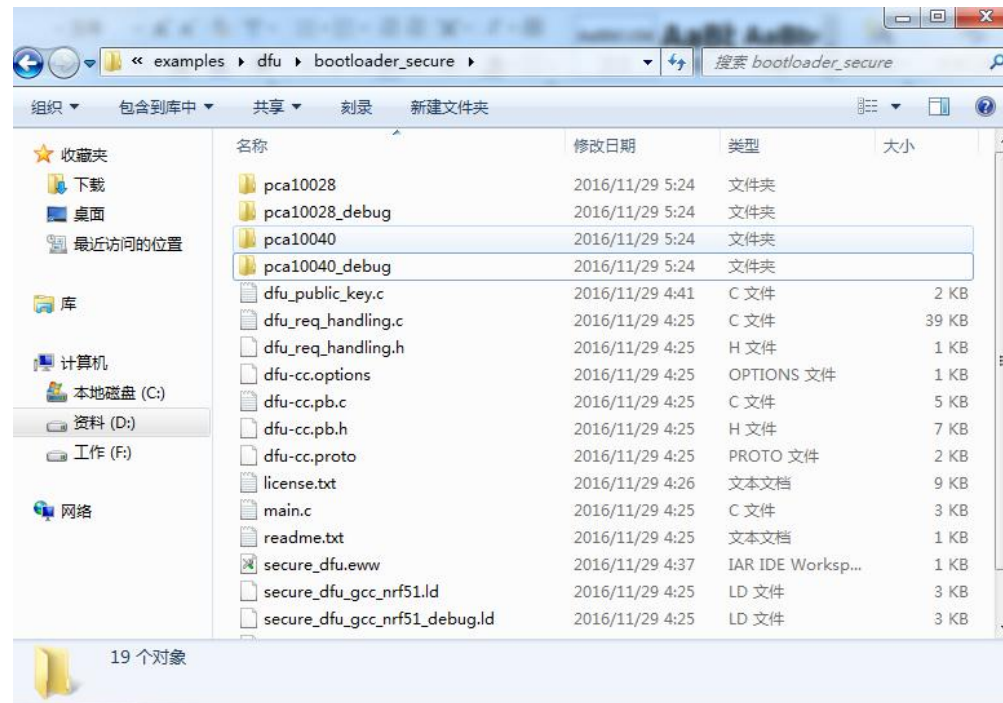


SDK12.0 nrf52832 空中升级详细步骤

小潘 921537102

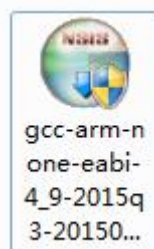
本文档为个人学习总结，只作为学习交流用途
欢迎大神指导纠正错误。

打开<sdk 路径>\examples\dfu\bootloader_secure 目录下为 bootloader 源文件存放的路径，如图。根据官方介绍，SDK12 以前的 bootloader 为 Legacy DFU，12 版本正式引入了 Secure DFU，是不能与旧版本的兼容的。打开该目录可以发现有两种 bootloader，有_debug 为调试版本，如果需要避免一些检查可以打开这个版本，本次实验不是使用 debug 版本。本次实验以 nrf52832 SDK12.2 为例，使用正式的 bootloader 版本。



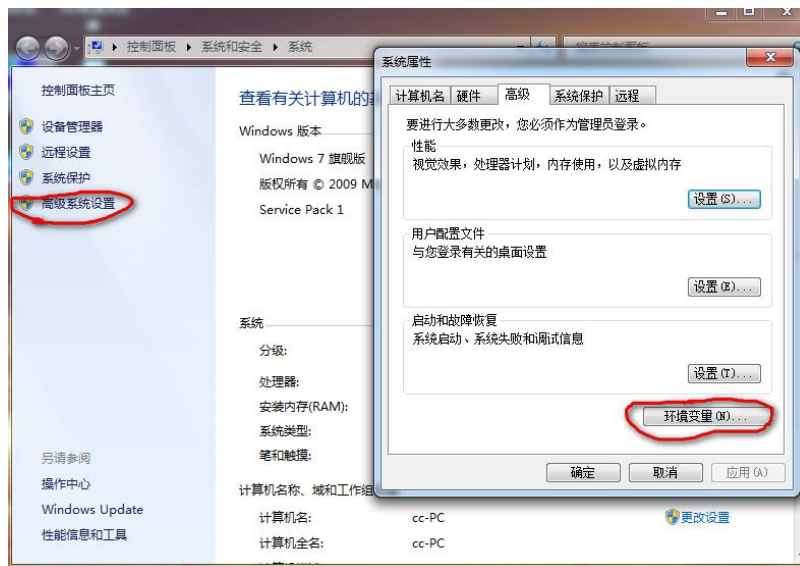
编译 bootloader 出现两个错误：1、uecc.h 文件找不到。2、需要生成私有的 key 首先要安装 micro-ecc

