Application Note

MSPM0 Bootloader (BSL) Host Implementation

ABSTRACT

This application note provides two kinds of demos: one demo uses an MCU (an MSPM0) as the host to update the firmware in MSPM0 target device through universal asynchronous receiver/transmitter (UART) or inter-integrated circuit (I2C) or serial peripheral interface (SPI) (co-work with SPI plugin demo), while the other demo uses a PC as the host that communicates with the backchannel UART of a XDS110 programmer. It can work with ROM-based BSL and also the default plugin demos or secondary BSL demo. The PC GUI can also convert target firmware from the TI-TXT hex format to a header file that can be used by the host MCU. The software package with examples and GUI are available in the SDK.

Table of Contents

List of Figures

List of Tables

Trademarks

Code Composer Studio™ is a trademark of Texas Instruments.
All trademarks are the property of their respective owners.

1 Introduction

1.1 Bootloader Introduction

A microcontroller bootloader is firmware that can be used to program the internal memory of the MCU using common interfaces. A bootloader enables quick and easy programming of the device through the entire life cycle, from development to production to firmware updates in the field.

MSPM0 devices are shipped with a highly customizable ROM-based bootloader supporting UART and I2C. It also provides the flash-based plugin interface demos for UART, I2C and SPI.

In this application note, the MCU being programmed is called the *target*, and the device or tool performing the update is called the *host*.

For more information about the MSPM0 bootloader (BSL), see the *MSPM0 Bootloader User's Guide*.

1.2 Implementation

This application note describes the implementation of two types of hosts: one is PC with an interface bridge like XDS110, the other is an MCU or processor. Figure 1-1 shows the signal connection diagram.

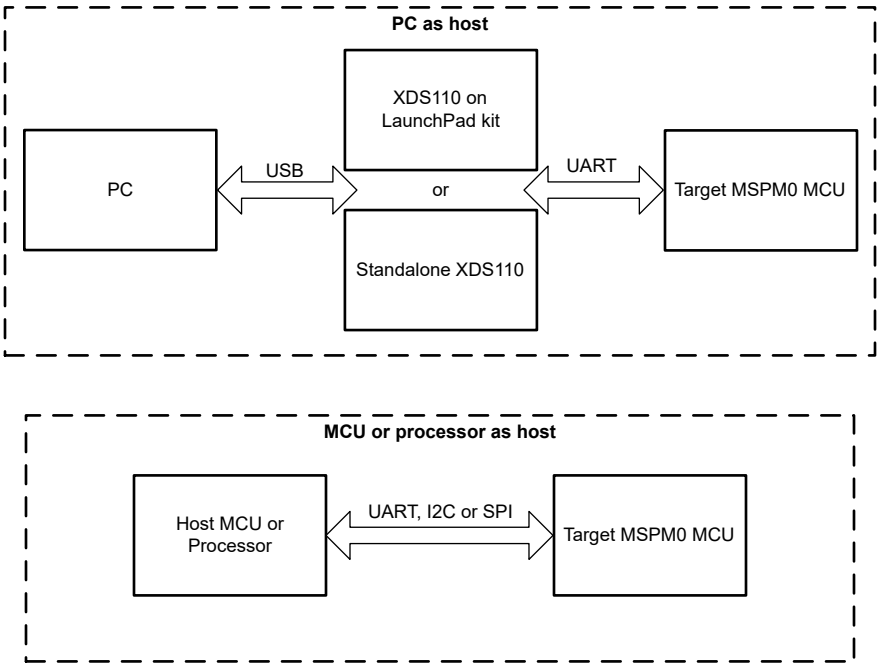

Figure 1-1. BSL Firmware Update System Block Diagram

MSPM0 UART is enabled with following configuration

- Baud rate: 9600 bps (can be changed)
- Data width: 8 bit
- One stop bit
- No parity

The I2C interface in the BSL acts as the I2C target. The PC acts as the controller and drives the communication.

- I2C target address is 0x48 by default (can be changed).
- External pullup resistors are required for the SCL and SDA lines.

The SPI module is being initialized in controller mode. Frame format, parity, data size and bit order should be configured in alignment with the target device being connected. The SPI internal functional clock is selected and divided from the clock sourced to this module. It can be MFCLK, LFCLK or BUSCLK according to the output SPI frequency that the user wants to attain. For the demo example, SPI is configured at 2 MHz sourced by MFCLK.

