Application Report
SPNA212–December 2015

Supporting Functional Safety Using SafeTI™ Diagnostic Library

R. Manoj, Girish Gajwani

ABSTRACT

The application report illustrates the use of the safety library towards enabling diagnostics or tests on diagnostics applicable to the Hercules™ safety microcontrollers and TPS65381 Power Management Integrated Chip (PMIC).

Contents

List of Figures

List of Tables

1 Introduction

Functional safety is part of the overall system or equipment operating in response to its inputs in a predictable manner from a safety perspective. The objective of functional safety is to minimize the likelihood of unacceptable risk of physical injury or damage to health of people, directly or indirectly. Different industries have their own specific standards that they can impose from a customer requirement or mandatory legislative perspective on safety critical systems(a system whose malfunction can result in death or serious injury to people) so as to reduce safety critical risk.

As more and more control systems are based on programmable electronic systems (PES), it is critical to meet functional safety requirements and expectations for PES.

Selection of the right mix of components (such as microcontrollers, power management IC for the design and development of PES) is a very important. A right choice can reduce the efforts needed to meet the functional safety targets.

The TI Hercules Safety MCU family is a high performance MCU family targeting general purpose functional safety applications. Hercules MCUs are also an integral part of many SafeTI functional safety design packages (www.ti.com/safeti). SafeTI design packages help enable compliance with safety standards by including semiconductor components, safety documents, tools and software, complementary embedded processing and analog components, quality manufacturing process and a safety development process. System level management of the external error response can often be simplified through the use of a TI TPS6538x power supply and effective safety companion devices developed for use with the Hercules family.

1.1 Acronyms and Descriptions

Table 1. Acronyms Used in This Document

Acronym	Description
AMUX	Analog MUX
API	Application programmable interface
CRC	Cyclic redundancy check
DWD	Digital watchdog
DMUX	Digital MUX
DWWD	Digital windowed watchdog
ESM	Error signaling module
LBIST	Logic built-in self test
PBIST	Programable built-in self test
PMIC	Power management integrated circuit (IC)
PES	Programmable electronic systems
PESSRAL	Programmable electronic systems in safety-related applications for Lifts
SECDED	Single error correction and double error detection
WDTI	Watchdog trigger input
FEE	Flash emulate EEPROM
ABIST	Analog Built In Self Test
PLL	Phase locked loop

2 Supporting Functional Safety – Hercules MCU and TPS65381-PMIC Using the SafeTI Diagnostic Library

This application report shows a possible mapping of the Hercules Safety MCU diagnostic features to the SafeTI diagnostic Library API, which can be used in an application to do the diagnostics or to provide the test of the diagnostic feature itself.

This application report is an example only. The designs shown are not warranted to comply with any specific functional safety standard, product safety standard, or security standard, a responsibility typically belonging to the OEM. This document cannot be used as a reference design ISO 22201:2009 PERSSRAL.

The Hercules MCU together with TPS65381 PMIC offer safety features in hardware [3]and the software stack(as shown in Figure 1), provides the software API that can efficiently be used in the control systems for detection and management of detected faults in the Hercules Safety MCU. As an example, the Elevator Control System found in Appendix A illustrates the various components in a real work use.

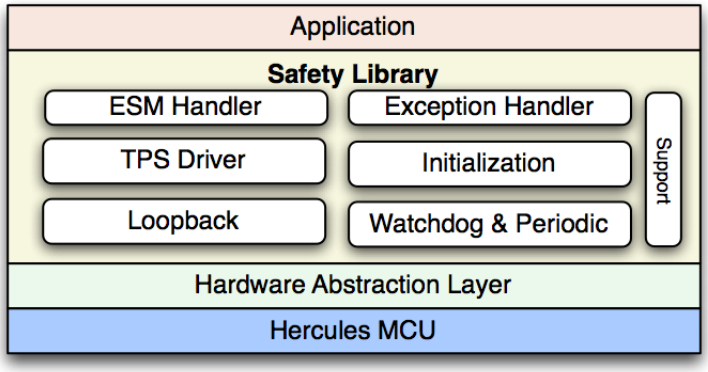

Figure 1. Software Stack With SafeTI Diagnostic Library

2.1 TI Hercules MCU Safety Overview

The Hercules MCU family of 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 considerations. 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, reset, CPU, Flash memory, SRAM and associated interconnect, is needed to help assure any functionally correct execution of software. Once correct operation of these elements is confirmed, software execution can begin on these elements in order to provide software-based diagnostics on other device elements, such as peripherals. The Hercules architecture also provides various safety mechanisms and technical recommendations for the use of safety mechanisms.

