User's Guide

MSPM0 Bootloader

ABSTRACT

The MSPM0 Bootloader (also known as BSL) provides a method to modify the device memory, both the application memory (Flash) and data memory (RAM) through a standard serial interface.

To invoke the Bootloader a specific entry sequence needs to be followed, it does not get executed on every power up. And the Bootloading session can be exited with a reset or a specific command to start the application.

Table of Contents

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

1.1 Overview of BSL Features

The bootstrap loader (BSL) provides a method to program or verify the device memory through a standard serial interfaces like UART, I2C.

Key features of the ROM BSL that are accessible through the serial interface include:

- Programming and erase of flash memory
- Ability to return a 32-bit CRC of a code or data region (1KB minimum region size) to verify programming
- Ability to enable code or data read-out (disabled by default)
- Ability to return a firmware version number via a pointer to the main flash
- Ability to specify a hardware invoke GPIO
- Access is always protected with a 256-bit password
- Configurable security alert handling for resisting brute force attacks
- Ability to add a new interface as flash plug-in
- Ability to use custom bootloader

1.2 Terminology

Bootloader (BSL) - Boot routine used to load data to the device memory

Bootcode (BCR) - Startup Routine that runs after BOOT Reset, to configure the device for executing application

BCR configuration - Configuration structure that contains all user configurable parameters for Bootcode, which resides in Non-main flash memory

BSL configuration - Configuration structure that contains all user configurable parameters for Bootloader, which resides in Non-main flash memory

1.3 Additional resources

1. Technical Reference Manual
    a. MSPM0 G-Series Microcontrollers
    b. MSPM0 L-Series Microcontrollers
    c. MSPM0 H-Series Microcontrollers
    d. MSPM0 C-Series Microcontrollers
2. Tools and examples
    a. MSPM0 Sofware Development Kit
3. App note
    a. MSPM0 Bootloader Implementation

2 BSL Options

The MSPM0 Devices can be classified into three categories based on the BSL features supported by default.

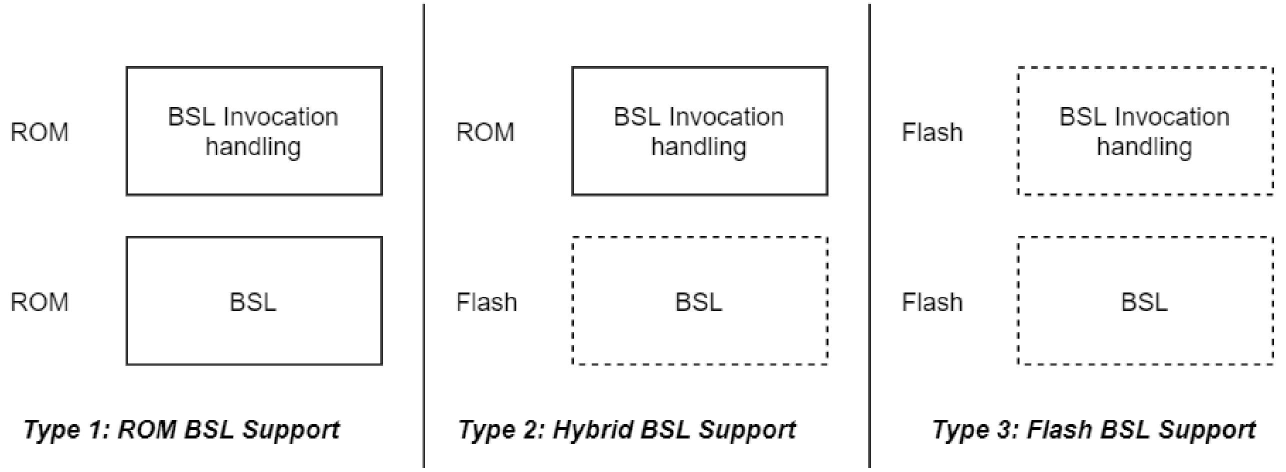

Figure 2-1. BSL options in MSPM0 Devices

Type 1: ROM BSL Support

Devices with ROM BSL implementation have the Bootloader embedded in ROM. It would take care of the invocation and core operations of updating the memory. The next few sections of this userguide contains the details of the features, architecture and protocol of the ROM BSL.

- Supports UART and I2C interfaces
- Available for use immediately in the factory fresh device
- Custom implementations in flash memory, in the form of secondary BSL or Interface plugins are also supported in these devices.

More details on the Secondary BSL or Interface plugin can be found at the later sections.

Type 2: Hybrid BSL Support

In these devices, the reponsibilities are shared between ROM and Flash. ROM Bootcode (BCR) checks for the invocation conditions on every boot, when Bootloader is enabled in the BCR Configuration. This part is handled similar to ROM BSL. When the condition is met the control is transfered to the BSL located in the Flash memory, where the core BSL operations are implemented.

- Can be a custom BSL implementation that supports any interface available in the device
- In the fresh device, the Flash BSL (and optionally application image), need to be loaded through the debugger interface for the first time.

The implementation and configuration process for Flash BSL in this category follows the same procedures as Secondary BSL. Refer to the Section 6 for more details.

Type 3: Flash BSL Support

These devices does not have ROM support for BSL activities. Flash implementation should handle the BSL invocation criteria and the functionalities completely.

- Can be a custom BSL implementation that supports any interface available in the device
- In the fresh device, the Flash BSL (and optionally application image), need to be loaded through the debugger interface for the first time.

Table 2-1. BSL Support in devices

Device Family	Devices	ROM BSL Support	Hybrid BSL Support	Flash BSL Support
MSPM0Gx	MSPM0Gx	✓		

Table 2-1. BSL Support in devices (continued)

Device Family	Devices	ROM BSL Support	Hybrid BSL Support	Flash BSL Support
MSPM0Lx	MSPM0Lx	✓		
MSPM0Cx	MSPM0C1105/6		✓	
	MSPM0C1104			✓
MSPM0Hx	MSPM0H321x		✓	

3 BSL Architecture

3.1 Design

The bootloader will be invoked by bootcode when a valid bootloader invocation condition is detected. It will be invoked only if the bootloader is enabled in the BSL mode field of the BCR configuration.

Once the bootloader is started, it first executes the "Init" phase where the initial checks of the BSL configuration is done and the device is configured for bootloader operations.

Next the bootloader enters "Interface Autodetection" phase. In this phase the BSL configures all the available BSL ROM interfaces and the flash plug-in interface , if registered . The BSL then polls for the data through all the interfaces one by one. When a valid connection packet is received in one of the interfaces, that interface will be considered as active interface for further communication and all other interfaces will be disabled. Interface discovery is done for 10 seconds, if no interface has been detected, the device is put in STANDBY mode.

Next the bootloader enters "Command reception" phase. In this phase BSL will be waiting in an infinite loop for commands from the host. When a valid command is received, the command is processed and the response from the BSL core is sent back to the host. Then it goes back to the loop and waits for the next command and so on. If the 'Start Application' command is received, the bootloader will trigger a system reset, after which the bootcode is executed and the application is invoked. This phase also has a timeout of 10 seconds. If there is no valid command received, bootloader is locked and it enters Sleep mode.

