Functional Safety Information

Functional Safety Manual for MSPM0G

Table of Contents

1 Introduction ... 3
2 MSPM0G Hardware Component Functional Safety Capability ... 4
3 Development Process for Management of Systematic Faults ... 5
3.1 TI New-Product Development Process ... 5
3.2 TI Functional Safety Development Process ... 6
4 MSPM0G Component Overview ... 7
4.1 Targeted Applications ... 8
4.2 Hardware Component Functional Safety Concept ... 8
4.3 Functional Safety Constraints and Assumptions ... 8
5 Description of Hardware Component Parts ... 9
5.1 ADC ... 9
5.2 Comparator ... 10
5.3 DAC ... 10
5.4 OPA ... 11
5.5 CPU ... 12
5.6 RAM ... 12
5.7 FLASH ... 12
5.8 GPIO ... 13
5.9 DMA ... 13
5.10 SPI ... 14
5.11 I2C ... 14
5.12 UART ... 15
5.13 Timers (TIMx) ... 16
5.14 Power Management Unit (PMU) ... 17
5.15 Clock Module (CKM) ... 17
5.16 CAN-FD ... 18
6 MSPM0G Management of Random Faults ... 19
6.1 Fault Reporting ... 19
6.2 Functional Safety Mechanism Categories ... 19
6.3 Description of Functional Safety Mechanisms ... 20
7 An In-Context Look at This Safety Element out of Context ... 25
7.1 System Functional Safety Concept Examples ... 25
A Summary of Recommended Functional Safety Mechanism Usage (Optional) ... 26
B Distributed Developments ... 30
B.1 How the Functional Safety Lifecycle Applies to TI Functional Safety Products ... 30
B.2 Activities Performed by Texas Instruments ... 30
B.3 Information Provided ... 31

List of Figures

Figure 3-1. TI New-Product Development Process ... 5
Figure 4-1. MSPM0G Block Diagram ... 7
Figure 4-2. MSPM0G Typical Application ... 8

List of Tables

Table 3-1. Functional Safety Activities Overlaid on top of TI's Standard Development Process ... 6
Table 5-1. UART Features ... 15
Table 5-2. TIMx Configurations ... 16
Table A-1. Legend of Functional Safety Mechanisms ... 26
Table A-2. Summary of Functional Safety Mechanisms ... 26
Table B-1. Activities Performed by Texas Instruments versus Performed by the customer ... 30

Table B-2. Product Functional Safety Documentation..... 31

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:

- MSPM0G3105
- MSPM0G3106
- MSPM0G3107
- MSPM0G3505
- MSPM0G3506
- MSPM0G3507

This 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 in the *Functional Safety Analysis Report* and is not repeated in this document:

- Summary of failure rates (FIT) of the component
- Summary of functional safety metrics of the hardware component for targeted standards
- 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
- Assumptions used in the calculation of functional safety metrics

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

- Results of assessments of compliance to targeted standards

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.

ADVANCE INFORMATION for preproduction products; subject to change without notice.

Trademarks

TI E2E™ is a trademark of Texas Instruments.
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:

- Was developed as a functional Safety Element out of Context (SEooC)
- Was developed according to the relevant requirements of ISO 26262:2018
- Achieves systematic integrity of ASIL-B
- Includes sufficient functional safety mechanisms for random fault integrity requirements of [ASIL-B]

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. Components being designed for functional safety applications will additionally follow the requirements of TI's functional safety development process, which is described in Section 3.2.

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

Section 3.1 shows the standard process.

