

1 Introduction

1.1 General description

The MCU bootloader is a standard bootloader for all Kinetis devices. It provides a standard interface to the device using any of the peripherals supported by the bootloader on a given NXP Kinetis device.

The MCU flash loader is a specific implementation of the MCU bootloader. For the flash loader implementation, the MCU bootloader command interface is packaged as an executable that is loaded from flash and executed from RAM. This configuration allows the user application to be placed at the beginning of the on-chip flash where it is automatically launched upon boot from flash. Other MCU bootloader implementations include a ROM-based bootloader and a flash-resident bootloader.

The MCU bootloader is available as source code for custom, flash-based implementations. Example applications are provided to demonstrate how to interface with the bootloader. Using the MCU flash loader to program a user application to the beginning of the Kinetis flash makes this implementation of the bootloader a one-time programming aid.

Contents

1 Introduction	1
1.1 General description.....	1
1.2 Configurations and releases.....	1
2 Customizing Kinetis flash loader	3
2.1 Customizing supported interface.....	3
2.2 Customizing other options.....	3
3 Flash loader demo example	4
3.1 Getting flash loader software.....	4
3.2 Compiling and downloading.....	5
3.3 Ruining the flash boot loader.....	6
4 References	8

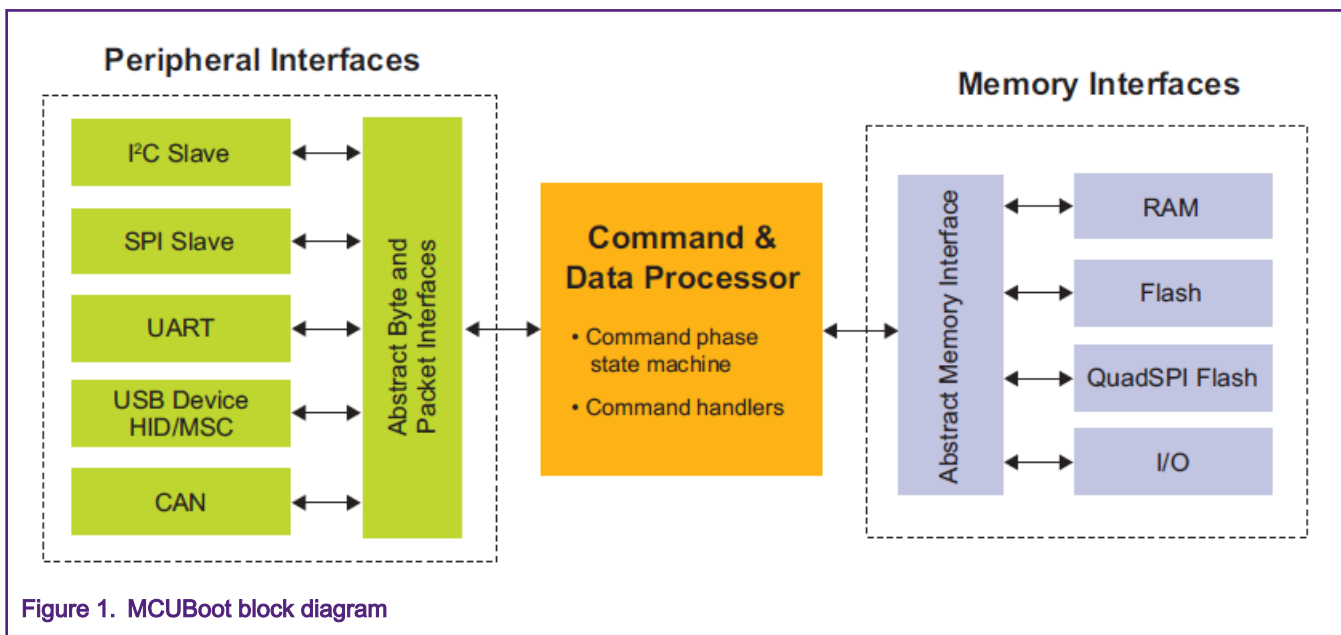


Figure 1. MCUBoot block diagram

1.2 Configurations and releases

The bootloader is delivered in two ways:

- As full source code that is highly configurable.



- Pre-programmed by NXP into ROM or flash on select NXP MCUs.

Host-side command line and GUI tools are available to communicate with the bootloader. The MCU bootloader uses startup, header files, and peripheral drivers from MCUXpresso SDK.

