Functional Safety Information

Functional Safety Manual for MSPM0G

Table of Contents

List of Figures

List of Tables

1 Introduction

This document is a functional safety manual for the Texas Instruments MSPM0G component. The specific orderable part numbers supported by this functional safety manual are as follows:

- MSPM0G1105
- MSPM0G1106
- MSPM0G1107
- MSPM0G1505
- MSPM0G1506
- MSPM0G1507
- MSPM0G3105
- MSPM0G3106
- MSPM0G3107
- MSPM0G3505
- MSPM0G3506
- MSPM0G3507

Functional safety manual provides information needed by system developers to help in the creation of a functional safety system using a MSPM0G component. This document includes:

- An overview of the component architecture
- An overview of the development process used to decrease the probability of systematic failures
- An overview of the functional safety architecture for management of random failures
- The details of architecture partitions and implemented functional safety mechanisms

The following information is documented separately and is not repeated in this document:

- Quantitative functional safety analysis (also known as FMEDA - Failure Modes, Effects, and Diagnostics Analysis) with detail of the different parts of the component, allowing for customized application of functional safety mechanisms

The user of this document should have a general familiarity with the MSPM0G component. For more information, refer to the MSPM0G310x-Q1 and MSPM0G350x-Q1 data sheets. This document is intended to be used in conjunction with the pertinent data sheets, technical reference manuals, and other component documentation.

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

Trademarks

ARM® is a registered trademark of Arm Limited.
All trademarks are the property of their respective owners.

2 MSPM0G Hardware Component Functional Safety Capability

This section summarizes the component functional safety capability.

This hardware component:

- Includes functional safety mechanisms to enable ease of adoption in automotive applications.
- Was developed as per Texas Instruments standard development process, which complies with the requirements of ISO 9001:2015/IATF 16949.

3 Development Process for Management of Systematic Faults

For functional safety development, it is necessary to manage both systematic and random faults. Texas Instruments follows a new-product development process for all of its components which helps to decrease the probability of systematic failures. This new-product development process is described in Section 3.1.

3.1 TI New-Product Development Process

Texas Instruments has been developing components for automotive and industrial markets since 1996. Automotive markets have strong requirements regarding quality management and product reliability. The TI new-product development process features many elements necessary to manage systematic faults. Additionally, the documentation and reports for these components can be used to assist with compliance to a wide range of standards for customer's end applications including automotive and industrial systems (e.g., ISO 26262-4, IEC 61508-2).

This component was developed using TI's new product development process which has been certified as compliant to ISO 9001 / IATF 16949 as assessed by Bureau Veritas (BV).

The standard development process breaks development into phases:

- Assess
- Plan
- Create
- Validate

Figure 3-1 shows the standard process.

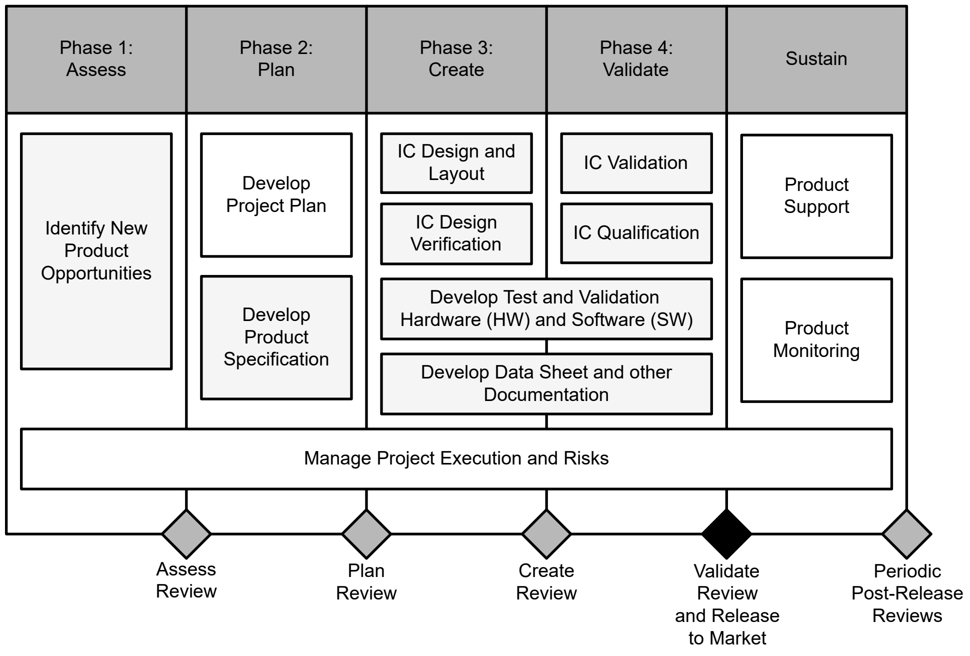
Phase 1: Assess
Phase 2: Plan
Phase 3: Create
Phase 4: Validate
Sustain
Identify New Product Opportunities
Develop Project Plan
Develop Product Specification
IC Design and Layout
IC Design Verification
IC Validation
IC Qualification
Develop Test and Validation Hardware (HW) and Software (SW)
Develop Data Sheet and other Documentation
Product Support
Product Monitoring
Manage Project Execution and Risks
Assess Review
Plan Review
Create Review
Validate Review and Release to Market
Periodic Post-Release Reviews

Figure 3-1. TI New-Product Development Process

4 MSPM0G Component Overview

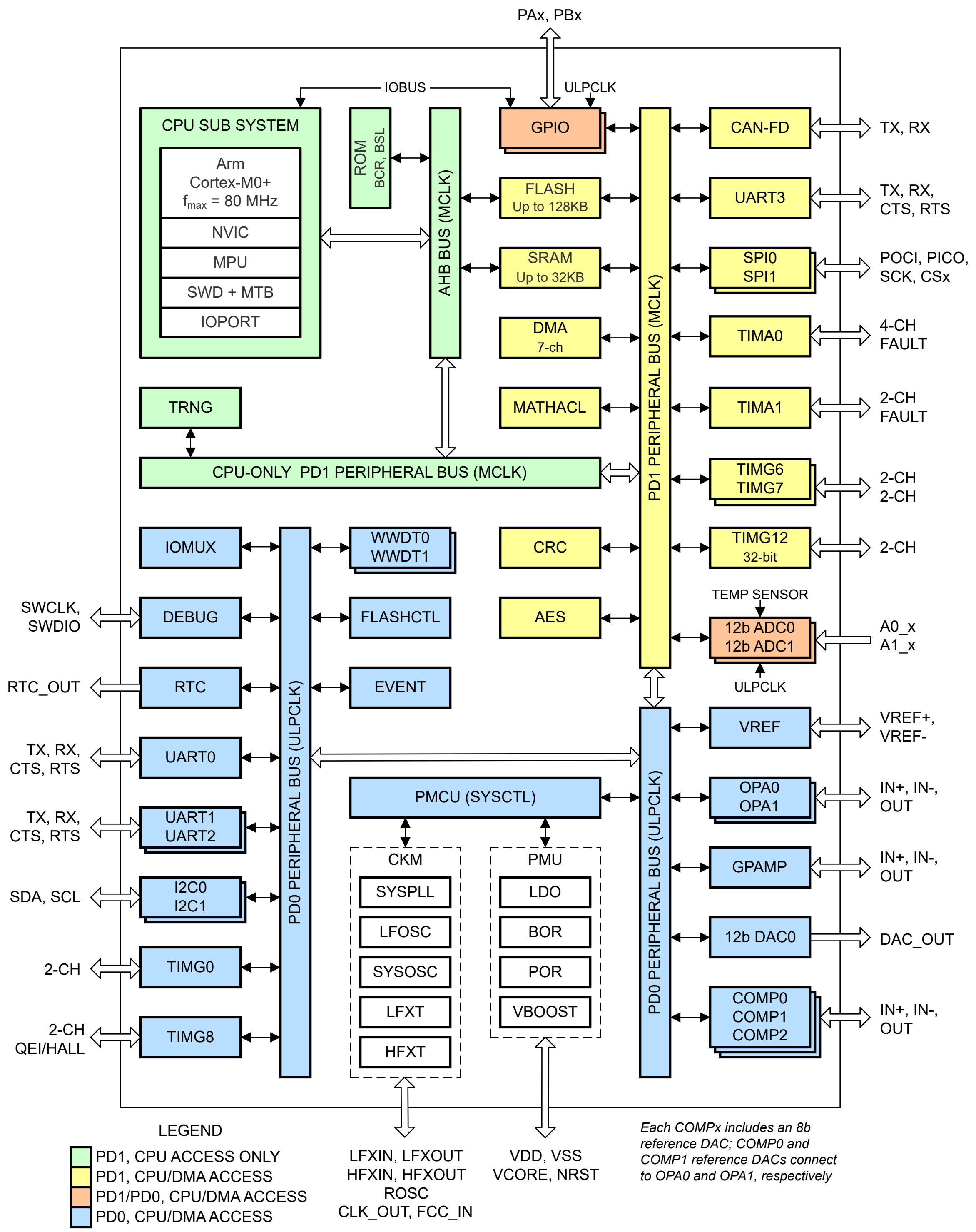

Figure 4-1. MSPM0G Block Diagram

4.1 Targeted Applications

The MSPM0G component is targeted at general-purpose functional safety applications. This is called Safety Element out of Context (SEooC) development according to ISO 26262-10:2018. In this case, the development is done based on assumptions on the conditions of the semiconductor component usage, and then the assumptions are verified at the system level. This section describes some of the target applications for this component, the component safety concept, and then describes the assumptions about the systems (also know as Assumptions of Use or AoU) that were made in performing the safety analysis.

