Maxim Bootloader Tools
User Guide

UG7510; Rev 0; 10/21

Abstract

Maxim provides generic and secure variants of flash bootloader firmware for microcontrollers that enable firmware update and protect customer IP by secure communication. This user guide provides a detailed explanation of how to communicate with Maxim bootloaders. The document should be used in conjunction with the target device bootloader user guide.

Table of Contents

List of Figures

List of Tables

Introduction

The bootloader is a special firmware that loads inside the main microcontroller, and generally the purpose of the bootloaders is to load the main application and provide firmware update functionality.

For some microcontrollers, Maxim provides the bootloader firmware that gives target microcontroller firmware update functionality and security. There are two types of Maxim bootloaders:

1. Generic bootloader (provides firmware update capability)
2. Secure bootloader (provides firmware updates + secure transfer of firmware)

The Maxim bootloader solution is capable of handling communication over I^2C, SPI, or UART interface. These bootloaders utilize a special communication protocol which is described in the bootloader user guide to configure bootloader options and perform the firmware update. Detailed information on the protocol is available in the target device's bootloader user guide.

Maxim provides collaterals to ease the porting effort of the customer for the target platform and enable an evaluation platform for the bootloaders. The collaterals are listed as follows:

- Bootloader Programmer Board: To enable I^2C/SPI communication through USB CDC ACM.
- Bootloader PC GUI Tool: A PC tool which provides GUI interface to configure target and load firmware.
- Bootloader Host Example: A platform-independent bootloader host example, which can be used by the customers while developing their bootloader host.

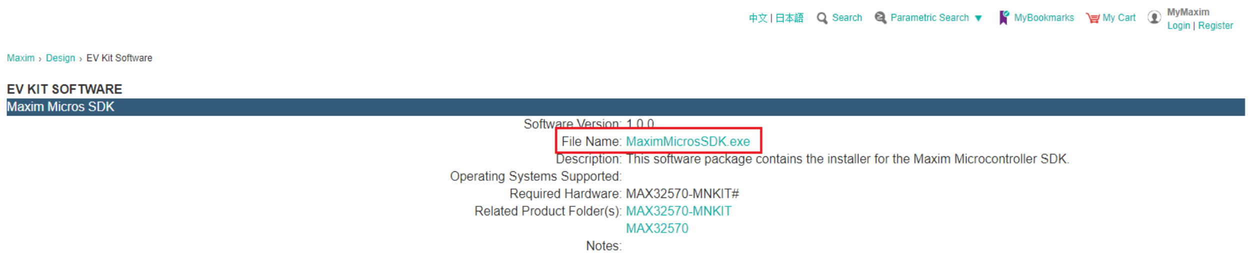
Bootloader-related software and tools are delivered to customers by Maxim Software Development Kit (MSDK).

The rest of this document is structured to guide users on steps for communicating with the bootloader and porting host programmer for the target platform.

Maxim SDK Installation

The MSDK includes all necessary tools and source code to use Maxim microcontrollers. For bootloader case, the MSDK installs PC tools and hosts programmer example projects on the target machine. If you do not have Maxim Micros SDK, download Maxim **Micros Software Development Kit** (SDK) from the website and install it on your machine to get all stuff.

https://www.maximintegrated.com/en/design/software-description.html/swpart=SFW0010820A



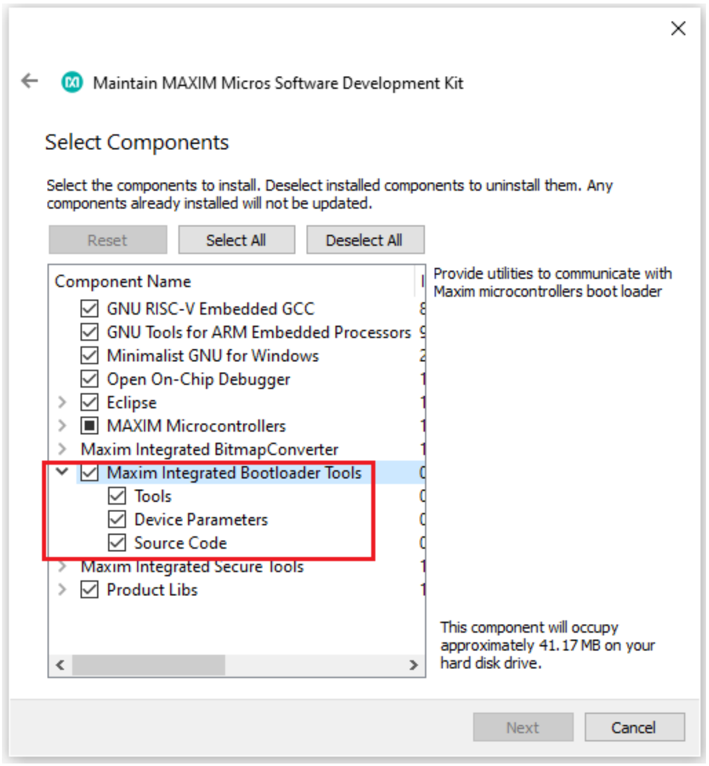
EV KIT SOFTWARE

Maxim Micros SDK

Software Version:	1.0.0
File Name:	MaximMicrosSDK.exe
Description:	This software package contains the installer for the Maxim Microcontroller SDK.
Operating Systems Supported:	
Required Hardware:	MAX32570-MNKIT#
Related Product Folder(s):	MAX32570-MNKIT MAX32570
Notes:	

Figure 1. Maxim software development kit (SDK) on Maxim web page.

To start the installation, after download double-click on the MaximMicros.exe file then follow instructions. The "Maxim Integrated Bootloader Tools" should be selected to bootloader related package to be installed on the machine.



Maintain MAXIM Micros Software Development Kit

Select Components

Select the components to install. Deselect installed components to uninstall them. Any components already installed will not be updated.

Component Name
- [✓] GNU RISC-V Embedded GCC
- [✓] GNU Tools for ARM Embedded Processors
- [✓] Minimalist GNU for Windows
- [✓] Open On-Chip Debugger
- [✓] Eclipse
- [■] MAXIM Microcontrollers
- Maxim Integrated BitmapConverter
- [✓] Maxim Integrated Bootloader Tools
  - [✓] Tools
  - [✓] Device Parameters
  - [✓] Source Code
- Maxim Integrated Secure Tools
- [✓] Product Libs

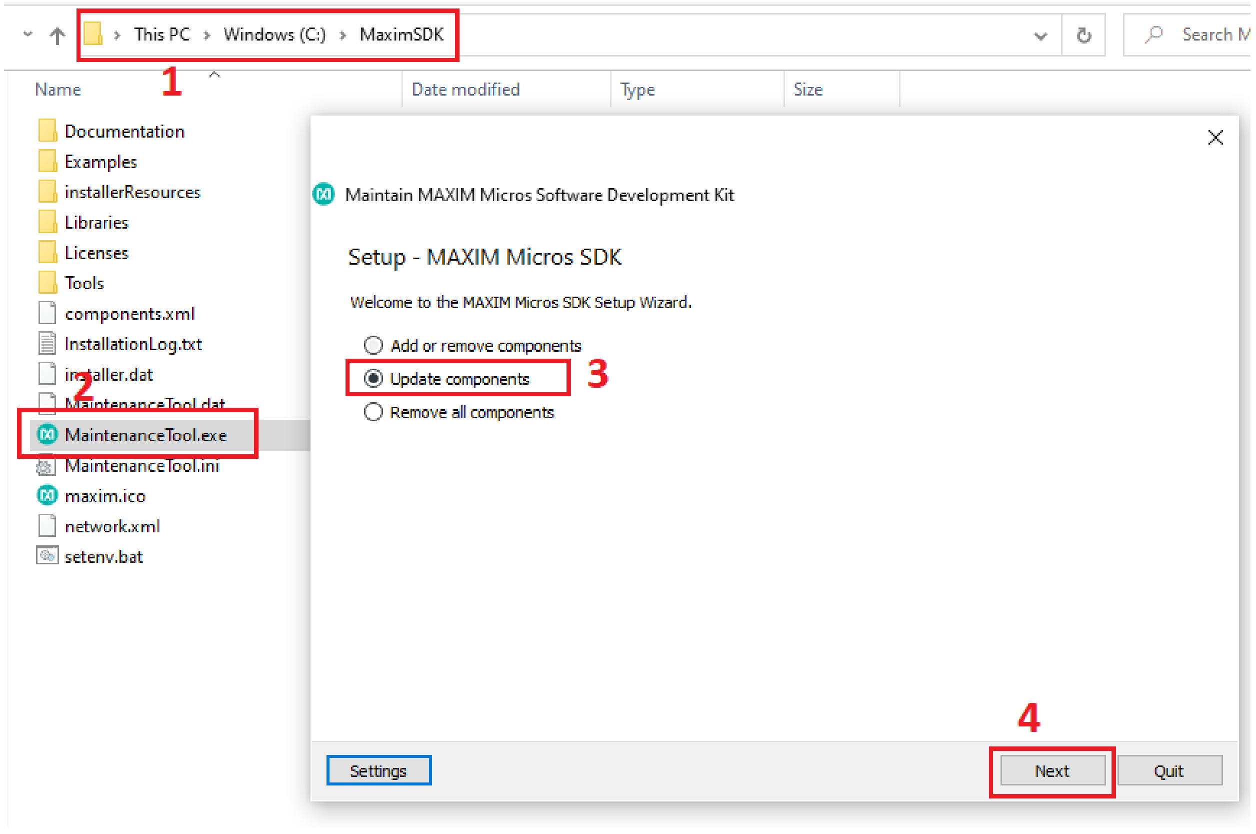
Provide utilities to communicate with Maxim microcontrollers boot loader

This component will occupy approximately 41.17 MB on your hard disk drive.

Figure 2. Bootloader tools installation package.

If you already have MSDK but do not have Bootloader GUI Tools, install the bootloader-related packages using the following steps:

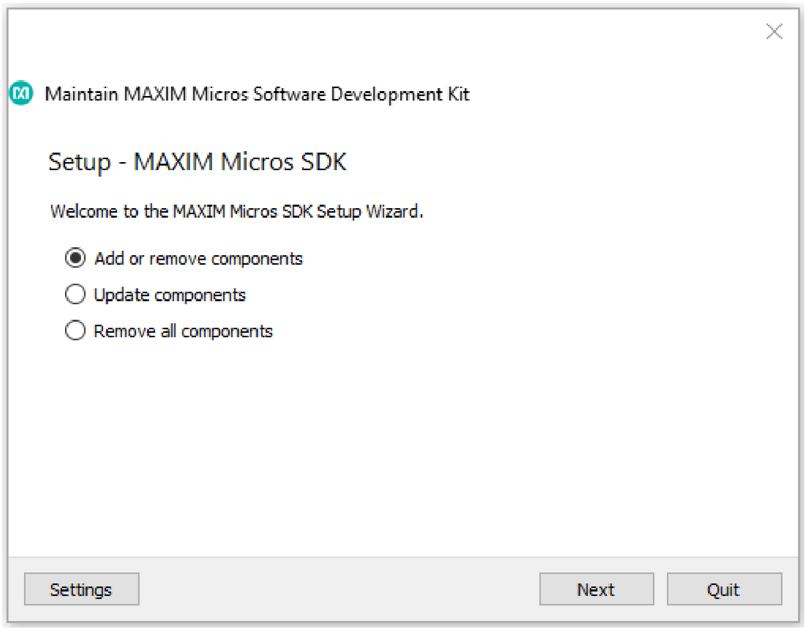
1. Go to the folder that you installed Maxim SDK. (The default installation path is C:/MaximSDK)
2. Run SDK MaintenanceTools.exe.
3. Select **Update components** to update the existing packages.
4. Click **Next** and install package.



Add or remove components [✓ Update components] Remove all components

Figure 3. MaintenanceTool update existing component.

Then, restart MaintenanceTools.exe and select the **Add or remove components** option. Click **Next**.



