WHITE PAPER

TEXAS INSTRUMENTS

Hercules™ Microcontrollers: Real-time MCUs for safety-critical products

Texas Instruments' authors:

Karl Greb
Functional Safety Technologist

Dev Pradhan
Hercules Product Line Manager

Introduction

Electronics continue to play an ever-increasing role in products whose operation is critical to the preservation of human life, whether on the factory floor, during your morning commute, in the operating theater or in a myriad other locations. Ensuring these products always operate in a "safe" manner and meet the stringent functional safety requirements of standards such as IEC 61508 or ISO 26262 is an arduous task. Robust development processes, thorough hazard and risk analyses, thoughtful system designs and careful selection of hardware and software components are all critical to ensuring functional safety. Further increasing the challenge are aggressive time-to-market demands and cost constraints that can be found even in safety-related systems.

For over 25 years, TI has been a valued partner to customers in the development of safety-critical applications. This experience has allowed TI to develop a unique expertise that combines the state of the art in semiconductor development and functional safety. Diligent work as a member of the ISO 26262 standard international working group, as well as partnerships with experienced functional safety experts have hardened and sharpened TI's functional safety expertise, enabling best-in-class hardware and software development capabilities.

The Hercules™ microcontroller (MCU) platform is the result of TI's long history of developing and shipping MCUs for safety-critical systems, providing a broad portfolio of devices specifically designed to meet the requirements of safety-critical medical, industrial and transportation applications. Robust application of core functional safety techniques and semiconductor industry best practices has resulted in an MCU platform that simplifies safety-critical system design, reduces software overhead and speeds certification, all of which result in lower system cost and faster time-to-market.

The Hercules platform comprises three families of scalable functional safety MCUs (see Figure 1) offering a balance of performance, memory, functional safety capability, peripherals, connectivity and cost:

RM4x Hercules MCUs

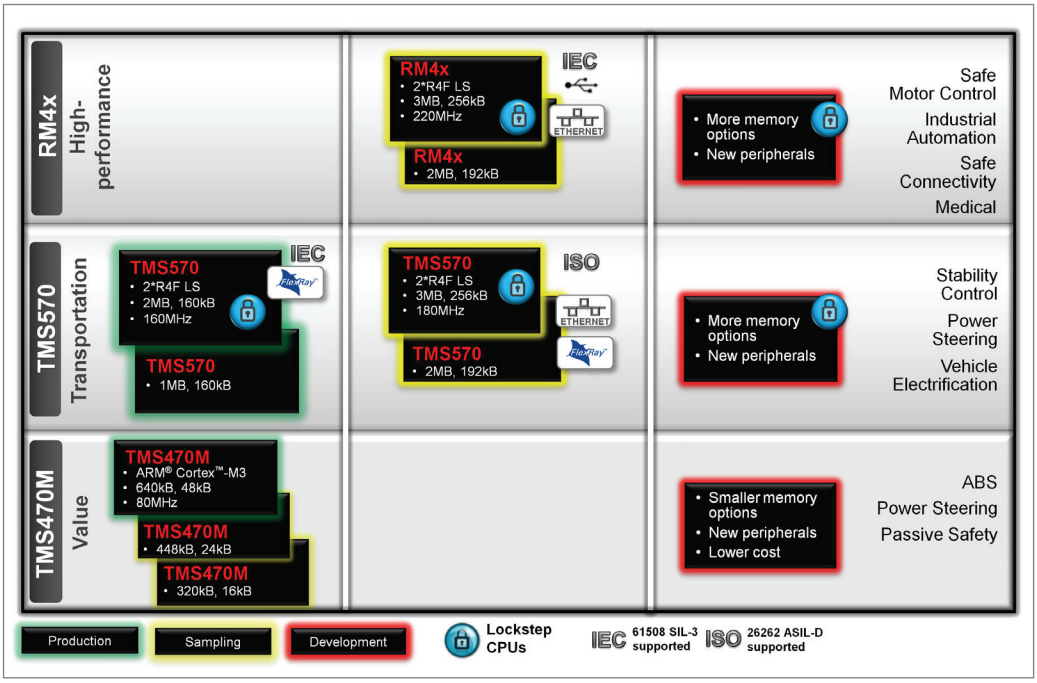
Designed to provide the highest levels of performance and safety for industrial automation, medical instrumentation, servo drive control and networked applications, the RM4x family of MCUs has dual ARM® Cortex™-R4 floating-point CPU cores operating in lockstep and provide up to 220MHz performance with up to 3 MB flash memory and 256 KB RAM. RM4x MCUs are developed in accordance to the IEC 61508 2nd edition standard with SIL-3 capability. These MCUs feature enhanced connectivity options including Ethernet, CAN and USB.

