User's Guide

MSPM33 NONMAIN Flash Memory Configuration Guide

Utkarsh Kripashankar and Yuhao Zhao

Abstract

The MSPM33 NONMAIN is a dedicated flash memory region that configures parameters related to chip start-up and the selection of extended functions. This document details the configuration of the NONMAIN region, and provides configuration guidance for use in applications.

Table of Contents

Trademarks

1 Introduction

Figure 1-1 shows the common platform memory map that all MSPM33 devices share. The map has four different memory regions.

- MAIN flash
- NONMAIN flash
- FACTORY region
- ROM

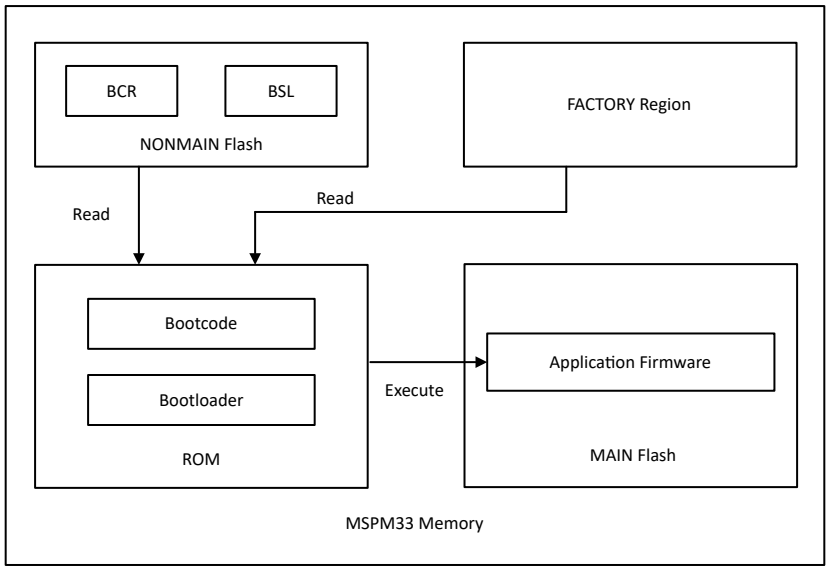


Figure 1-1. MSPM33 Memory Map

MAIN flash is the memory used to store executable code and data. The MSPM33 device supports single bank or dual bank MAIN flash. See the device-specific datasheet for more details.

NONMAIN flash is a dedicated region of flash memory which stores the configuration data used by the boot configuration routine (BCR) configuration and bootstrap loader (BSL) configuration to boot the device. The BCR and BSL have configuration policies which can remain at the default values (as is typical during development and evaluation), or be modified for specific purposes (as is typical during production programming), by altering the values programmed into the NONMAIN flash region.

Note

After erasing the NONMAIN flash, verify that the proper value loads into the NONMAIN flash. Failure to verify the load results in permanently locking the device after the next BOOTRST.

The FACTORY region is a memory-mapped flash region which provides read-only data describing the capabilities of a device, as well as any factory-provided trim information for use by application software.

The read-only memory (ROM) consists of an immutable root-of-trust boot configuration routine (BCR) and bootstrap loader (BSL). The BCR is always the first code to run on the Arm® Cortex®-M33 processor following a BOOTRST of the device before the main application starts. The BCR also runs using hardware or software invocation of the bootstrap loader (BSL) and authorizes the BSL entry. Figure 1-2 shows the high level boot flow for MSPM33 devices. For more details, refer to the *MSPM33C Security User's Guide*.

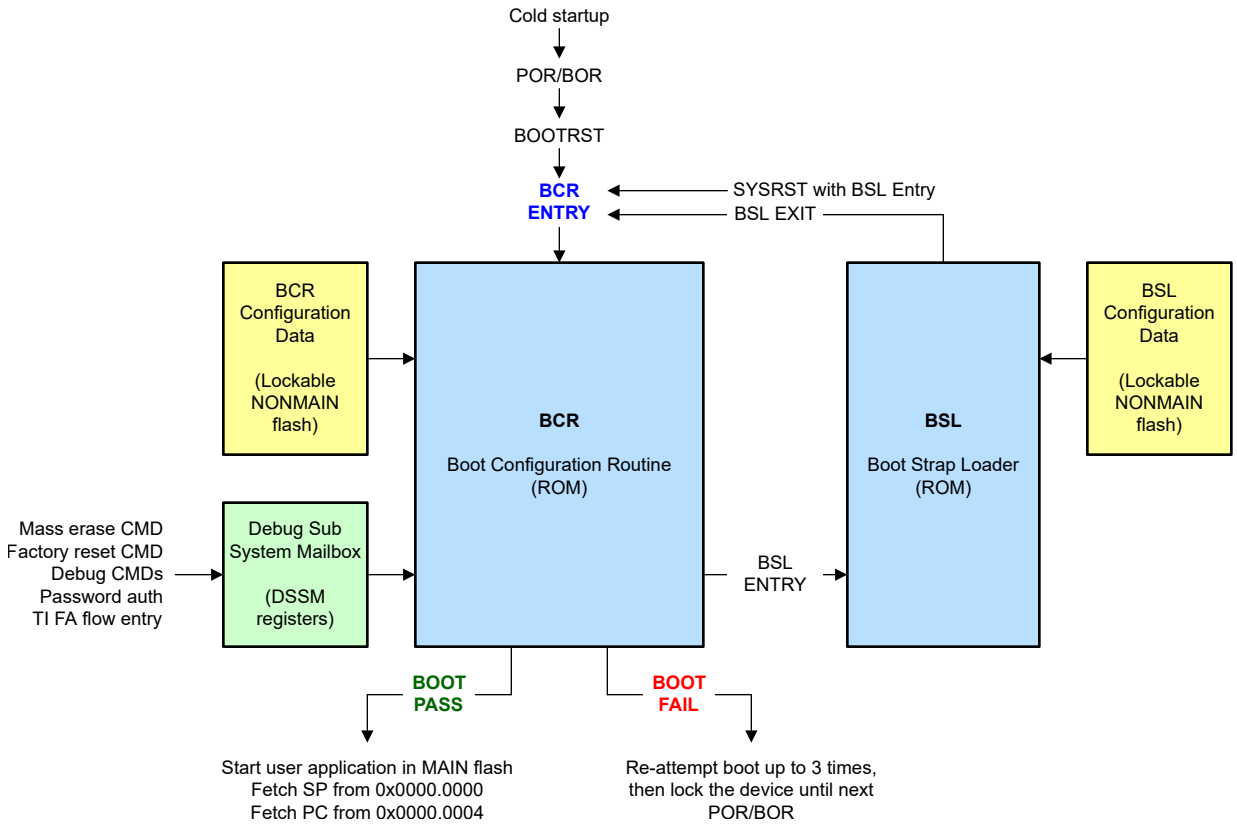


Figure 1-2. MSPM33 Boot Flow

The BCR sets up the device security policies, configures the device for operation, and optionally starts the BSL when BSL is triggered. If the BCR starts the BSL, use the BSL to program or verify the device memory (flash and SRAM) using a standard serial interface (UART, I2C or CAN).

1.1 Terminology

Bootcode (BCR)	Start-up routine that runs after BOOTRST, configuring the device to execute an application.
NONMAIN flash	NONMAIN flash is a dedicated region of flash memory which stores the configuration data used by the boot configuration routine (BCR) configuration and bootstrap loader (BSL) configuration to boot the device.
FACTORY flash	FACTORY region is a memory-mapped flash region which provides read-only data describing the capabilities of a device, as well as any factory-provided trim information for use by application software.
BCR configuration	Configuration structure that contains all user-configurable parameters for Bootcode, which resides in NONMAIN flash memory.
Bootstrap loader (BSL)	Boot routine used to load data to the device memory.
BSL configuration	Configuration structure that contains all user configurable parameters for Bootstrap loader, which resides in NONMAIN flash memory.
Customer Secure Code (CSC)	The code runs after TI boot code and executes from flash memory to further configure security elements. For more details, see the Secure Booting User's Guide

2 NONMAIN Architecture

This section provides a detailed introduction to the NONMAIN structure, supported features, and applications of various NONMAIN configurations.

2.1 MSPM33 Family Overview

The MSPM33 family supports different types of NONMAIN layout that limit the available features supported by the NONMAIN configuration.

Table 2-1 shows which NONMAIN layout type to use with which devices. For more details, see the *Boot Configuration* section of the device-specific technical reference manual, linked in Section 10.

Table 2-1. NONMAIN Layout Types

NONMAIN Layout Type	Supported Devices	Overview
Type A	MSPM33C32xx	• Includes CSC support • Passwords storage in a SHA256 hashed format • Application integrity check uses CRC32 or SHA256 hash options • System PLL configuration • Flash security configuration • UART default baud rate configuration • Supports ROM BSL only • Support OTP region for device configuration

2.2 NONMAIN Configuration Overview

MSPM33 NONMAIN flash memory has different types of structure layouts. In certain layout types, some features are reduced or not supported.

Table 2-2. NONMAIN Configuration Parameters

Configuration Parameters			Register Field	NONMAIN Layout Type
				A
BCR	BCR Configuration ID		BCRCONFIGID	✓
	Serial Wire Debug (SWD) Policy Configuration	Access Policy	BOOTCFG0	✓
		Debug Policy		
		Debug Password	DEBUG_LOCK	✓
			DEBUG_NS_LOCK	✓
	SWD Factory Reset Configuration	Factory Reset Access Property	BOOTCFG4	✓
		Factory Reset Password	MASS_ERASE_y	✓
	SWD Mass Erase Configuration	Mass Erase Access Property	BOOTCFG4	✓
		Mass Erase Password	FACTORY_RESET_y	✓
	Flash Security Configuration	Bank0 Main FLASH Write Erase Protection	BANK0_WRITE_ERASE_PROTECTION_A BANK0_WRITE_ERASE_PROTECTION_B	✓
		Bank0 Main FLASH Security Protection	BANK0_SECURITY_PROTECTION_A BANK0_SECURITY_PROTECTION_B	✓
		Bank0 Main FLASH Privilege Protection	BANK0_PRIVILEGE_PROTECTION_A BANK0_PRIVILEGE_PROTECTION_B	✓
		Bank1 Main FLASH Write Erase Protection	BANK1_WRITE_ERASE_PROTECTION_A BANK1_WRITE_ERASE_PROTECTION_B	✓
		Bank1 Main FLASH Security Protection	BANK1_SECURITY_PROTECTION_A BANK1_SECURITY_PROTECTION_B	✓
		Bank1 Main FLASH Privilege Protection	BANK1_PRIVILEGE_PROTECTION_A BANK1_PRIVILEGE_PROTECTION_B	✓
		Data FLASH Write Erase Protection	DBANK_WRITE_ERASE_PROTECTION	✓
		Data FLASH Security Protection	DBANK_SECURITY_PROTECTION	✓
		Data FLASH Privilege Protection	DBANK_PRIVILEGE_PROTECTION	✓
		NONMAIN Static Write Protection	BANK0_NM_USER_CONFIG	✓
	SYSPLL Configuration	System PLL Configuration During the Boot	BOOTCLK0 BOOTCLK1	✓
	Bad Device Conversion	Bad Device Conversion Password	BAD_CONV_y	✓
	Customer Secure Code (CSC)	CSC Policy	BOOTCFG2	✓
		Flash Bank Swap Policy		✓
		Debug Hold	BOOTCFG1	✓

Table 2-2. NONMAIN Configuration Parameters (continued)

Configuration Parameters			Register Field	NONMAIN Layout Type
				A
BCR	Fast Boot Mode		BOOTCFG3	✓
	Application Digest Check Configuration	Application Digest Check Policy	SECURE_BOOT_MODE	✓
		Application Start Address	USER_SECURE_APP_START_ADDR	
		Length of the Application	USER_SECURE_APP_LENGTH	
		Application Checksum	USER_SECURE_APP_HASH_y	
	BSL Policy	Enable Invoke Pin Check	BOOTCFG1	✓
		Enable BSL Mode	BOOTCFG3	
	BCR Checksum	CRC32-ISO3309	BCRCRC	✓
BSL	BSL Configuration ID		BSLCONFIGID	✓
	BSL Pin Configuration	Invoke Pin Configuration	BSLCONFIG0	✓
	UART Interface Configuration	Pin Configuration	BSLPINCFG0	✓
		Baud Rate Configuration	BSLCONFIG1	✓
	I2C Interface Configuration	Pin Configuration	BSLPINCFG1	✓
		Target Address Configuration	I2C_SLAVE_ADDR	✓
	CAN Interface Configuration	Pin Configuration	BSLPINCFG2	✓
	BSL Security Configuration	BSL Access Password	PASSWORD_y	✓
		BSL Read Out Feature	BSLCONFIG0	
		BSL Alert Configuration	BSLCONFIG1	
		App Integrity Check	APP_REV_POINTER	
	BSL Checksum	CRC32-ISO3309	BSLCRC	✓
Device Config	Customer Key Configuration	Key Count	KEY_COUNT	✓
		Key Revision	KEY_REV	
		Active Key for Secure Boot	MPUK_ACTIVE	
		Active Key for DSSM	DSSM_ACTIVE	
	Lifecycle Maintaining	Customer Code Provisioned	CODE_PROVISION	✓
		Failure Analysis Enabled	FAILURE_ANALYSIS	
		Decommission Enabled	DCOM	
	Hardware-protected Flash Security Configuration for HSSE Device	Bank0 Main FLASH Write Erase Protection	BANK0_WRITE_ERASE_PROTECTION_A BANK0_WRITE_ERASE_PROTECTION_B	✓
		Bank0 Main FLASH Security Protection	BANK0_SECURITY_PROTECTION_A BANK0_SECURITY_PROTECTION_B	✓
		Bank0 Main FLASH Privilege Protection	BANK0_PRIVILEGE_PROTECTION_A BANK0_PRIVILEGE_PROTECTION_B	✓

Table 2-2. NONMAIN Configuration Parameters (continued)

Configuration Parameters			Register Field	NONMAIN Layout Type
				A
Device Config	Hardware-protected Flash Security Configuration for HSSE Device	Bank1 Main FLASH Write Erase Protection	BANK1_WRITE_ERASE_PROTECTION_A BANK1_WRITE_ERASE_PROTECTION_B	✓
		Bank1 Main FLASH Security Protection	BANK1_SECURITY_PROTECTION_A BANK1_SECURITY_PROTECTION_B	✓
		Bank1 Main FLASH Privilege Protection	BANK1_PRIVILEGE_PROTECTION_A BANK1_PRIVILEGE_PROTECTION_B	✓
		Data FLASH Write Erase Protection	DBANK_WRITE_ERASE_PROTECTION	✓
		Data FLASH Security Protection	DBANK_SECURITY_PROTECTION	✓
		Data FLASH Privilege Protection	DBANK_PRIVILEGE_PROTECTION	✓

Note

For different types of NONMAIN layout, the register address offset can be different. See the device-specific technical reference manual for details, linked in Section 10.

Note

Device configuration (Device Config) is an OTP region used by the MSPM33 for maintaining lifecycle. HSSE is one of the lifecycle statuses for the MSPM33 (other statues are HSKP, HSFS, and HSFA). For more information, please refer to *MSPM33C Security User Guide* or *MSPM33 C3-Series 160MHz Microcontrollers Technical Reference Manual*

2.3 NONMAIN Memory

Table 2-3 shows the overall memory sections for NONMAIN flash. The register memory map is different per the NONMAIN layout type. Refer to the device-specific technical reference manual for further details, linked in Section 10.

Table 2-3. NONMAIN Memory Map

NONMAIN Section	Start Address	End Address
Device Config (OTP)	8010,1000h	8010,17FFh
BCR Configuration	8010.1800h	8010.1BFFh
BSL Configuration	8010.1C00h	8010.1FFFh

The MSPM33 devices provision an OTP region (2KB) in NONMAIN memory that is accessed with the following restrictions:

- The region can never be erased (factory reset, mass erase operations included).
- The region can be written in all life cycle statuses, including HSFS, HSKP, HSSE, and HSFA.

This OTP region contains the device configuration used by ROM boot, which is related to MSPM33 key provisioning, maintaining life cycle, and the hardware-protected secure and privilege attribute configuration for HSSE devices.

The user can also implement a hardware-based monotone counter feature on the unused OTP region. Since the sector can only be programmed, the feature effectively works as a nonvolatile up-counter (or down-counter depending on how the counter value is interpreted by software).

Refer to the device-specific datasheet to determine if the device supports an OTP region in NONMAIN memory.

Note

The OTP region cannot be erased and re-programmed. Incorrect programming to the defined OTP region (registers defined by TI) can cause an unrecoverable incorrect configuration on the customer keys, the life cycle maintenance, and the secure and privilege attributes on FLASH, which can lead to the MCU locking.

3 NONMAIN Configuration

This section provides a detailed introduction to the NONMAIN configuration, including the BCR configuration and BSL configuration.

3.1 BCR Configuration

The boot configuration routine (BCR) is the first firmware to run on the device after a BOOTRST. Users can set different properties for the device at the start-up stage of boot.

TI recommends using the SysConfig tool to customize the BCR configuration, see BCR Configuration With SysConfig for more details.