Maintain MAXIM Micros Software Development Kit

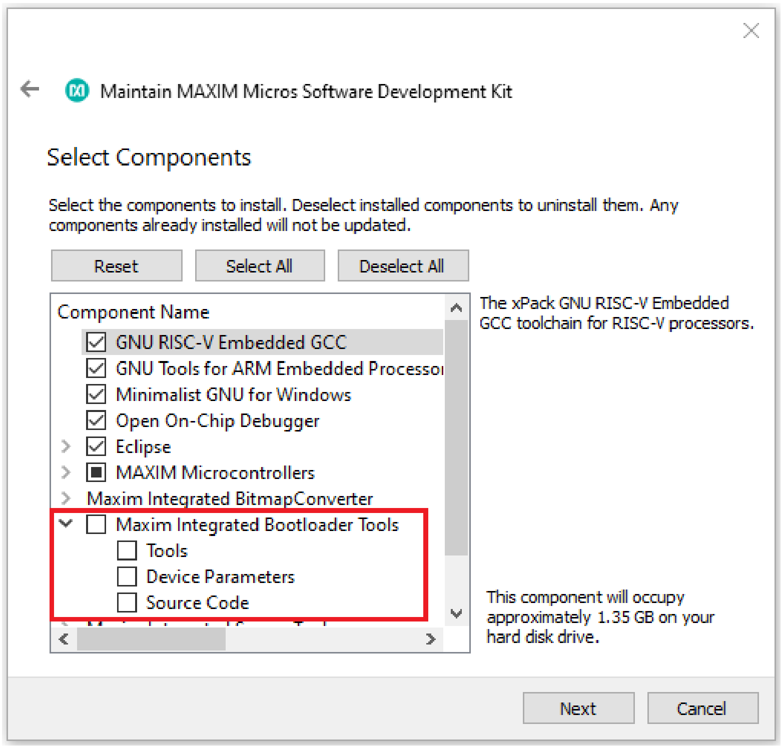
Setup - MAXIM Micros SDK

Welcome to the MAXIM Micros SDK Setup Wizard.

(●) Add or remove components
( ) Update components
( ) Remove all components

Figure 4. MaintenanceTool add remove component.

Check the "Maxim Integrated Bootloader Tools" component, and click **Next**. The bootloader-related packages will be installed.



Maintain MAXIM Micros Software Development Kit

Select Components

Select the components to install. Deselect installed components to uninstall them. Any components already installed will not be updated.

- [x] GNU RISC-V Embedded GCC
- [x] GNU Tools for ARM Embedded Processo
- [x] Minimalist GNU for Windows
- [x] Open On-Chip Debugger
- [x] Eclipse
- ■ MAXIM Microcontrollers
- Maxim Integrated BitmapConverter
- [ ] Maxim Integrated Bootloader Tools
  - [ ] Tools
  - [ ] Device Parameters
  - [ ] Source Code

The xPack GNU RISC-V Embedded GCC toolchain for RISC-V processors.

This component will occupy approximately 1.35 GB on your hard disk drive.

Figure 5. MaintenanceTool bootloader tools installation package.

Bootloader Programmer Board

The bootloader programmer board (Interface board) is designed by Maxim Integrated. The purpose of this hardware is to provide USB to I^{2}C/SPI bridge functionality. The firmware on this hardware is a customized version of **DAPLink** firmware. This adapter supports standard **DAPLink** firmware features (SWD interface and Drag-n-drop programming). Additionally, it works as a UART to I^{2}C/SPI bridge.

There are I^{2}C/SPI/UART and BL0/1 pins on the interface board as shown in Figure 6.

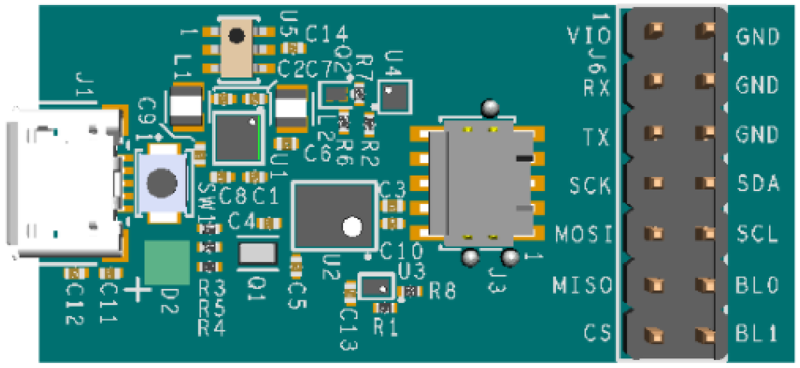


Figure 6. Maxim bootloader programmer board.

The BL0 and BL1 are generic-purpose GPIO pins. These pins are connected to target RESET and MFIO (Multi-Function Input-Output) pins to reset the target and keep it in the bootloader state.

To provide bridge functionality, a lightweight protocol layer has been added inside the **DAPLink** firmware. The protocol details are explained in Appendix A.

You need to make sure that the bootloader programmer board has proper DAPLink firmware installed on top of it. The related DAPLink firmware can be found in <sdk_installed_path>\Tools\BT\daplink. The default SDK path is C:\MaximSDK.

Depending on the target device's bootloader interface (I^{2}C/SPI/UART), corresponding interface connections between the bootloader programmer board and target device should be completed. For pin mapping, refer to the target bootloader user guide document.

Maxim Bootloader GUI Tool

Maxim Bootloader GUI Tool is a PC application that is capable of programming and configuring the target bootloader over "Bootloader Programmer" hardware as shown in the block diagram (Figure 7).

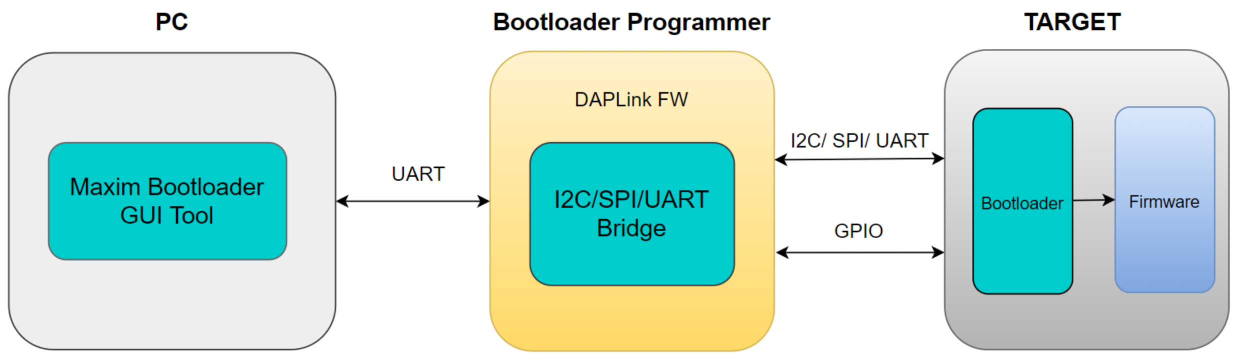


Figure 7. Block diagram of components.

This tool comes with MSDK installer and during the installation, the bootloader GUI tool is registered on the Start menu. The user needs to type "Maxim Bootloader Tools" in the Start menu and click it as shown in Figure 8.

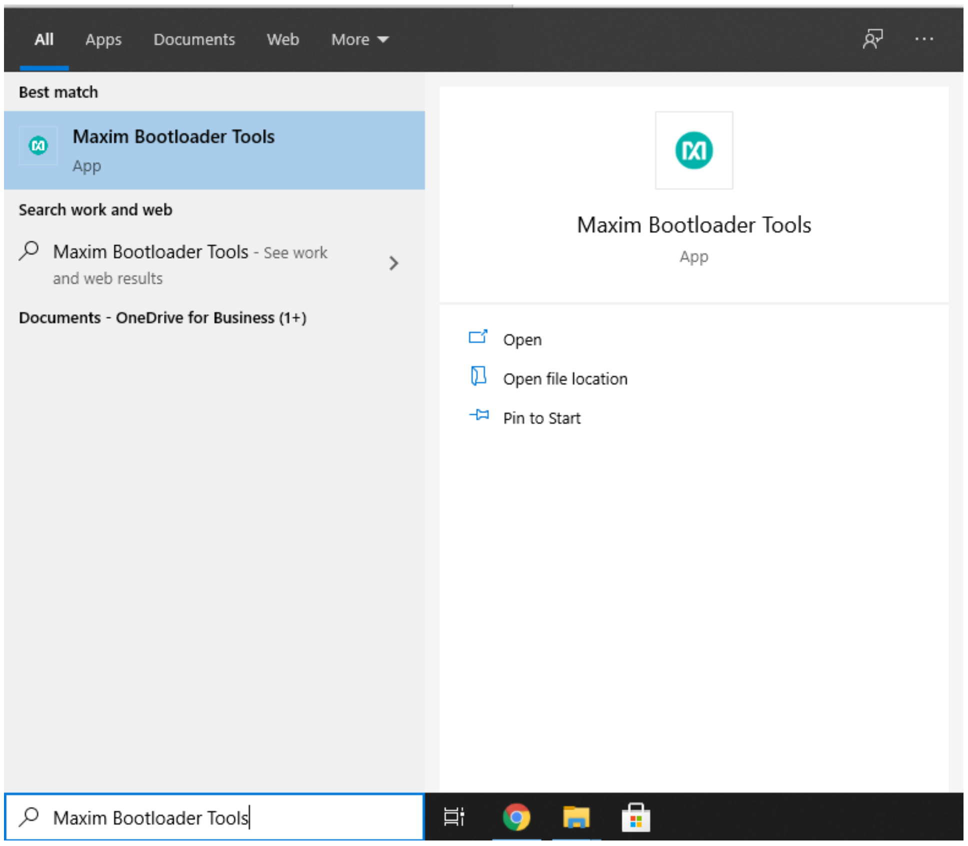


Figure 8. Maxim bootloader tool on start menu.

System Requirements

This section lists required equipment, instructions, and required connections for the proper set up of test platform. You may have a different target EV kit. Please check your EV kit bootloader user guide to get the target device pin configuration.

- Maxim Bootloader Programmer (MAX32625-PICO2 HW)
- Target EV kit (for this case MAX32660-EVSYS is used)
- 2 USB Type-A to Micro-USB Type B cables
- Some female–female cables
- Windows® PC (Windows 10) machine
- The latest Maxim SDK version must be installed on the PC. For details of the installation process, refer to the *Maxim Micro SDK Installation and Maintenance User Guide*.

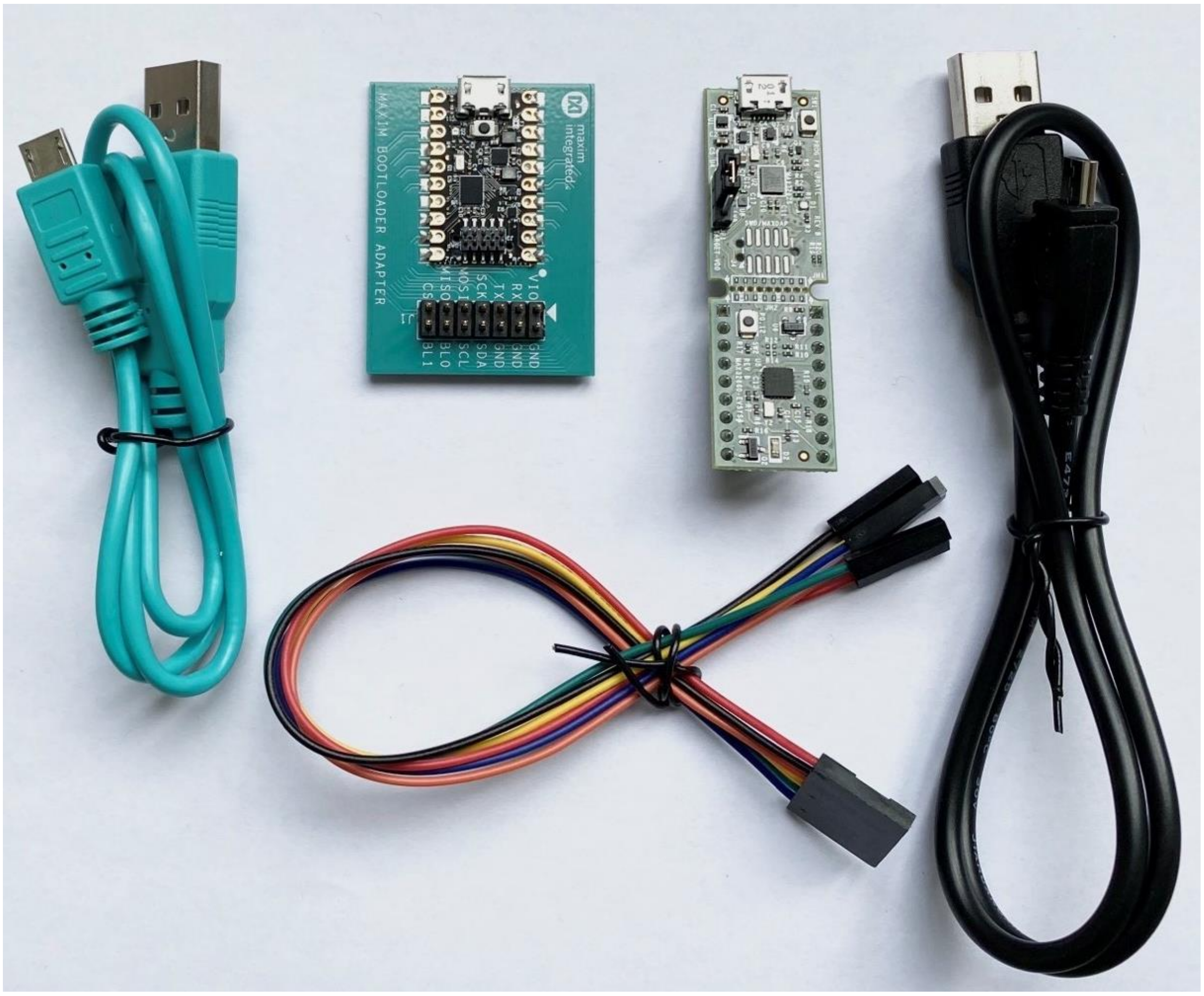


Figure 9. Components to test bootloader GUI tool.

