended.

5.33.4 Disable the RTP Pin Interface

The RTP peripheral has a device level parallel output port that is typically connected to an external tool or used to drive the input of a DMM. For production usage, the RTP pin interface can be disabled to block data output. One possible method is to drive the RTP clock input low and to drive the active low RTP enable input high. Disabling the RTP pin interface in a production environment is recommended.

5.33.5 Internal or External Watchdog

An internal or external watchdog can provide an indication of inadvertent activation of the RTP logic which results in impact to safety critical execution. For more information on these diagnostics, see Internal Watchdog or External Watchdog.

5.34 Parameter Overlay Module (POM)

The parameter overlay module (POM) is used to redirect Flash access to a different internal or external memory. This functionality is used during calibration to test updated sections of code or data without the need to perform a time consuming erase and program of Flash or rebuild of code. The POM is implemented as a CoreSight compliant peripheral.

5.34.1 Disable JTAG Port to Limit Functional Access

Most POM activities are initiated via an external debug tool that writes commands to the device using the JTAG port. The JTAG port can be disabled on production hardware as described in JTAG Debug/Trace/Calibration/Test Access. Disabling the JTAG port to limit POM module access is highly recommended.

5.34.2 Lockout of JTAG Access using AJSM

The Hercules platform includes the 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, trace, and calibration logic. An error in unlocking the AJSM results in no error response and no access to the debug logic. Use of the AJSM to lock out debug access is highly recommended.

5.34.3 Block Access to Memory Mapped Debug

The POM is 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). Blocking of access to memory-mapped POM registers is highly recommended.

5.34.4 CoreSight Debug Logic Key Enables

To enable operation of the POM it is necessary to write a defined 32-bit key to an unlock register in the POM. This debug lock protection provides an addition safety mechanism to limit undesired activation. Use of the POM unlock key is highly recommended.

5.34.5 Internal or External Watchdog

An internal or external watchdog can provide an indication of inadvertent activation of the POM logic which results in impact to safety critical execution. For more information on these diagnostics, see Internal Watchdog or External Watchdog.

6 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. 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.

Appendix A Summary of Recommended Safety Feature Usage

Table 2 provides a summary of the safety concept recommendations noted in Section 5 and organized by device partition. Each recommendation is given a unique identifier to aid in requirements management. For each safety feature or diagnostic the recommendation is noted in simplified form as follows:

- M --> Mandatory application
- ++ --> highly recommended
- + --> recommended
- O --> optional

In addition, a list of possible ISO 26262:2011 latent diagnostic measures for each device partition and safety feature and diagnostic combination is provided. For details on each safety feature or diagnostic, see Section 5.

Table 2. Summary of Safety Features and Diagnostics

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Feature Recommendation	Possible ISO 26262:2011 Latent Diagnostics
Power Supply	PWR1	Voltage monitor (VMON)	M	External Voltage Supervisor
	PWR2	External voltage supervisor	++	Voltage monitor (VMON)
Power Management Module (PMM)	PMM1	Lockstep PSCON	M	PSCON lockstep self test
	PMM2	Privileged mode access and multi-bit keys for control registers	M	Software test of register configuration and error response
	PMM3	Periodic software readback of static configuration registers	+	CPU lockstep
	PMM4	Software readback of written configuration	++	CPU lockstep
	PMM5	PSCON lockstep comparator self-test	++	Self-test autocoverage
Clock	CLK1	LPOCLKDET	++	DCC, ECLK, watchdog
	CLK2	PLL slip detector	++	DCC, ECLK, watchdog
	CLK3	Dual Clock Comparator (DCC)	++	DCC Autocoverage, ECLK, watchdog
	CLK4	External monitoring via ECLK	O	LPOCLKDET, PLL slip detector, DCC, watchdog
	CLK5A	Internal watchdog - DWD	O	External watchdog, software test of watchdog configuration and error response
	CLK5B	Internal watchdog - DWWD	+	External watchdog, software test of watchdog configuration and error response
	CLK5C	External watchdog	++	Internal watchdog, software test of watchdog configuration and error response
	CLK6	Periodic software readback of static clock configuration registers	+	CPU lockstep
	CLK7	Software readback of written configuration	++	CPU lockstep
Reset	RST1	External monitoring of warm reset	O	Watchdog
	RST2	Software check of last reset	++	CPU lockstep
	RST3	Software warm reset generation	O	CPU lockstep
	RST4	Glitch filtering on reset pins	M	Watchdog
	RST5	Use of status shadow registers	++	CPU lockstep
	RST6	External watchdog	++	Software test of external watchdog configuration and error response
	RST7	Periodic software readback of static configuration registers	+	CPU lockstep
	RST8	Software readback of written configuration	++	CPU lockstep
System Control	SYS1	Privileged mode access and multi-bit enable keys	M	Software test of register configuration and error response
	SYS2	Software readback of written configuration	++	CPU lockstep
	SYS3	Periodic software readback of static configuration registers	+	CPU lockstep

