

KE16Z MCU Bootloader Demo Applications User's Guide

by: NXP Semiconductors

1 Introduction

This document describes how to use the KE16Z MCU bootloader to load a user application on a KE16Z device.

2 Overview

This guide describes the steps required to use the NXP-provided KE16Z bootloader utilities to update the user application section to flash. On reset, the bootloader detects the user application and launches it. This functionality enables application developers to install new applications onto the KE16Z devices easily. It also provides a way to update the KE16Z device in the field without a debugger.

2.1 KE16Z MCU bootloader

The KE16Z bootloader serves as a standard bootloader for KE16Z devices. It provides a standard interface to communicate with the device using one of the peripherals (UART, SPI, I2C, or MSCAN) supported by a KE16Z device. The customers can use this bootloader source code to create a unique flash-resident bootloader that is compatible with tools that understand the bootloader interface and also supports application-specific features. NXP provides utilities which demonstrate 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. Check *blhost User's Guide* (document MCUBLHOSTUG) for detailed information.

Contents

1 Introduction.....	1
2 Overview.....	1
2.1 KE16Z 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
3 Getting Started with KE16Z MCU bootloader application.....	2
3.1 Plug it in.....	2
3.1.1 Select a board.....	2
3.1.2 Plug in the board.....	3
3.1.3 OpenSDA.....	3
3.1.4 PC Configuration.....	3
4 The host utility application.....	4
5 Returning to Flash- resident bootloader....	5
6 Appendix A - MCU flash-resident bootloader operation.....	5
6.1 Memory map overview.....	6
6.2 User application vector table.....	7
6.3 Bootloader Configuration Area (BCA).....	7
7 Appendix B - MCU Bootloader	



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 .

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 MBOOTFLT00LUG).

2.5 Toolchain requirement

Firmware projects:

- IAR Embedded Workbench for ARM[®] v.8.30.2
- Python v.2.7 (www.python.org)

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 (vcredist_x86.exe)
- Apple[®] Xcode v9.2 (for tools)
- GNU Compiler (GCC) v5.4.0 (for tools)

3 Getting Started with KE16Z MCU bootloader application

3.1 Plug it in

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-KE16Z as an example for demo

**Development
platforms..... 8**

**8 Appendix C - MCU
Bootloader Pin
mappings.....8**

9 Revision history..... 9

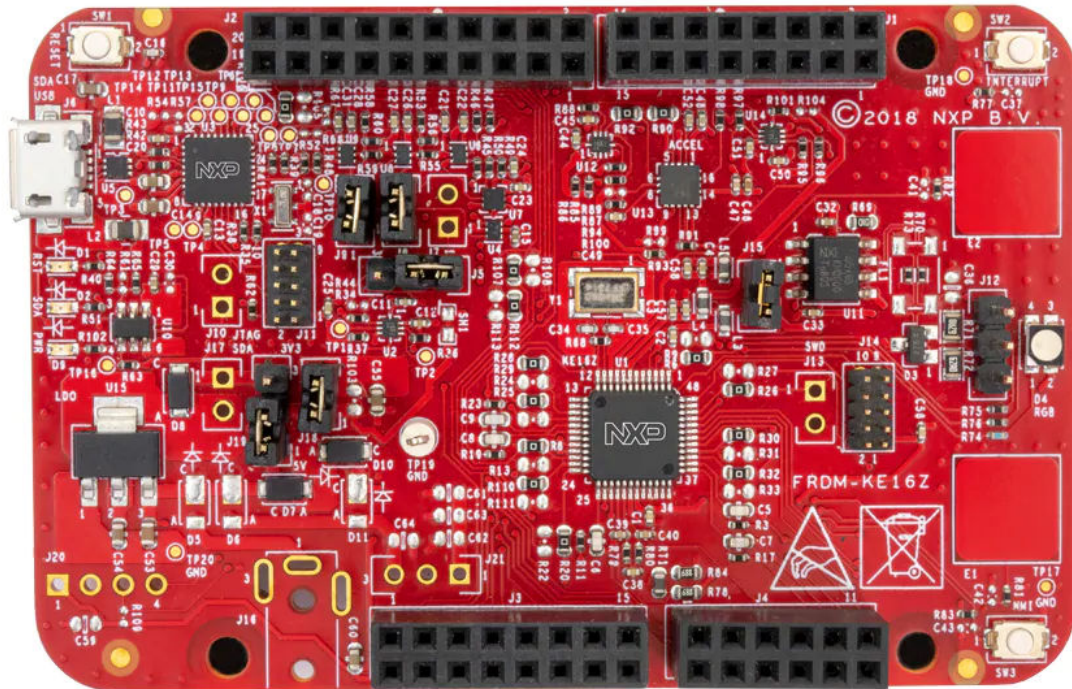


Figure 1. FRDM-KE16Z

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-K64F, go to this website and choose board FRDM-K64F 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:

The host utility application

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.