TMS570 Hercules MCUs

Designed to meet the performance and safety needs of transportation applications such as railway, aerospace and automotive systems, the TMS570 MCUs have dual ARM Cortex-R4 floating-point CPU cores operating in lockstep at up to 180MHz performance with up to 3 MB flash memory and 256 KB RAM. These MCUs provide connectivity, Ethernet, CAN and FlexRay™, typically required for transportation applications. Existing 130nm TMS570 MCUs are developed in accordance with the IEC 61508 1st edition standard with SIL-3 capability, while the new 65nm TMS570 MCUs are developed in accordance with the ISO 26262 standard with ASIL-D capability. All TMS570 MCUs are AEC-Q100 qualified for use in automotive applications.

TMS470M Hercules MCUs

Designed to provide cost-efficient safety features for applications requiring less performance, the TMS470M family is based on an ARM Cortex-M3 CPU core operating at up to 80MHz with up to 640 KB flash memory and 48 KB RAM. TMS470M MCUs are also AEC-Q100 qualified for use in automotive applications and support LIN and CAN networking.



▲ *Figure 1: The Hercules platform offers three families of safety MCUs providing a balance of performance, memory, peripherals, connectivity, cost, and scalability for industrial, medical, and transportation safety-critical applications.*

Addressing functional safety: systematic and random faults

State of the art functional safety generally recognizes two categories of faults: systematic and random. These faults become failures when a fault results in a loss of safety function or violates a safety goal. Systematic faults often arise from errors in the processes of development, manufacturing or operation. Examples of systematic faults include failure to verify designed functionality, manufacturing test escapes or operating a product outside of the guaranteed parameters. Faults in software are also considered systematic, as software is fully deterministic and, though challenging, can be formally proven correct before product implementation. Management of systematic faults is achieved via robust processes which include checks and balances on each development activity.

Functional safety standards such as IEC 61508 and ISO 26262 provide a framework for the management of systematic faults. A development process that follows the guidelines of such a standard can mitigate most systematic faults. Quality engineers will immediately notice an overlap with best practices in these standards and in traditional quality management standards. Application of a recognized quality management process, such as ISO/TS 16949 or AEC-Q100, is often considered a minimum capability necessary to start development for functional safety. The development process used by the Hercules safety MCUs has been assessed compliant to the ISO/TS 16949, ISO 9001, and AEC-Q100 quality standards, in addition to targeting IEC 61508 and ISO 26262.

Quality and reliability measures alone cannot guarantee functional safety; measures to manage random faults must also be considered. Random faults are those which are inherent to an application, use case or operating environment when implemented within designed parameters. We typically cannot reduce the inherent failure rate of random faults, so instead we focus on the use of safety mechanisms to detect and manage random faults. Examples of random faults include permanent failure of a hardware circuit, temporary corruption of SRAM data due to soft error or shorting of adjacent signals in an MCU package.

Management of random faults requires safety mechanisms that can detect faults in the targeted application. Mechanisms for functional safety must generally detect faults during normal operation, executing within the control loop of the target system. This puts a premium on periodically executed or continuously operating diagnostics over those which can be executed only at startup or shutdown of a system. As control loop timing becomes tighter, safety mechanisms will trend towards implementation of parallel diagnostics. For example, a system with 100ms control loop timing can generally use more serialized, periodic diagnostics than a 100us control loop system that most likely will require parallel and continuously operating diagnostics. Having safety diagnostics in hardware enables the Hercules safety MCU platform to support safety checks in tight control loops as these checks can be run in parallel.