Table 2. Summary of Safety Features and Diagnostics (continued)

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Feature Recommendation	Possible ISO 26262:2011 Latent Diagnostics
Error Signaling Module (ESM)	ESM1	Periodic software readback of static configuration registers	+	CPU lockstep
	ESM2A	Boot time software test of error path reporting	++	CPU lockstep
	ESM2B	Periodic software test of error path reporting	+	CPU lockstep
	ESM3	Use of status shadow registers	++	CPU lockstep
	ESM4	Software readback of written configuration	++	CPU lockstep
Cortex-R4F Central Processing Unit (CPU)	CPU1	Lockstep compare	M	CCM (lockstep) self test, LBIST
	CPU2A	Boot time execution of LBIST STC	++	LBIST autocoverage
	CPU2B	Periodic execution of LBIST STC	O	LBIST autocoverage
	CPU3	MPU	++	CPU lockstep, LBIST
	CPU4	Online profiling using PMU	O	CPU lockstep, LBIST
	CPU5A	Internal watchdog - DWD	O	External watchdog, software test of watchdog configuration and error response
	CPU5B	Internal watchdog - DWWD	+	External watchdog, software test of watchdog configuration and error response
	CPU5C	External watchdog	++	Internal watchdog, software test of watchdog configuration and error response
	CPU6	Illegal operation and instruction trapping	++	CPU lockstep, LBIST
	CPU7	Software readback of written configuration	++	CPU lockstep
	CPU8	Lockstep comparator (CCM) self-test	++	Self-test autocoverage
Primary Flash and Level 1 (L1) Interconnect	FLA1	Flash Data ECC	++	CPU lockstep, LBIST, ECC autocoverage
	FLA2	Hard error cache and livelock	M	CPU lockstep, LBIST
	FLA3	Flash wrapper address ECC	++	Software test of Flash wrapper ECC logic, ECC autocoverage
	FLA4	Address parity	++	CPU lockstep, LBIST
	FLA5A	Boot time hardware CRC check of Flash memory contents	++	CRC autocoverage
	FLA5B	Periodic hardware CRC check of Flash memory contents	+	CRC autocoverage
	FLA6	Bit multiplexing in Flash array	M	ECC, CRC test
	FLA7	Flash sector protection	++	CRC test
	FLA8	Periodic software readback of static configuration registers	+	CPU lockstep
	FLA9	Software readback of written configuration	++	CPU lockstep
Flash Emulated EEPROM (FEE)	FEE1	FEE Data ECC	++	Software test of data ECC in EEPROM emulation Flash wrapper, ECC autocoverage
	FEE2A	Boot time hardware CRC check of FEE memory contents	++	CRC autocoverage
	FEE2B	Periodic hardware CRC check of FEE memory contents	+	CRC autocoverage
	FEE3	Bit multiplexing in FEE array	M	ECC, CRC test
	FEE4	FEE sector protection	++	CRC test
	FEE5	Periodic software readback of static configuration registers	+	CPU lockstep
	FEE6	Software readback of written configuration	++	CPU lockstep

Table 2. Summary of Safety Features and Diagnostics (continued)