The GUI described in this application note was developed using Python3. A pre-built Windows executable (tested on Win10 64-bits) is included, along with the source code. Uniflash can also be used on PC side.

2 MCU Host Code Introduction

Two demos based on Code Composer Studio™ (CCS) are in the folder

< …\mspm0_sdk_xxxx\examples\nortos\LP_MSPM0xxxx\bsl

These demos can update the target MSPM0 device through UART, I2C or SPI.

The source code includes the target device firmware in application_image.h file that is converted from .txt image file by the GUI. For more details, see Section 2.2. It also includes the BSL password in the main.c file in the BSL_PW_RESET array. The password is defined in the non-main flash BSL configuration area BSLPW. Figure 2-1 shows a flow chart of the host project.

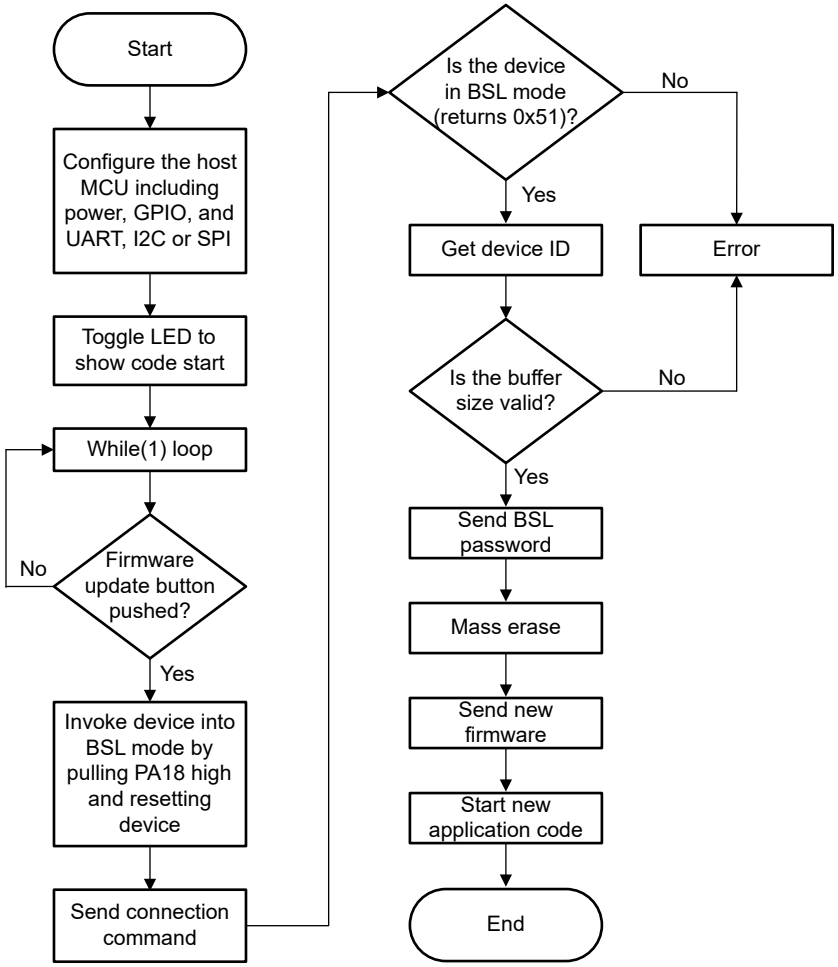

Figure 2-1. Flow Diagram of Host Project

2.1 Hardware Connection

This demo code uses a LP-MSPM0G3507 as the host side MCU. The hardware signals connection between host and target is shown in Table 2-1.

