

# MCU Bootloader Demo Applications User's Guide

by: NXP Semiconductors

## 1 Introduction

This document describes how to use the MCU bootloader to load a user application on a Kinetis device.

## 2 Overview

This guide describes the steps required to use the NXP-provided MCU bootloader utilities to both load the MCU bootloader image and use the bootloader to update the user application section of flash. On reset, the bootloader detects the user application and launches it. Some development platforms support re-entry of the bootloader from a user application. This functionality enables application developers to easily install new applications onto Kinetis devices. It also provides manufacturers a way to update Kinetis devices in the field without a debugger.

### 2.1 MCU bootloader

The MCU bootloader serves as a standard bootloader for Kinetis devices. It provides a standard interface to the device via all of the available peripherals supported by a Kinetis device. The MCU bootloader interface is available in several forms like ROM, serial flashloader, or a customized flash-resident bootloader. Some Kinetis devices have ROM containing MCU bootloader while others are pre-programmed from the factory with a one-time-use serial flashloader. For a customized interface, customers can leverage the MCU bootloader source code to create a unique flash-resident bootloader that is both compatible with tools that understand the bootloader interface and are capable of supporting application-specific features. NXP provides utilities which demonstrates how to interface with the bootloader.

### 2.2 Host utility

The blhost.exe utility is a cross-platform host program used to interface with devices running the MCU bootloader. It lists and requests the execution of all the commands supported by a given Kinetis device running the bootloader.

## Contents

<b>1 Introduction.....</b>	<b>1</b>
<b>2 Overview.....</b>	<b>1</b>
2.1 MCU bootloader.....	1
2.2 Host utility.....	1
2.3 led_demo user application.....	2
2.4 Host updater.....	2
2.5 Toolchain requirement.....	2
<b>3 Getting Started with     MCU bootloader     application.....</b>	<b>2</b>
3.1 Plug it in.....	2
3.1.1 Select a board.....	3
3.1.2 Plug in the board.....	3
3.1.3 OpenSDA.....	3
3.1.4 PC Configuration.....	3
3.2 Get Software.....	4
3.2.1 Go to MCUXpresso SDK Builder.....	4
3.2.2 Build the SDK.....	4
3.2.3 Install a Toolchain....	4
3.3 Build and Run.....	4
3.3.1 Import the example.....	4
3.3.2 Build the imported project.....	7
3.3.3 Run blhost.....	9
<b>4 The host utility     application.....</b>	<b>11</b>
<b>5 Windows GUI     updater application...</b>	<b>12</b>
5.1 Installing the user application.....	12
<b>6 Returning to Flash-     resident bootloader..</b>	<b>15</b>



Check *blhost User's Guide* (document MCUBLHOSTUG) for detailed information.

## 2.3 led\_demo user application

The `led_demo_<freedom/tower/maps>_<base_address>` projects are example demo firmware applications used to demonstrate how the MCU bootloader can load and launch user applications. The demo projects can be found in `<sdk_package>/boards/<board>/bootloader_examples/demo_apps/`.

## 2.4 Host updater

The KinetisFlashTool.exe host application is a Windows® OS GUI program used to update the user application image on the device running the MCU bootloader firmware application. For more information on the Kinetis Flash Tool application, see the *Kinetis Flash Tool User's Guide* (document MBOOTFLTOOLUG).

## 2.5 Toolchain requirement

Firmware projects:

- IAR Embedded Workbench for ARM® v.8.30.1
- Python v.2.7 ([www.python.org](http://www.python.org))
- MCUXpresso IDE v10.3
- Keil MDK v5.26 with corresponding device packs

Host projects:

- Microsoft® Visual Studio® Professional 2015 for Windows® OS Desktop
- Microsoft® .NET Framework 4.5 (included in Windows OS 8)
- Microsoft® Visual C++ Redistributable for Visual Studio 2015 (vc redistrib\_x86.exe)
- Apple® Xcode v9.2 (for tools)
- GNU Compiler (GCC) v5.4.0 (for tools)

# 3 Getting Started with MCU bootloader application

## 3.1 Plug it in

<b>7 Appendix A - MCU flash-resident bootloader operation.....</b>	<b>15</b>
7.1 Memory map overview.....	16
7.2 User application vector table..	17
7.3 Bootloader Configuration Area (BCA).....	17
<b>8 Appendix B - MCU Bootloader Development platforms.....</b>	<b>18</b>
<b>9 Appendix C - MCU Bootloader Pin mappings.....</b>	<b>29</b>
<b>10 Revision history.....</b>	<b>40</b>

### 3.1.1 Select a board

- Go to [MCU Bootloader for Kinetis microcontroller](#) and select the desired Kinetis boards from list of "Supported Devices"
- Take FRDM-K64F as an example for demo



Figure 1. FRDM-K64F

### 3.1.2 Plug in the board

Connect the board to the PC via the USB cable. The cable will run between the OpenSDA USB port on the board and the USB connector on the PC.

### 3.1.3 OpenSDA

Kinetis MCU boards are supplied with pre-loaded OpenSDA firmware. For software development on the Kinetis board, make sure that the latest OpenSDA bootloader and Firmware application are on the development board. This allows debugging, flash programming, and serial communication over a USB cable.

Find the latest OpenSDA firmware for the boards here:

[OpenSDA Update to the boards](#)

For the example FRDM-K64, go to this website and choose board FRDM-K64 to get the latest OpenSDA firmware.

### 3.1.4 PC Configuration

The output data from the example applications is available over the MCU UART. To get this output, the virtual COM port driver for the board should be installed. The board must be plugged into the PC before the driver installer is run.

The links for the latest OpenSDA Windows serial port drivers are provided in the web page:

[OpenSDA Update to the boards](#)

For example, for FRDM-K64, click on the above URL and choose board FRDM-K64. In the next web page, click on ARM CMSIS-DAP serial drivers to download Windows Serial Port Drivers.

Once the Windows Serial Port Drivers are installed on the PC, the port number of the board's virtual COM port can be determined by checking the "Ports" under Device Manager.

#### NOTE

All the Serial Port Drivers are installed automatically when installing MCUXpresso IDE.

The demonstration board is now ready to communicate with the PC.

## 3.2 Get Software

### 3.2.1 Go to MCUXpresso SDK Builder

Click on [MCUXpresso SDK Builder](#). Select the desired development board and click on "Build MCUXpresso SDK".

In "SDK Builder" page, choose the desired Host OS and the IDE tool chain.

Click on "+ Add software component" button, select mcu-boot and middleware from drop-down list. USB stack middleware should be selected if it exists.

For this example, select board as FRDM-K64F, Host OS as Windows OS, IDE tool chain as MCUXpresso IDE and mcu-boot and USB stack as middleware.

### 3.2.2 Build the SDK

Click on **Download SDK** button. A pop-up window **SDK Downloads** will show when the package is built successfully. Click on **Download SDK Archive** to download the package. The package is provided as a zip file.

### 3.2.3 Install a Toolchain

NXP offers a complimentary toolchain called [MCUXpresso IDE](#).



Figure 2. MCUXpresso IDE

The MCU Bootloader includes support for other tools such as [IAR](#) and [Keil](#).

## 3.3 Build and Run

### 3.3.1 Import the example

- Install the downloaded FRDM-K64F SDK package as shown in the previous section, "Get Software". Then drag and drop the zipped SDK package as it is to the "Installed SDKs" view in MCUXpresso IDE.

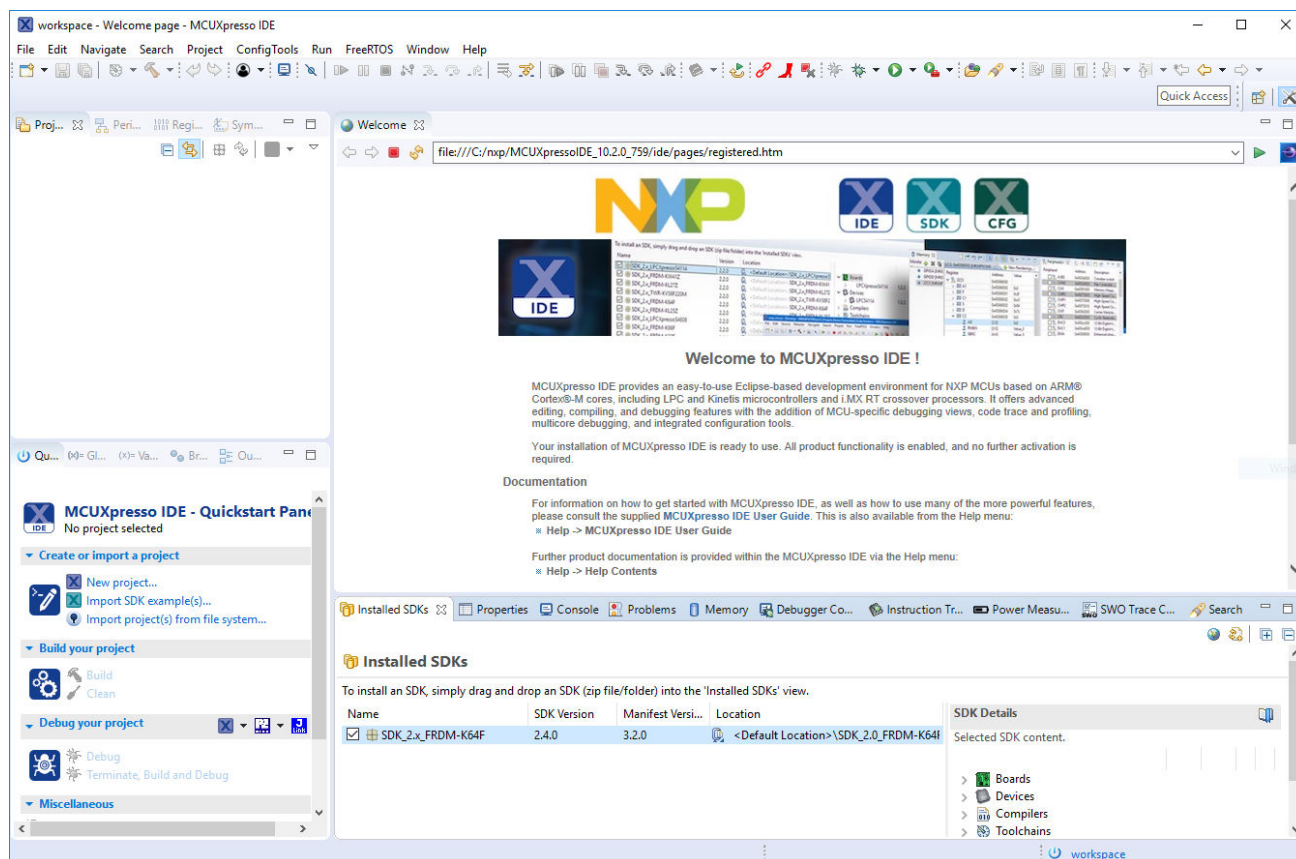


Figure 3. Install SDKs

Once the SDK installation is completed, click on 'Import SDK examples'.

- In the SDK import wizard, choose FRDM-K64F as the available board for your project.

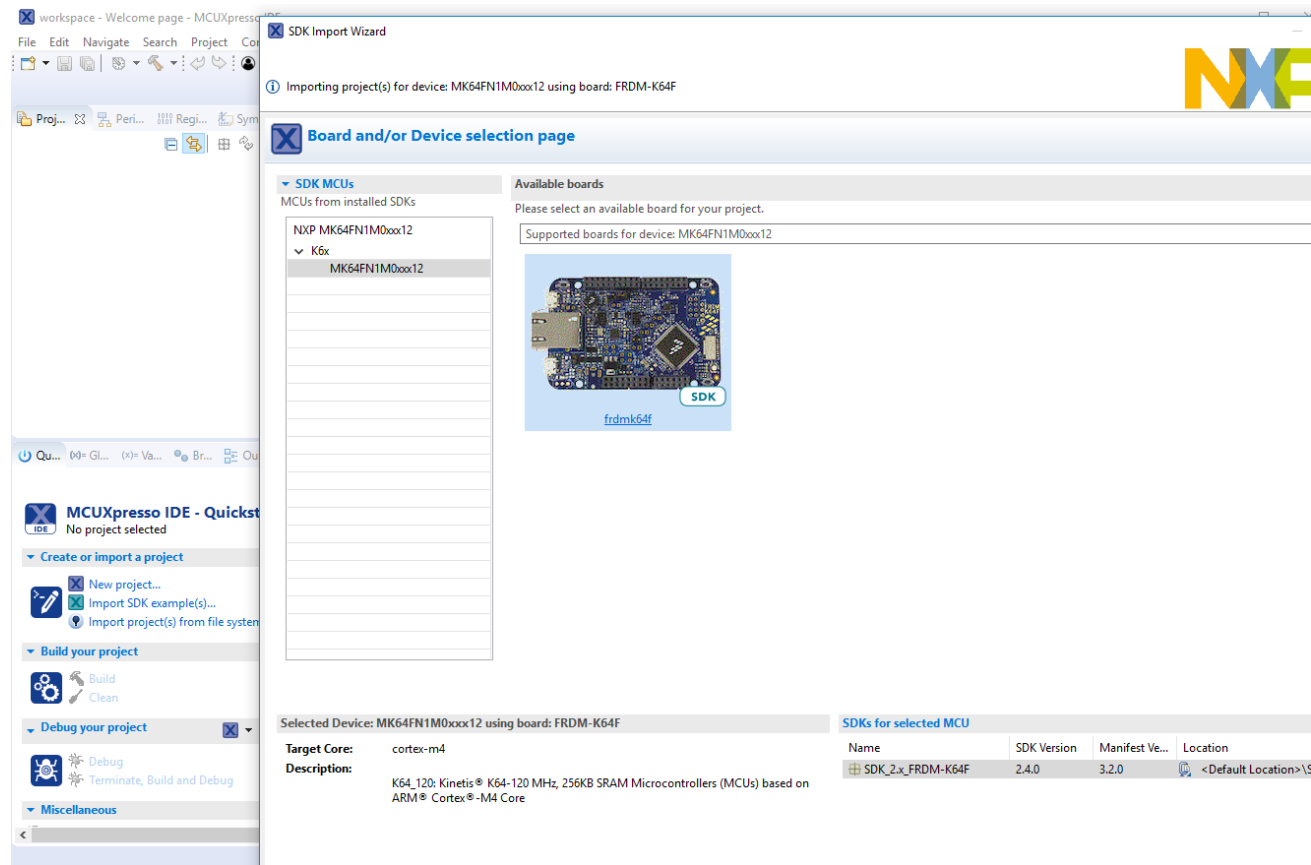


Figure 4. Board and/or Device selection page

- Import freedom\_bootloader example from the list of available projects.