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Feature Recommendation	Possible ISO 26262:2011 Latent Diagnostics
SRAM and Level 1 (L1) Interconnect	RAM1	Data ECC	++	CPU lockstep, LBIST, ECC autocoverage, PBIST
	RAM2	Hard error cache and livelock	M	CPU lockstep, LBIST
	RAM3	Correctable ECC profiling	+	CPU lockstep, LBIST
	RAM4	Address and control parity	++	CPU lockstep, LBIST, PBIST
	RAM5	Redundant address decode	++	ECC, PBIST
	RAM6	Data and ECC storage in multiple physical banks	M	ECC, autocoverage, PBIST
	RAM7A	Boot time PBIST check of SRAM	++	PBIST autocoverage
	RAM7B	Periodic PBIST check of SRAM	O	PBIST autocoverage
	RAM8	Bit multiplexing in SRAM array	M	ECC
	RAM9	Periodic hardware CRC check of SRAM contents	O	CRC autocoverage
	RAM10	Periodic software readback of static configuration registers	+	CRC autocoverage
	RAM11	Software readback of written configuration	++	CPU lockstep
	RAM12	Software test of SRAM wrapper redundant address decode and ECC	++	CPU lockstep
Level 2 and Level 3 (L2 and L3) Interconnect	INC1	Error trapping (including peripheral slave error trapping)	M	Software test of basic functionality and error response
	INC2	PCR access management	++	Software test of basic functionality and error response
	INC3A	Internal watchdog - DWD	O	External watchdog, software test of watchdog configuration and error response
	INC3B	Internal watchdog - DWWD	+	External watchdog, software test of watchdog configuration and error response
	INC3C	External watchdog	++	Internal watchdog, software test of watchdog configuration and error response
	INC4	Information redundancy	+	CPU lockstep
	INC5	Periodic software readback of static configuration registers	+	CPU lockstep
	INC6A	Boot time software test of basic functionality including error tests	++	CPU lockstep
	INC6B	Periodic software test of basic functionality including error tests	+	CPU lockstep
	INC7	Software readback of written configuration	++	CPU lockstep
EFuse Static Configuration	EFU1	Boot time autoload self-test	M	Self-test autocoverage
	EFU2	E-fuse ECC	M	Autoload self-test
	EFU3	Periodic software readback of static configuration registers	+	CPU lockstep
	EFU4	Software readback of written configuration	++	CPU lockstep
One Time Programmable (OTP) Flash Static Configuration	OTP1	Boot time autoload self-test	M	Self-test autocoverage
	OTP2	OTP ECC	M	Autoload self-test
	OTP3	Periodic software readback of static configuration registers	+	CPU lockstep
	OTP4	Software readback of written configuration	++	CPU lockstep

Table 2. Summary of Safety Features and Diagnostics (continued)

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Feature Recommendation	Possible ISO 26262:2011 Latent Diagnostics
Input/Output (I/O) Multiplexing (IOMM)	IOM1	Locking control registers	++	Software test of basic functionality and error response
	IOM2	Master ID filtering	M	Software test of basic functionality and error response
	IOM3	Error trapping	M	Software test of basic functionality and error response
	IOM4	Periodic software readback of static configuration registers	+	CPU lockstep
	IOM5A	Boot time software test of function using peripherals with analog I/O loopback including error tests	++	CPU lockstep
	IOM5B	Periodic software test of function using peripherals with analog I/O loopback including error tests	O	CPU lockstep
	IOM6	Software readback of written configuration	++	CPU lockstep
Vectored Interrupt Module (VIM)	VIM1	VIM SRAM Data Parity	++	PBIST
	VIM2A	Boot time PBIST check of VIM SRAM	++	PBIST autocoverage
	VIM2B	Periodic PBIST check of VIM SRAM	O	PBIST autocoverage
	VIM3	Bit multiplexing in VIM SRAM array	M	PBIST, parity
	VIM4	Periodic hardware CRC check of VIM SRAM contents	+	PBIST
	VIM5A	Boot time software test of VIM functionality including error tests	++	CPU lockstep
	VIM5B	Periodic software test of VIM functionality including error tests	++	CPU lockstep
	VIM6	Periodic software readback of static configuration registers	+	CPU lockstep
	VIM7	Software readback of written configuration	++	CPU lockstep
	VIM8A	Internal watchdog - DWD	O	External watchdog, software test of watchdog configuration and error response
	VIM8B	Internal watchdog - DWWD	+	External watchdog, software test of watchdog configuration and error response
	VIM8C	External watchdog	++	Internal watchdog, software test of watchdog configuration and error response
Real Time Interrupt (RTI) Operating System Timer	RTI1	1oo2 software voting using secondary free running counter	+	CPU lockstep, PMU cycle counter
	RTI2A	Internal watchdog - DWD	O	External watchdog, software test of watchdog configuration and error response
	RTI2B	Internal watchdog - DWWD	+	External watchdog, software test of watchdog configuration and error response
	RTI2C	External watchdog	++	Internal watchdog, software test of watchdog configuration and error response
	RTI3	Periodic software readback of static configuration registers	+	CPU lockstep
	RTI4	Software readback of written configuration	++	CPU lockstep

Table 2. Summary of Safety Features and Diagnostics (continued)