Table 2-1. Hardware Signal Connections

Interface	Host Device		Target Device		
	Signal	LP-MSPM0G3507	Signal	LP-MSPM0G3507	LP-MSPM0L1306
NRST	GPIO	PB0	NRST	NRST pin	NRST pin
Invoke	GPIO	PB16	Default Invoke pin	PA18	PA18
UART	RXD	PB7/UART1_RX	TXD	PA10/UART0_TX	PA23/UART0_TX
	TXD	PB6/UART1_TX	RXD	PA11/UART0_RX	PA22/UART0_RX
I2C	SCL	PB2/I2C1_SCL	SCL	PA1/I2C0_SCL	PA1/I2C0_SCL
	SDA	PB3/I2C1_SDA	SDA	PA0/I2C0_SDA	PA0/I2C0_SDA
SPI	SCLK	PB9/SPI1_SCLK	SCLK	PB9/SPI1_SCLK	PA6/SPI0_SCLK
	PICO	PB8/SPI1_PICO	PICO	PB8/SPI1_PICO	PA5/SPI0_PICO
	POCI	PB7/SPI1_POCI	POCI	PB7/SPI1_POCI	PA4/SPI0_POCI
	CS	PB6/SPI1_CS	CS	PB6/SPI1_CS	PA8/SPI0_CS

Note

Connect only one communication interface, either UART or I2C or SPI. The pins are just with default configuration that can be changed..

2.2 TXT to Header File Conversion

The MCU host firmware contains an image of the target application as a C header file. To do this job, this application note includes a conversion utility in the GUI MSPM0_BSL_GUI.exe in the following path:

< …\mspm0_sdk_xxxx\tools\bsl\BSL_GUI_EXE >.

1. Select TXT_to_H in the MoreOption menu.
2. Choose the TI-TXT format file to be converted. Sample files are provided in the folder named input.
3. Choose a folder for the output file (for example, choose the folder named Output).
4. Click the Convert button to start the conversion.

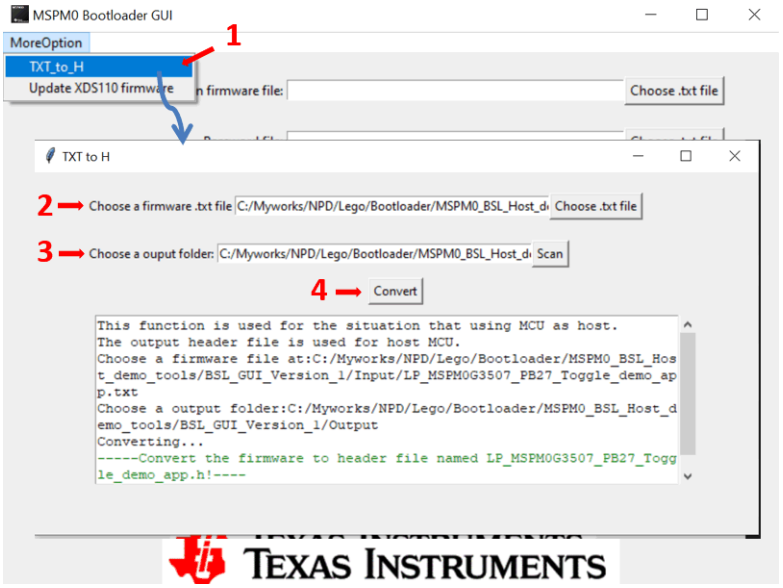

Figure 2-2. Steps to Convert TXT File to Header File

2.3 Step-by-Step Operation

The following steps describe how to program an MSPM0 MCU using a LP-MSPM0G3507 as the host. A MSPM0G3507 is used as target device, and UART is used for communication. A similar process can be used to program other MSPM0 devices through either UART, I2C or SPI by using the proper hardware connections (see Table 2-1).

1. Connect the hardware signals as shown in Figure 2-3. This example uses UART, so the I2C signals do not need to be connected.

