Device Firmware Update Sample Application Guide

Rev. 1.0 31-Mar-2023

KAGA FEI

Table of Contents

Revision History

Revision	Date	Comment
1.0	31-Mar-2023	Initial version

1 Introduction

This document describes the DFU (Device Firmware Update) sample Application applicable to KAGA FEI's BLE module.

2 Overview

2.1 Tool

It is assumed that the following are used in the application of DFU sample Application.

- Nordic SDK
- nRF Util ※Use Python3.7 ~ Pip in the process of installing
- Segger Embedded Studio and NRF board support package (download here to a quickstart guide『SES_NRF52_Quickstart_Guide_X_XX_JP.pdf』 for NRF on Segger Embedded Studio for installation and setup instructions).
- firmware writing tool (nRF Command Line Tools/ nRF Programmer)
- Supported KAGA FEI BLE module

Module	Chip
EC2820AA2	nRF52820
ES2820AA2	nRF52820
EC2832AAx	nRF52832
EJ2832AAx	nRF52832
ES2832AAx	nRF52832
EB2833AAx	nRF52833
ED2833AA2	nRF52833
EJ2833AA2	nRF52833
EB2840AAx	nRF52840
ED2840AAx	nRF52840
EJ2840AAx	nRF52840

- J-link Lite
  It is sold as an evaluation kit together with EVB. Please purchase as needed.

2.2 Bootloader

The DFU Sample Application Bootloader includes a DFU function, and it is possible to shift to DFU processing by performing specific operations. Please refer to the following Nordic URL for details of the configuration.

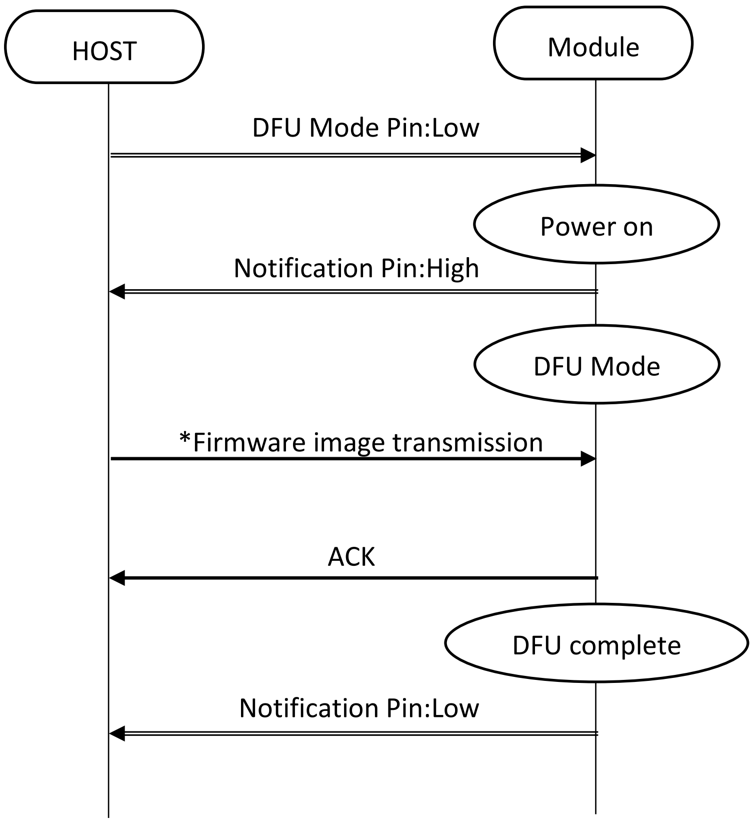
Reference URL：
https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.1.0/lib_bootloader_modules.html

※In this sample application, mbr is written because it is based on SDK 17.1.0, but please write softdevice used at your own risk.

2.3 Bootloader flowchart

Bootloader processes according to the following flowchart. Check the table for the notification pins of each module and the pins that enter DFU Mode. For details on how to change the Pin used, refer to "4.2.3 DFU Pin change".