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Feature Recommendation	Possible ISO 26262:2011 Latent Diagnostics
Direct Memory Access (DMA)	DMA1	Memory protection unit for bus master accesses	++	Software test of DMA MPU configuration and error response
	DMA2	Non-privileged bus master access	M	Software test of bus master function and error response
	DMA3	Information redundancy	+	CPU lockstep
	DMA4	DMA SRAM Data Parity	++	PBIST
	DMA5A	Boot time PBIST check of DMA SRAM	++	PBIST autocoverage
	DMA5B	Periodic PBIST check of DMA SRAM	O	PBIST autocoverage
	DMA6	Bit multiplexing in DMA SRAM array	M	PBIST, parity
	DMA7	Periodic hardware CRC check of DMA SRAM contents	+	PBIST
	DMA8	Periodic software readback of static configuration registers	+	CPU lockstep
	DMA9A	Boot time software test of basic functionality including error tests	++	CPU lockstep
	DMA9B	Periodic software test of basic functionality including error tests	+	CPU lockstep
	DMA10	Software readback of written configuration	++	CPU lockstep
High-End Timer (N2HET) Including HET Transfer Unit (HTU)	HET1	Memory protection unit for bus master accesses	++	Software test of HTU MPU configuration and error response
	HET2	Information redundancy	++	CPU lockstep
	HET3	Use of DCC as program sequence watchdog	++	DCC autocoverage
	HET4	Monitoring by second N2HET	+	CPU lockstep
	HET5A	Boot time software test of function using I/O loopback	++	CPU lockstep
	HET5B	Periodic software test of function using I/O loopback	O	CPU lockstep
	HET6	N2HET/HTU SRAM Data Parity	++	PBIST
	HET7A	Boot time PBIST check of N2HET/HTU SRAM	++	PBIST autocoverage
	HET7B	Periodic PBIST check of N2HET/HTU SRAM	O	PBIST autocoverage
	HET8	Bit multiplexing in N2HET/HTU SRAM array	M	PBIST, parity
	HET9	Periodic hardware CRC check of N2HET/HTU SRAM contents	O	PBIST
	HET10	Periodic software readback of static configuration registers	+	CPU lockstep
	HET11	Software readback of written configuration	++	CPU lockstep
Multi-Buffered Analog to Digital Converter (MibADC)	ADC1	Boot time input self test	++	CPU lockstep
	ADC2A	Boot time converter calibration	++	CPU lockstep
	ADC2B	Periodic converter calibration	O	CPU lockstep
	ADC3	Information redundancy techniques	++	CPU lockstep, ADC converter calibration, ADC self-test
	ADC4	MibADC SRAM Data Parity	++	PBIST
	ADC5A	Boot time PBIST check of MibADC SRAM	++	PBIST autocoverage
	ADC5B	Periodic PBIST check of MibADC SRAM	O	PBIST autocoverage
	ADC6	Bit multiplexing in MibADC SRAM array	M	PBIST, parity
	ADC7	Periodic hardware CRC check of MibADC SRAM contents	O	PBIST
	ADC8	Periodic software readback of static configuration registers	+	CPU lockstep
	ADC9	Software readback of written configuration	++	CPU lockstep

Table 2. Summary of Safety Features and Diagnostics (continued)

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Feature Recommendation	Possible ISO 26262:2011 Latent Diagnostics
Multi-Buffered Serial Peripheral Interface (MibSPI)	MSP1A	Boot time software test of function using I/O loopback	++	CPU lockstep
	MSP1B	Periodic software test of function using I/O loopback	O	CPU lockstep
	MSP2	Parity in message	++	CPU lockstep
	MSP3	Information redundancy techniques	++	CPU lockstep
	MSP4	MibSPI SRAM Data Parity	++	PBIST
	MSP5A	Boot time PBIST check of MibSPI SRAM	++	PBIST autocoverage
	MSP5B	Periodic PBIST check of MibSPI SRAM	O	PBIST autocoverage
	MSP6	Bit multiplexing in MibSPI SRAM array	M	PBIST, parity
	MSP7	Periodic hardware CRC check of MibSPI SRAM contents	O	PBIST
	MSP8	Periodic software readback of static configuration registers	+	CPU lockstep
	MSP9	Software readback of written configuration	++	CPU lockstep
Serial Peripheral Interface (SPI)	SPI1A	Boot time software test of function using I/O loopback	++	CPU lockstep
	SPI1B	Periodic software test of function using I/O loopback	O	CPU lockstep
	SPI2	Parity in message	++	CPU lockstep
	SPI3	Information redundancy techniques	++	CPU lockstep
	SPI4	Periodic software readback of static configuration registers	+	CPU lockstep
	SPI5	Software readback of written configuration	++	CPU lockstep
Inter-Integrated Circuit (I2C)	IIC1A	Boot time software test of function	++	CPU lockstep
	IIC1B	Periodic software test of function	O	CPU lockstep
	IIC2	Information redundancy techniques	++	CPU lockstep
	IIC3	Periodic software readback of static configuration registers	+	CPU lockstep
	IIC4	Software readback of written configuration	++	CPU lockstep
Serial Communications Interface (SCI)	SCI1A	Boot time software test of function using I/O loopback	++	CPU lockstep
	SCI1B	Periodic software test of function using I/O loopback	O	CPU lockstep
	SCI2	Information redundancy techniques	++	CPU lockstep
	SCI3	Periodic software readback of static configuration registers	+	CPU lockstep
	SCI4	Software readback of written configuration	++	CPU lockstep
Local Interconnect Network (LIN)	LIN1A	Boot time software test of function using I/O loopback	++	CPU lockstep
	LIN1B	Periodic software test of function using I/O loopback	O	CPU lockstep
	LIN2	Information redundancy techniques including end to end safing	++	CPU lockstep
	LIN3	Periodic software readback of static configuration registers	+	CPU lockstep
	LIN4	Software readback of written configuration	++	CPU lockstep

