**NXP Semiconductors**
User's Guide

Document Number: KTSFSCIBQSUG
Rev 6, 01/2018

# Kinetis Thread Stack and FSCI Bootloader Quick Start Guide

# Contents

**Chapter 1 Introduction.....3**
1.1 Deploying standalone bootloader firmware.....3
1.2 Folder locations.....4
1.3 Building Thread applications binaries with bootloader offset with IAR IDE.....4
1.4 Building Thread applications binaries with bootloader offset with MCUXpresso.....6

**Chapter 2 Entering FSCI bootloader mode.....8**

**Chapter 3 Using the Host SDK sample with the FSCI bootloader.....9**

**Chapter 4 Revision history.....10**

**Chapter 5 Copyright.....11**

# Chapter 1
# Introduction

Kinetis Thread Stack integrates with an Over-the-Wire (serial bus) bootloader with support for UART and SPI that uses the FSCI bus protocol. This process allows host devices to use the same serial bus protocol to drive both the bootloader and the THCI (Thread Host Controlled Interface) used by the Host Controlled Device examples.

## 1.1 Deploying standalone bootloader firmware

To deploy the standalone FSCI bootloader to the FRDM-KW41Z, choose one of the following options:

- From IAR ® EWARM, deploy the Debug or Release configuration of the following workspace: **boards\frdmkw41z \wireless_examples\framework\bootloader_fsci\bm\iar\bootloader_fsci_bm.eww** to the board using OpenSDA or J-Link pod debuggers. Make sure to select the appropriate driver under Options -> Debugger -> Setup -> Driver.
- From MCUXpresso IDE, deploy the Debug or Release configuration of the following workspace: **boards\frdmkw41z \wireless_examples\framework\bootloader_fsci\bm\bootloader_fsci_bm.xml** to the board using OpenSDA or J-Link pod debuggers
- Copy the **\tools\wireless\binaries\bootloader_fsci_frdmkw41z.bin** file to the FRDM-KW41Z mass storage device emulated via OpenSDA
- Using a J-Link pod and a tool such as SEGGER J-Link Tools, IAR EWARM or NXP Connectivity Test Tool for Kinetis Firmware Loader, erase the MCU and then load the **\tools\wireless\binaries\bootloader_fsci_frdmkw41z.bin** file to device flash.

If the standalone bootloader is loaded to the first flash sector and there is an application firmware following in the flash, by default the bootloader will load the application, instead of waiting for new firmware. In order to return to the FSCI bootloader mode, one must use one of the options presented in Section 6 *Entering FSCI Bootloader Mode*.

To configure a standalone bootloader to automatically wait for new firmware, one must rebuild the bootloader binary file in after setting the **gBootFlags** attribute to value 0x00UL in the file: **\boards\frdmkw41z\wireless_examples\framework \bootloader_fsci\src\bootloader\main.c.**

Default value of **gBootFlags**:

```
#pragma location = "BootFlags"
__root volatile const bootFlags_t gBootFlags =
{
0xFFFFFFFFFFFFFFFF
};
```

should be updated to

```
#pragma location = "BootFlags"
__root volatile const bootFlags_t gBootFlags =
{
0x00UL
};
```

## 1.2 Folder locations

The following folder tree locations relative to the Thread package installation path contain artifacts showcasing the use of the FSCI bootloader:

- **\boards\<board>\wireless_examples\thread\host_controlled_device** contains IAR and MCUXpresso example projects for the Host Controlled Device running on FRDM-KW41Z and USB-KW41Z
- **\tools\wireless\binaries** contains binary image files for the bootloader variants of the KW41Z: FSCI and OTA update bootloaders
- **\boards\<board>\wireless_examples\framework\bootloader_fsci** contains a common source tree and IAR and MCUXpresso projects for the bare-metal FSCI bootloader
- **\tools\wireless\host_sdk\hsdk-python\src\com\nxp\wireless_connectivity\test\bootloader** contains a Python language example script integrated with the Host SDK for Linux® OS or Windows® OS that shows how to drive the FSCI bootloader from a host system
- **\tools\wireless\host_sdk\hsdk\demo** contains a C language example program integrated with the Host SDK for Linux® OS that shows how to drive the FSCI bootloader from a host system
- **\docs\Kinetis FSCI Host Application Programming Interface User's Guide.pdf** contains a reference to deploying the Host SDK for using samples with a KW41Z development board running the Thread Host Controlled Device Application firmware