Example target applications include, but are not limited to, the following:

- Automotive - Person occupancy detection
- Automotive - Lighting
- Automotive - Seat heaters
- Automotive - Window control

Figure 4-2 shows a generic block diagram for a seat heater (body application) system. This diagram is only an example and may not represent a complete system.

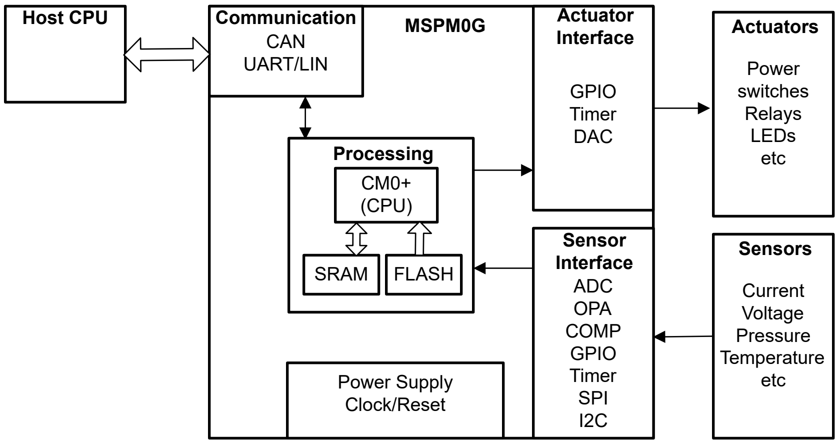

Figure 4-2. MSPM0G Typical Application

4.2 Hardware Component Functional Safety Concept

In case of internal errors in the component, one or more of the following actions can be taken:

- Take the actuator output pins to a safe state (for example, the fault logic in timers can take the outputs to a safe state).
- Reset the device.
- Communicate the error to the host CPU and let the system take the appropriate action.

4.3 Functional Safety Constraints and Assumptions

In creating a functional Safety Element out of Context (SEooC) concept and doing the functional safety analysis, TI generates a series of assumptions on system level design, functional safety concept, and requirements. These assumptions (sometimes called Assumptions of Use) are listed below. Additional assumptions about the detailed implementation of safety mechanisms are separately located in Section 6.3.

The MSPM0G Functional Safety Analysis was done under the following system assumptions:

- **[SA_1]** The MSPM0G MCU has interfaces to external sensors.
- **[SA_2]** The MSPM0G MCU has interfaces to external actuators.
- **[SA_3]** The MSPM0G MCU has interfaces to communicate with an external host controller.
- **[SA_4]** The MSPM0G MCU has a programmable CPU to execute a controller function taking sensor inputs and controlling actuator.
- **[SA_5]** The system integrator reviews the recommended diagnostics in the safety analysis report (FMEDA) and safety manual and determines the appropriate diagnostics to include in the system. These diagnostics are implemented according to the device safety manual and data sheet.

- **[SA_6]** The external power supply provides the appropriate power on each of the power inputs. These rails are monitored for deviations outside the device specifications and a reset is asserted if the voltage is outside the range.
- **[SA_7]** The MSPM0G MCU monitors failures on external clock (if present).
- **[SA_8]** The MSPM0G MCU monitors failures on external sensors.
- **[SA_9]** The MSPM0G MCU monitors failures on external actuators.
- **[SA_10]** In case of internal errors in the MSPM0G MCU or the interfacing sensors and actuators, the MSPM0G MCU is reset. The host controller monitors communication loss and determines that the MSPM0G MCU is in a faulted state.
- **[SA_11]** The system integrator provisions an actuator disable mechanism controller by the host controller.
- **[SA_14]** The system is assumed to have a FTTI > 10ms.

Listed below are the additional recommendations:

- **[COEX0]**: The following components are assumed to be not safety related (NSR components):
  - MATHACL
  - RTC
  - TRNG
  - AES
- **[COEX1]** TI recommends that components that are not in use are disabled in the application software.
- **[COEX2]** TI recommends that the interrupt sources of components that are not in use are disabled.
- **[COEX3]** TI recommends that DMA triggers of components that are not in use are disabled.
- **[COEX4]** TI recommends that unused fault inputs in timers are disabled.
- **[COEX5]** If external safety mechanisms are used, the system integrator is responsible for doing a dependent failure analysis at the system level.
- **[COEX6]** TI assumes that NSR components are not used in the safety context.
- **[COEX7]** TI recommends that debug is disabled in safety-critical applications.
- **[COEX8]** TI recommends that a default interrupt service routine is coded, even for unused interrupts.
- **[COEX9]** TI recommends that the application does not use IPs that are NSR as the trigger source of other IPs.
- **[COEX10]** TI recommends that the application does not program Flash during safety-critical tasks.

During integration activities these assumptions of use and integration guidelines described for this component shall be considered. Use caution if one of the above functional safety assumptions on this component cannot be met, as some identified gaps may be unresolvable at the system level.

5 Description of Hardware Component Parts

A semiconductor component can be divided into parts to enable a more granular functional safety analysis. This can be useful to help assign specific functional safety mechanisms to portions of the design where they provide coverage ending up with a more complete and customizable functional safety analysis. This section includes a brief description of each hardware part of this component and lists the functional safety mechanisms that can be applied to each. The content in this section is also summarized in Appendix A.

More details of the hardware components can be found in *MSPM0 G-Series 80-MHz Microcontrollers Technical Reference Manual*.

5.1 ADC

Both 12-bit analog-to-digital converter (ADC) modules in these devices, ADC0 and ADC1, support fast 12-bit conversions with single-ended inputs and simultaneous sampling operation.

ADC features include:

- 12-bit output resolution at 4Msps with greater than 11 ENOB
- HW averaging enables 14-bit effective resolution at 250ksps
- Up to 17 total external input channels with individual result storage registers
- Internal channels for temperature sensing, supply monitoring, and analog signal chain (interconnection with OPA, DAC)
- Software selectable reference:
  - Configurable internal reference voltage of 1.4V and 2.5V (requires decoupling capacitor on VREF± pins)
  - MCU supply voltage (VDD)
  - External reference supplied to the ADC through the VREF± pins
- Operates in RUN, SLEEP, and STOP modes

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

- ADC1
- ADC2
- ADC3
- ADC4
- ADC5
- ADC6

5.2 Comparator

The comparator peripheral in the device compares the voltage levels on two inputs terminals and provides a digital output based on this comparison. The comparator supports the following key features:

- Programmable hysteresis
- Programmable reference voltage:
  - External reference voltage (VREF IO)
  - Internal reference voltage (1.4V, 2.5V)
  - Integrated 8-bit reference DAC, the output can also connect to OPA input terminal internally as an output buffer.
- Configurable operation modes:
  - High-speed mode (40ns propagation delay)
  - Lower-power mode (1.5uA power consumption)
- Programmable output glitch filter delay
- Supports output wake-up device from all low-power modes
- Output connected to an advanced timer, fault-handling mechanism
- The IPSEL and IMSEL bits in comparator registers can be used to select the comparator channel inputs from device pins or from internal analog modules.

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

- COMP1
- COMP2
- COMP3
- COMP4
- COMP5

5.3 DAC

The 12-bit buffered digital-to-analog converter (DAC) in these devices converts a digital input value into an analog voltage to a buffered output channel. The DAC supports the following key features:

- Up to 1Msps output sampling rate
- 8-bit or 12-bit voltage-output resolution
- Self-calibration option for offset error correction
- Straight binary or twos-complement data format
- Integrated sample time generator for generation of predefined sampling rates
- Integrated FIFO and support DMA operation
- Two hardware triggers from event fabric for conversion
- Programmable voltage reference options:
  - Supply voltage (VDD)
  - External reference voltage (VREF IO)
  - Internal reference voltage (1.4V, 2.5V)

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

- DAC1
- DAC2
- DAC3

5.4 OPA

The zero-drift op amps (OPAs) in these devices (OPA0 and OPA1) are chopper stabilized operational amplifiers with rail-to-rail input and output and a programmable gain stage feedback loop.

The OPA peripherals support the following key features:

- Software-selectable zero-drift chopper stabilization for improved accuracy and drift performance
- Factory trimming to remove offset error
- 6MHz GBW in standard (STD) mode and 100µA quiescent current in low-power (LP) mode
- Burnout current source (BCS) integrated to monitor sensor health
- Programmable gain amplifier (PGA) up to 32x

The OPA features configurable input muxes P-MUX, N-MUX, and M-MUX to support various analog signal chain amplifier configurations that include general purpose, inverting, noninverting, unity gain, cascade, noninverting cascade, difference, and more. The following tables list the input channel mapping for each OPA.