1、根据提示安装 4.9-2015-q3-update 版本的 GCC compiler toolchain for ARM

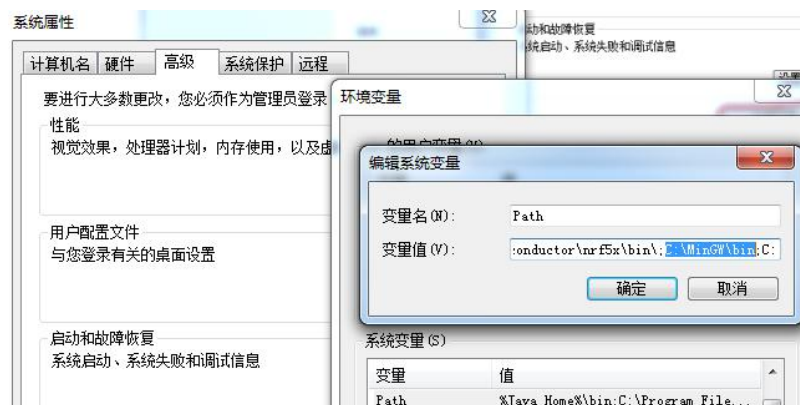


2、安装一个 WinGW 或者 GCC Make 确保可以运行 Make 编译，这里安装的是 WinGW，配置 win 系统环境变量 Path,将 MinGW 安装目录里的 bin 文件夹的地址添加到 PATH 里面，（注意：PATH 里两个目录之间以英文的;隔开）。打开 MinGW 的安装目录，打开 bin 文件夹，将 mingw32-make.exe 重命名为 make.exe，环境变量 Path 配置如图。