All microcontrollers of the Hercules family have an associated Safety Manual [3] that provides information needed to assist in the creation of a safety critical system. 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

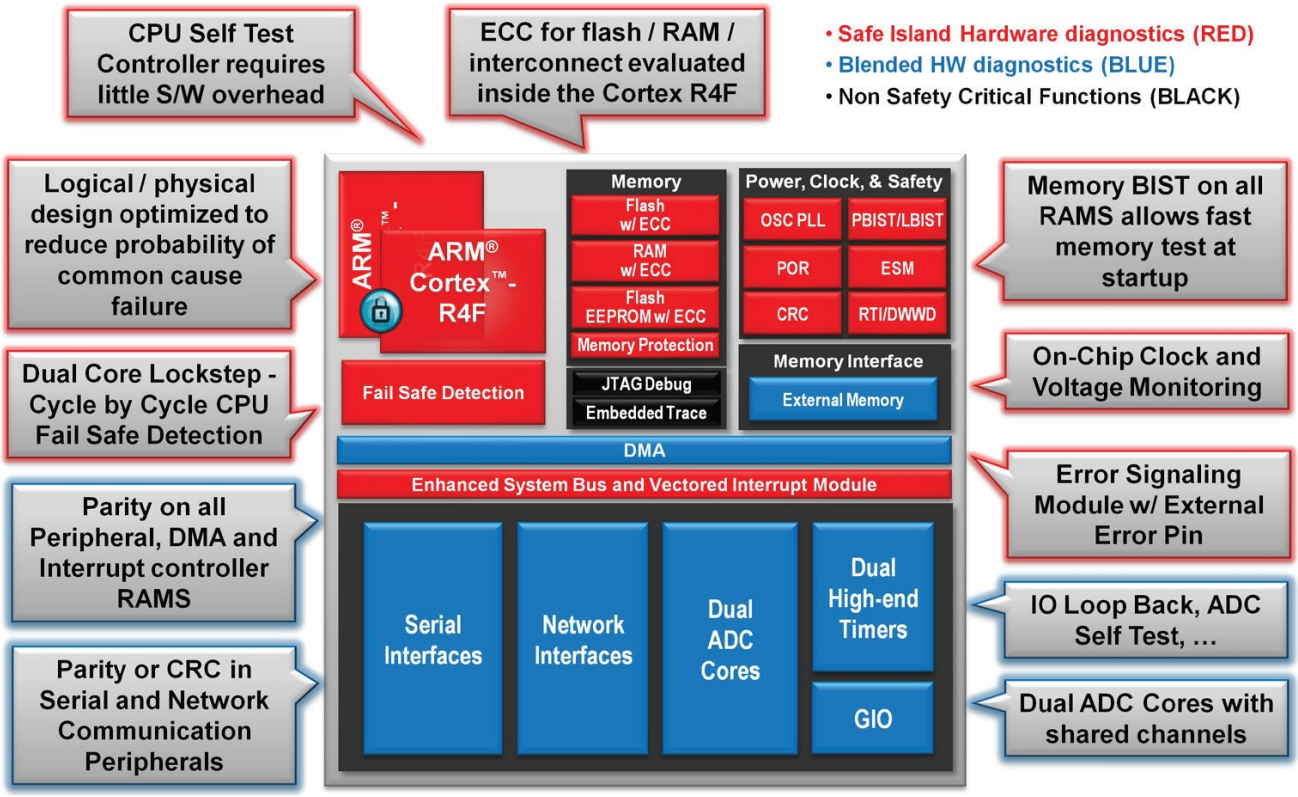

Figure 2. The Hercules Safety MCU Architecture Implements Hardware-Based Safety Features to Create a "Safe Island" From Which Faults in the Rest of the System Can be Detected

2.2 TPS65381 Power Management IC Overview

The TPS65381 [4] is a companion SafeTI chip for the Hercules microcontroller family, that includes certain functional-safety related features. It is a multi-rail power supply designed to supply microcontrollers in safety-critical applications, such as those found in automotive.

The device monitors undervoltage and overvoltage on all regulator outputs, battery voltage, and internal supply rails. A second band-gap reference, independent from the main band-gap reference, monitors for under and overvoltage, to avoid any drifts in the main band-gap reference being undetected. In addition, the device implements regulator current limits and temperature protections. The TPS6538x functional safety architecture features a question-answer watchdog, MCU error-signal monitor, check-mode for MCU error-signal monitor, clock monitoring on internal oscillators, self-check on clock monitor, CRC on non-volatile memory, and a reset circuit for the MCU. A built-in self-test (BIST) allows for monitoring the device functionality at start-up. Figure 3 shows an interfacing of TPS65381 with the Hercules microcontroller in an automotive use case.