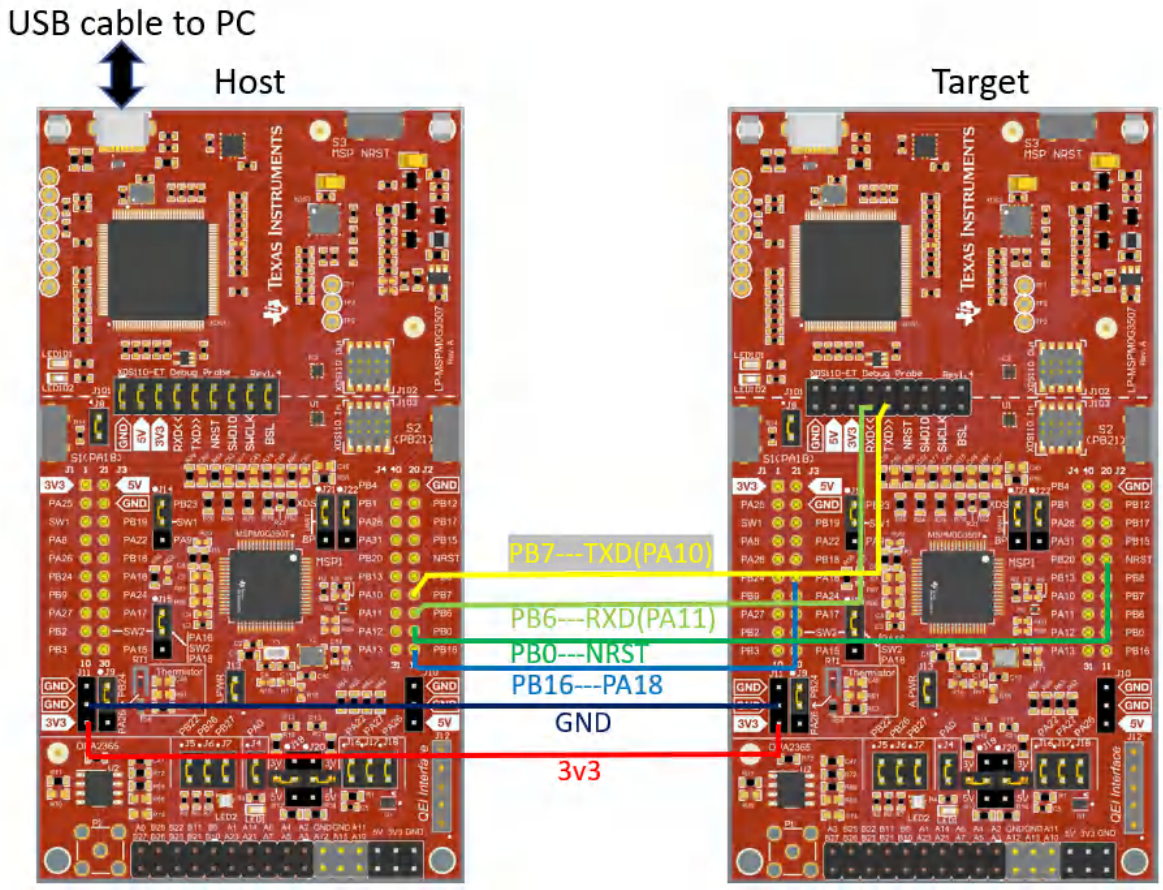

Figure 2-3. Hardware Signal Connections

2. Connect the jumpers as shown in Table 2-2.

Table 2-2. Jumper Connections

Board	Mode	Jumpers to Connect	Jumpers to Disconnect
LP-MSPM0G3507	Host	J101 (GND, 3V3), J4, J7, J9	None
	Target	J7, J21, J22 (to XDS)	All in J101

Note

If use LP-MSPM0L1306 as target board, jumper on J6 must be removed.

3. A UART demo project is available in the folder < …\mspm0_sdk_xxxx\examples\nortos\LP_MSPM0xxxx\bsl\bsl_host_mcu_uart>. Import the project into CCS.

