Document Number: MCUBOOTRN
Rev. 3, December 2018

MCU Bootloader Release Notes

by: NXP Semiconductors

Contents

1 Overview

These are the release notes for the MCU bootloader v2.7.0. For more information and getting started instructions, see the Getting Started section of this document.

The MCU bootloader is an application that is programmed into the internal flash memory of an MCU device. The bootloader detects communication traffic on one of the supported peripherals (USB-HID, USB-MSC, UART, SPI, I2C, and CAN), downloads a user application, and writes the application to internal flash. The bootloader stays resident on the flash along with the user application.

This release includes the PC-hosted Kinetis Flash Tool application. This application chooses a device application image and sends the image to the bootloader over USB-HID or UART.

2 Development tools

The MCU bootloader 2.7.0 was compiled and tested with the following development tools.

Firmware projects:

- MCUXpresso IDE v10.3
- Keil MDK v5.26 with corresponding packs
- IAR Embedded Workbench for ARM® v8.30.1

NOTE

The IAR tool binary path must be added to the system environment path. For example, C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.30\arm\bin.

Host projects:

- Microsoft Visual Studio® Professional 2015 for Windows® OS Desktop
- Microsoft .NET Framework v4.5 (included in Windows OS 8)
- Microsoft Visual Studio C++ Redistributable for Visual Studio 2015 (vcredist_x86.exe)
- Python v2.7 (www.python.org)

NOTE

The Python path must be added to the system environment path. For example, C:\Python27.

- Apple Xcode® v9.2 (for tools)
- Linux® OS GNU Compiler (GCC) v5.4.0, GNU C Library(glibc) v2.23 (for tools), libraries like libstdc++6, libudev-dev, libc6, and libgcc1 may also need to be installed
- Linux OS tools have been tested on Ubuntu 16.04 LTS
- Apple Mac® OS host tools have been tested on Mac OS High Sierra 10.13.3

3 System requirements

The system requirements are based on the requirements for the development tools and the Kinetis Flash Tool application.

The recommended PC configuration is 2 GHz processor, 2 GB RAM, and 2 GB free disk space.

Windows OS applications like QCBGenerator.exe require installation of Visual C++ redistributable 2013 or greater.

.Net framework 4.5

- JP version: www.microsoft.com/ja-JP/download/details.aspx?id=30653
- US version: www.microsoft.com/en-US/download/details.aspx?id=30653

VS++ redistributable 2013

- JP version: www.microsoft.com/ja-JP/download/details.aspx?id=40784
- US version: www.microsoft.com/en-us/download/details.aspx?id=40784

4 Target requirements

This release of the MCU bootloader supports the following platforms:

- FRDM-K64F Freedom Development platform
- FRDM-KV11Z Freedom Development platform
- FRDM-KV31F Freedom Development platform
- FRDM-K82F Freedom Development platform
- FRDM-KL82Z Freedom Development platform
- FRDM-K66F Freedom Development platform
- FRDM-KL28Z Freedom Development platform
- TWR-K22F120M Tower System module
- TWR-K64F120M Tower System module
- TWR-KV46F150M Tower System module
- TWR-KV31F120M Tower System module
- TWR-K65F180M Tower System module
- TWR-KV11Z75M Tower System module
- TWR-K80F150M Tower System module
- TWR-KL82Z72M Tower System module
- TWR-KL28Z72M Tower System module

- TWR-KV58F220M Tower System module

There are no special requirements for the hardware other than what the board requires to operate.

5 Release contents

This table describes the release contents.

Table 1. Release contents

Deliverable	Location
Demo applications	<sdk_package>/middleware/mcu-boot/apps/...
Executables tools and utilities	<sdk_package>/middleware/mcu-boot/bin/...
Documentation	<sdk_package>/middleware/mcu-boot/doc/...
Bootloader source code	<sdk_package>/middleware/mcu-boot/src/...
Tool chain build projects	<sdk_package>/boards/<board>/bootloader_examples/...
Tools build projects	<sdk_package>/middleware/mcu-boot/tools/tools/...

6 Getting started

In order to use the MCU bootloader to load a user application on the Kinetis devices, see the *MCU Bootloader Demo Applications User's Guide* (document ID MBOOTDEMOUG).

For MCU Flashloader-specific information, see the *Getting Started with the MCU Flashloader* (document ID MBOOTFLASHGS)

For porting information, see Chapter 10, "MCU bootloader porting" in the *MCU Bootloader v2.7.0 Reference Manual* (document ID MCUBOOTRM).

For customization, see Chapter 11, "Create a custom flash-resident bootloader" in the *MCU Bootloader v2.7.0 Reference Manual* (document ID MCUBOOTRM).

7 Features

The bootloader release contains source code and toolchain projects for building flash-resident bootloaders and flashloaders for the supported platforms (see Section 4, “Target Requirements”). A flash-resident bootloader stays resident in flash along with the user application. The flash-resident bootloader can be used to download and program an initial application image into a blank area on the flash, and to later update the application. In contrast, a flashloader gets replaced in flash by the user application and therefore, is a one-time programming aid.

The MCU bootloader supports the following communication interfaces for downloading an application:

- USB-HID
- USB-MSD
- UART
- I2C
- SPI
- CAN

NOTE

Not all interfaces are supported on all platforms. See the individual platform reference manual for supported interfaces.

Typically, USB-HID, USB-MSC, and UART connections are made directly to a PC, whereas I2C, SPI, and CAN require additional hardware. The bootloader, running on the target platform, acts as a communication slave. The bootloader can automatically detect which peripheral is being used to download the application and, for UART and CAN, automatically detect the baud rate.