There are three kinds of bootloader configuration provided by NXP, as shown in [Figure 2](#).

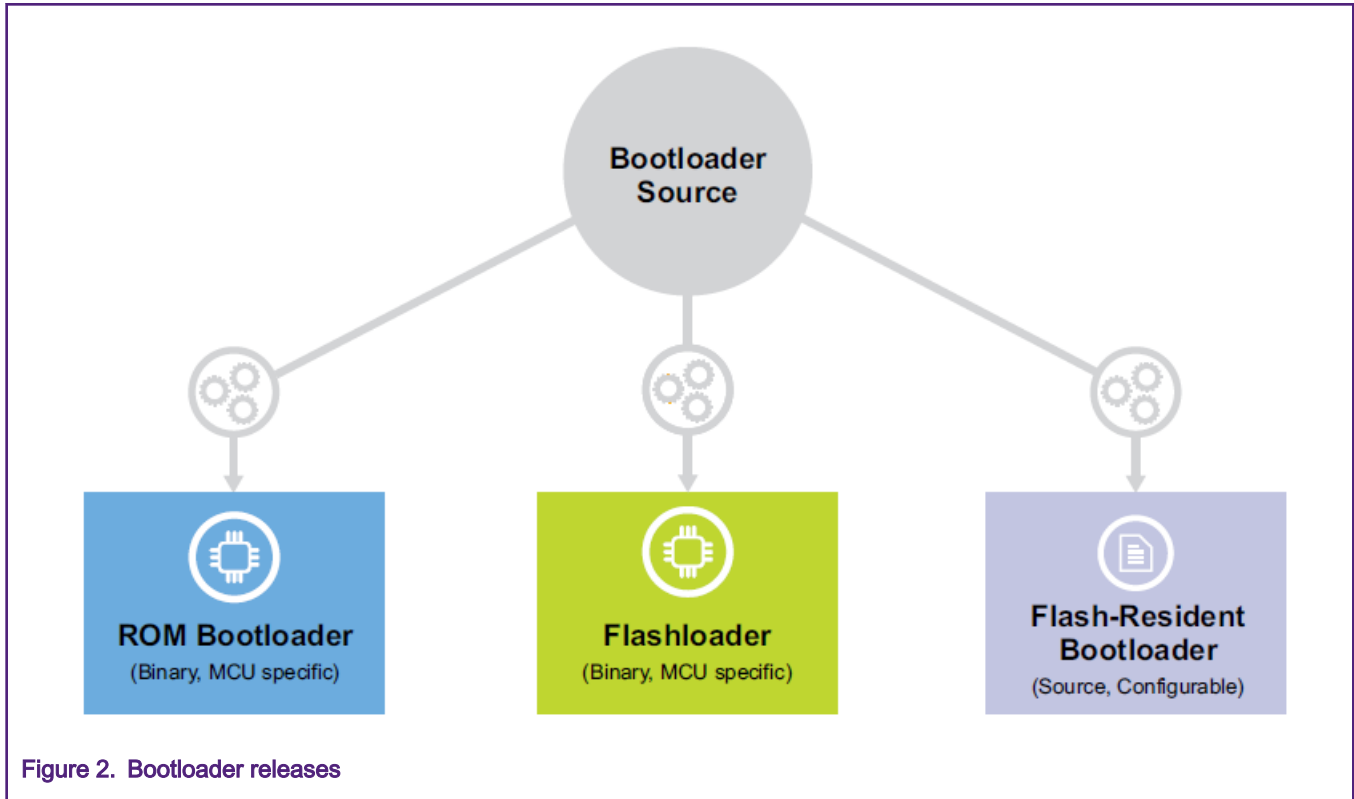


Figure 2. Bootloader releases

Table 1 describes features and differences.

Table 1. Features and differences

Bootloader configuration	ROM bootloader	Flash loader	Flash bootloader
User case	Factory flash programming and field update	Factory flash programming	Field update
Delivery	Binary preprogrammed in ROM by NXP (User cannot change)	Binary preprogrammed in flash by NXP	Source code provided in major release
Supported device	All Kinetis devices with a boot ROM	Select Kinetis devices without a ROM	Select Kinetis devices
Feature	Can run at system startup or callable from user application	Always run at system start-up	Can run at system startup or callable from user application
	Can jump to user application after peripheral timeout	Overwritten by user application	Can jump to user application after peripheral timeout

For KM35Z512, there is no ROM inside the chip and no factory pre-programmed flash loader before shipping. So NXP provides flash bootloader. Customer can download MCU flash boot loader source from KM35Z512 SDK.