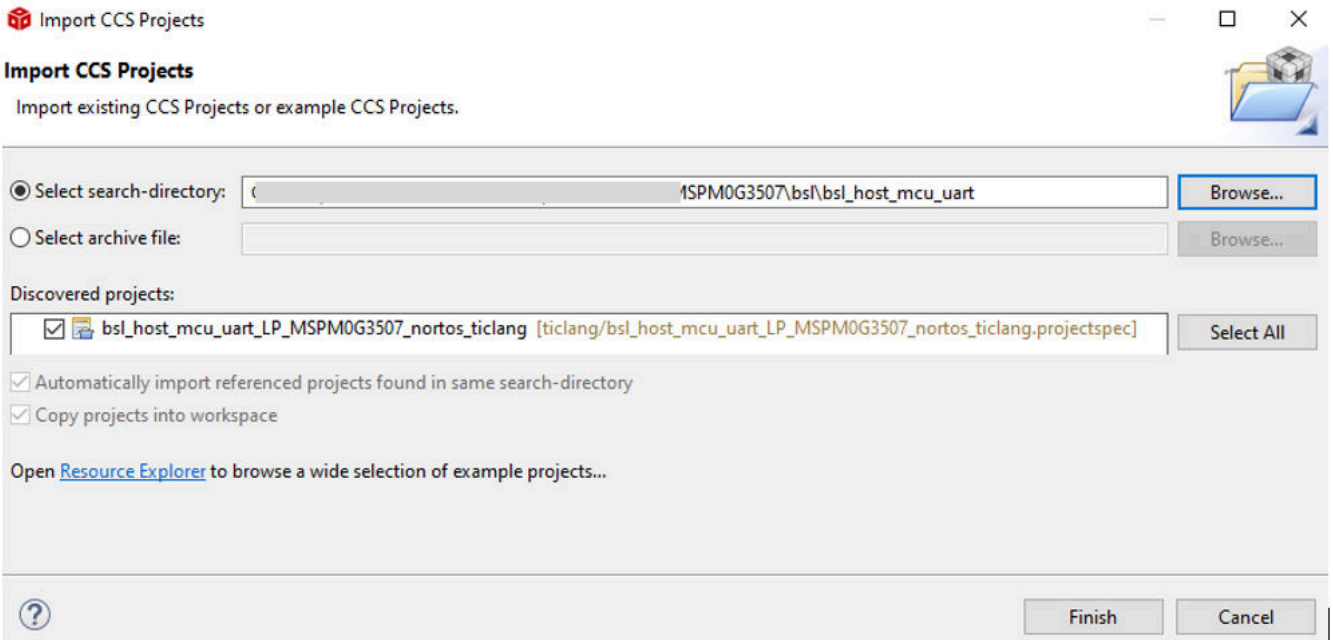

Figure 2-4. Import Host Project Into CCS

4. Modify the password in the bsl_password array in main.c, if necessary. The default password is all 0xFF with 32 bytes. The target BSL password is defined in the Non-Main memory. For more information, see the technical reference manual [1] [2] or the bootloader user's guide [3].
5. Prepare the target device firmware in TI-TXT hex format (see Figure 2-5).

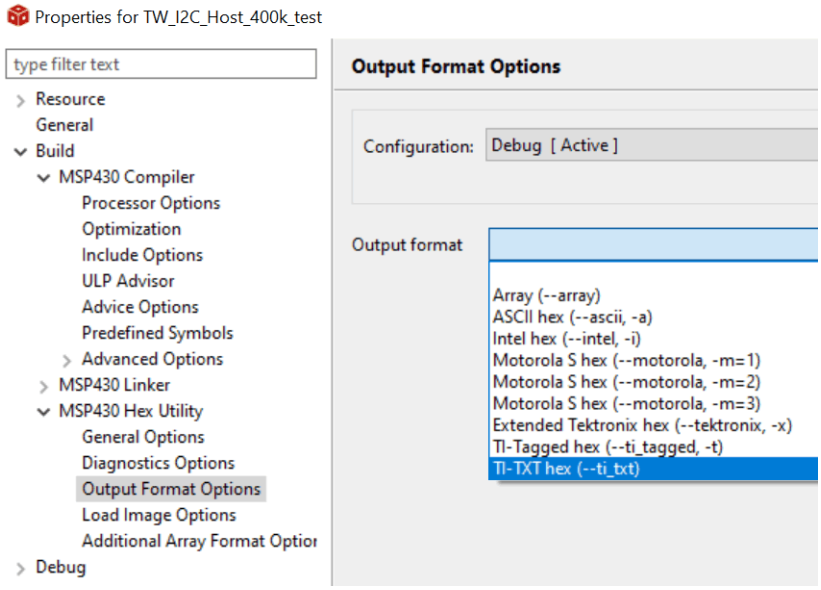

Figure 2-5. Generate TI-TXT Hex File in CCS

Two simple images are in the folder < …\mspm0_sdk_xxxx\tools\bsl\BSL_GUI_EXE\Input >.

6. Run the GUI MSPM0_BSL_GUI.exe to convert the target device firmware .txt format to a header file. For more details, see Section 2.2.
7. Copy the contents of the output file LP_MSPM0G3507_PB27_Toggle_demo_app.h into the host project file application_image.h.
8. Build the host project and download to LP-MSPM0G3507.
9. Push button S3 on the host board to initiate the firmware update. If the connections are correct and the firmware was updated, the green RGB LED blinks on the target board. If there is an error, LED1 turns on.