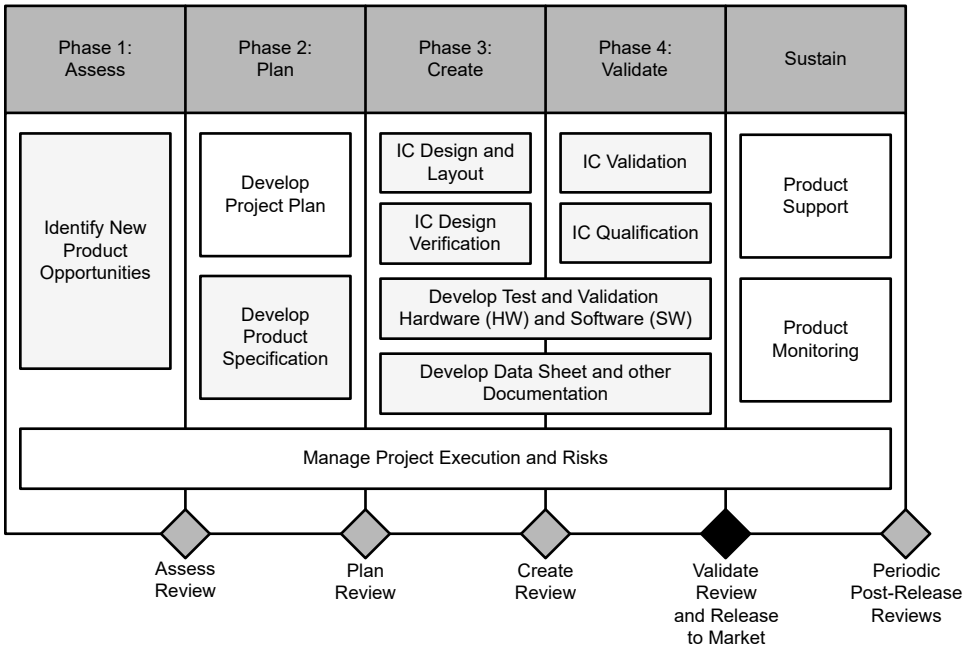

Figure 3-1. TI New-Product Development Process

3.2 TI Functional Safety Development Process

The TI functional safety development flow derives from ISO 26262 and IEC 61508 a set of requirements and methodologies to be applied to semiconductor development. This flow is combined with TI's standard new product development process to develop TI functional safety components. The details of this functional safety development flow are described in the TI internal specification - TI Functional Safety Hardware.

Key elements of the TI functional safety-development flow are as follows:

- Assumptions on system level design, functional safety concept, and requirements based on TI's experience with components in functional safety applications
- Qualitative and quantitative functional safety analysis techniques including analysis of silicon failure modes and application of functional safety mechanisms
- Base FIT rate estimation based on multiple industry standards and TI manufacturing data
- Documentation of functional safety work products during the component development
- Integration of lessons learned through multiple functional safety component developments, functional safety standard working groups, and the expertise of TI customers

Functional Safety Activities Overlaid on top of TI's Standard Development Process lists these functional safety development activities which are overlaid atop the standard development flow in Figure 3-1.

Refer to Appendix B for more information about which functional safety lifecycle activities TI performs.

The customer facing work products derived from this TI functional safety process are applicable to many other functional safety standards beyond ISO 26262 and IEC 61508.

Table 3-1. Functional Safety Activities Overlaid on top of TI's Standard Development Process

Assess	Plan	Create	Validate	Sustain and End-of-Life
Determine if functional safety process execution is required	Define component target SIL/ASIL capability	Develop component level functional safety requirements	Validate functional safety design in silicon	Document any reported issues (as needed)
Nominate a functional safety manager	Generate functional safety plan	Include functional safety requirements in design specification	Characterize the functional safety design	Perform incident reporting of sustaining operations (as needed)
End of Phase Audit	Verify the functional safety plan	Verify the design specification	Qualify the functional safety design (per AEC-Q100)	Update work products (as needed)
	Initiate functional safety case	Start functional safety design	Finalize functional safety case	
	Analyze target applications to generate system level functional safety assumptions	Perform qualitative analysis of design (i.e. failure mode analysis)	Perform assessment of project	
	End of Phase Audit	Verify the qualitative analysis	Release functional safety manual	
		Verify the functional safety design	Release functional safety analysis report	
		Perform quantitative analysis of design (i.e. FMEDA)	Release functional safety report	
		Verify the quantitative analysis	End of Phase Audit	
		Iterate functional safety design as necessary		
		End of Phase Audit		

