AN13352

Introduction to Flash Resident Bootloader for KE17Z

Rev. 0 — 31 August 2021 Application Note

1 Introduction

1.1 Overview

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 bootloader is available as source code for custom, flash-based implementations. Example applications are provided to demonstrate how to interface with the bootloader.

Contents

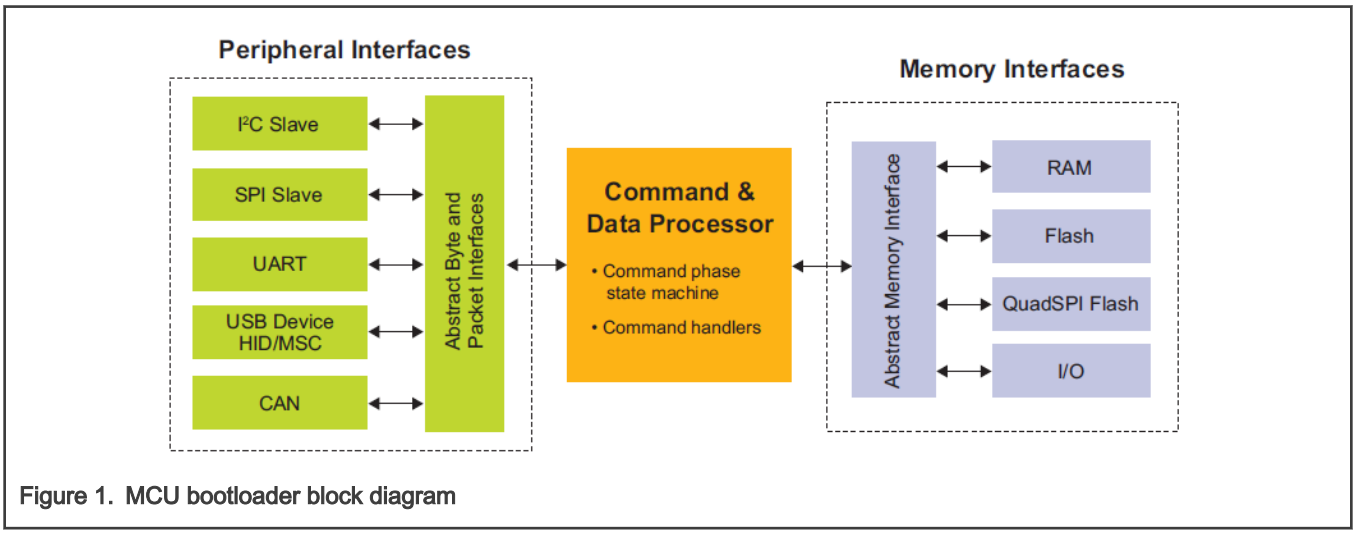

Figure 1. MCU bootloader block diagram

1.2 Different configurations and release ways

The MCU bootloader is delivered in two ways:

- As full source code that is highly configurable
- Pre-programmed by NXP® into ROM or flash on selected 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.

NXP provides three kinds of MCU bootloader, as shown in Figure 2.

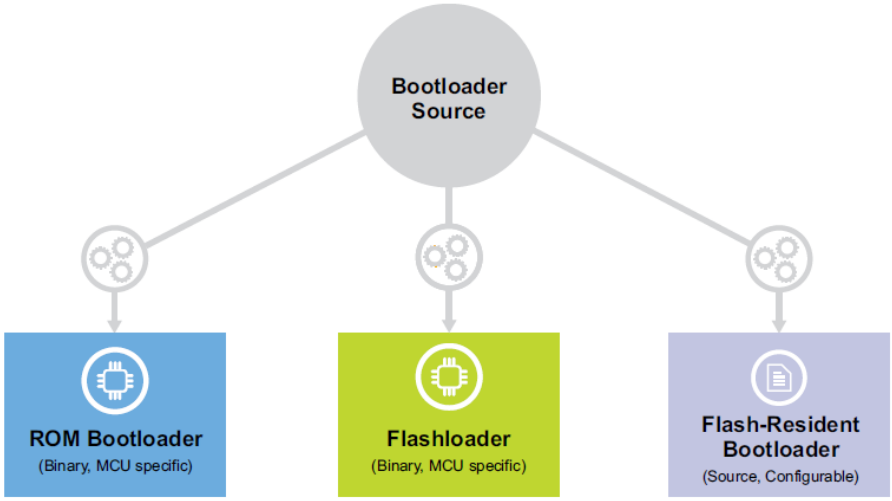

Figure 2. Three kinds of release of bootloader provided by NXP

Table 1 describes the main features and differences.