When implementing safety mechanisms, a developer must consider the possibility that a functional logic and its safety mechanism fail to identify failures caused by faults in commonly shared signals. This phenomenon is known as common mode failure. Common mode failure is often considered when implementing diagnostics with functional duplication, but should be considered for all diagnostics. While it can be difficult, if not impossible, to quantify the probability of common mode failure, there are best practices which can be applied to reduce common mode failure probability. The Hercules safety MCUs have applied many best practices to address the aspect of common mode failure which are described later in this paper.

"Safe island" architecture concept

The RM4x and TMS570 Hercules MCUs share a common safety architecture concept called a "safe island" approach. 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/clock/reset, CPU, flash memory, SRAM and associated interconnect – is needed to guarantee any functionally correct execution of software. Once we have confidence in the correct operation of these elements, we can use software executing on these elements 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.

Safing power, clock, and reset

Any comprehensive safety architecture must start with a solid foundation. Digital logic requires functional power, clock and reset generation to ensure proper operation. Hercules MCUs have multiple layers of hardware safety diagnostics to detect failures of this circuitry, as well as hooks to enable external devices to provide additional layers of diagnostics with reduced probability of common mode failure.

For the majority of Hercules MCUs, power is regulated externally and provided to the core and I/O supplies. An on-chip voltage monitor is implemented primarily to simplify power supply sequencing, but it can also provide a secondary diagnostic on the power supply. The voltage monitor will detect grossly out of range CPU core or I/O voltage and put the device into a reset state when a fault is detected. Primary diagnostic on power is recommended to be implemented off-chip for improved accuracy and reduced probability of common mode failure.

Hercules MCUs have multiple diagnostics to detect failure of the clocking subsystem. Failure of the input clock can be detected via a clock monitor diagnostic that uses an on-chip low power oscillator (LPO) as a reference clock which requires no external components. In case of input clock failure, the device can revert to operation using the LPO as the primary clock input. Internal clocks can also be output for monitoring by an external logic.

In addition to the clock monitor, frequency drift and clock slip can also be detected. The DCC (dual clock compare) diagnostic enables comparison of an on-chip clock versus a selected reference clock. If the logic is operating properly, the two clocks will operate at a fixed ratio; if faulty, the ratio will change from expected and an error will be signaled. The phase lock loop (PLL), which is used to generate the high frequency system clock from a low frequency clock input, has slip detection capabilities.

Faults in derived and distributed clocks must also be considered and can be diagnosed with a windowed watchdog. Unlike most single threshold watchdogs on the market, a windowed watchdog requires a CPU response within a defined time window. Not only does this indicate that the CPU remains operational, but it indicates that the CPU is operating on the correct time base. In a single threshold watchdog, if the watchdog is not reset in time, the system can be assumed to be in a faulty state. However, it is possible that a system could get in a state where the watchdog timer is reset multiple times before the threshold counter expires, thus preventing the faulty state from being detected. A windowed watchdog timer overcomes this issue by requiring the timer to be reset within a defined window of time.

Safing the MCU

The CPU core of the MCU is one of the most complex pieces of logic in the MCU and is involved in almost every on-chip operation. Because of this, the CPU is a natural candidate for the application of a lockstep safety mechanism. The lockstep CPU scheme implements a checker CPU, which is hardwired to be fed the same input as the functional CPU. Two blocks of the same logic, fed the same input, should in theory produce the same output. A core compare module monitors the outputs of the two CPU cores on a cycle-by-cycle basis and signals any errors to the system. This near instant fault detection in the Hercules MCU's comes with little penalty in power consumption and no impact to CPU performance. Also, in comparison to other elements on the MCU, the size overhead of the lockstep mechanism is minimal.

Whenever logic is duplicated, there is always a concern of common mode failure. To combat common mode failure, TI has implemented multiple best practices on the lockstep CPU subsystem. Temporal diversity of the two CPU cores is implemented, such that the CPU cores operate 1.5 or 2 cycles out of phase in order to mitigate common mode failure in clocking. A voltage guard ring is implemented around the CPU cores. Physical design diversity is implemented by flipping and rotating the checker CPU with respect to the functional CPU. The net result is a system that has no documented or reported cases of common mode failure to date.