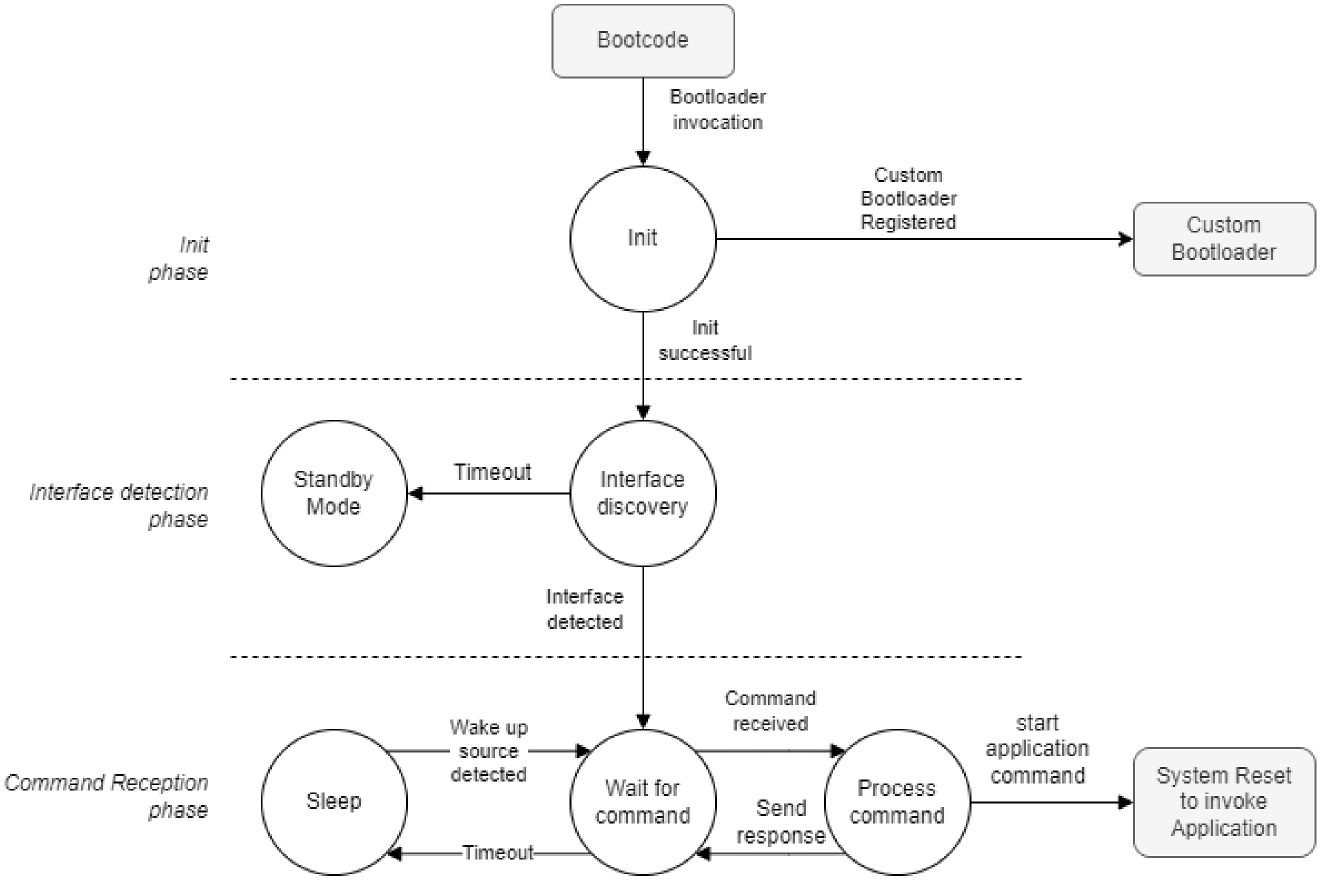

Figure 3-1. BSL Architecture

3.1.1 Timeout Feature

Bootloader will time out when no activity is detected and enters low-power mode to save power.

This has been implemented at the following two phases.

1. Interface autodetection
2. Command reception

3.1.1.1 Interface Autodetection

In Interface detection phase, if there is no valid connection command received for 10 seconds over any interface, the bootloader enters STANDBY mode.

POR is required to get out of this state and use Bootloader by creating the BSL invocation condition again.

3.1.1.2 Command Reception

In command reception phase, if no valid command is received for 10 seconds, the bootloader will enter SLEEP mode. To wake up the device from SLEEP mode, a data transaction should happen on the active interface.

Bootloader will be locked before entering Sleep mode to reduce attack surface. Hence after waking up from the low power mode, Bootloader needs to be unlocked again by sending the 256 bit BSL Password (See Unlock Bootloader command).

Note

Bootloader uses LFCLK for the timeout check. If application configures the external clock as the source for LFCLK, the same will be used by BSL, when invoked through Application BSL request

3.2 BSL Invocation

Bootloader shall be invoked only by the boot code, when any of the BSL invocation conditions is met and if Bootloader is enabled in BCR configuration.

When Fast boot mode is enabled in BCR configuration, bootloader can be invoked only by Debug mailbox command and Application request. Other checks are skipped to save execution time.

3.2.1 Blank Device

Bootcode detects the blank device by checking the erase state of the Stack pointer (0x00000000) and reset vector (0x00000004) address. When both the flash memory addresses are found to be blank, then Bootloader will be invoked.

3.2.2 Application Request

To invoke the bootloader from an application, set the RESETLEVEL as BOOTLOADERENTRY and trigger reset through the RESETCMD register. This sequence causes a system reset, and the bootcode is executed and the bootloader is invoked.

Because a system reset is issued, all the peripheral configurations are reset while exiting the application.

3.2.3 GPIO Based Invocation

GPIO used for BSL invocation can be configured in the BSL Configuration in Non-main memory.

Fresh devices will have the TI programmed default pin detail in BSL configuration.

GPIO pin based invocation can be disabled in the BCR configuration. It is enabled by default.

GPIO should be asserted before the POR, and the state should be maintained for at-least T_start ms after POR. Then the GPIO pin state can be de-asserted.

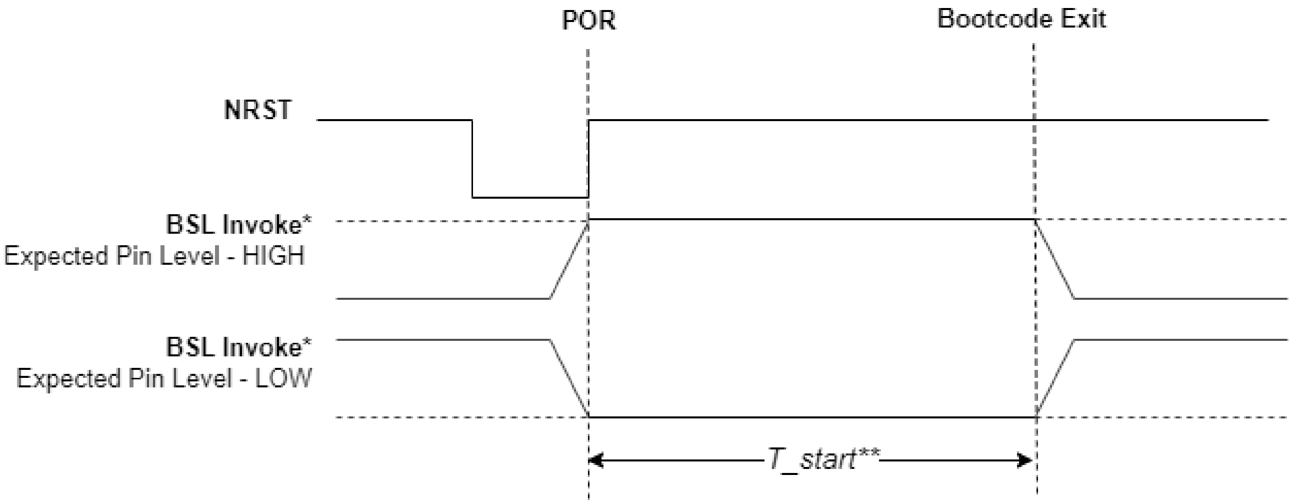

Figure 3-2. Invocation From GPIO

** - GPIO pin to be used as 'BSL Invoke' and 'Expected Pin Level' can be configured in BSL configuration*

*** - T_start refers to the Cold boot startup time, specified in the device specific data sheet*

Note

If pin based BSL invocation is enabled, the configured GPIO pin should be either pulled high or low. It should not be left floating, which could cause unexpected BSL execution.

3.2.4 Debug Mailbox Command

When debug interface is available, Bootloader invocation command can be sent through the Debug Subsystem Mailbox (DSSM).

For more details on the DSSM command usage, please refer to MSPM0x_Technical_Reference_Manual DSSM commands section.

3.2.5 Pre-Boot Application Verification

When the application CRC/Hash verification is configured in the BCR configuration, application memory integrity is verified on every boot. If the verification fails and bootloader is enabled, the bootloader will be invoked.
If Bootloader is disabled, then it will lead to boot failure. Refer to Boot Configuration Section in Technical Reference Manual of the device for more details.

3.3 Memory

3.3.1 SRAM Memory Usage

The SRAM Memory layout explains the memory used for bootloader's operation.

- Data and Stack section - Used by BSL for it's operation. While exiting the bootloader, these sections of the SRAM are cleared.
- Variable Buffer Space - Buffer space used for storing the data packets received /sent during the BSL communication

The SRAM memory allowed for read and write access by the host is BSL Buffer Start Address to [SRAM end address - 0x120], where SRAM end address is determined by the SRAM memory available in each device. Since the same SRAM space is shared with variable buffer space, it has chances of getting overwritten during an SRAM write/read operation.