3 PC Host Example

The PC host uses a software GUI (MSPM0_BSL_GUI.exe or Uniflash) and a USB-to-UART bridge. Two hardware bridges are included (can be chosen in the MSPM0_BSL_GUI.exe): one is XDS110 on the MSPM0 LaunchPad kit and the other is a standalone XDS110. Both of the bridges support the backchannel UART that can be used as a USB-to-UART bridge. The XDS110 on the LaunchPad kit supports NRST pin and BSL invoke pin controlling that can be used by the GUI to start BSL mode on the MCU. For the standalone XDS110, two GPIO output pins (IOOUT0 and IOOUT1) in the AUX connection port can be used to control the NRST pin and BSL invoke pin on the target device and start BSL mode. (This is implemented with MSPM0_BSL_GUI.exe).

3.1 Prepare the Image File and Password File

Before downloading the firmware with the GUI, prepare two files: the application firmware file and the BSL password file.

The GUI (MSPM0_BSL_GUI.exe) supports only the TI-TXT format. For details on how to generate this format image file with CCS, see Step 4 in Section 2.3.

The format of the password file is similar to the TI-TXT format as shown in Figure 3-1. The BSL password is defined in the Non-Main memory. For more information, see the technical reference manual [1] [2] or the bootloader user's guide [3]. If the BSL password is not the default (all 0xFF), modify the password file. A default password file named BSL_Password32_Default.txt is available in this folder: < … \mspm0_sdk_xxxx\tools\bsl\BSL_GUI_EXE\Input >.

```
@password
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
q
```

Figure 3-1. BSL Default Password File (BSL_Password32_Default.txt)

3.2 Steps to Download Image File Into Device

1. Connect the target device and the XDS110 to the PC. When using the XDS110 integrated in the LaunchPad kit, connect the micro USB cable to the PC as Figure 3-2.

   The ROM-based BSL UART pins for MSPM0G3507 are PA10 and PA11, and the pins are directly connected to the XDS110 backchannel UART, so all of the jumpers in J101 are required (see Table 3-2).

   On the LP-MSPM0L1306, the XDS110 backchannel UART pins are different from the BSL UART pins, so disconnect TXD and RXD in J101 and use jumper wires (see Table 3-2).

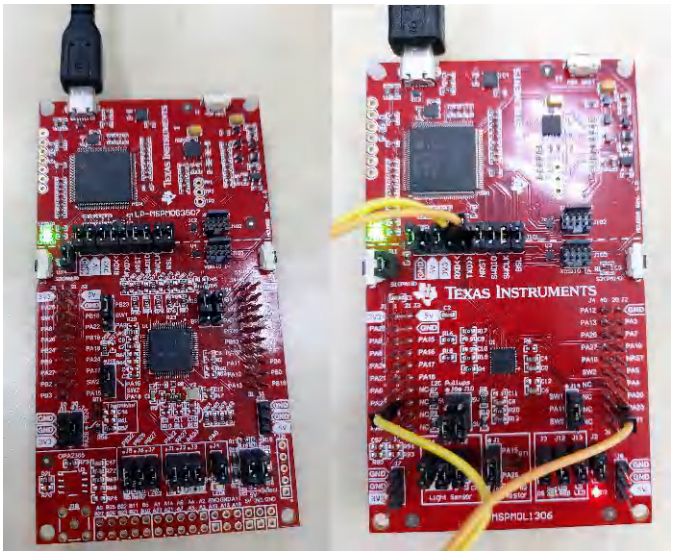

Figure 3-2. LaunchPad Kit Connection (Left: LP-MSPM0G3507, Right: LP-MSPM0L1306)

Table 3-1. Jumpers Connection

Boards	Mode	Jumpers Need Populated	Jumpers Need Unpopulated
LP-MSPM0G3507	Target	J101 (GND, 3V3, TXD, RXD, NRST, BSL), J4, J7, J21, J22 (To XDS)	NA
LP-MSPM0L1306	Target	J101 (GND, 3V3, NRST, BSL), J2, J3	J101 (TXD, RXD)

For standalone XDS110, the auxiliary interface (AUX) uses the signal connections in Table 3-2.