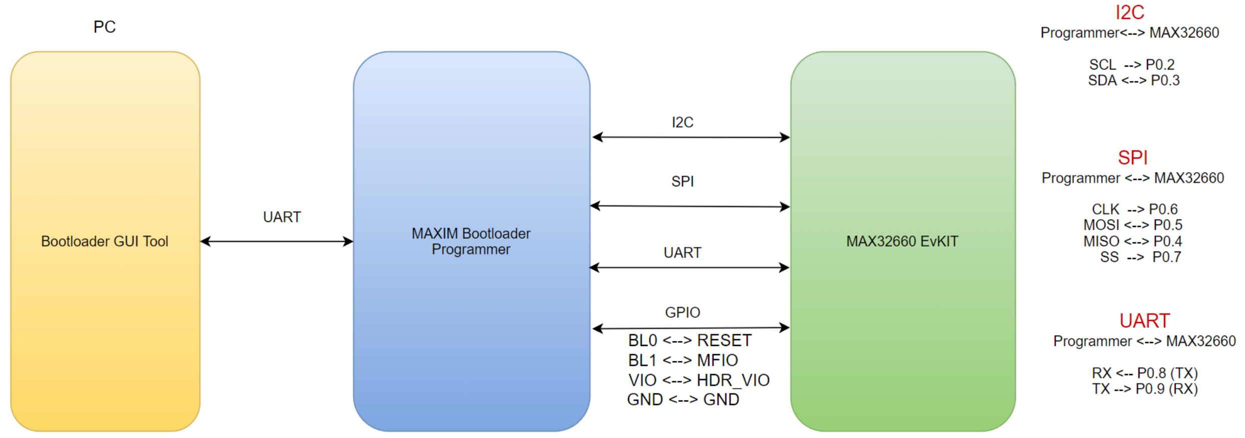
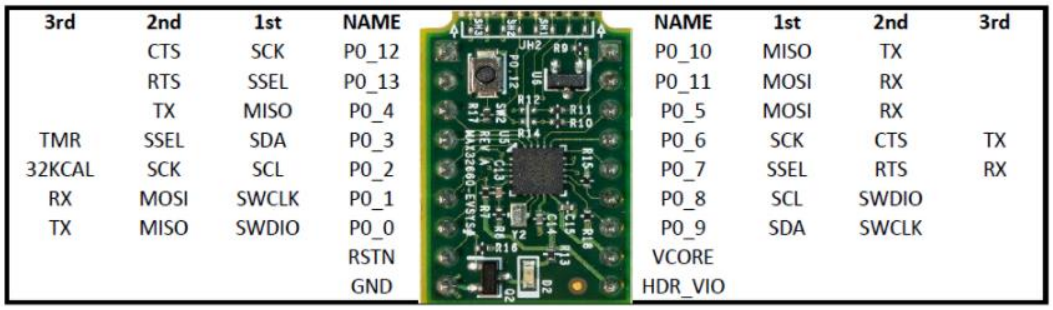


Figure 10. The overall system required pin connections.



3rd	2nd	1st	NAME	NAME	1st	2nd	3rd
	CTS	SCK	P0_12	P0_10	MISO	TX	
	RTS	SSEL	P0_13	P0_11	MOSI	RX	
	TX	MISO	P0_4	P0_5	MOSI	RX	
TMR	SSEL	SDA	P0_3	P0_6	SCK	CTS	TX
32KCAL	SCK	SCL	P0_2	P0_7	SSEL	RTS	RX
RX	MOSI	SWCLK	P0_1	P0_8	SCL	SWDIO	
TX	MISO	SWDIO	P0_0	P0_9	SDA	SWCLK	
			RSTN	VCORE			
			GND	HDR_VIO			

Figure 11. MAX32660 EVSYS pins.

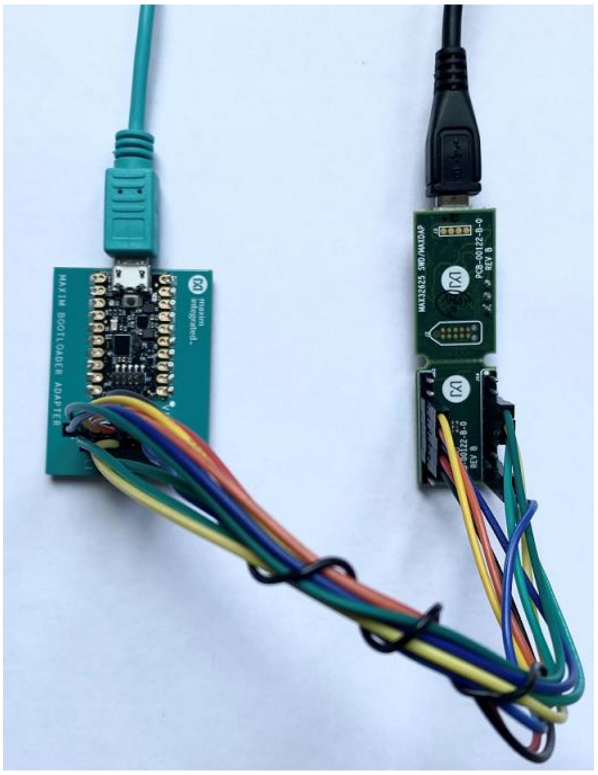


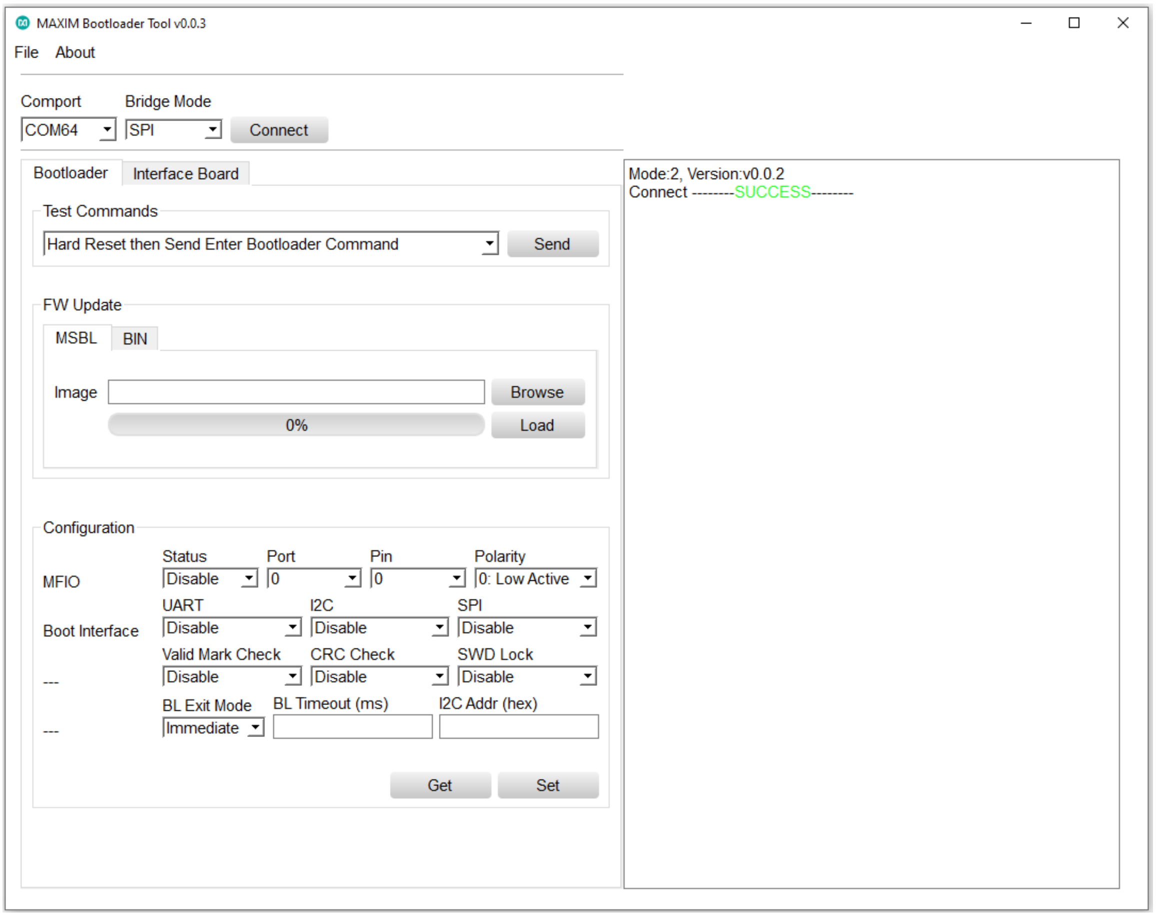
Figure 12. Bootloader GUI tool test setup.

Maxim Bootloader Tool Usage

The main screen of the bootloader GUI tool is shown in Figure 13 Before communicating with the target, the bootloader programmer board needs to be configured in the following manner:

- Select Comport: This was created by the "Maxim Bootloader Programmer" interface.
- Bridge Mode: The interface that is required to communicate with the target.
- Click **Connect** button.

You should see "SUCCESS" on the right side for success case.



Comport: COM64
Bridge Mode: SPI

Test Commands: Hard Reset then Send Enter Bootloader Command

```
Mode:2, Version:v0.0.2
Connect --------SUCCESS--------
```

MFIO	Status: Disable	Port: 0	Pin: 0	Polarity: 0: Low Active
Boot Interface	UART: Disable	I2C: Disable	SPI: Disable	
---	Valid Mark Check: Disable	CRC Check: Disable	SWD Lock: Disable	
---	BL Exit Mode: Immediate	BL Timeout (ms):	I2C Addr (hex):	

Figure 13. Maxim bootloader GUI tool.

After connecting to the adapter the system is ready to communicate with the bootloader for further operations. Commands that do not require parameter for the bootloader are added under "Test Commands" drop-down under **Bootloader** tab. See Figure 14.

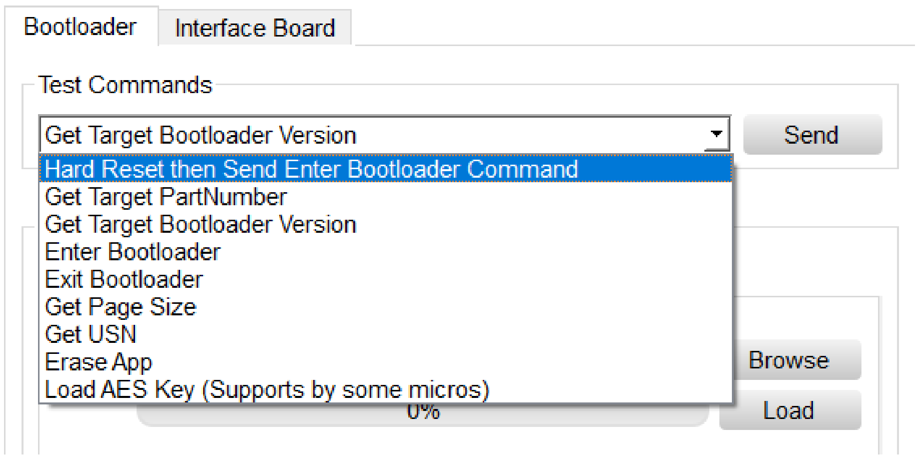
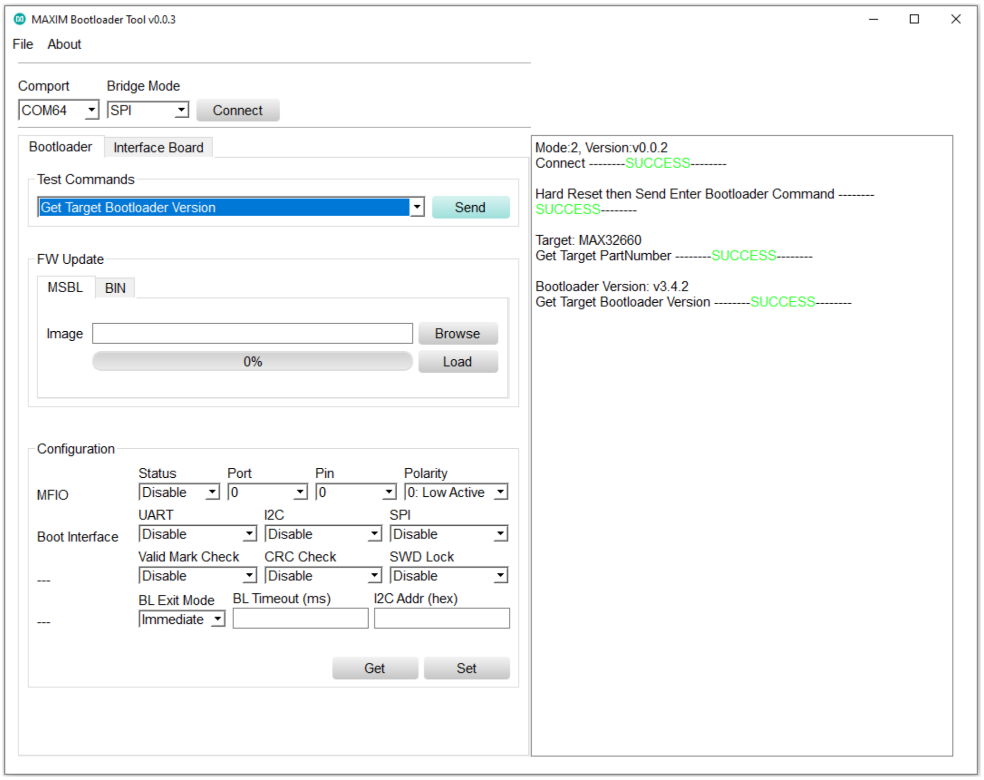

Bootloader
Interface Board
Test Commands
Get Target Bootloader Version
Send
Hard Reset then Send Enter Bootloader Command
Get Target PartNumber
Get Target Bootloader Version
Enter Bootloader
Exit Bootloader
Get Page Size
Get USN
Erase App
Load AES Key (Supports by some micros)
Browse
0%
Load

Figure 14. Bootloader test command.