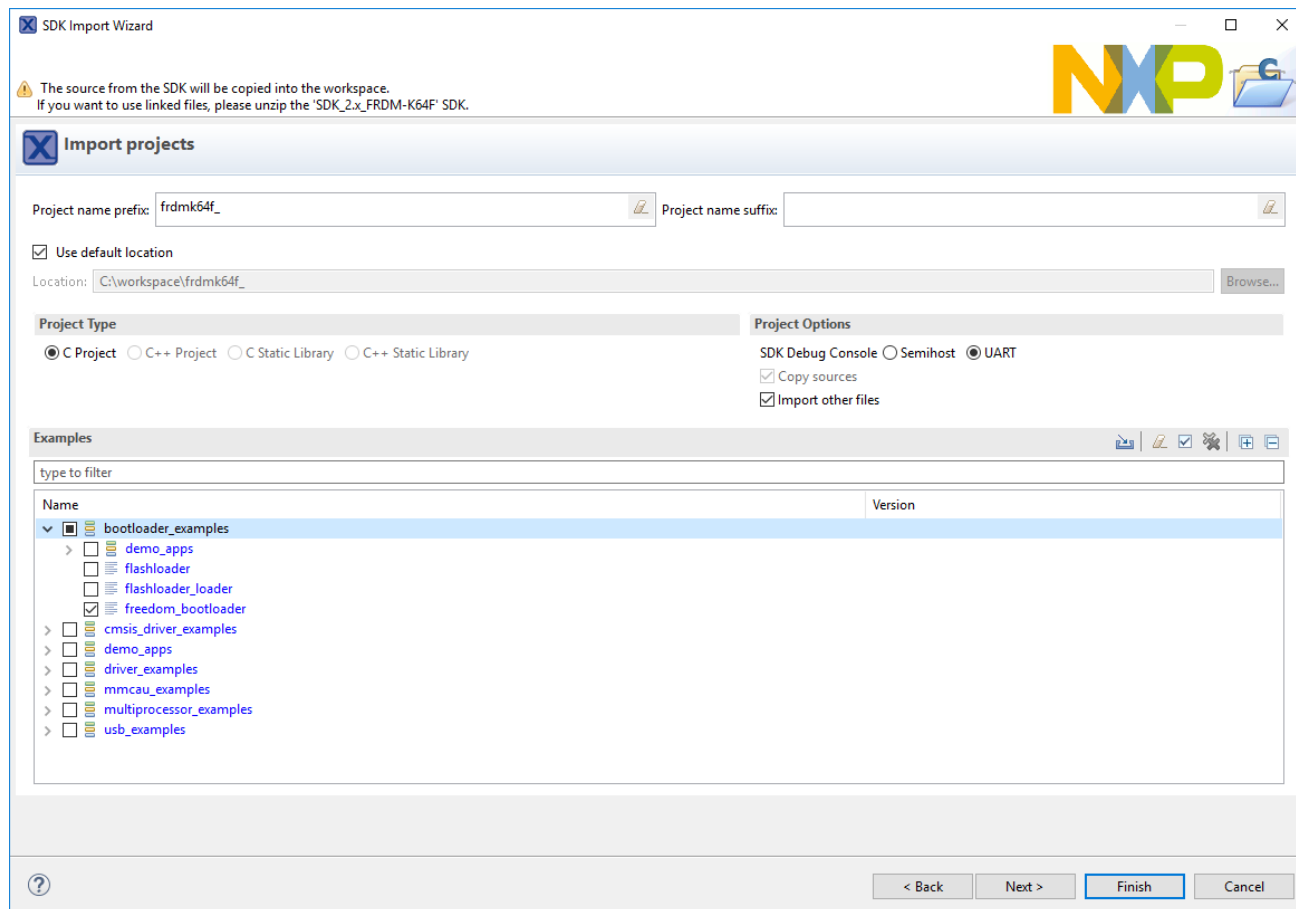


Figure 5. Import projects

**NOTE**

Once, the freedom\_bootloader example is imported successfully, the MCUXpresso IDE automatically creates the corresponding standalone project in your workspace.

### 3.3.2 Build the imported project

- Build the project by clicking "Build" button

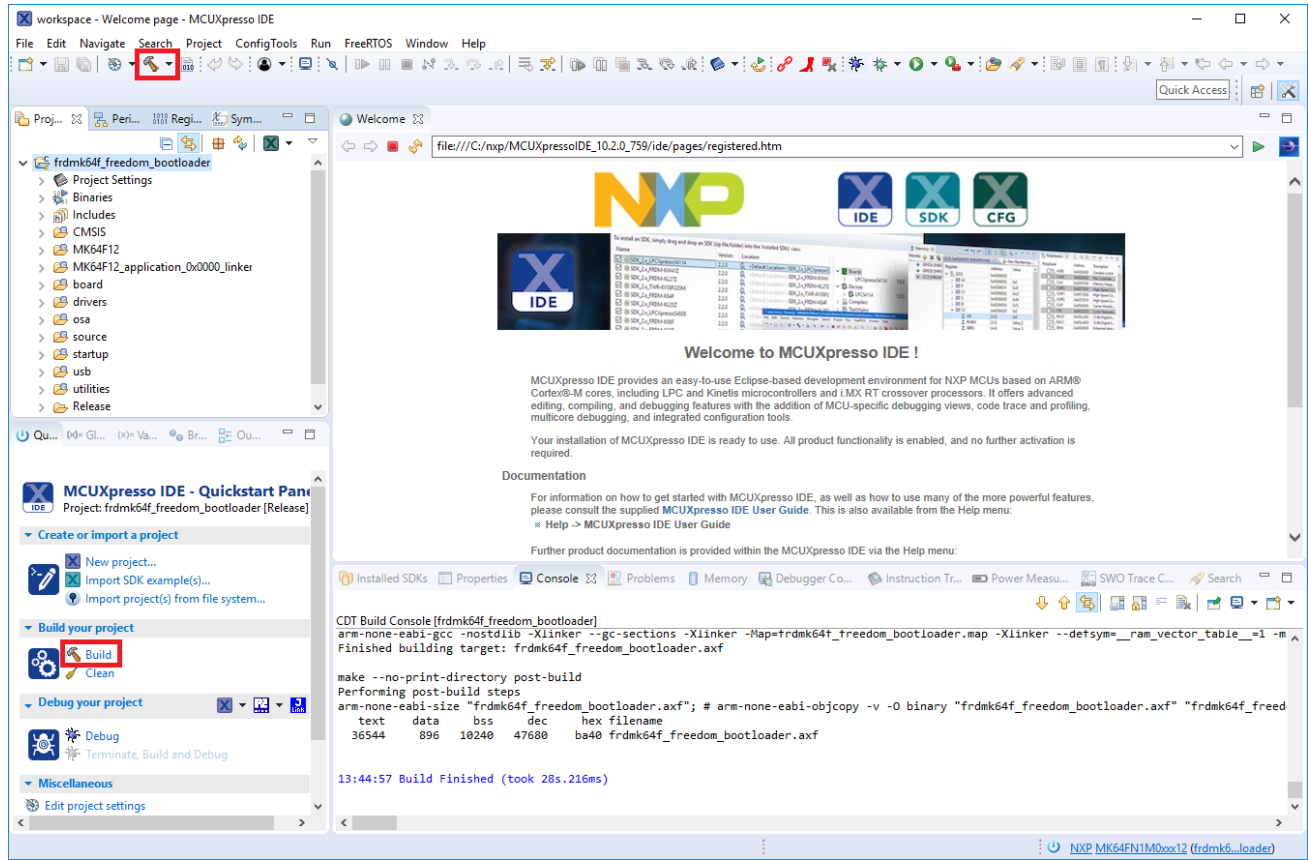


Figure 6. Build project

- Load the project by clicking on “Debug” button



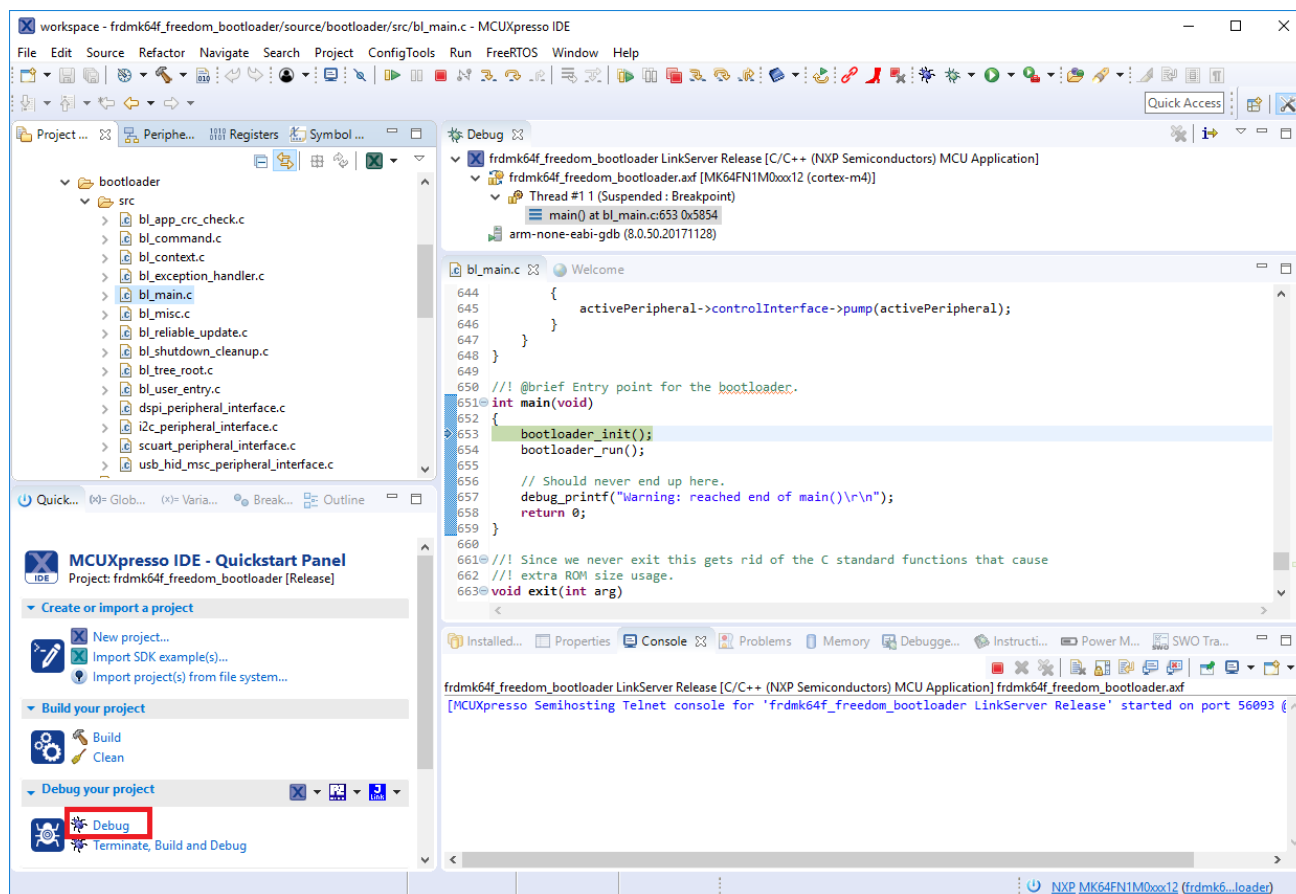
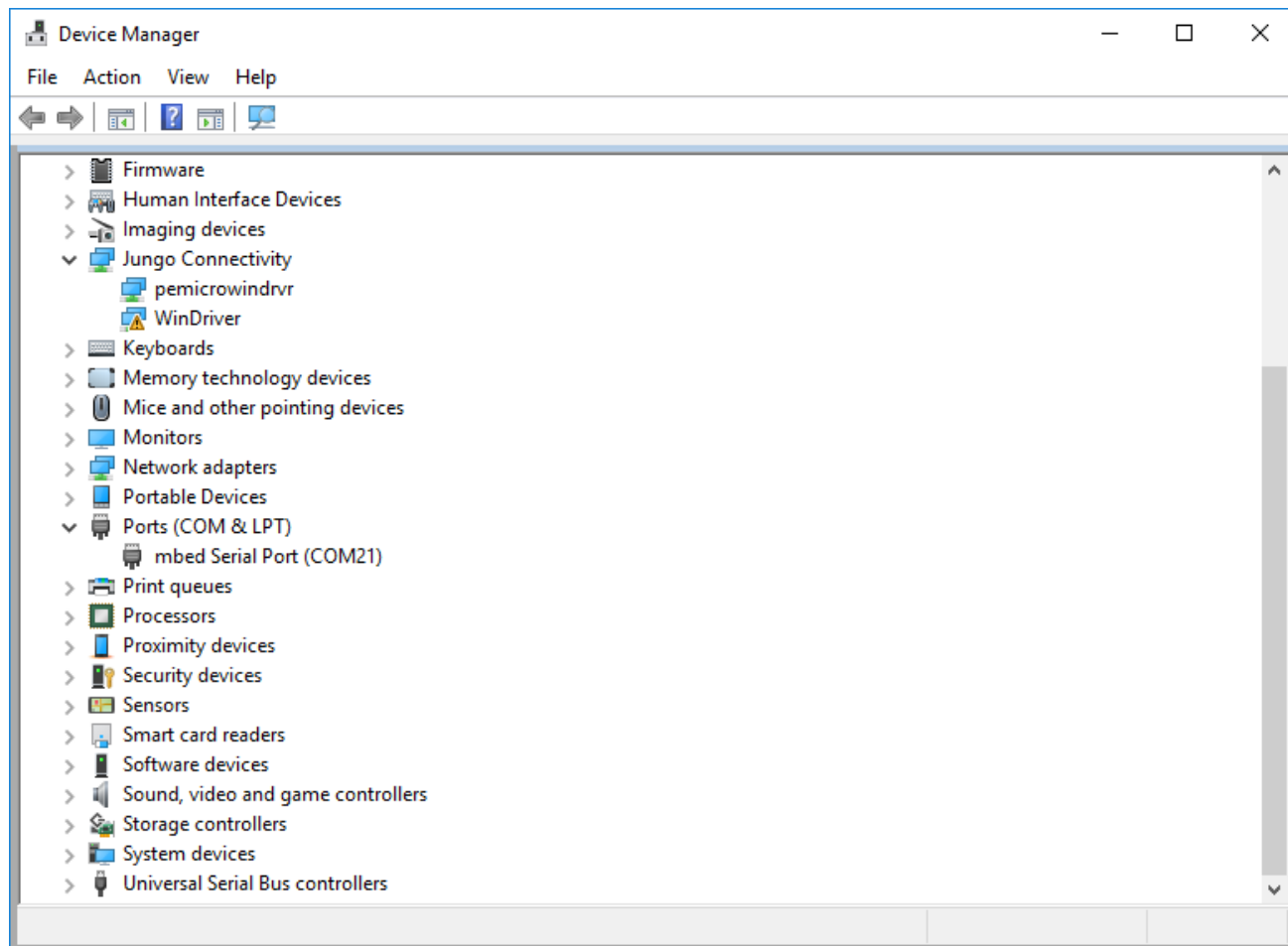


Figure 7. Debug project

### 3.3.3 Run blhost

- Determine the port number virtual COM port of the board. Click on "Ports" under the "Device Manager" to get this information.