1. The connection to OPA and DAC_OUT connects using the PA15 pin. When connecting DAC_OUT to OPA, avoid using external circuitry on the PA15 pin.

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

- OA1
- OA2
- OA3

5.5 CPU

The CPU subsystem (MCPUSS) implements an ARM® Cortex®-M0+ CPU, an instruction prefetch, cache, a system timer, a memory protection unit, and interrupt management features. The ARM Cortex-M0+ is a cost-effective, 32-bit CPU which delivers high performance and low power to embedded applications. Key features of the CPU subsystem include:

- ARM Cortex-M0+ CPU supporting clock frequencies from 32kHz to 80MHz
  - ARMv6-M thumb instruction set (little endian) with single-cycle 32x32 multiply instruction
  - Single-cycle access to GPIO registers through ARM single-cycle I/O port
- Prefetch logic to improve sequential code execution and I-cache with four 64-bit cache lines
- System timer (SysTick) with 24-bit down counter and automatic reload
- Memory protection unit (MPU) with 8 programmable regions
- Nested vectored interrupt controller (NVIC) with 4 programmable priority levels and tail-chaining
- Interrupt groups for expanding the total interrupt sources, with jump index for low interrupt latency

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

- CPU1
- CPU2
- CPU3
- CPU4

5.6 RAM

MSPM0Gxx MCUs include a low power, high-performance SRAM memory with zero wait state access across the supported CPU frequency range of the device. MSPM0Gxx MCUs also provides up to 32KB of SRAM with hardware parity. SRAM memory can be used for storing volatile information such as the call stack, heap, global data, and code. The SRAM memory content is fully retained in run, sleep, stop, and standby operating modes and is lost in shutdown mode. A write protection mechanism is provided to allow the application to prevent unintended modifications to the SRAM memory. Write protection is useful when placing executable code into SRAM as write protection provides a level of protection against unintentional overwrites of code by either the CPU or DMA. Placing code in SRAM can improve performance of critical loops by enabling zero wait state operation and lower power consumption.

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

- SYSMEM1
- SYSMEM2
- SYSMEM3
- SYSMEM4

5.7 FLASH

A single bank of non-volatile Flash memory is provided for storing executable program code and application data. Key features of the Flash include:

- Hardware ECC protection (encode and decode) with single-bit error correction and double-bit error detection
- In-circuit program and erase operations supported across the entire recommended supply range
- Small 1kB sector sizes (minimum erase resolution of 1kB)
- Up to 100,000 program and erase cycles on the lower 32kB of the Flash memory, with up to 10,000 program and erase cycles on the remaining Flash memory (devices with 32kB support 100,000 cycles on the entire Flash memory). For a complete description of the Flash memory, refer to the NVM chapter of the technical reference manual.

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

- FLASH1
- FXBAR2
- FXBAR3
- FXBAR4

5.8 GPIO

The general purpose input output (GPIO) peripheral provides the user with a means to write data out and read data in, to and from, the device pins. Through the use of the Port A and Port B GPIO peripherals, these devices support up to 60 GPIO pins.

The key features of the GPIO module include:

- 0 wait state MMR access from CPU
- Set, clear, and toggle multiple bits without the need of a read-modify-write construct in software
- GPIOs with *Standard With Wake* drive functionality able to wake the device from SHUTDOWN mode
- *FastWake* feature enables low-power wakeup from STOP and STANDBY modes for any GPIO port
- User controlled input filtering

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

- GPIO1
- GPIO2
- GPIO3

5.9 DMA

The direct memory access (DMA) controller allows movement of data from one memory address to another without CPU intervention. For example, the DMA can be used to move data from ADC conversion memory to SRAM. The DMA reduces system power consumption by allowing the CPU to remain in low power mode, without having to awaken to move data to or from a peripheral.

The DMA in these devices supports the following key features:

- Seven independent DMA transfer channels
  - Four basic channel support (single-transfer modes)
  - Three full-feature channel support (repeated-transfer modes)
- Configurable DMA channel priorities
- Byte (8-bit), short word (16-bit), word (32-bit) and long word (64-bit) or mixed byte, and word transfer capability
- Transfer counter block size supports up to 64k transfers of any data type
- Configurable DMA transfer trigger selection
- Active channel interruption to service other channels
- Early interrupt generation for ping-pong buffer architecture
- Cascading channels upon completion of activity on another channel
- Stride mode to support data re-organization, such as 3-phase metering applications

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

- DMA1
- DMA2
- DMA3
- DMA4

5.10 SPI

The serial peripheral interface (SPI) peripherals in these devices support the following key features:

- Support ULPCLK/2 bit rate and up to 32Mbits/s in both controller and peripheral mode
- Configurable as a controller or a peripheral
- Configurable chip select for both controller and peripheral
- Programmable clock prescaler and bit rate
- Programmable data frame size from
- Programmable data frame size from 7-bits to 16-bits (peripheral mode)
- Separated transmit and receive FIFOs support DMA data transfer

- Supports TI mode, Motorola mode and National Microwire format

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

- SPI1
- SPI2
- SPI3
- SPI4
- SPI5

5.11 I2C

The inter-integrated circuit interface (I^2C) peripherals in these devices provide bidirectional data transfer with other I2C devices on the bus and support the following key features:

- 7-bit and 10-bit addressing mode with multiple 7-bit target addresses
- Multiple-controller transmitter or receiver mode
- Target receiver or transmitter mode with configurable clock stretching
- Support Standard-mode (Sm), with a bit rate up to 100 kbit/s
- Support Fast-mode (Fm), with a bit rate up to 400 kbit/s
- Support Fast-mode Plus (Fm+), with a bit rate up to 1 Mbit/s
- Separated transmit and receive FIFOs support DMA data transfer
- Support SMBus 3.0 with PEC, ARP, timeout detection and host support
- Wakeup from low-power mode on address match
- Support analog and digital glitch filter for input signal glitch suppression

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

- I2C1
- I2C2
- I2C3
- I2C4
- I2C5
- I2C6
- I2C7

5.12 UART

The UART peripherals (UART0, UART1, UART2, and UART3) provide the following key features:

- Standard asynchronous communication bits for start, stop, and parity
- Fully programmable serial interface
  - Five, six, seven, or eight data bits
  - Even, odd, stick, or no-parity bit generation and detection
  - One or two stop bit generation
  - Line-break detection
  - Glitch filter on the input signals
  - Programmable baud rate generation with oversampling by 16, 8, or 3
  - Local Interconnect Network (LIN) mode support
- Separated transmit and receive FIFOs support DMA data transfer
- Support transmit and receive loopback mode operation
- Refer to Table 5-1 for detailed information on supported protocols

Table 5-1. UART Features

UART Features	UART0 (Extend)	UART1 and 2 (Main)	UART3 (Main)
Active in stop and standby mode	Yes	Yes	-
Separate transmit and receive FIFOs	Yes	Yes	Yes
Support hardware flow control	Yes	Yes	Yes
Support 9-bit configuration	Yes	Yes	Yes
Support LIN mode	Yes	-	-
Support DALI	Yes	-	-
Support IrDA	Yes	-	-
Support ISO7816 Smart Card	Yes	-	-
Support Manchester coding	Yes	-	-

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

- UART1
- UART2
- UART3
- UART4
- UART5
- UART6

5.13 Timers (TIMx)

The timer peripherals in these devices support the following key features, for specific configuration see Table 5-2:

Specific features for the general-purpose timer (TIMGx) include:

- 16-bit up, down, up-down or down-up counter, with repeat-reload mode
- 32-bit up, down, up-down or down-up counter, with repeat-reload mode
- Selectable and configurable clock source
- 8-bit programmable prescaler to divide the counter clock frequency
- Two independent channels for:
  - Output compare
  - Input capture
  - PWM output
  - One-shot mode
- Support quadrature encoder interface (QEI) for positioning and movement sensing
- Support synchronization and cross trigger among different TIMx instances in the same power domain
- Support interrupt and DMA trigger generation and cross peripherals (such as ADC) trigger capability
- Cross trigger event logic for Hall sensor inputs

Specific features for the advanced timer (TIMAx) include:

- 16-bit down or up-down counter, with repeat-reload mode
- Selectable and configurable clock source
- 8-bit programmable prescaler to divide the counter clock frequency
- Repeat counter to generate an interrupt or event only after a given number of cycles of the counter
- Up to four independent channels for:
  - Output compare
  - Input capture
  - PWM output
  - One-shot mode
- Shadow register for load and CC register
- Complementary output PWM
- Asymmetric PWM with programmable dead band insertion
- Fault handling mechanism to verify the output signals in a safe user-defined state when a fault condition is encountered
- Support synchronization and cross trigger among different TIMx instances in the same power domain
- Support interrupt and DMA trigger generation and cross peripherals (such as ADC) trigger capability
- Two additional capture/compare channels for internal events