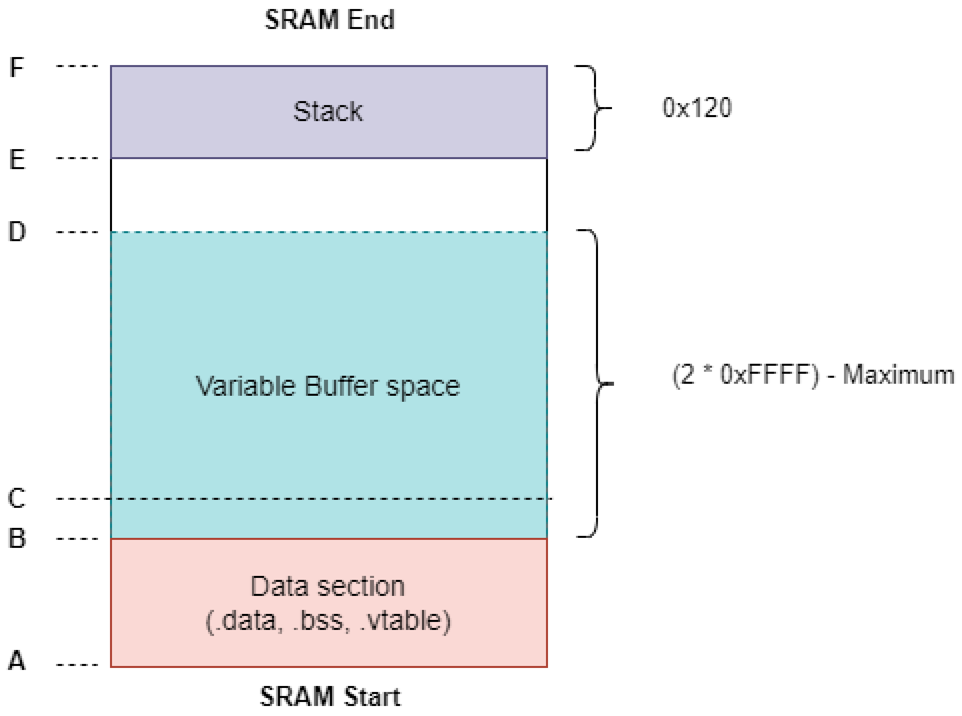

Figure 3-3. SRAM Usage

A - SRAM Start Address (0x20000000)

B- 'BSL Buffer Start Address' known from the 'Get Device Info' command response, when no Flash plug-in interface is registered

C- 'BSL Buffer Start Address' known from the 'Get Device Info' command response. It will be same as 'B' when no Flash plug-in interface is registered

D- BSL Buffer End Address ='BSL Buffer Start Address' + (2 * 'BSL Max Buffer size'), where BSL Buffer Start Address and BSL Max Buffer Size can be known from 'Get Device Info' command response

E- Start address of the stack allocation (E - 0x120). It will be same as 'D' when the 'BSL Max Buffer size' is less than 0xFFFF

F- End address of the SRAM memory available in the device. Refer to device specific data sheet to know this.

Section B- C :

- Data section that will be allocated for Flash Plug-in operation, when registered in BSL configuration

Section C- D:

- Buffer space used to store data packets
- Maximum size is (2 * 0xFFFF)

Section C-E:

- Available memory for SRAM read and write operation through BSL commands

3.4 BSL Configuration

BSL configuration in Non-main Flash memory allows certain parameters used by BSL to be customized by the user.

To know more about the available configurations, refer MSPM0 Technical Reference Manual - Configuration Memory section .

3.5 BSL Status

BSL status gives the details about any error occurred during the execution, that resulted in unresponsive BSL. This 32-bit status information is stored in a specific SRAM address "0x200000C0", and can be read through the debugger, if the debug-access is enabled in the BCR configuration.

Table 3-1. Interpretation of BSL Status

Byte 4	Byte 3	Byte 2	Byte 1
Reserved	HW Error	HW Error details	SW Error

Error	Description
0x01	BSL Configuration CRC Error

HW Error

Error	Description	HW Error Details
0x07	NMI exception	NMIIDX - NMI Interrupt Index register data

4 Bootloader Protocol

4.1 Packet Format

The BSL data packet has a layered structure. The BSL core command contains the actual command data to be processed by the BSL. In addition to the standard BSL commands, there can be wrapper data before and after each core command known as the peripheral interface code (PI Code). This wrapper data is information that is specific to the peripheral and protocol being used, and it contains information that allows for correct transmission of the BSL core command. Taken together, the wrapper and core command constitute a BSL data packet

PI Code	BSL Core Data	PI Code

4.2 UART and I2C BSL Protocol

The data packets of the UART and I2C BSL protocol have the following structure.

- **Header** byte indicates the protocol used and the packet type (command or response packet).
- **Length** field contains the size of the BSL Core Data in bytes.
- **BSL core data**, contains the Command / Response ID and Address, data as needed by the command
- **CRC** field contains the CRC calculated for the data in "BSL core data" to check the integrity. The CRC can be either 32bit CRC or 16bit CRC depending on the device. In either case the field length will be constant. In case of 16bit CRC, remaining bytes are appended with 0's.

PI Code		**BSL Core Data**	**PI Code**
Header (1 byte)	Length (2 byte)	BSL Core Command/Response	CRC (4 byte)

Based on the Core data field, the data packet is classified as either Command packet or response packet.

Command packet is the first packet which is transmitted to the BSL. The second packet is the Response packet which is received from the BSL. Response packet contains two components BSL acknowledgment and BSL Core response. In these two, Acknowledgment is received from BSL for every command packet sent. But the BSL core response is not received for every command.

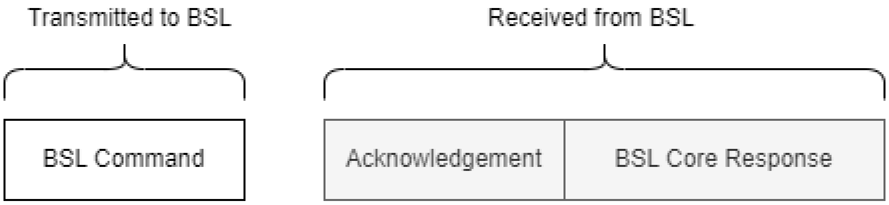

Figure 4-1. BSL Protocol

4.2.1 BSL Acknowledgment

The peripheral interface section of the BSL software parses the wrapper section of the BSL data packet. If there are errors with the data transmission, an error message is sent immediately. An ACK is sent after all data has been successfully received and does not mean that the command has been correctly executed (or even that the command was valid) but, rather, that the data packet was formatted correctly and passed on to the BSL core software for interpretation.

The BSL protocol dictates that every BSL data packet sent is responded to with a single byte acknowledgment in addition to any BSL data packets that are sent. Table lists the acknowledgment responses from the BSL. If an acknowledgment byte other than ACK is sent, the BSL does not send any BSL data packets. The host programmer must check the acknowledgment error and retry transmission.

Data	Meaning
0x00	BSL_ACK (Packet received successfully)
0x51	BSL_ERROR_HEADER_INCORRECT
0x52	BSL_ERROR_CHECKSUM_INCORRECT
0x53	BSL_ERROR_PACKET_SIZE_ZERO

Data	Meaning
0x54	BSL_ERROR_PACKET_SIZE_TOO_BIG
0x55	BSL_ERROR_UNKNOWN_ERROR
0x56	BSL_ERROR_UNKNOWN_BAUD_RATE

4.2.2 Peripheral Configuration

4.2.2.1 UART

UART is enabled with following configuration:

- UART0 is used
- Baud rate 9600 bps by default. It can be updated by Change Baud Rate command
- Data width: 8 bit
- Stop bits: 1
- No parity
- Pins used for RXD and TXD are taken from the BSL configuration

4.2.2.2 I2C

The I2C interface in the BSL can act as the I2C target. The host acts as a controller and drives the communication.

- I2C0 is used
- The I2C target address is 0x48, by default. It can be configured in BSL Configuration
- External pullup is needed for SCL and SDA lines
- Pins used for SDA and SCL are taken from the BSL configuration

Note

- The BSL does not validate the details of the pins configured in the non-main for the UART and I2C interfaces. The BSL expects the pin configuration to be correct.
- Do not use the same pins for both UART and I2C.

4.2.2.3 CRC

CRC is calculated using the hardware CRC module. In devices that support CRC32, 32 bit CRC value is used. In devices that support only CRC16, 16 bit CRC is used and remaining bytes are appended with 0's to constitute 4 bytes. Refer to device specific datasheet to identify the supported CRC polynomial.

CRC for the data is calculated with:

- CRC32-ISO3309 (or) CRC16-CCITT polynomial
- Bit reversed configuration
- Initial seed - 0xFFFFFFFF

4.3 Bootloader Core Commands

BSL Command	Protected	CMD	Start Address	Data (Bytes)	BSL Core response
CMD Connection	No	0x12	-	-	No
CMD Unlock Bootloader	No	0x21	-	D1…D32 (Password)	Yes
CMD Flash Range Erase	Yes	0x23	A1...A4	A1...A4 (End address)	Yes
CMD Mass Erase	Yes	0x15	-	-	Yes
CMD Program Data	Yes	0x20	A1...A4	D1…Dn,	Yes
CMD Program Data Fast	Yes	0x24	A1...A4	D1…Dn,	No
CMD Memory Read back	Yes	0x29	A1...A4	L1...L4	Yes
CMD Factory Reset	Yes	0x30	-	D1...D16 (Password)	Yes

BSL Command	Protected	CMD	Start Address	Data (Bytes)	BSL Core response
CMD Get Device Info	No	0x19	-	-	Yes
CMD Standalone Verification	Yes	0x26	A1...A4	L1...L4	Yes
CMD Start application	No	0x40	-	-	No
CMD Change Baud rate	No	0x52	-	D1 (Baud rate ID)	No

Abbreviations:

A1...A4

Address bytes, where A1 is the least significant byte

D1...Dn

Data bytes, where 'n' is limited by the BSL Max buffer size

L1...L4

Data Length bytes, where C1 is the least significant byte

4.3.1 Connection

Structure

Header	Length		CMD	CRC32			
0x80	0x01	0x00	0x12	C1	C2	C3	C4

Description

Connection command is the first command used to establish the connection between the Host and the Target through a specific interface (UART or I2C).