4 MSPM0G Component Overview

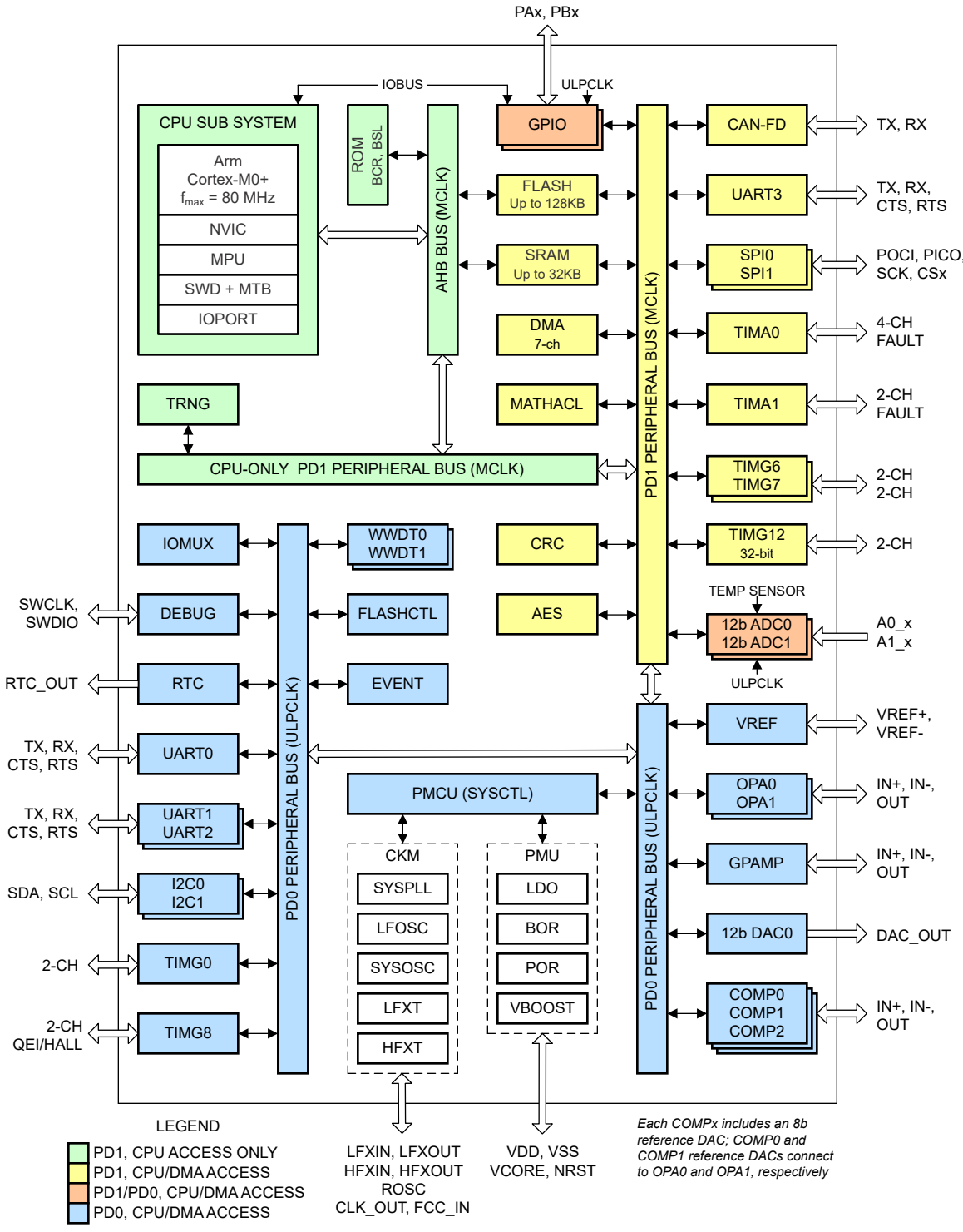

Figure 4-1. MSPM0G Block Diagram

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 method is also used to meet the related requirements of IEC 61508 at the semiconductor 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.