3.1.1 BCR Configuration ID

TI defines a default BCR configuration ID for different types of NONMAIN layouts. Users can overwrite the configuration ID for application usage. Perform a factory reset to recover the default BCR configuration ID.

Table 3-1 shows the default values of each device.

Table 3-1. Default BCR Configuration ID

NONMAIN Layout Type	Device Family	Default Value
Type A	MSPM33C321A	0x05000000

3.1.2 Serial Wire Debug (SWD) Policy

The serial wire debug (SWD) related policies configure the available functions in the physical debug interface of the device. MSPM33 devices support three generic security scenarios: no restrictions, custom restrictions, and fully restricted. Table 3-2 shows the three generic security scenarios, from least restrictive to most restrictive, including the differences between each level.

Table 3-2. Generic Security Levels

Scenario	SW-DP Access Policy	APP Debug Policy	Mass Erase Policy	Factory Reset Policy
No restrictions	EN	L0 – Full access	EN	EN
Custom restrictions	EN	L0 – Full access L1 – EN SECURE with PW, EN NON SECURE L2 – EN SECURE with PW, EN NON SECURE with PW DIS – Disabled	EN, EN with PW, DIS	EN, EN with PW,DIS
Fully restricted	DIS	Do not care(1) (access in not possible when SW-DP is disabled)		

(1) When the SW-DP policy is *SW-DP disabled*, the mass erase and factory reset policies are a *do not care* from the point of view of the SWD interface. However, if the bootstrap loader (BSL) is enabled, the mass erase and factory reset policies do impact the available BSL functions. See the BSL security section (Section 3.2.4) for details on securing the BSL.

By default, TI provides MSPM33 in an unrestricted state. The unrestricted state allows for easy production programming, evaluation, and development. However, this unrestricted state is not recommended for mass production, as the unrestricted states leave a large attack surface present.

There are three main uses of the SWD interface to take into consideration when determining protection needs:

- Application debug access: Debugging in the IDE tool
- Mass erase access: Erase the MAIN memory region
- Factory reset access: Erase the MAIN memory region and reset the NONMAIN device configuration memory to TI factory defaults (no restrictions)

The SWD security policies are implemented as 16-bit pattern-match fields in the NONMAIN memory, with the following characteristics:

- An exact pattern match is required to enable lower security states

- Any value in the 16-bit field mismatching the exact defined patterns results in a maximally secure state for the respective parameter

SWD also supports customizing eight sets of passwords (128 bits in total), used to unlock the device using a debug subsystem mailbox (DSSM) command. See the *Hardware Programming and Debugger Guide for MSPM0* application note for detailed instructions.

3.1.2.1 Access Policy

BCR supports modifying the SWD access policy through the BOOTCFG0.SWDP_MODE field, which configures the device in a maximally restrictive state.

When this field is disabled (SW-DP disabled), the physical debug port (SW-DP) is completely disabled, and all of the SWD-accessible functions (application debug, mass erase, and factory reset) are inaccessible through the SWD, regardless of individual configurations.

If the BSL is enabled, then the mass erase and factory reset configuration fields are still used by the BSL to authorize mass erase or factory reset commands originating from the BSL interface. See Section 3.2 for details.

3.1.2.2 Debug Policy

When the SWD access policy remains enabled, the user can further modify the SWD debug policy through the BOOTCFG0.DEBUGACCESS field.

The MSPM33 supports four levels of SWD access policy: full access, all disabled, encryption on secure debug with a *128-bit* password, and encryption on secure debug and non-secure debug with two *128-bit* passwords. The MSPM33 can apply different passwords on secure debug and non-secure debug. The device only verifies the password during the boot stage. The user generates a DSSM command for password authentication, which includes a reset behavior to unlock the SWD interface.

The MSPM33 provides password storage with SHA2-256 digest. The SHA2-256 digest storage pattern does not directly store the *128-bit* password in the NONMAIN memory region. Instead, the user generates a *256-bit* hash value from the *128-bit* password and stores the hash value in the NONMAIN region, which enhances the security of the password storage. The hash values are stored in the DEBUG_LOCK field and DEBUG_NS_LOCK field.

Below is how the user generates and sets a SHA2-256 digest password:

1. Determine the *128-bit* password that unlocks the SWD interface. Set the password into 32-bit alignment to form four *32-bit* passwords.
2. Reverse the four *32-bit* password endianness as SHA uses an *8-bit* addressing format.
3. When calculating the hash value, combine the four reversed *32-bit* password as one string.
4. Calculate the SHA256 value of the input string.
5. Break the output SHA256 value into eight *32-bit* words.
6. Reverse the endianness of the output of the eight *32-bit* words as the SHA2-256 calculation uses an *8-bit* addressing format.
7. Store the reversed passwords in the DEBUG_LOCK field or DEBUG_NS_LOCK field in sequence.

Note

The *0* value in the password is important. Do not delete the *0* value during calculation.

The flow is designed for all devices that support *256-bit* SHA2-256 digest password generation, including:

- SWD access
- Factory reset
- Mass erase
- Bad device conversion
- BSL access (*256-bit* password)

Section 4.2 shows how to set the password using the SysConfig tool. Section 6.3 shows how to unlock the device using the CCS tool.

3.1.2.2.1 SHA2-256 Password Example

Below is a calculation example for the SHA2-256 digest password, if the user sets the *128-bit* password as *0123456789ABCDEF67452301EFCDAB89*.

1. Set the password into 32-bit alignment: 0x01234567, 0x89ABCDEF, 0x67452301, 0xEFCDAB89
2. Reverse the password endianness as SHA2-256 is calculated in byte: 0x67452301, 0xEFCDAB89, 0x01234567, 0x89ABCDEF
3. Combine the four reversed *32-bit* password as one string: 67452301EFCDAB890123456789ABCDEF
4. Calculate the SHA2-256 value (select *HEX* as the input encoding): 8420347FCB0F019E15564A0F65B8E197ECDF9F92D1ECA2BBAB1B8CB314C763DA (SHA256 online tool)
5. Break up the SHA2-256 value into eight *32-bit* words: 0x8420347F, 0xCB0F019E, 0x15564A0F, 0x65B8E197, 0xECDF9F92, 0xD1ECA2BB, 0xAB1B8CB3, 0x14C763DA
6. Reverse the output endianness: 0x7F342084, 0x9E010FCB, 0x0F4A5615, 0x97E1B865, 0x929FDFEC, 0xBBA2ECD1, 0xB38C1BAB, 0xDA63C714
7. Store the eight *32-bit* passwords in the DEBUG_LOCK field or DEBUG_NS_LOCK field in order

3.1.2.3 Mass Erase and Factory Reset Policy

A *SWD Mass Erase* is an erase of the MAIN flash regions only, which typically includes the user application. The BCR and BSL policies stored in the NONMAIN flash region are not affected by a mass erase. A mass erase is useful for erasing all application code and data while leaving the device NONMAIN configuration intact.

A *SWD Factory Reset* is an erase of the MAIN flash regions followed by a reset of the NONMAIN flash region to default values. Such an erase is useful for completely resetting the BCR and BSL device boot policies while also erasing the application code and data.

The BCR configuration provides mass erase and factory reset functions through commands sent to the device over the SWD interface from a debug probe using the DSSM. These commands are not available when SWD is fully restricted (SW-DP disabled). When the device is not configured for fully restricted SW-DP disabled), there are three configure positions possible for the mass erase and factory reset commands: enabled, enabled with a unique 128-bit password, or disabled.

Note

When *Factory Reset* or *Mass Erase* is individually configured as disable, then the BSL *cannot* perform the factory reset command or mass erase command.

See the device-specific datasheet to determine the supported *128-bit* password storage pattern (SHA2-256). Refer to Section 3.1.2.2 for instructions on calculating the SHA2-256 digest password, and Section 8.2 for the flow to generate a *SWD Mass Erase* or *Factory Reset*.

3.1.3 Flash Security Configuration

The flash security configuration policies specify which sectors of flash memory are locked from modification, which sectors are set to security protection, and which sectors are set to privilege protection. There are three types of protection: static write erase protection, security protection, and privilege protection.

Static write erase protection can be applied to the MAIN flash memory, DATA flash memory, and NONMAIN flash memory. Static write protection enables the placement of a fixed, user-defined application, data, or NONMAIN configuration in the flash memory that has the following characteristics:

1. Once programmed and locked, the locked sectors are not modifiable by the application code or ROM bootloader.
2. If placed at the beginning of the flash memory, the application is the first code that executes when the ROM boot configuration routine transfers execution to the user application.

Note

When the DSSM command generates a mass erase or factory reset, the command overrides the specified static write protection policy. In contrast, the BSL command does *not* generate an override, and the corresponding protected flash area remains unchanged.

Security protection and privilege protection can only be applied to MAIN flash memory and DATA flash memory. Security protection defines the range of the secure region. Any code and data in the secure region is protected from the non-secure region. Privilege protection defines the range of the privilege region. Any code and data in the privilege region is protected from the non-privilege region.

Note

For the secure and non-secure partition on MSPM33, final secure attribute depends on not only GSC configuration but also SAU and IDAU setting. Please refer to *Enable Trustzone for Secure Application in MSPM33 MCUs*for detailed configuration.

3.1.3.1 MAIN Flash Security Configuration

For static write protection on bank 0, the BANK0_WRITE_ERASE_PROTECTION_ A and BANK0_WRITE_ERASE_PROTECTION_ B fields protect the whole 512KB memory. A value of *0* indicates the write protection applies to the corresponding sectors. For BANK0_WRITE_ERASE_PROTECTION_ A, the 1 bit represents one sector in flash memory (2KB). For BANK0_WRITE_ERASE_PROTECTION_ B, the 1 bit represents eight sectors in flash memory (16KB).

For static write protection on bank 1, the BANK1_WRITE_ERASE_PROTECTION_ A and BANK1_WRITE_ERASE_PROTECTION_ B fields protect the whole 512KB memory. A value of *0* indicates the write protection applies to the corresponding sectors. For BANK1_WRITE_ERASE_PROTECTION_ A, the 1 bit represents one sector in flash memory (2KB). For BANK1_WRITE_ERASE_PROTECTION_ B, the 1 bit represents eight sectors in flash memory (16KB).

For security protection on bank 0, the BANK0_SECURITY_PROTECTION_A and BANK0_SECURITY_PROTECTION_B fields define the secure regions in the whole 512KB memory. A value of *0* indicates the corresponding sectors are secure. For BANK0_SECURITY_PROTECTION_A, the 1 bit represents one sector in flash memory (2KB). For BANK0_SECURITY_PROTECTION_B, the 1 bit represents eight sectors in flash memory (16KB).

For security protection on bank 1, the BANK1_SECURITY_PROTECTION_A and BANK1_SECURITY_PROTECTION_B fields define the secure regions in the whole 512KB memory. A value of *0* indicates the corresponding sectors are secure. For BANK1_SECURITY_PROTECTION_A, the 1 bit represents one sector in flash memory (2KB). For BANK1_SECURITY_PROTECTION_B, the 1 bit represents eight sectors in flash memory (16KB).

For privilege protection on bank 0, the BANK0_PRIVILEGE_PROTECTION_A and BANK0_PRIVILEGE_PROTECTION_B fields define the privileged regions in the whole 512KB memory. A value of *0* indicates the corresponding sectors are privileged. For BANK0_PRIVILEGE_PROTECTION_A, the 1 bit represents one sector in flash memory (2KB). For BANK0_PRIVILEGE_PROTECTION_B, the 1 bit represents eight sectors in flash memory (16KB).

For privilege protection on bank 1, the BANK1_PRIVILEGE_PROTECTION_A field and BANK1_PRIVILEGE_PROTECTION_B fields define the privileged regions in the whole 512KB memory. A value of *0* indicates the corresponding sectors are privileged. For BANK1_PRIVILEGE_PROTECTION_A, the 1 bit represents one sector in flash memory (2KB). For BANK1_PRIVILEGE_PROTECTION_B, the 1 bit represents eight sectors in flash memory (16KB).

3.1.3.2 DATA Flash Security Configuration

For static write protection on DATA FLASH, the DBANK_WRITE_ERASE_PROTECTION field protects the whole data bank memory. A value of *0* indicates the write protection applies to the corresponding sectors. For DBANK_WRITE_ERASE_PROTECTION, the 1 bit represents one sector in flash memory (2KB).

For security protection on DATA FLASH, the DBANK_SECURITY_PROTECTION field defines the secure regions in the whole data bank memory. A value of *0* indicates the corresponding sectors are secure. For DBANK_SECURITY_PROTECTION, the 1 bit represents one sector in flash memory (2KB).

For privilege protection on DATA FLASH, the DBANK_PRIVILEGE_PROTECTION field defines the privileged regions in the whole data bank memory. A value of *0* indicates the corresponding sectors are privileged. For DBANK_PRIVILEGE_PROTECTION, the 1 bit represents one sector in flash memory (2KB).

3.1.3.3 NONMAIN Flash Static Write Protection

When users configure the BANK0_NM_USER_CONFIG to apply static write erase portection on NONMAIN region, the entire NONMAIN region is write-locked. This lock makes the region functionally immutable when the boot configuration routine transfers execution to either the bootstrap loader or the user application code in the MAIN flash. Any attempt to program or erase the NONMAIN region by the application code or the bootstrap loader results in a hardware flash operation error, and the sector remains unmodified.

3.1.4 SYSPLL Configuration During Boot

With the default configuration, ROM boot is executed with a 32MHz MCLK, sourced by the SYSOSC, and no SYSPLL. To speed up the boot process, the user can configure a SYSPLL as well as the clock source. The user can configure the following settings:

- SYSPLL predivider (PDIV) and SYSPLL feedback divider (QDIV)
- Clock source selection – SYSOSC (32MHz) or HFXT (20MHz or 40MHz)

Set the SYSPLL configuration using the BOOTCLK0 and BOOTCLK1 fields in the NONMAIN flash memory.

3.1.5 Customer Secure Code (CSC)

Customer secure code (CSC) is the code that runs after the TI boot code. The CSC executes from flash memory to further configure security elements. The CSC is customer-owned secure software, and customers must verify that CSC is located at the 0x0 address. Figure 3-1 shows the simplified flow of CSC execution.

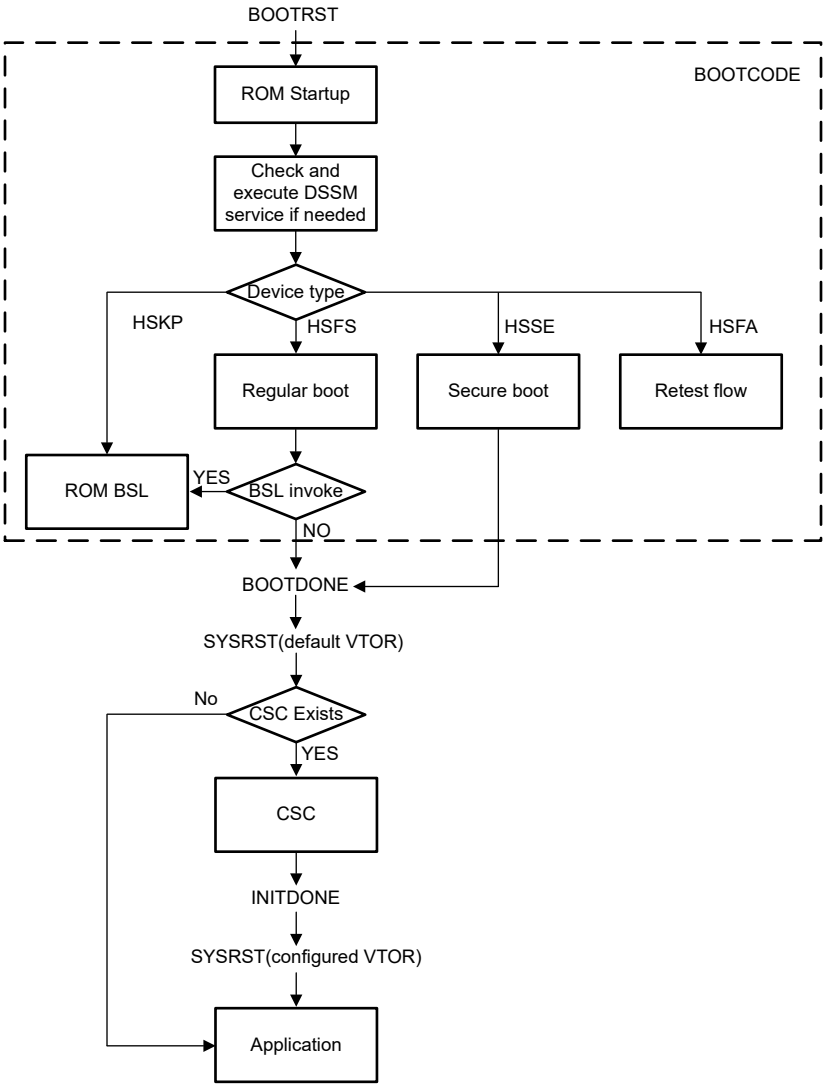


Figure 3-1. Secure Boot and Start Sequence

The TI security model concept trusts both TI boot code and CSC. To trust the CSC, TI recommends enabling *Debug Hold* and *Flash Memory Static Write Protection*. For the implementation on CSC, please refer to *Evaluate Customer Secure Code on MSPM33 MCU*

Note

In the flow, the device type HSKP, HSFS, HSSE and HSFA are the life cycle statuses for the MSPM33 device. For more information, please refer to *MSPM33C Security User Guide* or *MSPM33 C3-Series 160MHz Microcontrollers Technical Reference Manual*

3.1.5.1 CSC Policy

By default, CSC is disabled. The user enables CSC using the BOOTCFG2.CSCEXISTS field. When the CSC completes execution, the system undergoes a second SYSRST with configured VTOR, followed by the main application starting.