Table 1. Main features and differences

Bootloader configurations	ROM Bootloader	Flash loader	Flash-resident 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 KE17Z, there is no ROM inside the chip and no factory pre-programmed Flashloader before shipping. NXP provides **flash resident bootloader**. Customers can download MCU flash resident bootloader source code from KE17Z SDK.

For details about different bootloader configurations and release ways, see MCUBOOT: MCU Bootloader for NXP Microcontrollers

2 Customizing Kinetis Flash loader

- SDK version: SDK_2_10_0_FRDM-KE17Z
- IDE used in this application note: Keil V5.3.1
- bootloader_config.h is located in:

```
\middleware\mcu-boot\targets\MKE17Z7\src
```

There are many options to be customized.

2.1 Modifying supported interface

MCU flash resident bootloader can be tailored to support different serial interfaces. Modify the instances of the interface, as shown in Figure 3. You can modify those macros to customize according to hardware requirements.

```
#define BL_CONFIG_LPUART_0 (1)
#define BL_CONFIG_LPUART (BL_CONFIG_LPUART_0)

#define BL_CONFIG_LPI2C_0 (1)
#define BL_CONFIG_LPI2C (BL_CONFIG_LPI2C_0)

#define BL_CONFIG_LPSPI_0 (1)
#define BL_CONFIG_LPSPI (BL_CONFIG_LPSPI_0)
```

Figure 3. Modify supported interface

2.2 Modifying other options

There are some other options that can be customized in **boolotader_config.h**, as shown in Figure 4.

```
#define BL_CONFIG_LPUART_0 (1)
#define BL_CONFIG_LPUART (BL_CONFIG_LPUART_0)

#define BL_CONFIG_LPI2C_0 (1)
#define BL_CONFIG_LPI2C (BL_CONFIG_LPI2C_0)

#define BL_CONFIG_LPSPI_0 (1)
#define BL_CONFIG_LPSPI (BL_CONFIG_LPSPI_0)
```

Figure 4. Other bootloader configurations in bootloader_config.h

Table 2. Description for other bootloader configurations

Name	Description
BL_FEATURE_CRC_CHECK	This option enables/disables the CRC check feature on each command or data package reception. To make sure all packet receptions are correct, set this option to **1**.
BL_FEATURE_FILL_MEMORY	This option enables/disables the filling memory feature. To enable flash program ability, set this option to **1**.
BL_FEATURE_READ_MEMORY	This option enables/disables the reading memory feature. For a simplest bootloader where users can read your application data, set this option to **0**.
BL_FEATURE_QSPI_MODULE	For KE17Z, set this option to **0**.

Table continues on the next page...

Table 2. Description for other bootloader configurations (continued)

Name	Description
BL_FEATURE_ENCRYPTION	For KE17Z, set this option to **0**.
BL_FEATURE_UART_AUTOBAUD_IRQ	To enable UART interface autobaud detection feature, set this option to **1**.
BL_APP_VECTOR_TABLE_ADDRESS	The default application start address is `0xA000`.

3 Flash resident bootloader demo example

3.1 Get flash resident bootloader software

The Flash resident bootloader source code and example are now as a middleware embedded in MCU SDK.

1. Go to MCUXpresso SDK Dashboard, click **Select Board**, search **KE17Z**, and select **FRDM-KE17Z**, as shown in Figure 5.