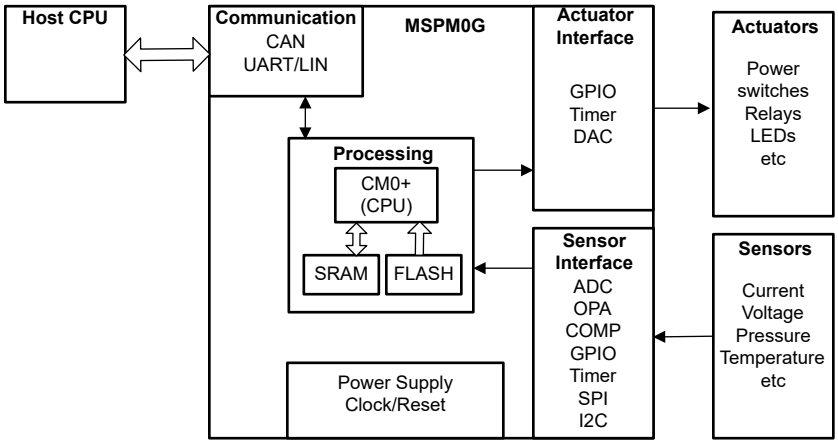

Figure 4-2. MSPM0G Typical Application

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 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 system integrator shall follow all requirements in the component data sheet.

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. This section is intended to provide additional details about the assignment of functional safety mechanisms that can be found in the Safety Analysis Report. The content in this section is also summarized in Appendix A.

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

For more details, see the ADC 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):

- ADC1
- ADC2
- ADC3
- ADC4

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. It 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
- Support output wake up device from all low power modes
- Output connected to 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.

For more details, see the COMP 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):

- COMP1
- COMP2

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 and it 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)

For more details, see the DAC 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):

- 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/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.

For more details, see the OPA 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):

- OA1
- OA2

5.5 CPU

The CPU sub system (MCPUSS) implements an ARM Cortex-M0+ CPU, an instruction pre-fetch/cache, a system timer, a memory protection unit, and interrupt management features. The ARM Cortex-M0+ is a cost-optimized, 32-bit CPU which delivers high performance and low power to embedded applications. Key features of the CPU Sub System 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 via ARM single-cycle IO port
- Pre-fetch logic to improve sequential code execution, and I-cache with 4 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

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 it 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):

- 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/erase cycles on the lower 32kB of the flash memory, with up to 10,000 program/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, see 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

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/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

For more details, see the GPIO 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):

- GPIO1
- GPIO2

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 support 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

For more details, see the DMA 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):

- DMA1
- DMA2

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 32 16Mbits/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

For more details, see the SPI chapter of the 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):

- SPI1
- SPI2
- SPI3

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

For more details, see the I2C chapter of the 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):

- I2C1
- I2C2

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 DAM data transfer
- Support transmit and receive loopback mode operation
- See 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	-	-

For more details, see the UART 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):

- UART1
- UART2

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/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 ensure 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	–

For more details, see the TIMx 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):

- TIM1

- TIM2

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 itself 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

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 (4 to 48MHz)
5. **SYSPLL**: system phase locked loop with three outputs (32 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

For more details, see the CKM 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):

- 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

For more details, see the CAN-FD 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):

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

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,MCAN2: Periodic read of static configuration registers

Periodic software read of static configuration registers is one of the methods listed in ISO26262. This mechanism involves the software verifying the values of static configuration registers against a known reference. One of the methods is to compute a CRC signature for the static configuration register values and periodically computing the CRC and checking against expected CRC signature.

6.3.2 ADC2: Software test of function

In this method, internal DAC output is used to setup a known voltage on the ADC input channel and the other parameters of the ADC like the sampling time, reference sources, sequencer modes are setup similar to the actual application. 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/conversion operation is in progress, TOVIFG flag will be 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 which can be used to check if the input signal is within predefined threshold values set by software. The ADC result that goes into MEMRES or FIFO is what gets checked against the threshold values of the window comparator.

Based on the comparison it 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/or resolution configuration.

6.3.5 OA2: Test of OA using internal DAC as a driver

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

6.3.6 COMP2: Software test of Comparator using internal DAC

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

6.3.7 WDT: Windowed watch dog timer

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

For more details, see the WWDT chapter of the *MSPM0 L-Series 32-MHz Microcontrollers Technical Reference Manual*.