For details about different bootloader configurations and releases, see [Documentation](#).

2 Customizing Kinetis flash loader

The `bootloader_config.h` in the `middleware\mcu-boot\targets\MKM35Z512Z7\src` contain many custom options that can be modified by users.

2.1 Customizing supported interface

MCU flash loader can be tailored to support different serial interfaces and modify each interface's instance. The macros can be customized according to hardware requirements, as shown in [Figure 3](#).

```
#define BL_CONFIG_UART (BL_CONFIG_UART_2)
#define BL_CONFIG_SPI (BL_CONFIG_SPI_1)
#define BL_CONFIG_I2C (BL_CONFIG_I2C_0)
//@}
```

Figure 3. Changed supported interface

2.2 Customizing other options

[Figure 4](#) shows some other options that can be customized in the `bootloader_config.h`.

```
#if defined(BL_TARGET_RAM)
#define BL_FEATURE_FLASH_SECURITY (0)
#else
#define BL_FEATURE_FLASH_SECURITY (1)
#endif

#if !defined(BL_TARGET_RAM)
#define BL_FEATURE_CRC_CHECK (1)
#endif

#define BL_FEATURE_FILL_MEMORY (1)
#define BL_FEATURE_READ_MEMORY (1)

#define BL_FEATURE_QSPI_MODULE (0)
#define BL_FEATURE_ENCRYPTION (0)

#define BL_FEATURE_UART_AUTOBAUD_IRQ (1)
```

Figure 4. Other boot loader configurations in `bootloader_config.h`

Table 2. Other options

Name	Description
BL_FEATURE_CRC_CHECK	This option is to enable/disable CRC check feature on each command or data package reception. Usually, this option needs to set to 1 to make sure all packet reception is correct.
BL_FEATURE_FILL_MEMORY	This option is to enable/disable filling the memory feature. Usually, this option needs to set to 1 to enable flash program ability.

Table continues on the next page...

Table 2. Other options (continued)

Name	Description
BL_FEATURE_READ_MEMORY	This option is to enable/disable reading the memory feature. For a simplest bootloader where you want user to read your application data, this option can be set to 0 .
BL_FEATURE_QSPI_MODULE	For KM35Z512, this option needs to be set to 0 .
BL_FEATURE_ENCRYPTION	For KM35Z512, this option needs to set to 0 .
BL_FEATURE_UART_AUTOBAUD_IRQ	To enable the UART interface autobaud detection feature, this option need to be set to 1 .

3 Flash loader demo example

3.1 Getting flash loader software

MCU-boot source code and example are now a middleware embedded in MCU SDK.

1. Go to <https://mcuxpresso.nxp.com/en/dashboard>, click **Select Board** and search **KM35Z512**. Select **TWR-KM35Z512Z75M**, as shown in [Figure 5](#).