Protected

No

Command Returns

Only BSL Acknowledgment

Example

Host: 80 01 00 12 3A 61 44 DE

BSL: 00

4.3.2 Get Device Info

Structure

Header	Length		CMD	CRC32			
0x80	0x01	0x00	0x19	C1	C2	C3	C4

Description

The command is used to get the version information and buffer size available for data transaction.

Protected

No

Command Returns

BSL Acknowledgment and BSL core response with device information. See Device info for more details.

Example

Host: 80 01 00 19 B2 B8 96 49

BSL:00 08 19 00 31 00 01 00 01 00 00 00 00 01 00 C0 06 60 01 00 20 01 00 00 00 01 00 00 00 49 61 57 8C

4.3.3 Unlock Bootloader

Structure

Header	Length		CMD	Data	CRC32			
0x80	0x21	0x00	0x21	D1...D32	C1	C2	C3	C4

Description

The command is used to unlock the bootloader. Only after bootloader unlock, all the protected commands listed in Section 4.3 are processed by the BSL.

If the host sends an incorrect password, the BSL sends the error response, then device enters sleep mode for 2 seconds. During this 2-second window, no command is received or processed. After 2 seconds, the host can send the next password. If an incorrect password is sent 3 times, a security alert action is taken. The security alert action is configurable in the BSL configuration. See Section 4.5.1.1 for more details.

Protected

No

Data

32 byte BSL password stored in the BSL configuration memory. Fresh device would have the default value all 0xFF.

Command Returns

BSL Acknowledgment and BSL core response with Message about the Status of operation. See section BSL Core Message for more details.

Example

Host: 80 21 00 21 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 02 AA F0 3D

BSL: 00 08 02 00 3B 00 38 02 94 82

4.3.4 Program Data

Structure

Header	Length		CMD	Address	Data	CRC32			
0x80	L1	L2	0x20	A1...A4	D1...Dn	C1	C2	C3	C4

Description

The program command is used to write the data D1 through Dn in the memory address starting from A1...A4. This command performs the blocking write. Once the programming is done a Message response is sent to Host.

Programming is allowed to Main flash (application memory), Non-main Flash (configuration memory) and SRAM memory. For details on the absolute address range please refer to the device specific data sheet.

Flash memory should be erased by the Host before programming. See Flash range erase, Mass erase for more details on erasing the Main flash region. Non-main flash can only be erased by the Factory reset command.

Due to the Flash controller characteristic, the start address and Length of the data should be 8 byte aligned for flash programming.

Note

SRAM memory is not fully accessible by the Host. See Section 3.3.1 for more details.

Protected

Yes

Address

Start address of the memory region to be programmed. A1...A4, where A1 is the Least Significant Byte of the 32 bit address.

Data

Data bytes to be written in the specified address. Maximum size of the data that can be sent is limited by the Buffer size of the device. Buffer size is known from Get Device Info command .

Command Returns

BSL Acknowledgment and BSL core response with Message about the Status of the operation. See section Section 4.4.1 for more details.

Example

Host: 80 0D 00 20 00 00 00 00 00 00 00 04 00 00 00 08 7A DC AE B8

BSL: 00 08 02 00 3B 00 38 02 94 82

4.3.5 Program Data Fast

Structure

Header	Length		CMD	Address	Data	CRC32			
0x80	L1	L2	0x24	A1...A4	D1...Dn	C1	C2	C3	C4

Description

The Program Data Fast command is identical to the Program Data command , except that this command performs the non-blocking write, to speed up the programming process. To this command BSL will not send the BSL core message response to indicate if the programming was successful.

Protected

Yes

Address

Start address of the memory region to be programmed. A1...A4, where A1 is the least significant byte of the 32 bit address.

Data

Data bytes to be written in the specified address. Maximum size of the data that can be sent is limited by the Buffer size of the device. Buffer size is known from Get Device Info command .

Command Returns

BSL Acknowledgment.

Example

Host: 80 0D 00 24 00 01 00 00 01 02 03 04 05 06 07 08 72 10 2A 18

BSL: 00

4.3.6 Readback Data

Structure

Header	Length		CMD	Address	Data	CRC32			
0x80	0x09	0x00	0x29	A1...A4	L1...L4	C1	C2	C3	C4

Description

This command is used to readout the data starting from address A1...A4.

Readout should be enabled in the BSL configuration, to readout the data using this command. It is disabled by default in the BSL configuration.

Data readout is allowed for Main flash (application memory), Non-main Flash (configuration memory) and SRAM memory.

Note

SRAM memory is not fully accessible by the Host. See SRAM memory usage for more details.

Protected

Yes

Address

Start address of the memory region to be read back. A1...A4, where A1 is the least significant byte of the 32 bit address.

Data

Size of the data to be read in bytes, L1...L4, where L1 is the least significant byte. Maximum size of the data that can be read is limited by the Buffer size of the device. Buffer size is known from Get Device Info command.

Command Returns

BSL Acknowledgment and BSL core response with requested data, if the readback command is valid. See Section 4.4.3 for more details.

If the readback command, had invalid address / length or if the readout was disabled, the corresponding error will be sent as the Message Response following the BSL acknowledgment.

Example

Host: 80 09 00 29 00 0C 00 00 08 00 00 00 32 9D B0 35

BSL: 00 08 09 00 30 FF FF FF FF FF FF FF FF F6 2B A1 73

4.3.7 Flash Range Erase

Structure

Header	Length		CMD	Address	Data	CRC32			
0x80	0x09	0x00	0x23	A1...A4(Start Address)	A1...A4(End Address)	C1	C2	C3	C4

Description

The Flash range erase command is used to erase the specified flash memory region. Flash is erased sector wise (1 KB), erasing lesser size is not possible.

When the Start and End address reside in different flash sectors, the BSL erases all the flash sectors in between the Start and End address, including sectors that contains these addresses.

This command can be used to erase only the Main Flash memory. Non-main erase is not possible.

End address should never be less than Start address.

Protected

Yes

Address

Start address of the memory region to be erased. A1...A4, where A1 is the Least Significant Byte of the 32 bit address.

Data

End address of the memory region to be erased. A1...A4, where A1 is the Least Significant Byte of the 32 bit address.

Command Returns

BSL Acknowledgment and BSL core response with Message about the Status of the operation. See section Section 4.4.1 for more details.

Example

Host: 80 09 00 23 00 01 00 00 FF 03 00 00 2B E6 BE D8

BSL: 00 08 02 00 3B 00 38 02 94 82

4.3.8 Mass Erase

Structure

Header	Length		CMD	CRC32			
0x80	0x01	0x00	0x15	C1	C2	C3	C4

Description

The mass erase command erases the complete Main Flash memory (application memory) available in the device.

The Mass erase configuration in BCR configuration memory doesn't affect this BSL command.

When a flash region is static write protected in BCR configuration memory, the region can't be erased.

Protected

Yes

Command Returns

BSL Acknowledgment and BSL core response with Message about the Status of the operation. See Section 4.4.1 for more details.

Example

Host: 80 01 00 15 99 F4 20 40

BSL: 00 08 02 00 3B 00 38 02 94 82

4.3.9 Factory Reset

Structure

Header	Length		CMD	Data	CRC32			
0x80	L1	L2	0x30	D1...D16	C1	C2	C3	C4

Description

The factory reset command erases the complete Main Flash (application) memory and Non-main flash (configuration) memory.

Processing this command, is impacted by the factory reset configuration in the BCR configuration memory.

Factory reset is

- Allowed without password, if 'Enabled'
- Allowed with password, if 'Enabled with Password'
- Not allowed, if 'Disabled'

When a flash region is static write protected in BCR configuration memory, the region can't be erased .

Protected

Yes

Data

16 byte factory reset password stored in the BCR configuration memory. The default password is all 0xFF. Password is required only when Factory reset is "Enabled with Password" in BCR configuration.

Command Returns

BSL Acknowledgment and BSL core response with Message about the Status of the operation. See Section 4.4.1 for more details.

> **CAUTION**
>
> After doing the factory reset, until the Non-main configuration has been restored, the system is highly vulnerable to a potential lockout situation in which it is impossible to get access to the device again.