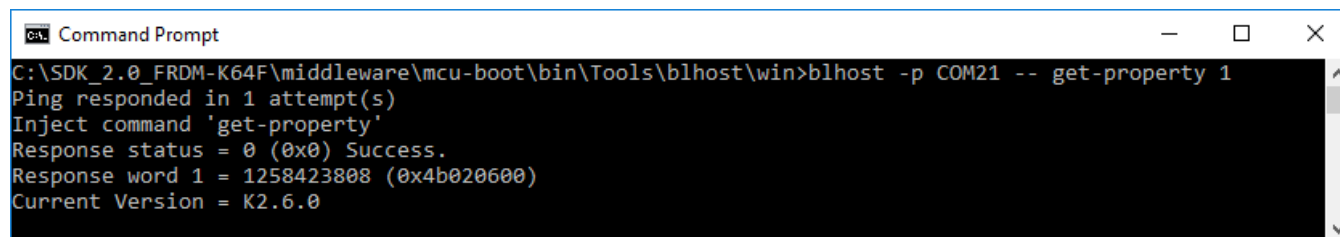
4 The host utility application

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 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.

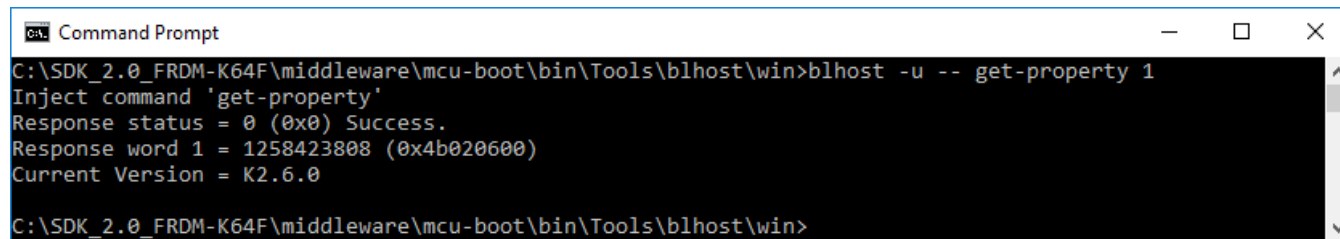


```
C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>blhost -p COM21 -- get-property 1
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258423808 (0x4b020600)
Current Version = K2.6.0
```

Figure 2. 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.



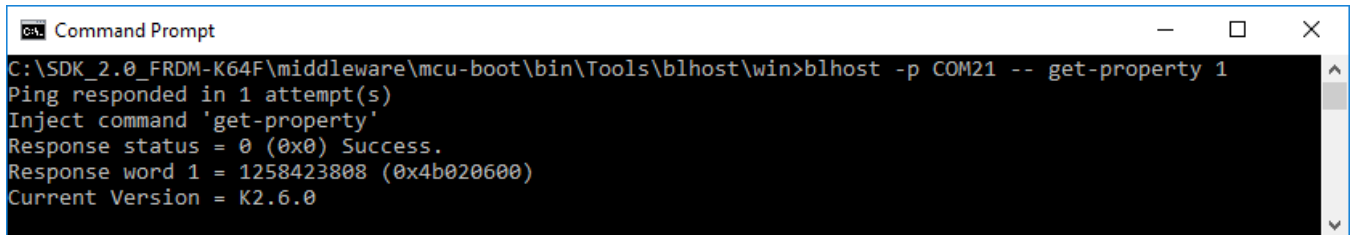
```
C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>blhost -u -- get-property 1
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258423808 (0x4b020600)
Current Version = K2.6.0
C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>
```

Figure 3. Host communication with MCU bootloader using the USB peripheral

5 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.

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 4. Back to the MCU bootloader interface

Pressing the "Reset" button allows the MCU Bootloader to again launch the led_demo application.

6 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.