6.3.8 CPU1: CPU test using software test library

The CortexM0+ CPU and MPU will be tested using the ARM Software Test Library (STL).

6.3.9 CPU2: Software test of CPU data busses

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

6.3.10 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, parity check is performed in hardware and compared against the stored parity bits. This mechanism can detect single bit errors.

6.3.11 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.

6.3.12 DAC2: DAC test using internal ADC as DAC output checker

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

6.3.13 DAC3: DAC FIFO underrun interrupt

The FIFO in DAC has an inbuilt hardware check to detect FIFO underflows and set a flag. This can be utilized to check for unexpected runtime errors causing a FIFO underflow.

6.3.14 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/FLASH) to a fixed destination (SRAM/CRC engine). Periodically the diagnostic channel can be triggered in software and the proper transfer of data can be checked in software.

6.3.15 GPIO1: GPIO test using pin IO loopback

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

6.3.16 TIM1: Test for PWM generation

In this test, a second timer can be used to check the generated PWM signal properties. In order to do that, 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 fro both Timers

6.3.17 I2C1: Software test of I2C function using internal loopback mechanism

The I^2C modules can be placed into an internal loopback mode for diagnostic or debug work by setting the LPBK bit in the I^2C controller configuration I2Cx.MCR register. In loopback mode, the SDA and SCL signals from the controller part of the I^2C are tied to the SDA and SCL signals of the target part of the I^2C 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 I2C configuration can be similar to the application configuration with respect to the bit rate, FIDO usage etc. The completion of the test can be timed to be within expected limits to detect any faults in the bit rate timing.

6.3.18 SPI1 : Software test of SPI function

The SPI can be placed into an internal loopback in Controller mode for 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. Application can use FIFO to check out FIFO behavior. For checking the clock setting, the test execution time can be timed using timers.

6.3.19 SPI3: SPI periodic safety message exchange

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

6.3.20 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 will be ignored when loopback is enabled. Use an SoC timer/ watch dog to indicate if the communication is completed within the expected amount of time.

6.3.21 SYSCTL1: MCLK monitor

A digital clock monitor is provided to ensure that MCLK is alive. An MCLK fault is asserted by the MCLK monitor if there is no MCLK activity within a period of 1-12 LFCLK cycles. An MCLK fault is always considered fatal to the system and will generate 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.

6.3.22 SYSCTL2: HFCLK startup monitor

The HFXT takes time to start after being enabled. A startup 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 startup 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.

6.3.23 SYSCTL3: LFCLK monitor

A low-power analog circuitry clock monitor is provided to ensure that LFCLK is running when it 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. It is not intended to be used to verify that the frequency of LFCLK is within a specific tolerance.

6.3.24 SYSCTL4: RTC monitor

RTC can be used to check clock accuracy periodically.

6.3.25 SYSCTL6: SYSPLL startup monitor

The SYSPLL takes time to start and settle after being enabled. A startup 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 startup/settling time has expired, the SYSPLL status is tested. If the SYSPLL started successfully, the SYSPLL startup monitor will assert the SYSPLLGOOD bit in the CLKSTATUS register and

the SYSPLLGOOD interrupt will also be asserted. If the SYSPLL did not start within the specified time, the SYSPLLOFF bit will be set, indicating that the SYSPLL was dead at startup.

6.3.26 SYSCTL8: Brownout Reset (BOR) Supervisor

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

6.3.27 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.

6.3.28 SYSCTL10: External voltage monitor

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

6.3.29 SYSCTL11: Boot process monitor

In the event of a boot fail during execution of the boot configuration routine (BCR), SYSCTL will assert 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)

6.3.30 SYSCTL12: TRIM bits parity protection

The SoC has a couple of registers (SHUTDNSTOREx) which retain their content during SHUTDOWN mode. Since there is no active state monitoring by the CPU possible in SHUTDOWN mode, a parity protection has been added to these registers. On the detection of a parity error, the PMU will issue a POR reset request. If the device was in SHUTDOWN mode, the device will not reset right away. The device will wakeup with a next wakeup event and the PMU will issue the POR request and clear the SHUTDNSTOREx registers. The SW will see this wakeup as POR reset, rather then a wakeup from SHUTDOWN mode, which indicates the content in the SHUTDNSTOREx registers has been reset.