**Figure 8. Device Manager**

- In downloaded package, Windows blhost is located at `middleware\mcu-boot\bin\Tools\blhost\win`.

Open a command prompt and run blhost to initiate communication and inject commands to the bootloader. For instance, bootloader can be connected over UART with option "-p". Run the blhost command get-property to get information from bootloader.

Type `blhost -p COM21 -- get-property 12` to get memory regions reserved by the bootloader

Type `blhost -p COM21 -- get-property 1` to get current bootloader version

See blhost User's Guide (Document ID: MCUBLHOSTUG) for additional information.

```

C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>blhost -p COM21 -- get-property 12
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 0 (0x0)
Response word 2 = 40959 (0x9fff)
Response word 3 = 536805376 (0x1fff0000)
Response word 4 = 536816639 (0x1fff2bff)
Reserved Regions =
    Region0: 0x0-0x9FFF (40 KB)
    Region1: 0x1FFF0000-0x1FFF2BFF (11 KB)

C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>blhost -p COM21 -- get-property 1
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258423808 (0x4b020600)
Current Version = K2.6.0

```

Figure 9. Command Prompt

## 4 The host utility application

This section describes simple use of the blhost host utility program to demonstrate communication with the MCU bootloader.

- Open a command prompt in the directory containing blhost. For Windows OS, blhost is located in <sdk\_package>/middleware/mcu-boot/bin/Tools/blhost/win. To open a command prompt, go to the Windows OS start menu and type "cmd" in the search box at the bottom of the window. Navigate to the blhost folder using change directory (CD) commands.
- Type *blhost --help* to see the complete usage of the blhost utility.

For this exercise, verify the Kinetis device is running the flash-resident bootloader.

- Click the "Reset" button on the platform.
- Determine the COM port that the platform is connected to. For this example, the device is connected to COM21.
- Type *blhost -p COM21 -- get-property 1* to get the bootloader version from the MCU bootloader.
- The screen shot below indicates that blhost.exe is successfully communicating with the MCU bootloader on the platform.

```

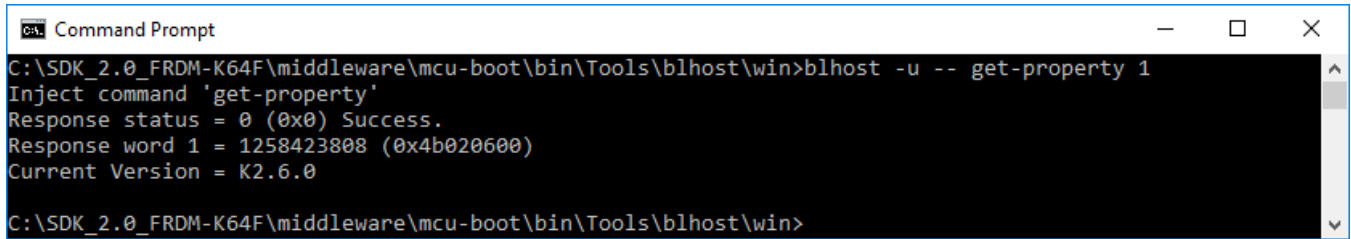
C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>blhost -p COM21 -- get-property 1
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258423808 (0x4b020600)
Current Version = K2.6.0

```

Figure 10. Host communication with MCU bootloader

For communicating with MCU bootloader using USB:

- Press the "Reset" button on the platform.
- Type *blhost -u -- get-property 1* to get the bootloader version from the MCU bootloader.
- The screen shot below indicates that blhost.exe is successfully communicating with the MCU bootloader on the platform.



```
C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>blhost -u -- get-property 1
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258423808 (0x4b020600)
Current Version = K2.6.0
C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>
```

Figure 11. Host communication with MCU bootloader using the USB peripheral

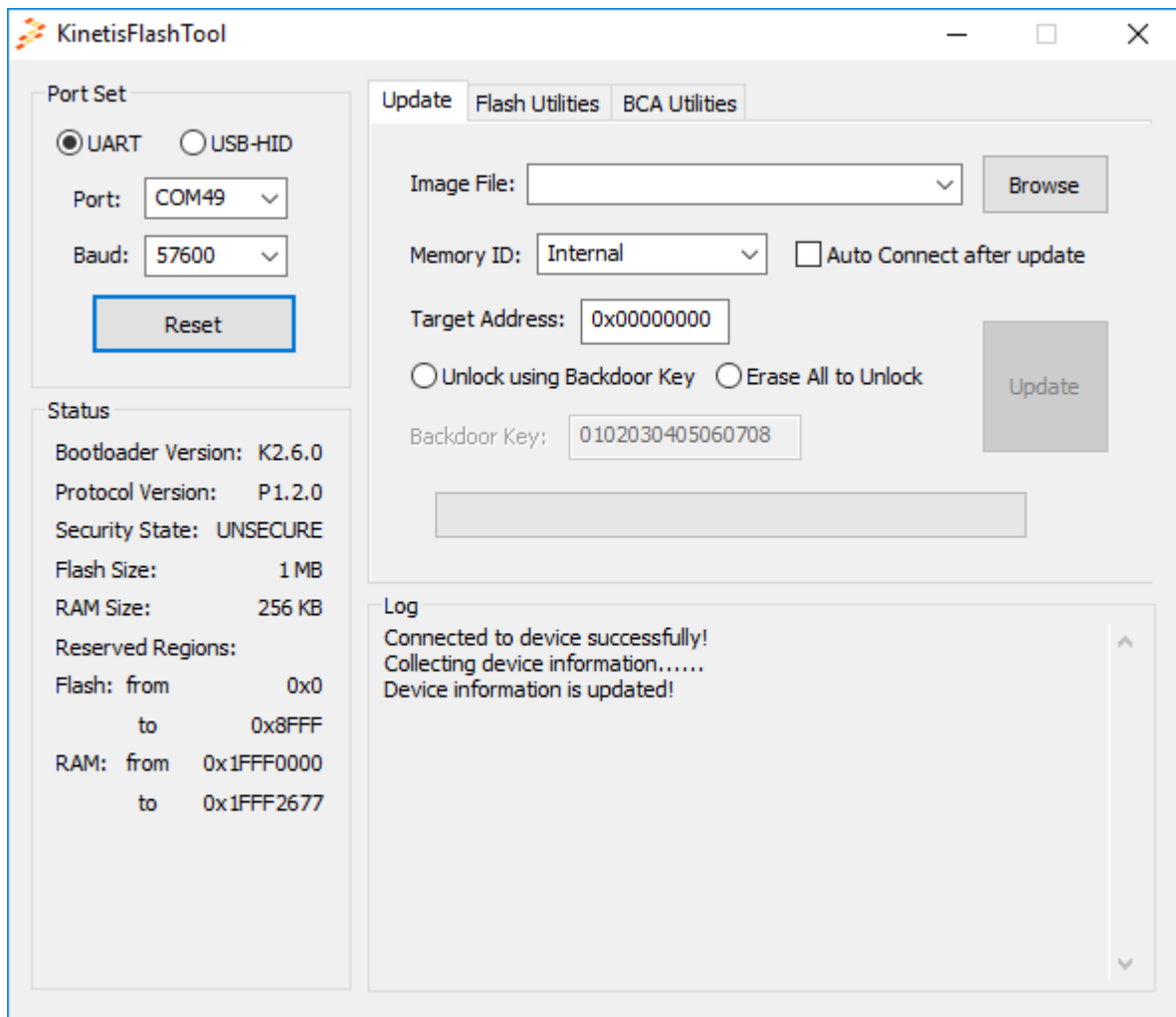
## 5 Windows GUI updater application

This section describes how to use the Windows GUI updater application, KinetisFlashTool.exe, to install an example user application onto the platform.

### 5.1 Installing the user application

The FRDM-K64F platform is used in this example. Similar steps can be used for other development platforms.

1. Click the "Reset" button on the platform.
2. Navigate to <sdk\_package>/middleware/mcu-boot/bin/Tools/KinetisFlashTool/win directory.
3. Double-click the KinetisFlashTool.exe file to launch the application.
4. Connect the device.
  - Select the COM49 device from the "Port" drop-down box.
  - Click the "Connect" button.



**Figure 12. Connect the device**

5. Select the image file.

- Select the led\_demo\_freedom\_a000.bin application image using the "Browse" button.
- Set the base address to 0xA000.

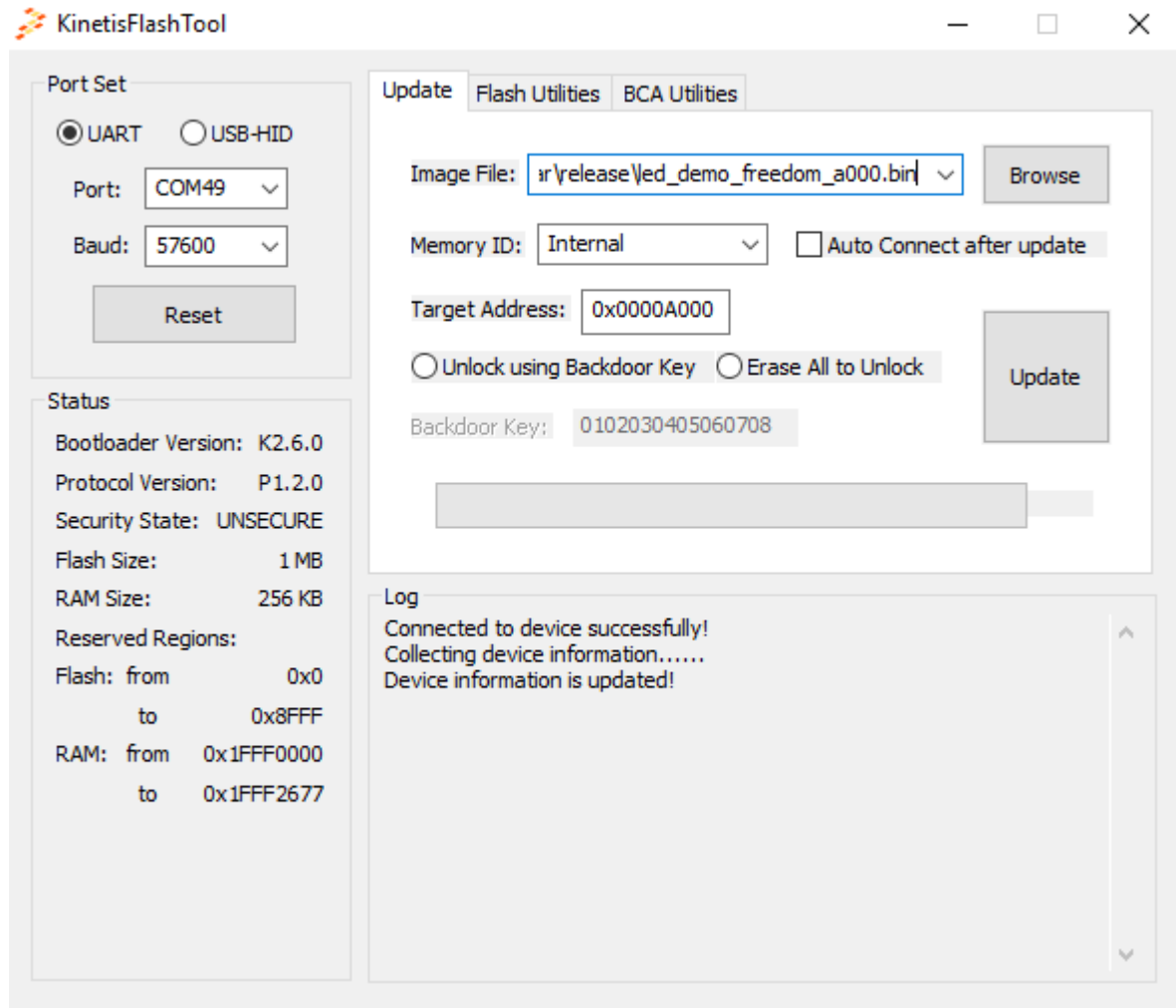


Figure 13. Browse for the user application

6. Update the image.
  - Click the "Update" button to write the application image to the device flash.
  - Wait for the application to start. The waiting time is determined by the timeout parameter.

```

C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>blhost -p COM21 -- get-property 1
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258423808 (0x4b020600)
Current Version = K2.6.0
  
```

Figure 14. Perform the update

7. At this point, the led(s) on the target board is noticeably blinking indicating that the MCU bootloader successfully installed the led\_demo user application.
8. Reprogram the device without exiting the application if you re-enter the bootloader by pressing the boot pin button (see Appendix B to determine if the platform has a boot pin button) and resetting the board.
9. Click the "Close" button at the top right corner when finished.

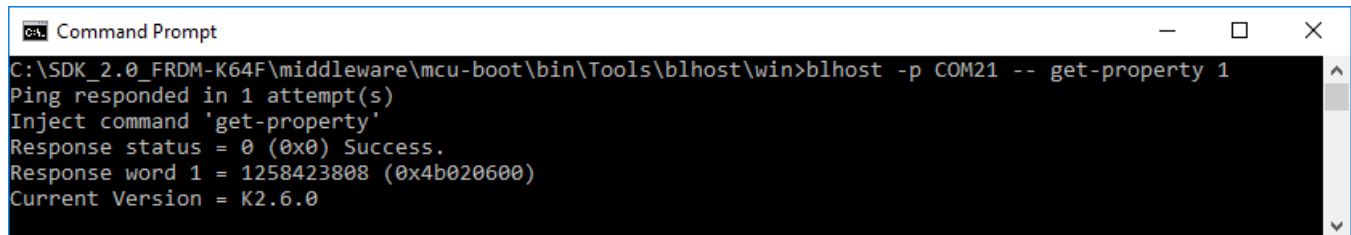
## 6 Returning to Flash-resident bootloader

Some development platforms support re-entry of the bootloader from a user application. See Appendix B to determine if the board has a "Boot Pin" button listed.