Table 5-2. TIMx Configurations

TIMER NAME	POWER DOMAIN	RESOLUTION	PRESCALER	REPEAT COUNTER	CAPTURE / COMPARE CHANNELS	PHASE LOAD	SHADOW LOAD	SHADOW CC	DEADBAND	FAULT	QEI
TIMG0	PD0	16-bit	8-bit	–	2	–	–	–	–	–	–
TIMG6	PD1	16-bit	8-bit	–	2	–	–	–	–	–	–
TIMG7	PD1	16-bit	8-bit	–	2	–	Yes	Yes	–	–	–
TIMG8	PD0	16-bit	8-bit	–	2	–	–	–	–	–	Yes
TIMG12	PD1	32-bit	–	–	2	–	–	Yes	–	–	–
TIMA0	PD1	16-bit	8-bit	8-bit	4	Yes	Yes	Yes	Yes	Yes	–
TIMA1	PD1	16-bit	8-bit	8-bit	2	Yes	Yes	Yes	Yes	Yes	–

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

- TIM1
- TIM2
- TIM3

- TIM4
- TIM5
- TIM6

5.14 Power Management Unit (PMU)

The power management unit (PMU) generates the internally regulated core supplies for the device and provides supervision of the external supply (VDD). The PMU also contains the bandgap voltage reference used by the PMU, as well as analog peripherals. Key features of the PMU include:

- Power-on reset (POR) supply monitor
- Brown-out reset (BOR) supply monitor with early warning capability using three programmable thresholds
- Core regulator with support for RUN, SLEEP, STOP, and STANDBY operating modes to dynamically balance performance with power consumption
- Parity-protected trim to immediately generate a power-on reset (POR) in the event that a power management trim is corrupted. For more details, see the PMU chapter of the *MSPM0 G-Series 80-MHz Microcontrollers Technical Reference Manual*.

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

- SYSCTL8
- SYSCTL10
- SYSCTL14
- SYSCTL15
- SYSCTL16

5.15 Clock Module (CKM)

The clock module provides the following oscillators:

1. **LFOSC**: Internal low-frequency oscillator (32KHz)
2. **SYSOSC**: Internal high-frequency oscillator (4MHz or 32MHz with factory trim, 16MHz or 24MHz with user trim)
3. **LFXT/LFCKIN**: Low-frequency external crystal oscillator or digital clock input (32KHz)
4. **HFXT/HFCKIN**: High-frequency external crystal oscillator or digital clock input (4MHz to 48MHz)
5. **SYSPLL**: System phase locked loop with three outputs (32MHz to 80MHz)

The following clocks are distributed by the clock module for use by the processor, bus, and peripherals:

- **MCLK**: Main system clock for PD1 peripherals, derived from SYSOSC, LFCLK, or HSCLK, active in RUN and SLEEP modes
- **CPUCLK**: Clock for the processor (derived from MCLK), active in RUN mode
- **ULPCLK**: Ultra-low power clock for PD0 peripherals, active in RUN, SLEEP, STOP, and STANDBY modes
- **MFCLK**: 4MHz fixed mid-frequency clock for peripherals, available in RUN, SLEEP, and STOP modes
- **MFPCLK**: 4MHz fixed mid-frequency precision clock, available in RUN, SLEEP, and STOP modes
- **LFCLK**: 32kHz fixed low-frequency clock for peripherals or MCLK, active in RUN, SLEEP, STOP, and STANDBY modes
- **ADCCLK**: ADC clock, available in RUN, SLEEP and STOP modes
- **CLK_OUT**: Used to output a clock externally, available in RUN, SLEEP, STOP, and STANDBY modes
- **HFCLK**: High-frequency clock derived from HFXT or HFCLK_IN, available in RUN and SLEEP mode
- **HSCLK**: High-speed clock derived from HFCLK or the SYSPLL, available in RUN and

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

- SYSCTL1
- SYSCTL2
- SYSCTL3
- SYSCTL4
- SYSCTL5
- SYSCTL6

- WDT
- SYSCTL9
- SYSCTL11
- SYSCTL12

5.16 CAN-FD

The controller area network (CAN) controller enables communication with a CAN2.0A, CAN2.0B, or CAN-FD bus and is compliant to ISO 11898-1:2015 standard supporting up to 5Mbit/s bit rate. Key features of the CAN-FD peripheral include:

- Full support for 64-byte CAN-FD frames
- Dedicated 1kB message SRAM with ECC
- Configurable transmit FIFO, transmit queue and event FIFO (up to 32 elements)
- Up to 32 dedicated transmit buffers and 64 dedicated receive buffers
- Two configurable receive FIFOs (up to 64 elements each)
- Up to 128 filter elements
- Two interrupt lines
- Power-down and wake-up support
- Timestamp counter

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

- MCAN1
- MCAN2
- MCAN3
- MCAN4
- MCAN5
- MCAN6
- MCAN7

5.17 Events

The event manager transfers digital events from one entity (for example, a peripheral) to another (for example, a second peripheral, the DMA, or the CPU). The event manager implements event transfer through a defined set of event publishers (generators) and subscribers (receivers), which are interconnected through an event fabric containing a combination of static and programmable routes. Events that are transferred by the event manager include:

- Peripheral event transferred to the CPU as an interrupt request (IRQ) (Static Event)
  - Example: RTC interrupt is sent to the CPU
- Peripheral event transferred to the DMA as a DMA trigger (DMA Event)
  - Example: UART data receives trigger to DMA to request a DMA transfer
- Peripheral event transferred to another peripheral to directly trigger an action in hardware (Generic Event).
  - Example: TIMx timer peripheral publishes a periodic event to the ADC subscriber port, and the ADC uses the event to trigger start-of-sampling

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

- EVENT1
- EVENT2

5.18 IOMUX

The IOMUX manages the selection of the peripheral function used on a digital I/O. The IOMUX also provides the controls for the output driver, input path, and the wake-up logic for wakeup from SHUTDOWN mode.

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

Table 5-3. IOMUX Safety Mechanisms

Safety Mechanism	Description	Faults/Failure modes
GPIO1	Software test of function using I/O loopback	This tests the functioning of GPIO. This test also provides coverage for the faults in the IOMUX logic and the I/O drivers.
IOMUX1	Periodic Software Readback of Static Configuration Registers	Targets the static configuration registers in IOMUX.
IOMUX2	IOMUX coverage as part of other IP safety mechanisms.	In general, any fault in IOMUX is covered by the safety mechanisms of the IPs.

5.19 VREF

The shared voltage reference module (VREF) in these devices contain a configurable voltage reference buffer, which allows users to supply a stable reference to on-board analog peripherals. VREF also supports bringing in an external reference for applications where higher accuracy is required. VREF features include:

- 1.4V and 2.5V user-selectable internal references
- Internal reference supports full speed ADC operation
- Support for bringing in an external reference on VREF± device pins
- Requires a decoupling capacitor placed on VREF± pins for proper operation

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

Table 5-4. VREF Safety Mechanisms

Safety Mechanism	Description	Faults/Failure modes
ADC4	ADC window comparator	If ADC is using the internal reference, faults in the VREF can result in ADC results being out of range. These failure modes can be detected using the window comparator in ADC.
COMP3	External pin input to COMP	If the comparator uses the internal reference, then the test which checks the comparator function also provides coverage of the faults in the internal reference.
REF1	Periodic software readback of static configuration registers	Targets the static configuration registers in comparator.
REF2	VREF to ADC reference input	VREF can be tested as part of test which checks the ADC functioning by choosing to use the internal reference for the ADC test.

5.20 WWDT

The windowed watchdog timer (WWDT) can be used to supervise the operation of the device, specifically code execution. The WWDT can be used to generate a reset or an interrupt if the application software does not successfully reset the watchdog within a specified window of time. Key features of the WWDT include:

- 25-bit counter
- Programmable clock divider
- Eight software selectable watchdog timer periods
- Eight software selectable window sizes
- Support for stopping the WWDT automatically when entering a sleep mode
- Interval timer mode for applications which do not require watchdog functionality

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

- WDT1
- WDT2
- WDT3

5.21 CRC

The cyclical redundancy check (CRC) module provides a signature for an input data sequence. Key features of the CRC module include:

- Support for 16-bit CRC based on CRC16-CCITT
- Support for 32-bit CRC based on CRC32-ISO3309
- Support for bit reversal

Many safety mechanisms involve computing CRC on a block of data. This module can be used to do those checks efficiently (to provide diagnostic coverage on a specific function):

Table 5-5. CRC Safety Mechanisms

Safety Mechanism	Description	Faults/Failure modes
CRC (latent fault coverage)	CRC Checker	Any fault in CRC computation is covered by the CRC check being different from the golden CRC.
CRC1 (latent fault coverage)	Periodic software readback of static configuration registers	Targets the static configuration registers in CRC.

Note

Since CRC is a safety mechanism, only latent faults are considered.

6 MSPM0G Management of Random Faults

For a functional safety critical development it is necessary to manage both systematic and random faults. The MSPM0G component architecture includes many functional safety mechanisms, which can detect and respond to random faults when used correctly. This section of the document describes the architectural functional safety concept for each sub-block of theMSPM0G component. The system integrator shall review the recommended functional safety mechanisms in the functional safety analysis report (FMEDA) in addition to this safety manual to determine the appropriate functional safety mechanisms to include in their system. The component data sheet or technical reference manual (if available) are useful tools for finding more specific information about the implementation of these features.