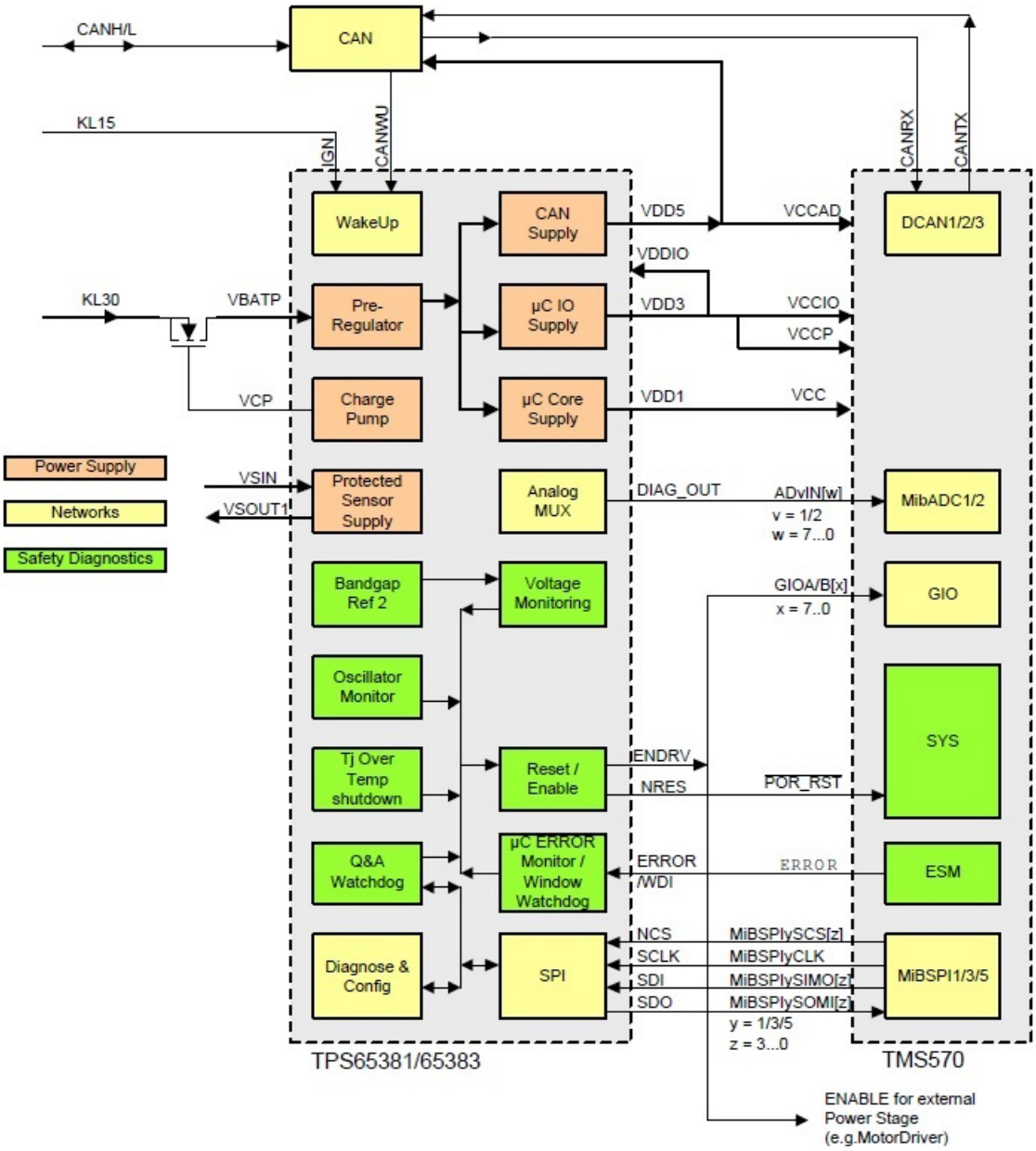

Figure 3. Interfacing of the TPS65381 Device With Hercules Microcontroller

2.3 SafeTI Diagnostic Library and Compliance Support Package (CSP)

The SafeTI Diagnostic Library serves as the software implementation of the safety manual for the microcontroller. It provides interfaces to several safety mechanisms that are described in the safety manual. Based on the final system requirements, the system integrator can use these APIs to incorporate appropriate mechanisms in the final system to meet safety requirements [7].