### NOTE

If the MCU on the platform has a built-in bootloader in ROM, the boot pin returns to the ROM, not to the flash-resident bootloader. The following platforms have the bootloader in ROM: MK80, MK82, MKL28, and MKL82.

To return to the MCU bootloader interface, hold the "Boot Pin" button and press and release the "Reset" button on the target board. When the device resets, the MCU bootloader detects the press on the boot pin and does not jump to the user application. Verify the bootloader mode by running the blhost.exe tool again.



```

C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>blhost -p COM21 -- get-property 1
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258423808 (0x4b020600)
Current Version = K2.6.0
  
```

**Figure 15. Back to the MCU bootloader interface**

Pressing the "Reset" button allows the MCU Bootloader to again launch the led\_demo application.

## 7 Appendix A - MCU flash-resident bootloader operation

This section describes the linkage between the MCU flash-resident bootloader and the user application. The demonstration described above illustrates a simple collaboration between the MCU bootloader and the led\_demo application. The considerations are:

- The flash-resident bootloader is located in flash at address 0.
- The user application is located in flash above the bootloader at BL\_APP\_VECTOR\_TABLE\_ADDRESS
- The vector table for the User Application is placed at the beginning of the application image.
- The Bootloader Configuration Area (BCA) is placed at 0x3C0 from the beginning of the image.

### NOTE

The base address of a user application with a flash-resident bootloader is different than the application base address when using a ROM-based bootloader. The application linker file needs to be updated to link the image to the correct base address. In addition, the application vector table also needs to be updated based on the correct application location.

## 7.1 Memory map overview

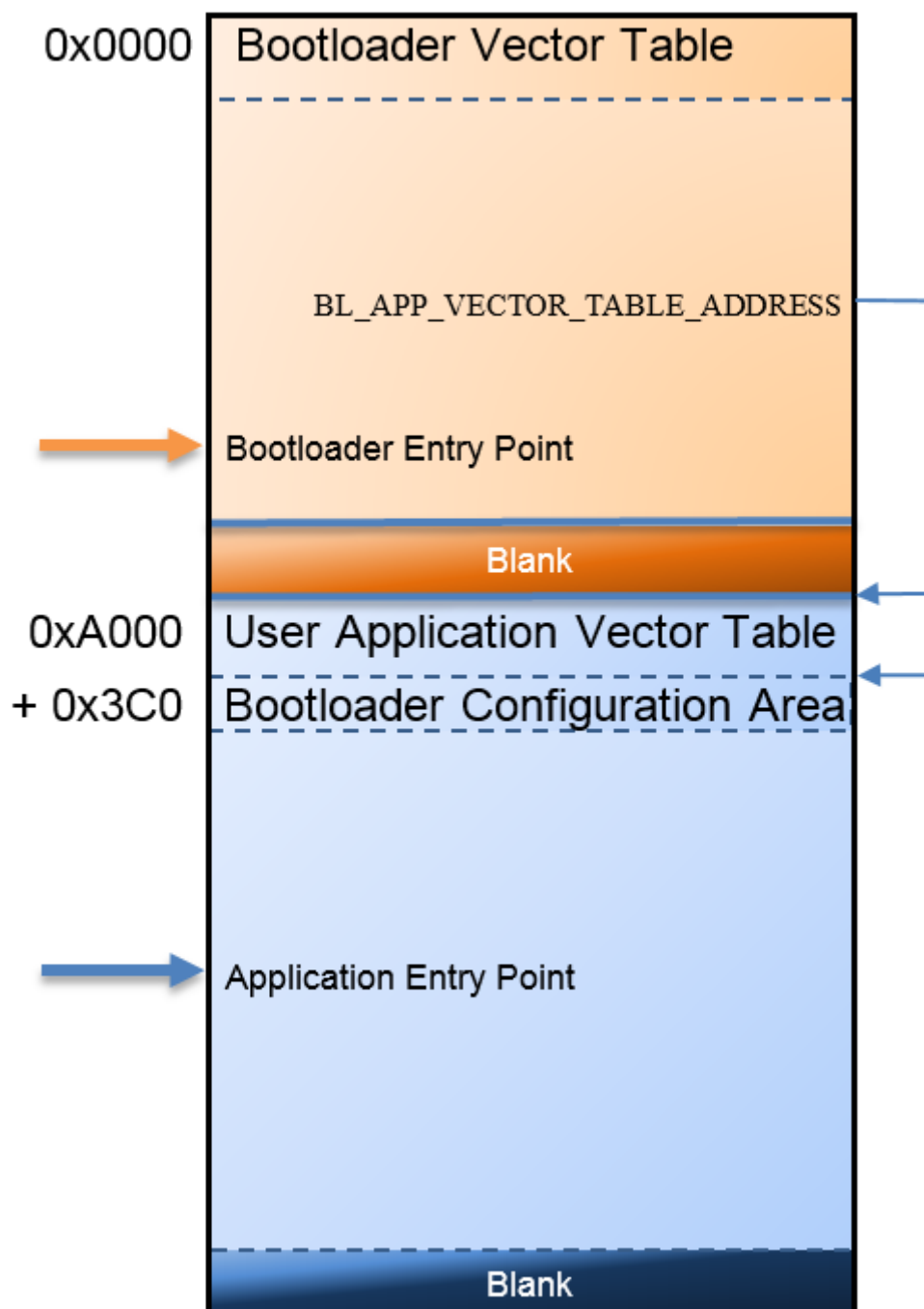


Figure 16. Device memory map



## User Application Image Start at BL\_APP\_VECTOR\_TABLE\_ADDRESS

offset 0x000	stackPointer			
	entryPoint			
	...			
offset 0x3C0	tag = 'kcfg'			
	crcStartAddress			
	crcByteCount			
	crcExpectedValue			
	enabledPeripherals	i2cSlaveAddress	peripheralDetectionTimeoutMs	
	usbVid		usbPid	
	usbStringsPointer			
	clockFlags	clockDivider	bootFlags	reserved
	mmcauConfigPointer			
	keyBlobPointer			
	reserved	canConfig1	canConfig2	
	canTxId		canRxId	

Figure 17. User application vector table and Bootloader Configuration Area (BCA)

## 7.2 User application vector table

The MCU bootloader checks BL\_APP\_VECTOR\_TABLE\_ADDRESS+0 for the User Application stack pointer and BL\_APP\_VECTOR\_TABLE\_ADDRESS+4 for the User Application entry point. Initially, this area is erased (0xFF) and the bootloader remains in its command interface.

After a user application is installed to BL\_APP\_VECTOR\_TABLE\_ADDRESS, the bootloader jumps to the application after a period specified by peripheralDetectionTimeoutMs in the Bootloader Configuration Area (BCA).

## 7.3 Bootloader Configuration Area (BCA)

The Bootloader Configuration Area is located at offset 0x3C0 from the beginning of the User Application image. This information is read by the MCU bootloader early during the bootloader initialization to set up clocks and gather other information relevant to detecting active peripherals. If the first four bytes of the BCA are not 'kcfg', the bootloader does not use any information from the BCA on flash.

## 8 Appendix B - MCU Bootloader Development platforms

All boards must be in their default factory state for jumper settings.