3.1.5.2 Flash Bank Swap Policy

In multiple bank devices, the bank swap is disabled by default.

Users can enable the bank swap using the BOOTCFG2.FLASHBANKSWAPPOLICY field. Boot code reads this configuration and writes to the SYSCTL.SECCFG.FLBANKSWPPOLICY with the appropriate KEY (0xCA). With the hardware default, the swappable configuration is ENABLED, and the lower bank is used as a logical bank 0. The default value of the BOOTCFG2.FLASHBANKSWAPPOLICY field disables the bank swap policy.

The bank swap behavior happens in the CSC stage, while the behavior goes into effect after the device issues INITDONE. CSC writes to the SYSCTL.SECCFG.FLBANKSWP with the appropriate KEY (0x58) to determine which bank as logical bank 0.

1. If the lower bank is as logical bank 0, the address for the lower bank and upper bank does not change.
2. If the upper bank is as logical bank 0, the address for the lower bank and upper bank is exchanged.

This mechanism enforces the policy where after INITDOWN is issued, and in general the execution bank is the logical bank 0 (lower address), and save location for firmware updates is the logical bank 1 (upper address).

Note

When the bank swap happens to make the upper bank as logical bank 0, only the address is swapped. Data in the lower bank and upper bank is not transferred.

Note

No matter if bank swap enables or not, multiple banks have the same property (write, read, and execute) with different memory addresses.

For more details on bank swap features, see the *Evaluate Customer Secure Code on MSPM33 MCU*.

3.1.5.3 Debug Hold

For devices that support CSC features, the BCR configuration provides access ability to lock of the SWD interface during CSC execution. The device does not release debug access until the INITDOWN is issued at the end of CSC.

The debug hold feature is disabled by default, and TI recommends users keep this feature disabled during application debugging. When the product enters the mass production stage with CSC enabled, TI recommends enabling debug hold using the BOOTCFG1.DEBUGHOLD field to protect CSC in an inaccessible state.

3.1.6 Fast Boot Mode

Reduce the execution time of the BCR by enabling fast boot mode. Fast boot mode speeds up the boot process using the following methods:

- Limiting the BSL entry: By enabling fast boot mode, enter the bootloader using only the SYSCTL register invoke method and the DSSM invoke method.
- Bypassing the application digest check, even if the application digest check is enabled. See Section 3.1.7 for more information.

Set fast boot mode to be enabled using the BOOTCFG3.FASTBOOTMODE field in the NONMAIN flash memory.

3.1.7 Application Digest Check

The BCR supports the execution of a complete CRC32 or SHA2-256 integrity check of the application code and data contained in the MAIN flash regions during the boot process, before starting the user application. The application digest check is useful for verifying the integrity of some or all of the application code and data before execution.

Follow these steps to enable the application digest check:

1. Select the application digest check policy, which can be set as CRC32 or SHA2-256 (SECURE_BOOT_MODE field).
2. Set the 32-bit starting address of the application digest check (USER_SECURE_APP_START_ADDR field).
3. Set the length of the application (specified in bytes) for which the CRC or SHA digest check applies (USER_SECURE_APP_LENGTH field).
4. Calculate the CRC32 or SHA2-256 value with the given data (specified in bytes).

5. Break the pre-calculated CRC32 or SHA2-256 value into 32-bit format and store the value in the BCR configuration (USER_SECURE_APP_HASH_y field).

In the event that an application digest check fails at boot, the application in the MAIN flash does not start. If the BSL is enabled, then the BSL is entered. If the BSL is not enabled, then the boot fails.

3.1.7.1 CRC32 Digest Check Example

Figure 3-2 is a calculation example for the CRC32 integrity check, if the user sets the start address as 0x0001.0000 and the length as eight bytes. The application data refers to Section 3.1.7.

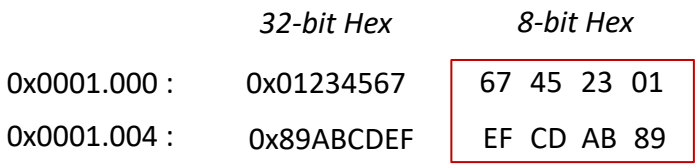


Figure 3-2. Example Data for CRC32 Digest Check

Select the *JAMCRC* as the calculation model to get the CRC32 output value. The *JAMCRC* CRC model has the reversed input and output bit, and uses the polynomial of 0x04C11DB7 with an initial value of 0xFFFFFFFF, and an output XOR value of 0x00000000.

Follow these steps to calculate the example CRC32 value:

1. Select the application digest check policy as check with CRC (SECURE_BOOT_MODE = Abhor).
2. Set the 32-bit starting address of the application digest check (USER_SECURE_APP_ST ART_ADDR = 0001.0000h).
3. Set the length of the application (specified in bytes) for which the CRC digest check applies (USER_SECURE_APP_LENGTH = 8h).
4. Calculate the CRC32 output with the given input:
   a. Reverse the input data endianness as CRC32 is calculated using an 8-bit format: 67, 45, 23, 01, EF, CD, AB, 89.
   b. Combine the input 8-bit data as one string: 67452301EFCDAB89.
   c. Calculate the CRC32 value (select *HEX* as the input encoding, *JAMCRC* as the model): 0x24648719 (CRC32 online tool).
5. Register the CRC32 output value in the BCR configuration (USER_SECURE_APP_HASH_1 = 2464.8719h). If there are multiple USER_SECURE_APP_HASH_y registers, only the first 32-bit word of the DIGEST is used for CRC32.

Note

Verify that the length of the application CRC32 digest check is an even number.

3.1.7.2 SHA2-256 Digest Check Example

Figure 3-3 is a calculation example for the SHA2-256 integrity check, if the user sets the start address as 0x0001.0000 and the length as eight bytes. Figure 3-3 shows the example application data.

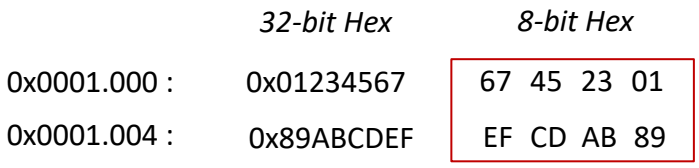


Figure 3-3. Example Data for SHA2-256 Digest Check

Follow these steps to calculate the example SHA2-256 value:

1. Select the application digest check policy as SHA2-256 (SECURE_BOOT_MODE = CCDDh).
2. Set the 32-bit starting address of the application digest check (USER_SECURE_APP_ST ART_ADDR = 0001.0000h).
3. Set the length of the application (specified in bytes) for which the SHA2-256 digest check applies (USER_SECURE_APP_LENGTH = 8h).
4. Calculate the SHA2-256 output with the given input:
   a. Reverse the input 32-bit data endianness as SHA2-256 is calculated in byte: 67, 45, 23, 01, EF, CD, AB, 89.
   b. Combine the input 8-bit data as one string: 67452301EFCDAB89.
   c. Calculate the SHA2-256 value (select *HEX* as the input encoding): FDD232547CEEE35ADD35783CA7D518D2A49ABCCD0B60B028C8A9C629EBA6201F (SHA2-256 online tool).
   d. Break up the SHA2-256 value into eight 32-bit words: 0xFDD23254, 0x7CEEE35A, 0xDD35783C, 0xA7D518D2, 0xA49ABCCD, 0x0B60B028, 0xC8A9C629, 0xEBA6201F.
5. Reverse the 32-bit output words endianness: 0x5432D2FD, 0x5AE3EE7C, 0x3C7835DD, 0xD218D5A7, 0xCDBC9AA4, 0x28B0600B, 0x29C6A9C8, 0x1F20A6EB.
6. Store the eight 32-bit SHA2-256 checksums in the USER_SECURE_APP_HASH_y field in order.

3.1.8 BSL Policy

For devices that support ROM BSL, TI provides access ability to the device memory through the BSL interface. One default BSL invoke pin is used for hardware invoking the BSL, see Section 3.2.2 for more details.

The user can disable the BSL through the BOOTCFG3.BOOTLOADER_MODE field, which fully disables BSL mode. The BSL software invoke (through the SYSRST) is also blocked.

Alternatively, disabling the BSL invoke pin check using the BOOTCFG1.BSL_PIN_INVOKE field blocks the BSL hardware entry. After disabling the invoke pin check, the device does *not* check the invoke pin status during the boot, so that the BSL hardware entry is disabled. Users can enter the BSL using the software invoke method in the application code.

3.1.9 BCR Checksum

The MSPM33 device provides a CRC check for BCR configuration data during the boot stage. If the device supports CRC32-ISO3309, the 32-bit CRC digest is applied.

The following lists the configuration for CRC calculation:

- Select a polynomial with the preselected CRC standard (CRC32-ISO3309)
- Reflect the input and output value
- Initial value is set as 0xFFFFFFFF (or 0xFFFF)
- Final XOR value is set as 0x00000000 (or 0x0000)

TI recommends using the SysConfig tool to automatically calculate BCR checksum, see Section 4.2 for more details.

3.1.9.1 CRC Check Fail Handling

If the BCR configuration data fails the CRC check during boot, a catastrophic boot error results and the following limitations are imposed:

- The error cause is logged in the CFG-AP as a boot diagnostic
- The BSL is not invoked, even if the BSL was configured as enabled
- The user application is not started
- No application debug access is enabled
- A pending SWD factory reset command, if enabled or enabled with password, is honored
- A pending TI failure analysis flow entry, if enabled, is honored
- The boot process re-attempts up to three times
  - If the 2nd or 3rd attempt passes, the device boots normally
  - If the 3rd attempt does not pass, no further boot attempts are made until the next BOR or POR

3.2 BSL Configuration

The bootstrap loader (BSL) routine is the ROM based firmware used to load data to the device memory. Users set different properties for the BSL entry and execution through the BSL configuration of the NONMAIN region.

TI recommends using the SysConfig tool to customize the BSL configuration, see Section 4.3 for more details.

3.2.1 BSL Configuration ID

TI defines a default BSL configuration ID for different NONMAIN layouts. Users can overwrite the BSL ID for application usage. Perform a factory reset to recover the default BSL configuration ID.

Default BSL Configuration ID shows the default values of each device:

Table 3-3. Default BSL Configuration ID

NONMAIN Layout Type	Device Family	Default Value
Type A	MSPM33C321A	0x05000000

3.2.2 Invoke Pin Configuration

The bootloader supports hardware invoking after a BOOTRST by using a GPIO. TI devices are configured with a specific GPIO and polarity. See the device-specific datasheet for the default invoke pin assignment.

Users modify the default invoke pin using the BSLCONFIG0 field. The invoke pin configuration contains the:

- Pad
- Port
- Pin
- Polarity

If the initial status of the default invoke pin in the schematic design matches the invoke pin configuration, the device enters BSL after power up and never runs application code.

3.2.3 ROM-Based Communication Interface

The MSPM33 supports a configurable I2C, UART, or CAN as the serial wire BSL interface. The ROM-based BSL allows the user to customize the UART, I2C or CAN pins, but not include the communication interface instance (for example, from UC0 to UC1). Users can customize the UART, I2C or CAN pins, see the device-specific datasheet for the default pin assignment.

For the device with a USB interface, the ROM-based interface further supports the device firmware upgrade (DFU) option.

3.2.3.1 UART Interface

For the device that supports the ROM BSL, the UART interface is available. The BSL configuration supports the modification of the configuration for the UART pins (pad and MUX) through the BSLPINCFG0 field. The attribute property of the device pin restricts the available pins as the UART instance is unchanged. See the device-specific datasheet for the available pins.

The default baud rate of the UART is 9600. Users can dynamically modify the UART communication baud rate in the BSL command (the command is sent with the 9600 baud rate). Furthermore, the layout of the NONMAIN region supports a modified default baud rate through the BSLCONFIG1.UART_BAUD_RATE field.

3.2.3.2 I2C Interface

For the device that supports the ROM BSL, the I2C interface is available. The BSL configuration supports the modification of the configuration for the I2C pins (pad and MUX) through the BSLPINCFG1 field. The attribute property of the device pin restricts the available pins for the specific I2C instance. See the device-specific datasheet for the available pins.

The default target address of the I2C is 0x48. The MSPM33 supports a modify target address through the I2C_SLAVE_ADDR field. The maximum support is a 7-bit target address.

3.2.3.3 CAN Interface

For the device that supports the ROM BSL, the CAN interface is available. The BSL configuration supports the modification of the configuration of the CAN pins (pad and MUX) through the BSLPINCFG2 field. The attribute property of the device pin restricts the available pins for the specific CAN instance. See the device-specific datasheet for the available pins.

The default mode is CAN-FD with 1Mbps (nominal and data). The default sampling point is 80%. Acceptable MCAN protocol IDs are 0x3 or 0x5.

For more details on the BSL UART, I2C and CAN interfaces, refer to the *MSPM33 Bootloader User's Guide*.

3.2.4 BSL Security Configuration

The BSL core provides a branch of security features to prevent potential attack:

- Access password
- Read-out
- Alert
- Application integrity check

The security features always take effect in the ROM BSL.

3.2.4.1 Access Password

A user-specified password protects ROM BSL access. The password is located in the PASSWORD_y field. There is no option to disable the password for BSL access.

Before unlocking the BSL by sending the correct password to the BSL core, there is no access granted to most BSL functions. Refer to the *MSPM33 Bootloader (BSL) User's Guide* for more details. If an incorrect password is provided to the BSL, the BSL halts for two seconds, after which make a secondary attempt to send the correct password. After three failed password attempts, the security alert feature (Section 3.2.4.3) activates.

The BSL access password supports 256-bit SHA encrypted (the length of the original password is 256-bit), refer to Section 3.1.2.2 for the SHA2-256 calculation flow. The BSL configuration stores the hash value of the password (256-bit, calculated from 256-bit data with all bytes as 0xFF).

3.2.4.2 Read-Out Feature

The device that supports a ROM BSL has a configurable property in the BSL configuration to read the device memory back through the BSL interface. All flash memories are fully accessible through the BSL host, while partial SRAM memory is restricted to access. See the *SRAM Memory Usage* section of the *MSPM33 Bootloader (BSL) User's Guide* for more details.

By default, the read-out feature is disabled for safety. Users can enable the read-out features through the BSLCONFIG0.READOUT field in the BSL configuration.

3.2.4.3 Alert Feature

The alert feature is applied in the ROM BSL for cases where there are occurrences of more than three incorrect BSL password checks. By default, the device does not perform additional actions when a security alert is raised. Users can configure the alert behavior through the BSLCONFIG1.SECURITY_ALERT_LEVEL field.

In the BSL configuration, two additional alert behaviors are supported:

- Trigger a BSL factory reset. If the sectors in the MAIN or NONMAIN flash are static-write protected, the sectors are not affected by the BSL factory reset.
- Reconfigure the NONMAIN region to disable the BSL. Reconfiguring is not supported if the NONMAIN flash is static-write protected.

Set the flash memory static write protection (Section 3.1.3) to retain specific MAIN flash memories or the entire NONMAIN flash memory.

3.2.4.4 Application Integrity Check

The BSL supports returning an application version number through the BSL interface. This support allows the BSL host to interrogate the firmware version without the ability to read the firmware. The version field address is 32-bits in length. Program the version value in the corresponding MAIN flash.

Use the APP_REV_POINTER field to set the address of the application version word. The version data only returns if the specified address corresponds to a valid flash memory address. If the given flash address is not programmed, then the ROM BSL returns 0h.

3.2.5 BSL Checksum

The MSPM33 device provides a CRC check for the BSL configuration data during the boot stage. The BSL CRC checksum shares the same calculation flow as the BCR checksum. See Section 3.1.9 for a detailed introduction.

TI recommends using the SysConfig tool to automatically calculate the BSL checksum, see Section 4.3 for more details.

3.2.5.1 CRC Check Fail Handling

If the BSL configuration data fails the CRC check during BSL invocation, a catastrophic boot error results and the following limitations are imposed:

- The error cause is logged in the CFG-AP as a boot diagnostic
- The BSL is not invoked, even if the BSL is configured as enabled
- The user application is not started
- No application debug access is enabled
- The boot process re-attempts up to three times
  - If the 2nd or 3rd attempts pass, the device boots normally
  - If the 3rd attempt does not pass, no further boot attempts are made until the next BOR or POR

4 NONMAIN Configuration With SysConfig

This section introduces how to configure the various functional options of the NONMAIN memory using the TI graphical configuration tool, SysConfig.

4.1 SysConfig Introduction

The SysConfig tool is an intuitive and comprehensive collection of graphical utilities for configuring pins, peripherals, subsystems, and other components, which helps users manage, expose, and resolve conflicts visually so that more time is spent on creating applications. The output files of SysConfig include header and source files for use with software development kit (SDK) examples or to configure custom projects.