Table 2. Summary of Safety Features and Diagnostics (continued)

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Feature Recommendation	Possible ISO 26262:2011 Latent Diagnostics
Controller Area Network (DCAN)	CAN1A	Boot time software test of function using I/O loopback	++	CPU lockstep
	CAN1B	Periodic software test of function using I/O loopback	O	CPU lockstep
	CAN2	Information redundancy techniques including end to end safing	++	CPU lockstep
	CAN3	DCAN SRAM Data Parity	++	PBIST
	CAN4A	Boot time PBIST check of DCAN SRAM	++	PBIST autocoverage
	CAN4B	Periodic PBIST check of DCAN SRAM	O	PBIST autocoverage
	CAN5	Bit multiplexing in DCAN SRAM array	M	PBIST, parity
	CAN6	Periodic hardware CRC check of DCAN SRAM contents	O	PBIST
	CAN7	Periodic software readback of static configuration registers	+	CPU lockstep
	CAN8	Software readback of written configuration	++	CPU lockstep
FlexRay Including FlexRay Transfer Unit (FTU)	FRY1	Memory protection unit for bus master accesses	++	Software test of DMA MPU configuration and error response
	FRY2	Non-privileged bus master access	M	Software test of bus master function and error response
	FRY3A	Boot time software test of function using I/O loopback in PHY	++	CPU lockstep
	FRY3B	Periodic software test of function using I/O loopback in transceiver	O	CPU lockstep
	FRY4	Information redundancy techniques including end to end safing	++	CPU lockstep
	FRY5	1oo2 Voting using Both FlexRay Channels	+	CPU lockstep
	FRY6	FlexRay and FTU SRAM Data Parity	++	PBIST
	FRY7A	Boot time PBIST check of FlexRay and FTU SRAM	++	PBIST autocoverage
	FRY7B	Periodic PBIST check of FlexRay and FTU SRAM	O	PBIST autocoverage
	FRY8	Bit multiplexing in FlexRay and FTU SRAM array	M	PBIST, parity
	FRY9	Periodic hardware CRC check of FlexRay and FTU SRAM contents	O	PBIST
	FRY10	Periodic software readback of static configuration registers	+	CPU lockstep
	FRY11	Software readback of written configuration	++	CPU lockstep
General Purpose Input/Output (GIO)	GIO1A	Boot time software test of function using I/O checking	++	CPU lockstep
	GIO1B	Periodic software test of function using I/O checking	O	CPU lockstep
	GIO2	Information redundancy techniques	++	CPU lockstep
	GIO3	Periodic software readback of static configuration registers	+	CPU lockstep
	GIO4	Software readback of written configuration	++	CPU lockstep

Table 2. Summary of Safety Features and Diagnostics (continued)