As effective as a lockstep CPU scheme can be, it cannot detect a failure in circuitry until the circuitry is used. Because of this, TI implements another layer of MCU diagnostics with LBIST (logic built-in self-test) diagnostics. LBIST provides very high diagnostic coverage on the CPU at a transistor level, using the same design for test (DFT) structures used to test the part during manufacturing. LBIST can be initiated via software at startup, shutdown or in small test slices during functional operation. LBIST requires much less execution time than comparable software solutions, while also reducing the memory overhead needed to store software diagnostics. The Hercules LBIST technology is unique in that the hardware tests can be executed in batch at power up or in periodic time slices during normal operation.

Combined with the other integrated safety features of the Hercules architecture, hardware-based BIST reduces software overhead related to safety diagnostics up to 30% (see Figure 2). These overhead savings are typically used by device integrators to implement advanced functionality to further differentiate their products.

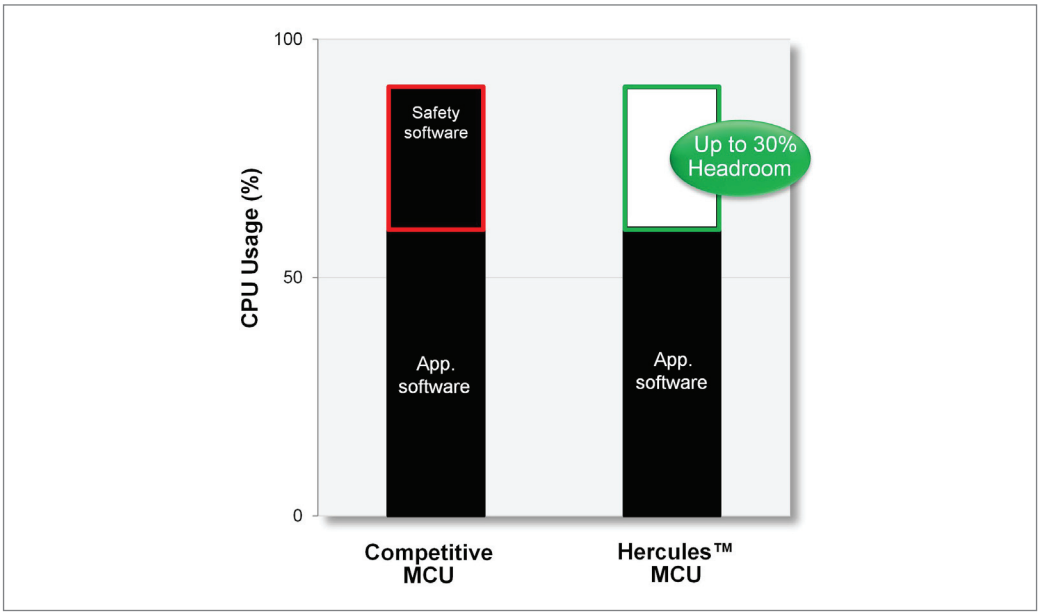


Figure 2: By integrating safety features in hardware, the Hercules architecture reduces software overhead related to safety diagnostics by up to 30%, leaving additional headroom to implement advanced functionality that differentiates products.

Safing the CPU memory

The "safe island" concept includes implementation of hardware-safety mechanisms for the internal flash and SRAM memories. In a typical system, the flash is used as the CPU instruction memory and the SRAM provides CPU data memory. Both memories, and the connection between these memories and the CPU, must be functioning properly to ensure correct operation of software. The primary run-time diagnostic for the SRAM and flash is ECC (error correcting code) logic on data augmented by address and control parity.