The Diagnostic library is a collection of functions with access to safety functions and response handlers for various safety mechanisms. The Diagnostic library runs in the context of the caller's protection environment and responses are handled in the context of interrupt or exception. The Diagnostic library also provides the TPS6538x driver package for the Hercules Safety MCU, which provides driver level API to interface the Hercules MCU with TPS6538x power supply chip and make use of the various TPS6538x device features such as voltage monitoring, watchdog monitoring, error monitoring. Table 3 through Table 7 show the example API provided the TPS6538x driver package.

The SafeTI Diagnostic library also has a compliance support package (CSP) release. This release provides the necessary documentation and reports to assist customers using the SafeTI Hercules Diagnostic Library to help meet safety requirements.

3 Safety Measures for PESSRAL

Table 2 is partial illustration only; use this as a guideline when you are developing your system and is not intended to be exhaustive or complete in nature.

Table 2. Safety Measures for PESSRAL

Component and Functional Safety Requirement	Measures to be Taken	Mapping to Hercules Safety Manual [1]	Mapping to Hercules SafeTI Diagnostic Library for test of the diagnostic	API Usage	Comments
Structure: Any system can be divided into three subsystems: an input subsystem, Logic subsystem, and output subsystem. The structure should be such that it can capture any random failure.	Single channel implementation supporting self-test and monitoring using the self-test controller. The other way to satisfy this feature is to use two channels of subsystems, which may not be a cost effective solution.	CPU2A/B	SL_Selftest_STC	This Diagnostic Library API can be used to: • Run CPU LBIST • Verify STC logic by running self-test diagnostics • Fault injection for the STC	The CPU self-test controller (STC) is used to test the ARM-CPU core using the deterministic logic built-in self-test (LBIST) controller as the test engine.
Processing Units: Processing units form the core of the Logic subsystem, and any failure in the processing unit can lead to incorrect results, Such a failure must be detectable.	CPU compare module (CCMR4) for the lockstep architecture in Hercules.	CPU1	SL_SelfTest_ CCMR4F	This SafeTI Diagnostic Library API can be used to: • Verify CCM logic by running self-test diagnostics • Fault injection on: – CCM compare error output signal – CCM self-test error signal	Hercules architecture implements two instances of the Cortex®-R4F CPU that are running in lock step to detect faults that may result in unsafe operating conditions. The CCM-R4F detects faults and signals them to an error signaling module (ESM).
Invariant memories ranges: Invariant memories (like Flash) store important information (like Instruction Opcode for logic execution). A bit error can lead to severe system fault. Any bit flip in the invariant memory must be detectable and correctable. Single bit errors must be correctable and 2-bit failures must be detectable. Sometimes RAM, based on the application, can store static tables.Once initialized they do not change so it can be considered as invariant memory.	Single Error Correction and Double Error Detection (SECDED) for L2FMC and TCRAM, signature verification on any memory subsystem using cyclic redundancy check (CRC)	FLA1	SL_SelfTest_Flash	Capability to run: • Single bit error correction self test • Double bit error detection self test • Fault injection using 2-bit error	The Flash memory can be protected by SECDED. The main program memory is protected by the SECDED circuit inside of the Cortex R4 CPU
		FLA5A/B	SL_CRC_Calculate	Signature verification of the memory contents.	The CRC controller is a module that is used to perform CRC to verify the integrity of the memory system.
		RAM1	SL_SelfTest_SRAM	Capability to run: • Single bit error correction self test • Double bit error detection self- test • Fault injection using 2-bit error	TCRAM interface module features dedicated for SECDED support.
		RAM9	SL_CRC_Calculate	Signature verification of the memory contents.	The CRC controller is a module that is used to perform CRC to verify the integrity of memory system.

Table 2. Safety Measures for PESSRAL (continued)