Example

Scenario 1: Factory Reset without Password

Host: 80 01 00 30 DE 20 24 0B

BSL: 00 08 02 00 3B 00 38 02 94 82

Scenario 2: Factory Reset with Password

Host : 80 11 00 30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 8A 28 EA DC

BSL : 00 08 02 00 3b 00 38 02 94 82

4.3.10 Standalone Verification

Structure

Header	Length		CMD	Address	Data	CRC32			
0x80	0x09	0x00	0x26	A1...A4	D1...D4	C1	C2	C3	C4

Description

This command is used to verify the CRC of the data stored at the given memory range. This enables faster verification of the programmed data. It requires the data size to be minimum 1kB.

CRC verification is allowed for Main flash (application memory), Non-main Flash (configuration memory) and SRAM memory.

> **Note**
>
> SRAM memory is not fully accessible by the Host. Refer Section 3.3.1 for more details.

Protected

Yes

Address

Start address of the memory region to be verified. A1...A4, where A1 is the least significant byte of the 32 bit address.

Data

Size of the data to be verified in bytes, L1...L4, where L1 is the least significant byte. 1kB <= size <= 64 KB.

Command Returns

BSL Acknowledgment and BSL core response with CRC value calculated for the requested memory region. Refer Section 4.4.5for more details.

If the verification command, had invalid address / length, the corresponding error will be sent as the Message Response following the BSL acknowledgment. Refer Section 4.4.1.

Example

Host: 80 09 00 26 00 00 00 00 00 0C 00 00 1C E9 07 E1

BSL: 00 08 05 00 32 A0 45 71 82 91 1F 94 EC

4.3.11 Start Application

Structure

Header	Length		CMD	CRC32			
0x80	0x01	0x00	0x40	C1	C2	C3	C4

Description

Start application command, issues a system reset, which causes Bootloader exit, rerun of Bootcode, which in turn starts the application.

Protected

No

Command Returns

BSL Acknowledgment

Example

Host:80 01 00 40 E2 51 21 5B

BSL:00

4.3.12 Change Baud Rate

Structure

Header	Length		CMD	Data	CRC32			
0x80	0x02	0x00	0x52	D1	C1	C2	C3	C4

Description

The command can be used to change the baud rate of the UART interface. The new baud rate will take effect after sending out the BSL acknowledgment for this packet.

The default baud rate of the BSL UART is 9600 bps.

Note

After updating the baud rate, if a wrong BSL password is sent with the Unlock Bootloader command, the baud rate settings will be reverted to default value. Further communication should happen in the default baud rate.

Protected

No

Data

D1 Baud rate as specified in the table.

ID	Baud Rate (bps)
1	4800
2	9600
3	19200
4	38400
5	57600
6	115200
7	1000000
8	2000000
9	3000000

Command Returns

BSL Acknowledgment

Example

Host: 80 02 00 52 03 6C 83 A2 AF

BSL: 00

4.4 Bootloader Core Response

BSL Response	RSP	Data
Memory read back	0x30	D1…Dn
Get Device Info	0x31	D1…D24
Standalone verification	0x32	D1…D4
Message	0x3B	MSG
Detailed Error	0x3A	D1..D3

Abbreviations:

MSG

A byte containing a response from the BSL core describing the result of the requested action. This can either be an error code or an acknowledgment of a successful operation. In cases where the BSL is required to respond with data (for example, memory, version, CRC, or buffer size), no successful operation reply occurs, and the BSL core immediately sends the data.

D1..Dn

Data bytes, where 'n' is limited by the BSL maximum buffer size

4.4.1 Bootloader Core Message

Structure

Header	Length		RSP	Data	CRC32			
0x08	0x02	0x00	0x3B	MSG	C1	C2	C3	C4

Description

For some commands, BSL sends the message response to the host, which indicates the status of the processed command. The table lists all the possible messages from the BSL.

MSG	Meaning	Possible Cause[1]
0x00	Operation Successful	
0x01	BSL Locked error	The BSL is not yet unlocked with Bootloader unlock password command or After BSL unlock, a timeout would have occurred in the command reception phase
0x02	BSL password error	Incorrect password has been sent to unlock bootloader.
0x03	Multiple BSL password error. Security Alert action is taken.	Incorrect password has been sent to unlock bootloader 3 times.
0x04	Unknown Command	The command given to the BSL was not recognized as valid command
0x05	Invalid memory range	The given memory range is invalid.
0x06	Invalid command	The command given to the BSL is a known command but it is invalid at that time instant and cannot be processed.
0x07	Factory reset disabled	Factory reset is disabled in the BCR configuration
0x08	Factory reset password error	Incorrect or no password has been sent with factory reset command, when the BCR configuration has factory reset 'Enabled with password'
0x09	Read out error	Memory read out be disabled in BCR configuration
0x0A	Invalid address or length alignment	Start address or data length for the flash programming is not 8-byte aligned
0x0B	Invalid length for standalone verification	Data size sent for standalone verification is less than 1KB

[1] Possible causes listed here, are not the only reason for the status or error. It only lists the probable software reasons for the error, that can be corrected by the host.

4.4.2 Detailed Error

Structure

Header	Length		RSP	Data	CRC32			
0x08	0x02	0x00	0x3A	D1..D3	C1	C2	C3	C4

Description

D1 - Error Type

D3,D2 - Error Details

Possible Values

Error type		Error Details	
Value	Description	Value	Description
0xF0	Flash Error	0xXX	Contains FLASHCTL.STATCMD register value

4.4.3 Memory Readback

Structure

Header	Length		RSP	Data	CRC32			
0x08	L1	L2	0x30	D1...Dn	C1	C2	C3	C4

Description

The command returns the requested data in response to the Readback command

Data

Data D1..Dn where 'n' is limited by the BSL maximum buffer size.

4.4.4 Device Info

Structure

Header	Length		RSP	Data	CRC32			
0x08	0x19	0x00	0x31	D1...D24	C1	C2	C3	C4

Description

The command returns the version information and BSL buffer size in response to the Get Identity Command

Data

Identity Byte	Data Byte
Command Interpreter version	[D02-D01]
Build ID	[D04- D03]
Application version	[D08-D05]
Active Plug-in interface version	[D10-D09]
BSL Max buffer size	[D12-D11]
BSL Buffer Start address	[D16-D13]
BCR Configuration ID	[D20-D17]
BSL Configuration ID	[D24- D21]

Application version:

32 bit application version is taken from the address specified in BSL configuration

BSL Buffer size:

RAM data buffer size available to store the BSL data packets that are sent/received.

Example

Host: 80 01 00 19 B2 B8 96 49

BSL: 00 08 19 00 31 00 01 00 01 00 00 00 00 01 00 C0 06 60 01 00 20 01 00 00 00 01 00 00 00 49 61 57 8C

In the above given response,

Command Interpreter version - 0x0100

Build ID - 0x0100

Application version - 0x00000000

Active Plug-in interface version - 0x0001

BSL Max buffer size - 0x06C0

BSL Buffer Start address - 0x20000160

BCR Config ID - 0x00000001

BSL Config ID - 0x00000001

4.4.5 Standalone Verification

Structure

Header	Length		RSP	Data	CRC32			
0x08	0x05	0x00	0x32	D1...D4	C1	C2	C3	C4

Description

The command returns the CRC value in response to the standalone verification command

Data

32 bit CRC value calculated for the requested memory region. D1...D4, where D1 is the least significant byte of the CRC32.

4.5 Bootloader Security

4.5.1 Password Protected Commands

All the commands that can access the data in the memory directly or indirectly, are password protected. The password is configurable in the BSL configuration in Non-main memory.

When an incorrect password is sent, device will sleep for next 2 seconds and does not accept any command during this period to make brute force attacks harder. When incorrect password is sent for 3 times, Security alert action is taken by the BSL.

4.5.1.1 Security Alert

A security alert can be configured to any of the following three modes, in the BSL configuration.

1. **Factory Reset** - Factory reset erases the complete main and non-main flash memories. The erase is done irrespective of the factory reset mode in the BCR configuration. If certain parts of the flash are static write protected, the BSL cannot erase the complete flash memory. In this case once non-main is erased, the bootloader password returns to the default. BSL can then be unlocked and programmed with the new password .
2. **Disable Bootloader** - Disables the bootloader in the BCR configuration and exits from BSL. No further entry to BSL is possible, unless the BCR non-main configuration is updated to enable the bootloader. The BSL is not disabled if static write protection is enabled for non-main.

   Also this feature is supported in devices that have more than 2KB SRAM memory.
3. **Do Nothing** - Does not take any action.

Note

In case of 'Factory reset' security alert configuration, if non-main is left in the erased state, the device is locked and is not possible to gain access to the device again.

4.5.2 BSL Entry

Entry to Bootloader is allowed only through the Bootcode.

Bootloader can be disabled in BCR configuration.

5 Sample Program Flow with Bootloader

This section explains the typical sequence followed by the BSL host for loading the image through the Bootloader. This sample sequence erases the flash memory and programs new firmware in it.