6.3.31 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 will generate a none-maskable-interrupt (NMI) event and reconfigure the BOR circuit to be a BOR supervisor reset. This allows SW to either warn the user of a depleting battery or initiate a graceful shutdown of the system application, before the BOR circuit will issue a BOR supervisor reset.

6.3.32 SYSCTL15: External voltage monitor

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

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

The MCAN module can be set into internal loop back mode by programming MCAN_TEST.LBCK and MCAN_CCCR.MON bits to 1. The internal loop back mode is used for a Hot Selftest. The Hot Selftest 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.

6.3.34 MCAN4: SRAM ECC

The message RAM in MCAN module stores computes 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/double-bit errors and associated info such as RAM address and data bit(s) that are in error.

6.3.35 MCAN5: Software test of ECC check logic

It is possible to test the functionality of ECC detection logic by forcing an ECC error into the data output from the memory, and seeing 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 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

6.3.36 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 it 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.

6.3.37 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. E.g. multiple messages with the same time stamp (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.

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, does processing of 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 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 (Optional)

Appendix Asummarizes 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 could 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	Software test for periodic read of static configured MMRs	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
ADC2	ADC sample and conversion test	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
WDT	Watchdog Timeout Event	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
ADC3	ADC Trigger overflow	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
ADC4	Window comparator	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent t

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
COMP1	Software Read Back of Written Configuration	Safety Mechanism	Hardware/ 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
CPU1	ARM STL	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
DAC1	Periodic Software Read Back of Written Configuration	Safety Mechanism	Hardware/ 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 Under-run interrupt	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
DMA1	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
DMA2	Software test	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
FXBAR1	Use hardware redundancy by accessing same flash location by CPU and DMA	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
FXBAR2	Periodic Software Read Back of FLASH data	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	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
GPIO2	Periodic Software Readback of Static Configuration Registers	Safety Mechanism	Hardware/ 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 Static Configuration Registers	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
I2C1	Software test of function using I/O loopback	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
I2C2	Periodic Software Read Back of Static Configuration Registers	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
IOMUX1	Periodic Software Readback of Static Configuration Registers	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

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
MCAN2	Information Redundancy Techniques Including End-to-End Safing	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	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
MCAN4	SRAM ECC	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
MCAN5	Software Test of ECC Logic	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
MCAN6	Timeout on FIFO Activity	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
MCAN7	Timestamp Consistency checks	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
OA1	Software Read Back of Written Configuration	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
OA2	DAC8(COMPDAC) to OA and then to ADC Loopback	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	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
SPI3	SPI PERIODIC Safety Message check	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
SYSCTL1	MCLK monitor	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
SYSCTL2	HFCLK Startup monitor	Safety Mechanism	Hardware/ Software	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
SYSCTL4	RTC 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	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
SYSCTL6	SYSPLL Startup monitor	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
SYSCTL8	Brownout Reset (BOR) Supervisor	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
SYSCTL9	FCC counter logic to calculate clock frequencies	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
SYSCTL10	Extrenal voltage monitor	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
SYSCTL11	Boot process monitor	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
SYSCTL12	Parity protection	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
SYSCTL13	SYSCTL3V State machine	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
SYSCTL14	Brownout Voltage Monitor	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
SYSCTL15	External Voltage Supervisor	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
SYSMEM4	RAM Parity	Safety Mechanism	Hardware/ Software	Periodic/On-Demand	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	Hardware/ Software	Periodic/On-Demand	Application dependent	Reset the device	Application dependent
REF1	Periodic Software Read Back of static configuration registers.	Safety Mechanism	Hardware/ 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

B Distributed Developments

A Development Interface Agreement (DIA) is intended to capture the agreement between two parties towards the management of each party's responsibilities related to the development of a functional safety system. TI functional safety components are typically designed for many different systems and are considered to be Safety Elements out of Context (SEooC) hardware components. The system integrator is then responsible for taking the information provided in the hardware component safety manual, safety analysis report and safety report to perform system integration activities. Because there is no distribution of development activities, TI does not accept DIAs with system integrators.