For a quick test, select a command and send it.

- Send: "Hard Reset then Send Enter Bootloader Command"
- Send: "Get Target PartNumber"
- Send: "Get Target Bootloader Version"

Each test command result and the bootloader response is printed on the right side as shown in Figure 15 so that please check right side to see command execution result.



Comport: COM64
Bridge Mode: SPI

Test Commands: Get Target Bootloader Version

FW Update: MSBL | BIN
Image:

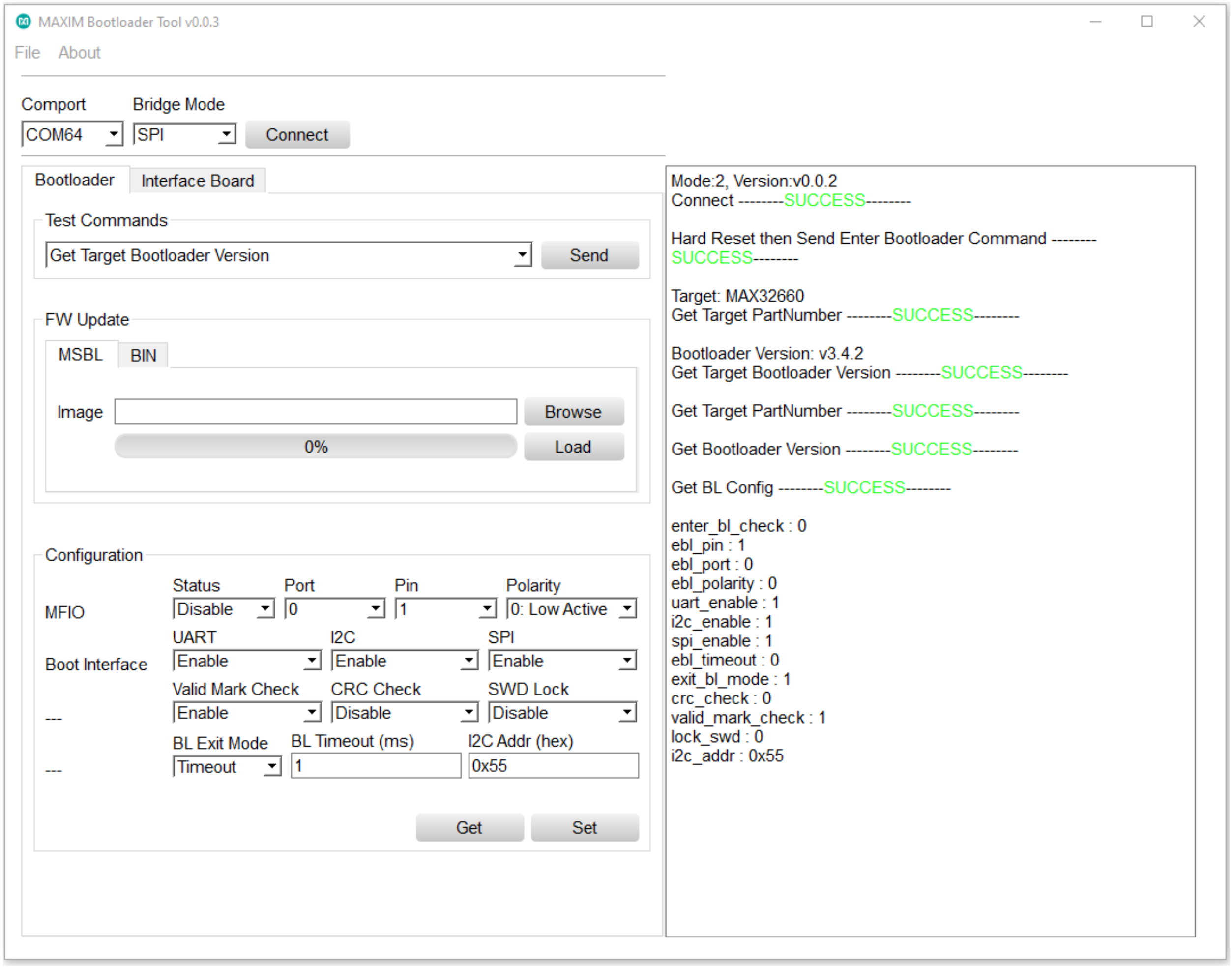
Configuration

	Status	Port	Pin	Polarity
MFIO	Disable	0	0	0: Low Active
	UART	I2C	SPI	
Boot Interface	Disable	Disable	Disable	
	Valid Mark Check	CRC Check	SWD Lock	
---	Disable	Disable	Disable	
	BL Exit Mode	BL Timeout (ms)	I2C Addr (hex)	
---	Immediate			

```
Mode:2, Version:v0.0.2
Connect --------SUCCESS--------

Hard Reset then Send Enter Bootloader Command --------
SUCCESS--------

Target: MAX32660
Get Target PartNumber --------SUCCESS--------

Bootloader Version: v3.4.2
Get Target Bootloader Version --------SUCCESS--------
```

Figure 15. Bootloader GUI tool running some bootloader test command.

The bootloader configuration parameters can get and set over the "Configuration" section. For more information about bootloader commands and configuration parameters, refer to target bootloader user guide.



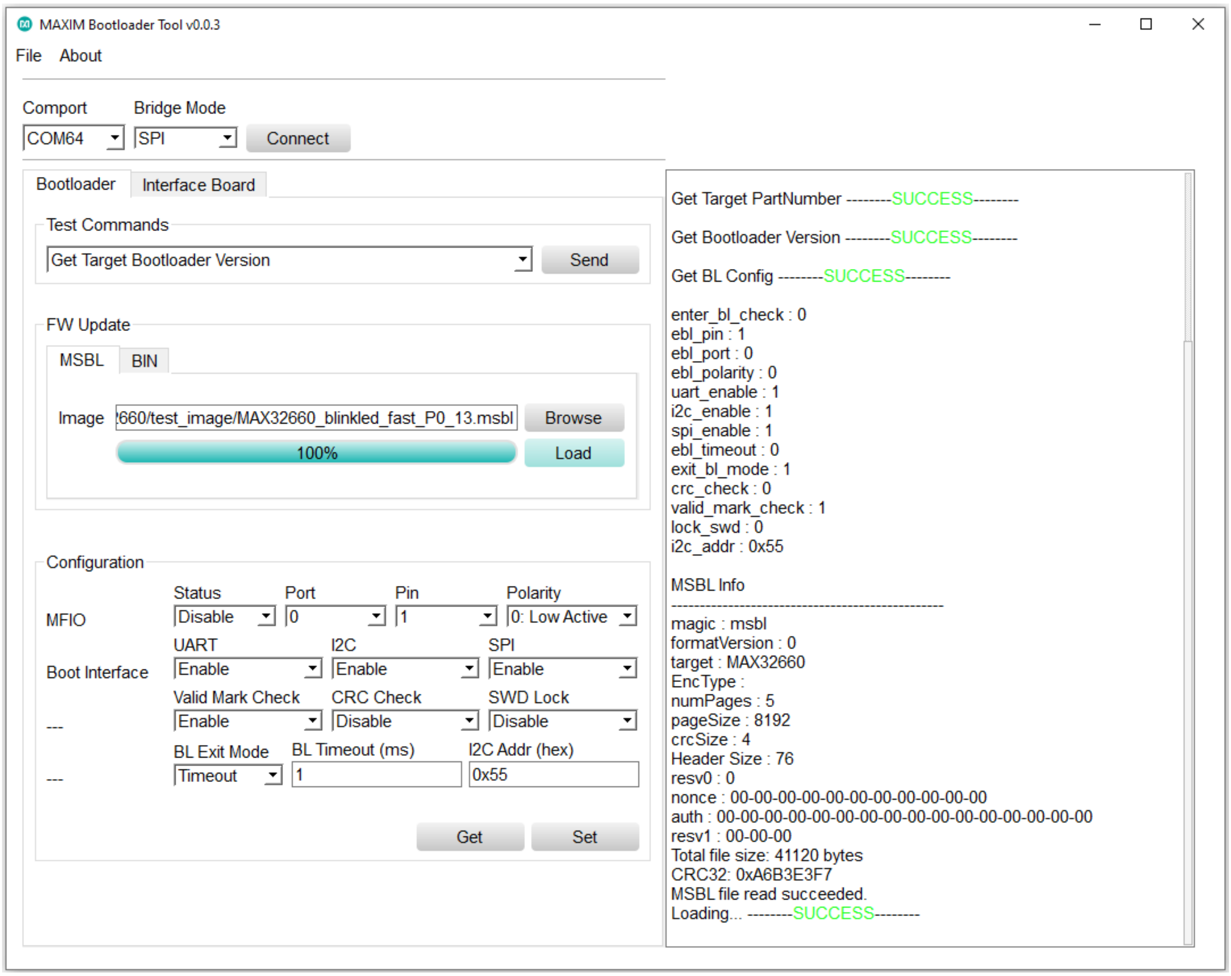
```
Mode:2, Version:v0.0.2
Connect --------SUCCESS--------

Hard Reset then Send Enter Bootloader Command --------
SUCCESS--------

Target: MAX32660
Get Target PartNumber --------SUCCESS--------

Bootloader Version: v3.4.2
Get Target Bootloader Version --------SUCCESS--------

Get Target PartNumber --------SUCCESS--------

Get Bootloader Version --------SUCCESS--------

Get BL Config --------SUCCESS--------

enter_bl_check : 0
ebl_pin : 1
ebl_port : 0
ebl_polarity : 0
uart_enable : 1
i2c_enable : 1
spi_enable : 1
ebl_timeout : 0
exit_bl_mode : 1
crc_check : 0
valid_mark_check : 1
lock_swd : 0
i2c_addr : 0x55
```

Figure 16. Bootloader GUI tool reading bootloader configuration parameters.

To update the target firmware, use the "FW Update" section, To start firmware update:

- Click the **Browse** button and select image (*.msbl or *.bin file)
- Then, click the **Load** button.

Successful load results can be seen in Figure 17.