- Bootloader shall be started through Blank device detection, Pin based invocation or Application request.
- Once it is invoked, send the connection command to establish connection with the BSL through the desired interface.
- If UART interface is used, the baud rate can be changed to a higher value, to speed up the further communication and is optional.
- To erase the flash memory completely use the Mass erase command. Only when there is a need to update Non-Main flash use factory reset command. Because if the Non-main flash is erased and left unprogrammed the device is locked.
- Program the firmware image
- Do the CRC verification of the programmed memory region to check the correctness of data programmed. This is an optional step.
- Application can be started with 'Start application' command.

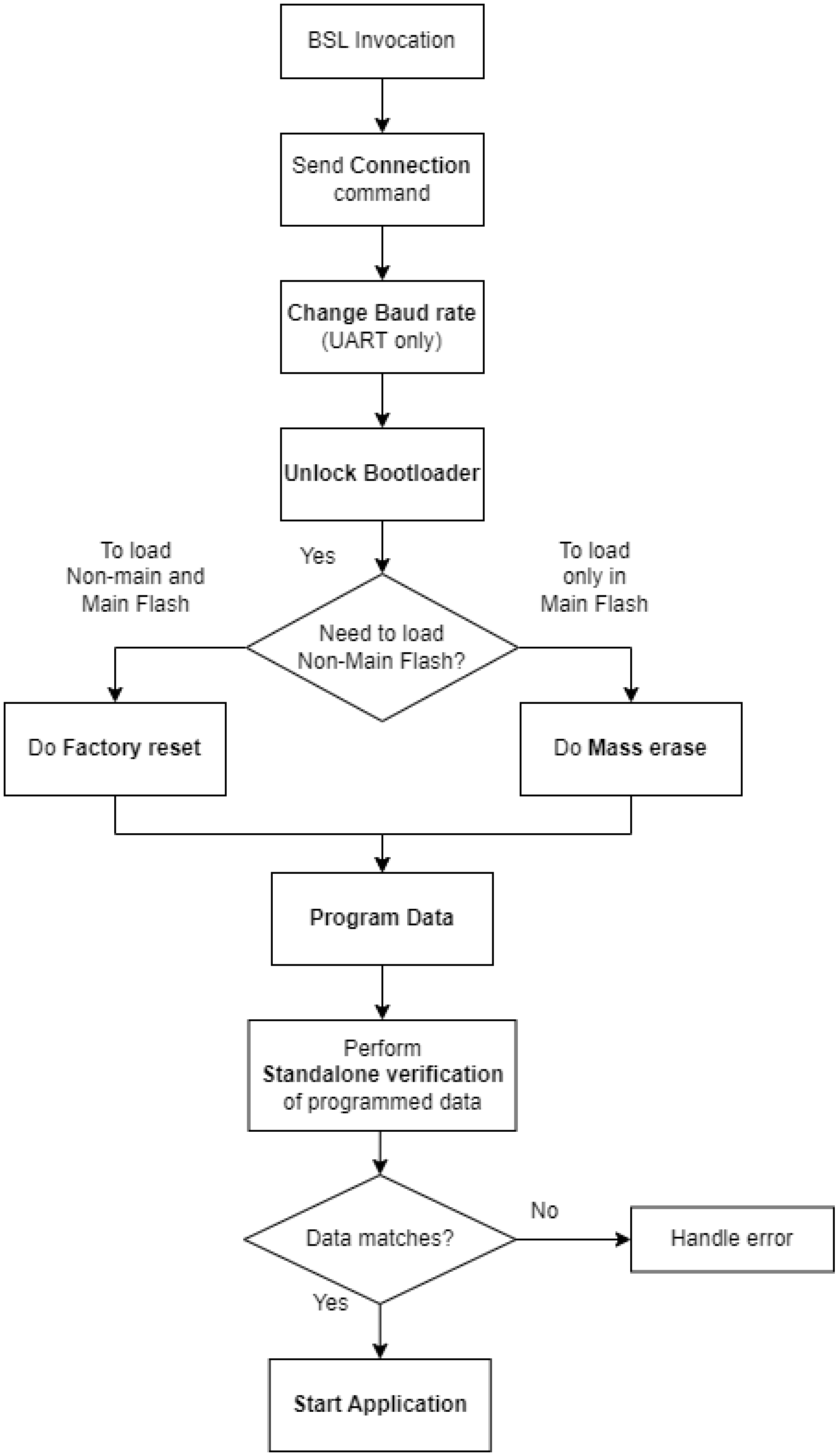
BSL Invocation
Send Connection command
Change Baud rate (UART only)
Unlock Bootloader
Yes
To load Non-main and Main Flash
To load only in Main Flash
Need to load Non-Main Flash?
Do Factory reset
Do Mass erase
Program Data
Perform Standalone verification of programmed data
Data matches?
No
Handle error
Yes
Start Application

Figure 5-1. BSL Host Sequence

6 Flash based BSL

Flash Bootloader is a user implemented bootloader routine placed in the flash memory. Thus allowing for customisation and update. The implementation of Flash BSL varies based on the BSL Type of the device.

- Type 1 devices with ROM BSL support provide an option to implement a custom Bootloader in Flash memory, referred to as a Secondary / Alternate Bootloader. In this type of devices, ROM BCR will take care of the invocation, the Flash implementation is expected to handle BSL operations (such as program and memory readback), through any of the serial interfaces available in the device.
- Type 2 devices lacking ROM BSL support, uses the same implementation as Secondary BSL.

Both Type 1 & Type 2 Flash BSL implementations follow identical invocation flows and configuration procedures. Refer to Section 6.1 for Type 1 & 2 devices.

- Type 3 devices, would require BSL invocation handling in addition to BSL operations.

Refer to Section 6.2 for Type 3 devices.

Note

If Flash BSL memory is not write protected, the bootloading process could erase the Flash BSL code, potentially causing device lockup. Recovery may or may not be possible depending on the debug configurations.

6.1 Secondary BSL / ROM Invoked Flash BSL

When the ROM BCR detects the BSL invocation condition, it will invoke this Flash-based BSL, which is expected to handle the BSL operations through the serial interfaces.

To achieve this, load the BSL in the Main flash memory in any address other than 0x0000. As application is expected to be placed at the start of the Flash memory. And then register it in the BSL configuration in Non-main Flash. When BSL is invoked next time through one of the invocation methods, device will check for the Flash BSL configuration field and branches to it, and the control is not expected to return back to ROM.

BSL mode configuration in the BCR configuration is applicable for Flash BSL as well. When this setting is disabled, BSL in flash will not be invoked.

The flash memory region where the BSL is loaded should be write protected in the BCR configuration, to avoid unintended erase during the bootloading process. Nonmain write protection is optional. But care should be taken to ensure the Flash BSL pointer is retained in the BSL configuration in Nonmain memory.

6.1.1 Secondary BSL Example

An example of secondary Bootloader for Type 1 devices is given as part of SDK examples for reference. This section gives more details on that.

Description

This Sample Secondary Bootloader supports programming / verifying data in the memory, with same BSL protocol format as Primary BSL (ROM BSL) in the device and can be invoked in the same way as the ROM BSL.

It supports the following major functions

- Program data
- Flash memory erase
- Readback data
- CRC verification
- Start Application

It uses UART interface to communicate with the Host.

This example takes care of secondary bootloader implementation as well as the registration of it. Hence once this image is loaded to the device, the primary bootloader in the device can't be used. Only the secondary

Bootloader will be active. To revert the device to use primary Bootloader, SWD_Factory_Reset command has to be used (Factory reset through Debug Subsystem Mailbox). .

Example Usage

- Connect UART_RX and UART_TX with the BSL Host (any microcontroller with UART).
- Compile and load the example.
- Create BSL invocation condition using BSL Invoke pin or any other invocation methods.
- Send the **GetDeviceInfo** command from the host.
- Device should respond back with the version information and SRAM buffer space available.
- Similarly Send erase, program, verification commands to program data in the memory.

Software File Details

File Name	Details
secondary_bsl.c	Initializes the peripherals needed for BSL operation. Receives command packets from Communication interface and passes it to Command processing layer. Also takes care of registering the secondary bootloader in the BSL configuration memory.
bsl_ci.c	Interprets the command packets, process the command and sends back the response to Host
bsl_ci.h	Contains the definitions of BSL commands and Responses. Also the function declarations of bsl_ci.c
bsl_uart.c	Handles the communication between the Host and the BSL core
bsl_uart.h	Contains definitions of BSL Acknowledgment and function declarations of bsl_uart.c
ti_msp_dl_config.h	Contains device specific configurations like UART pins, base addresses of the peripherals used, etc.
boot_config.h	Contains BCR and BSL configuration structure
factory_config.c	Implements functions to fetch factory configured device specific data like SRAM memory size.
factory_config.h	Contains Factory configuration structure and function declaration of factory_config.c
startup_mspm0x_ticlang	Startup file that contains vector table, reset handler and other handlers
mspm0x.cmd	Linker command file that specifies memory region where the Secondary Bootloader image should reside in the memory and the SRAM region where it should operate.