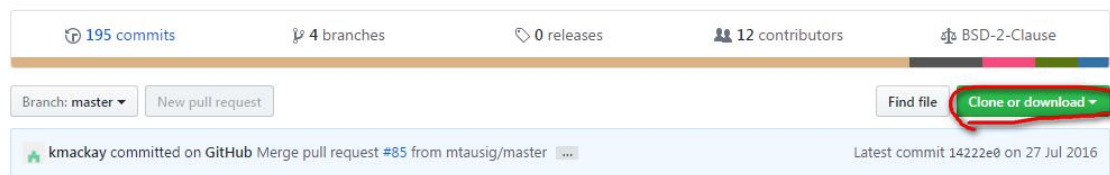
在系统变量 Path 编辑



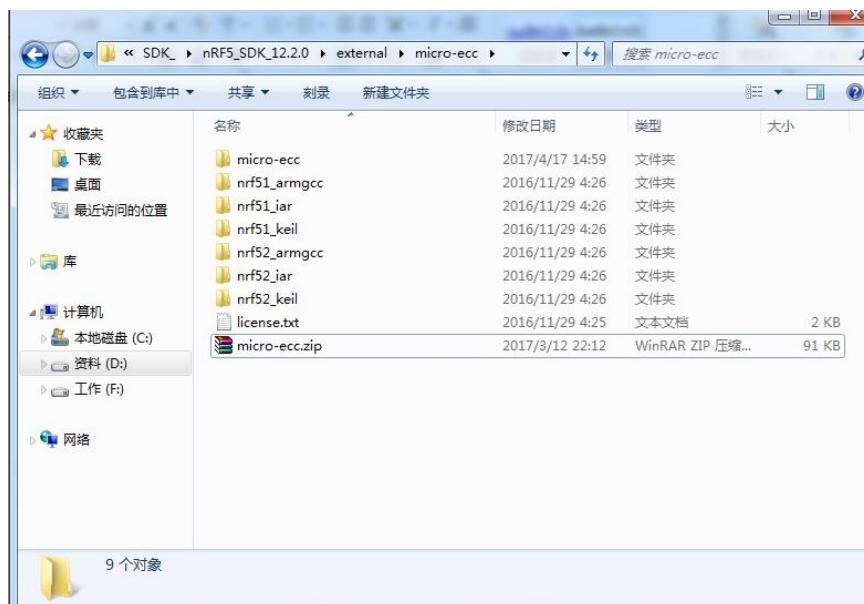
3、安装之后在 GitHub 中，github 地址为

<https://github.com/kmackay/micro-ecc>

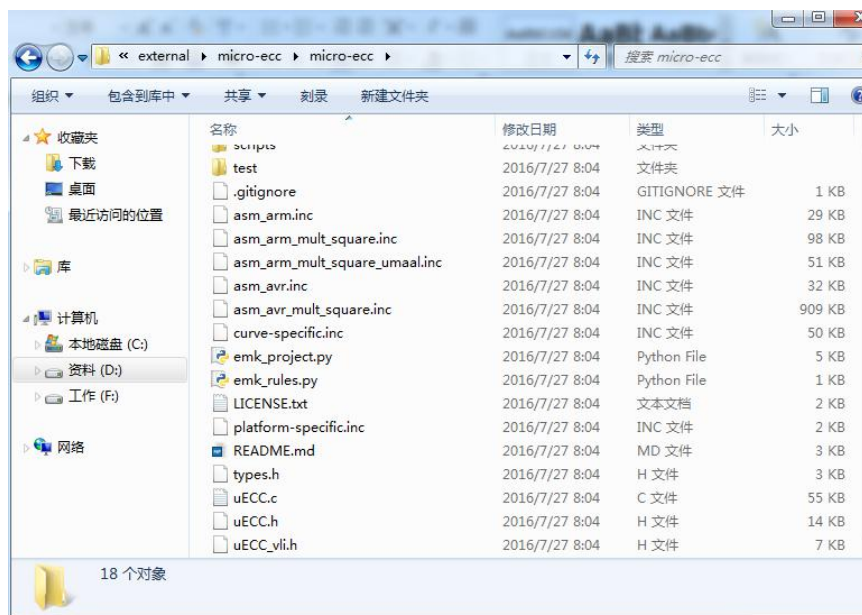
ECDH and ECDSA for 8-bit, 32-bit, and 64-bit processors.



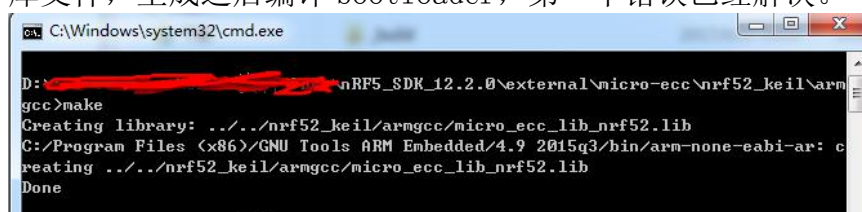
下载 ecc 的 zip 源码复制到 <SDK 路径>\external\micro-ecc 目录下，micro-ecc-master 改名为 micro-ecc，并解压文件。



注意源文件的路径



4、在 bootloader 中找到 external\micro-ecc\nrf52_keil\armgcc 这个路径这里使用的是 nrf52 和 keil 编译器，在这个目录中有 Makefile 文件，shift+鼠标右键在此处打开 DOS 命令，输入 make，则开始编译创建 micro_ecc_lib_nrf52.lib 库文件，生成之后编译 bootloader，第一个错误已经解决。



接下来为第二步 Key 的生成

Zip 文件的打包和 key 的生成需要用到 nrfutil 软件,注意,是新版本的 nrfutil 软件(version 1.5.0 or later)与以前不一样的是,是 Python Package,首先在 GitHub 下载 nrfutil 的相关文件,在这里我安装了 Python2.7,使用 Python 来安装 nrfutil。

Nrfutil 相关文件下载地址:

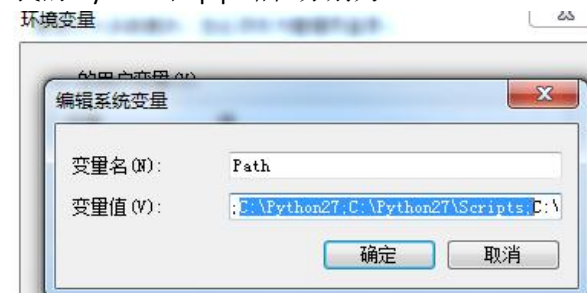
<https://github.com/NordicSemiconductor/pc-nrfutil/>

1、安装 Python2.7,将 Python 的路径添加到系统的环境变量中。(方法与上面安装 WINGW 类似)