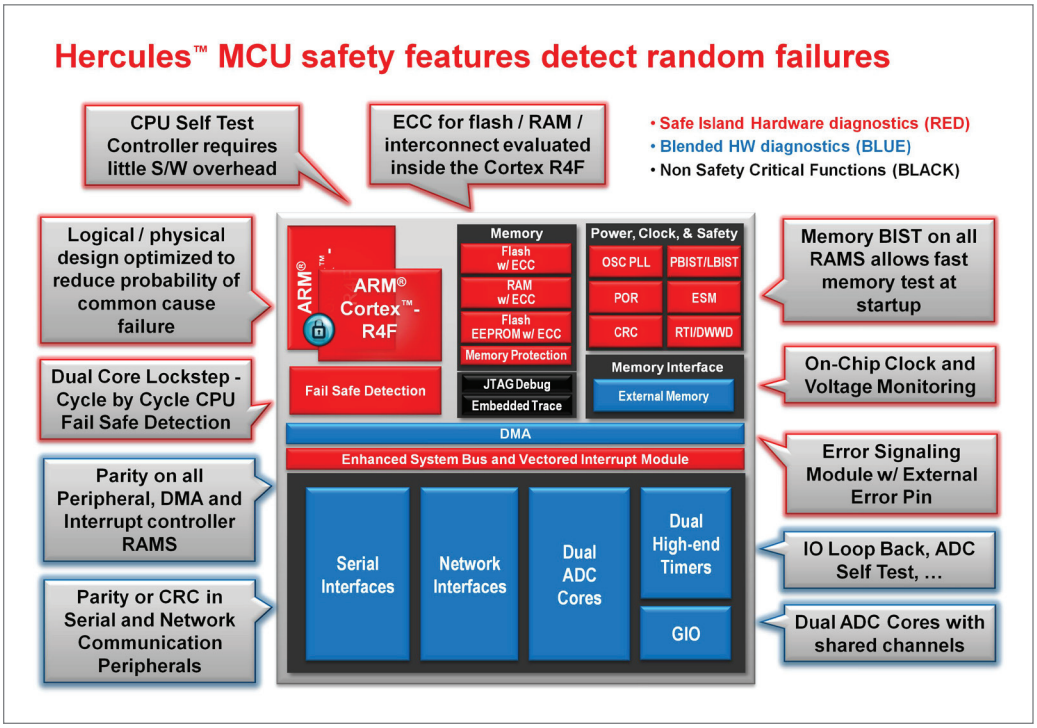
ECC technology is a well-accepted and effective way to detect random faults. ECC encodes data in a way that enables systems to not only detect when memory has been corrupted, but also correct single-bit errors if desired so system operation can continue uninterrupted.

Traditionally, the ECC controller is integrated into the memory block itself. When a memory read is initiated, the data is retrieved and the ECC controller confirms that the memory has not been corrupted. The data is then sent out over the memory bus to the CPU for use. One drawback with this approach is that data is vulnerable while it is on the memory bus; if a random fault alters the data as it is being transferred in either direction, the CPU will not detect this. Another limitation is that each memory subsystem must have its own ECC controller.

TI addresses these issues by moving the ECC controller into the CPU. When writing to memory, data is encoded before it is put on the memory bus. When reading, data integrity is checked once it has been retrieved by the CPU. In this way, the integrity of data is checked for the entire time it is not being stored; any random faults that have occurred while data is on the memory bus or in memory will be detected. Due to the tight coupling of the ECC control logic in the CPU, there is no overhead incurred for this safety mechanism (i.e. zero impact on CPU performance or latency).

The Hercules ECC controller also gives developers the flexibility to manage the level of responsiveness to single-bit memory errors. While a certain number of single-bit errors can be expected to occur over the lifetime operation of a product, it may be the case that a system regularly experiencing single-bit errors is at high risk for a future unrecoverable system failure. To detect such a condition, developers have the option of logging single-bit errors and setting a threshold to trigger a system-level alert if a critical number of errors occur.

Similar to the CPU, a boot time BIST engine is used to provide additional test coverage beyond what is checked in run-time. TI's proprietary PBIST (programmable memory BIST) provides very high transistor-level diagnostic coverage on all SRAM memories on the device, including those implemented in peripheral subsystems. PBIST is typically executed at start-up or shut-down because the tests are by nature destructive to memory contents.