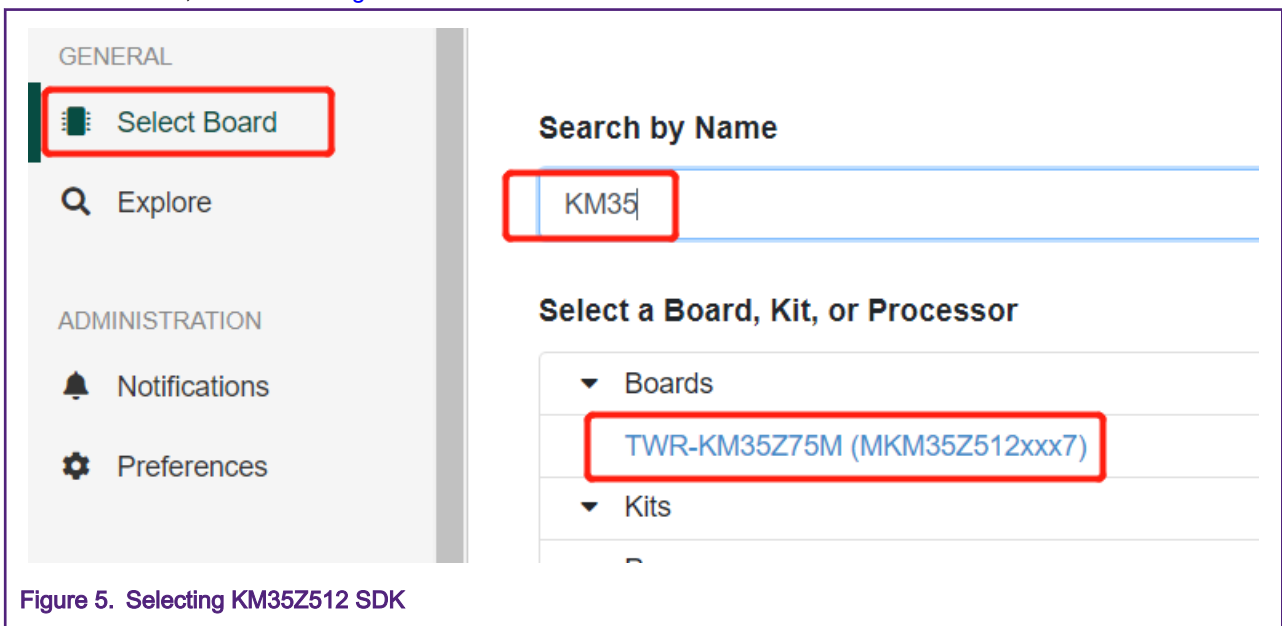


Figure 5. Selecting KM35Z512 SDK

2. Click **Build SDK**, select **mcu-boot**, and then click **Download SDK**, as shown in [Figure 6](#).

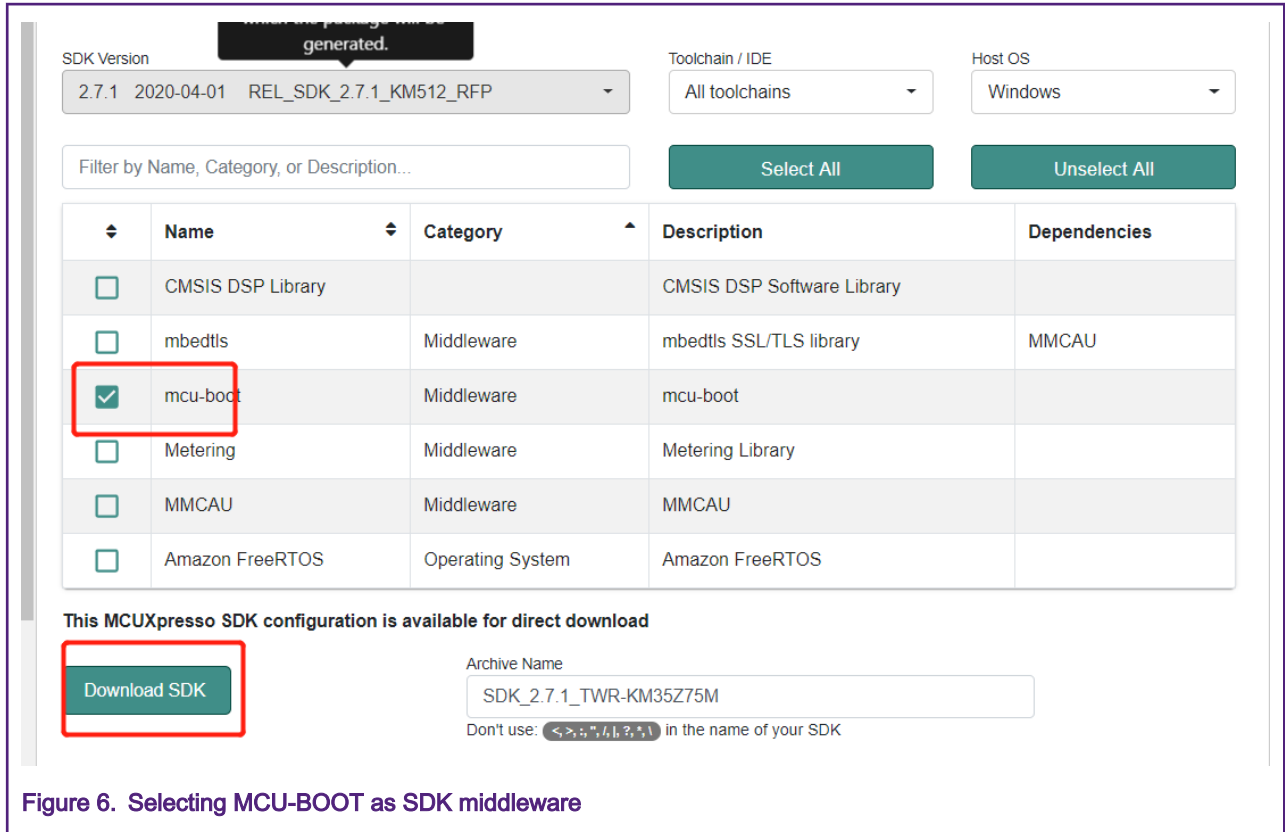


Figure 6. Selecting MCU-BOOT as SDK middleware

3.2 Compiling and downloading

3.2.1 Example project and source code location

Table 3. Example project and source code location

Name	Location	Description
tower_bootloader	<i>lboards\twrKM35Z512z75m lbootloader_examples</i>	KM35Z512 flash loader example project.
led_demo_tower_a000	<i>lboards\twrKM35Z512z75m lbootloader_examples\demo_apps</i>	Example <code>led_blink</code> demo compile @ 0xA000, use for bootloader application example.
Mcu-boot	<i>lmiddleware\mcu-boot</i>	Mcu-boot source code, PC host source code, PC host binary.
blhost	<i>lmcu-boot\bin\Tools\blhost</i>	Precompiled CLI PC host tool for mcu-boot.
KinetisFlashTool	<i>lmiddleware\mcu-boot\bin\Tools lKinetisFlashTool</i>	Precompiled GUI PC host tool for mcu-boot.

3.2.2 Compiling a demo project

1. Compile the `tower_bootloader` example and download to the target board. This is the flash loader demo project. It is necessary to reset MCU after bootloader programming to let code running.
2. Compile the `led_demo_tower_a000` example and create a binary file (.bin) from the output elf. This is the `led_blink` demo project used as flash loader application demo.

Save the `led_demo_tower_a000.bin` to a known location. This binary will be used in the following sections.

In this document, we copy the binary (`led_demo_tower_a000.bin`) to `SDK_2.7.1_TWR-KM35Z75M\middleware\mcu-boot\bin\Tools\blhost\win`.

3.3 Ruining the flash boot loader

3.3.1 Using KinetisFlashTool

KinetisFlashTool is a GUI wrapper of BLHOST. It's much easier to use. The following sections will guide you how to download application demo (`led_demo_tower_a000.bin`) via KinetiFlashTool. For more detailed information, see *Kinetis Flash Tool User's Guide* (document [MBOOTFLTOOLUG](#)).

1. Click **UART** and select COM and baud rate. Press the **Reset** button on the board and click **Connect** to connect the board.
2. If connected successful, the status field will show varies information for the target, such as, Flash start address, flash size, RAM start address, and RAM size, etc. It indicated that KinetisFlashTool is connected to flash loader.
3. Select **Image** (`led_demo_tower_a000.bin`) and enter **0xA000** in the **Target Address**. Then, click **Update** to finish the application update.