The application image is downloaded to the target through a series of command and data packets sent from a host PC or embedded host platform.

8 Host tools

The bootloader release contains source code and build projects for the following PC-based host tools:

- blhost: command line debug tool for sending individual commands to the bootloader.
- KinetisFlashTool: GUI application for downloading and flashing an application image.
- elftosb: command line tool for converting an ELF-formatted application image to SB format.
- MfgTool2: GUI application used in factory production.

For more information, see the *blhost User's Guide* (document MCUBLHOSTUG) and *Kinetis Flash Tool User's Guide* (document MBOOTFLTOOLUG), the *elftosb User's Guide* (document MBOOTELFTOSBUG), and the *MCU Bootloader Manufacturing Tool User's Guide* (document MBOOTMFGTOOLUG).

9 New features

The following new feature was introduced in this release:

- Added support for MCUXpresso IDE

10 Known issues

KDS is no longer supported. MCUXpresso IDE is now supported. The known issues are as follows:

- In MCUXpresso IDE, the debug build of flashloader projects for FRDM-KV11Z, TWR-KV11Z75M, TWR-KV46F150M can not fit. Please use release build.
- Post-action of flashloader project debug build in MCUXpresso IDE needs to be updated manually. Right-click the flashloader project and select Properties -> C/C++ Build -> Settings -> Build steps -> Post-build steps -> Command -> Edit

```
For Windows
Debug Build:
arm-none-eabi-objcopy -I elf32-littlearm -O binary "${ProjName}.axf" "${ProjName}.bin"
python ../postaction/create_fl_image_mcux.py "${ProjName}.axf" "${ProjName}.bin"
"flashloader_image.c"
mkdir -p "../../${ProjName}_loader/Debug/"
cp "flashloader_image.c" "../../${ProjName}_loader/Debug/"

For Linux:
Debug Build:
arm-none-eabi-objcopy -I elf32-littlearm -O binary "${ProjName}.axf" "${ProjName}.bin"
python ../postaction/create_fl_image_mcux.py "${ProjName}.axf" "${ProjName}.bin"
"flashloader_image.c"
```

```
mkdir -p "../../${ProjName}_loader/debug/"
cp "flashloader_image.c" "../../${ProjName}_loader/debug/"
```

- blhost command flash-erase-all is not supported by flash-resident bootloader if flash access control is set (any of FTFA_XACC register is not 0xFF).
- When using USB-MSC (drag-and-drop to an attached target), the USB connection may abort with a timeout if the SB file contains a command to erase QSPI. Use a different peripheral interface to erase the QSPI, such as USB-HID.

11 Tool notes

- A new buspal build project has not been released. Use the previous release by downloading NXP_Kinetis_Bootloader_2_0_0 package from www.nxp.com/MCUBOOT
- When changing an IAR project to generate additional output (Options->Output Converter->Generate additional output), the output filename extension (Linker->Output->Output filename) must be changed to ‘out’.
- The blhost tool accepts any speed setting in the “--buspal” option for the SPI and I2C peripherals. However, the maximum effective speed settings when using the BusPal example are approximately 300 kHz for I2C and 8000 kHz for SPI.
- When running blhost on Mac and Linux OS, the tool may be required to run as super user in order to have access to the device ports, i.e., 'sudo blhost'.
- When running blhost on Linux OS, it may be necessary to press the reset button on the target platform after the CDC device has been enumerated by the OS. This restarts the bootloader's UART autobaud operation.
- When running on Mac OS, open the "cu" device instead of the "tty" device. This prevents the OS from trying to initialize the target as a modem, i.e., 'blhost -p /dev/cu.usbmodem*'
- When using MCUXpresso IDE, setting -flto can reduce compiled code size by about 10%. However, when -flto is set, no C source will show while debugging. Because size is critical to bootloader projects, -flto is turned on for some of bootloader projects. -flto can be set by checking Properties -> C/C++ Build -> Settings -> Tool Settings -> Optimization -> Enable Link-time optimization (-flto)

Each bootloader target supports one or more of the following project types:

- tower bootloader: bootloader that executes from target flash memory on the Tower platform.
- freedom bootloader: bootloader that executes from target flash memory on the Freedom platform.
- maps bootloader: bootloader that executes from target flash memory on the MAPS platform.
- flashloader: bootloader that executes from target RAM memory on either the Freedom, Tower, or MAPS platform.
- flashloader_loader: bootstrap loader that executes from flash memory on either the Freedom or Tower platform. This loader copies an image of the flashloader into RAM, then executes the flashloader from RAM.

The flashloader_loader project uses the output of the flashloader build to create the flashloader image to load into RAM. For this reason, the flashloader project must be built before building the flashloader_loader project. Also, install Python27 for successful flashloader image generation.

When debugging flashloader with ARM Keil MDK compiler, there is no need to use the load button. The "Start/Stop Debug Session" button correctly downloads the flashloader image into the target RAM address and starts executing it. However, the flashloader_loader and flash-resident bootloader are required to use the load button to properly program the internal flash.

There is an intermittent issue with ARM Keil MDK when it uses J-Link to load the image. An error message pops up indicating flash download failed. The workaround to this issue is to load the Keil-built image separately, using J-Link, and not through Keil.

12 Revision history

The following table contains a history of changes made to this document.

Table 2. Revision history

Revision number	Date	Substantive changes
0	04/2016	MCU bootloader v2.0.0 initial release
1	05/2018	MCU Bootloader v2.5.0 release
2	09/2018	MCU Bootloader v2.6.0 release
3	11/2018	MCU Bootloader v2.7.0 release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: December 2018
Document identifier: MCUBOOTRN

arm