6.1 Memory map overview

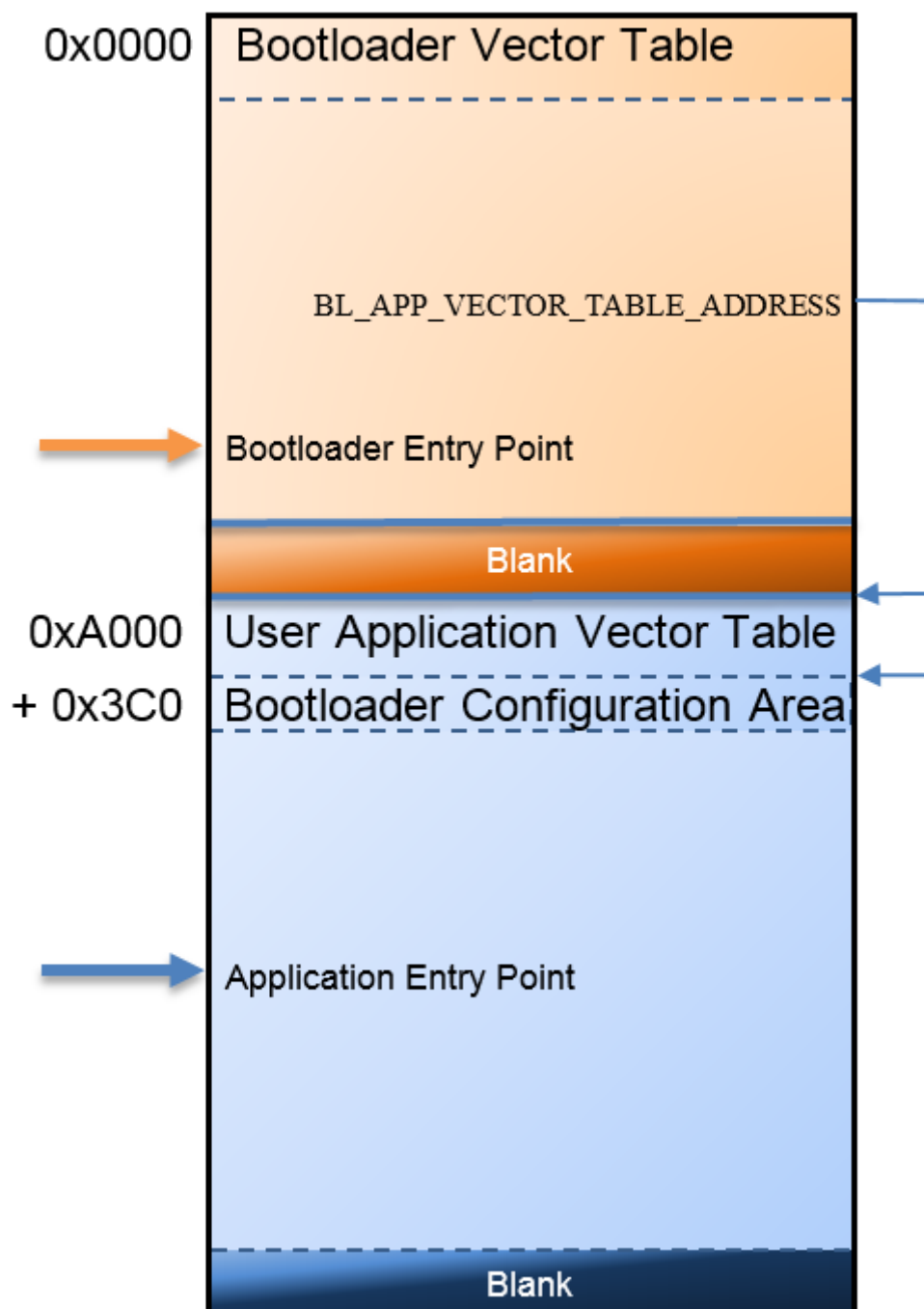


Figure 5. Device memory map

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

6.2 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).

6.3 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.

7 Appendix B - MCU Bootloader Development platforms

The KE16Z boards must be in their default factory state for jumper settings.