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Feature Recommendation	Possible ISO 26262:2011 Latent Diagnostics
Ethernet	ETH1	Non-privileged bus master access	M	Software test of bus master function and error response
	ETH2A	Boot time software test of function using I/O loopback in PHY	++	CPU lockstep
	ETH2B	Periodic software test of function using I/O loopback in PHY	O	CPU lockstep
	ETH3	Information redundancy techniques including end to end safing	++	CPU lockstep
	ETH4A	Boot time PBIST check of Ethernet SRAM	++	PBIST autocoverage
	ETH4B	Periodic PBIST check of Ethernet SRAM	O	PBIST autocoverage
	ETH5	Bit multiplexing in Ethernet SRAM array	M	PBIST, parity
	ETH6	Periodic hardware CRC check of Ethernet SRAM contents	O	PBIST
	ETH7	Periodic software readback of static configuration registers	+	CPU lockstep
	ETH8	Software readback of written configuration	++	CPU lockstep
External Memory Interface (EMIF)	EMF1	Information redundancy techniques	++	CPU lockstep
	EMF2A	Boot time hardware CRC check of external memory	++	CRC autocoverage
	EMF2B	Periodic hardware CRC check of external memory	O	CRC autocoverage
	EMF3	Periodic software readback of static configuration registers	+	CPU lockstep
	EMF4	Software readback of written configuration	++	CPU lockstep
Joint Technical Action Group (JTAG) Debug/Trace/ Calibration Access	JTG1	Hardware disable of JTAG port	+	Watchdog detection of inadvertent activation that impacts program flow
	JTG2	Lockout of JTAG access using AJSM	++	Watchdog detection of inadvertent activation that impacts program flow
	JTG3A	Internal watchdog - DWD	O	External watchdog, software test of watchdog configuration and error response
	JTG3B	Internal watchdog - DWWD	+	External watchdog, software test of watchdog configuration and error response
	JTG3C	External watchdog	++	Internal watchdog, software test of watchdog configuration and error response
Cortex-R4F Central Processing Unit (CPU) Debug and Trace	DBG1	Hardware disable of JTAG port	+	Watchdog detection of inadvertent activation that impacts program flow
	DBG2	Lockout of JTAG access using AJSM	++	Watchdog detection of inadvertent activation that impacts program flow
	DBG3	Use MPUs to block access to memory-mapped debug	++	Watchdog detection of inadvertent activation that impacts program flow
	DBG4	Use of CoreSight debug logic key enable scheme	++	Watchdog detection of inadvertent activation that impacts program flow
	DBG5A	Internal watchdog - DWD	O	External watchdog, software test of watchdog configuration and error response
	DBG5B	Internal watchdog - DWWD	+	External watchdog, software test of watchdog configuration and error response
	DBG5C	External watchdog	++	Internal watchdog, software test of watchdog configuration and error response

Table 2. Summary of Safety Features and Diagnostics (continued)

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Feature Recommendation	Possible ISO 26262:2011 Latent Diagnostics
Data Modification Module (DMM)	DMM1	Hardware disable of JTAG port	+	Detection of inadvertent activation that impacts program flow
	DMM2	Lockout of JTAG access using AJSM	++	Watchdog detection of inadvertent activation that impacts program flow
	DMM3	Use MPUs to block access to memory-mapped debug	++	Watchdog detection of inadvertent activation that impacts program flow
	DMM4	Disable DMM pin interface	++	Watchdog detection of inadvertent activation that impacts program flow
	DMM5A	Internal watchdog - DWD	O	External watchdog, software test of watchdog configuration and error response
	DMM5B	Internal watchdog - DWWD	+	External watchdog, software test of watchdog configuration and error response
	DMM5C	External watchdog	++	Internal watchdog, software test of watchdog configuration and error response
RAM Trace Port (RTP)	RTP1	Hardware disable of JTAG port	+	Watchdog detection of inadvertent activation that impacts program flow
	RTP2	Lockout of JTAG access using AJSM	++	Watchdog detection of inadvertent activation that impacts program flow
	RTP3	Use MPUs to block access to memory-mapped debug	++	Watchdog detection of inadvertent activation that impacts program flow
	RTP4	Disable RTP pin interface	++	Watchdog detection of inadvertent activation that impacts program flow
	RTP5A	Internal watchdog - DWD	O	External watchdog, software test of watchdog configuration and error response
	RTP5B	Internal watchdog - DWWD	+	External watchdog, software test of watchdog configuration and error response
	RTP5C	External watchdog	++	Internal watchdog, software test of watchdog configuration and error response

Table 2. Summary of Safety Features and Diagnostics (continued)

Device Partition	Unique Identifier	Safety Feature or Diagnostic	Feature Recommendation	Possible ISO 26262:2011 Latent Diagnostics
Parameter Overlay Module (POM)	POM1	Hardware disable of JTAG port	+	Watchdog detection of inadvertent activation that impacts program flow
	POM2	Lockout of JTAG access using AJSM	++	Watchdog detection of inadvertent activation that impacts program flow
	POM3	Use MPUs to block access to memory-mapped debug	++	Watchdog detection of inadvertent activation that impacts program flow
	POM4	Use of CoreSight debug logic key enable scheme	++	Watchdog detection of inadvertent activation that impacts program flow
	POM5A	Internal watchdog - DWD	O	External watchdog, software test of watchdog configuration and error response
	POM5B	Internal watchdog - DWWD	+	External watchdog, software test of watchdog configuration and error response
	POM5C	External watchdog	++	Internal watchdog, software test of watchdog configuration and error response

Appendix B 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 61508:2010 and ISO 26262:2011 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 9 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 9. 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 3. Activities Performed by TI vs. Performed by SEooC Customer

Safety Lifecycle Activity	TI Execution	SEooC Customer Execution
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	Yes
Allocation of end equipment requirements to sub-systems, hardware components, and software components	Assumptions made	Yes
Definition of MCU safety requirements	Yes	No
MCU architecture and design execution	Yes	No
MCU level safety analysis	Yes	No
MCU level verification and validation	Yes	No
Integration of MCU into end equipment	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 4 summarizes the product safety documentation that TI can provide to customers to assist in development of safety systems.