6.1 Fault Reporting

Internal faults are reported to the host controller through the host bus.

6.2 Functional Safety Mechanism Categories

This section includes a description of the different types of functional safety mechanisms that are applied to the design blocks of the MSPM0G component.

The functional safety mechanism categories are defined as follows:

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

6.3 Description of Functional Safety Mechanisms

This section provides a brief summary of the functional safety mechanisms available on this component.

6.3.1 ADC1, COMP1, DAC1, DMA1, GPIO2, TIM2, I2C2, IOMUX1, OA1, SPI2, UART2, SYSCTL5, MCAN3, CPU4, CRC1, EVENT1, REF1, WDT1: Periodic Read of Static Configuration Registers

Periodic software reads of static configuration registers is one of the methods listed in ISO26262 (Table D-4, Chapter 5). This mechanism involves the software verifying the values of static configuration registers against a known reference. One of the methods is to compute a reference CRC signature for the static configuration register values. During the course of the application, the CRC is computed periodically and compared with the reference CRC signature.

Note

Static configuration registers are those registers whose contents are configured once and remains constant through the course of application.

6.3.2 ADC2: Software Test of Functionality

In this method, the internal DAC output is used to setup a known voltage on the ADC input channel and the other parameters of the ADC, for example, the sampling time, reference sources, and sequencer modes are setup similar to the actual application. A software trigger can be utilized to trigger the ADC. The output of the ADC can be compared against the expected range.

6.3.3 ADC3: ADC Trigger Overflow Check

If a trigger to ADC is fired, while a sample or conversion operation is in progress, the TOVIFG flag is set. This flag can be configured to generate an interrupt and appropriate action can be taken.

6.3.4 ADC4: Window Comparator

There is one window comparator unit available in the ADC that can be used to check if the input signal is within the predefined threshold values set by software. The ADC result (digital value corresponding to the input signal level) that goes into MEMRES or FIFO is what gets checked against the threshold values of the window comparator.

Based on the comparison, the window comparator can generate 3 interrupt conditions:

1. LOWIFG – Conversion result is below the low threshold (WCLOW)
2. HIGHIFG – Conversion result is above the high threshold (WCHIGH)
3. INIFG – Conversion result is in between, or equal to, the low and high thresholds

The window comparator low and high threshold values are global for all channels and the window comparison feature can be enabled for each channel, as needed, using the WINCOMP bit in the MEMCTL register.

When the ADC result data format (CTL2.DF) or resolution (CTL2.RES) configuration is changed, the window comparator threshold values are not reset by hardware and are retained as is. The software application is expected to reconfigure the threshold values as appropriate after changing the data format and resolution configuration.

6.3.5 ADC5: Test of Window Comparator

This test checks if the window comparator is working. As part of the ADC2 test, or a separate test, the ADC2 can be set so the threshold results in the ADC crossing the predefined thresholds. The test can check if the window comparator logic detects these conditions.

6.3.6 ADC6: ADC Plausibility Checks

This checks the plausibility of inputs being sampled by ADC. For example, if information is related to:

- Expected bandwidth of the input signal
- Range of the signal
- Expected sampling rate

Checks can be performed to see if the readings meet these plausibility conditions. In addition to doing checks on the ADC output, the time duration from the last interrupt can be checked in software and compared against the expected ADC sampling rate. If the duration from the last interrupt is beyond the acceptable range, corrective actions can be taken.

6.3.7 OA2: Test of OA Using Internal DAC as a Driver

In this test method, the DAC output is chosen as input to OPA. The OPA is configured per the application configuration. The output of OPA can be measured using ADC.

6.3.8 OA3: ADC Monitoring of OA Output

OA outputs are connected to ADC. ADC can be used to check the operation of OA. This can be a stand alone test or continuous monitoring, depending the application context.

6.3.9 COMP2: Software Test of Comparator Using Internal DAC

In this test method, one input to the comparator is connected to DAC, the other input of comparator is connected to the internal 8-bit DAC. The result of the comparison can be checked using the comparator output monitoring.

6.3.10 COMP3: External Pin Input to COMP

In case the 12-bit DAC is unavailable, an external input can be connected to the positive terminal input and the negative terminal is fed by internal DAC8 (COMPDAC). Software can vary the DAC8 setting and check the output from COMP (for example, 0 or 1).

6.3.11 COMP4: Comparator Hysteresis

The comparator has a built-in hysteresis mechanism which can filter out noise around the comparator thresholds.

6.3.12 COMP5: Redundant Comparator

In this method, the same signal is connected to two different comparators. The outputs from the two comparators are compared in the interrupt routine. By using two comparators to monitor the same signal, a fault in one of the comparators can be detected, even if one of the comparators is faulty, because the other comparator can still respond.

6.3.13 WDT: Windowed Watchdog Timer

TI recommends using a windowed watchdog timer (WWDT) to monitor the application program sequence. The WWDT can be used to generate a reset or an interrupt if the application software does not successfully reset the watchdog within a specified window of time. Key features of the WWDT include:

- 25-bit counter
- Programmable clock divider
- Eight software selectable watchdog timer periods
- Eight software selectable window sizes
- Support for stopping the WWDT automatically when entering a sleep mode
- Interval timer mode for applications that do not require watchdog functionality

For more details, see the WWDT chapter of the *MSPM0 G-Series 80-MHz Microcontrollers Technical Reference Manual*.

6.3.14 WDT2: WWDT Counter Check

At start-up, the counter can be configured in the interval-timer mode and the clock dividers can be configured to smaller values so that the counter overflows fast. In the interrupt routine, WDT can be reset using the RSTCTL register. If the WDT timer does not timeout in the given time, an error is flagged.

6.3.15 WDT3: WWDT Software Test

The WWDT module can be tested periodically by inducing a fault condition and checking that the fault is triggering a reset/nmi. The test can include checking that counter is operational.

6.3.16 REF2: VREF to ADC Reference Input

The internal reference voltage given out from VREF is used as a reference voltage in ADC for ADC operations and the ADC converted result is compared against the expected digital value.

6.3.17 CPU1: CPU Test Using Software Test Library

The Cortex-M0+ CPU and MPU are tested using the ARM Software Test Library (STL).

6.3.18 CPU2: Software Test of CPU Data Buses

This test method involves writing a known value to different addresses in the memory map and reading back the values and checking the readback value.

6.3.19 CPU3: Software Redundancy

In this method, the computation done by the primary function is also computed in alternate function, and the results of the primary and alternate functions are compared. This provides redundancy in software and uses the same underlying hardware.

6.3.20 SYSMEM1: Software Read of Memory, DMA Write

In this method, different data values can be written to different address locations in SRAM from DMA, and the data can be verified by CPU reads (a CRC check can also be performed).

6.3.21 SYSMEM2: DMA Read from SRAM, CPU Write

Different data values can be written to different address locations within SRAM by CPU and DMA can be configured to copy this data to another region in SRAM. CPU can be use to verify the transferred data.

6.3.22 SYSMEM3: Parity Logic Test

In this method, any one of the bit values is flipped and a read from the same location is performed. The test checks that a parity error occurs. Using unchecked memory aperture, data bits can be written independent of parity bits. When the same location is accessed from a parity checked region, a parity error occurs. This can be used to check the functioning of parity logic.

6.3.23 SYSMEM4: Parity Protection on SRAM

The RAM in MSPM0Gx parts have parity-check bits associated with each byte of data. When RAM contents are read, a parity check is performed in hardware and compared against the stored parity bits. This mechanism can detect single-bit errors.

6.3.24 FLASH1: FLASH Single Error Correction, Double Error Detection Mechanism

The FLASH in MSPM0Gx parts have ECC check bits associated with data bits. When FLASH contents are read, the expected code bits are computed in hardware and compared against the stored code bits. This mechanism can correct single-bit errors and can detect double-bit errors. The address bits are also included in the code computation to cover errors in address decoding logic.

6.3.25 FXBAR2: Periodic Software Readback of FLASH data

In this method, a known set of test data values are stored in FLASH and periodically this FLASH data is read and compared against expected data.

6.3.26 FXBAR3: Software Test of ECC Checker Logic

In this method, a few locations in FLASH can have corrupted data or ECC bits. These locations can be read to check that the ECC checker logic and interrupt/NMI generation is working properly.

6.3.27 FXBAR4: Write Protection of FLASH

MSPM0 MCUs implement a static write protection scheme to lock out user-defined sectors in the MAIN FLASH from unintended program and erase.

6.3.28 DAC2: DAC Test Using Internal ADC as DAC Output Checker

In this test method, DAC is setup per application configuration, the output of DAC is monitored using ADC. Anytime DAC values are updated, ADC has to be triggered, allowing for the DAC settling time and the ADC output to be checked against the expected value.

6.3.29 DAC3: DAC FIFO Underrun Interrupt

The FIFO in DAC has a built-in hardware check to detect FIFO underflows and set a flag. This can be used to check for unexpected runtime errors, which causes a FIFO underflow condition.