Comport: COM64
Bridge Mode: SPI

Test Commands: Get Target Bootloader Version

FW Update

Image: 660/test_image/MAX32660_blinkled_fast_P0_13.msbl

100%

Configuration

	Status	Port	Pin	Polarity
MFIO	Disable	0	1	0: Low Active
	UART	I2C	SPI	
Boot Interface	Enable	Enable	Enable	
	Valid Mark Check	CRC Check	SWD Lock	
---	Enable	Disable	Disable	
	BL Exit Mode	BL Timeout (ms)	I2C Addr (hex)	
---	Timeout	1	0x55	

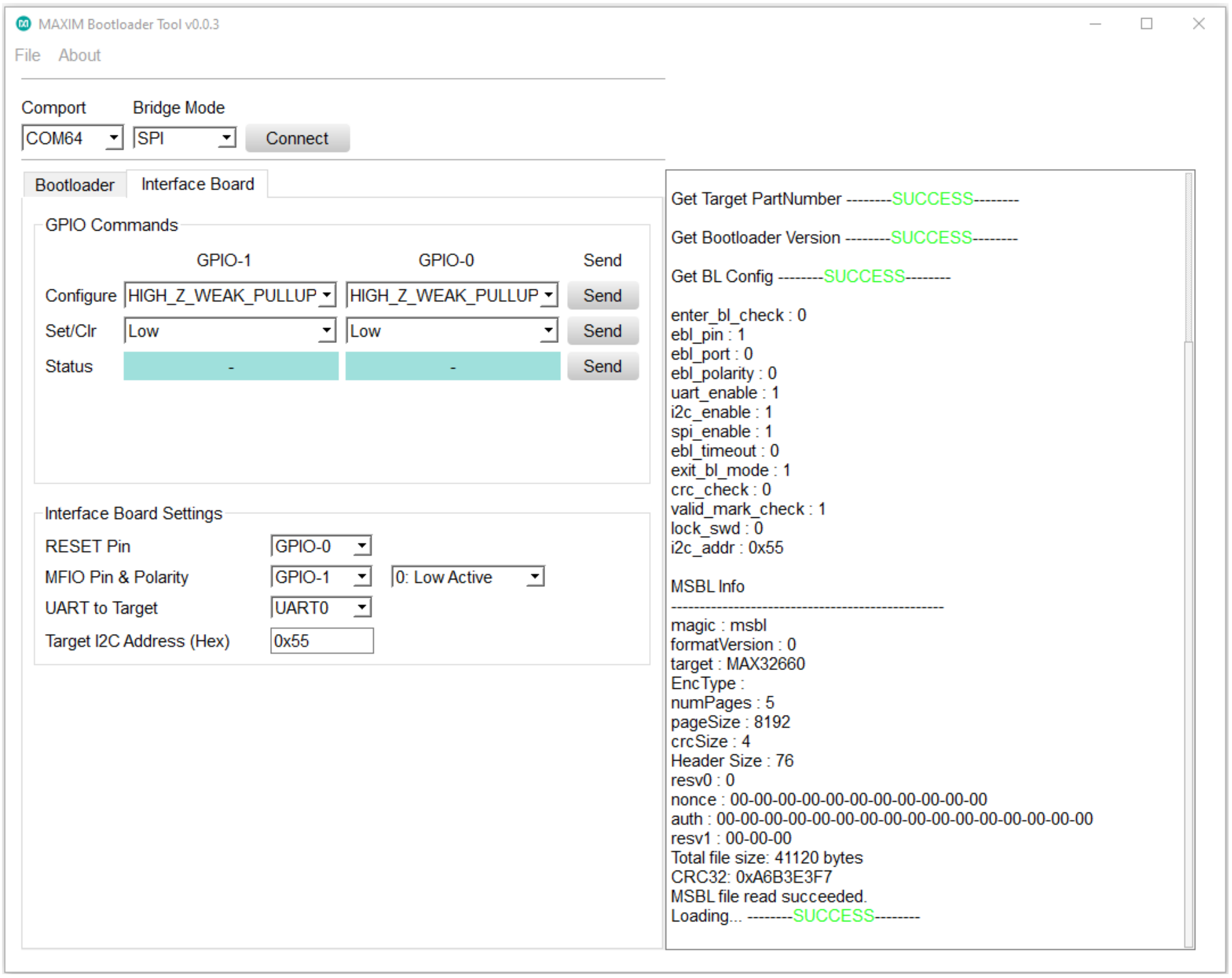
```
Get Target PartNumber --------SUCCESS--------

Get Bootloader Version --------SUCCESS--------

Get BL Config --------SUCCESS--------

enter_bl_check : 0
ebl_pin : 1
ebl_port : 0
ebl_polarity : 0
uart_enable : 1
i2c_enable : 1
spi_enable : 1
ebl_timeout : 0
exit_bl_mode : 1
crc_check : 0
valid_mark_check : 1
lock_swd : 0
i2c_addr : 0x55

MSBL Info
------------------------------------------------
magic : msbl
formatVersion : 0
target : MAX32660
EncType :
numPages : 5
pageSize : 8192
crcSize : 4
Header Size : 76
resv0 : 0
nonce : 00-00-00-00-00-00-00-00-00-00-00
auth : 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
resv1 : 00-00-00
Total file size: 41120 bytes
CRC32: 0xA6B3E3F7
MSBL file read succeeded.
Loading... --------SUCCESS--------
```

Figure 17. Bootloader GUI tool firmware loading.

If you have .bin and key file instead of .msbl (Maxim secure bootloader) you should use **BIN** tab, then select .bin file and provide the key file path.

As mentioned in the Maxim *Bootloader Programmer* section, the **Interface Board** tab is used to configure the adapter The BL0/1 pins are used to RESET and MFIO on the bootloader side. The "Interface Board" section provides a way to test and configure these pins.

For test purposes, you can configure GPIO, set the logic level to high or low, and read GPIO status.



```
Get Target PartNumber --------SUCCESS--------

Get Bootloader Version --------SUCCESS--------

Get BL Config --------SUCCESS--------

enter_bl_check : 0
ebl_pin : 1
ebl_port : 0
ebl_polarity : 0
uart_enable : 1
i2c_enable : 1
spi_enable : 1
ebl_timeout : 0
exit_bl_mode : 1
crc_check : 0
valid_mark_check : 1
lock_swd : 0
i2c_addr : 0x55

MSBL Info
------------------------------------------------
magic : msbl
formatVersion : 0
target : MAX32660
EncType :
numPages : 5
pageSize : 8192
crcSize : 4
Header Size : 76
resv0 : 0
nonce : 00-00-00-00-00-00-00-00-00-00-00
auth : 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
resv1 : 00-00-00
Total file size: 41120 bytes
CRC32: 0xA6B3E3F7
MSBL file read succeeded.
Loading... --------SUCCESS--------
```

Figure 18. Maxim bootloader tool testing and configuring bootloader programmer board.

Interface board settings:

- RESET Pin: To select RESET pin, which pin (BL0/1) will be connected to target RESET pin.
- MFIO Pin and Polarity: To select MFIO pin and polarity, which pin (BL0/1) will be used for target MFIO pin.
- UART to Target: There are two universal asynchronous receiver-transmitters (UARTs) on the host programmer board (UART0/2).
- Target I^{2}C Address: The I^{2}C address that is used between the Bootloader Programmer Board–Target I^{2}C communication.

Bootloader Host Example

Maxim provides a platform-independent bootloader host programmer example project with source code to users, so that they easily and quickly port it on their system and communicate with the bootloader.

To be a reference, the platform-independent bootloader host code is ported on the MAX32665 SDK. The MAX32665 SDK has a "Bootloader Host" example project which is able to communicate with Maxim bootloader firmware.

System Requirement

This section lists required equipment, instructions, and required connections to the setup of EV kit properly. You may have a different target EV kit, check your EV kit bootloader user guide to get your target device pin configuration.

- MAX32665 EV kit
- Target EV kit (in this case, MAX32660-EVSYS is used)
- CMSIS-DAP debug adaptor (MAX32625-PICO EV kit)
- Three USB Type-A to Micro-USB Type B cables
- Some female–female cables
- Windows PC (Windows 10) machine
- The latest Maxim SDK version must be installed on the PC. For details on the installation process, refer to the *Maxim Micro SDK Installation and Maintenance User Guide*.

Pin connection and final setup screen:

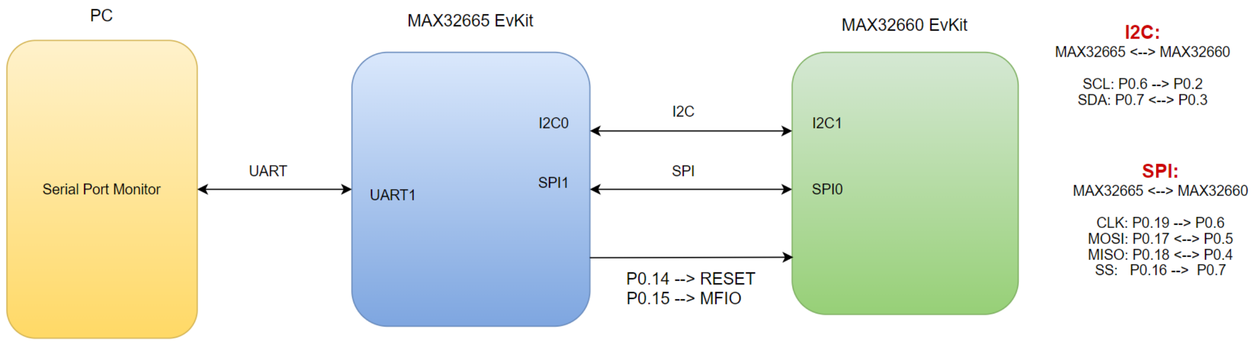


Figure 19. Bootloader host example setup block diagram

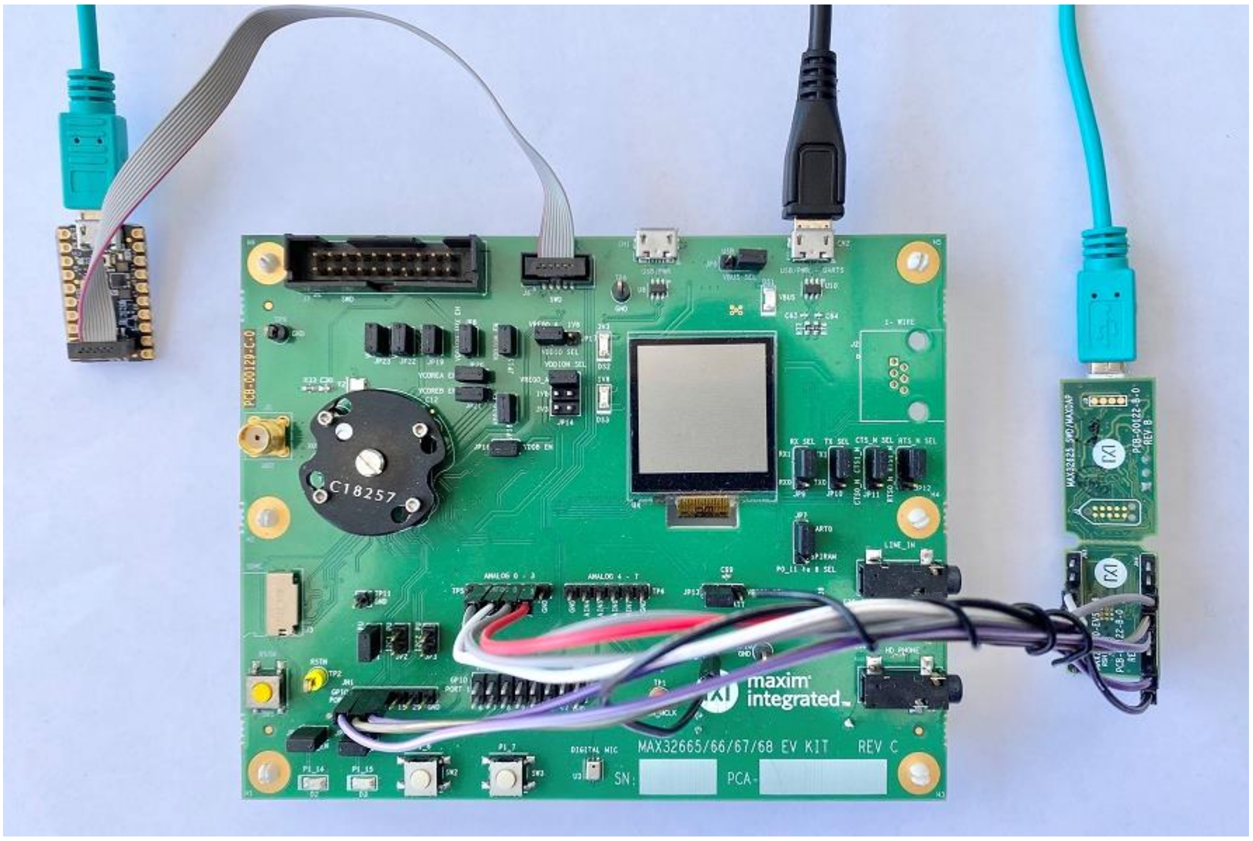


Figure 20. Bootloader host example full setup.

To create, build, and load bootloader host programmer example, following are the steps.

Creating Bootloader Host Example Project

- Run the Eclipse MaximSDK desktop application.

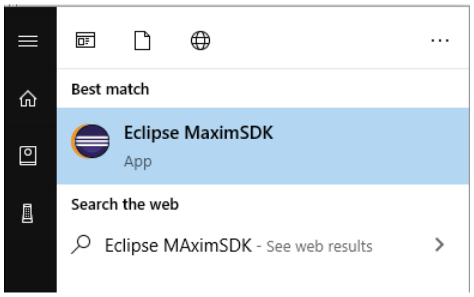


Figure 21. Starting Eclipse on the start menu.

- Select the workspace folder that contains the demo example and click **Launch**. Note that it is best to choose a path that contains no spaces.

- Start the wizard by clicking **File → New → Maxim Microcontrollers**.