Table 4. 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, including system level assumptions of use	Public, no NDA required	Available
Safety Analysis Report Summary for TMS570LS31x and TMS570LS21x ARM Safety Critical Microcontrollers (SPNU521)	Summary of FIT rates and device safety metrics according to ISO 26262 and/or IEC 61508 at device level.	NDA required	Available

Table 4. Product Safety Documentation (continued)

Deliverable Name	Contents	Confidentiality	Availability
Detailed Safety Analysis Report for TMS570LS31x and TMS570LS21x ARM Safety Critical Microcontrollers (SPNU523)	Full results of all available safety analysis documented in a format that allows computation of custom metrics	NDA required	Available
Safety Case 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

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. The summary of individual persons involved in each activity, their job roles, proof of competence, and so forth is captured as required per standard in the Safety Report.

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 TMS570LS31x and TMS570LS21x ARM Safety Critical Microcontrollers* (SPNU521) and the *Detailed Safety Analysis Report for TMS570LS31x and TMS570LS21x ARM Safety Critical Microcontrollers* (SPNU523) 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 5.

Table 5. 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
Safety Analysis Report Summary for TMS570LS31x and TMS570LS21x ARM Safety Critical Microcontrollers (SPNU521)	XML, Microsoft Excel®	Adobe PDF
Detailed Safety Analysis Report for TMS570LS31x and TMS570LS21x ARM Safety Critical Microcontrollers (SPNU523)	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.

Appendix C Revision History

This document has been revised from SPNU511 to SPNU511A because of the following technical change(s).

Table 6. SPNU511A Revisions

Location	Additions, Deletes, and Edits
Section 1	Updated description of safety documents to reflect split of safety analysis report and safety case report into multiple docs
Throughout document	Updated references to ISO 26262 to reflect IS version released in November 2011 rather than FDIS version
Appendix B	Removed section 3.5; relevant data became new Appendix B - DIA
Section 5	Updated to align with the Safety Analysis Report Summary for TMS570LS31x/21x Hercules ARM Safety Critical Microcontrollers (SPNU521) and the 3) Safety Analysis Report Summary for RM48x Hercules ARM Safety Critical Microcontrollers (SPNU522)
Section 5.2 - Section 5.4	Added recommendations to PMM, Clock, and Reset
Section 5.6	Created new section for ESM
Section 5.7 - Section 5.11	Added recommendations to CPU, Flash, FEE, SRAM, L2/L3
Section 5.14 - Section 5.15	Added recommendations to IOMM, VIM
Section 5.17 - Section 5.18	Added recommendations to DMA, N2HET
Section 5.20 - Section 5.28	Added recommendations to MibSPI, SPI, I2C, SCI, LIN, DCAN, FlexRay, GIO, Ethernet
Section 5.31 - Section 5.34	Added recommendations to CPU debug/trace, DMM, RTP, ROM
Appendix A	New section added to summarize recommendations from Section 5
Appendix B	New section for generic SEooC DIA, used previous version section 3.5 as starting point
Appendix C	New section added for revision history

This document has been revised from SPNU511A to SPNU511B because of the following technical change(s).