6.3.30 DMA2: Software Test of DMA Function

In this test mechanism, one of the DMA channels is dedicated to diagnostic test. This channel can be configured to do transfers of known data content from a fixed source (SRAM or FLASH) to a fixed destination (SRAM or CRC engine). Periodically the diagnostic channel can be triggered in software and the proper transfer of data can be checked in software.

6.3.31 DMA3: Software DMA Channel Test

In this method, DMA channels in use are periodically triggered after changing the source address to FLASH and destination address to SRAM. Known data is transferred from FLASH to SRAM and the data in SRAM is compared to the data in FLASH. In addition, a timer can be set up to monitor completion within a reasonable time.

6.3.32 DMA4: CRC Check of the Transferred Data

In this method, on a DMA transfer-done interrupt, data coherency can be checked. The data packet can include an expected CRC value. In software, CRC is computed and checked against the expected CRC.

6.3.33 GPIO1: GPIO Test Using Pin I/O Loopback

In this test method, the GPIO pin is set up with a known value and the status of the pin can be read back using the DIN register.

6.3.34 GPIO3: GPIO Multiple (Redundant) Outputs

If GPIO is controlling critical functions, the same outputs can be generated on two or more pins and an external function can be controlled based on majority voting. In the event a trigger mechanism (triggers can be from other peripherals, like GP timers, and so forth) is used to change the state of the GPIO pins, one subscriber can be used for one set of pins and another subscriber can be used for the other set of pins.

Note

The redundant pins must not be adjacent to prevent any coupling. TI recommends that the redundant pins are from two different GPIO instances.

6.3.35 TIM1: Test for PWM Generation

In this test, a second timer can be used to check that the PWM signal properties generate properly. To make this check, the PWM output from the main timer is looped to another TIMER. The measuring timer can then be then used to measure the pulse width and period of the PWM waveform.

Note

Use the same clock for both timers.

6.3.36 TIM3: Test for Fault Generation

In this method, when the timer is idle, the polarity of the fault pin can be changed (FCTL.FSENEXTx) to the alternate value. This triggers a fault that can be checked by reading the status register. This method can be used to check if the fault detection circuit is working as expected.

6.3.37 TIM4: Fault Detection to Take the PWMs to Safe State

If the timer outputs are driving power stages, the over(under)current or over(under)voltage signals from the power stage can be connected to fault pins. These fault signals can be monitored by the timer. The timer can be configured to take the output to a safe state by appropriately configuring the fault entry action. For more details, refer to the TIMx chapter of *MSPM0Lx22x Microcontrollers Technical Reference Manual*.

6.3.38 TIM5: Input Capture on Two or More Timer Instances

In the case where timing parameters of safety-critical inputs are measured, two or more spins can be connected to the same signal (redundancy), and the measured parameters (in different timer instances) can be compared to determine if the parameters are within the acceptable range. Input capture must be done on different timer instances. This covers faults in the clock divider as the measured timing parameters deviate from the expected values on the second instance.

6.3.39 TIM6: Timer Period Monitoring

The timer generates periodic interrupts. In the interrupt routine, the timer period can be monitored to check if there is any deviation.

6.3.40 I2C1: Software Test of I^{2}C Function Using Internal Loopback Mechanism

The I^{2}C modules can be placed into an internal loopback mode for diagnostic, or debug work, by setting the LPBK bit in the I^{2}C controller configuration I2Cx.MCR register. In loopback mode, the SDA and SCL signals from the controller part of the I^{2}C are tied to the SDA and SCL signals of the target part of the I^{2}C module to allow internal testing of the device without having to connect the I/Os.

This loopback mechanism can be used to transmit known data from transmit to receive. The I^{2}C configuration can be similar to the application configuration, regarding bit rate, FIFO usage, and so forth. The completion of the test can be timed to be within expected limits to detect any faults in the bit rate timing.

6.3.41 I2C3, SPI4, UART3, MCAN2: Information Redundancy Techniques Including End-to-End Safing

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

To provide diagnostic coverage for network elements outside the MCU (wiring harness, connectors, transceiver), end-to-end safety 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 verify the correctness of a transmission. These checksums, sequence counter, and timeout expectation (or time stamp) are applied, in addition to any protocol-level parity and checksums. As these are generated and evaluated by the software, at either end of the communication, the whole communication path is safed, resulting in end-to-end safing.

6.3.42 I2C4, SPI5, UART4: Transmission Redundancy

This is an application-level diagnostic where information is transferred several times in sequence using the same module instance and then compared. When the same data path is used for duplicate transmissions, transmission redundancy is only useful for detecting transient faults. The diagnostic coverage can be improved by sending inverted data during the redundant transmission.

6.3.43 I2C5, UART5: Timeout Monitoring

Periodic safety messages (an application-level technique) can be transmitted and received. When a safety message is not received in the expected time interval, the event can indicate loss of data communication. This monitoring is done continuously in the application.

6.3.44 I2C6: Test of CRC function

I^{2}C has a packet error check mechanism in the SMBUS mode. To check this function, messages are sent with corrupted data and a check is made to see if the CRC logic flags an error.

6.3.45 I2C7: Packet Error check in SMBUS Mode

When I^{2}C is used in SMBUS mode, the packet error check is done in hardware as part of the protocol. This is a fault-avoidance measure.

6.3.46 IOMUX2: IOMUX Coverage as Part of Other IP Safety Mechanisms

The majority of faults in the IOMUX logic are covered by the safety mechanisms of the blocks interfacing with the IOMUX (refer to Table 6-1).

Table 6-1. IOMUX Coverage

Signal Category	Diagnostics That Address IOMUX
Serial ports	I2C1, I2C3, SPI1, SPI4, SPI5, UART1, UART3, UART4, UART5,MCAN1, MCAN2
Timers	TIM1, TIM3, TIM4, TIM5
GPIO static pins	GPIO1, GPIO3
GPIO interrupts (periodic)	WDT
Clock inputs	SYSCTL1, SYSCTL3, SYSCTL9

6.3.47 SPI1: Software Test of SPI Function

The SPI can be placed into an internal loopback in controller mode by setting the LBM bit in the SPI.CTL1 register. In loopback mode, data transmitted on the TX output is received on the RX input. FIFO related faults can be diagnosed using this test. For checking the clock setting, the test execution time can be timed using timers.

6.3.48 SPI3: SPI Periodic Safety Message Exchange

An application-level check can be added to periodically send and receive a test message when SPI is in peripheral mode. In software, a timeout mechanism must be implemented to cover this.

6.3.49 UART1: Software Test of UART Function

The UART can be placed into an internal loopback mode for diagnostic, or debug work, by setting the LBE bit in the UART.CTL0 register. In loopback mode, data transmitted on the TXD output is received on the RXD input. Data received on the RXD IO pin is ignored when loopback is enabled. A timer can be used to check if the communication completed within the expected amount of time.

6.3.50 UART6: UART Error Flags

UART has error flags for frame error, break error, parity error, and overrun error.

6.3.51 SYSCTL1: MCLK Monitor

A digital clock monitor is provided to verify that MCLK is active. An MCLK fault is asserted by the MCLK monitor if there is no MCLK activity within a period of 1 to 12 LFCLK cycles. An MCLK fault is always considered fatal to the system and generates a BOOTRST.

The MCLK monitor can be enabled once LFCLK is configured and running. To enable the MCLK monitor, set the MCLKDEADCHK bit in the MCLKCFG register in SYSCTL. When enabled, the MCLK monitor runs in all operating modes except for STANDBY1 and SHUTDOWN.

6.3.52 SYSCTL2: HFCLK Start-Up Monitor

The HFXT takes time to start after being enabled. A start-up monitor is provided to indicate to the application software if the HFXT has successfully started, at which point the HFCLK can be selected to source a variety of system functions. The HFCLK start-up monitor also supports checking the HFCLK_IN digital clock input for a clock stuck fault.

When HFXT is started, or the HFCLK_IN is selected as the HFCLK source, the HFCLKGOOD and HFCLKOFF bits in the CLKSTATUS register in SYSCTL are cleared.

6.3.53 SYSCTL3: LFCLK Monitor

A low-power analog circuitry clock monitor is provided to verify that LFCLK is running when LFCLK is not sourced internally (for example, in cases when LFCLK is sourced from LFXT or LFCLK_IN and not from LFOSC). The LFCLK monitor is only intended to check for clock stuck faults. The monitor is not intended to be used to verify that the frequency of LFCLK is within a specific tolerance.

6.3.54 SYSCTL6: SYSPLL Start-Up Monitor

The SYSPLL takes time to start and settle after being enabled. A start-up monitor is provided to indicate to the application software if the SYSPLL has successfully started, at which point the clock outputs from the SYSPLL can be selected to source a variety of system functions.

When the SYSPLL is started, the SYSPLLGOOD and SYSPLLOFF bits in the CLKSTATUS register in SYSCTL are cleared. After the start-up and settling time has expired, the SYSPLL status is tested. If the SYSPLL started successfully, the SYSPLL start-up monitor asserts the SYSPLLGOOD bit in the CLKSTATUS register and the SYSPLLGOOD interrupt is also asserted. If the SYSPLL did not start within the specified time, the SYSPLLOFF bit is set, indicating that the SYSPLL was dead at start-up.