The SysConfig tool is delivered as a standalone installer. Manually link the installer into TI's Code Composer Studio™ (CCS) software, IAR Embedded Workbench® (IAR), or Keil®, or use the cloud tools portal. As part of the TI ecosystem, CCS has integrated SysConfig by default. See the SysConfig User Guide for more details.

Figure 4-1 shows the part of the tool which configures the NONMAIN region, and users can select the *question mark* for more details on the component.

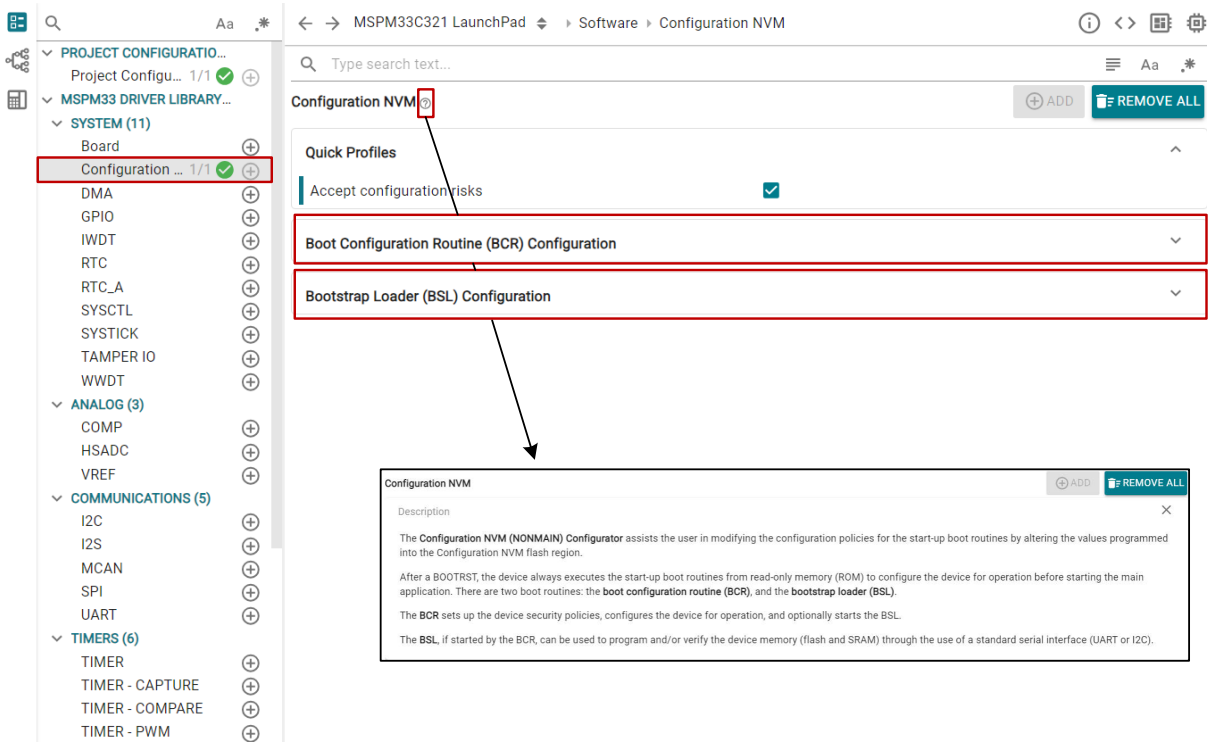


Figure 4-1. SysConfig Tool: Configure the NVM

4.2 BCR Configuration With SysConfig

Figure 4-2 shows the BCR configuration of the MSPM33C321A device (Type-A layout). According to the BCR configuration, the access policy, debug policy, mass erase, and factory reset policy can be configured for different security scenarios, see Section 3.1.2 for a detailed introduction. The BCR configuration ID is used with a default value and does not support modification in the tool. The CRC checksum is automatically calculated based on the customized BCR configuration.

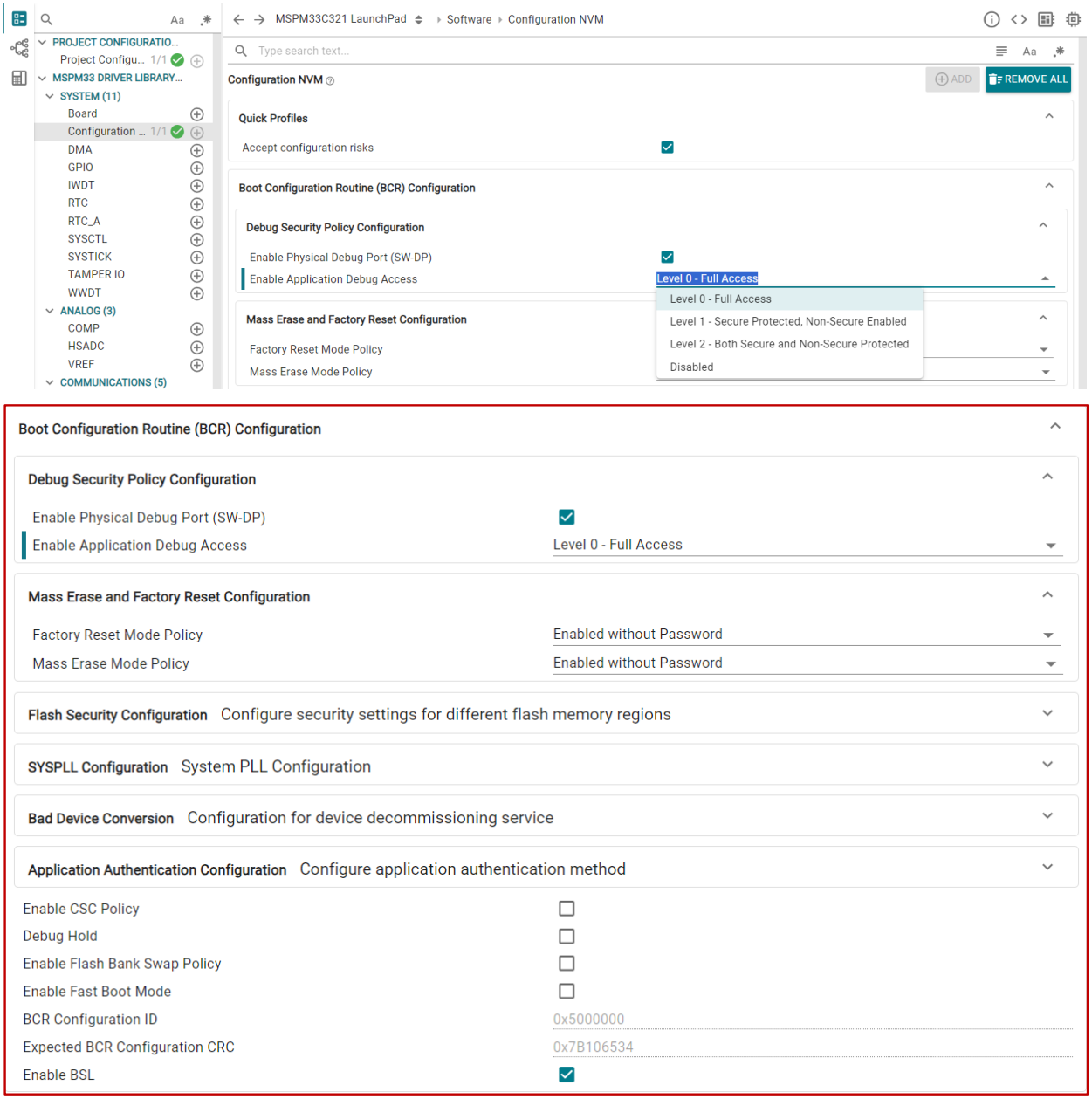


Figure 4-2. BCR Configurations

4.2.1 Password Configuration

For the devices supporting the SHA2-256 password, set the 256-bit encrypted password in the NONMAIN BCR configuration. Figure 4-3 shows the SWD password configuration of the MSPM33C321A device (Type-A).

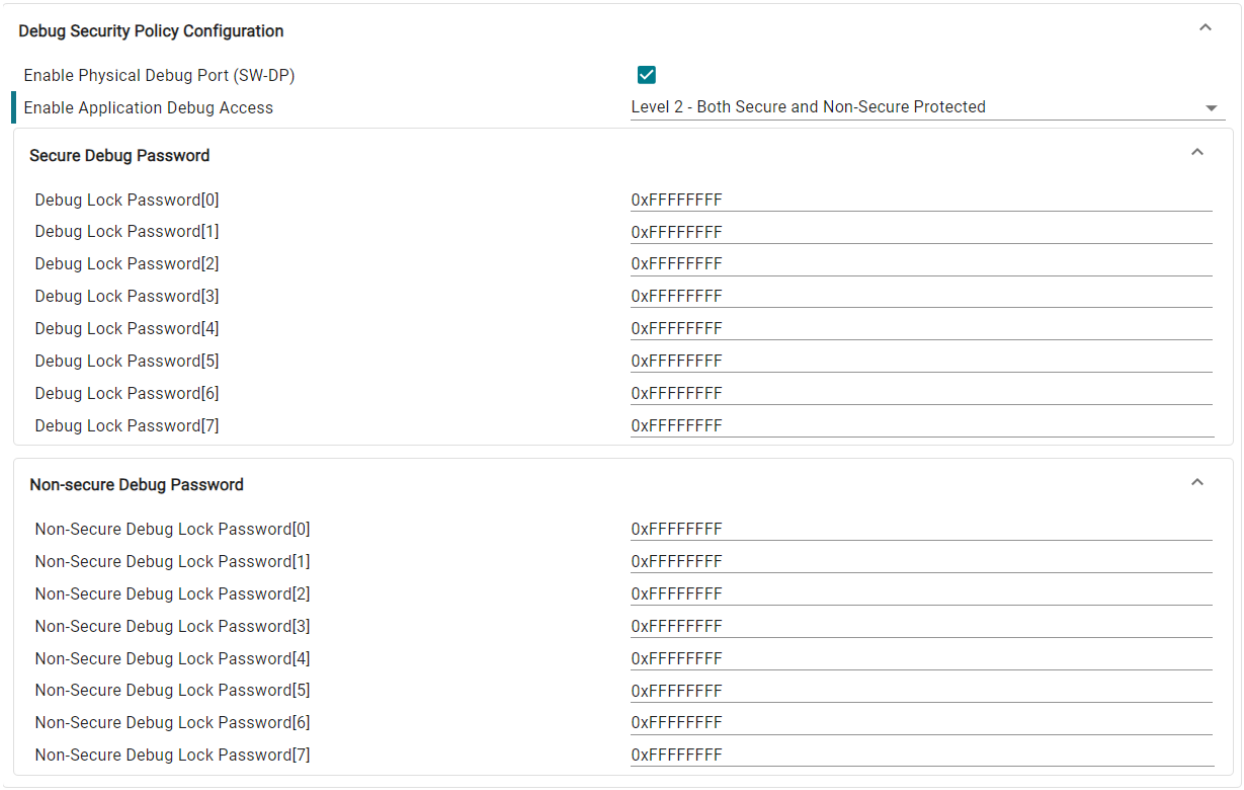


Figure 4-3. SHA2-256 SWD Password Configuration

The SysConfig tool integrates the SHA2-256 password generation flow (see Section 3.1.2.2), shown as Figure 4-4. If the device supports the SHA2-256 password, select the question mark next to the *Secure Debug Password* to display the example process flow. A SHA256 online tool is also introduced in the example.

The example flow is designed for all 256-bit SHA2-256 password generation, including:

- Debug access
- Factory reset
- Mass erase
- Bad device conversion
- BSL access

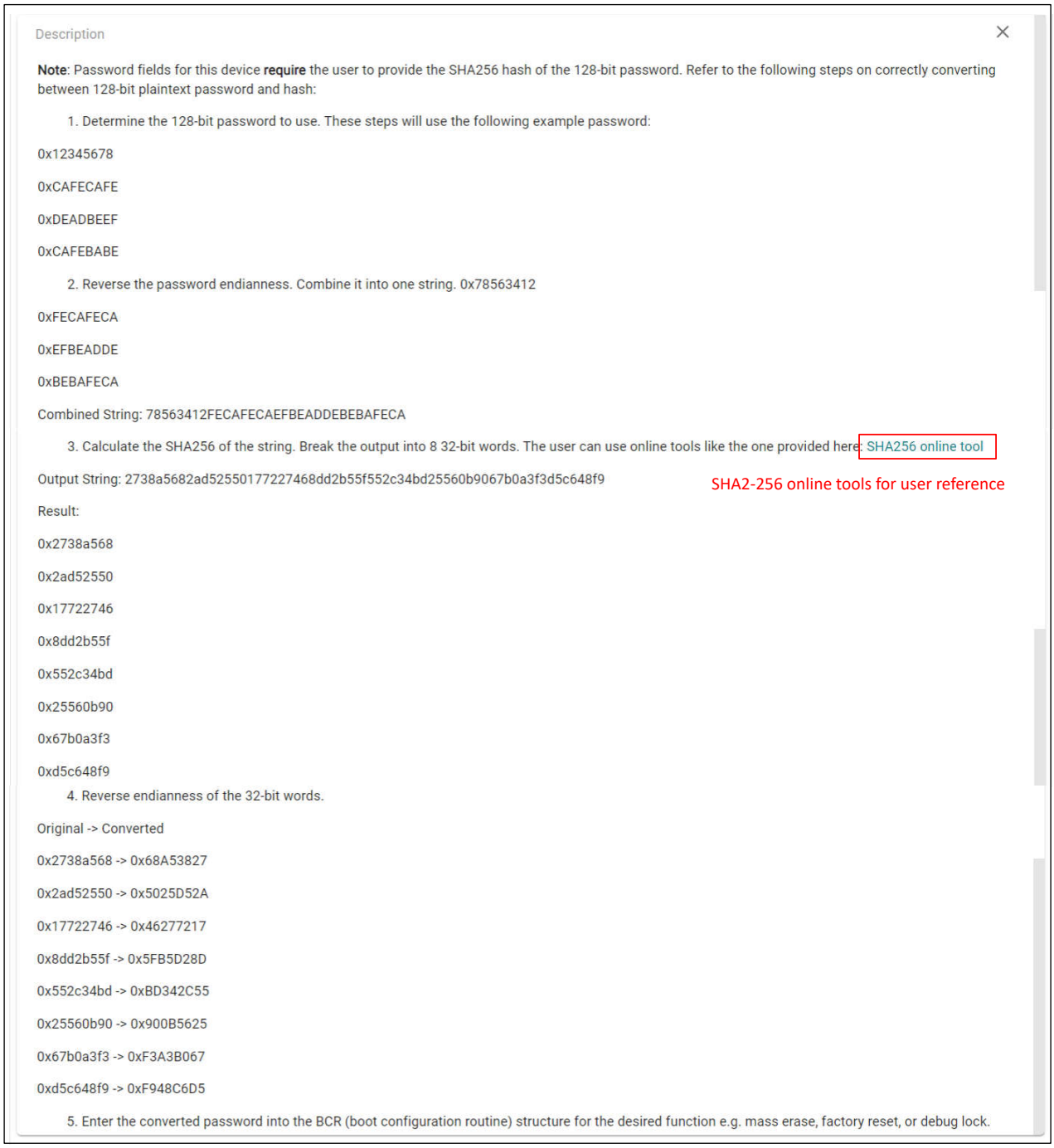


Figure 4-4. SHA2-256 Password Generation Flow

4.2.2 SYSPLL Configuration

Using the SysConfig tool, users can configure three levels of SYSPLL operation mode, as shown in Figure 4-5.

- NOPLL – PLL is not used
- PLL_80 – PLL is configured for 80MHz operation
- PLL_CUSTOM – PLL uses custom configuration

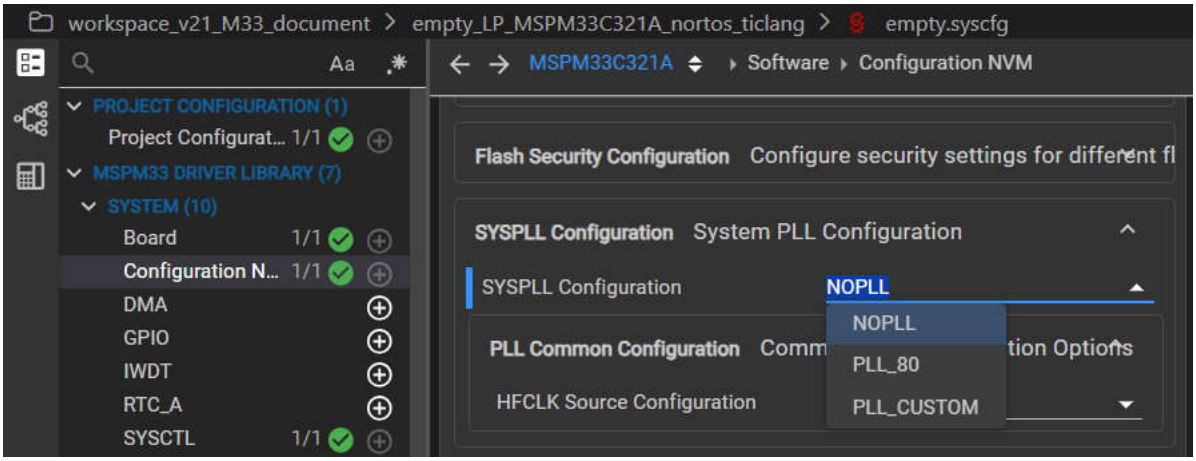


Figure 4-5. SYSPLL Configuration

Users can also configure the high-frequency clock source which can be used as an input to the system PLL when PLL is enabled, or directly as HFCLK when PLL is disabled. The source can be configured to the three following options, as shown in Figure 4-6:

- DEFAULT (0xFF): Use SYSOSC (16MHz)
- HFXT_20MHZ (0xAA): Use 20MHz HFXT crystal
- HFXT_40MHZ (0xBB): Use 40MHz HFXT crystal

This configuration affects the input source of PLL (when PLL is enabled), the direct HFCLK source (when PLL is disabled), and the MCAN clock source configuration.