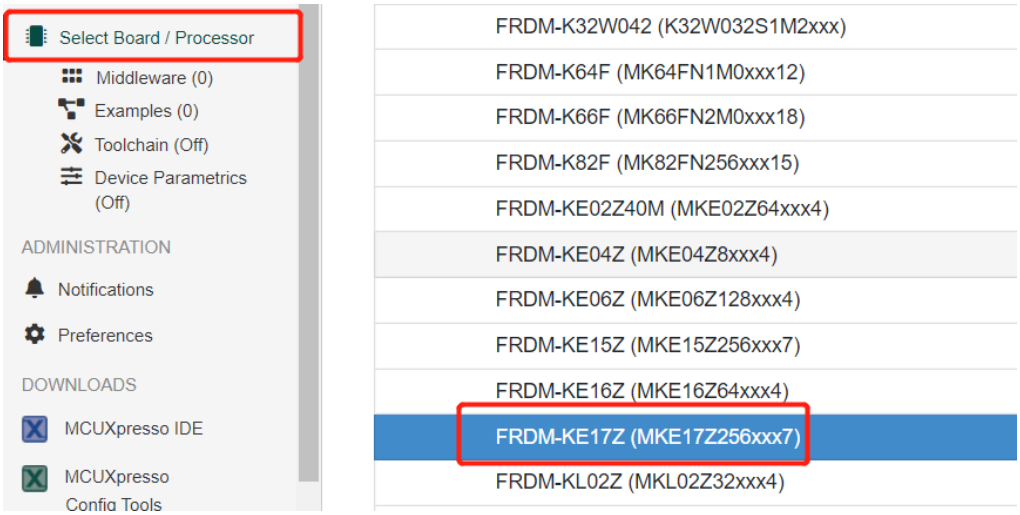

Figure 5. Select FRDM-KE17Z SDK

2. Click **Build SDK**, select **mcu-boot**, and click **Download SDK**, as shown in Figure 6.

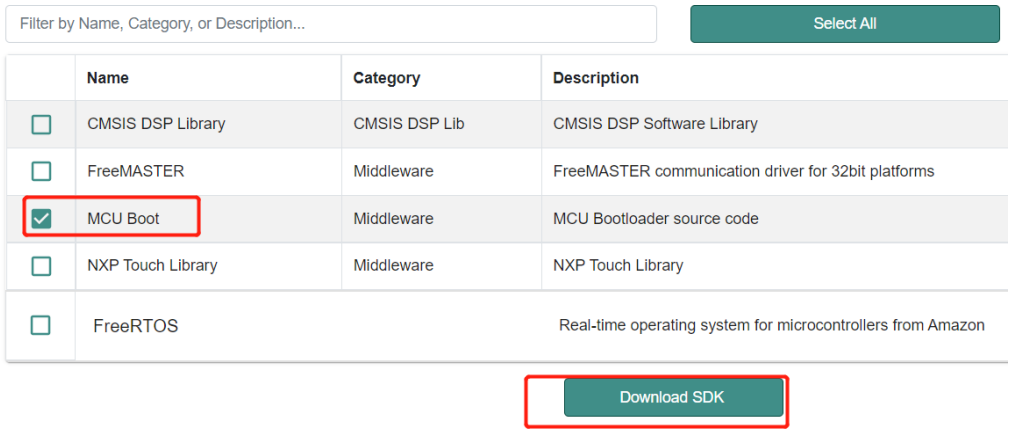

Figure 6. Select MCU-BOOT as SDK middleware

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
freedom_bootloader	`\boards\frdmke17z\bootloader_examples\freedom_bootloader`	KE17Z flash loader example project.
led_blinky	`\boards\frdmke17z\demo_apps\led_blinky`	This is a simple `led_blinky` demo, but starts at `0x0000_0000`. To relocate code start address to 0xA000, modify the linker file.
Mcu-boot	`\middleware\mcu-boot`	Mcu-boot source code, PC host source code, PC host binary.
blhost	`\mcu-boot\bin\Tools\blhost`	Precompiled CLI PC host tool for mcu-boot.
KinetisFlashTool	`\middleware\mcu-boot\bin\Tools\KinetisFlashTool`	Precompiled GUI PC host tool for mcu-boot.