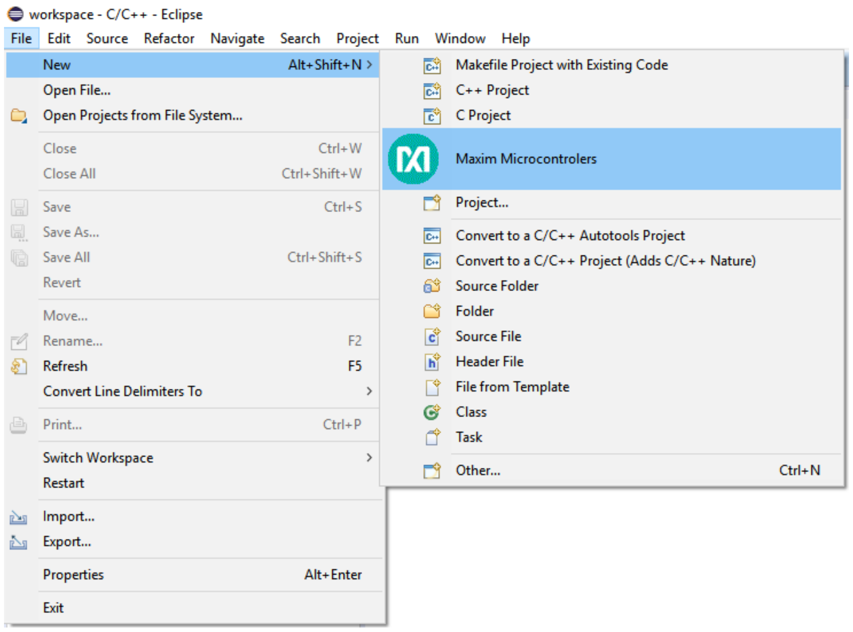
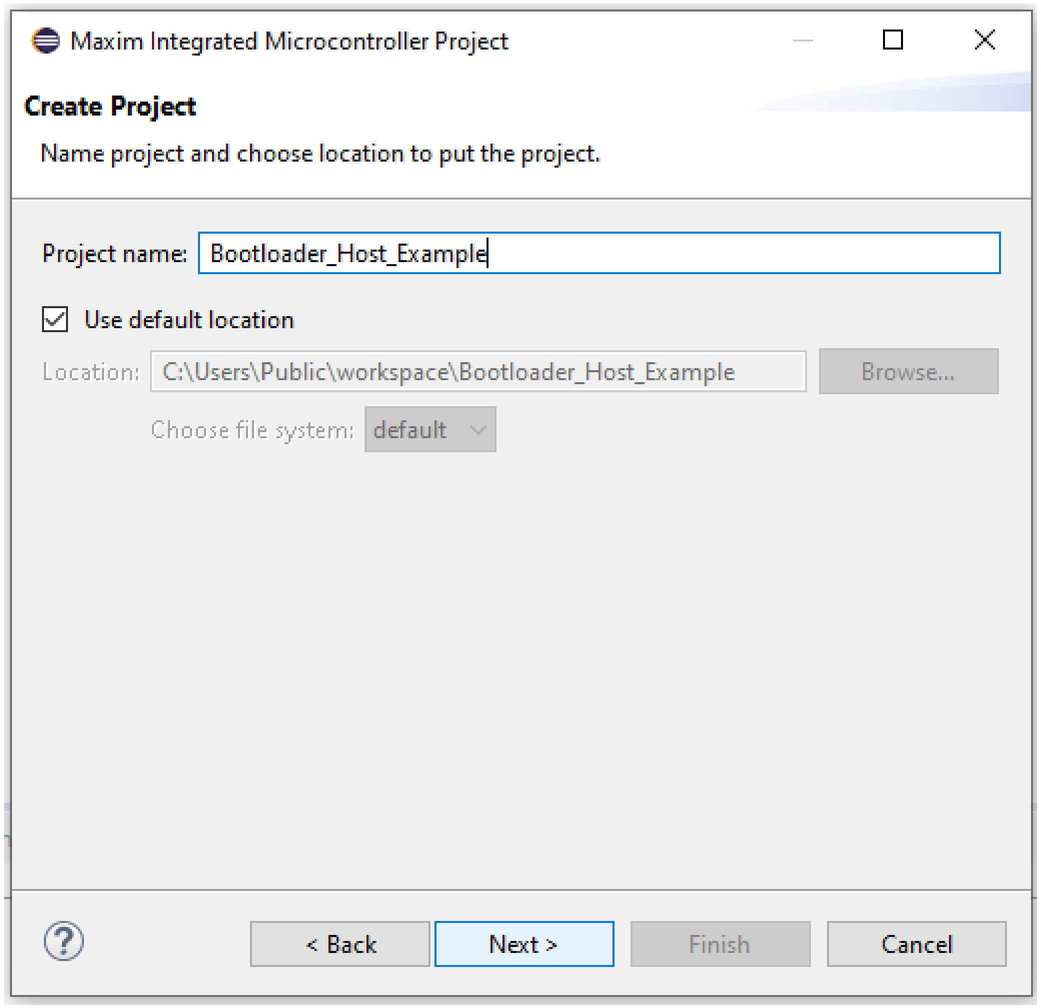


Figure 22. New project wizard.

- Enter the Project name and click **Next**.


Maxim Integrated Microcontroller Project

Create Project

Name project and choose location to put the project.

Project name: Bootloader_Host_Example

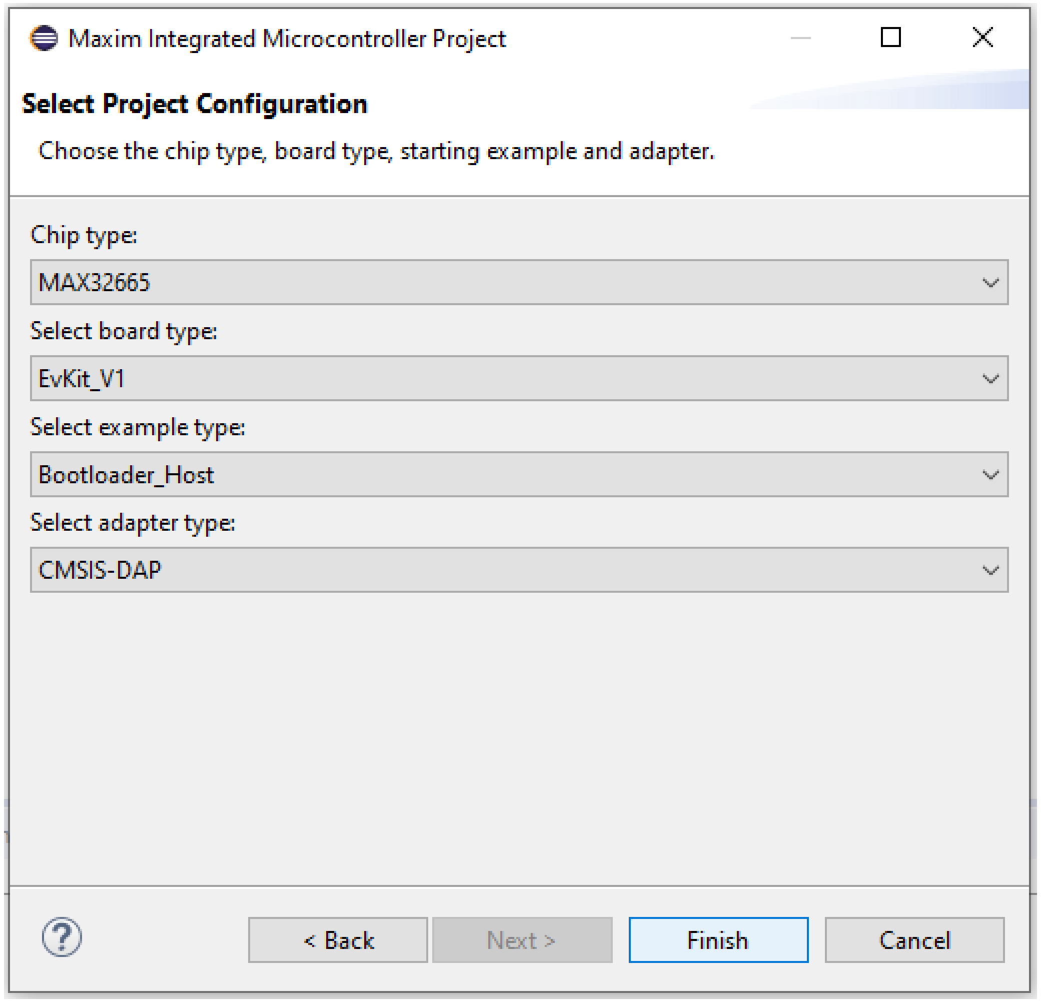
☑ Use default location

Location: C:\Users\Public\workspace\Bootloader_Host_Example

Choose file system: default

Figure 23. Set project name.

- Select MAX32665 as the Chip Type, EvKit_V1 as board type, Bootloader Host example as the example type, and adapter type (CMSIS-DAP in here) in the Select Project Configuration window.


Maxim Integrated Microcontroller Project

Select Project Configuration

Choose the chip type, board type, starting example and adapter.

Chip type: MAX32665

Select board type: EvKit_V1

Select example type: Bootloader_Host

Select adapter type: CMSIS-DAP

Figure 24. Select project configuration.

- Then, click the **Finish** button. The demo project is created.

Building and Loading an Example Project

To build the example project, right-click the project in Project Explorer and then click **Build Project**.

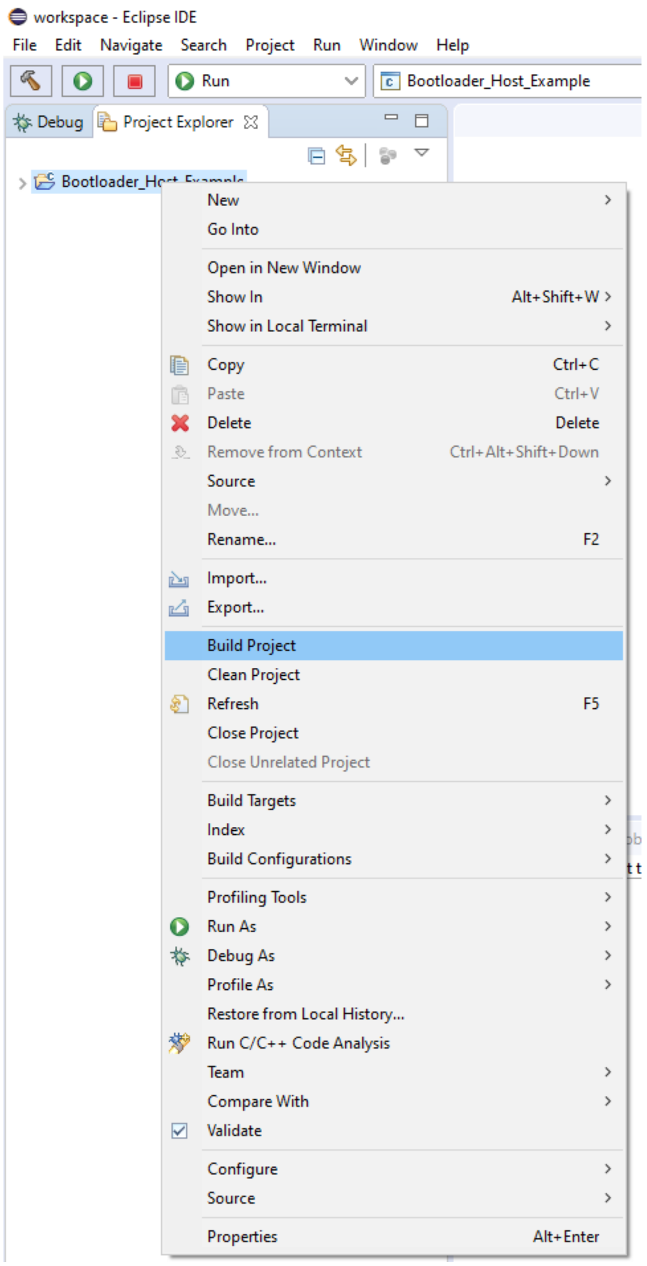
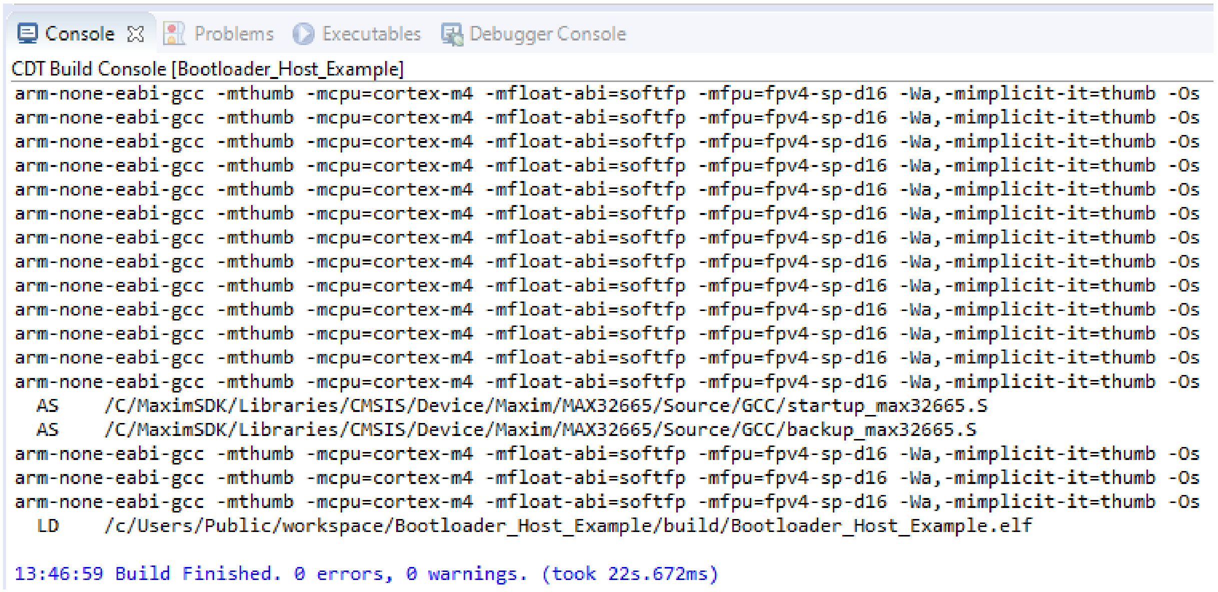


Figure 25. Build project.