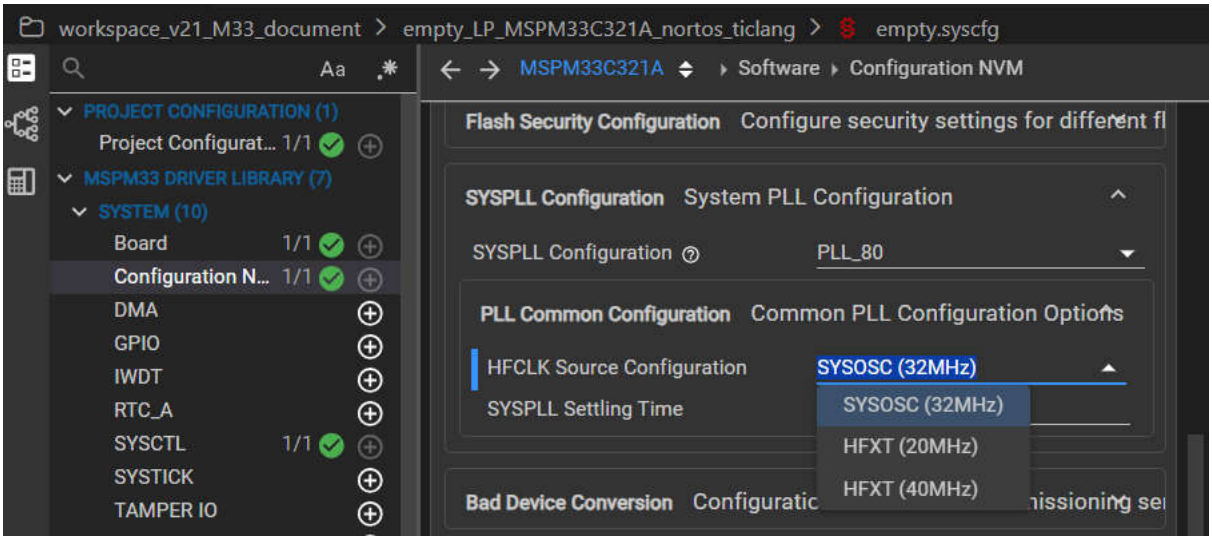


Figure 4-6. HFCLK Source Configuration

4.2.3 Flash Security Configuration

To enable the static write protection, security protection, or privilege protection, set the bit of the relating field as 0. The minimum memory range is 2KB (one sector) or 16KB (eight sectors) and is dependent on the protected sector order. For more details, see Section 3.1.3.

Using the SysConfig tool, users can choose specific sectors to apply static write protection, security protection, or privilege protection.

Figure 4-7 shows how to configure the static write protection on the MAIN FLASH bank0 of the MSPM33C321A device. For the first 32 sectors, write protection can be applied to each sector (2KB) individually. For the remaining sectors, write protection can be applied to each block (eight sectors) individually.

Figure 4-8 shows how to configure the security protection on the MAIN FLASH bank1 of the MSPM33C321A device. For the first 32 sectors, security protection can be applied to each sector (2KB) individually. For the remaining sectors, security protection can be applied to each block (eight sectors) individually.

Figure 4-9 shows how to configure the privilege protection on DATA FLASH of the MSPM33C321A device. For the first 32 sectors, privilege protection can be applied to each sector (2KB) individually. For the remaining sectors, privilege protection can be applied to each blocks (eight sectors) individually.

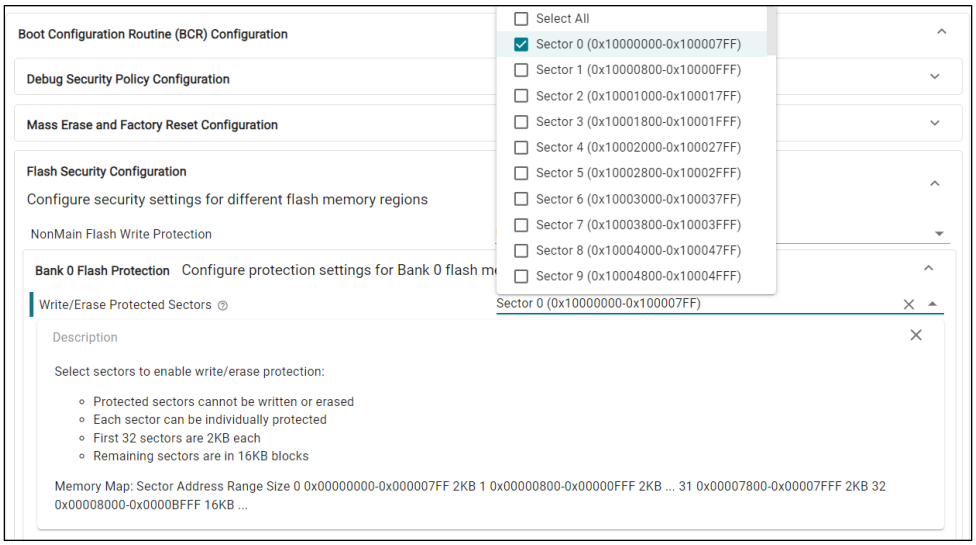


Figure 4-7. Flash Static Write Protection

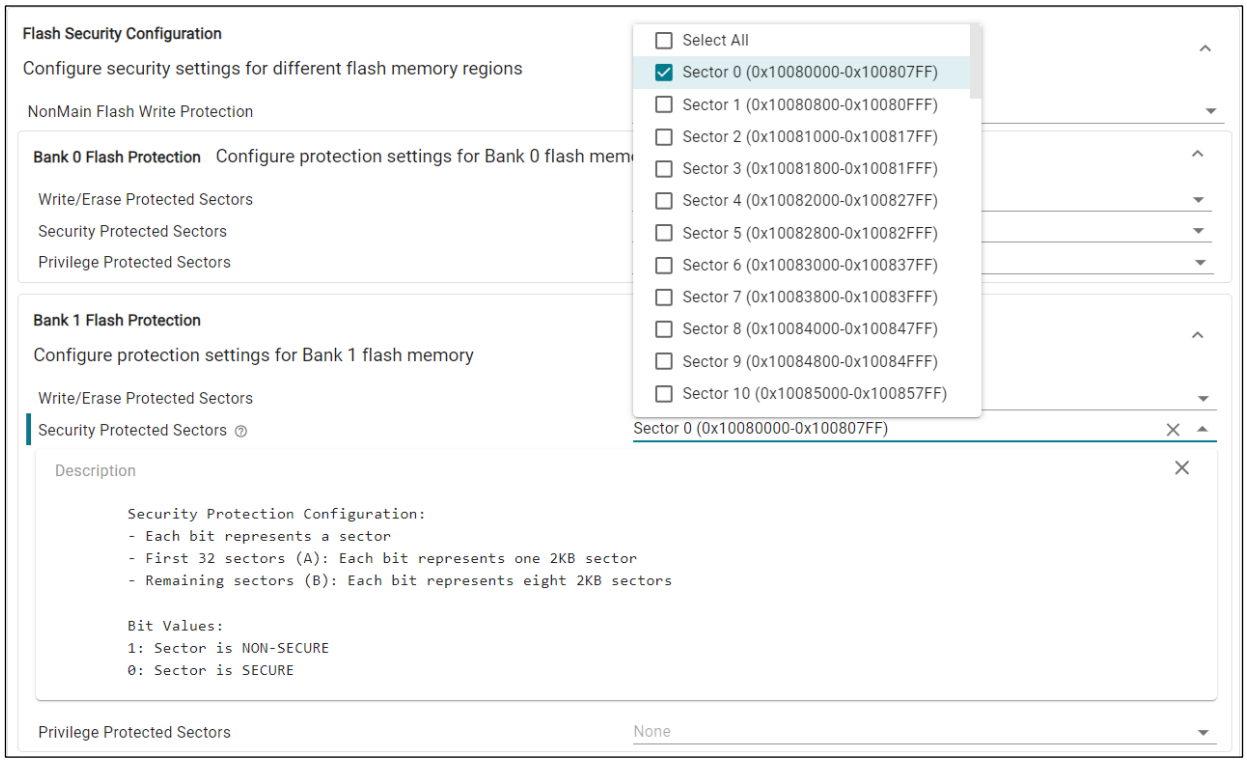


Figure 4-8. Flash Security Protection

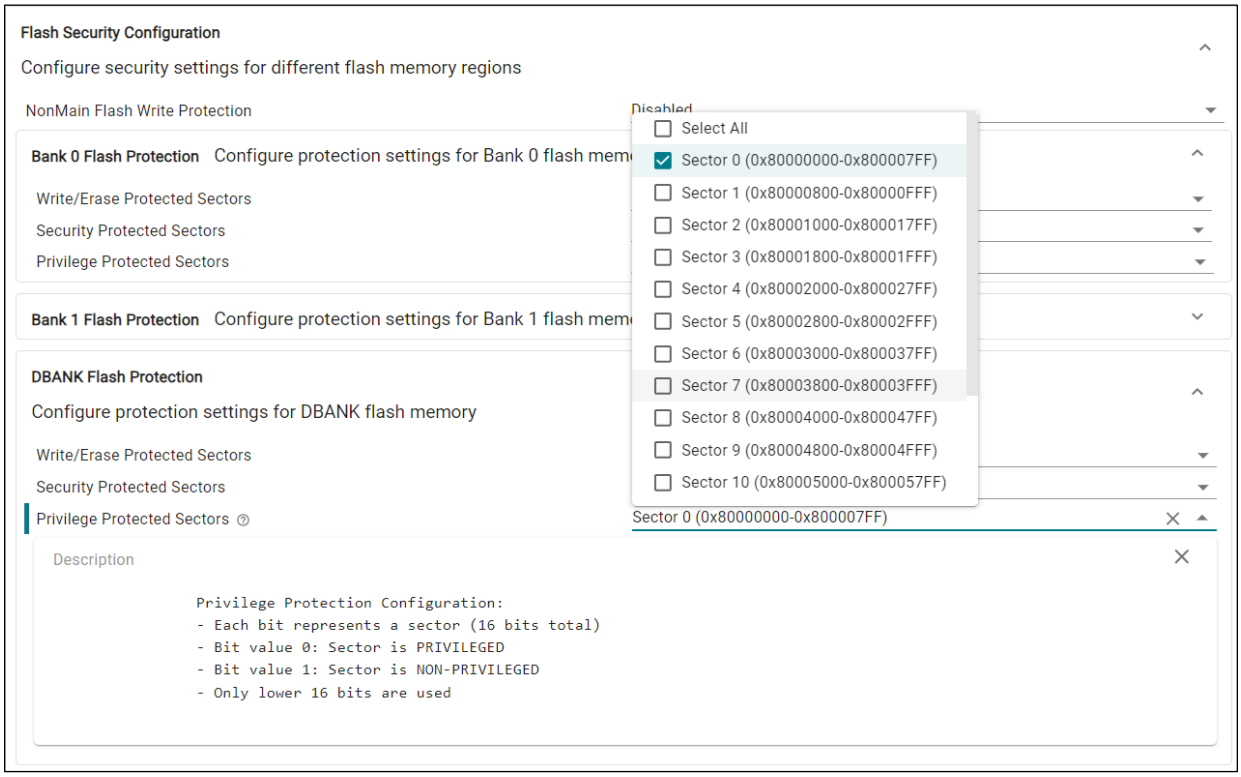


Figure 4-9. Flash Privilege Protection

The SysConfig tool also supports NONMAIN static write protection, after the device locks the write and erase operation on the NONMAIN region, the NONMAIN region is then immutable. Only the factory reset generated using the DSSM command restores the NONMAIN.

4.2.4 Application Digest Check

For application digest check, users can set up CRC32 digest check or SHA2-256 digest check in NONMAIN BCR configuration.

Figure 4-10 shows the CRC32 digest check Configuration of MSPM33C321A (TYPE-A) based on the setting in Section 3.1.7.1. Example data is Figure 3-2 located at 0x00010000.

Figure 4-11 shows the SHA2-256 digest check Configuration of MSPM33C321A (TYPE-A) based on the setting in Section 3.1.7.2.Example data is Figure 3-3 located at 0x00010000.

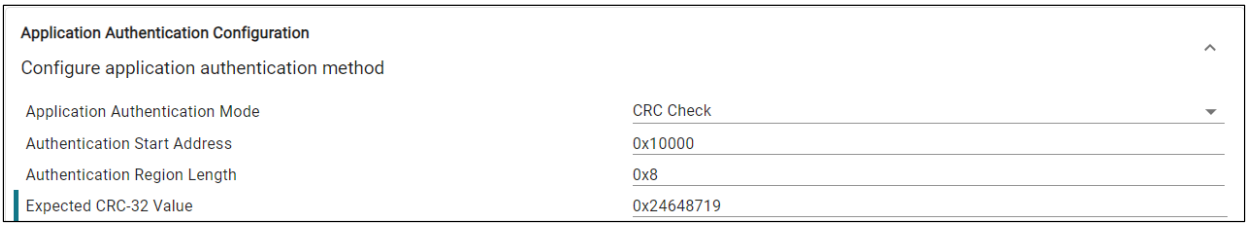


Figure 4-10. CRC32 Digest Check Configuration

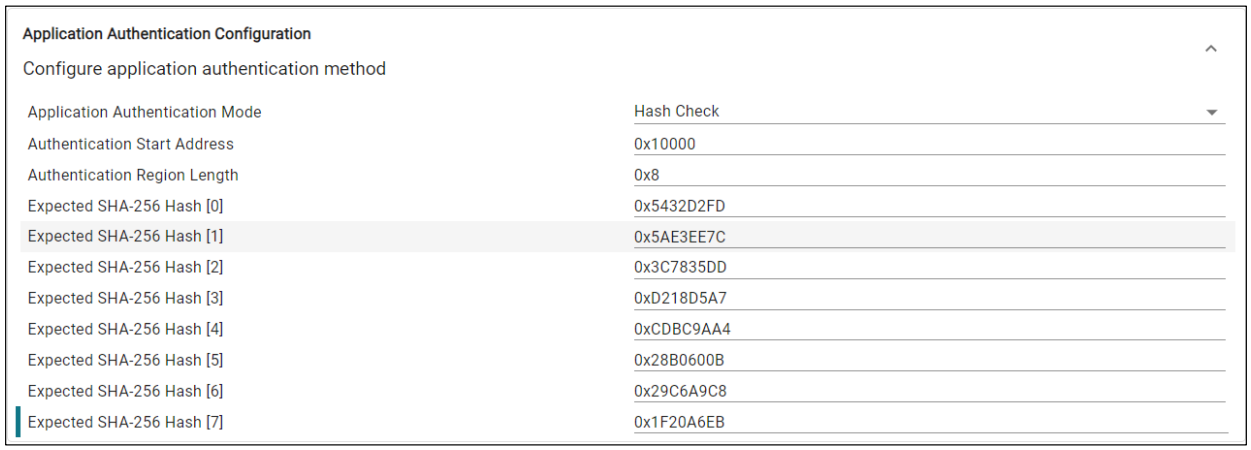


Figure 4-11. SHA2-256 Digest Check Configuration

4.2.5 Other BCR Configurations

Some BCR configurations in SysConfig are straightforward and can be enabled or disabled, including the following:

- Enable fast boot mode
- Enable debug hold
- Enable CSC policy
- Enable flash bank swap policy
- Enable BSL mode
- Enable BSL invoke pin check
  - In the SysConfig tool, instead of the *BCR Configuration*, the *BSL Configuration* includes the selection of *Enable BSL Invoke Pin Check*, along with other BSL GPIO invoke configurations.

4.3 BSL Configuration With SysConfig

Figure 4-12 shows the BSL configuration of the MSPM33C321A device (Type-A layout).

Figure 4-12. BSL Configurations

4.3.1 BSL Access Password

For the device to support ROM BSL, the 256-bit BSL access password is always enabled. The password is SHA2-256 encrypted and the original password is all 0xFF (256-bit), so that the SysConfig tool stores the generated SHA2-256 value of the original password in the PASSWORD_y field.