[flowchart]



Module	DFU Mode Pin	Notification Pin
EC2820AA2	P0.14	P0.03
ES2820AA2	P0.14	P0.03
EC2832AAx	P0.18	P0.20
EJ2832AAx	P0.18	P0.20
ES2832AAx	P0.18	P0.20
EB2833AAx	P0.28	P0.30
ED2833AA2	P0.28	P0.30
EJ2833AA2	P0.28	P0.30
EB2840AAx	P0.25	P0.20
ED2840AAx	P0.28	P0.29
EJ2840AAx	P0.28	P0.29

3 DFU structure

3.1 DFU

In the DFU Sample Application, it is set to write the MBR, Bootloader and its setting value area to FLASH.

Reference URL：https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.1.0/lib_bootloader.html

3.2 Write FLASH region

Regarding the writing operation of DFU Sample Application, normal writing is performed in Dual Bank, but if the region is insufficient, change to Single Bank and operate. See Nordic Info Center below for details.

Reference URL：
https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.1.0/lib_bootloader_dfu_banks.html

3.3 DFU packets

Please refer to the following Nordic Info Center for details of communication packets used in DFU.

Reference URL：
https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.1.0/lib_dfu_transport.html

※The above URL is SDK17.1.0. Please check according to your version.

3.4 Private Key File

DFU uses a private key file to prevent third parties from writing.

3.4.1 Tool installation for private key file creation

The private key file used in DFU is created using Nordic “nRF Util”. For details on installation, refer to the following URL.

URL：https://infocenter.nordicsemi.com/topic/ug_nrfutil/UG/nrfutil/nrfutil_installing.html

3.4.2 Private key file creation procedure

Follow the steps below to create a private key file, and generate a source file embedded in the DFU Sample Application from the private key file. Copy the created file (ex: dfu_public_key.c) directly under the unzipped folder.

[private key creation]

Command：*nrfutil keys generate XXXXXXX.pem*

※The created private key file is used for rewriting the firmware. Please be careful about the management method.

[Bootloader Create source file]

Command：*nrfutil keys display --key pk --format code XXXXXXX.pem --out_file dfu_public_key.c*

※If you change the output file name, please also change the file set in the project

4 Compile

4.1 Nordic SDK

DFU Sample Application uses Nordic SDK. Download and decompress the following version of Nordic SDK and copy the DFU Sample Application folder under " nRF5_SDK_17.1.0_ddde560 \ examples".

The following SDK version is used in DFU Sample Application, but please consider the latest version when using it for development.

Also, copy the source file created in “3.4.2 Private key file creation procedure” to the DFU Sample Application folder.

SDK Version：17.1.0 nRF5 SDK

Download URL：https://www.nordicsemi.com/Software-and-tools/Software/nRF5-SDK/Download#infotabs

※Copyright of Nordic source code belongs to Nordic

4.2 UART

In the DFU Sample Application, the UART Pin is set according to the following specifications so that it can be connected to a PC via EVB and evaluated. The UART communication specifications are set as follows in accordance with the Nordic SDK specifications. For details on how to change the Pin used, refer to "4.2.3 DFU Pin change".

[Module UART Pin Assignment]

Module	RX	TX	CTS	RTS
EC2820AA2	P0.04	P0.05	P0.06	P0.07
ES2820AA2	P0.08	P0.06	P0.07	P0.05
EC2832AAx	P0.08	P0.06	P0.07	P0.05
EJ2832AAx	P0.08	P0.06	P0.07	P0.05
ES2832AAx	P0.08	P0.06	P0.07	P0.05
EB2833AAx	P0.08	P0.06	P0.07	P0.05
ED2833AA2	P0.08	P0.06	P0.07	P0.05
EJ2833AA2	P0.08	P0.06	P0.07	P0.05
EB2840AAx	P0.08	P0.06	P0.07	P0.05
ED2840AAx	P0.08	P0.06	P0.07	P0.05
EJ2840AAx	P0.08	P0.06	P0.07	P0.05

[UART specification]

Baud rate	115200bps
Data	8bit
Parity	None
Stop	1bit
Hardware flow control	Enable

4.2.1 Change UART Pin

When changing the Pin used for UART communication, it becomes possible by changing the following definition.