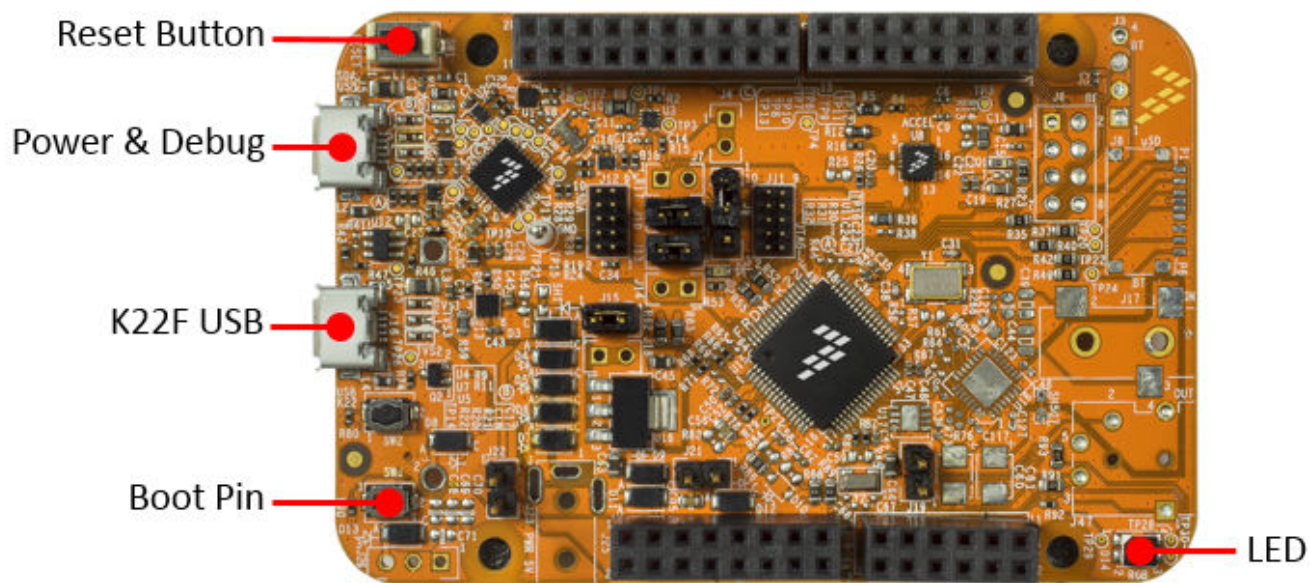


Figure 18. FRDM-K22F platform

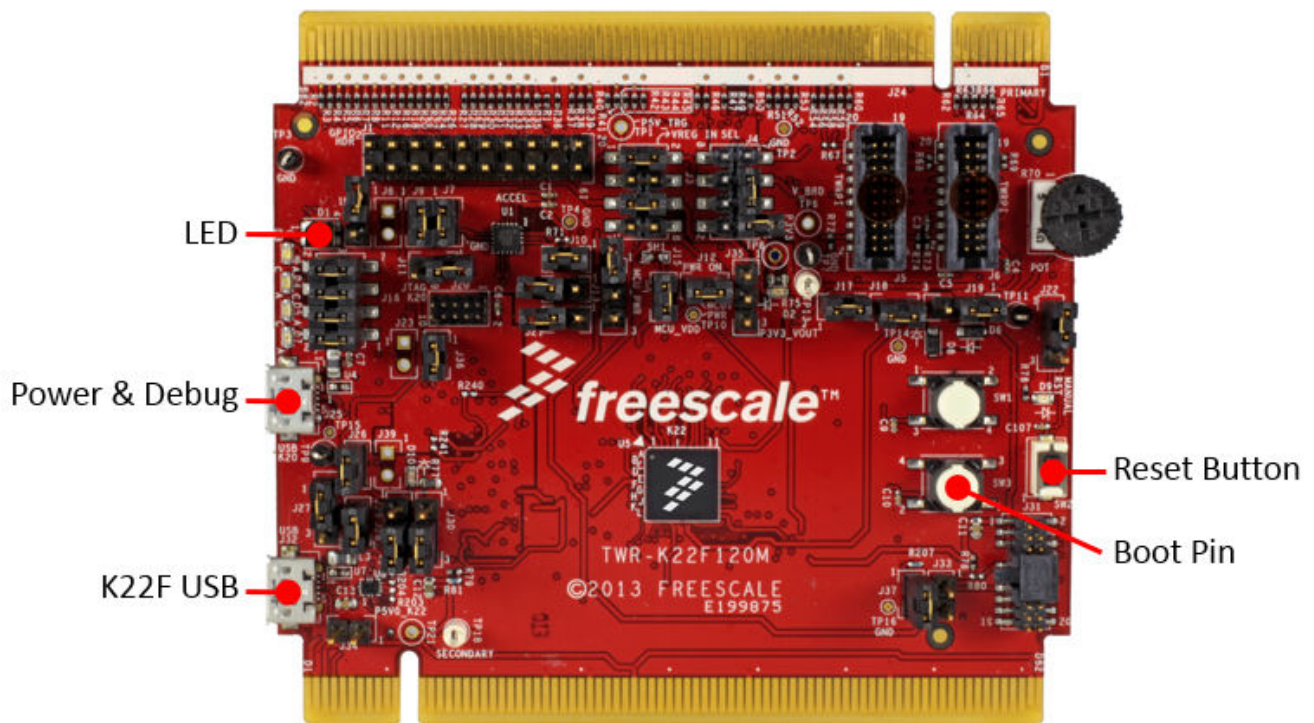


Figure 19. TWR-K22F120M platform



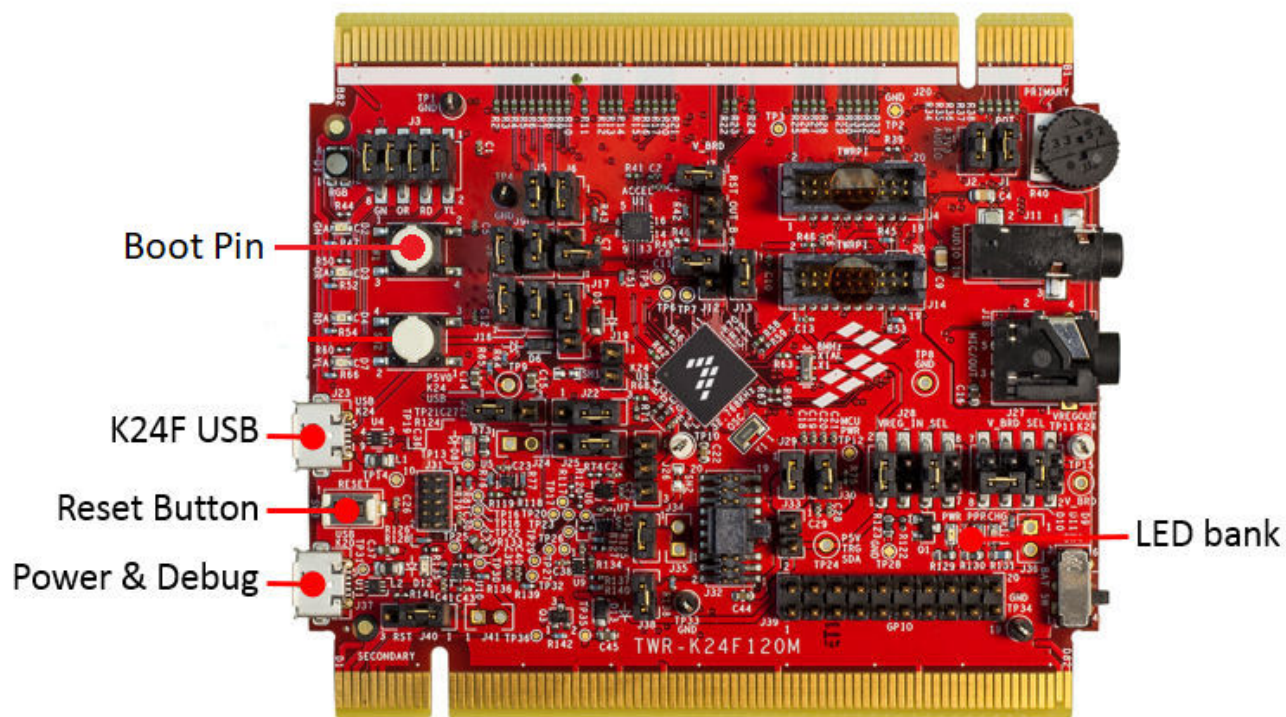


Figure 20. TWR-K24F120M platform

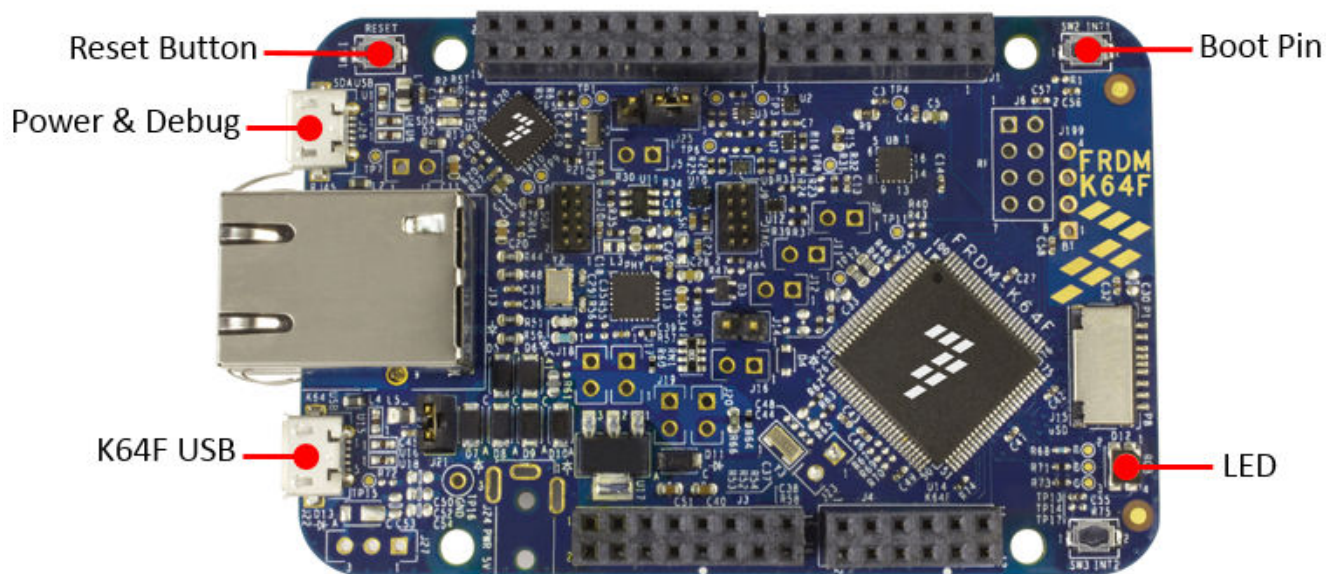


Figure 21. FRDM-K64F platform

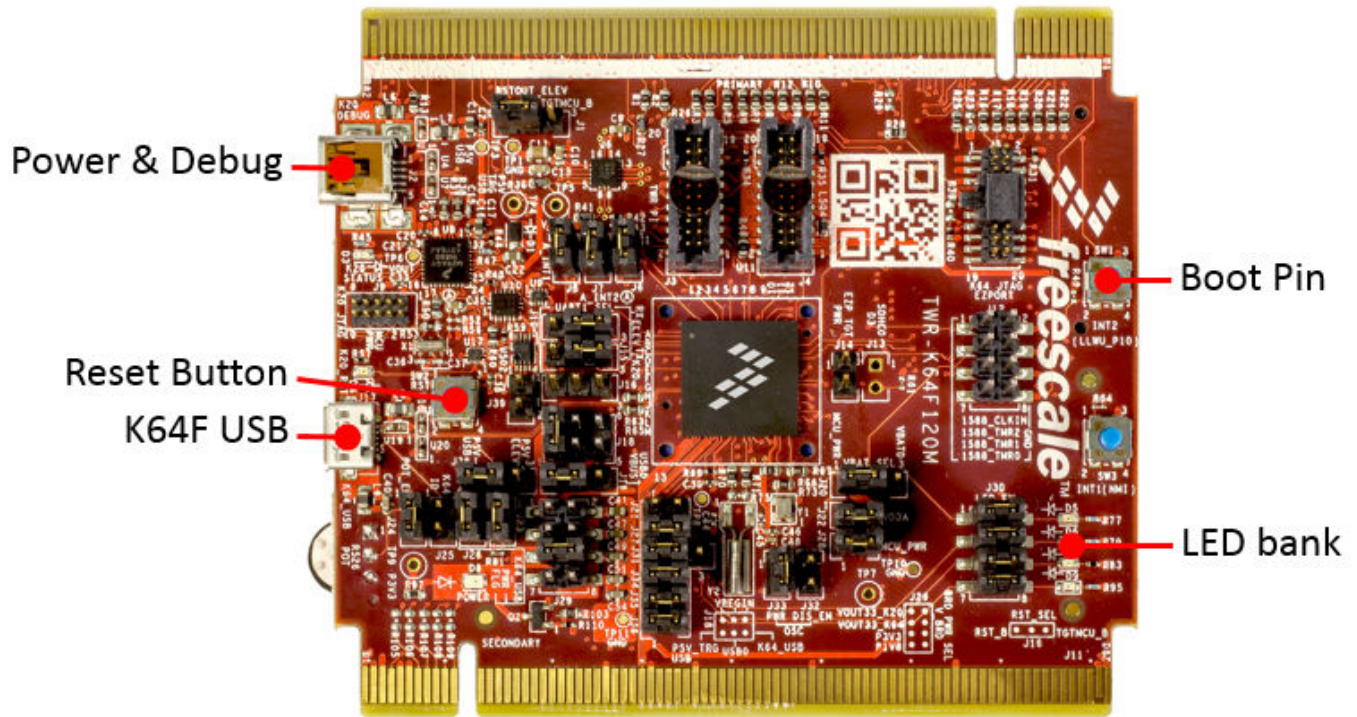


Figure 22. TWR-K64F120M platform



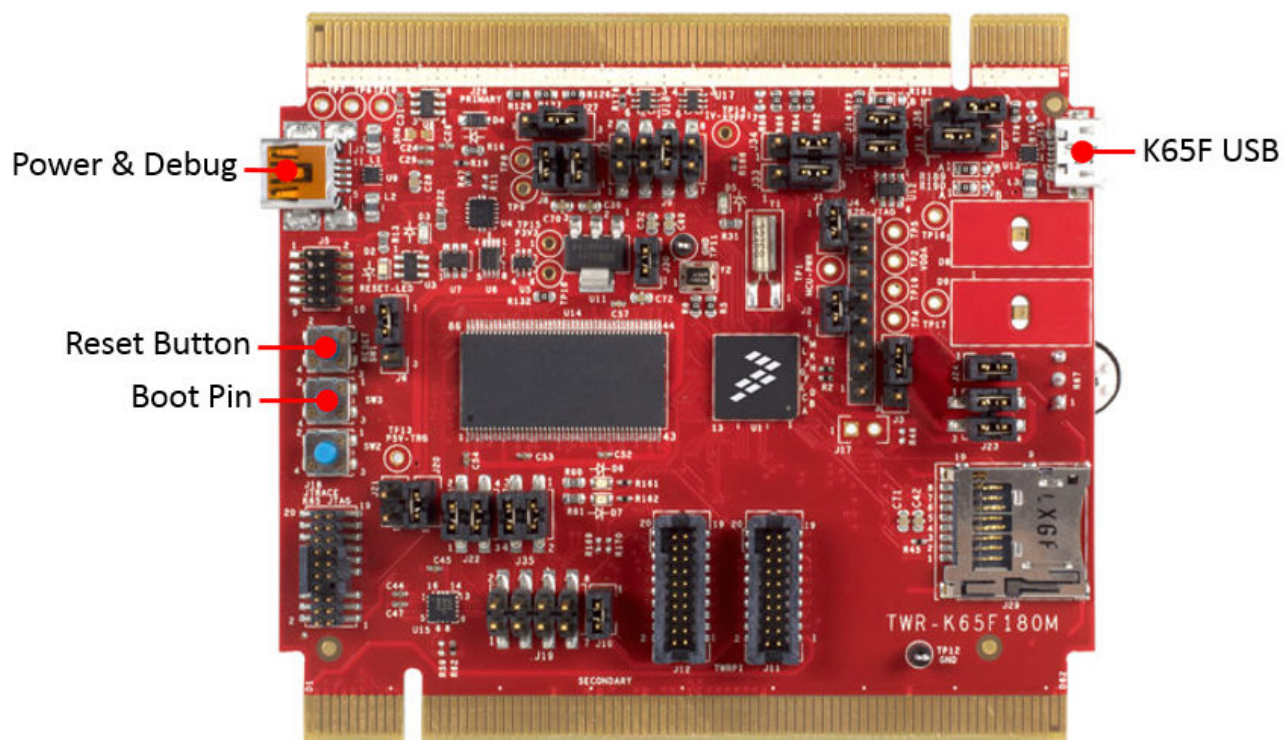


Figure 23. TWR-K65F180M platform

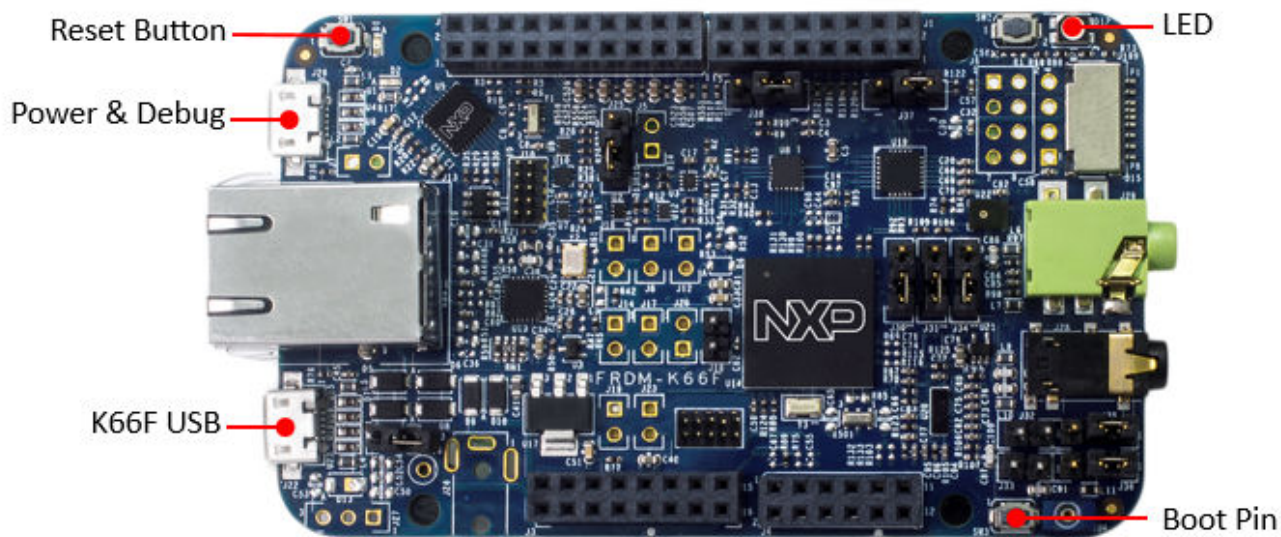


Figure 24. FRDM-K66F platform

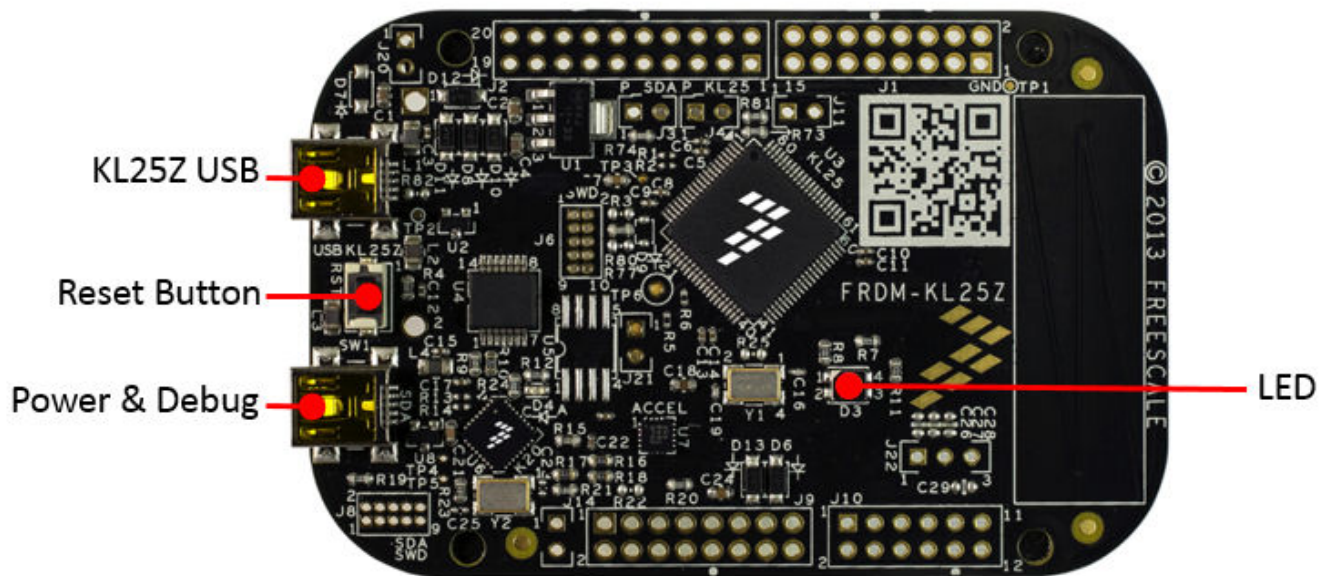


Figure 25. FRDM-KL25Z platform



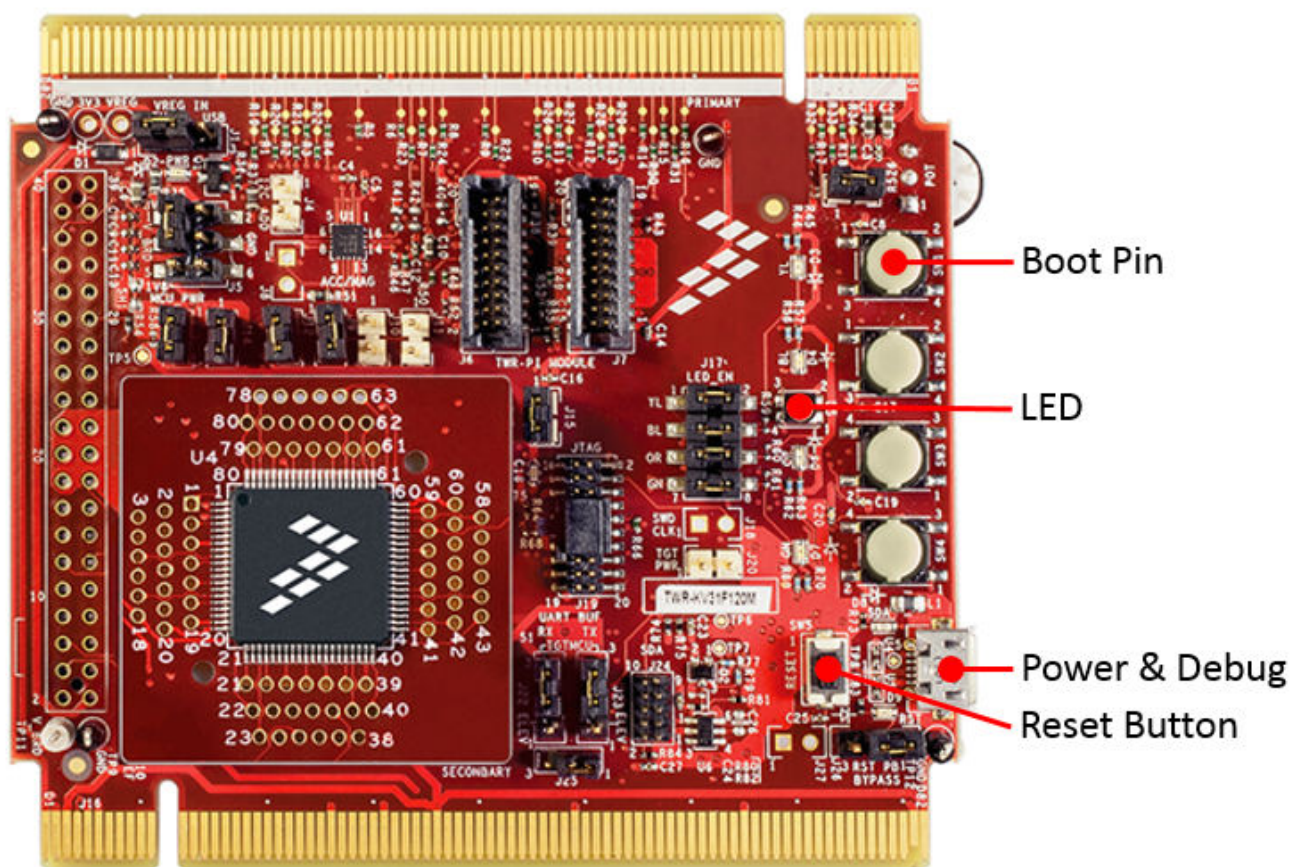


Figure 26. TWR-KV31F120M platform



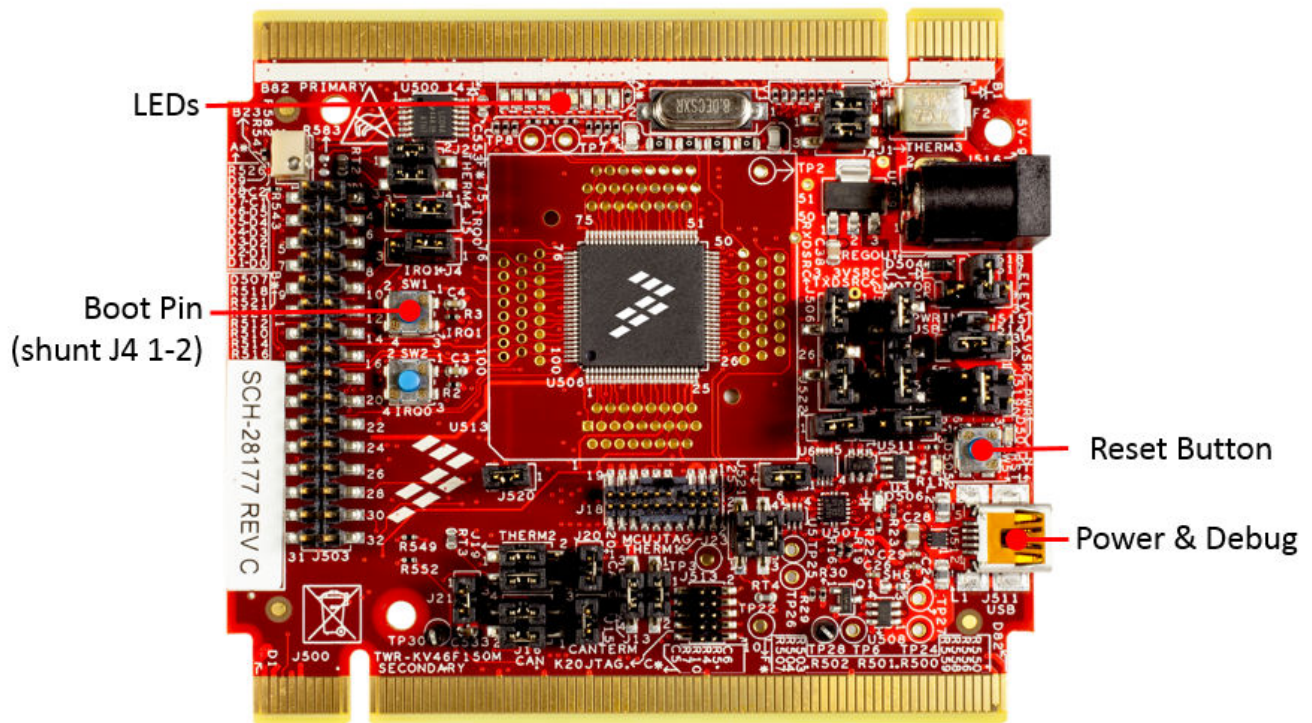


Figure 27. TWR-KV46F150M platform

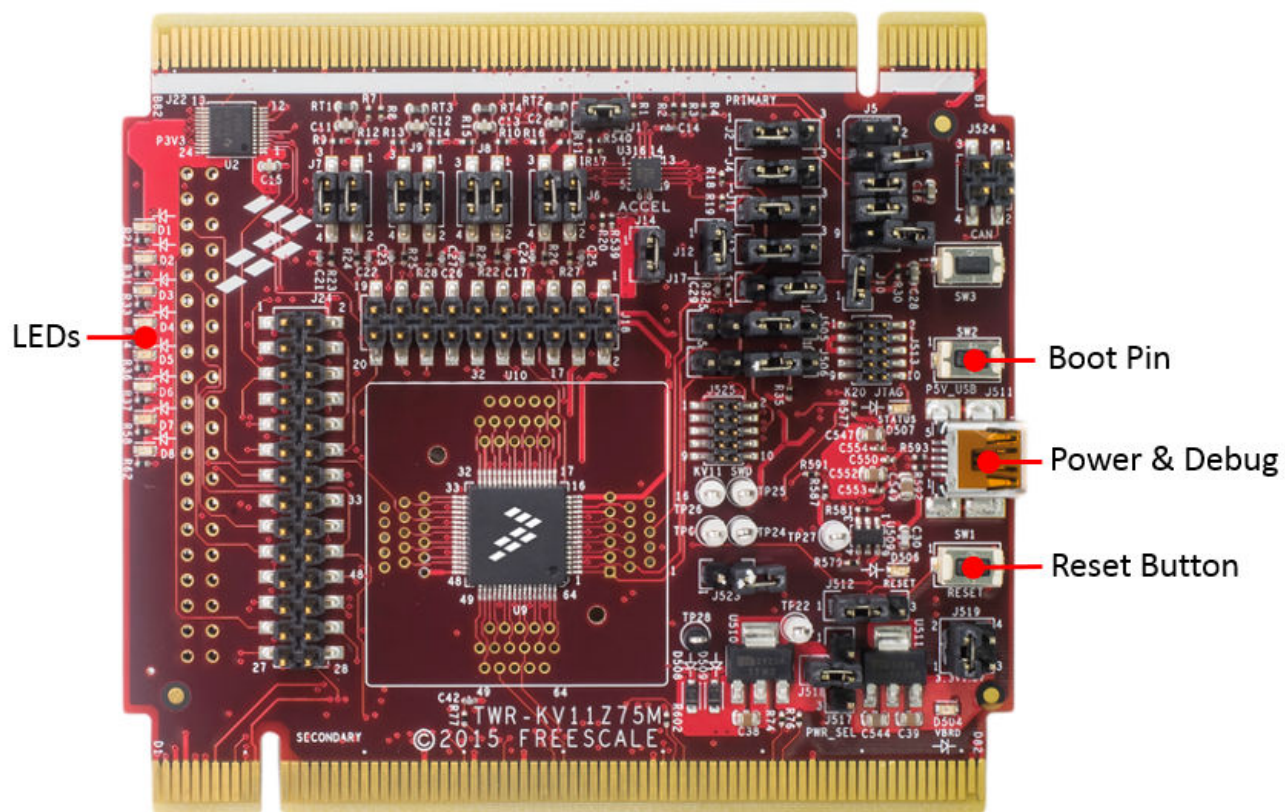


Figure 28. TWR-KV11Z75M platform



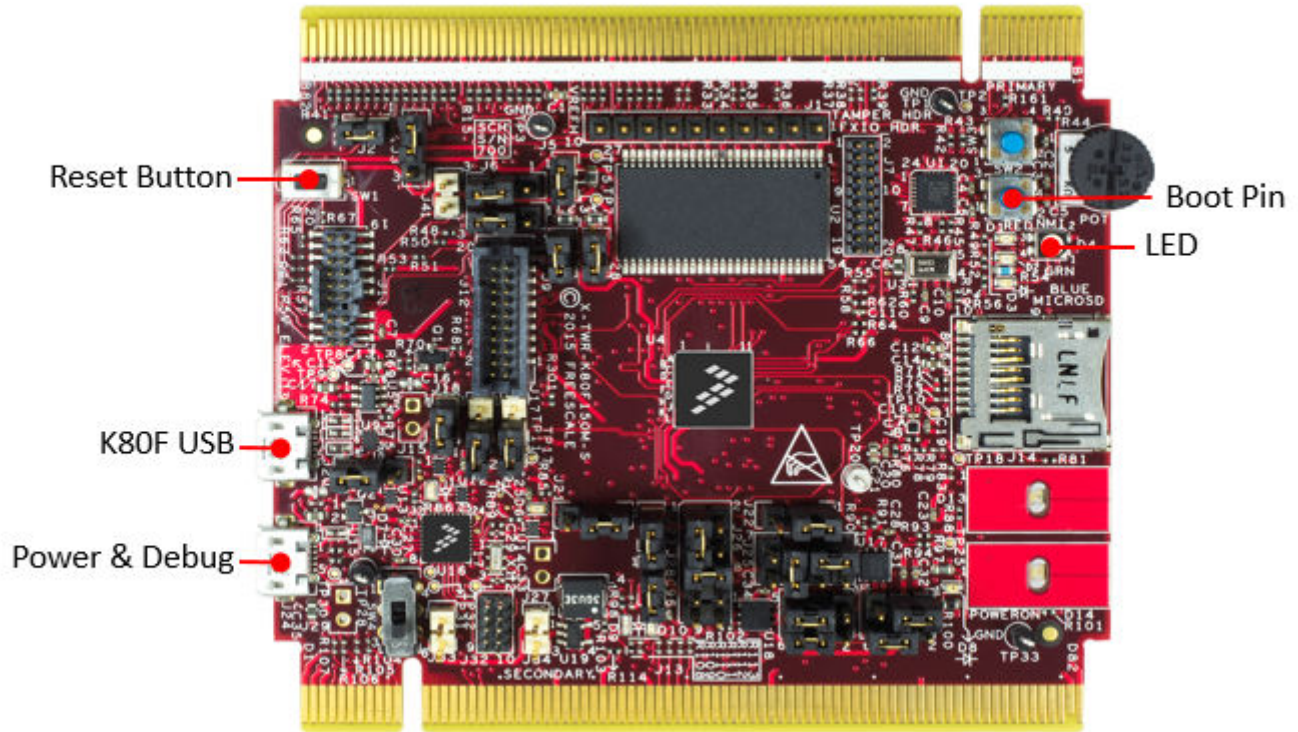


Figure 29. TWR-K80F150M platform

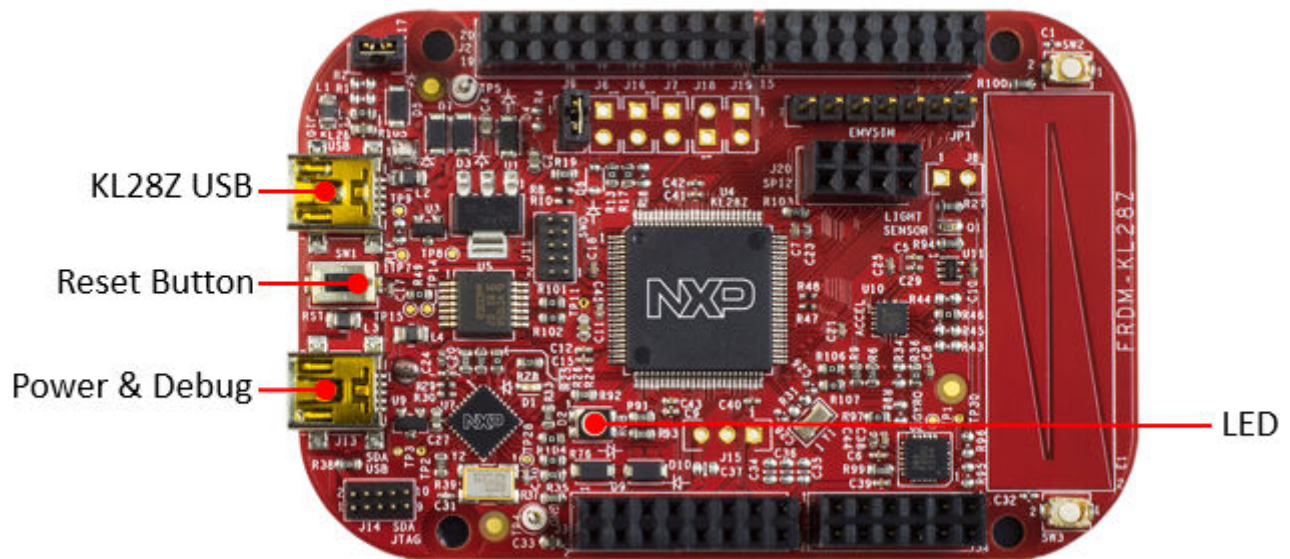


Figure 30. FRDM-KL28Z platform

# FRDM-K82F

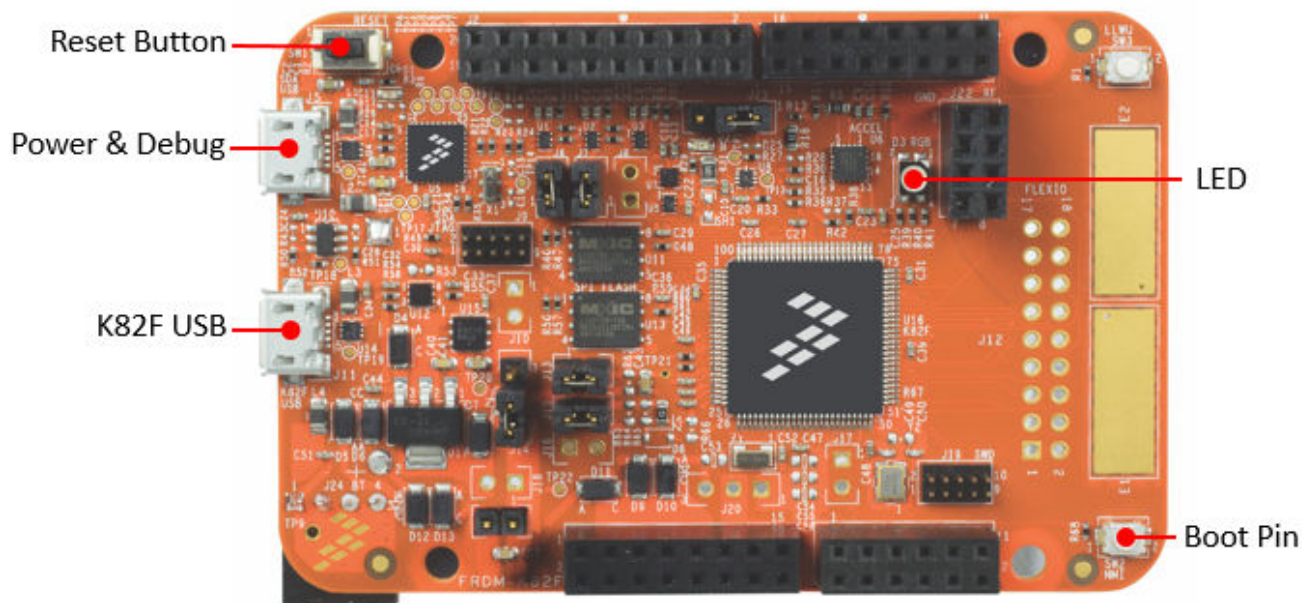


Figure 31. FRDM-K82F platform

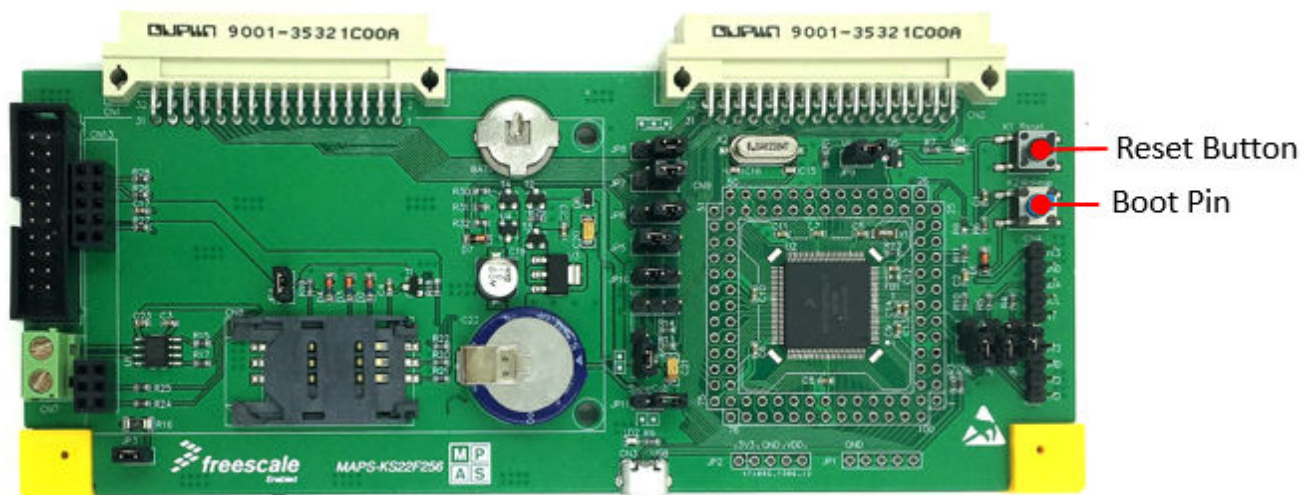


Figure 32. MAPS-KS22 platform

## 9 Appendix C - MCU Bootloader Pin mappings

**Table 1. MK22F128R/256R/512R bootloader/flashloader – TWR-K22F120M/FRDM-K22F**

Peripheral	Instance	Port	Signal	AltMode	TWR-K22F120M Test points	FRDM-K22F Test points
UART	1	PTE0	UART1_TX	3	OpenSDA port J25	mbed J5
		PTE1	UART1_RX			
I2C	1	PTC10	I2C1_SCL	2	J9 pin 1	J1 pin 13
		PTC11	I2C1_SDA		J7 pin 1	J2 pin 7
SPI	1	PTD4	SPI1_PCS0	7	J16 pin 1	J2 pin 6
		PTD5	SPI1_SCK		J16 pin 3	J2 pin 12
		PTD6	SPI1_SOUT		J16 pin 5	J2 pin 8
		PTD7	SPI1_SIN		J16 pin 7	J2 pin 10
USB	-	-	USB0_DP	Default	USB J32	USB J16
			USB0_DM			

**Table 2. MK24F25612 bootloader – TWR-K24F120M**

Peripheral	Instance	Port	Signal	AltMode	TWR-K24F120M Test points
UART	1	PTE0	UART1_TX	3	mbed port J37
		PTE1	UART1_RX		
I2C	1	PTC10	I2C1_SCL	2	Elev B50