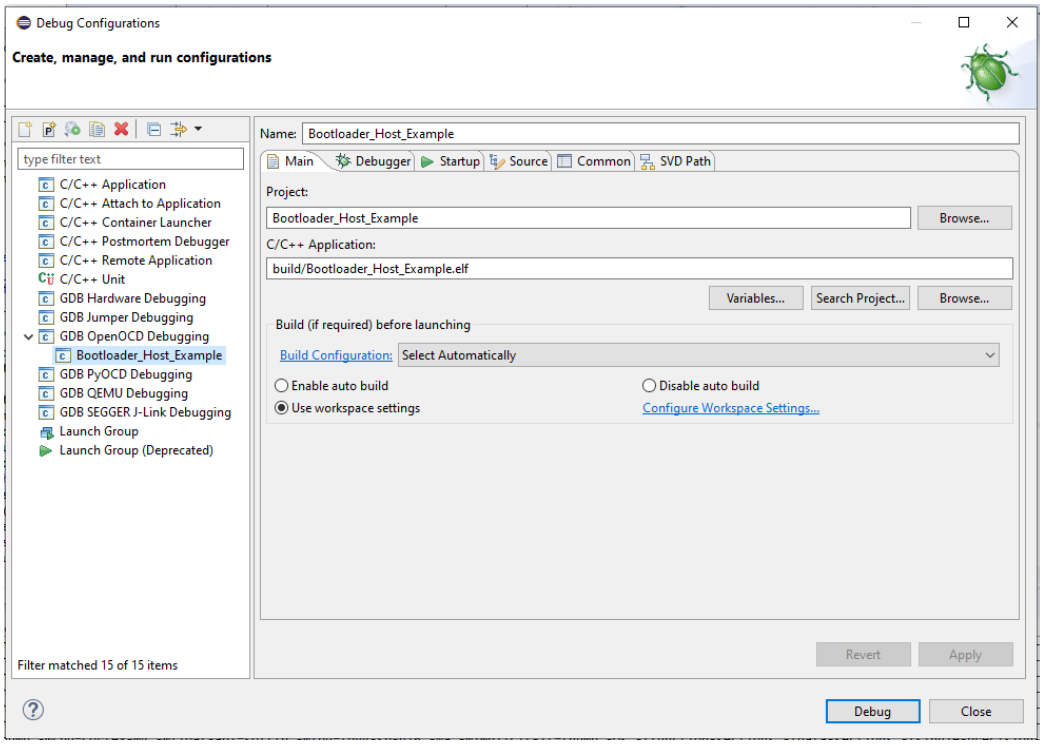
After the building is complete, check that the build has completed successfully, as shown in Figure 26.



```
CDT Build Console [Bootloader_Host_Example]
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
  AS    /C/MaximSDK/Libraries/CMSIS/Device/Maxim/MAX32665/Source/GCC/startup_max32665.S
  AS    /C/MaximSDK/Libraries/CMSIS/Device/Maxim/MAX32665/Source/GCC/backup_max32665.S
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
arm-none-eabi-gcc -mthumb -mcpu=cortex-m4 -mfloat-abi=softfp -mfpu=fpv4-sp-d16 -Wa,-mimplicit-it=thumb -Os
  LD    /c/Users/Public/workspace/Bootloader_Host_Example/build/Bootloader_Host_Example.elf

13:46:59 Build Finished. 0 errors, 0 warnings. (took 22s.672ms)
```

Figure 26. CDT build console output.

To load firmware on the board, right-click on the project and select **Debug As → Debug Configuration**, then select project and click the **Debug** button. The firmware is loaded in the MAX32665 device onto EV kit.



Name: Bootloader_Host_Example

Project: Bootloader_Host_Example

C/C++ Application: build/Bootloader_Host_Example.elf

Build (if required) before launching

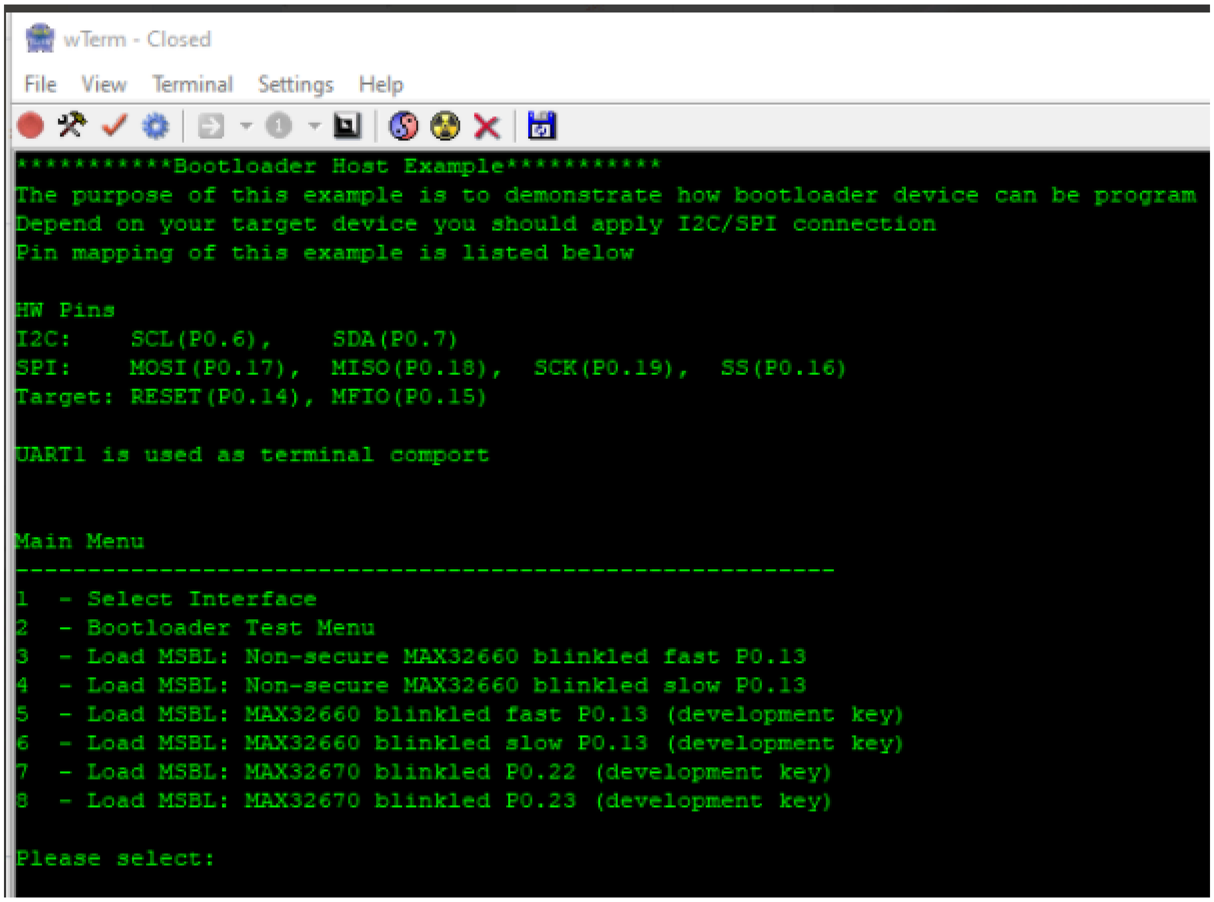
Build Configuration: Select Automatically

Enable auto build
Disable auto build
Use workspace settings [✓]
Configure Workspace Settings...

Figure 27. Loading binary on target EV kit.

Bootloader Host Example Menu

After preparing the setup, building Bootloader Host example then loading it on MAX32665 EV kit if you start to execute it, the menu (Figure 28) appears on the PC terminal side. It is an interactive example you can select to know which operation is executed. Select it by typing the number on the PC side keyboard.



```
***********Bootloader Host Example***********
The purpose of this example is to demonstrate how bootloader device can be program
Depend on your target device you should apply I2C/SPI connection
Pin mapping of this example is listed below

HW Pins
I2C:    SCL(P0.6),    SDA(P0.7)
SPI:    MOSI(P0.17),  MISO(P0.18),  SCK(P0.19),  SS(P0.16)
Target: RESET(P0.14), MFIO(P0.15)

UART1 is used as terminal comport


Main Menu
---------------------------------------------------------
1  - Select Interface
2  - Bootloader Test Menu
3  - Load MSBL: Non-secure MAX32660 blinkled fast P0.13
4  - Load MSBL: Non-secure MAX32660 blinkled slow P0.13
5  - Load MSBL: MAX32660 blinkled fast P0.13 (development key)
6  - Load MSBL: MAX32660 blinkled slow P0.13 (development key)
7  - Load MSBL: MAX32670 blinkled P0.22 (development key)
8  - Load MSBL: MAX32670 blinkled P0.23 (development key)

Please select:
```

Figure 28. Bootloader host programmer example console menu.

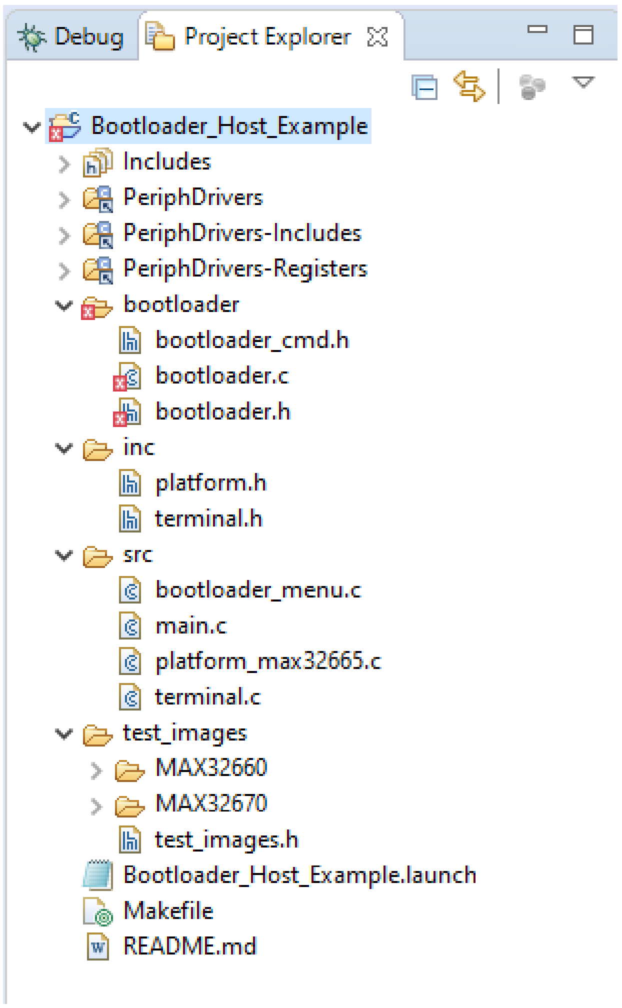
The host and target communication interface can be selected by the Select Interface menu.

Individual bootloader test commands can be tested by the Bootloader Test menu.

The example project includes some test .msbl images for the MAX32660 and MAX32670. These binaries can be loaded on the MAX32660 or MAX32670 target.