Component and Functional Safety Requirement	Measures to be Taken	Mapping to Hercules Safety Manual [1]	Mapping to Hercules SafeTI Diagnostic Library for test of the diagnostic	API Usage	Comments
Variable memory ranges: RAM can be considered as one of the variable memories. It may store importation data structures of system like task control blocks (TCB) of an operating system. A bit error can lead to severe system fault. Any bit flip in the variant memory must be detectable and correctable. Single bit errors must be correctable and ODD 2-bit failures must be detectable.	SECDED for L2FMC and TCRAM, signature verification on any memory subsystem using CRC	FEE1	SL_SelfTest_FEE	Capability to run: • Single bit error correction self test • Double bit error detection self test • Fault injection using 2-bit error	The FEE memory can be protected by SECDED. The main program memory is protected by the SECDED circuit inside of the Cortex-R4 CPU.
		FEE2A/B	SL_CRC_Calculate	Signature verification of the memory contents.	The CRC controller is a module that is used to perform CRC to verify the integrity of memory system.
		RAM1	SL_SelfTest_SRAM	Capability to run: • Single bit error correction self test • Double bit error detection self test • Fault injection using 2-bit error	TCRAM interface module features dedicated for SECDED support.
		RAM9	SL_CRC_Calculate	Signature verification of the memory contents	The CRC controller is a module that is used to perform CRC to verify the integrity of the memory system.
		RAM7A/B	SL_SelfTest_PBIST	Executes PBIST tests on RAM groups with different algorithms	The programmable built-in self-test (PBIST) controller architecture provides a run-time-programmable memory BIST engine for varying levels of coverage across many embedded memory instances.
Clock: Many system peripherals like ADC, SPI, and so forth are driven by peripheral clocks that are generated using a main system clock. Failure in clock generation and incorrect clock variations can lead to system failure and must be detectable	Low-power oscillator clock detect	CLK1	ESM_Application_ Callback	• ESM_Application_Callback of the diagnostic library captures ESM errors related to LPOCLK the clock monitor esm error.	The low-power oscillator clock detector (LPOCLKDET) is a safety diagnostic that can be used to detect failure of the primary clock
	PLL Slip Detector	CLK2	ESM_Application_ Callback	• ESM_Application_Callback of the diagnostic library captures ESM errors related to PLL Slip error .	The PLL logic includes an embedded diagnostic that can detect a slip of the PLL output clock. Error response and indication is dependent on the programming of the PLL control registers that are located in the system module.
	Dual Clock Comparator	CLK3	ESM_Application_ Callback	• ESM_Application_Callback of the diagnostic library captures ESM errors related to clock comparison.	The DCC can be used to detect incorrect frequencies and drift between clock sources.
	Watchdog with separate time base	CLK5A/B	N/A	N/A	The Hercules platform supports the use of an internal watchdog that has a separate time base. It has two modes of operation: digital watchdog (DWD) and digital windowed watchdog (DWWD). HALCoGen provides API for configuration and usage of internal watchdogs
	External Watchdog	CLK5C	Watchdog Support API for External Watchdog (see Table 3)	The API supports: • Handling of the complete watchdog configuration • Sending of the watchdog responses to the TPS65381 device • Fault Injection by sending a bad answer	TPS65381 device include a closed-loop digital watchdog.

Table 2. Safety Measures for PESSRAL (continued)

Component and Functional Safety Requirement	Measures to be Taken	Mapping to Hercules Safety Manual [1]	Mapping to Hercules SafeTI Diagnostic Library for test of the diagnostic	API Usage	Comments
Program sequence: Incorrect execution flow of a program or the wrong program sequence must be detectable	Watchdog with separate time base	CLK5B	N/A	N/A	The Hercules platform supports the use of an internal watchdog that has a separate time base. It has two modes of operation: digital watchdog (DWD) and digital windowed watchdog (DWWD). HALCoGen provides API for configuration and usage of internal watchdogs
	External Watchdog	CLK5C	Watchdog support API for external watchdog (see Table 3)	The API supports: • Handling of the complete watchdog configuration • Sending of the watchdog responses to the TPS65381 device • Injection of the watchdog (by sending of a bad answer) For meeting this requirement it may be best to use the watchdog Q&A configuration.	TPS65381 device include a closed-loop digital watchdog. This watchdog requires that specific trigger messages be passed between the MCU and the TPS65381 device at specific timing intervals in order to enable operation of the safing path or external power stages through the ENDRV pin. The TPS65381-Q1 device has two different watchdog timer configurations for the external MCU to send the watchdog trigger: • Watchdog trigger input configuration (WDTI configuration). • Question-answer configuration (Q&A configuration).
Input/output and communication links: Failure in the input output and communication links must be detectable as it can lead to failure in end systems	Information redundancy techniques	GIO2, PWM2, MSP3, ADC3, HET2, SPI3, IC2, SCI2, LIN2,CAN2, FRY4	N/A	N/A	Information redundancy techniques can be applied via software as an additional runtime diagnostic on GIO function. HALCoGen provides supporting API to implement this diagnostic.
	Boot time input self test	ADC1	SL_SelfTest_ADC	• Capability to run – Self test on given ADC input PIN	The Hercules MibADC module implements an input self-test engine that can detect short to ADREFLO,ADREFHI or open input.
	Boot time and periodic converter calibration	ADC2A/B	SL_adcCalibration	Calibration API can be used for: • Calibration that improves converter accuracy • Software comparison of the conversion of known reference values from the calibration logic provides a diagnostic on the converter functionality. • Repeated execution of the calibration routine can be used to detect drift during application.	ADC includes specific hardware that allows application program to calibrate the ADC. The calibration can be performed periodically and the calibrated offset can be stored so as to analyze if there is serious deviation in the calibrated offset.
	Boot time PBIST check of MibADC SRAM	ADC5A/B	SL_SelfTest_PBIST	Execute PBIST tests on RAM groups with different algorithms	The PBIST controller architecture provides a run-time-programmable memory BIST engine for varying levels of coverage across many embedded memory instances.