Customization

This example gives a reference implementation for secondary bootloader. It can be customized as needed. BSL Core layer (secondary_bsl.c, bsl_ci.c) or interface layer (bsl_uart.c) are the major places where customization will be done.

Steps to be followed

- Modify the code as needed
- Once the changes are done, compile the code
- Modify the flash write protection settings in BCR configuration as appropriate
- Calculate the CRC for BCR configuration and store the new CRC value
- Compile the code again
- Load the customized BSL image

6.2 Standalone Flash BSL

This Flash BSL implementation is expected to take care of the invocation mechanisms along with the BSL operations. When device comes out of reset, it will always execute the code placed in the start of the flash memory.

To add the Flash BSL support, Invocation and BSL implementation shall be loaded into the flash. And the flash memory region where the BSL is loaded should be write protected in the BCR configuration, to avoid unintended erase during the bootloading process.

6.2.1 Standalone Flash BSL Example

MSPM0 SDK contains an example for Flash BSL implementation for Type 3 devices. This section explains that in detail.

Description

Flash BSL example supports programming / verifying data in the memory. One of UART/I2C interface can be used to communicate with the Host.

It can be invoked in any of the three way as follows:

- No Application at APP START
- BSL PIN Inovation
- Application based invocation

It supports the following functions

- Program data
- Get Device info
- Flash memory erase
- Readback data
- CRC verification
- Start Application

Flash BSL region is static write protected using bootcode user configuration. Once this image is loaded to the device, the Flash Bootloader will be active and will be looking for invocation condition on every Reset. When no BSL request is found it branches to application. Later if the Flash BSL needs an update or it needs to be removed, then SWD_Factory_Reset command has to be used to erase the Flash BSL from the flash memory.

Users can configure the interface pins and other settings. They can also selectively enable only required features to optimize memory footprint.

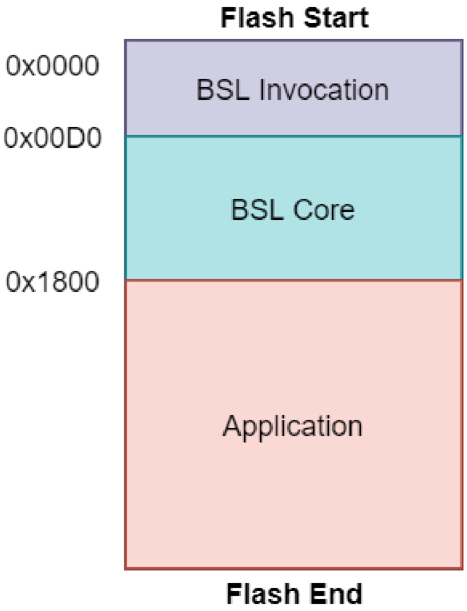

Figure 6-1. Standalone Flash BSL Memory Layout

In the Flash memory, Invocation logic is placed at the start address, 0x0000. Then the BSL is placed. These 2 combined would occupy the first 6 KB of flash, by default. And the application is expected to start at 0x1800. These addresses can vary based on the features added or removed in BSL.

Example Usage

- Connect UART_RX and UART_TX or I2C_SDA and I2C SCL with the BSL Host

- Enable only the required BSL functionalities in **flash_bsl_modules.h**
- Compile, load the example.
- Create BSL invocation condition using BSL Invoke pin or any other invocation methods.
- Send GetDeviceInfo command from the host. Device should respond back with the version information and SRAM buffer space available.
- Similarly Send erase, program, verification commands to program data in the memory.

Software File Details

File Name	Details
flashBSL.c	Initializes the peripherals needed for BSL operation. Receives command packets from Communication interface and passes it to Command processing layer. This gets executed once BSL invocation is requested.
flashBSL_defines.h	Contains configurations and other definitions for flashBSL.c
flashBSL_modules.h	Contains the different features of Flash BSL that can be configured as required
flashBSL_invocation .c	Checks different invocation conditions and branches to application or BSL based on the result
flashBSL_invocation .h	Contains configurations and macro defintions related to invocation
flashBSL_uart .c	Handles the communication between the Host and the BSL core
flashBSL_uart .h	Contains definitions of BSL Acknowledgment
flashBSL_i2c .c	Handles the communication between the Host and the BSL core
flashBSL_i2c .h	Contains definitions of BSL Acknowledgment
flashBSL_ci.c	Interprets the command packets, process the command and sends back the response to Host
flashBSL_ci.h	Contains the definitions of BSL commands and Responses. Also the function declarations of flashBSL_ci.c
boot_config .c	Contains the Boot configuration details. It also additionally contains the BSL configuration section that can be customised based on the Flash BSL implementation. Note: It is not derived from SYSCFG tool.
boot_config .h	Contains the definitions for boot_config .c
startup_mspm0c1104_ticlang.c	Startup file for the BSL. It handles the BSL and Invocation as two independent routines.
mspm0c1104.cmd	Handles the memory layout mentioned above.

7 Interface Plug-in

ROM Bootloader provides an option to add custom interface implementation to the ROM BSL core as a Flash Plug-in. This gives an advantage of customizing the interface, without reimplementing the complete BSL core.

With this feature either:

- A new interface, which is not available in ROM BSL can be added to the Interface list for Auto detection. Ex: SPI, CAN, etc. (Or)
- ROM interface implementation (UART / I2C) can be overridden

To use this option, the Flash plug-in image should be loaded to Main flash and registered in BSL configuration in Non-main Flash memory. Refer to MSPM0xx Technical reference manual - Configuration memory (Non-main) section.

Also the flash memory region where plug-in is loaded and Non-main configuration memory should be write protected in the BCR configuration, to prevent flash plug-in and it's registration in the non-main getting erased during bootloading process.

Note

Erasing the Flash plug-in region in Main flash, could cause device lock up. Hence write protection is needed.

7.1 Implementation

The Plug-in has to take care of data handling and communication with the BSL Host. This Interface Flash plug-in is coupled with ROM BSL core through the following 4 APIs.

- Init
- Receive
- Transmit
- Deinit

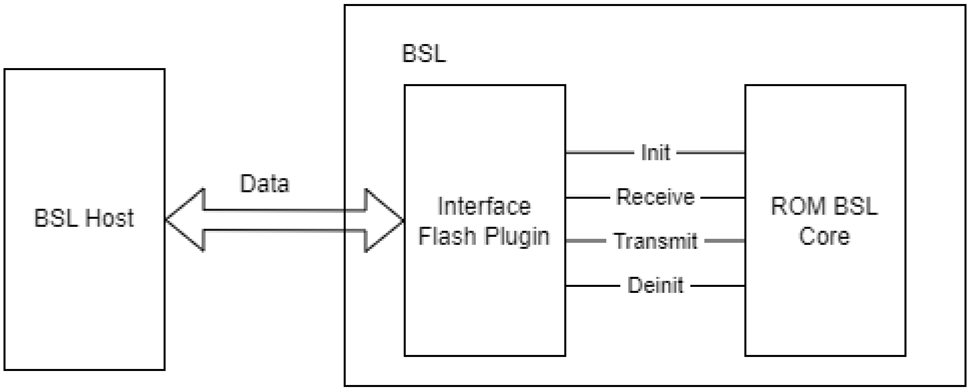

Figure 7-1. Plug-in Implementation

Flash plug-in image shall be built like any other application and will be loaded to main flash. But unlike application, the startup code or main function will not be executed. Only the above mentioned 4 APIs will be called by the ROM BSL through the hooks registered in the BSL configuration in Non-main memory.

7.1.1 Init

Prototype

uint16_t init(uint8_t* buffer, uint16_t bufferSize);

buffer - Pointer to the SRAM data buffer sent from the BSL core.

bufferSize - 1/2 of the SRAM memory size available to use as data buffer. If two buffers are used 1 to transmit and 1 to receive, the address of one buffer will be **'buffer'** and the address of the other buffer will be **'buffer + bufferSize'**

return - Returns the 16 bit Plug-in version information

Description

Init function should take care of configuring the desired interface and initializing the parameters for data handling. It should also take care of initializing any other global variables used, since no startup code will be ran to initialize them.

7.1.2 Receive

Prototype

uint32_t receive(void);

return - Returns the 32 bit start address of the Received data packet. The data packet should have the same format as described in the ROM BSL protocol Section 4.

Description

Receive function should take care of reading the data packets from the BSL host. It should share the packet with BSL core only when the complete packet is received and correctness of the data is checked (CRC verification of data). Also the address should be shared only once per data packet. If this function is called by BSL core, when there is no data packet received or when it is in progress, it should return '0'.

If a data packet is received successfully without any issues, it shall acknowledge to the host, as ROM BSL plug-in does (Refer ROM BSL acknowledgments). In case of any issues, it shall be reported to host through a NACK and the packet should not be shared with ROM BSL core.