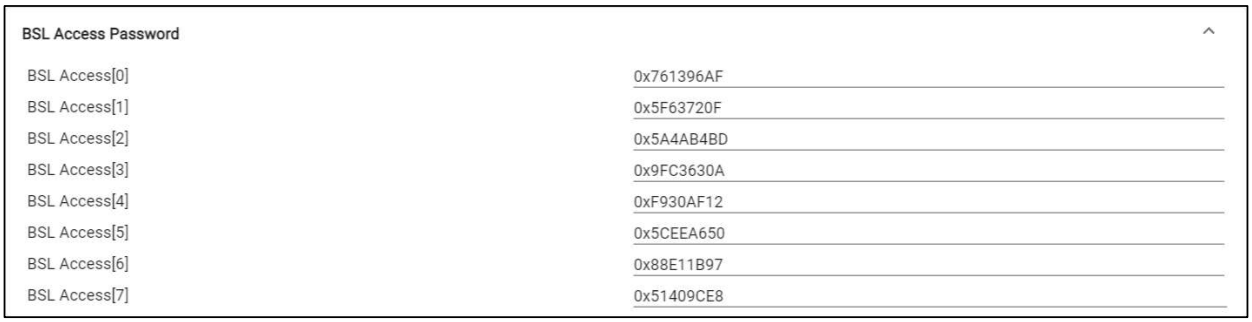


Figure 4-13. SHA2-256 BSL Default Access Password

4.3.2 BSL Invoke Pin Configuration

The device supports the ROM BSL using a customized invoke pin property. Section 4.3 shows the details of the configuration.

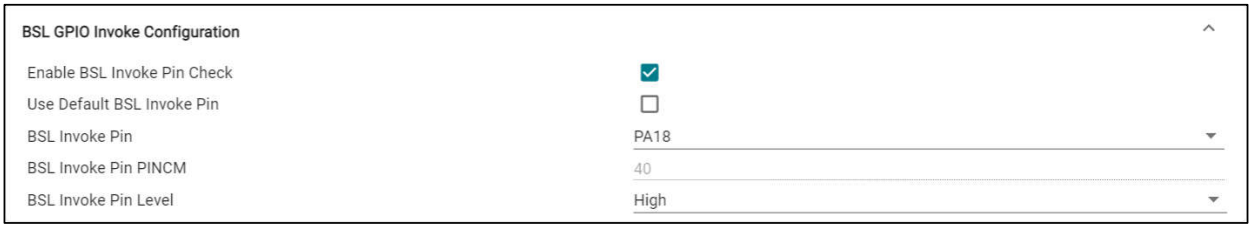


Figure 4-14. BSL Invoke Pin

If the user disables the BSL invoke pin check in the BSL configuration, the BCR skips the hardware pin invoke check while the software BSL entry is still available. After disablement, the BOOTCFG1.BSL_PIN_INVOKE field of the BCR configuration is set to disabled in the SysConfig tool, and the BSL pin configurations remain unchanged.

4.3.3 BSL Communication Interface

For the device that supports the ROM BSL, the I2C, UART and CAN interfaces are provided as common BSL interfaces. For the device that supports the USB peripheral, a USB interface supporting the DFU is applied as the BSL interface.

The BSL configuration provides flexibility for users to customize the BSL interface, as shown in Section 4.3. By default, the BSL configuration supports pin selection, I2C target address changes, and UART baud rate modifications.

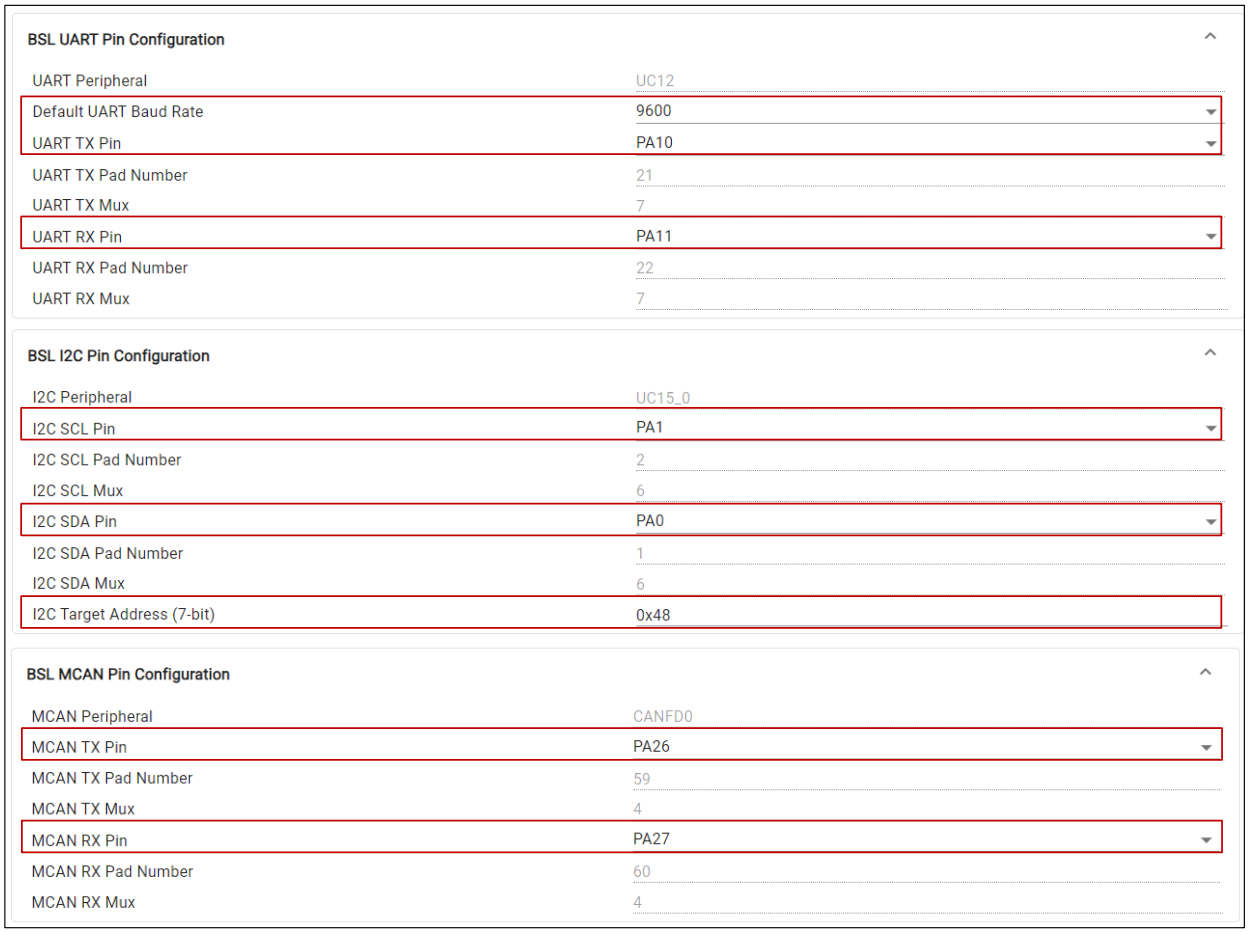


Figure 4-15. BSL Communication Interface

4.3.4 Other BCR Configurations

Other than the BSL configuration ID, there are a few more features that support customization using the SysConfig tool, as shown in Figure 4-12, including:

- BSL application version
- BSL read out enable
- BSL security alert configuration

Whenever the BSL configuration changes, the SysConfig tool automatically calculates the *CRC Checksum* value, so the user only focuses on the customized BSL features and can easily generate the NONMAIN firmware files.

5 NONMAIN Configuration in Application Code

The NONMAIN region also supports dynamic modification with the FLASHCTL, which is widely used in the application code and customized bootloader. Figure 5-1 and SDK example provide a generic flow for modifying the NONMAIN flash field in the application codes for user reference.

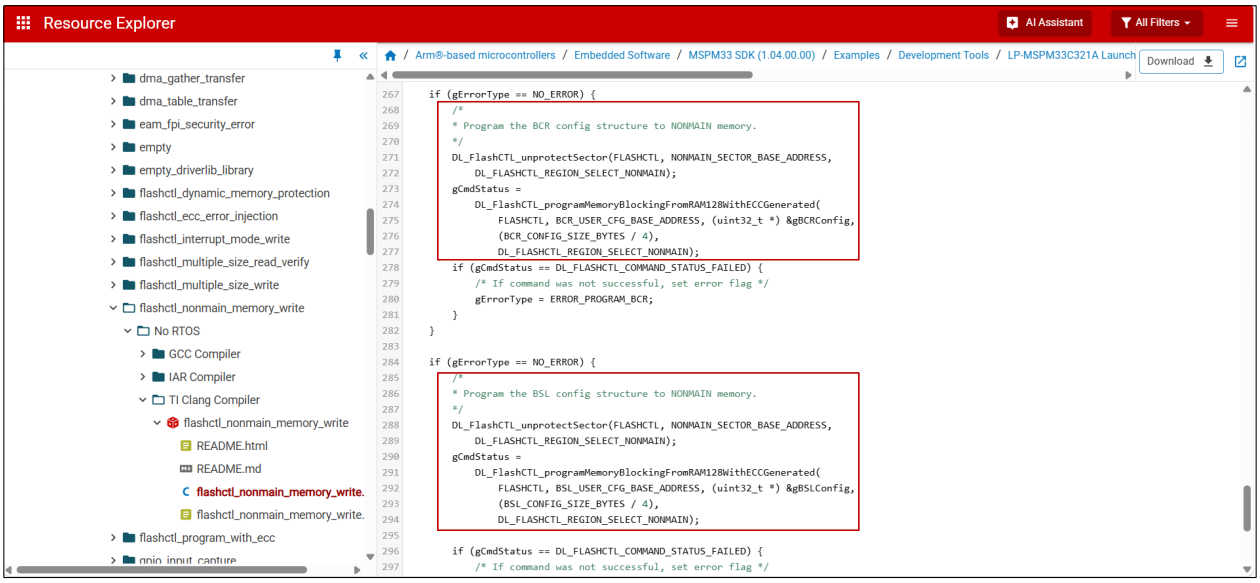


Figure 5-1. Example Project With NONMAIN Modification

ECC is required to program with the NONMAIN memory, as the boot code reads the NONMAIN memory with an ECC check. Every NONMIAN register modification requires the user to erase the entire NONMAIN sector and load in the new value of the entire NONMAIN configuration register.

TI recommends erasing and programing the NONMAIN field in a safety state. Frequently updated NONMAIN configurations in the application code introduces the risk that the NONMAIN field becomes empty (unstable power supply, and so forth.).

6 NONMAIN Operation With IDE Tool

TI recommends using the SysConfig tool to manage the NONMAIN configuration. The SysConfig tool integrates into the TI Code Composer Studio™ (CCS) software. TI provides the standalone version of the SysConfig tool to use with other IDE tools, such as Keil® or IAR®. For more details on using the SysConfig with different IDE tools, see Section 10.

6.1 NONMAIN Configuration Files

Figure 6-1 shows that the SysConfig tool generates two files (boot_config.c and boot_config.h) to configure the NONMAIN flash memory.

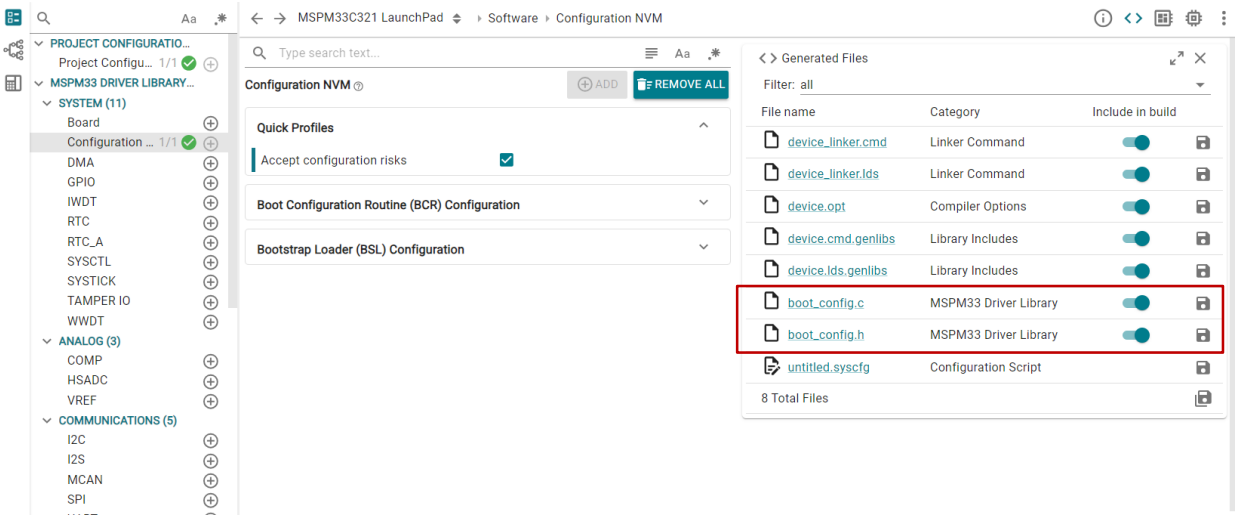


Figure 6-1. Configuration Files

The boot_config.c file includes all of the configurations of the NONMAIN structure, and the configurations automatically compile into the output file. The boot_config.c file includes the default value set in the SysConfig tool. Users *cannot* manually modify these two files, as the SysConfig tool manages and overwrites the files. After users modify the project-specific .syscfg files in the NVM component, these two files regenerate and update to the latest configuration.

To use the SysConfig tool to perform additional modifications, exclude the two files from build by unchecking the button *include in build*. Then, the SysConfig generated files (boot_config.c and boot_config.h) do not compile so the user can manually add the files into the project and perform additional modifications.

6.2 Project Erase Property

The NONMAIN flash region is a nonvolatile memory which requires the erase operation before introducing a new write value. To successfully load the NONMAIN region firmware into the device, set the erase property to erase the NONMAIN.

For Code Composer Studio™ studio, right-click on the project and open the project property. Figure 6-2 shows the details.

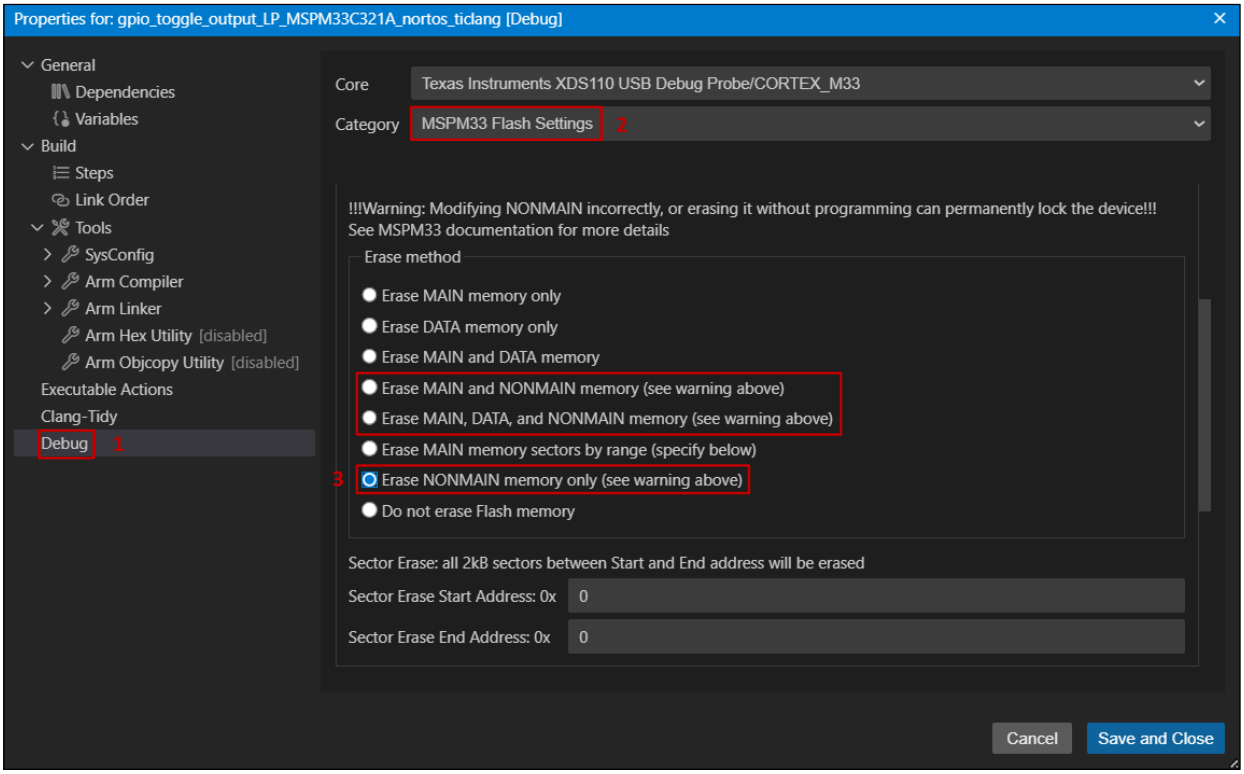


Figure 6-2. CCS Erase Property Settings

For Keil®, select and open the options for target, add NONMAIN in the debug settings. Figure 6-3 shows the Keil erase property settings.