Table 3-2. Standalone Signal Connection

Signal	Standalone XDS110		Target Device		
	Signal	AUX Port	Signal	LP-MSPM0G3507	LP-MSPM0L1306
NRST	IO output	IOOUT0	NRST	NRST pin	NRST pin
Invoke	IO output	IOOUT1	Default: Invoke pin	PA18	PA18
UART	RXD	UARTRX	TXD	PA10/UART0_TX	PA23/UART0_TX
	TXD	UARTTX	RXD	PA11/UART0_RX	PA22/UART0_RX

2. Use the GUI to download the image to the target.
   a. Choose the TI-TXT format image file that need to be downloaded. (There are two demo images in the folder named input)
   b. Choose the TI-TXT format password file (a default file is in the *input* folder). For details on preparing this file, see Section 3.1.
   c. Choose hardware bridge.
   d. Click the download button.

   The GUI automatically invokes the BSL so there is no need to manually invoke the BSL during this operation.

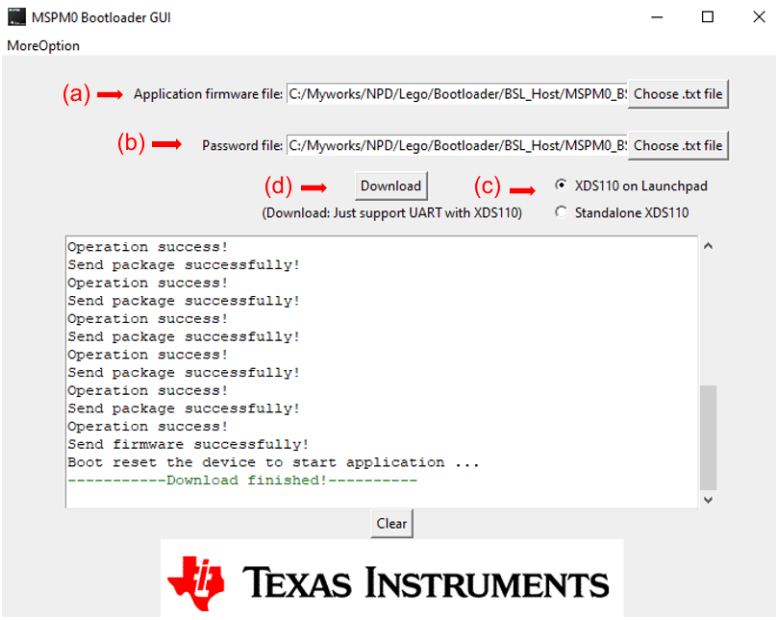

Figure 3-3. Steps to Download Image by GUI With UART

3. If using the XDS110, this GUI supports XDS110 firmware version firmware_3.0.0.20 or higher. If errors occur when download the image, update the XDS110 firmware.

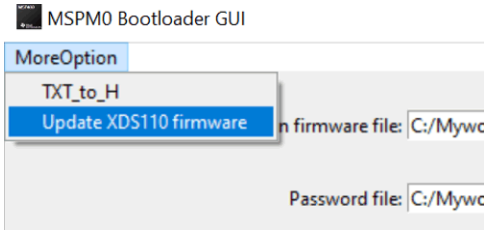

Figure 3-4. Update XDS110 Firmware

4 References

1. Texas Instruments: *MSPM0 G-Series 80-MHz Microcontrollers Technical Reference Manual*
2. Texas Instruments: *MSPM0 L-Series 32-MHz Microcontrollers Technical Reference Manual*
3. Texas Instruments: *MSPM0 Bootloader User's Guide*

Revision History

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

Changes from Revision * (March 2023) to Revision A (July 2023)	Page
• Update Abstract	1
• Updated the numbering format for tables, figures and cross-references throughout the document	2
• Updated Section 1.1	2
• Updated Section 1.2	2
• Updated Section 2	3
• Updated Section 2.1	4
• Updated Section 2.2	4
• Updated Section 2.3	5
• Updated Section 3	7
• Updated Section 3.1	7
• Updated Section 3.2	7

IMPORTANT NOTICE AND DISCLAIMER

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

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

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

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

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

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