## 1.3 Building Thread applications binaries with bootloader offset with IAR IDE

To build a Thread application in IAR® IDE, which is offset to load starting at second sector of the flash, thus making the binary compatible with the FSCI bootloader, ensure the IAR Project Options for the Linker configuration match the settings below.

To build a bootloader-compatible application, for the Host Controlled Device, start with the workspace in: **\boards\frdmkw41z\wireless_examples\thread\host_controlled_device\freertos\iar** and ensure gUseBootloaderLink_d is set to 1.

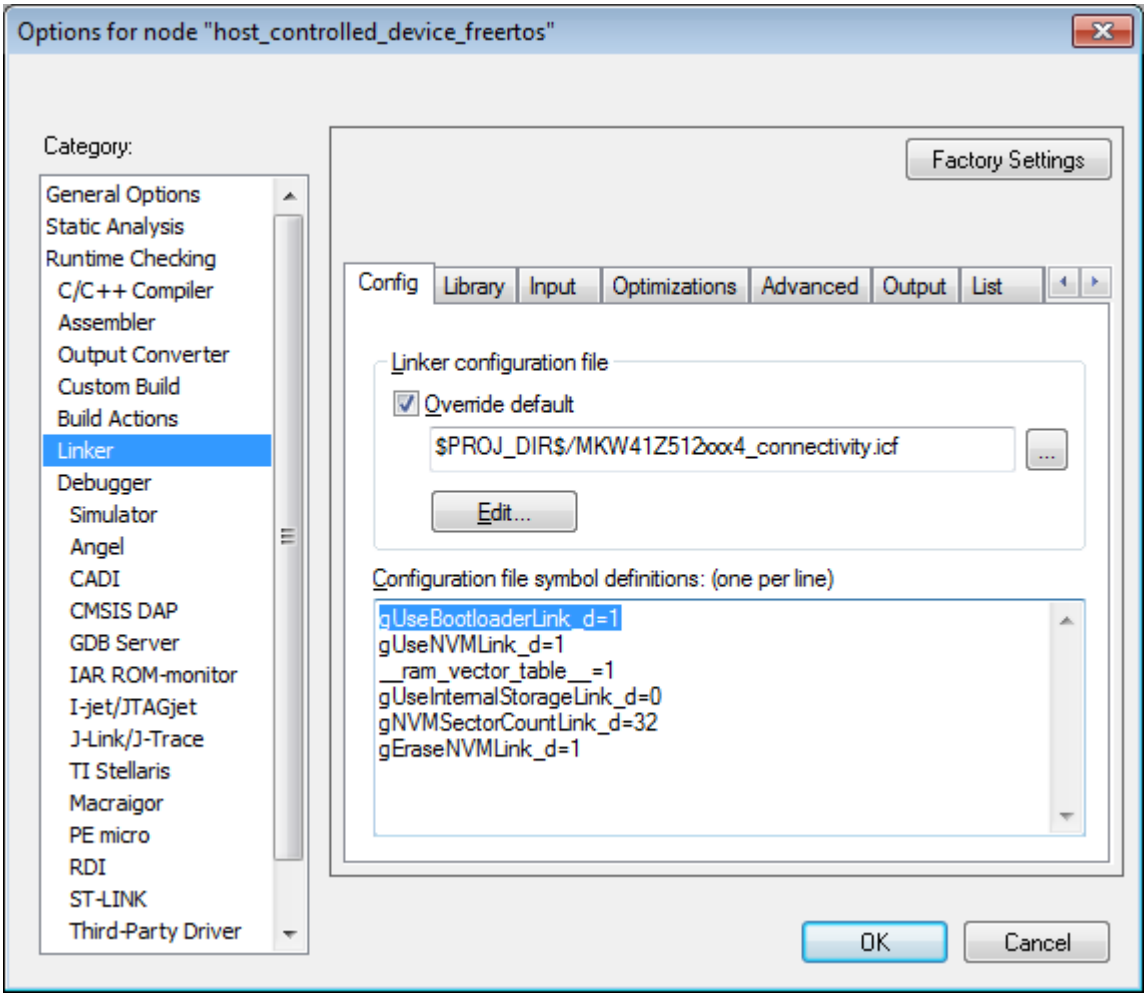


**Figure 1. IAR Linker Configuration Options**

- In the **Project Options -> Output Converter** switch the **Output format** to **binary**:

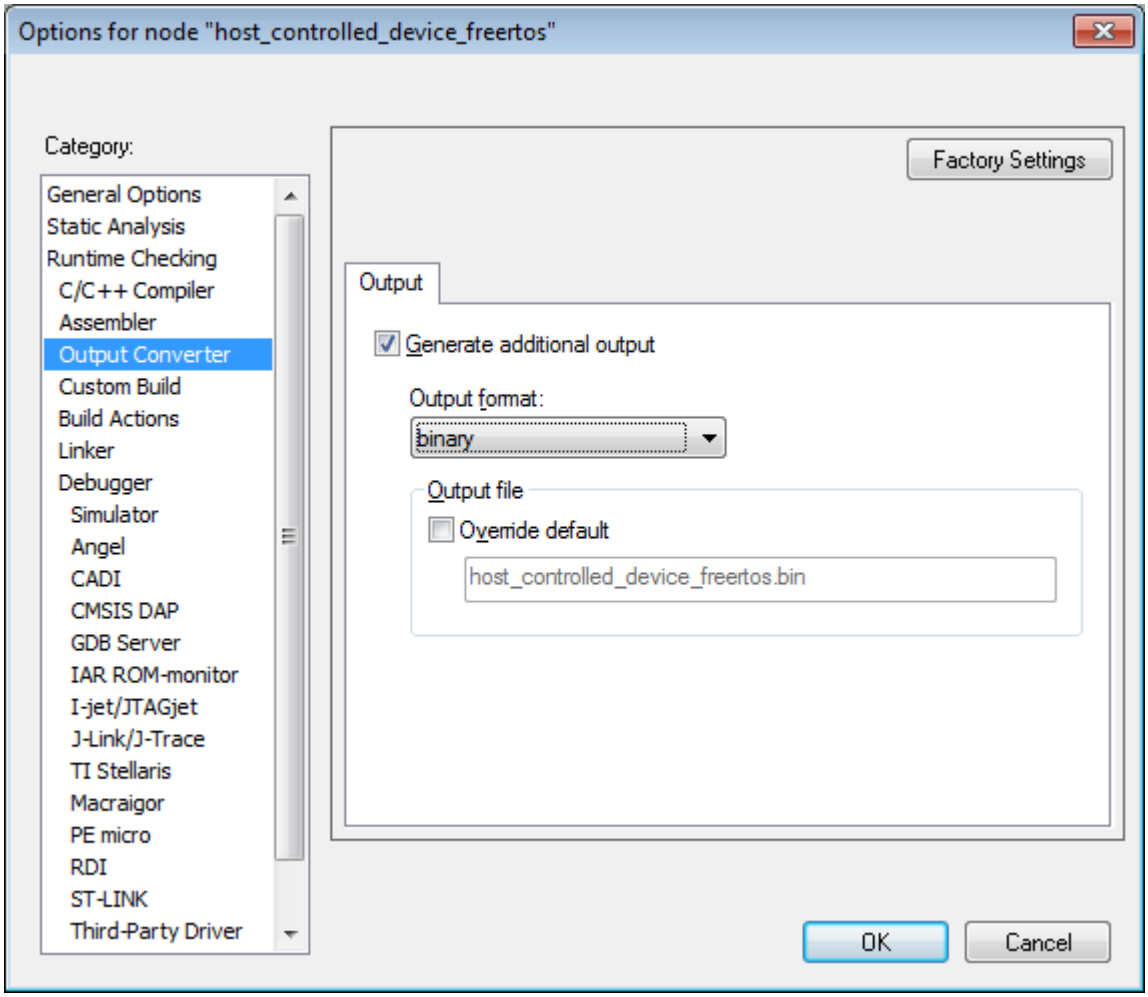


**Figure 2. Project Options - Output Convertor**

- Build the project, the bootloader compatible *.bin file is generated in the workspace folder, for example, at: **\workspace \frdmkw41z_wireless_examples_thread_host_controlled_device_freertos\Debug**

# 1.4 Building Thread applications binaries with bootloader offset with MCUXpresso

To build a Thread application in MCUXpresso IDE, which loads starting at the second sector of the flash, making the binary compatible with the FSCI bootloader, ensure the MCUXpresso Project Options for the Linker configuration match the settings below.

To build a bootloader-compatible application, for the Host Controlled Device, start with the workspace in: \boards\frdmkw41z \wireless_examples\thread\host_controlled_device\freertos\. Dynamic configuration of the linker files is not supported, thus one must use a predefined working .ld file of a bootloader-compatible application. The user is requested to overwrite MKW41Z512xxx4_connectivity.ld with the one under boards\frdmkw41z\wireless_examples\thread\reed_ota_client\freertos.

- In the **Properties -> C/C++ Build -> Settings -> Build Steps -> Post-build steps** click on **Edit…** and uncomment the second line:

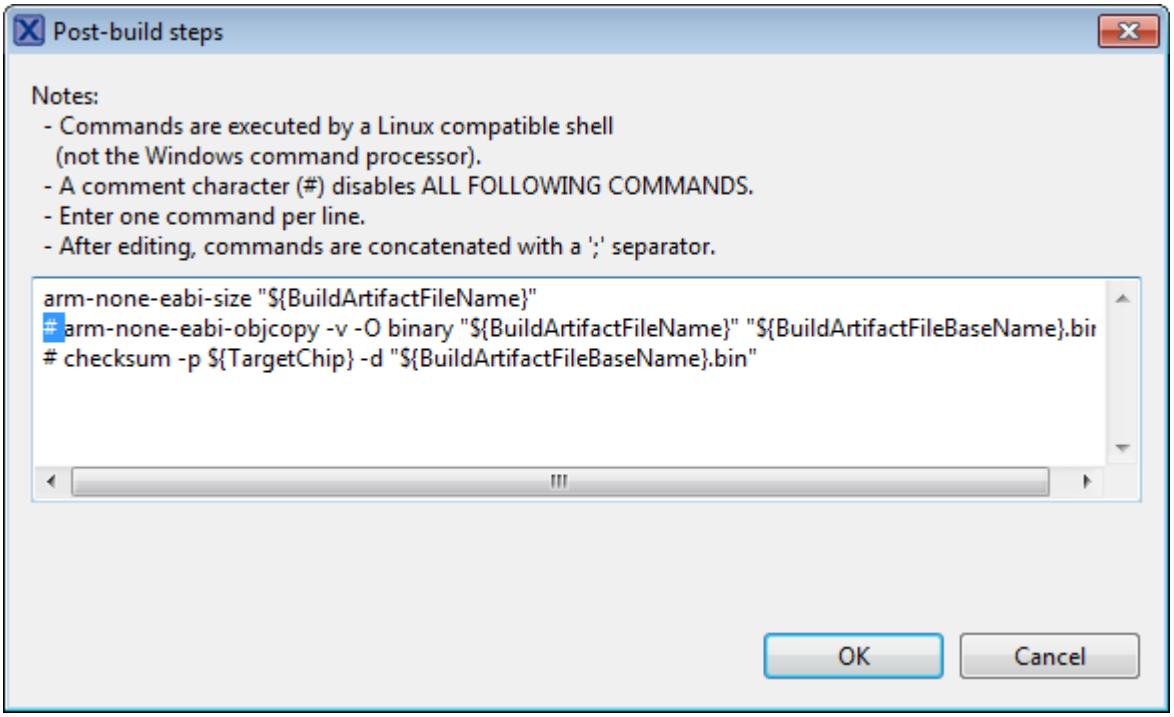


**Figure 3. Convert Image to Binary**

- Build the project, the bootloader compatible *.bin file is generated in the workspace folder, for example, at: **\workspace \frdmkw41z_wireless_examples_thread_host_controlled_device_freertos\Debug**

# Chapter 2
# Entering FSCI bootloader mode

After a valid application is programmed starting with the second flash sector, the bootloader jumps to the application automatically and the bootloader mode must be triggered specifically.

To trigger bootloader mode, do the following:

- FRDM-KW41Z: hold SW1(RST) and press SW4, then release SW1 first and SW4 second
- USB-KW41Z: hold SW2 (RST) and press SW1, then release SW2 first and SW1 second
- If THCI is active in the Thread firmware, send the command to enter bootloader mode over the THCI/FSCI serial bus; this command has:
  - THCI/FSCI OpCode Group: 0xA3
  - THCI/FSCI OpCode: 0xCF

When receiving the command, the device auto-resets and remains in bootloader mode.

# Chapter 3
# Using the Host SDK sample with the FSCI bootloader

To deploy the Host SDK to a host Linux system OS that works with the Thread THCI (Host Control Interface/Device), follow the deployment procedure in the **docs\Kinetis FSCI Host Application Programming Interface.pdf**, then run the script at: **\tools\wireless\host_sdk\hsdk-python\src\com\nxp\wireless_connectivity\test\bootloader fsci_bootloader.py** providing as command line arguments the device serial port and a binary firmware file compatible with the bootloader that has been built as shown in the section above. For example,