6.3.55 SYSCTL8: Brownout Reset (BOR) Supervisor

The brownout reset (BOR) supervisor monitors the external supply (VDD) and asserts or deasserts a BOR violation to SYSCTL.

6.3.56 SYSCTL9: FCC Counter Logic to Calculate Clock Frequencies

The frequency clock counter (FCC) enables flexible in-system testing and calibration of a variety of oscillators and clocks on the device. The FCC counts the number of clock periods seen on the selected source clock within a known trigger period (derived from a secondary reference source) to provide an estimation of the frequency of the source clock.

6.3.57 SYSCTL10: External Voltage Monitor

Use an external voltage monitor on VCORE PAD to monitor the LDO output.

6.3.58 SYSCTL11: Boot Process Monitor

In the event of a boot fail during execution of the boot configuration routine (BCR), SYSCTL asserts a BOOTRST to re-attempt a successful boot. A boot fail can be caused by any of the following:

1. Boot configuration data integrity error (this can be used for BOOTROM/ FLASH)
2. Device trim integrity error (this can be used for FLASH)
3. BCR timeout (BCR takes significantly longer than expected to complete for any other reason, this can be used for BOOTROM)

6.3.59 SYSCTL14: Brownout Voltage Monitor

The brownout circuit can be configured to monitor the external supply (VDD) above the BOR reset level. If tripped, the monitor generates a non-maskable interrupt (NMI) event and reconfigures the BOR circuit to be a BOR supervisor reset. This allows software to either warn the user of a depleting battery, or initiate a graceful shutdown of the system application, before the BOR circuit issues a BOR supervisor reset.

6.3.60 SYSCTL15: External Voltage Monitor

An external voltage supervisor can be used to monitor the power supplies (main supply and the internal LDO output).

6.3.61 SYSCTL16: External Watchdog Timer

An external watchdog monitor can be employed to check (sanity check) the working of MCU. In case watchdog monitor flags an error, the system software can take the required action.

6.3.62 MCAN1: Software test of function using I/O Loopback

The MCAN module can be set into internal loopback mode by programming MCAN_TEST.LBCK and MCAN_CCCR.MON bits to 1. The internal loopback mode is used for a hot self-test. The hot self-test allows the

MCAN module to be tested without affecting a running CAN system connected to the TX and RX pins. In this mode, the RX pin is disconnected from the MCAN module and the TX pin is held recessive.

6.3.63 MCAN4: SRAM ECC

The message RAM in the MCAN module stores computed check bits (ECC bits) for each word of data stored. An ECC memory always protects against single-bit errors in the data. Address decode logic for the memory is not covered by the memory ECC logic. There is an ECC aggregator module inside the module that aggregates status from the ECC memories into a single interrupt to the host. The aggregator also supports software readable status of ECC single- and double-bit errors and associated info such as RAM address and data bit (or bits) that are in error.

6.3.64 MCAN5: Software Test of ECC Check Logic

Testing the functionality of the ECC detection logic is possible by forcing an ECC error into the data output from the memory and checking if the ECC detection logic reports an error. A shared module interface, which is referred to as an ECC aggregator, provides the system integrator with access to configure the logic and force errors. Reporting of forced errors uses the same mechanism that reports unforced errors. Error response, diagnostic testability, and any necessary software requirements are defined by the software implemented by the system integrator

6.3.65 MCAN6: MCAN Timeout Function

MCAN implements a timeout counter that is programmed during the INIT phase for the module. The timeout function can be continuous (preset by writing to TOCC.TOP on a periodic basis before the timeout function expires) or associated with Tx, Rx0, or Rx1 FIFOs (a FIFO empty presets the counter and first push starts the down counting). A CAN system implementing a periodic messaging can use the timeout diagnostic to ascertain the presence of a system heart beat.

6.3.66 MCAN7: MCAN Timestamp Function

MCAN implements a timestamp for received and transmitted messages. An external timestamp counter is required for CAN FD messages. The timestamp counter provided by MCAN is equipped with a prescaler for tradeoff between resolution and wraparound period. The timestamp counter value is stored in the message buffer for each transmitted or received message. Software can perform sanity checks on messages to determine if the messages have been sent in the order expected by the system as a diagnostic. For example, multiple messages with the same timestamp (taking into consideration the wraparound time) are not expected as the CAN protocol can carry one message at a time. End-to-end safing that includes numbering the messages can be used to indicate linear incrementing timestamps that software can verify.

6.3.67 CRC: CRC Checker

The hardware CRC module can be used to speed up some of the safety mechanisms involving CRC computation. Any fault within the CRC logic is detected when the CRC signature is incorrect.

6.3.68 EVENT2: Interrupt Connectivity Check

The interrupt connectivity between the CPU and the peripheral blocks can be checked by writing to the ISET (interrupt set) register in the peripheral block and checking that the corresponding flag is set in NVIC. Once the check is complete, the interrupt flag can be cleared by writing to the ICLR register in the peripheral. The flag in the NVIC can also be cleared by software.

7 An In-Context Look at This Safety Element out of Context

This section contains a Safety Element out of Context (SEooC) analysis of the MSPM0G component. Texas Instruments has made assumptions on the typical safety system configurations using this component. System level safety analysis is the responsibility of the developer of these systems and not Texas Instruments. As such, this section is intended to be informative only to help explain how to use the features of this component to assist the system designer in integrating this component into a system. This section describes example use cases and goes into more detail on how to identify hardware parts that are critical to the safety function and how to configure the associated functional safety mechanisms. Please note that the system designer may choose to use this component in safety-relevant systems beyond those mentioned below.

7.1 System Functional Safety Concept Examples

Person Occupancy Detect System (PODS)

In this system, MSPM0G interfaces with pressure sensors, processes the pressure sensor data to determine if the seat is occupied or not, and communicates this information to the host CPU on a CAN bus. At the system level, the information regarding seat occupancy is used to conditionally enable the airbag system associated with a seat, based on whether the seat is occupied or not.

At a system level, to guard against failures in the sensors or sensor data processing (MSPM0G), redundant sensors can be added and MSPM0G can monitor the data from redundant sensors to check for any faults and communicate to the host CPU. The host CPU can display a warning to the driver about the failure.

A Summary of Recommended Functional Safety Mechanism Usage

Appendix A summarizes the functional safety mechanisms present in hardware or recommend for implementation in software or at the system level as described in Section 5. Table A-1 describes each column in Table A-2 and gives examples of what content can appear in each cell.

Table A-1. Legend of Functional Safety Mechanisms

Functional Safety Mechanism	Description
TI Safety Mechanism Unique Identifier	A unique identifier assigned to this safety mechanism for easier tracking.
Safety Mechanism Name	The full name of this safety mechanism.
Safety Mechanism Category	**Safety Mechanism** - This test provides coverage for faults on the primary function. It may also provide coverage on another safety mechanism. **Test for Safety Mechanism** - This test provides coverage for faults of a safety mechanism only. It does not provide coverage on the primary function. **Fault Avoidance** - This is typically a feature used to improve the effectiveness of a related safety mechanism.
Safety Mechanism Type	Can be either hardware, software, a combination of both hardware and software, or system. See Section 6.2 for more details.
Safety Mechanism Operation Interval	The timing behavior of the safety mechanism with respect to the test interval defined for a functional safety requirement / functional safety goal. Can be either continuous, or on-demand. **Continuous** - the safety mechanism constantly monitors the hardware-under-test for a failure condition. **Periodic or On-Demand** - the safety mechanism is executed periodically, when demanded by the application. This includes Built-In Self-Tests that are executed one time per drive cycle or once every few hours.
Test Execution Time	Time period required for the safety mechanism to complete, not including error reporting time. Note: Certain parameters are not set until there is a concrete implementation in a specific component. When component specific information is required, the component data sheet should be referenced. Note: For software-driven tests, the majority contribution of the Test Execution Time is often software implementation-dependent.
Action on Detected Fault	The response that this safety mechanism takes when an error is detected. Note: For software-driven tests, the Action on Detected Fault may depend on software implementation.
Time to Report	Typical time required for safety mechanism to indicate a detected fault to the system. Note: For software-driven tests, the majority contribution of the Time to Report is often software implementation-dependent.

