

Getting Started with the Flashloader for Kinetis

by: NXP Semiconductors

1 Introduction

This document describes how to interface with the flashloader to program a user application image into the on-chip flash for Kinetis devices. The flashloader is pre-programmed into the Kinetis device during manufacturing and enables flash programming without a debugger.

2 Overview

This document describes the steps required to program a user application image into Kinetis flash memory, utilizing the standardized MCU bootloader command interface. In the factory, the device boots from flash memory and loads the flashloader into RAM. Running from RAM, the flashloader has access to the entire flash array for placement of the user application. After the user application is programmed into flash memory, the flashloader is no longer available.

2.1 Flashloader

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

The flashloader is a specific implementation of the MCU bootloader. For the flashloader 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 flashloader to program a user application to the beginning of the Kinetis flash makes this implementation of the bootloader a one-time programming aid.

Developers creating a manufacturing flow for their hardware and software implementations may find it necessary to restore the flashloader such that the device works as it did from the NXP factory. To accomplish this, use an external debugger to program

Contents

1 Introduction.....	1
2 Overview.....	1
2.1 Flashloader.....	1
2.2 Host utilities.....	2
3 Flashloader application.....	2
3.1 Connecting to the Kinetis platform.....	2
3.2 The host utility application - blhost.....	2
3.3 Flashing the user application.....	3
3.4 The KinetisFlashTool application.....	4
4 User application: vector table offset.....	4
5 User application: Flash Configuration Area.....	4
6 Revision History.....	4



the `flashloader_loader.bin` file built from `flashloader_loader` project to the Kinetis on-chip flash. The exact method for doing this varies depending on hardware design and available tools.

2.2 Host utilities

The `blhost` utility is a command-line host program used to interface with devices running the MCU bootloader. It can list and request execution of all of the commands supported by a given Kinetis device running the bootloader. The `KinetisFlashTool` is a GUI host application that can be used to program an application image.

NOTE

These applications are released as part of MCU bootloader release package available on mcuxpresso.nxp.com. They are available in the SDK bootloader folder: `<sdk_package>/middleware/mcu-boot/bin/Tools`.

3 Flashloader application

The Kinetis platform must be connected to a host computer to interface with the MCU flashloader application. After the platform is connected, use the `blhost` or `KinetisFlashTool` application to program a user application into the Kinetis flash memory.

3.1 Connecting to the Kinetis platform

The flashloader supports UART and USB connections to a computer. See the Reference Manual for a specific device to determine the peripherals are supported by the flashloader application and the way the signals are routed to the pins of the Kinetis platform. After the Kinetis platform is powered up and there is a physical serial/USB connection between the Kinetis platform and host. The Kinetis device is ready to receive commands.

For this example, a Kinetis device is connected to a serial-to-USB converter that enumerates on a Windows® operating systems PC as a Serial Port on COMx.

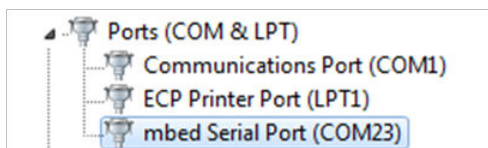


Figure 1. UART connection to MCU platform



Figure 2. Alternate UART connection to MCU platform

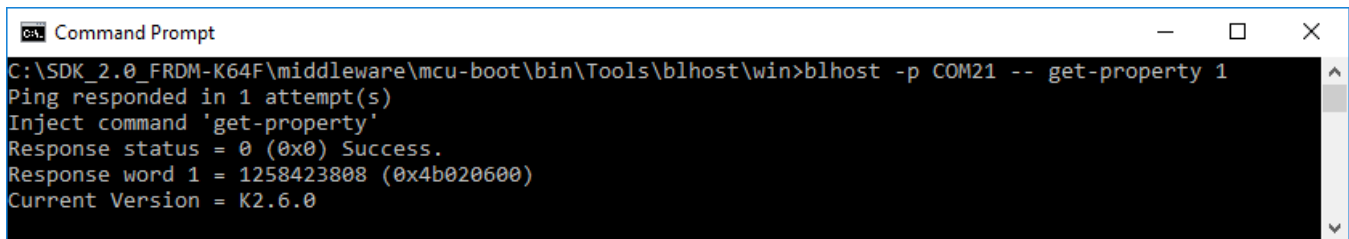
3.2 The host utility application - blhost

This section describes how `blhost` host utility program is used to communicate with the MCU bootloader.

- Open a command prompt in the directory containing blhost. For Windows OS, it is <sdk_package>/middleware/mcu-boot/bin/Tools/blhost/win.
- Type *blhost --help* to see the complete usage of the blhost utility.

Verify if the Kinetis device is connected and is running the flashloader firmware application.