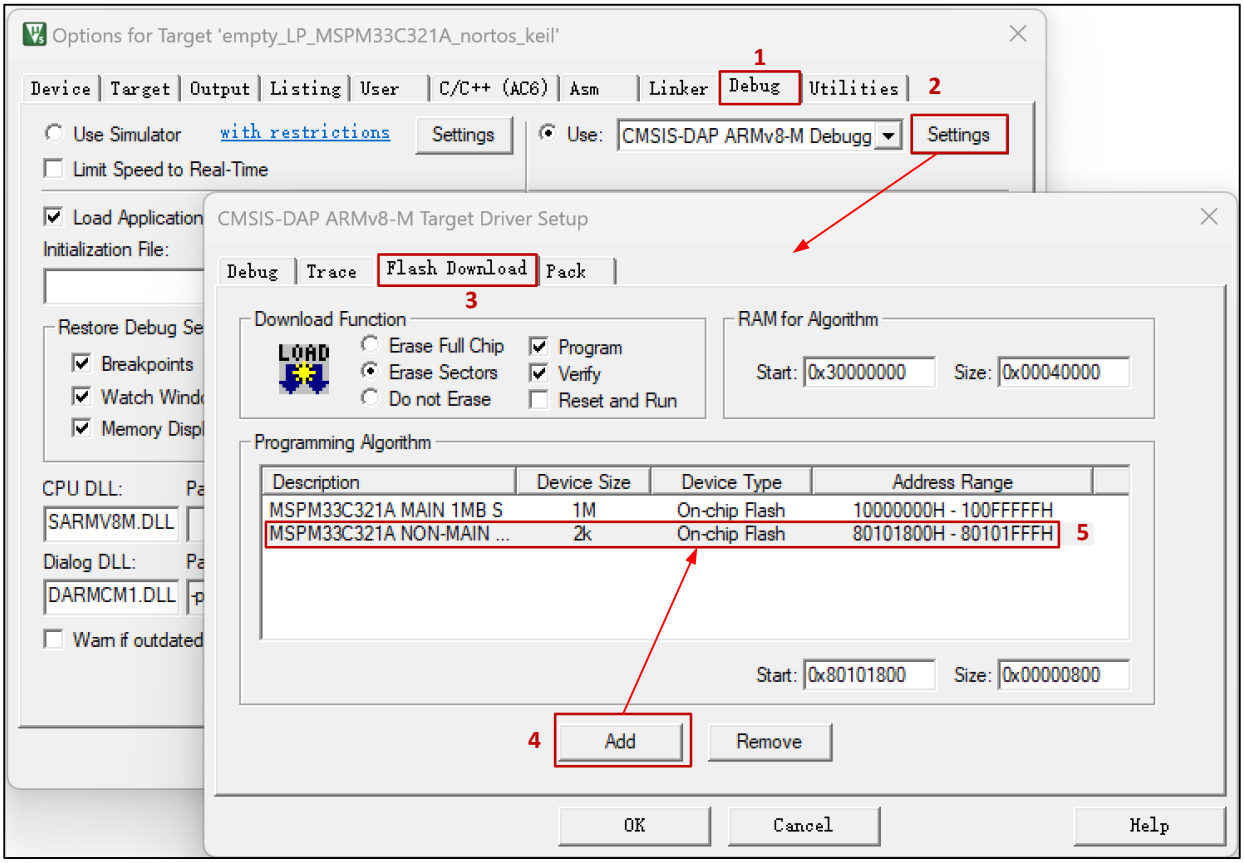


Figure 6-3. Keil Erase Property Settings

For IAR®, right-click the project and open the options. Verify that the non-main region is added with an extra parameter. Figure 6-4 shows the IAR settings.

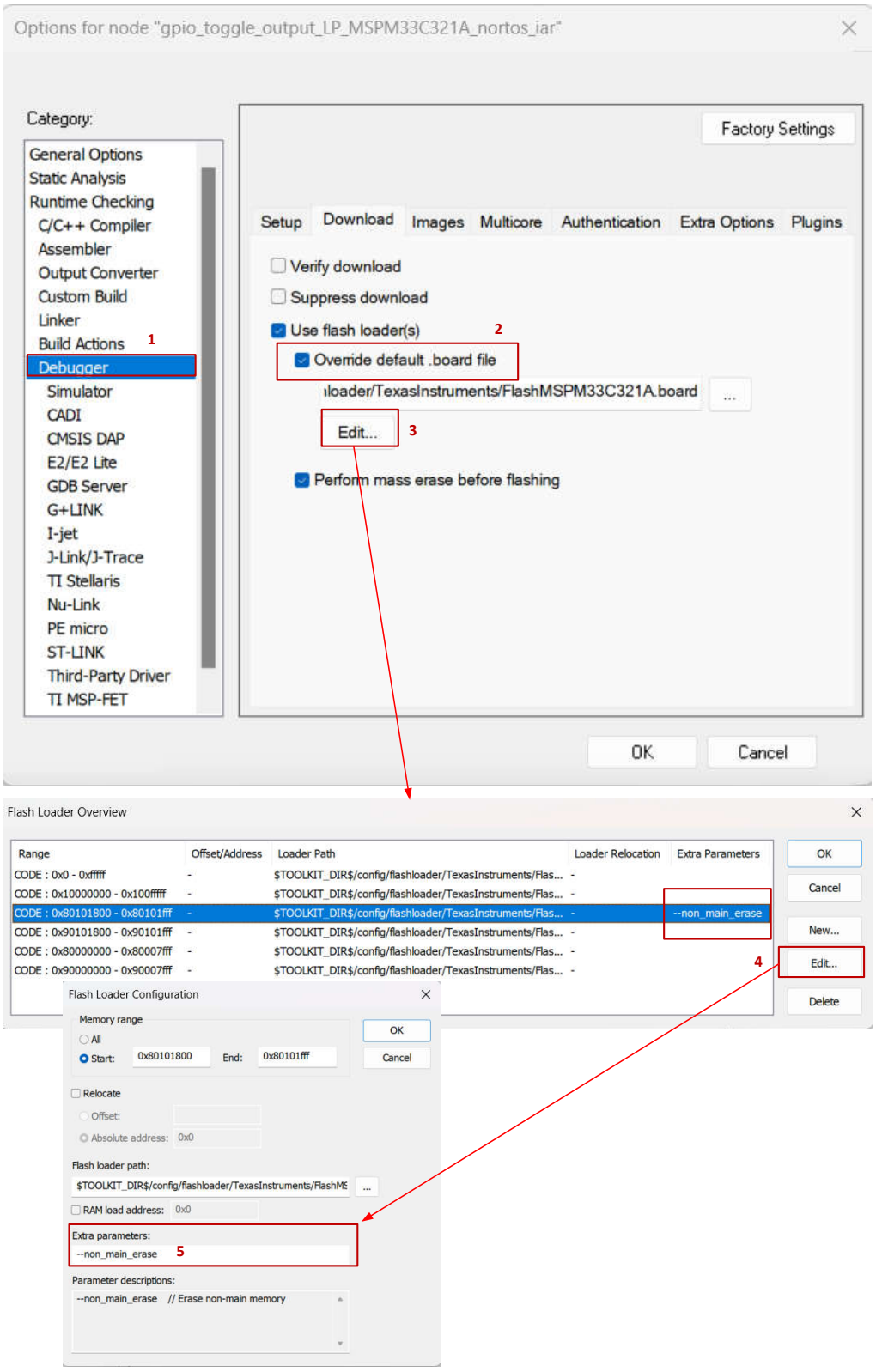


Figure 6-4. IAR Erase Property Settings

After the NONMAIN configuration is finalized and programmed, TI recommends removing the NONMAIN field to avoid reprogramming NONMAIN with the same value each time.

6.3 Password-Protected Debug

When the SWD debug policy is set as an encryption with a password, generate the DSSM command for SWD password authentication before debugging or downloading new firmware.

TI provides the .ccxml file for the MSPM33 debug configurations with the XDS110 tool. Figure 6-5 shows the steps to set the 128-bit password for following DSSM command:

1. Open the *.ccxml* file in the *targetConfigs* folder.
2. Select *Advanced* to open the XDS110 *Target Configuration*.
3. Select *MSPM33 device* to open the *Device Properties Page*.
4. Input the 128-bit password in the SWD password box.

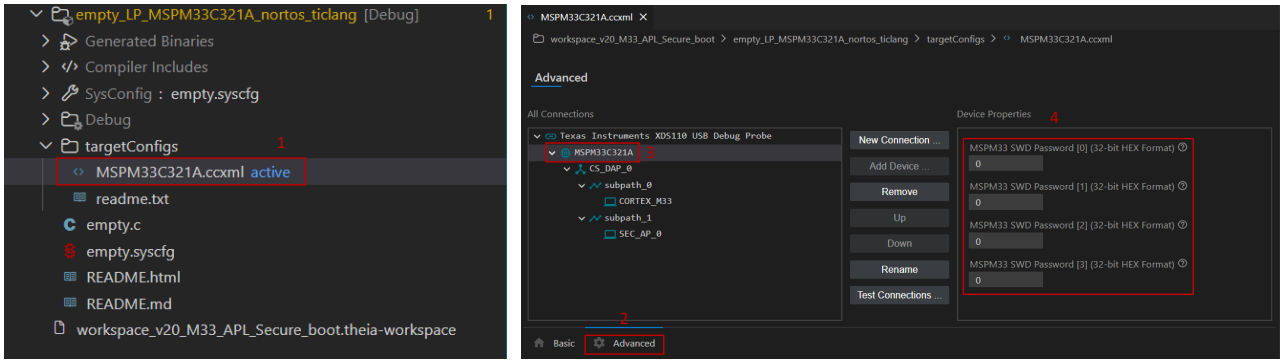


Figure 6-5. Set Password With .ccxml File

Note

The password set in the .ccxml file is applicable for the all DSSM operation passwords, including SWD access, factory reset, and mass erase.

After setting the correct password, run the DSSM script to unlock the SWD interface. Figure 6-6 shows the process.

1. Locate *targetConfigs* in the project and select the *.ccxml* file.
2. Right-click the .ccxml file and select *Start Project-less Debug*.
3. Select *Scripts* at the top-menu, then select *MSPM33 Device Commands*.
4. Select and generate the DSSM command.
5. Select *DebugAccessPasswordAuthentication_Auto* to unlock the SWD interface.
6. After successfully completing the command execution, locate the top menu. Select *Run* to *Debug Project* or *Flash Project*.

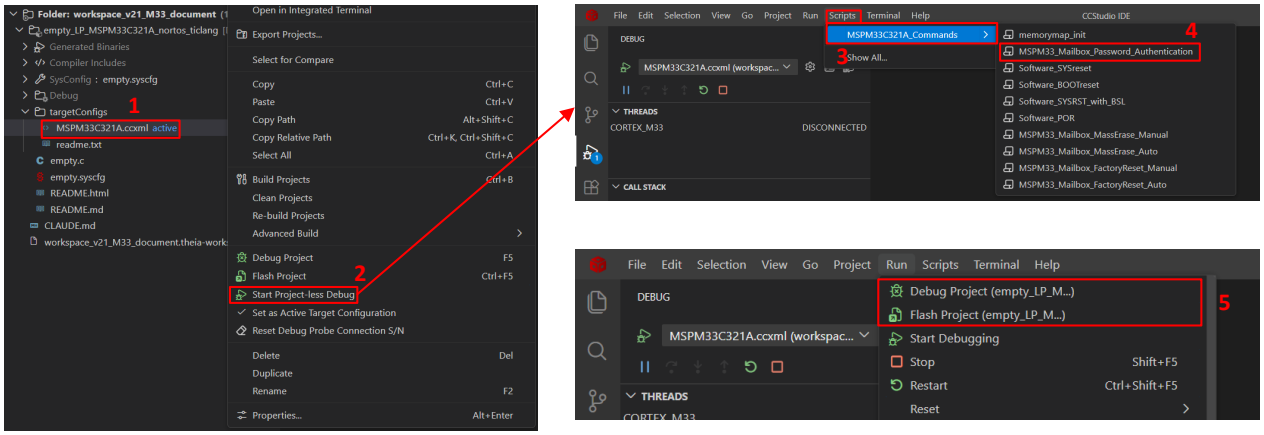


Figure 6-6. DSSM Command With CCS v21

Note

After successfully executing the password authentication sequence for debug access, the device remains in an unlocked state until the next BOOTRST.

7 NONMAIN Operation With Programmer Tool

The NONMAIN is a dedicated region of flash memory which stores the configuration data used by the BCR and BSL to boot the device. The region is not used for any other purpose. To change any parameter in the boot configuration, erase the entire NONMAIN sector and re-program both the BCR and BSL configuration structures with the desired settings.

Regarding the risk that an empty NONMAIN field permanently locks the device, TI recommends users reduce the time when the NONMAIN region is empty and wait for new data to load. Two kinds of approaches are preferred:

- Separately erase the existing NONMAIN and load the new NONMAIN immediately.
- Erase the MAIN and NONMAIN together, and write the NONMAIN and MAIN in sequence.
  - If the programmer *cannot* verify the order of erase and program, then load the MAIN and NONMAIN firmware separately.

7.1 NONMAIN Operation With UniFlash

UniFlash is a software tool for programming on-chip flash on TI microcontrollers and wireless connectivity devices, and onboard flashes for TI processors. UniFlash provides both graphical and command-line interfaces.

To program NONMAIN flash memory, select the erase property which includes NONMAIN memory. Figure 7-1 shows this process.

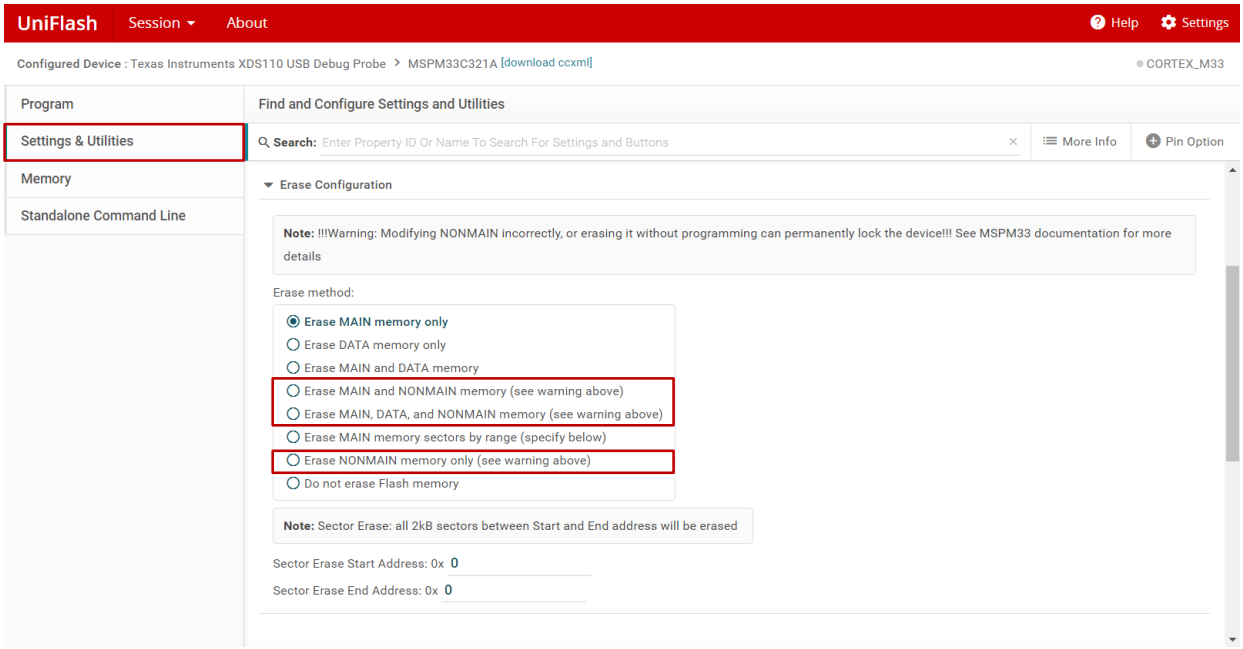


Figure 7-1. NONMAIN Operation With UniFlash

To program the NONMAIN only, follow these steps:

1. Prepare the firmware to only include the NONMAIN region firmware. The TI-Hex or Intel-Hex format easily processes to get the separate NONMAIN region firmware.
2. Select the erase property as *Erase NONMAIN memory only (see warning above)*.
3. Program the device. Only the NONMAIN is erased and reprogrammed.

7.2 NONMAIN Operation With J-Flash

J-Flash™ is a flash programming software by SEGGER that allows users to program the internal and external flash memory of embedded microcontrollers (MCUs). Control J-Flash using a GUI (J-Flash) or a command line interface.

To program the NONMAIN field, enable the NONMAIN bank and select the erase property to include the NONMAIN field. Figure 7-2 shows the NONMAIN operation with J-Flash.