TI functional safety components are products that TI represents, promotes or markets as helping customers mitigate functional safety related risks in an end application and/or as compliant with an industry functional safety standard or FS-QM. For more information about TI functional safety components, go to TI.com/functionalsafety.

B.1 How the Functional Safety Lifecycle Applies to TI Functional Safety Products

TI has tailored the functional safety lifecycles of ISO 26262 and IEC 61508 to best match the needs of a functional Safety Element out of Context (SEooC) development. The functional safety standards are written in the context of the functional safety systems, which means that some requirements only apply at the system level. Since TI functional safety components are hardware or software components, TI has tailored the functional safety activities to create new product development processes for hardware and for software that makes sure state-of-the-art techniques and measures are applied as appropriate. These new product development processes have been certified by third-party functional safety experts. To find these certifications, go to TI.com/functionalsafety.

B.2 Activities Performed by Texas Instruments

The TI functional safety products are hardware components developed as functional Safety Elements out of Context. As such, TI's functional safety activities focus on those related to management of functional safety around hardware component development. System level architecture, design, and functional safety analysis are not within the scope of TI activities and are the responsibility of the customer. Some techniques for integrating the SEooC safety analysis of this hardware component into the system level can be found in ISO 26262-11.

Table B-1. Activities Performed by Texas Instruments versus Performed by the customer

Functional Safety Lifecycle Activity[1]	TI Execution	Customer Execution
Management of functional safety	Yes	Yes
Definition of end equipment and item	No	Yes
Hazard analysis and risk assessment (of end equipment/ item)	No	Yes
Creation of end equipment functional safety concept	No. Assumptions made for internal development.	Yes
Allocation of end equipment requirements to sub-systems, hardware components, and software components	No. Assumptions made for internal development.	Yes
Definition of hardware component safety requirements	Yes	No
Hardware component architecture and design execution	Yes	No
Hardware component functional safety analysis	Yes	No
Hardware component verification and validation (V&V)	V&V executed to support internal development.	Yes
Integration of hardware component into end equipment	No	Yes
Verification of IC performance in end equipment	No	Yes

Table B-1. Activities Performed by Texas Instruments versus Performed by the customer (continued)

Functional Safety Lifecycle Activity[1]	TI Execution	Customer Execution
Selection of safety mechanisms to be applied to IC	No	Yes
End equipment level verification and validation	No	Yes
End equipment level functional safety analysis	No	Yes
End equipment level functional safety assessment	No	Yes
End equipment release to production	No	Yes
Management of functional safety issues in production	Support provided as needed	Yes

(1) For component technical questions, ask our TI E2E™ support experts.

B.3 Information Provided

Texas instruments has summarized what it considers the most critical functional safety work products that are available to the customer either publicly or under a nondisclosure agreement (NDA). NDAs are required to protect proprietary and sensitive information disclosed in certain functional safety documents.

Table B-2. Product Functional Safety Documentation

Deliverable Name	Contents
Functional Safety Product Preview	Overview of functional safety considerations in product development and product architecture. Delivered ahead of public product announcement.
Functional Safety Manual	User guide for the functional safety features of the product, including system level assumptions of use.
Functional Safety Analysis Report	Results of all available functional safety analysis documented in a format that allows computation of custom metrics.
Functional Safety Report[1]	Summary of arguments and evidence of compliance to functional safety standards. References a specific component, component family, or TI process that was analyzed.
Assessment Certificate[1]	Evidence of compliance to functional safety standards. References a specific component, component family, or TI process that was analyzed. Provided by a 3rd party functional safety assessor.

(1) When an Assessment Certificate is available for a TI functional safety product, the Functional Safety Report may not be provided. When a Functional Safety Report is provided, an Assessment Certificate may not be available. These two documents fulfill the same functional safety requirements and will be used interchangeably depending on the TI functional safety product.

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