Porting Guide

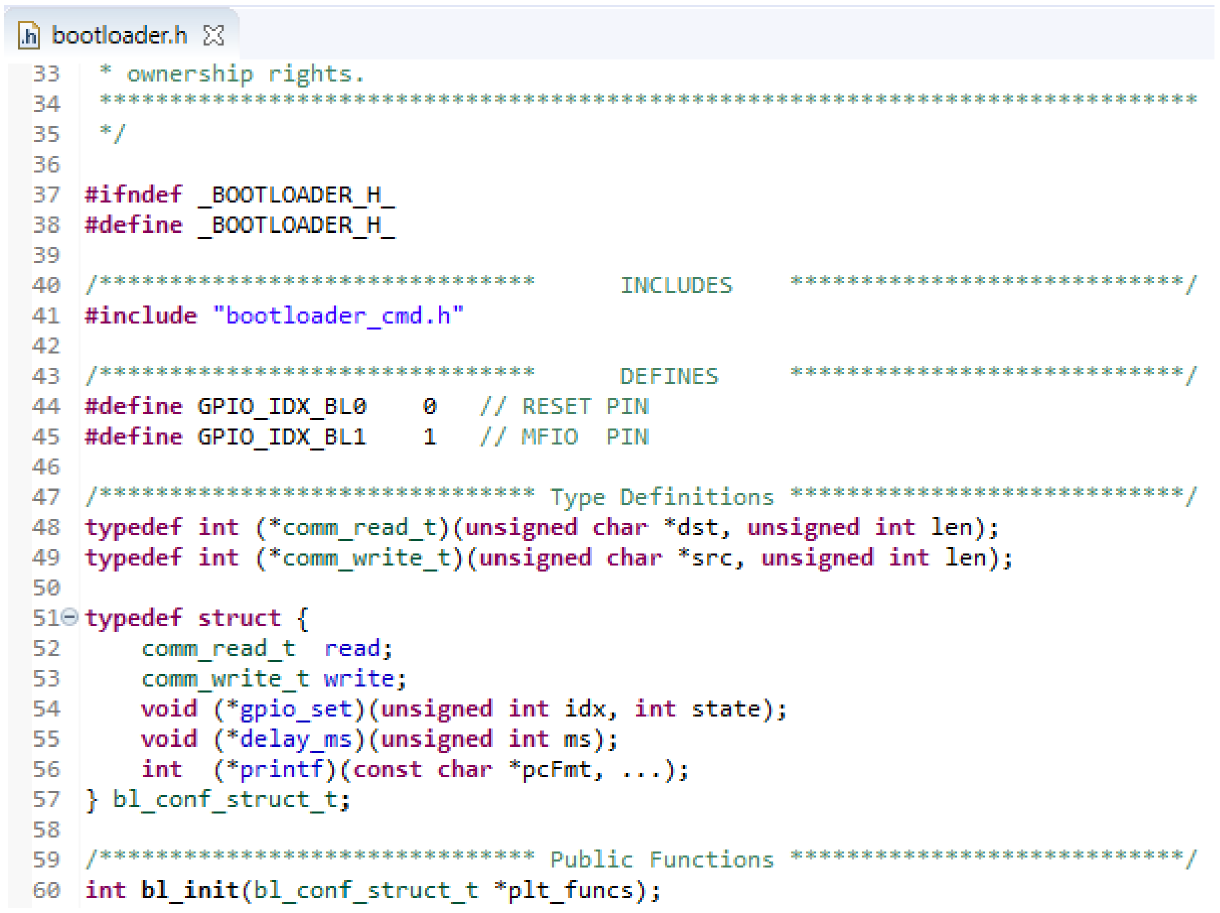
The bootloader host programmer source code is designed to be hardware-independent so it can be ported on any system. To demonstrate it, the source code is ported on the MAX32665 microcontroller.



```
Bootloader_Host_Example
    Includes
    PeriphDrivers
    PeriphDrivers-Includes
    PeriphDrivers-Registers
    bootloader
        bootloader_cmd.h
        bootloader.c
        bootloader.h
    inc
        platform.h
        terminal.h
    src
        bootloader_menu.c
        main.c
        platform_max32665.c
        terminal.c
    test_images
        MAX32660
        MAX32670
        test_images.h
    Bootloader_Host_Example.launch
    Makefile
    README.md
```

Figure 29. Bootloader host programmer example folders.

The files under the bootloader folder are platform-independent bootloader host programmer source code. These files (bootloader/bootloader.*) can be used as it is. The platform-related functions need to be registered during the initialization function call (bl_init function).



bootloader.h

```
 * ownership rights.
 *******************************************************************************
 */

#ifndef _BOOTLOADER_H_
#define _BOOTLOADER_H_

/********************************      INCLUDES    *****************************/
#include "bootloader_cmd.h"

/********************************      DEFINES     *****************************/
#define GPIO_IDX_BL0    0   // RESET PIN
#define GPIO_IDX_BL1    1   // MFIO  PIN

/******************************** Type Definitions ****************************/
typedef int (*comm_read_t)(unsigned char *dst, unsigned int len);
typedef int (*comm_write_t)(unsigned char *src, unsigned int len);

typedef struct {
    comm_read_t  read;
    comm_write_t write;
    void (*gpio_set)(unsigned int idx, int state);
    void (*delay_ms)(unsigned int ms);
    int  (*printf)(const char *pcFmt, ...);
} bl_conf_struct_t;

/******************************** Public Functions ****************************/
int bl_init(bl_conf_struct_t *plt_funcs);
```

Figure 30. Platform-independent bootloader layer initialization function.

The functions that need to be registered are as follows:

- read/write: I^{2}C, SPI, or UART read-write functions.
- gpio_set: The function that can set or clear GPIO state.
- delay_ms: milisecond delay function.
- printf: It is the debug purpose function.

After initialization (calling bl_init function) the other interface functions in bootloader.h file can be called.

Appendix A: Bootloader Programmer Board Protocol

Maxim bootloader programmer protocol is inserted inside DAPLink firmware to make it work as a bridge for I^2C/SPI and so that it is able to drive the predefined GPIO pins.

In idle mode, DAPLink firmware works in UART bridge mode, and the firmware checks data that gets from the PC side. If a magic command is received to the interface board firmware, it switches to bridge mode depending on the command.

The general bridge command format is as follows:

Magic (8 bytes) + Len (1 byte) + Group Code (1 byte) + Command (Len-1 bytes)

Magic word (8 bytes): DE EF AA 55 23 41 16 DC

Switch Mode Commands Detail

Table 1. Switch Mode Commands

GROUP CODE	COMMAND	DETAIL
B	I2C	Switch to I^2C bridge mode
	SPI	Switch to SPI bridge mode
	UART	Switch to UART bridge mode (Use current UART)
	UART0	Switch to UART bridge mode (Use UART0)
	UART2	Switch to UART bridge mode (Use UART2)

Request and response packet format are listed in the following figures:

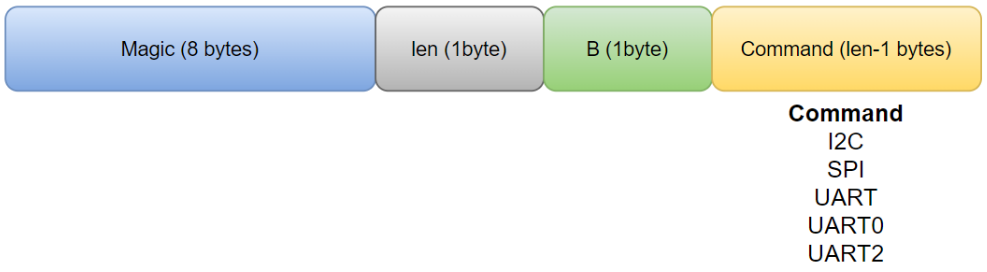


Figure 31. Request packet (from PC to interface board).

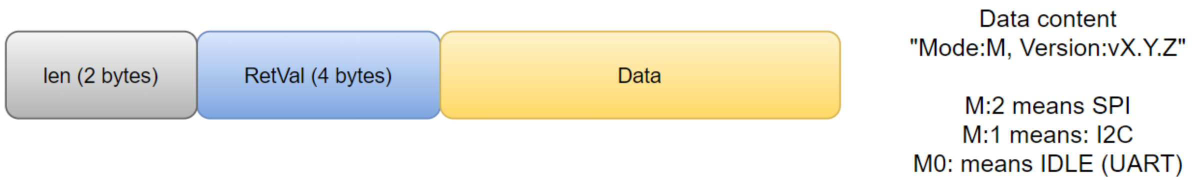


Figure 32. Response packet (from bootloader programmer board to PC).

On the switch-mode response command, the interface board sends its current status and firmware version.

RetVal is 4 bytes, 0 (zero) means SUCCESS nonzero means fail.

Request and response packet example is as follows:

- Request (hex): DE EF AA 55 23 41 16 DC 04 42 49 32 43
- Response: 00 1A 00 00 00 00 Mode:1, Version:v0.0.2

GPIO Commands

There are two general-purpose GPIOs on the interface board. These GPIOs can be set, clear, or read.

Two GPIOs are managed by 1 byte, which means for each GPIO 4 bits are used to set/clear/configure or read.

7-4 bits of 1 byte are used for GPIO1.

0-3 bits of 1 byte are used for GPIO0.

Table 2. GPIO Commands

GROUP CODE	COMMAND	DETAİL
G	0x00	Configure GPIOs padding GPIO padding can be: 0: Set pad to high impedance, weak pullup 1: Set pad to open-drain with high impedance input buffer enabled 2: Set pad to open-drain with weak pullup 3: Set pad to high impedance, input buffer enabled 4: Set pad to normal drive mode for high and low output 5: Set pad to slow-drive mode, which is the normal mode with negative feedback to slow edge transitions 6: Set pad to flash drive mode, which is the normal mode with a transistor drive to drive fast high, and low 7: Set pad to weak pulldown mode 8: Set pad to open-source mode, transistor drive to high 9: Set pad to open-source with weak pulldown mode, transistor drive to high, weak pulldown to GND for low Other values do not affect.
	0x01	Set/Clear GPIOs
	0x02	Get GPIOs status

Interface board response format (same for all bridge commands):

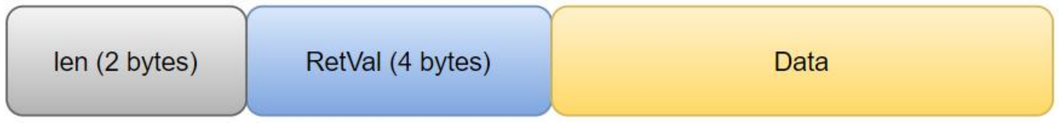


Figure 33. Adapter response for GPIO commands.

Data part does not exist for configure, set/clear response packet, it is 1 byte for getting the GPIO status command.

GPIO set/clear request packet should be as follows:

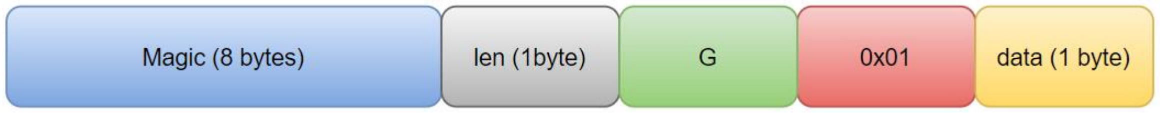


Figure 34. GPIO set/clear command request.

As mentioned earlier, each GPIO pin is managed by 4 bits; for set/clear case, if nibble is 0 it means clear and if it is 1 it means set. The 0x0F (15) means do not touch it, leave it as it is.

I^{2}C/SPI Bridge Mode Packet Format

After the interface board switches to bridge mode for I^{2}C or SPI, each request should be sent in the following format:

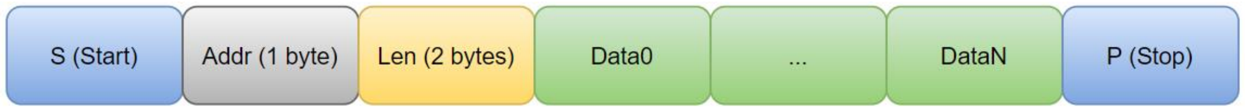


Figure 35. I^{2}C/SPI read/write request packet.

- Each request should start with 'S' and end with 'P' character.
- After 'S', address byte (I2C address) should be provided The LSB bit of address indicates read or write, 0:Write, 1:Read.
- After address bytes, 2 bytes length should be sent (big-endian format).

The read and write commands are the same; to read data from the target bootloader the LSB bit of address byte should be set to 1; and for read command length bytes indicate the number of bytes that will be read from bootloader.

Response packet format:

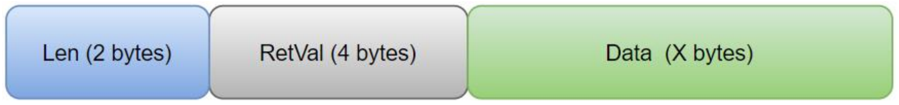


Figure 36. I^{2}C/SPI read/write response packet.

On response,

- Len is data len + 4 bytes (for RetVal),
- The RetVal is the number of data that read or write by adapter over I^{2}C or SPI.
- Data section valid for read command, for write request this section does not exist on response packet.

UART Bridge Mode Packet Format

There is no protocol for UART bridge mode and the adapter directly sends and receives the data that it read over UART.

Trademarks

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	10/21	Initial release	—