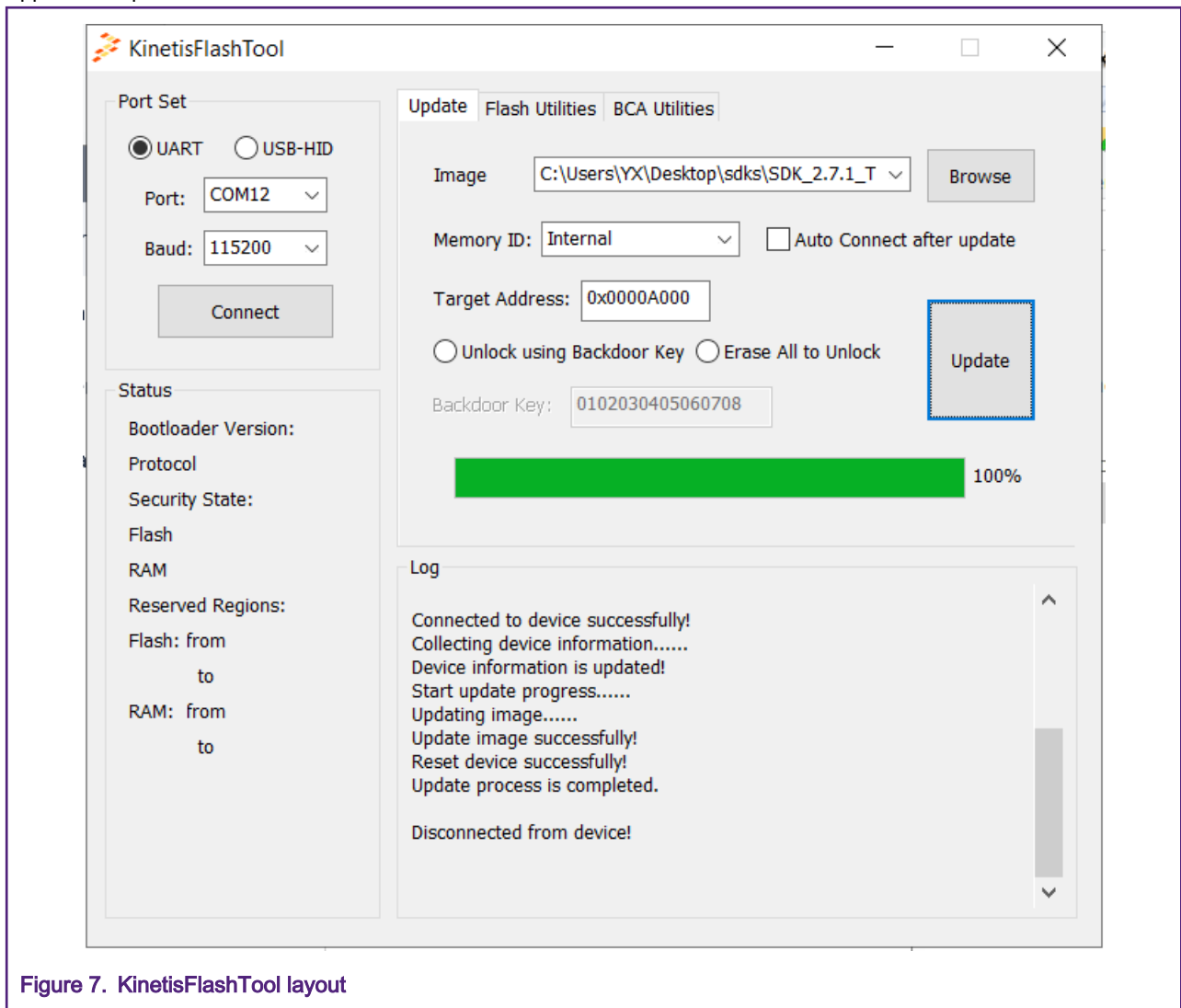


Figure 7. KinetisFlashTool layout

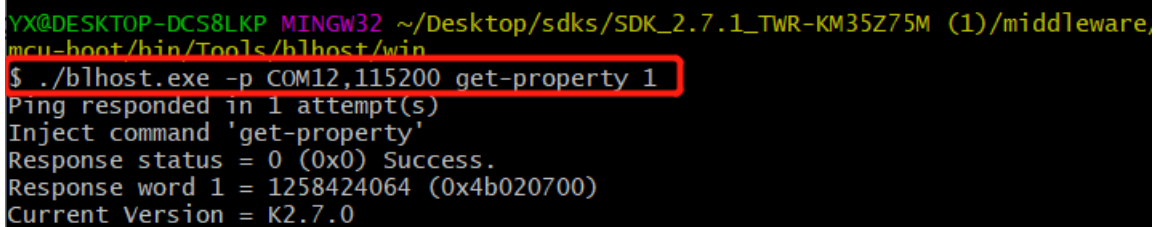
3.3.2 Using BLHOST

BLHOST is Command Line Interface (CLI) tool which uses for the PC host to communicate with Kinetis Flash loader or Kinetis ROM bootloader. The following steps will guide you how to use BLHOST to communicate with flash boot loader and download image to the target.

1. Use `./blhost.exe -p COMX,115200 get-property 1` to ping with the target.

If the connection is OK, the target will respond with a **Ping respond packet** and return the boot loader version information. This is the first command you need to execute.