		PTC11	I2C1_SDA		Elev B51
SPI	1	PTD4	SPI1_PCS0	7	J3 pin 1
		PTD5	SPI1_SCK		J3 pin 3
		PTD6	SPI1_SOUT		J3 pin 5
		PTD7	SPI1_SIN		J3 pin 7
USB	-	-	USB0_DP	Default	USB J23
			USB0_DM		

### NOTE

If testing the UART interface on mbed port J37, add shunts on J25 pin 2-3. If testing the UART interface on TWR-SER, add shunts on J25 pin 1-2 and J22 pin 1-2.

**Table 3. MKL25Z4 bootloader – FRDM-KL25Z**

Peripheral	Instance	Port	Signal	AltMode	FRDM-KL25Z Test points
UART	0	PTA2	UART0_TX	2	OpenSDA
		PTA1	UART0_RX		
I2C	0	PTC8	I2C0_SCL	2	J1 pin 14
		PTC9	I2C0_SDA		J1 pin 16
SPI	0	PTD0	SPI0_PCS0	2	J2 pin 6
		PTD1	SPI0_SCK		J2 pin 12
		PTD2	SPI0_SOUT		J2 pin 8
		PTD3	SPI0_SIN		J2 pin 10
USB	-	-	USB0_DP	Default	USB connector KL25Z
			USB0_DM		

**Table 4. MK64F12 bootloader – TWR-K64F120M**

Peripheral	Instance	Port	Signal	AltMode	TWR-K64F120M Test points
UART	1	PTC4	UART1_TX	3	OpenSDA J2
		PTC3	UART1_RX		
I2C	1	PTC10	I2C1_SCL	2	Elev A75 or J3 pin 3
		PTC11	I2C1_SDA		Elev B71 or J3 pin 4
SPI	0	PTD0	SPI1_PCS0	2	Elev B46
		PTD1	SPI1_SCK		Elev B48
		PTD2	SPI1_SOUT		Elev B45
		PTD3	SPI1_SIN		Elev B44
USB	-	-	USB0_DP	Default	USB J17
			USB0_DM		

**Table 5. MK64F12 flashloader – TWR-K64F120M**

Peripheral	Instance	Port	Signal	AltMode	TWR-K64F120M Test points
UART	0	PTB17	UART0_TX	3	Elev B11
		PTB16	UART0_RX		Elev B10
I2C	1	PTC10	I2C1_SCL	2	Elev A75 or J3 pin 3
		PTC11	I2C1_SDA		Elev B71 or J3 pin 4
SPI	0	PTD0	SPI1_PCS0	2	Elev B46

*Table continues on the next page...*

**Table 5. MK64F12 flashloader – TWR-K64F120M (continued)**

Peripheral	Instance	Port	Signal	AltMode	TWR-K64F120M Test points
		PTD1	SPI1_SCK		Elev B48
		PTD2	SPI1_SOUT		Elev B45
		PTD3	SPI1_SIN		Elev B44
USB	-	-	USB0_DP	Default	USB J17
			USB0_DM		

**Table 6. MK64F12 bootloader/flashloader – FRDM-K64F**

Peripheral	Instance	Port	Signal	AltMode	FRDM-K64F Test points
UART	0	PTB17	UART0_TX	3	mbed port J26
		PTB16	UART0_RX		
I2C	1	PTC10	I2C1_SCL	2	J4 pin 12
		PTC11	I2C1_SDA		J4 pin 10