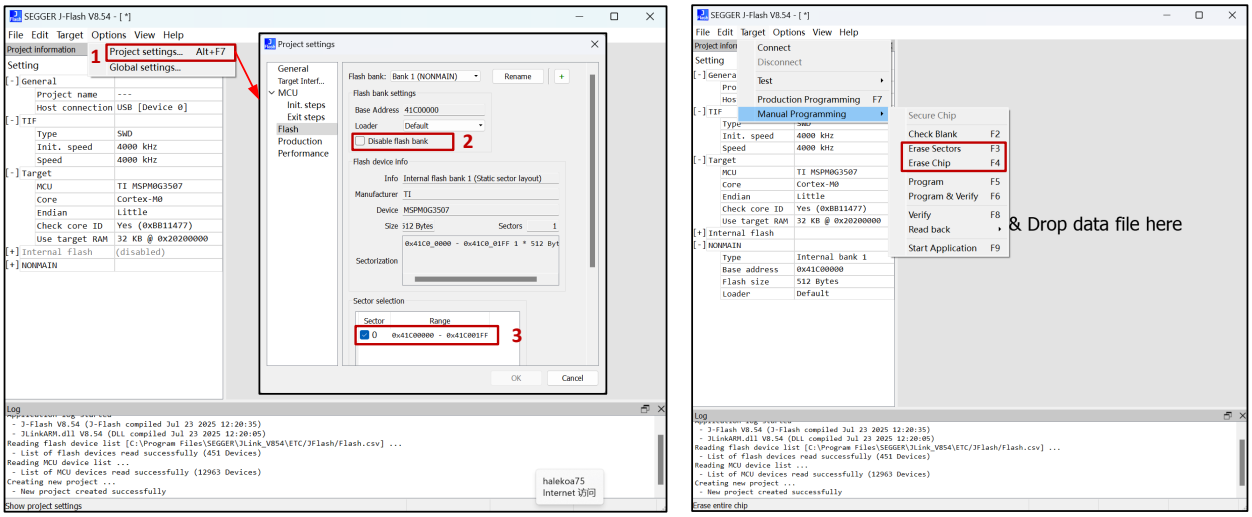


Figure 7-2. NONMAIN Operation With J-Flash

If selecting the manual programming method, make sure the device is not reset before loading the new NONMAIN firmware. If the device resets, the empty NONMAIN permanently locks the device.

Note

TI recommends using the latest J-Flash version for the new MSPM33 device.

To program the NONMAIN only, follow these instructions:

1. Prepare the firmware, only including the NONMAIN region firmware.
2. Select the erase property to exclude the MAIN flash memory: either disable the MAIN flash or select none of the MAIN flash sectors.
3. Program the device with *erase sectors*. Only the NONMAIN field erases and reprograms.

8 Frequently Asked Questions (FAQs)

8.1 MCU Locked State Analysis

If the MSPM33 device enters the locked state, the MSPM33 potentially *cannot* connect through the SWD interface. This section introduces the common reasons why the MCU is locked.

8.1.1 Hardware Issue Analysis

The following hardware factors can cause the device to lockup:

- Defects in the hardware circuit design
- Problem connecting to the debugger
- Periodic external reset signal (to NRST)

8.1.1.1 Hardware Circuit Design

Verify that the MSPM33 device has the appropriate external capacitor in a few power pins to run in a stable manner. Check that the circuit is connected to the VDD, Vcore (if supported), and NRST pin and the pin voltage.

Find the reference circuits in the *Basic Application Schematic* section of device-specific datasheet. Expect that the Vcore voltage (if supported) is 1.35V (typical), representing that the internal PMU circuit for the digital core functions normally.

8.1.1.2 Debugger Connection

If debugging the MSPM33 device with a standalone debugger, check the hardware pin assignment. To successfully connect the device, verify that SWDIO and SWCLK are well-connected in the correct order. Sometimes, the debugger does *not* power the device by default; check that the device is properly powered using the onboard or external power rail.

If debugging the MSPM33 device in a noisy environment, try a lower debugging rate, or add a small capacitor in parallel to the SWCLK and SWDIO pin.

8.1.1.3 External Reset Signal

If an external reset signal connects to the NRST line, disable the reset signal or keep the signal high when debugging the MSPM33 device. The reset signal (low active) generates to the NRST pin and breaks the SWD connection. If a continuous reset signal is in place, the device can enter into a locked state.

8.1.2 Software Issue Analysis

The following software factors can cause the device to lockup:

- CPU enters a fault state
- BCR configuration
- Low power mode (STOP or STANDBY)
- SHUTDOWN IO state
- SWD IO (SWCLK or SWDIO) function disabled
- WDT or IWDT reset (if enabled in application code)
- Software POR or BOOTRST

8.1.2.1 CPU Enters a Fault State

When the CPU enters a fault state, the Arm® device automatically performs a continual self-reset until the CPU leaves the fault state. The CPU enters the fault state during a nested exception (for example, double-hard fault or NMI), caused by illegal CPU activities, such as invalid address modification or peripheral misconfiguration.

When the CPU enters a fault state, the continuous reset behavior breaks the SWD connection, and the device enters the locked state.

8.1.2.2 BCR Configuration

When the BCR configuration is set to SW-DP disabled or SWD-access disabled, the device enters the locked state in debug mode. When the BCR configuration is set to SWD-access encrypted, the device enters the locked state in debug mode if incorrect passwords are loaded.

Additionally, loading the incorrect NONMAIN configuration can result in the device entering a permanently locked state.

8.1.2.3 Low Power Mode (STOP or STANDBY)

When the device enters STOP or STANDBY mode, the Arm® Advanced High-performance Bus Access Port (AHB-AP, Arm Debug) core is not discoverable, making the device enter a locked state. To actively connect to the AHB-AP, verify that the debugger first connects the PWR-AP. For more details, refer to the *Hardware Programming and Debugger Guide for MSPM0* application note.

XDS110 supports connecting the device in STOP or STANDBY mode. For other 3rd party debuggers, such as J-Link™, try the latest software version which fixes this low-power mode debug issue.

8.1.2.4 SHUTDOWN IO State

The digital IO pin states latch and retain upon entry to SHUTDOWN. The digital IO pin states include:

- Output low or high
- Pullup or pulldown
- Input or output enabled
- Drive configuration

After exiting SHUTDOWN mode, the IOs are held in the previous state until released by the application software setting the RELEASE bit in the SHDNIOREL register along with the matching KEY value.

When exiting SHUTDOWN, the serial wire debug (SWD) pins also remain locked until the application software sets the RELEASE bit. As a result, a debug connection cannot establish when waking up from SHUTDOWN mode until the IOs are released by application software.

The TI debug tool—XDS110—currently supports directly debugging the device wake up from SHUTDOWN mode.

8.1.2.5 SWD IO Function

After a POR, when the boot process completes and the CPU starts the application, serial wire debug (SWD) IOs are configured in SWD mode (SWDIO is pulled high, and SWCLK is pulled low).

Re-configuring the SWD pins as general-purpose input-output (GPIO) pins is possible in software to enable the use of these pins in an application when debug support is no longer required. To disable the functionality of the SWD, set the DISABLE bit in the SWDCFG register in SYSCTL along with the KEY (62h).

Once the SWD pin functions are disabled, the device remains in the lock state, and the SWD pin functions can only be re-enabled by triggering a POR.

8.1.2.6 WDT or IWDT Reset

When the WDT or IWDT is not fed within the correct time window, a reset occurs and can cause the device to enter a locked state.

Normally, the debugger generates SYSRST through the SYSCTL to disable the WDT. If the reset does not generate, or is set as the CPU level reset, the WDT can cause the device to enter a locked state and assert a reset during the SWD connection stage.

For MSPM33 devices containing the IWDT peripheral, the debugger generates the LFSS reset or IWDT reset to disable IWDT. Otherwise, the device can enter a locked state, as a SYSRST does not reset the LFSS component, including IWDT.

8.1.2.7 Software POR or BOOTRST

A software POR or BOOTRST breaks the SWD connection and can cause the device to enter a locked state if the break occurs when establishing the SWD connection.

Disable the software POR and BOOTRST to stably debug the MSPM33 device.

8.2 Unlock the MSPM33 Device

This section describes unlocking the device when debug-access fails due to a software issue.

8.2.1 Force MCU to Enter BSL Mode

For the device that supports a BSL, and the BSL is set as enabled (default enabled), use the hardware invoke method to force the MCU to enter BSL mode. The MCU then executes the BSL code in the ROM and does not execute the application code. The device remains connected to the SEC_AP and CS_DAP cores in BSL mode.

Note

When the MCU enters BSL mode, users cannot connect to the Arm® Cortex®-M33 core because the BSL is not debuggable by default. But users can still send a DSSM command as the CS_DAP core is available for connection.

In BSL mode, users can send a BSL command as Section 8.2.2, or send a DSSM command asSection 8.2.3 to recover the device. This approach help users unlock the device if a software issue causes the MCU to enter a locked state.

The following lists the recommended methods to force the MCU to enter BSL mode:

- Connect the BSL invoke pin (the default is PA18) to the VCC (with or without pullup resistor). Re-power the device.
- Connect the BSL invoke pin (the default is PA18) to the VCC (with or without pullup resistor). Force NRST low for more than 1s to trigger a POR.

Note

When the MCU enters BSL mode, the MCU enters STANDBY0 mode if the BSL connection command is not sent to a UART, I2C or CAN interface within 10s.

The approach is not viable in the following situations:

- The NONMAIN configuration has CRC check failure.
- The NONMAIN configuration disables the SWD interface or locks the debug access.
- IWDT is enabled through the device supporting the VBAT feature, requiring users to manually turn off the VBAT to disable the IWDT.
- The BSL hardware invoke method is disabled.

8.2.2 Send BSL Command

If the NONMAIN configuration disables the SWD interface or locks the debug access, send the BSL command to unlock the device after BSL mode entry.

The BSL communication interface (see Section 3.2.3) is applied to send the BSL command. The ROM BSL core supports the factory reset command to restore the default NONMAIN configurations. Refer to the *MSPM33 Bootloader User's Guide* for more details.

The approach is not viable in the following situations:

- The BCR configuration sets factory reset as disabled.
- The BCR configuration sets the NONMAIN static write protection as enabled.
- The BSL is disabled in the BCR configuration.

8.2.3 Generate DSSM Command

The MSPM33 device supports the DSSM command, including mass erase and factory reset.
The DSSM command is executed in boot code. Refer to Section 6.3, which uses the DebugAccessPasswordAuthentication_Auto command as an example to demonstrate how to use the DSSM command with the CCS tool. Other DSSM commands (factory reset or mass erase) use the same approach, including setting four 32-bit passwords in the same location of the .ccxml file.

The following command list is supported by the CCS tool:

- DebugAccessPasswordAuthentication_Auto
- FactoryReset_Auto
- FactoryReset_Manual
- FactoryResetPasswordAuthentication_Auto

- FactoryResetPasswordAuthentication_Manual
- MassErase_Auto
- MassErase_Manual
- MassErasePasswordAuthentication_Auto
- MassErasePasswordAuthentication_Manual

TI recommends using FactoryReset_Auto to unlock the device. The FactoryReset_Auto command can generate the reset signal in the NRST pin and restore the default NONMAIN configurations.

In cases where software issues break the factory reset process (normally caused by software reset), select FactoryReset_Manual to unlock the device by following these steps:

1. Follow the steps show in Section 6.3 to start a project-less debug.
2. Keep the NRST line in the low state (connect to GND).
3. Generate the DSSM command with FactoryReset_Manual (keep the NRST line low).
4. When *Press the reset button* appears in the CCS console window, release the NRST line (disconnect from GND).
5. The factory reset command executes.

If the FactoryReset_Manual unlock device fails, force the MCU into BSL mode with the BSL invoke pin connected to VCC (if BSL mode exists and is enabled), and generate the FactoryReset_Auto command. During the command execution, keep the BSL invoke pin (the default is PA18) connected to VCC to avoid the device run application code.

This approach is not viable if:

- The NONMAIN configurations disable the SWD factory reset command.
- IWDT enables with a device-supporting VBAT feature, requiring the user to manually turn off the VBAT to disable the IWDT.

See *[FAQ] MSPM33C321A: Recover Device by Factory Reset* to generate a factory reset using different tools.

8.3 Debug Error Overview

This section describes common debug errors found when using CCS tool and details possible recovery methods.

8.3.1 No Error Code: DAP Connection Error

Possible root causes:

- SWD connection issues.
- Broken MSPM33 device.
- The CPU enters a fault state.
- A software (POR or BOOTRST) or hardware reset occurs during connection.

Possible recovery methods:

- Check the SWD hardware connection.
- Check that the VCC, NRST, and Vcore (if supported) are within the datasheet specifications.
- Force NRST low, and reconnect the device. If a DAP connection error does not appear, try the methods detailed in Section 8.2.

8.3.2 No Error Code: Connection to the MSPM33 Core Failed

Possible root causes:

- The NONMAIN configurations disable or encrypt the SWD interface.
- The CPU enters a fault state.
- A software (POR or BOOTRST) or hardware reset occurs during connection.

Possible recovery methods:

- Verify that no low pulse is occurring on the NRST line.
- Generate SWD access password authentication by DSSM if the SWD interface is encrypted.

- Follow steps listed in Section 8.2.

8.3.3 Error – 6305: PRSC Module Failed to Write a Routine Register

Refer to Section 8.3.2.

8.3.4 Error – 260: An Attempt to Connect to the XDS110 Failed

Possible root causes:

- The XDS110 is not connected or occupied.
- Invalid XDS110 firmware, invalid XDS110 serial number, and so forth.

Possible recovery methods:

- Repower the XDS110 and check the USB connection.
- Reset the XDS110 firmware.

8.3.5 Error – 261: Invalid Response From the XDS110

Possible root causes:

- The XDS110 debugger enters a fault state.
- The CPU enters a fault state.

Possible recovery methods:

- Repower the XDS110 debugger.
- Follow the instructions in Section 8.2.

8.3.6 Error – 615: Target Fails to Identify a Correctly Formatted SWD Header

Refer to Section 8.3.5.

8.3.7 Error – 1001: Requested Operation is Not Supported on This Device

Possible root causes:

- A software or hardware reset occurs.
- The CPU enters a fault state.

Possible recovery methods:

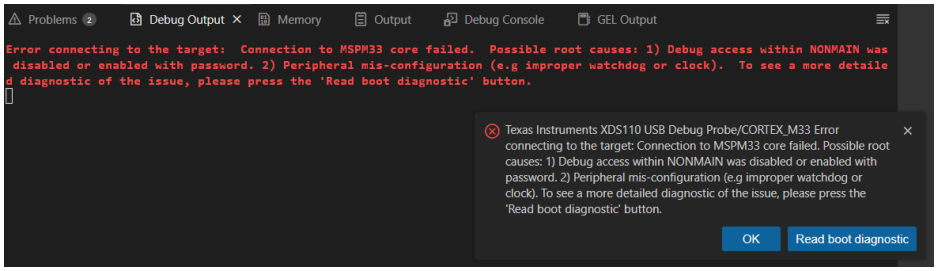
- Check that there is no low pulse occurring on the NRST line.
- If the device enters a locked state, follow the instructions in Section 8.2.

8.3.8 Error – 2131: Unable to Access Device Register

Refer to Section 8.3.7.

8.4 MSPM33 Boot Diagnostic

When the device enters into the locked state and the DAP remains connected, the CCS tool provides the method to read the boot diagnostic of the MSPM33 device, as shown in Figure 8-1.



CCS Version 20.x

Figure 8-1. Read Boot Diagnostic Using the CCS Tool

Table 8-1 lists the common boot diagnostic read values.

Table 8-1. Common Boot Diagnostic Read Values

Device Diagnostic Read	Detailed Description
0x0000.0000	Boot failed. Possible reason is that the NRST line is held low or the clock supply is invalid.
0x0800.0007	Boot success. The application code is executed and leads the device to a lockup state.
0x0800.0016	Boot failed. Possible reason is that the NONMAIN CRC checksum verification fails.
0x0804.004C	Boot failed. Secure boot attempted to authenticate the application image stored in flash, but the authentication failed.
0x0804.110A	Boot success. BSL is invoked and executed.
0x0807.0000	Boot failed. NMI error is triggered in boot code.

9 Summary

This user's guide provides an overview of the NONMAIN configuration among the MSPM33 family and step-by-step instructions to configure the NONMAIN register. In addition to basic knowledge, the document also lists references and further reading materials for users.

10 References

- Texas Instruments, *MSPM33 C3-Series 160MHz Microcontrollers Technical Reference Manual*
- Texas Instruments, *MSPM33 Bootloader User's Guide*
- Texas Instruments, *MSPM33C Security User's Guide*
- Texas Instruments, *Evaluate Customer Secure Code on MSPM33 MCU*
- Texas Instruments, *Enable Trustzone for Secure Application in MSPM33 MCUs*
- Texas Instruments, *QuickStart Guide for Code Composer Studio*
- Texas Instruments, *MSPM33 SDK QuickStart Guide for IAR*
- Texas Instruments, *MSPM33 SDK QuickStart Guide for Keil*
- *Texas Instruments*, *UniFlash Flash Programming Tool*
- *Texas Instruments*, *C-GANG Design and Development Tool*
- *SEGGER Microcontroller GmbH*, *J-Flash Program Internal & External Microcontroller Flash Tool*
- *Emn178.Github.io*, *CRC32 and SHA256 Online Tool*

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), 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 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, TI’s General Quality Guidelines, 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. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

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

Last updated 10/2025