[UART Pin Setting Source code (nrf_dfu_serial_uart.c)]

```
    nrf_drv_uart_config_t uart_config = NRF_DRV_UART_DEFAULT_CONFIG;

    uart_config.pseltxd   = TX_PIN_NUMBER;
    uart_config.pselrxd   = RX_PIN_NUMBER;
    uart_config.pselcts   = CTS_PIN_NUMBER;
    uart_config.pselrts   = RTS_PIN_NUMBER;
    uart_config.hwfc      = NRF_DFU_SERIAL_UART_USES_HWFC ?
                                NRF_UART_HWFC_ENABLED : NRF_UART_HWFC_DISABLED;
    uart_config.p_context = &m_serial;

    err_code =  nrf_drv_uart_init(&m_uart, &uart_config, uart_event_handler);
```

[UART Pin Definition (pca10040.h/pca10056.h/pca10100.h/module_config.h)]

```
#define RX_PIN_NUMBER  8
#define TX_PIN_NUMBER  6
#define CTS_PIN_NUMBER 7
#define RTS_PIN_NUMBER 5
#define HWFC           true
```

※Note that it is referenced in other projects.

Module	File
EC2820AA2	module_config.h
ES2820AA2	pca10100.h
EC2832AAx	pca10040.h
EJ2832AAx	pca10040.h
ES2832AAx	pca10040.h
EB2833AAx	pca10100.h
ED2833AA2	pca10100.h
EJ2833AA2	pca10100.h
EB2840AAx	pca10056.h
ED2840AAx	pca10056.h
EJ2840AAx	pca10056.h

※Only EC2820AA2 defines UART Pin in module_config.h.

4.2.2 UART Baud rate

To change the Baud rate used for UART communication, change the following definitions.

[UART Baud rate Setting(nrf_drv_uart.h)]

```
/**@brief UART default configuration. */
#define NRF_DRV_UART_DEFAULT_CONFIG                                                  \
{                                                                                    \
    .pseltxd            = NRF_UART_PSEL_DISCONNECTED,                                \
    .pselrxd            = NRF_UART_PSEL_DISCONNECTED,                                \
    .pselcts            = NRF_UART_PSEL_DISCONNECTED,                                \
    .pselrts            = NRF_UART_PSEL_DISCONNECTED,                                \
    .p_context          = NULL,                                                      \
    .hwfc               = (nrf_uart_hwfc_t)UART_DEFAULT_CONFIG_HWFC,                 \
    .parity             = (nrf_uart_parity_t)UART_DEFAULT_CONFIG_PARITY,             \
    .baudrate           = (nrf_uart_baudrate_t)UART_DEFAULT_CONFIG_BAUDRATE,         \
    .interrupt_priority = UART_DEFAULT_CONFIG_IRQ_PRIORITY,                          \
    NRF_DRV_UART_DEFAULT_CONFIG_USE_EASY_DMA                                         \
}
```

[UART Baud rate Definition ("config_32" or "config_40" or "config_20" or "config_33"/sdk_config.h)]

```
// <o> UART_DEFAULT_CONFIG_BAUDRATE  - Default Baudrate

// <323584=> 1200 baud
// <643072=> 2400 baud
// <1290240=> 4800 baud
// <2576384=> 9600 baud
// <3862528=> 14400 baud
// <5152768=> 19200 baud
// <7716864=> 28800 baud
// <10289152=> 38400 baud
// <15400960=> 57600 baud
// <20615168=> 76800 baud
// <30801920=> 115200 baud
// <61865984=> 230400 baud
// <67108864=> 250000 baud
// <121634816=> 460800 baud
// <251658240=> 921600 baud
// <268435456=> 1000000 baud

#ifndef UART_DEFAULT_CONFIG_BAUDRATE
#define UART_DEFAULT_CONFIG_BAUDRATE 30801920
#endif
```

※Change the value to be changed referring to the value described in the comment column. The definition file depends on the module used.

4.2.3 DFU Pin change

The pins that enter the DFU mode and the pins that notify the status of the DFU are defined below. When changing, change the following definition.

[Pin Definition (module_config.h)]

```
#define DFU_ENTER_PIN           18
#define DFU_NOTICE_PIN          20
```

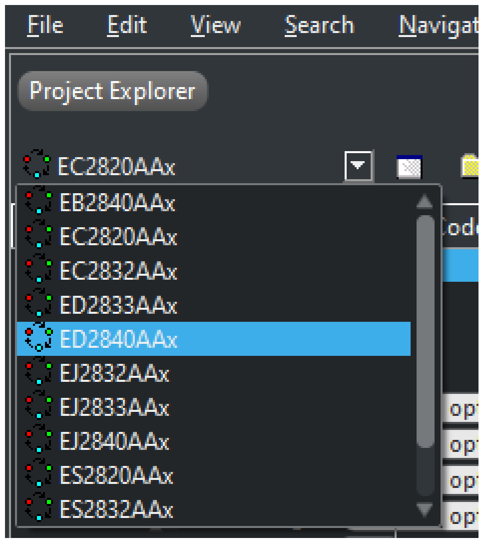
4.3 micro-ecc

To compile Sample software, micro-ecc is required. Please create a library according to Nordic's procedure below

URL：
https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.1.0/lib_crypto_backend_micro_ecc.html*
Note micro-ecc is used with nRF52832, nRF52820 and nRF52833.