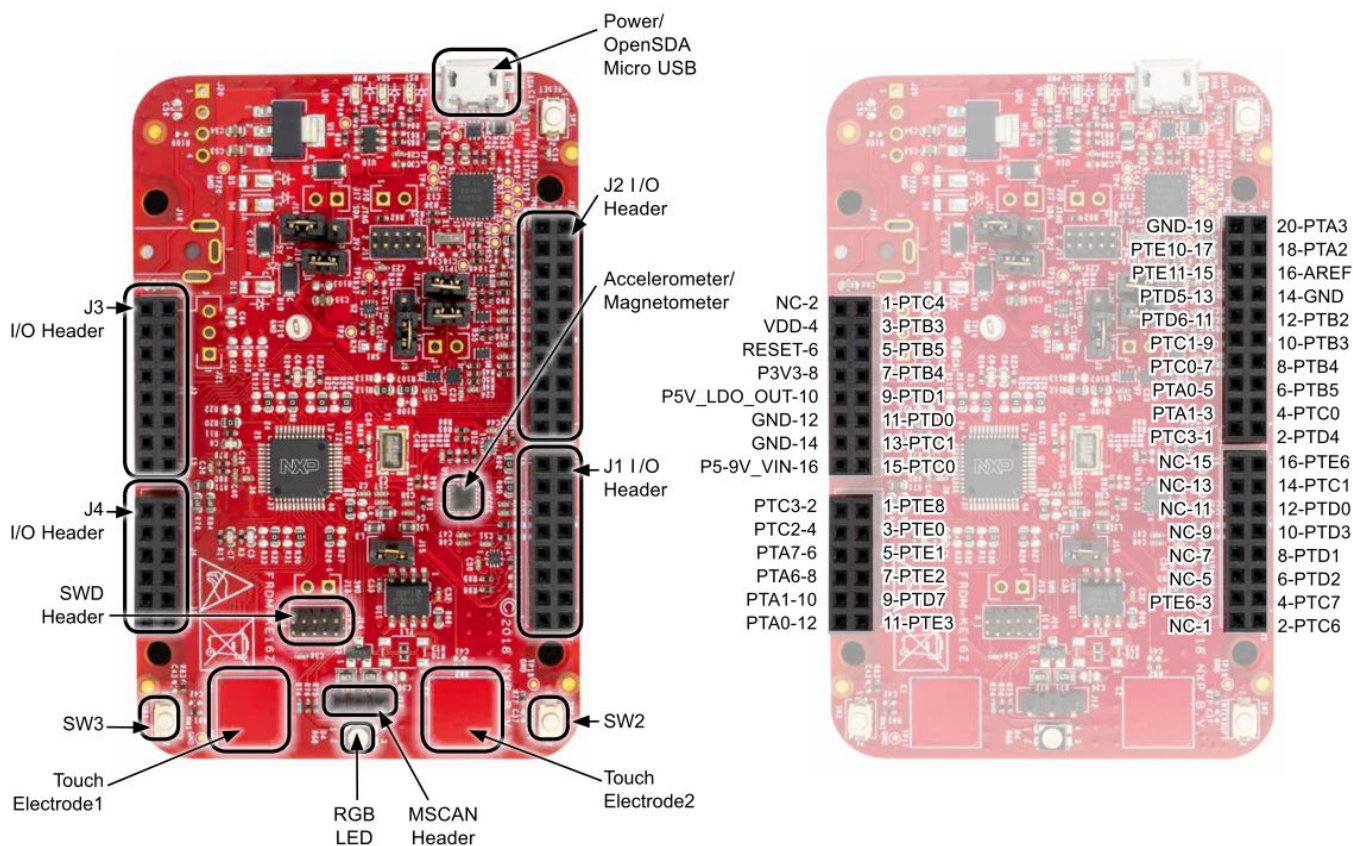


Figure 7. FRDM-KE16Z platform

8 Appendix C - MCU Bootloader Pin mappings

Table 1. KE16Z bootloader – FRDM-KE16Z

Peripheral	Instance	Port	AltMode	FRDM-KE16Z
LPUART	0	PTB0, LPUART0_RX	2	OpenSDA
		PTB1, LPUART0_TX		
LPI2C	0	PTA3, I2C0_SCL	3	J2, pin 20
		PTA2, I2C0_SDA		J2, pin 18
LPSPI	0	PTE0, SPI0_SCK	2	J4 pin 3
		PTE1, SPI0_SIN	2	J4 pin 5
		PTE2, SPI0_SOUT	2	J4, pin 7

Table continues on the next page...

Table 1. KE16Z bootloader – FRDM-KE16Z (continued)

Peripheral	Instance	Port	AltMode	FRDM-KE16Z
		PTA7, SPI0_PCS3	3	J4, pin 6
CAN	0	PTC7, CAN0_TX	5	CANH, J12, pin 3
		PTC6, CAN0_RX		CANL, J12, pin 1
Boot Pin	/	PTD2	1	SW3

9 Revision history

This table summarizes revisions to this document.

Table 2. 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
6	5/2019	KE16Z updates

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.

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.

© NXP B.V. 2019.

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: 05/2019

Document identifier: MBOOTDEMOUG

The logo for Arm, consisting of the word "arm" in a lowercase, blue, sans-serif font.