7.1.3 Transmit

Prototype

uint8_t send(uint8_t* data, uint16_t length);

data - Pointer to the BSL core response packet to be sent to the host. It will have the same format as described in section Section 4

length - Length of the BSL core response packet excluding the length of the CRC 4 bytes.

return - Returns '1' if the transmission is successful. '0' if the transmission failed.

Description

Transmit has to send the BSL core response packet to the host. It should also take care of calculating and adding CRC for the packet. Transmission API should return only after completely transmitting the data.

7.1.4 Deinit

Prototype

bool deinit(void);

Return

Returns 'True' if deinit is done.

Description

This function resets the interface configuration and unregisters interrupt handlers if registered.

7.1.5 Important Notes

Important points to keep in mind, while developing Flash Plug-in.

1. Main Flash memory region, where Flash plug-in is loaded should be static write protected
2. All the Global variables should be initialized by the 'Init' function

3. Function prototypes for the 4 plug-in APIs should be as specified in the BSL userguide.
4. SRAM memory usage
    a. VTOR - Start of SRAM (0x20000000). If interrupts are used, VTOR should be placed at the start of the SRAM, as the ROM BSL uses that address space
    b. Stack start address - End of SRAM memory available in the device
    c. Stack size - Should not exceed the ROM BSL stack size
    d. Data section (.data, .bss) - 'BSL Buffer Start address' returned by the Get Device Info command, when no Flash plug-in is registered in the device, should be the start address of the Data section.
    e. Data section size - Size consumed by the Data section (.data, .bss) should be configured in the BSL Non-main configuration memory.

7.2 Flash Plug-in Type

The type of communication interface used in the Flash plug-in is configured in the BSLPLUGINCFG.PLUGINTYPE field in BSL configuration memory (see the technical reference manual).

If the plug-in type has any of the ROM interface types (UART or I2C), the flash plug-in overrides the ROM implementation. If the type has a new type that is not available in ROM (for example, SPI), the plug-in is added as a new interface.

Table 7-1. Plug-in Type Values

No.	Interface	Plug-in Type
1	UART	0x1000
2	I2C	0x2000
3	Any other interface	0xXXXX

As shown in Figure 7-2, after checking the flash plug-in type, the updated interface list is used for auto detection. Even after flash plug-in registration the ROM interfaces can be used if those interfaces are not overridden.

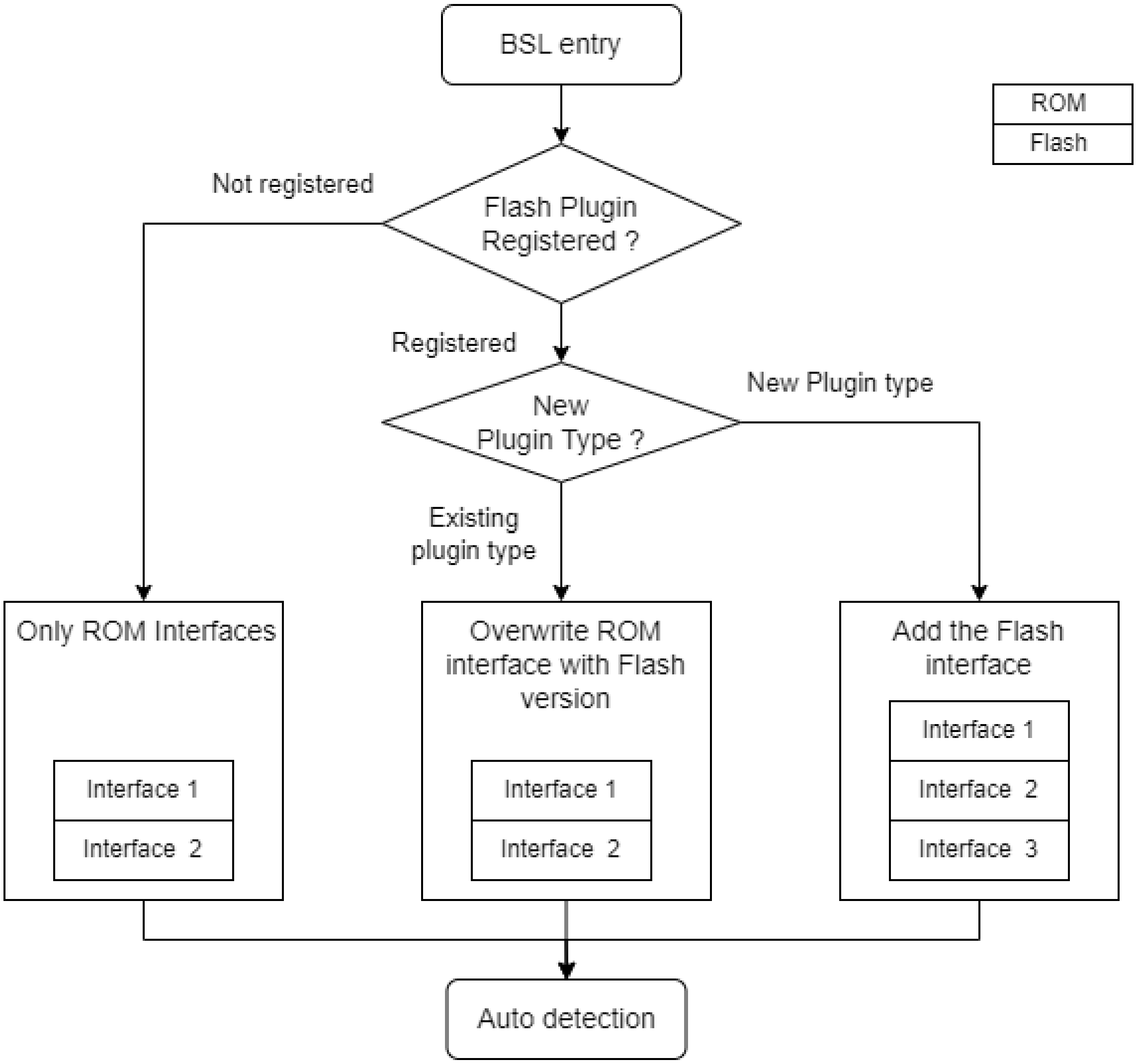

Figure 7-2. Flash Plug-in

7.3 Overriding an Existing Interface

This feature allows user to override existing ROM BSL interfaces, UART & I2C with the Flash version. In this case once flash plug-in is loaded, ROM version will become inactive and can never be used. To revert the device to use ROM interface use SWD_Factory_Reset (Factory reset through Debug Subsystem Mailbox).

7.3.1 UART Interface Flash Plug-in Example

A sample flash plug-in that uses UART communication, is given as part of SDK examples for reference. This section gives more details on that.

Description

UART interface flash plug-in handles the data transaction between the BSL host and the ROM BSL through the following 4 API Hooks.

- BSL_PI_UART_init
- BSL_PI_UART_receive
- BSL_PI_UART_send
- BSL_PI_UART_deinit

The UART flash plug-in is primarily used to override the ROM BSL UART interface with custom implementation when there is a need.

Example Usage

- Connect UART_RX and UART_TX with the BSL Host (any microcontroller with UART interface).
- Compile and load the example.
- Create BSL invocation condition using BSL Invoke pin or any other invocation methods.
- Send **Connection** command from the host. A BSL acknowledgment is received on success.
- Send **GetDeviceInfo** command from the host.
- BSL responds with the UART interface flash plug-in version information.
- Similarly send erase, program, verification commands to program data in the memory.

Software File Details

File Name	Details
bsl_uart.c	Handles the communication between the host and the BSL core. Defines the four interface APIs Init, Receive, Send and Deinit.
bsl_uart.h	Contains definitions of BSL acknowledgment and function declarations of bsl_uart.c
ti_msp_dl_config.h	Contains device specific configurations like UART pins, clock configurations, and others.
boot_config.h	Contains BCR and BSL configuration structure
startup_mspm0x_ticlang	Startup file that contains only the default handler function definition. Unlike a typical startup file does not have interrupt vector table or reset handler, because these features are never used in the flash plug-in and are removed to reduce the memory consumption.
mspm0x.cmd	Linker command file that specifies memory region where the flash plug-in image resides in the memory and the SRAM region for operation.

Customization

This example gives a reference implementation for a flash plug-in. It can be customized as needed. The interface flash plug-in APIs are the major places of change.

Steps to be followed:

- Modify the flash plug-in APIs as needed
- When the changes are done, compile the code
- Calculate the CRC for the BSL configuration if the API pointers are updated
- Modify the flash write protection settings in BCR configuration as appropriate
- Calculate the CRC for BCR configuration
- Store the new CRC in the BCR and BSL configuration.
- Compile the code again.
- Load the flash plug-in image

8 Revision History

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

Changes from February 28, 2023 to January 31, 2026 (from Revision * (February 2023) to Revision A (January 2026)) **Page**

- Added different BSL Types across MSPM0 devices... 25

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