Table 2. Safety Measures for PESSRAL (continued)

Component and Functional Safety Requirement	Measures to be Taken	Mapping to Hercules Safety Manual [1]	Mapping to Hercules SafeTI Diagnostic Library for test of the diagnostic	API Usage	Comments
TPS65381 Error Monitoring: TPS65381 device has the capability to monitor error pin of an external MCU and take appropriate actions. Error monitoring failure can lead to incorrect operation of the end system	Self test on TPS error monitoring	N/A	TPS_TestError PinMonitoring	The API creates an error in Hercules MCU and verifies that the TPS error pin monitoring is working.	TPS65381 can be configured to monitor nerror pin output on the MCU.
TPS65381 analog and logical functional units: Any failure in analog and digital functional units must be detectable	Self test on the analog and logical functional units	N/A	BIST support API (see Table 4)	API enables the configuration and usage of the built in self tests on logical and analog functions in TPS65381	TPS65381 has several analog and logical functional units correct functioning of these units is necessary as an important aspect of the safety of the end system.
TPS65381 configuration registers: Configuration registers in the TPS65381 device have to be protected appropriately. A runaway code can overwrite the TPS65381 configuration registers resulting in failure of the end system.	Usage of the CRC diagnostic on configuration registers is recommended	N/A	CRC Support API (seeTable 5)	Provides API for: • EEPROM(used for storing analog trim values) CRC check. • Calculation of the predetermined golden CRC value. • Getting CRC error status • Injecting fault in CRC golden value which results in failure in the CRC check.	The CRC controller is a diagnostic module that performs CRC to verify the integrity of the SPI-mapped register space. The responsibility of the CRC controller is to calculate the signature for a set of data and then compare the calculated signature value against a predetermined good-signature value.
	Usage of write protection on configuration registers is recommended.	N/A	TPS_Protect Configuration Registers	• The API can be used to enable or disable write protection on the configuration registers	TPS65381 device supports SW_LOCK command that can be used to enable the write-access lock. A register cannot be written after write-access lock protection is set. The lock is cleared by software or by a power-on reset.
	Usage of register read back and comparison is recommended.	N/A	TPS_RegReadback Compare	• The API can be called periodically during the operation of the system to check and verify that the TPS65381 configuration has not unintentionally changed.	Every time the TPS65381 driver changes, the configuration registers in the TPS65381 device update a data structure that stores the current configuration register settings. The readback compare reads the configuration register values (from the TPS65381 device) and compares them against the stored data structure.
TPS65381 ENDRV pin Failure: ENDRV pin of the TPS device is the enable output signal for peripherals such as Motor drive IC. The failure in the ENDRV pin operation can result in incorrect operation of the end system.	Periodic read back of the ENDRV pin failure information for taking corrective actions is recommended.	N/A	TPS_GetENDRV_ ErrorStatus	The API returns error status information if any in the ENDRV pin operation.	The ENDRV pin features a read-back circuit to compare the external ENDRV level with the internally applied ENDRV level. This is to detect any possible failures in the ENDRV pullup or pulldown components. A failure can be detected by the MCU through the SPI register SAFETY_STAT_4, bit 1.