3.2.2 Compiling demo project

1. Compile **freedom_bootloader** example and download to target board. It is the flash loader demo project. To run the code, reset the MCU after the bootloader programming.
2. Open the `led_blinky` demo, modify the image code start address from `0x0000_0000` to `0x0000_A000` in the linker file, and save the linker file.

led_blinky.c | MKE17Z256xxx7_flash.scf

```
18  **
19  **      SPDX-License-Identifier: BSD-3-Clause
20  **
21  **      http:                 www.nxp.com
22  **      mail:                 support@nxp.com
23  **
24  ** ###################################################################
25  */
26
27  #define m_interrupts_start          0x0000A000
28  #define m_interrupts_size           0x00000200
29
30  #define m_flash_config_start        0x0000A400
31  #define m_flash_config_size         0x00000010
32
33  #define m_text_start                0x0000A410
34  #define m_text_size                 0x0003FBF0
```

Figure 7. Modifying Linker file from 0x0000_0000 to 0x0000_A000

3. Compile **led_blinky** example and create a binary file (`.bin`) from the output elf. It is the `led_blink` demo project used as flash loader application demo.
4. Save the **led_blinky.bin** to a known location. This binary is used in following sections.

   In this application note, copy the binary (`led_blinky.bin`) to `\middleware\mcu-boot\bin\Tools\blhost\win`.

3.3 Ruining the flash bootloader

3.3.1 Using KinetisFlashTool

KinetisFlashTool is a GUI wrapper of BLHOST. It is much more intuitive and easy to use. To download application demo (led_blinky.bin) via KinetiFlashTool, perform the following steps. For more detailed information, see *Kinetis Flash Tool User's Guide* (document MBOOTFLTOOLUG).

1. Select **UART**, **COM**, and **Baud** rate. To connect the board, press the **Reset** button on the board and click **Connect**.
2. If the connection is successful, the status field shows varies information for the target, such as, Flash start address, flash size, RAM start address, and RAM size. It indicates that KinetisFlashTool is connected to flash loader.
3. To update the application, select **led_blinky.bin** for **Image**, enter `0xA000` for **Target Address**, and click **Update**.