Table 7. SPNU511B Revisions

Location	Additions, Deletes, and Edits
Throughout document	Corrected literature number references
Throughout document	Split document to separate versions to support TMS570LS31x/21x and RM48x products
Throughout document	Clarified that latent fault definition is based on ISO 26262:2011 definition of latent fault
Throughout document	Changed references from "TI system basis chip" to "TI TPS6538x"
Throughout document	Replaced all references to "RAM" with references to "SRAM" to harmonize document.
Throughout document	Corrected references to ISO 26262-10 to reflect IS version released in late 2012
Throughout document	Harmonized diagnostic entry titles for "Internal or External Watchdog"
Section 1	Updated description of safety documents to reflect current documentation plans.
Section 2	Typographical error - replace "flash" with "that".
Section 3.2	Included exida Certification data on TMS570LS20x/10x to IEC 61508:2010
Section 5.2.2	Corrected title and text to reflect that multi-bit keys are present for configuration registers rather than a multi-register write sequence for module configuration.
Section 5.2.5	Extracted text on PSCON lockstep compare self-test as separate diagnostic for easier tracing. Resulted in changed section numbering for existing sections
Section 5.3.5	Added caution to avoid changing DWD clock source after DWD is enabled to avoid unpredictable behavior.
Section 5.7.8	Extracted text on CPU lockstep compare (CCM) self-test as separate diagnostic for easier tracing. Resulted in changed section numbering for existing sections
Section 5.11.1	Updated section name and clarified peripheral error trapping.
Section 5.10.12	Introduced recommendation for test of SRAM wrapper address decode diagnostic. Resulted in changed numbering for subsequent notes section

Table 7. SPNU511B Revisions (continued)

Location	Additions, Deletes, and Edits
Section 5.11.6	Updated title and description of test for basic functionality of interconnect based on customer feedback.
Section 5.12.3	Added periodic diagnostic on Efuse logic configuration registers.
Section 5.12.4	Added readback of written configuration diagnostic on Efuse logic configuration registers.
Section 5.13.3	Added periodic diagnostic on OTP logic configuration registers.
Section 5.13.4	Added readback of written configuration diagnostic on OTP logic configuration registers. Resulted in changed number for subsequent notes section.
Section 5.15.5	Updated title and description of test for basic functionality of VIM based on customer feedback.
Section 5.16.4	Added readback of written configuration diagnostic on RTI. Resulted in changed number for subsequent notes section.
Section 5.17.9	Updated title and description of test for basic functionality of DMA based on customer feedback.
Section 5.17.10	Added readback of written configuration diagnostic on DMA logic configuration registers.
Section 5.18.12	Updated N2HET/HTU notes based on customer feedback.
Section 5.18.11	Added readback of written configuration diagnostic on N2HET/HTU logic configuration registers. Resulted in changed number for subsequent notes section.
Section 5.19.10	Updated MibADC notes based on customer feedback.
Section 5.19.9	Added readback of written configuration diagnostic on MibADC logic configuration registers. Resulted in changed number for subsequent notes section.
Section 5.20.9	Added readback of written configuration diagnostic on MibSPI logic configuration registers. Resulted in changed number for subsequent notes section.
Section 5.21.2	Replaced "MIBSPI" with "SPI"
Section 5.21.5	Added readback of written configuration diagnostic on SPI logic configuration registers.
Section 5.22.4	Added readback of written configuration diagnostic on I2C logic configuration registers.
Section 5.23.4	Added readback of written configuration diagnostic on SCI logic configuration registers.
Section 5.24.4	Added readback of written configuration diagnostic on LIN logic configuration registers. Resulted in changed number for subsequent notes section.
Section 5.25.8	Added readback of written configuration diagnostic on DCAN logic configuration registers.
Section 5.26.11	Added readback of written configuration diagnostic on FRAY logic configuration registers. Resulted in changed number for subsequent notes section.
Section 5.27.4	Added readback of written configuration diagnostic on GIO logic configuration registers. Resulted in changed number for subsequent notes section.
Section 5.28.8	Added readback of written configuration diagnostic on Ethernet logic configuration registers.
Section 5.29.4	Added readback of written configuration diagnostic on EMIF logic configuration registers. Resulted in changed number for subsequent notes section.
Section 5.30.3	Replaced note on use of watchdog for detection of inadvertent activation with a dedicated diagnostic entry. Resulted in changed section numbering for existing sections
Section 5.31.5	Replaced note on use of watchdog for detection of inadvertent activation with a dedicated diagnostic entry.
Section 5.32.5	Replaced note on use of watchdog for detection of inadvertent activation with a dedicated diagnostic entry.
Section 5.33.5	Replaced note on use of watchdog for detection of inadvertent activation with a dedicated diagnostic entry.
Section 5.34.5	Replaced note on use of watchdog for detection of inadvertent activation with a dedicated diagnostic entry.
Table 2	Updated Table 2 to reflect newly added diagnostics. Split VIM5 to VIM5A/VIM5B. Updated naming of PMM2, INC1, INC6A, INC6B, IOM5A, IOM5B, DMA9A, DMA9B. Corrected typo "OTP2 - Efuse ECC" to "OTP 2 - OTP ECC".
Table 4	Updated description of safety documents to reflect current documentation plans and corrected error in table.
Table 5	Updated description of safety document output formats to reflect current documentation plans and corrected error in table.
Section 6	Added links to the Next Steps section

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		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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