Table 2. Safety Measures for PESSRAL (continued)

Component and Functional Safety Requirement	Measures to be Taken	Mapping to Hercules Safety Manual [1]	Mapping to Hercules SafeTI Diagnostic Library for test of the diagnostic	API Usage	Comments
TPS65381 NRES pin failure: Cold-reset (NPOR_RST) output signal for µC. In some systems, the NRES pin is connected to the NPOR_RST pin of the MCU. The TPS resets the MCU in case of severe errors by active low signal on the NRES pin. The incorrect operation of the NRES pin leads to improper functioning of the system.	Enable NRES monitoring and do a periodic check of NRES Error.(When the NRES_ERR flag is set, there is a difference in the external NRES pin level and configured NRES level from the TPS65381 device)	N/A	NRES Error Monitoring API (see Table 6)	Capability to configure and use NRES pin error monitoring	The NRES pin features a readback of the external NRES level. The value can be read on the DIAG_OUT pin and in the SPI register SAFETY_STAT_3, bit 5.
TPS65381 analog and digital signals: Incorrect values on the analog voltages and digital signals in TPS device may cause a failure in the end system.	Analog and digital MUX diagnostics must be used as a periodic and boot-time safety diagnostics. Redundant monitoring of the safety critical voltage and important digital signals of TPS65381 device is necessary.	N/A	MUX Diagnostics API (see Table 7)	Capability to: • Enable or disable analog or digital signal on MUX_OUT pin. • Check enable AMUX signal limits.	TPS65381 device analog and digital MUX facilitates external pin interconnect tests by feeding back to the input pin state, internal module self-test status, or safety-critical comparator output.

Table 3. TPS65381 External Watchdog Support API

API	Description
TPS_WatchdogInit	API to set up and initialize TPS watchdog
TPS_SetWatchdogMode	Set Watchdog configuration in either Q and A mode or in WDTI mode
TPS_ConfigureWatchdogWindows	Configure watchdog windows
TPS_ConfigureWatchdogReset	Enable or disable watchdog reset
TPS_UpdateActiveWDToken	Update active Watchdog Token in the TPS device. The updated token is used to send appropriate watchdog answer to theTPS device.
TPS_SendWdgResponse	The API is used for sending watchdog answer to the TPS device. The API sends the correct response to the TPS device based on the currently active token.
TPS_GetWatchdogFailureStatus	The API provides the status of the watchdog failure status flag
TPS_ClearWatchdogFailureStatusFlag	The API clears the watchdog fail status bit in the register SAFETY_ERR_STAT TPS register
TPS_GetWatchdogAnswerCount	The API provides watchdog failure count information
TPS_GetWatchdogErrorType	The API provides failure status information of the watchdog
TPS_GetWatchdogFailCount	The API provides watchdog failure count information
TPS_WDSetTokenSeedValue	Set Token Seed value for the watchdog
TPS_WDSetTokenFDBCKValue	Set Token Feedback value for the watchdog
TPS_FaultInjectWD	Inject fault in watchdog

Table 4. BIST Support API

API	Description
TPS_ConfigureBISTatStartup	The API enables and disables BIST at startup
TPS_StartBIST	The API starts the built-in self-test ABIST/LBIST (manually starting)
TPS_GetBISTRunningStatus	The API gets the built in self-test running status
TPS_GetABISTTestStatus	The API gets the built in self-test running status
TPS_GetLBISTTestStatus	The API gets the built in self-test running status

Table 5. CRC Support API

API	Description
TPS_ConfigureSafetyCheckControl	Sets the safety check control register flags
TPS_StartEECRCCheck	The API starts EEPROM CRC check
TPS_InitializeDatastringforCRCCaclcuation	The API initializes the 64-bit data string for 8-bit CRC calculation
TPS_CalculateCRC8	Calculate the CRC 8 value of the TPS65381 configuration registers (stored as 64-bit data)
TPS_GetCRCErrorStatus	The API is used to get the error status of Configuration register CRC calculation and EEPROM CRC calculation.
TPS_GetEECRCCheckRunningStatus	The API gets the EE CRC check running status
TPS_FaultInjectCRC	Inject CRC fault
TPS_UpdateRegisterSafetyCfgCRC	Update SAFETY_CFG_CRC register

Table 6. NRES Error Monitoring API

API	Description
TPS_ConfigureNRESMonitoring	Configure NRES pin monitoring
TPS_ConfigureNRESMonitoring	The API provides failure information of NRES pin (read back state of NRES PIN)

Table 7. MUX Diagnostics API

API	Description
TPS_EnableAMUXSignal	Enable particular AMUX (analog signal to be enabled on the MUX output) signal
TPS_DisableMUXDiagnostic	Disable MUX diagnostics
TPS_CheckEnabledAMUXSignalLimits	Check enabled AMUX signal limits (check to see if the analog signal is within the limits)
TPS_EnableDMUXSignal	Enable particular DMUX signal