4.4 Module Selection

Supported modules have specific configurations for each project. By using the Segger Embedded Studio drop-down list to select the target for compiling the application, the relevant settings required for the application are loaded into the project.



5 Write the firmware

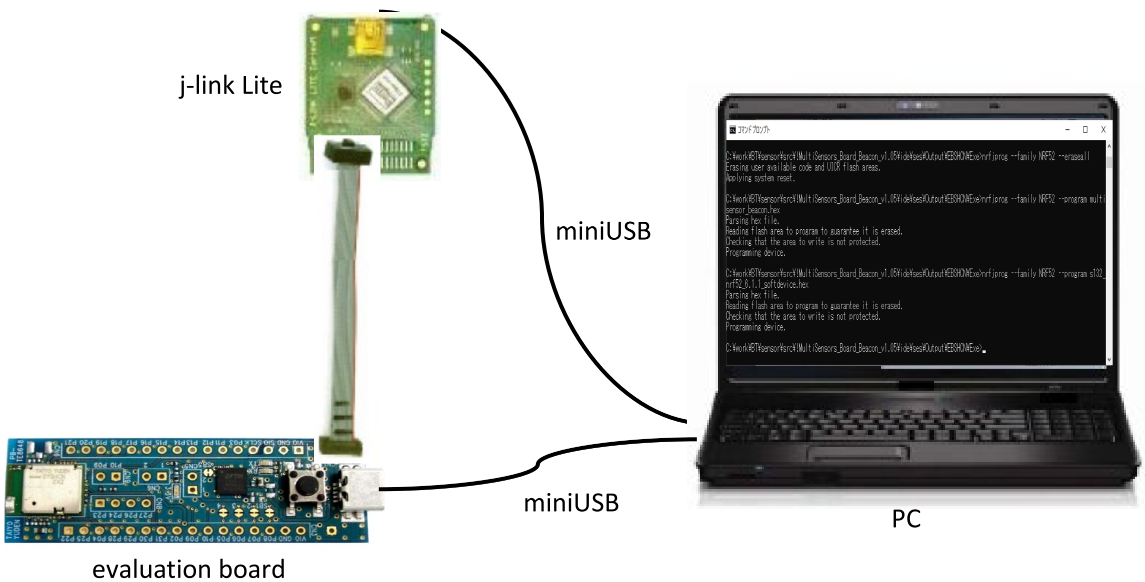
5.1 Writing using J-link

5.1.1 Connecting the evaluation board and PC

When writing the firmware, connect the PC and j-link Lite via USB as shown below, and also connect the evaluation board via USB to supply power.

For the following writing procedure, perform either method with the PC and module connected.

【Connection example】



5.1.2 Writing with nRF Command Line Tools

Execute the command in the following procedure to write the firmware to the evaluation board.

1. Erase of FLASH region

   ・Command：*nrfjprog --family NRF52 --recover*

   ※FLASH region must be deleted before writing.