SPI	0	PTD0	SPI1_PCS0	2	J2 pin 6
		PTD1	SPI1_SCK		J2 pin 12
		PTD2	SPI1_SOUT		J2 pin 8
		PTD3	SPI1_SIN		J6 pin 10
USB	-	-	USB0_DP	Default	USB J22
			USB0_DM		

**Table 7. MK65F18 bootloader – TWR-K65F180M**

Peripheral	Instance	Port	Signal	AltMode	TWR-K65F180M Test points
UART	2	PTE16	UART2_TX	3	mbed port J7
		PTE17	UART2_RX		
I2C	0	PTD8	I2C0_SCL	2	J13 pin 2
		PTD9	I2C0_SDA		J14 pin 1
SPI	2	PTD11	SPI2_PCS0	2	Elev B46
		PTD12	SPI2_SCK		Elev B48
		PTD13	SPI2_SOUT		Elev B45
		PTD14	SPI2_SIN		Elev B44
USB	-	-	USB0_DP	Default	TWR_SER USB J14
			USB0_DM		

Table continues on the next page...



**Table 7. MK65F18 bootloader – TWR-K65F180M (continued)**

Peripheral	Instance	Port	Signal	AltMode	TWR-K65F180M Test points
HS USB	-	-	USB1_DM	Default	USB J15
			USB1_DP		

**Table 8. MK65F18 flashloader – TWR-K65F180M**

Peripheral	Instance	Port	Signal	AltMode	TWR-K65F180M Test points
UART	4	PTE24	UART4_TX	3	Elev A48
		PTE25	UART4_RX		Elev A47
I2C	0	PTD8	I2C0_SCL	2	J13 pin 2
		PTD9	I2C0_SDA		J14 pin 1
SPI	2	PTD11	SPI2_PCS0	2	Elev B46
		PTD12	SPI2_SCK		Elev B48
		PTD13	SPI2_SOUT		Elev B45
		PTD14	SPI2_SIN		Elev B44
USB	-	-	USB0_DP	Default	TWR-SER USB J14
			USB0_DM		

**Table 9. MK66F18 bootloader/flashloader – FRDM-K66F**

Peripheral	Instance	Port	Signal	AltMode	FRDM-K66F Test points
UART	0	PTB17	UART0_TX	3	port J26
		PTB16	UART0_RX		
I2C	1	PTC10	I2C1_SCL	2	J2 pin 20
		PTC11	I2C1_SDA		J2 pin 18
SPI	0	PTD0	SPI1_PCS0	2	J2 pin 6
		PTD1	SPI1_SCK		J2 pin 12
		PTD2	SPI1_SOUT		J2 pin 8
		PTD3	SPI1_SIN		J6 pin 10
HS USB	-	-	USB1_DP	Default	USB J22
			USB1_DM		



**Table 10. MKV30F12810 bootloader/flashloader – TWR-KV30F100M**

Peripheral	Instance	Port	Signal	AltMode	TWR-KV30F100M Test points
UART	1	PTC3	UART1_RX	3	Elev B47
		PTC4	UART1_TX		Elev A37
I2C	0	PTB0	I2C0_SCL	2	Elev A30
		PTB1	I2C0_SDA		Elev B28
SPI	0	PTE16	SPI0_PCS0	2	Elev B46
		PTE17	SPI0_SCK		Elev B48
		PTE18	SPI0_SOUT		Elev B45
		PTE19	SPI0_SIN		Elev B44