References

1. *TMS570LS12x/11x 16/32-Bit RISC Flash Microcontroller Technical Reference Manual* (SPNU515)
2. *RM46x 16/32-Bit RISC Flash Microcontroller Technical Reference Manual* (SPNU514)
3. *Safety Manual for RM46x Hercules™ ARM®- Based Safety Critical Microcontrollers User's Guide* (SPNU551)
4. *TPS65381-Q1 Multi-Rail Power Supply for Microcontrollers in Safety Applications Data Manual* (SLVSBC4)
5. ISO 22201:2009 Design and development of programmable electronic systems in safety-related applications for lifts
6. ISO 26262-1:2011 Road vehicles -- Functional safety
7. SafeTI Diagnostic Library including Hercules Safety MCU Diagnostic Library and TPS65381 PMIC Driver

Introduction to Elevator Control System

A.1 Elevator Control System

CAUTION

Appendix A provides a high level, simplified illustration of a typical elevator control system. It is NOT intended to be complete, nor reflect customer-specific application needs or mandatory legislative compliance requirements that might apply to specific installation, a responsibility of the OEM and/or system integrator.

An elevator control system is responsible for coordinating all aspects of elevator service such as travel speed, accelerating, decelerating, door opening speed, leveling and hall lantern signals. It accepts inputs (button signals) and produces outputs (elevator cars moving, doors opening, and so forth).

A.1.1 Elevator Control System Components

The elevator as a control system has a number of components (as shown in Figure 4). These can basically be divided into the following:

- Inputs
- Outputs
- Controllers

Inputs that include:

- Sensors (magnetic, infrared, and so forth)
- Buttons (hall buttons, floor request buttons, and so forth)
- Key controls
- System controls

Outputs that include:

- Actuators (door opening device, electric motors, brakes)
- Bells (emergency bell, and so forth)
- Displays (load bell, and so forth)

Controllers in elevators enable operations such as command control, visual monitoring, and so forth, which ensures that the elevators are functioning efficiently. Different types of controllers are available:

- Relay-based controllers
- PLC controller (uses microcontrollers)

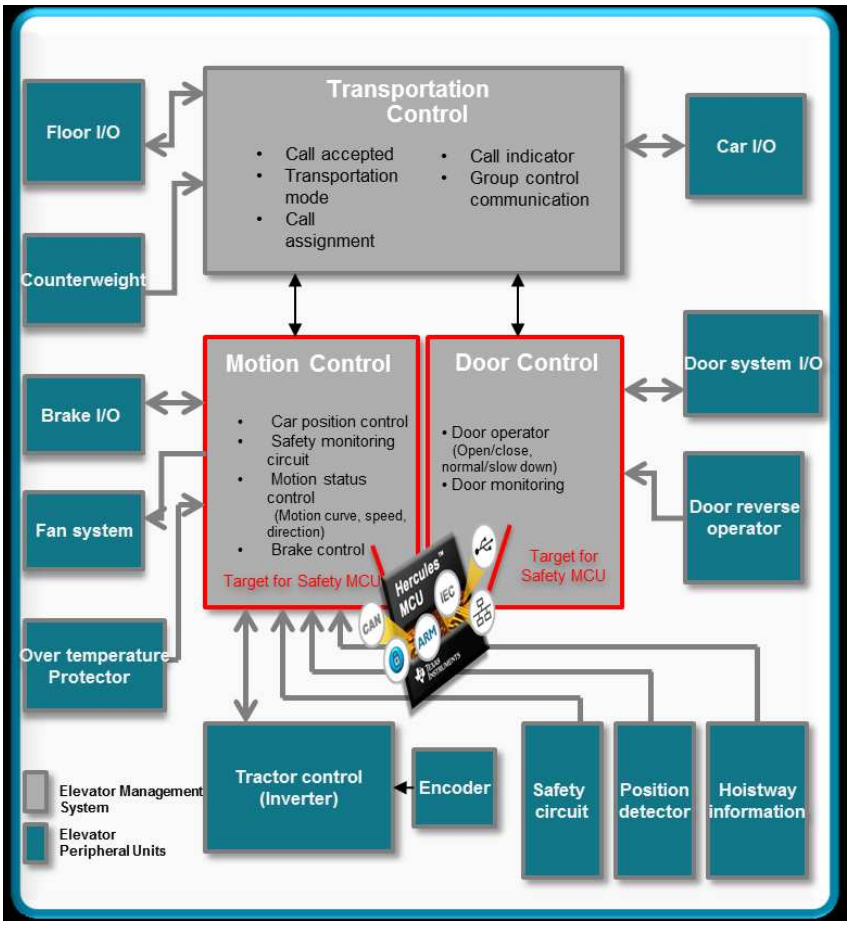

Figure 4. Simplified Elevator Control System

IMPORTANT NOTICE AND DISCLAIMER

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

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

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

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

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

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