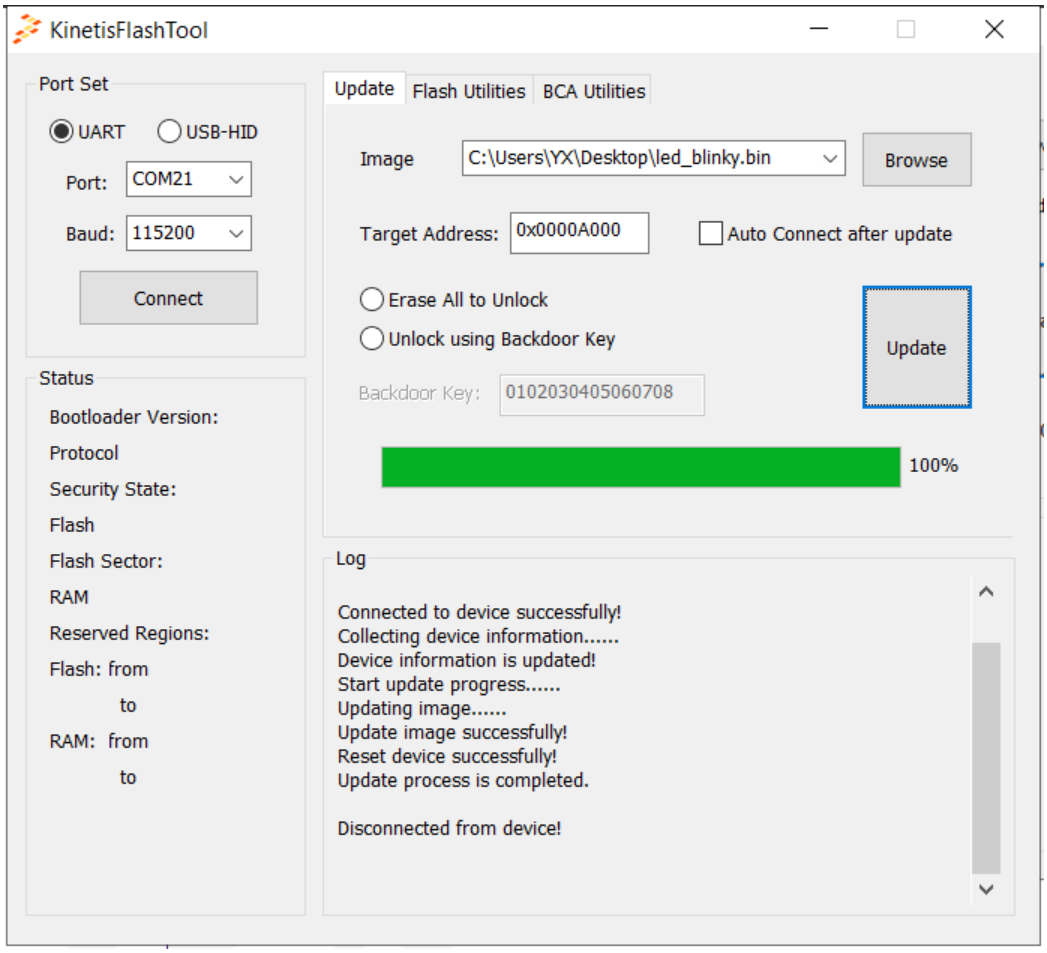

Figure 8. KinetisFlashTool layout

3.3.2 Using BLHOST

BLHOST is Command Line Interface (CLI) tool used for PC host to communicate with KinetisBootloader. To use BLHOST to communicate with flash boot loader and download image to the target, perform the following steps.

1. **./blhost.exe -p COMX,115200 get-property 1**

   Use this command to ping with target. If the connection is OK, the target responds with a **Ping respond packet** message and returns the boot loader version information.

   The **COMX** is the virtual serial port number on your PC. For windows, see device manager for details (eg: COM21).

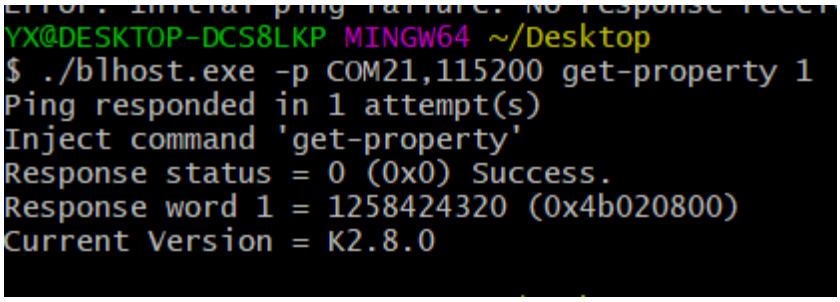

Figure 9. STEP1: BLHOST ping target

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

   Use this command to erase target memory start from `0xA000` with the size of `0xA000`.

```
YX@DESKTOP-DCS8LKP MINGW64 ~/Desktop
$ ./blhost.exe -p COM21,115200 flash-erase-region 0xA000 0xA000
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_blinky.bin.**

   Use this command to download `led_blinky.bin` to the target at `0xA000`.

```
YX@DESKTOP-DCS8LKP MINGW64 ~/Desktop
$ ./blhost.exe -p COM21,115200 write-memory 0xA000 led_blinky.bin
Ping responded in 1 attempt(s)
Inject command 'write-memory'
Preparing to send 4136 (0x1028) 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 4136 of 4136 bytes.
```

Figure 11. SETP3: BLHOST fill memory

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

   Use this command to boot the application at `0xA000` and jump to `led_blinky`.

```
YX@DESKTOP-DCS8LKP MINGW64 ~/Desktop
$ ./blhost.exe -p COM21,115200 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 Reference

1. *blhost User's Guide* (document MCUBLHOSTUG)
2. MCUBOOT: MCU Bootloader for NXP Microcontrollers
3. *Getting Started with the MCU Flashloader* (document MBOOTFLASHGS)
4. *Kinetis Flash Tool User's Guide* (document MBOOTFLTOOLUG)
5. *Introduction to Kinetis Flashloader for KM35* (document AN12888)

5 Revision history

Rev.	Date	Description
0	31 August 2021	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — 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.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

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

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

Date of release: 31 August 2021

Document identifier: AN13352

arm