**Table 11. MK02F12810 bootloader/flashloader – TWR-KV30F100M**

Peripheral	Instance	Port	Signal	AltMode	TWR-KV30F100M Test points
UART	1	PTC3	UART1_RX	3	Elev B47
		PTC4	UART1_TX		Elev A37
I2C	0	PTB0	I2C0_SCL	2	Elev A30
		PTB1	I2C0_SDA		Elev B28
SPI	0	PTE16	SPI0_PCS0	2	Elev B46
		PTE17	SPI0_SCK		Elev B48
		PTE18	SPI0_SOUT		Elev B45
		PTE19	SPI0_SIN		Elev B44

**Table 12. MKV31F128/256/512 bootloader – TWR-KV31F120M**

Peripheral	Instance	Port	Signal	AltMode	TWR-KV31F120M Test points
UART	0	PTB16	UART0_TX	3	OpenSDA port
		PTB17	UART0_RX		
I2C	0	PTD2	I2C0_SCL	7	J9 pin 2
		PTD3	I2C0_SDA		J12 pin 1
SPI	0	PTE16	SPI0_PCS0	2	Elev B46
		PTE17	SPI0_SCK		Elev B48
		PTE18	SPI0_SOUT		Elev B45
		PTE19	SPI0_SIN		Elev B44

**Table 13. MKV31F512 bootloader – FRDM-KV31F**

Peripheral	Instance	Port	Signal	AltMode	TWR-KV31F120M Test points
UART	0	PTB17	UART0_TX	3	OpenSDA port
		PTB16	UART0_RX		
I2C	0	PTB2	I2C0_SCL	2	J2 pin 20
		PTB1	I2C0_SDA		J2 pin 18
SPI	0	PTE16	SPI0_PCS0	2	J1 pin 15
		PTE17	SPI0_SCK		J2 pin 12
		PTE18	SPI0_SOUT		J2 pin 8
		PTE19	SPI0_SIN		J2 pin 10

**Table 14. MKV46F15 bootloader – TWR-KV46F150M**

Peripheral	Instance	Port	Signal	AltMode	TWR-KV46F150M Test points
UART	0	PTD6	UART0_TX	3	OpenSDA port
		PTD7	UART0_RX		
I2C	0	PTB0	I2C0_SCL	2	Elev B28 or J501 pin 22
		PTB1	I2C0_SDA		Elev B27 or J501 pin 33
SPI	0	PTE16	SPI0_PCS0	2	Elev A27 or J501 pin 10
		PTE17	SPI0_SCK		Elev A28 or J501 pin 12
		PTE18	SPI0_SOUT		Elev B29 or J501 pin 18
		PTE19	SPI0_SIN		Elev B30 or J501 pin 20
FlexCAN	0	PTA12	CAN0_TX	2	J13 pin 1
		PTA13	CAN0_RX		J13 pin 2

**Table 15. MKV11Z7 bootloader – TWR-KV11Z75M**

Peripheral	Instance	Port	Signal	AltMode	TWR-KV11Z75M Test points
UART	0	PTD17	UART0_TX	3	OpenSDA J511
		PTD16	UART0_RX		
I2C	0	PTB0	I2C0_SCL	2	J18 pin 17

*Table continues on the next page...*

**Table 15. MKV11Z7 bootloader – TWR-KV11Z75M (continued)**

Peripheral	Instance	Port	Signal	AltMode	TWR-KV11Z75M Test points
		PTB1	I2C0_SDA		J18 pin 18
SPI	0	PTE16	SPI0_PCS0	2	J18 pin 5
		PTE17	SPI0_SCK		J18 pin 6
		PTE18	SPI0_SOUT		J18 pin 7
		PTE19	SPI0_SIN		J18 pin 8
FlexCAN	0	PTA24	CAN0_TX	2	J24 pin 13
		PTA25	CAN0_RX		J24 pin 14

**NOTE**

If testing the UART interface on OpenSDA port J511, add shunts on J505 pin 2-3 and J506 pin 2-3.

**Table 16. MKV11Z7 flashloader – TWR-KV11Z75M**

Peripheral	Instance	Port	Signal	AltMode	TWR-KV11Z75M Test points
UART	0	PTD6	UART0_TX	3	J24 pin 27
		PTD7	UART0_RX		J24 pin 28
I2C	0	PTB0	I2C0_SCL	2	J18 pin 17
		PTB1	I2C0_SDA		J18 pin 18
SPI	0	PTE16	SPI0_PCS0	2	J18 pin 5
		PTE17	SPI0_SCK		J18 pin 6
		PTE18	SPI0_SOUT		J18 pin 7
		PTE19	SPI0_SIN		J18 pin 8
FlexCAN	0	PTA24	CAN0_TX	2	J24 pin 13
		PTA25	CAN0_RX		J24 pin 14

**Table 17. KS22F256 Bootloader - MAPS-KS22**

Peripheral	Instance	Port	Signal	AltMode	MAPS-KS22F256 Test points	
UART	1	PTE0	UART1_TX	3	M1-5	OpenSDA port on Dock CN14
		PTE1	UART1_RX		M1-6	
I2C	0	PTB0	LPI2C0_SCL	2	CN4-4	
		PTB1	LPI2C0_SDA		CN4-3	
SPI	1	PTD4	SPI1_PCS0	7	CN13-1	
-		PTD5	SPI1_SCK		CN13-3	
-		PTD6	SPI1_SOUT		CN13-30	

Table continues on the next page...

**Table 17. KS22F256 Bootloader - MAPS-KS22 (continued)**

Peripheral	Instance	Port	Signal	AltMode	MAPS-KS22F256 Test points
-	0	PTD7	SPI1_SIN	Default	CN13-29
USB		-	USB0_DP		CN3
-		-	USB0_DM		
FlexCAN	0	PTB18	CAN0_TX	2	CN7-1(CANH)
-		PTB19	CAN0_RX		CN7-2(CANH)

**NOTE**

CAN connection – option 1, use CAN transceiver on board:

1. Put jumper on J5 pin 1-2, and keep default jumpers on J5 5-6 and 7-8
2. Connect TWR-KV11Z75M J524 pin 2 to BusPal (KV46) J13 pin 2
3. Connect TWR-KV11Z75M J524 pin 1 to BusPal (KV46) J13 pin 1

CAN connection - option 2, use CAN transceiver on TWR-SER board:

1. Remove the jumpers on TWR-SER J5 pins 5-6 and pins 7-8
2. Wring CAN0\_TX  
Wire CAN0\_TX on TWR-KV11Z75M J24 pin 13 to TWR-SER J5 pin 8 - signal name C\_TXD
3. Wring CAN0\_RX  
Wire CAN0\_TX on TWR-KV11Z75M J24 pin 14 to TWR-SER J5 pin 6 - signal name C\_RXD
4. Connect to BusPal (KV46)  
Connect TWR-SER CANH, J7 pin 1 to BusPal KV46 J13 pin 2  
Connect TWR-SER CANH, J7 pin 3 to BusPal KV46 J13 pin 1

**Table 18. MKV58F22 bootloader - TWR-KV58F220M**

Peripheral	Instance	Port	Signal	AltMode	TWR-KV11Z128M Test points
UART	0	PTB1	UART0_TX	7	mbed port J22
		PTB0	UART0_RX		
I2C	0	PTB2	I2C0_SCL	2	J14 pin 7
		PTB3	I2C0_SDA		J14 pin 5
SPI	0	PTE16	SPI0_PCS0	2	Elev B46
		PTE17	SPI0_SCK		Elev B48
		PTE18	SPI0_SOUT		Elev B45
		PTE19	SPI0_SIN		Elev B44
FlexCAN	0	PTB16	CAN0_TX	5	Elev B68
		PTB17	CAN0_RX		Elev B67

**Table 19. MKV58F22 flashloader - TWR-KV58F220M**

Peripheral	Instance	Port	Signal	AltMode	TWR-KV11Z128M Test points
UART	0	PTD7	UART0_TX	3	Elev A80
		PTD6	UART0_RX		J31 pin 14
I2C	0	PTB0	I2C0_SCL	2	J24 pin 2
		PTB1	I2C0_SDA		J25 pin 2
SPI	0	PTE16	SPI0_PCS0	2	Elev B46
		PTE17	SPI0_SCK		Elev B48
		PTE18	SPI0_SOUT		Elev B45
		PTE19	SPI0_SIN		Elev B44
FlexCAN	0	PTB16	CAN0_TX	2	Elev B68
		PTB17	CAN0_RX		Elev B67

**NOTE**

CAN connection - Option 1, use the CAN transceiver on the board

1. CAN0\_TX, Elev B68 -> TWR-SER J5 pin 8 , TWR-SER J7 pin 1 -> BusPal (KV46) J13 pin 2
2. CAN0\_RX, Elev B67 -> TWR-SER J5 pin 6 , TWR-SER J7 pin 3 -> BusPal (KV46) J13 pin 1

**Bootloader for ROMs:**

- Define macros to use ROM bootloader pins for all devices having ROM bootloader:  

```
#define BL_FEATURE_ROM_UART_PORT (0)
#define BL_FEATURE_ROM_I2C_PORT (0)
#define BL_FEATURE_ROM_SPI_PORT (0)
```
- User ROM boot pin - PA4

**Table 20. MK80F256 bootloader - TWR-K80F150M**

Peripheral	Instance	Port	Signal	AltMode	TWR-K80F150M Test points
LPUART	1	PTC3	UART1_RX	3	Mbed port J24
		PTC4	UART1_TX		
I2C	1	PTC10	I2C1_SCL	2	Elev B50
		PTC11	I2C1_SDA		Elev B51
SPI	1	PTD4	SPI1_PCS0	7	Elev A78
		PTD5	SPI1_SCK		Elev A79
		PTD6	SPI1_SOUT		Elev A80
		PTD7	SPI1_SIN		Elev A56

*Table continues on the next page...*

**Table 20. MK80F256 bootloader - TWR-K80F150M (continued)**

USB	0	-	USB0_DP	Default	J19
		-	USB0_DM		

**Table 21. MK80F256 bootloader - FRDM-K82F**

Peripheral	Instance	Port	Signal	AltMode	FRDM-K82F Test points
LPUART	1	PTC14	UART4_RX	2	Mbed port J5
		PTC15	UART4_TX		
I2C	1	PTC10	I2C1_SCL	2	J1 pin 12
		PTC11	I2C1_SDA		J1 pin 14
SPI	1	PTD4	SPI1_PCS0	7	J2 pin 6
		PTD5	SPI1_SCK		J2 pin 12
		PTD6	SPI1_SOUT		J2 pin 10
		PTD7	SPI1_SIN		J2 pin 8
USB	0	-	USB0_DP	Default	J11
		-	USB0_DM		

**Table 22. MKL28Z7 bootloader - TWR-KL28Z72M**

Peripheral	Instance	Port	Signal	AltMode	TWR-KL28Z72M Test points
LPUART	0	PTA1	UART0_RX	2	Mbedport J10
		PTA2	UART0_TX		
LPI2C	1	PTE1	I2C1_SCL	6	Elev B45
		PTE0	I2C1_SDA		Elev A35
LPSP1	2	PTB20	SPI2_PCS0	7	Elev B9
		PTB21	SPI2_SCK		Elev B7
		PTB22	SPI2_SOUT		Elev B10
		PTB23	SPI2_SIN		Elev B11
USB	0	-	USB0_DP	Default	J29
		-	USB0_DM		

**Table 23. MKL28Z7 bootloader - FRDM-KL28Z**

Peripheral	Instance	Port	Signal	AltMode	FRDM-KL28Z Test points
------------	----------	------	--------	---------	---------------------------

*Table continues on the next page...*

**Table 23. MKL28Z7 bootloader - FRDM-KL28Z (continued)**

LPUART	0	PTB16	UART0_RX	3	Mbed port J13
		PTB17	UART0_TX		
LPI2C	1	PTC1	I2C1_SCL	2	J4 pin 12
		PTC2	I2C1_SDA		J4 pin 10
LPSPI	2	PTB20	SPI2_PCS0	7	J20 pin 4
		PTB21	SPI2_SCK		J20 pin 5
		PTB22	SPI2_SOUT		J20 pin 6
		PTB23	SPI2_SIN		J20 pin 7
USB	0	-	USB0_DP	Default	J10
		-	USB0_DM		

**Table 24. MKL81Z7 bootloader - TWR-KL82Z72M**

Peripheral	Instance	Port	Signal	AltMode	TWR-KL82Z72M Test points
LPUART	1	PTC3	UART1_RX	3	Mbed port J24
		PTC4	UART1_TX		
I2C	1	PTC10	I2C1_SCL	2	Elev B50
		PTC11	I2C1_SDA		Elev B51
SPI	1	PTD4	SPI1_PCS0	7	Elev A78
		PTD5	SPI1_SCK		Elev A79
		PTD6	SPI1_SOUT		Elev A80
		PTD7	SPI1_SIN		Elev A56
USB	0	-	USB0_DP	Default	J11
		-	USB0_DM		

**Table 25. MKL82Z7 bootloader - FRDM-KL82Z**

Peripheral	Instance	Port	Signal	AltMode	FRDM-K82F Test points
LPUART	0	PTB16	UART0_RX	3	Mbed port J5
		PTB17	UART0_TX		
I2C	1	PTC10	I2C1_SCL	2	J2 pin 20
		PTC11	I2C1_SDA		J2 pin 18
SPI	1	PTD4	SPI1_PCS0	7	J2 pin 4
		PTD5	SPI1_SCK		J2 pin 5

*Table continues on the next page...*

**Table 25. MKL82Z7 bootloader - FRDM-KL82Z (continued)**

USB	0	PTD6	SPI1_SOUT	Default	J2 pin 6
		PTD7	SPI1_SIN		J2 pin 7
		-	USB0_DP		J11
		-	USB0_DM		

## 10 Revision history

This table summarizes revisions to this document.

**Table 26. Revision history**

Revision number	Date	Substantive changes
0	07/2015	Kinetis Bootloader 1.2.0 initial release
1	12/2015	Updates for standalone Kinetis KS22F256 bootloader v1.0.0 based on Kinetis bootloader v1.2.0 initial release.
2	04/2016	Kinetis Bootloader v2.0.0 release
3	05/2018	MCU Bootloader v2.5.0 release
4	09/2018	MCU Bootloader v2.6.0 release
5	11/2018	MCU Bootloader v2.7.0 release





#### **How To Reach Us**

##### **Home Page:**

[nxp.com](http://nxp.com)

##### **Web Support:**

[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2018 NXP B.V.

© NXP B.V. 2018.

All rights reserved.

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: [salesaddresses@nxp.com](mailto:salesaddresses@nxp.com)

Date of release: December 2018

Document identifier: MBOOTDEMOUG

The ARM logo, consisting of the lowercase letters "arm" in a blue, sans-serif font.