The **COMX** is the virtual serial port number on your PC. For windows, please see the device manager for details (eg: COM12).



```

YX@DESKTOP-DCS8LKP MINGW32 ~/Desktop/sdks/SDK_2.7.1_TWR-KM35Z75M (1)/middleware/
mcu-boot/bin/Tools/blhost/win
$ ./blhost.exe -p COM12,115200 get-property 1
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258424064 (0x4b020700)
Current Version = K2.7.0
  
```

Figure 8. STEP1: BLHOST ping target

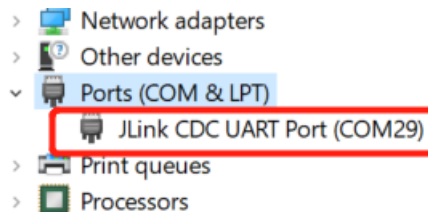
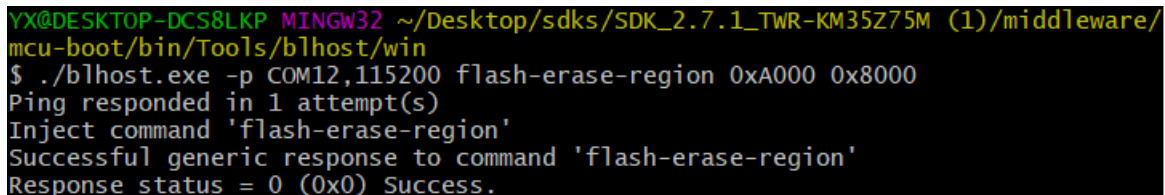


Figure 9. Virtual COM can be found in Device Manager

2. `./blhost.exe -p COMX,115200 flash-erase-region 0xA000 0xA000`

This command will erase the target memory starting from `0xA000` with the size of `0xA000`.



```

YX@DESKTOP-DCS8LKP MINGW32 ~/Desktop/sdks/SDK_2.7.1_TWR-KM35Z75M (1)/middleware/
mcu-boot/bin/Tools/blhost/win
$ ./blhost.exe -p COM12,115200 flash-erase-region 0xA000 0x8000
Ping responded in 1 attempt(s)
Inject command 'flash-erase-region'
Successful generic response to command 'flash-erase-region'
Response status = 0 (0x0) Success.
  
```

Figure 10. SETP2: BLHOST erase flash

3. `./blhost.exe -p COMX,115200 write-memory 0xA000 led_demo_tower_a000.bin`

This command will download the `led_demo_tower_a000.bin` to the target at the address of `0xA000`.

```
YX@DESKTOP-DCS8LKP MINGW32 ~/Desktop/sdks/SDK_2.7.1_TWR-KM35Z75M (1)/middleware/  
mcu-boot/bin/Tools/blhost/win  
$ ./blhost.exe -p COM12,115200 write-memory 0xA000 led_demo_tower_a000.bin  
Ping responded in 1 attempt(s)  
Inject command 'write-memory'  
Preparing to send 1720 (0x6b8) bytes to the target.  
Successful generic response to command 'write-memory'  
(1/1)100% Completed!  
Successful generic response to command 'write-memory'  
Response status = 0 (0x0) Success.  
Wrote 1720 of 1720 bytes.
```

Figure 11. SETP3: BLHOST fill memory

4. `./blhost.exe -p COMX,115200 execute 0xA000 0 0`

This command will boot application at 0xA000 and jump to `led_demo_tower_a000`.

```
YX@DESKTOP-DCS8LKP MINGW32 ~/Desktop/sdks/SDK_2.7.1_TWR-KM35Z75M (1)/middleware/  
mcu-boot/bin/Tools/blhost/win  
$ ./blhost.exe -p COM12,115200 execute 0xA000 0 0  
Ping responded in 1 attempt(s)  
Inject command 'execute'  
Successful generic response to command 'execute'  
Response status = 0 (0x0) Success.
```

Figure 12. SETP4: BLHOST execute

4 References

1. *blhost User's Guide* (document [MCUBLHOSTUG](#))
2. *Getting Started with the MCU Flashloader* (document [MBOOTFLASHGS](#))
3. *Kinetis Flash Tool User's Guide* (document [MBOOTFLTOOLUG](#))
4. MCUBOOT landing page: [MCUBOOT: MCU Bootloader for NXP microcontrollers](#)

How To Reach Us

Home Page:

nxp.com

Web Support:

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.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QoriQ, QoriQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, UMEMS, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2020.

All rights reserved.

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

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: June 2020
Document identifier: AN12888

