MCUBOOTRN

MCU Bootloader Release Notes

Rev. 10 — 01 June 2021 Release Notes

Contents

1 Overview

This release note is for the MCU bootloader. For more information and getting started instructions, see the Getting Started section of this document.

The MCU bootloader is an application programmed into the internal flash memory or RAM 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 is compiled and tested with the following development tools.

Firmware projects:

- MCUXpresso IDE v11.3
- MDK-Arm Microcontroller Development Kit (Keil)® 5.33 with corresponding packs
- IAR Embedded Workbench for Arm® v8.30.9
- Makefiles support with GCC revision 9-2020-q2-update GCC9 from Arm Embedded

NOTE

Add IAR tool binary path to the system environment path. For example, C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.50\arm\bin.

- Python v2.7 (www.python.org)

NOTE

Add Python path to the system environment path. For example, C:\Python27.

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-K22F Freedom Development platform
- FRDM-KV31F Freedom Development platform
- FRDM-K66F Freedom Development platform
- FRDM-K32L2A4S Freedom Development platform
- TWR-KM35Z75M Tower System module
- FRDM-KE16Z Freedom Development platform

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
Executables tools and utilities	<sdk_package>/middleware/mcu-boot/bin/Tools/...
Bootloader source code	<sdk_package>/middleware/mcu-boot/src/...
MCU bootloader examples	<sdk_package>/boards/<board>/bootloader_examples/...

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 Reference Manual* (document ID MCUBOOTRM).

For customization, see Chapter 11, "Create a custom flash-resident bootloader" in the *MCU Bootloader 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 resides in flash along with the user application. The flash-resident bootloader downloads and then programs 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

Each platform does not support all interfaces. See the individual platform reference manual for supported interfaces.

Typically, USB-HID, USB-MSC, and UART connect 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 detects the peripheral which is being used to download the application. If the peripherals being used are UART and CAN, it detects the baud rate too.

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:

- KinetisFlashTool: GUI application for downloading and flashing an application image.
- MfgTool2: GUI application used in factory production.

Host tools need to be used with MCUBOOT can be downloaded from nxp.com/MCUBOOT.

- blhost: command-line debug tool for sending individual commands to the bootloader.
- elftosb: command-line tool for converting an ELF/SREC formatted application image to SB format.

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 Known issues

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

- In MCUXpresso IDE, post-action of flashloader project debug build 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/MAC**

  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 the FTFA_XACC registers 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.
- On Mac OS, for frdm-k82, reliable-update command over usb is successful but returns incorrect status.

10 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 UART autobaud operation of the bootloader.
- 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. So, 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 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.

11 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
4	06/2019	Updated Known Issue
5	10/2019	MCU Bootloader v2.7.0 with MCUXpresso SDK2.7.0 release
6	01/2020	Added a platform FRDM-K32L2A4S in "Target Requirements"
7	03/2020	Added a platform TWR-KM35Z75M in "Target Requirements"
8	07/2020	Updated for MCUXpresso SDK v2.8.0
9	24 December 2020	Updated for MCUXpresso SDK v2.9.0
10	01 June 2021	Updated for MCUXpresso SDK v2.10.0

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.

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.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX,EMBRACE, GREENCHIP, HITAG, 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, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, 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. 2016-2021. **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: 01 June 2021

Document identifier: MCUBOOTRN

arm