Table A-2. Summary of Functional Safety Mechanisms

TI Safety Mechanism Unique Identifier	Safety Mechanism Name	Safety Mechanism Category	Safety Mechanism Type	Safety Mechanism Operation Interval	Test Execution Time	Action on Detected Fault	Time to Report
ADC1	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
ADC2	ADC software test of functionality	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
WDT	Windowed watchdog event	Safety Mechanism	Hardware + Software	Continuous	Application dependent	Reset the device	Application dependent
ADC3	ADC Trigger overflow	Safety Mechanism	Hardware	Continuous	Application dependent	Reset the device	Application dependent
ADC4	ADC window comparator	Safety Mechanism	Hardware	Continuous	Application dependent	Reset the device	Application dependent
ADC5	Test of window comparator	Test for Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
ADC6	ADC trigger/output plausibility check	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
COMP1	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
COMP2	DAC to COMP Loopback	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
COMP3	External pin input to COMP	Safety Mechanism	System Level Diagnostic	Periodic / On Demand	Application dependent	Reset the device	Application dependent
COMP4	Comparator Hysteresis	Safety Mechanism	Hardware	Continuous	Application dependent	Reset the device	Application dependent
COMP5	Redundant comparator	Safety Mechanism	Hardware + Software	Continuous	Application dependent	Reset the device	Application dependent
CPU1	ARM Software Test Library	Safety Mechanism	Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
CPU2	Write/Read back of data to different regions of memory to detect faults in the bus interconnect components.	Safety Mechanism	Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
SYSCTL11	Boot Process Timeout	Safety Mechanism	Hardware + Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
CPU3	Software redundancy	Safety Mechanism	Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
CPU4	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent

Table A-2. Summary of Functional Safety Mechanisms (continued)

TI Safety Mechanism Unique Identifier	Safety Mechanism Name	Safety Mechanism Category	Safety Mechanism Type	Safety Mechanism Operation Interval	Test Execution Time	Action on Detected Fault	Time to Report
CRC	CRC Checker	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
CRC1	Periodic Software Read Back of Static Configuration Registers	Test for Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
DAC1	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
DAC2	DAC to ADC Loopback	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
DAC3	FIFO Underrun interrupt	Safety Mechanism	Hardware	Continuous	Application dependent	Reset the device	Application dependent
DMA1	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Software	Periodic / On-Demand	Application dependent	Reset the device	Application dependent
DMA2	Software DMA transfer test	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
DMA3	Software DMA channel test	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
DMA4	CRC check of the transferred data.	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
EVENT1	Periodic Software Readback of Static Configuration Registers	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
EVENT2	Interrupt connectivity checker	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
GPIO3	GPIO multiple (redundant) outputs	Safety Mechanism	Hardware + Software	Continuous	Application dependent	Reset the device	Application dependent
FXBAR2	Periodic Software Read Back of FLASH data	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
FLASH1	FLASH ECC checker	Safety Mechanism	Hardware	Continuous	Application dependent	Reset the device	Application dependent
FXBAR3	Software test of ECC checker logic.	Test for Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
FXBAR4	Write protection of FLASH	Safety Mechanism	Hardware	Continuous	Application dependent	Reset the device	Application dependent
GPIO1	Software test of function using I/O loopback	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent

Table A-2. Summary of Functional Safety Mechanisms (continued)

TI Safety Mechanism Unique Identifier	Safety Mechanism Name	Safety Mechanism Category	Safety Mechanism Type	Safety Mechanism Operation Interval	Test Execution Time	Action on Detected Fault	Time to Report
GPIO2	Periodic Software Readback of Static Configuration Registers	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
TIM1	Test for basic PWM generation	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
TIM2	Periodic Software Read Back of IP Static Configuration Registers	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
TIM3	Test for fault generation	Test for Safety Mechanism	System	Periodic / On Demand	Application dependent	Reset the device	Application dependent
TIM4	Fault detection to take the PWMs to safe state	Safety Mechanism	Hardware + Software	Continuous	Application dependent	Reset the device	Application dependent
TIM5	Input capture on two or more timer instances	Safety Mechanism	Hardware	Continuous	Application dependent	Reset the device	Application dependent
TIM6	Timer period monitoring.	Safety Mechanism	Hardware + Software	Continuous	Application dependent	Reset the device	Application dependent
I2C1	Software test of function using I/O loopback	Safety Mechanism	Hardware + Software	Periodic	Application dependent	Reset the device	Application dependent
I2C2	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Software	Periodic	Application dependent	Reset the device	Application dependent
I2C3	Information Redundancy Techniques Including End-to-End Safing	Safety Mechanism	Software	Periodic	Application dependent	Reset the device	Application dependent
I2C4	Transmission redundancy	Safety Mechanism	Software	Periodic	Application dependent	Reset the device	Application dependent
I2C5	Timeout monitoring	Safety Mechanism /Test of Safety Mechanism	Software	Continuous	Application dependent	Reset the device	Application dependent
I2C6	Test of CRC function	Safety Mechanism /Test of Safety Mechanism	Hardware + Software	Periodic	Application dependent	Reset the device	Application dependent
I2C7	Packet error check in SMBUS mode.	Fault Avoidance	N/A	Continuous	Application dependent	Reset the device	Application dependent
IOMUX1	Periodic Software Readback of Static Configuration Registers	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent

Table A-2. Summary of Functional Safety Mechanisms (continued)

TI Safety Mechanism Unique Identifier	Safety Mechanism Name	Safety Mechanism Category	Safety Mechanism Type	Safety Mechanism Operation Interval	Test Execution Time	Action on Detected Fault	Time to Report
IOMUX2	IOMUX coverage as part of other IP safety mechanisms.	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
MCAN1	Software test of function using I/O loopback	Safety Mechanism	Hardware + Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
MCAN3	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
MCAN2	Information Redundancy Techniques Including End-to-End Safing	Safety Mechanism	Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
MCAN4	SRAM ECC	Safety Mechanism	Hardware	Periodic/On Demand	Application dependent	Reset the device	Application dependent
MCAN5	Software Test of ECC Logic	Test of Safety Mechanism	Hardware + Software	Continuous	Application dependent	Reset the device	Application dependent
MCAN6	Timeout on FIFO Activity	Safety Mechanism	Hardware + Software	Continuous	Application dependent	Reset the device	Application dependent
MCAN7	Timestamp Consistency checks	Safety Mechanism	Hardware / Software	Continuous	Application dependent	Reset the device	Application dependent
OA1	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
OA2	OA test using ADC	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
OA3	ADC monitoring of OA output	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SPI1	Software test of function using I/O loopback	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SPI2	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SPI3	SPI periodic Safety Message checks	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SPI4	Information Redundancy Techniques Including End-to-End Safing	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SPI5	Transmission redundancy	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent

Table A-2. Summary of Functional Safety Mechanisms (continued)

TI Safety Mechanism Unique Identifier	Safety Mechanism Name	Safety Mechanism Category	Safety Mechanism Type	Safety Mechanism Operation Interval	Test Execution Time	Action on Detected Fault	Time to Report
SYSCTL1	MCLK monitor	Safety Mechanism	Hardware + Software	Continuous	Application dependent	Reset the device	Application dependent
SYSCTL2	HFCLK Startup monitor	Fault Avoidance	N/A	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SYSCTL3	LFCLK Monitor	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SYSCTL5	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SYSCTL6	SYSPLL Startup monitor	Fault Avoidance	N/A	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SYSCTL8	Brownout Reset (BOR) Supervisor	Safety Mechanism	Hardware	Periodic / On-Demand	Application dependent	Reset the device	Application dependent
SYSCTL9	Clock frequency measurement	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SYSCTL10	External voltage monitor	Safety Mechanism	System Level Diagnostic	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SYSCTL14	Brownout Voltage Monitor	Safety Mechanism	Hardware	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SYSCTL15	External Voltage Supervisor	Safety Mechanism	System Level Diagnostic	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SYSCTL16	External Watch dog timer	Safety Mechanism	System Level Diagnostic	Periodic / On Demand	Application dependent	Reset the device	Application dependent
SYSMEM1	Software read of memory DMA	Safety Mechanism	Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
SYSMEM2	Software read of memory CPU	Safety Mechanism	Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
SYSMEM3	Parity logic test	Test for Safety Mechanism	Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
SYSMEM4	RAM Parity	Safety Mechanism	Hardware	Continuous	Application dependent	Reset the device	Application dependent
UART1	Software test of function using I/O loopback	Safety Mechanism	Hardware + Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
UART2	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent

Table A-2. Summary of Functional Safety Mechanisms (continued)

TI Safety Mechanism Unique Identifier	Safety Mechanism Name	Safety Mechanism Category	Safety Mechanism Type	Safety Mechanism Operation Interval	Test Execution Time	Action on Detected Fault	Time to Report
UART3	Information Redundancy Techniques Including End-to-End Safing	Safety Mechanism	Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
UART4	Transmission redundancy	Safety Mechanism	Software	Periodic/On Demand	Application dependent	Reset the device	Application dependent
UART5	Timeout monitoring	Safety Mechanism/Test for Safety Mechanism	Software	Continuous	Application dependent	Reset the device	Application dependent
UART6	UART error flags	Safety Mechanism	Hardware	Continuous	Application dependent	Reset the device	Application dependent
REF1	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
REF2	VREF to ADC Reference input	Safety Mechanism	Hardware + Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
WDT1	Periodic Software Read Back of Static Configuration Registers	Test for Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
WDT2	WWDT counter check	Test for Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent
WDT3	WWDT software test	Test for Safety Mechanism	Software	Periodic / On Demand	Application dependent	Reset the device	Application dependent

9 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision * (March 2024) to Revision A (November 2024) **Page**

- Removed statements regarding ASIL-B compliance, re-worded to state the development process is quality managed.......... 3
- Updated the part list.......... 3
- Removed *Advanced Information* note.......... 3
- Added addiitonal information related to the safety mechanisms.......... 20