▲ *Figure 3: The Hercules safety MCU architecture implements hardware-based safety features to create a "Safe Island" from which faults in the rest of the system can be detected.*

Safing peripherals

Safety mechanisms on the Hercules MCUs are not limited to logic in the "safe island." Hardware safety mechanisms are allocated to peripherals for management of random faults that are not easily captured by software. Examples include parity on peripheral SRAMs to check data integrity, I/O loopback capabilities to check the input/output paths from peripherals to I/O pads, PBIST of peripheral memories, memory protection units on all bus masters and limitation of control for critical configuration registers based on CPU privilege level. In addition, the Hercules MCUs have a rich peripheral set which includes multiple instances of several peripherals, such as Analog to Digital converters (ADCs) and the high-end timer modules. This allows customers to implement multiple channels of input or output on an application basis.

Managing error response

In the event that a failure of any MCU component does occur, a system event is triggered through the MCU's error signaling module (ESM). This module provides a programmable error response based on error severity. Potential responses include generation of a CPU interrupt, CPU non-maskable interrupt and/or error pin response. The error pin response allows notification to an external observer that the MCU is in a faulty state. An external observer can then perform system-level mitigation as required and critical errors logged in the ESM will remain latched through a warm reset, enabling the possibility for post-reset diagnostics by the CPU.

Capturing operating history

In some applications, it is desirable to log events from the microcontroller for future failure analysis, similar to the "black box" recorders used in aviation. Select Hercules devices support dedicated, non-intrusive code execution and data trace. The output from these trace interfaces can be captured by external monitors for system-level logging. In addition, many Hercules MCUs include emulated EEPROM memory which can be reprogrammed in-system to log critical faults for later failure analysis. If a system failure does occur, developers will have access to what the application was doing moments before the failure.

Accelerating development and certification

The Hercules development platform provides the detailed functional safety information needed to support customer system certification to common functional safety standards. TI provides a comprehensive functional safety manual and safety analysis report targeted at either IEC 61508 2nd edition or ISO 26262 standards.

In addition to materials to facilitate certification, TI provides all of the design tools, utilities, documentation, development kits and support required to facilitate a fast – and safe – design flow, including:

Hercules development kits (HDK)

TI offers a low-cost USB-based evaluation kit as well as development board for each of its Hercules MCU families.

Real-time operating system (RTOS)

Several of TI's operating system partners have ported their RTOSes to the Hercules platform to provide reliable multitasking and management of real-time deadlines.

Integrated development environment (IDE)

The Hercules platform is supported by several leading IDEs from ARM tools partners, as well as TI's Code Composer Studio™, which is included free with each of the Hercules development kits.

HALCoGen

This GUI-based tool generates the low-level drivers for all of the MCUs peripherals. In addition to simplifying peripheral configuration, HALCoGen provides drivers that are optimized for both performance and flash footprint.

nowECC

This tool automatically generates the ECC values for code and data tables.

nowFlash

This tool provides both a GUI and command-line interface for programming the flash memory of the Hercules MCUs.

Phase lock loop (PLL) calculator

The PLL module integrated into the Hercules architecture is powerful but can be complex to program. This utility simplifies configuration of the PLL module by allowing developers to define the frequencies they need and then calculating the appropriate PLL values.

High-end timer IDE

This GUI-based utility simplifies the programming of the timer module, which is used for various timing-related functions including PWM signal generation, frequency/pulse measurements, etc. The tool includes a simulator as well as a waveform display to help developers to quickly define complex timing signals.

Safety developer network

Many of TI's partners also have experience developing safety-critical applications and understand the special design requirements that comprise such systems. Resources are available to facilitate safe design, evaluate system architectures, take products through safety assessment, help qualify an existing code base and provide safety certified software, among others. Developers can find support from within TI's extensive developers network, as well as 24/7 support from TI's E2E Community.

The Hercules safety microcontroller platform with the RM4x, TMS570 and TMS470M families gives developers the ability to find the best fit processor for their product with many options in performance, flash and RAM memory, connectivity and safety features. Based on the ARM Cortex CPU, the Hercules MCU — with its hardware-based safety features, comprehensive functional safety manual and safety analysis report, provides developers with all the key ingredients they need for their microcontroller to build and certify their safety-critical application. Get started today with an easy to use, inexpensive Hercules development kit.

E042210