2. Writing SoftDevice/Mastr Boot Record(MBR)

   ・Command：*nrfjprog --family NRF52 --program mbr_nrf52_2.4.1_mbr.hex*

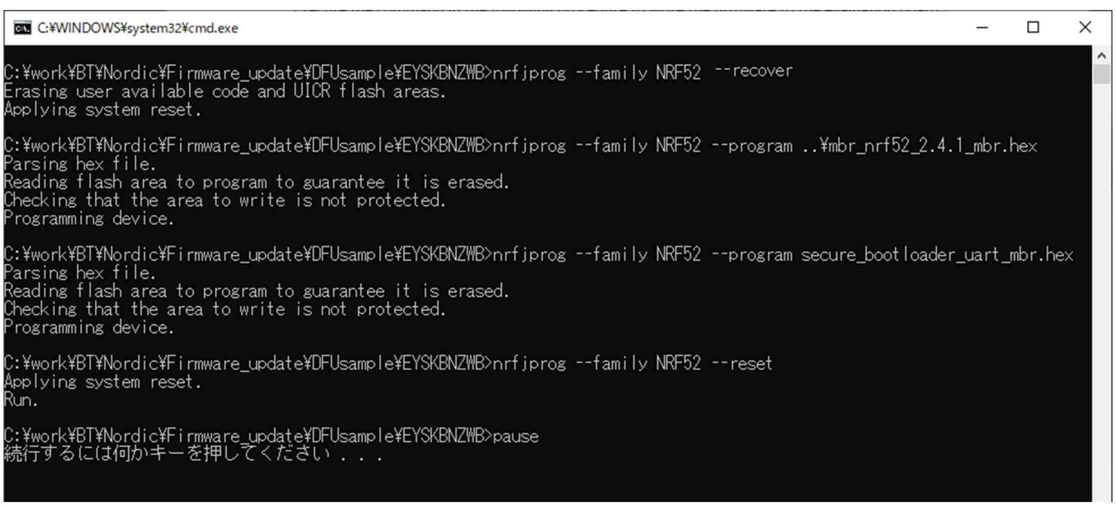
3. Writing Bootloader firmware

   ・Command：*nrfjprog --family NRF52 --program XXXXXXXX.hex*

   ※Specify the HEX file to be written.

4. Please press the reset switch on the evaluation board.

[Execution screen]



5.1.3 Writing with nRF Connect Programmer

Write the firmware by referring to the following Nordic site.

URL：
https://infocenter.nordicsemi.com/topic/ug_nc_programmer/UG/nrf_connect_programmer/ncp_application_overview.html

1. Select the connected j-link Lite from “Select Device” on the upper left.
2. Execute “Erase all” from the Device menu on the right.
3. Click "Add HEX file" of the menu file on the upper right and select the HEX file of the Softdevice to be written.
4. Execute “Write” from the right side menu Device to write
5. Click “Add HEX file” in the menu file on the upper right again, and select the HEX file of the firmware of the sensor to be written.
6. Execute "Write" again from the menu Device on the right to write.
7. Please press the reset switch on the evaluation board.

5.1.4 Writing with Segger Embedded Studio

When writing using segger embedded studio, perform the following procedure to write.

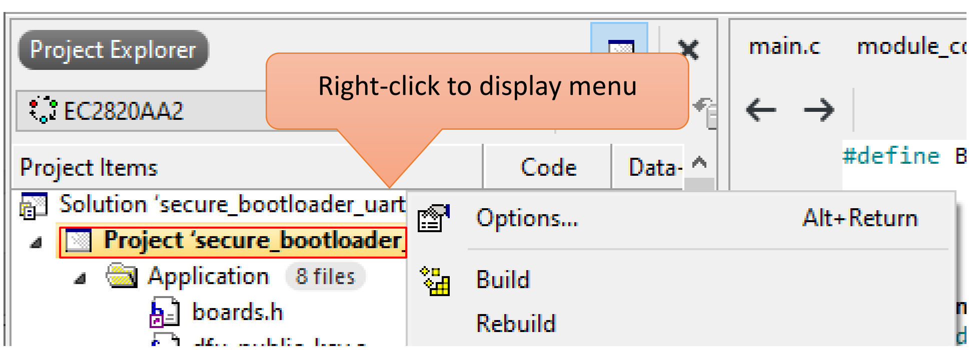
In the DFU Sample Application, mbr_nrf52_2.4.1_mbr.hex is set to be written. To change to Softdevice, etc., select Debug → Loader from the menu displayed in “Edit Options ...” below, and change the file of “Additional Load File [0]”.

[Erase of FLASH region]

1. Click "Target" from Menu and execute "Connect J-Link"
2. Click “Target” from Menu and execute “Erase All”

[Bootloader writing]

1. Click "Target" from Menu and execute "Connect J-Link"
2. Click "Target" from Menu and execute "Download secure_bootloader_uart_mbr"



Company names, product names, etc. described in the text are registered trademarks or trademarks of the respective companies. TM, (R) mark, etc. are not specified in the text.

Some of our source code uses Nordic source code, and the copyright of that part belongs to Nordic.