- It is assumed that the Kinetis platform is fresh out of reset.
- Check under the COM port in Device Manager that the Kinetis platform is connected. In this example, the device is connected to COMx.
- Type *blhost -p COMx -- get-property 1* to get the flashloader version from the flashloader.
- Below screen shot shows that blhost is successfully communicating with the Kinetis 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

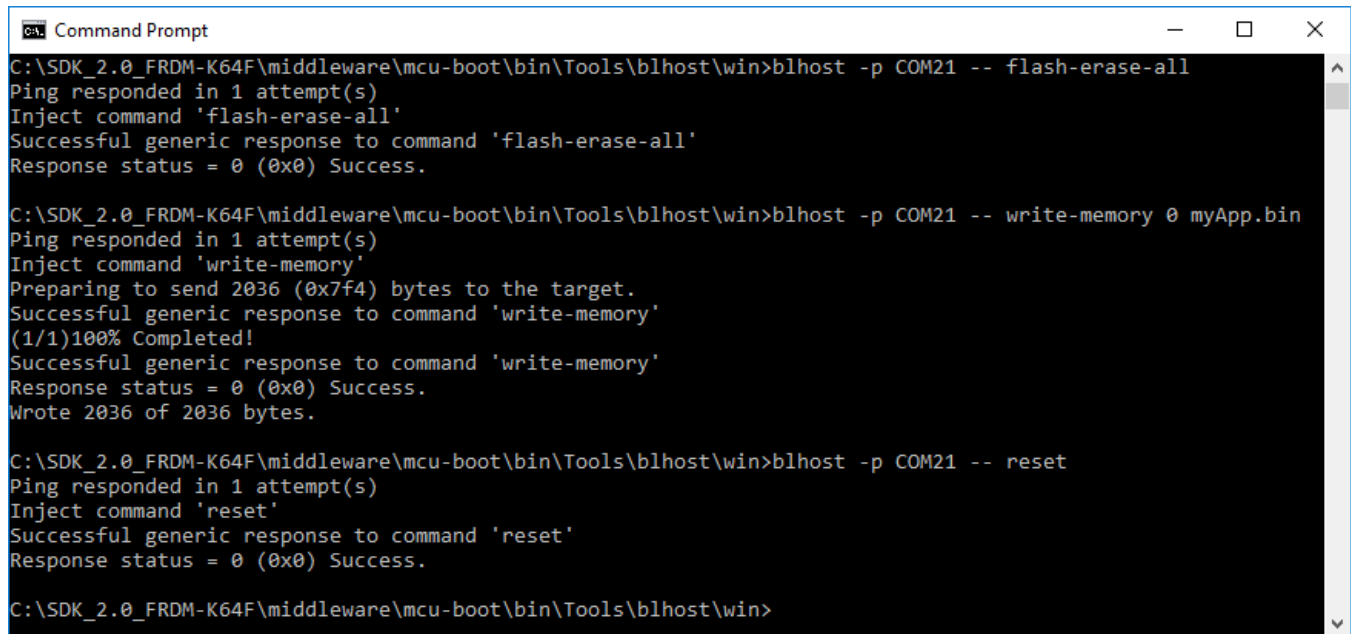
```

Figure 3. Host communication with MCU flashloader

3.3 Flashing the user application

After communication has been established between the flashloader and the host, it issue two commands to program the Kinetis flash memory with a user application.

- *blhost -p COMx -- flash-erase-all* - Erases the entire flash array.
- *blhost -p COMx -- write-memory 0 myApp.bin* – Writes the myApp.bin binary image to address 0 of the Kinetis flash memory.
- [Optional] *blhost -p COMx -- reset* – Resets the Kinetis platform and launches the user application. Note the flashloader is no longer running on the device, so further commands issued from the blhost utility fail.
- After issuing the reset command, allow 5 seconds for the user application to start running.
- Below screen shot shows the successful completion of the above commands.



```
C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>blhost -p COM21 -- flash-erase-all
Ping responded in 1 attempt(s)
Inject command 'flash-erase-all'
Successful generic response to command 'flash-erase-all'
Response status = 0 (0x0) Success.

C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>blhost -p COM21 -- write-memory 0 myApp.bin
Ping responded in 1 attempt(s)
Inject command 'write-memory'
Preparing to send 2036 (0x7f4) 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 2036 of 2036 bytes.

C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>blhost -p COM21 -- reset
Ping responded in 1 attempt(s)
Inject command 'reset'
Successful generic response to command 'reset'
Response status = 0 (0x0) Success.

C:\SDK_2.0_FRDM-K64F\middleware\mcu-boot\bin\Tools\blhost\win>
```

Figure 4. Programming a user application using the MCU flashloader

3.4 The KinetisFlashTool application

The KinetisFlashTool application can also be used to program an application to the device. See the *Kinetis Flash Tool User's Guide* (document MBOOTFLTOOLUG) for more information.

4 User application: vector table offset

“[Chapter 3, MCU flashloader application](#) discusses how to program the Kinetis flash memory with the *myApp.bin* user application. When creating the user application, the vector table of the application must be located at the beginning address of the flash memory region.

When booting from flash, the Kinetis device considers offset 0, the initial stack pointer and offset 4, the entry point for the application.

5 User application: Flash Configuration Area

The Flash Configuration Area (0x400-0x40F) must be populated with known values as per the specific Kinetis platform reference manual. In particular, values of the FSEC (0x40C) and FOPT (0x40D) locations may prevent future writes to the Kinetis flash. If user application code, other than the vector table, is linked to begin at offset 0x410, the default erased value (0xFF) of these locations makes the device secure. However, mass erase is enabled.

6 Revision History

This table summarizes revisions to this document.

Table 1. Revision history

Revision number	Date	Substantive changes
0	12/2014	Initial release
1	07/2015	Kinetis Bootloader 1.2.0 updates
2	04/2016	Kinetis Bootloader 2.0 updates
3	05/2018	MCU Bootloader v2.5.0 release
4	09/2018	MCU Bootloader v2.6.0 release

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.

© 2018 NXP B.V.