#export PYTHONPATH=$PYTHONPATH:<hsdk-path>/hsdk-python/src/

#python fsci_bootloader.py /dev/ttyACM0 thread_host_controlled_device.bin

The script does the following:

- Sends the THCI command for the device to reset, enter and remain in bootloader mode.
- Sends the command to cancel an image as a safety check and to verify if the bootloader is responsive.
- Sends the command to start firmware update for a new image.
- Pushes chunks of the firmware image file sequentially until the full firmware is programmed and displays intermediate progress as percent of binary file content loaded.
- Sets the flags to commit the image as valid.
- Resets the device to ensure it boots to the new firmware. Additionally, one may use the C mirror of the script, located at **hsdk/demo/FsciBootloader.c** . The command line arguments are the same used for the Python script.

# Chapter 4
# Revision history

This table summarizes revisions to this document.

**Table 1. Revision history**

| Revision number | Date | Substantive changes |
|---|---|---|
| 0 | 03/2016 | Initial release |
| 1 | 04/2016 | Updates for the Thread KW41 Alpha Release |
| 2 | 08/2016 | Updates for the Thread KW41 Beta Release |
| 3 | 09/2016 | Updates for the Thread KW41 GA Release |
| 4 | 12/2016 | Updates for the Thread KW24D GA Release |
| 5 | 03/2017 | Updates for the Thread KW41Z Maintenance Release with Support for MCUXpresso |
| 6 | 01/2018 | Updates for the Thread KW41Z Maintenance Release |

# Chapter 5
# Copyright

***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.