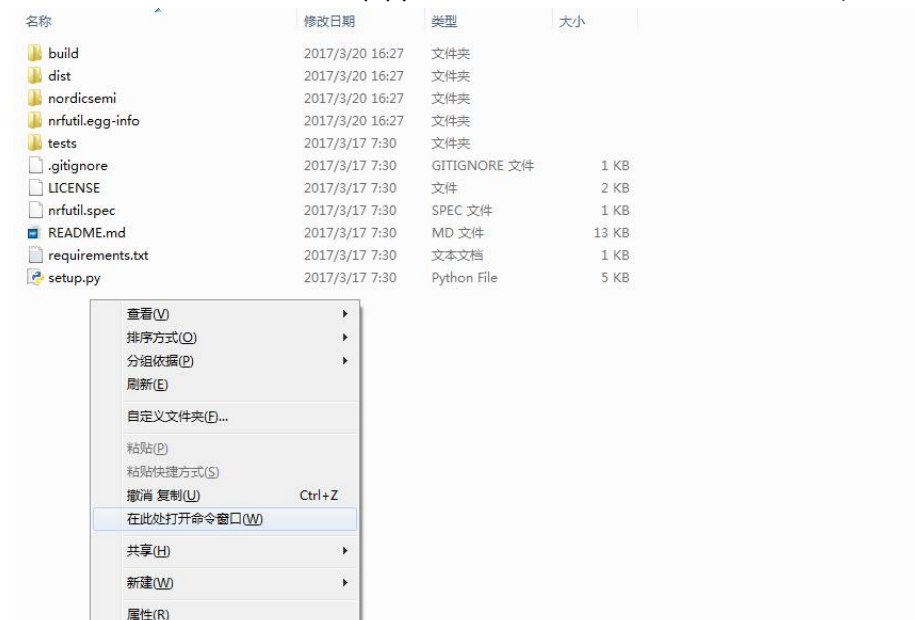
2、安装好了 Python2.7 之后 pip 已经安装上了,将 pip 的路径配置到环境变量中,打开 cmd,输入 pip 出现命令列表则表示已经安装成功

我的 Python 和 pip 路径分别为:



3、GitHub 中下载 nrfutil 的相关文件 <https://github.com/NordicSemiconductor/pc-nrfutil/>

4、解压压缩包并打开有 setup.py 文件的目录,在此处打开 dos 命令(shift+右键打开)。



5、输入 `python setup.py install` 这步骤需要有网络的时候运行,安装可能需要等待几分钟时间。

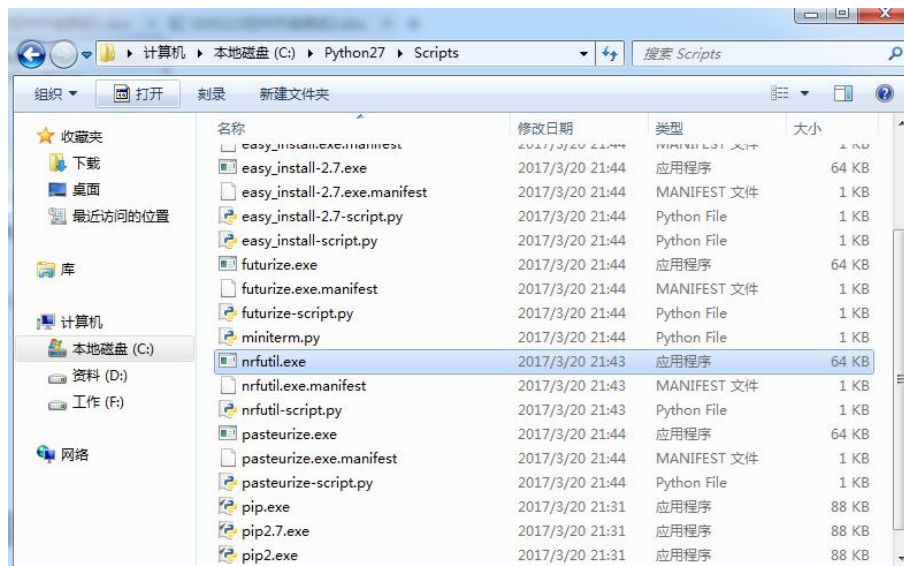

```

copying nrfutil.egg-info\entry_points.txt -> build\bdist.win32\egg\EGG-INFO
copying nrfutil.egg-info\not-zip-safe -> build\bdist.win32\egg\EGG-INFO
copying nrfutil.egg-info\requires.txt -> build\bdist.win32\egg\EGG-INFO
copying nrfutil.egg-info\top_level.txt -> build\bdist.win32\egg\EGG-INFO
creating dist
creating 'dist\nrfutil-2.2.0-py2.7.egg' and adding 'build\bdist.win32\egg' to it
removing 'build\bdist.win32\egg' (and everything under it)
Processing nrfutil-2.2.0-py2.7.egg
creating c:\python27\lib\site-packages\nrfutil-2.2.0-py2.7.egg
Extracting nrfutil-2.2.0-py2.7.egg to c:\python27\lib\site-packages
Adding nrfutil 2.2.0 to easy-install.pth file
Installing nrfutil-script.py script to C:\Python27\Scripts
Installing nrfutil.exe script to C:\Python27\Scripts
Installing nrfutil.exe.manifest script to C:\Python27\Scripts

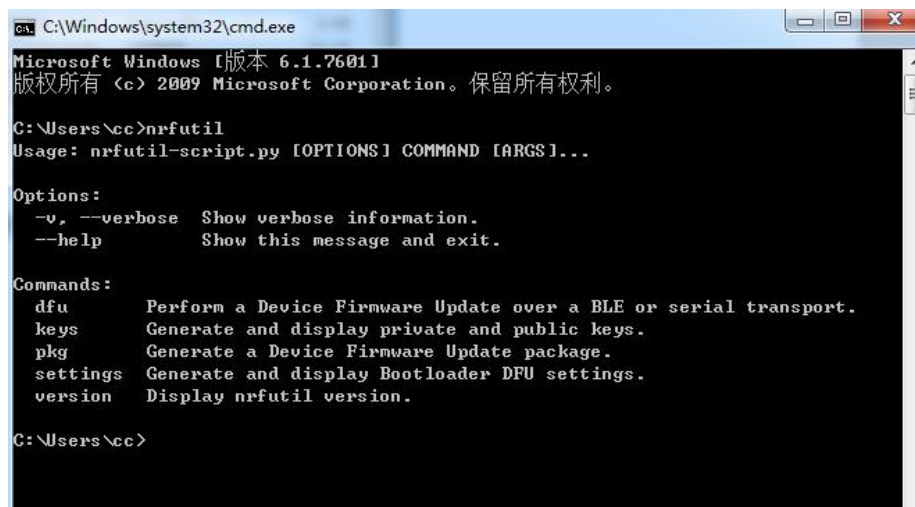
Installed c:\python27\lib\site-packages\nrfutil-2.2.0-py2.7.egg
Processing dependencies for nrfutil==2.2.0
Searching for pc-ble-driver-py>=0.8.0
Reading https://pypi.python.org/simple/pc-ble-driver-py/
Best match: pc-ble-driver-py 0.8.1
Downloading https://pypi.python.org/packages/cd/c6/7da5b94043b828e8c6445d39a0130
644a03edb20bf4f37c523d4bb7baee6/pc_ble_driver_py-0.8.1.tar.gz#md5=94c617a2b0a897
89ddf482c9e44b21ac

```

6、安装成功之后，这个时候<python 安装目录>\Python27\Scripts 的目录下出现了 nrfutil.exe 则表示安装完成。



7、在 DOS 命令下输入 nrfutil 可以获取更多关于 nrfutil 的信息，则表示 nrfutil 安装成功

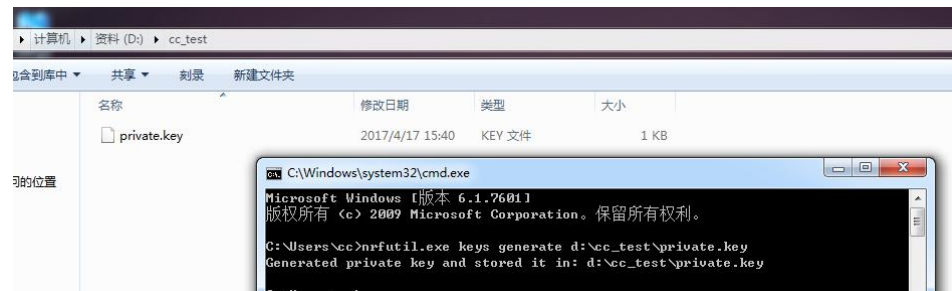


现在可以使用 nrfutil 来生成 key 了

在 cmd 命令中输入以下内容:

```
nrfutil.exe keys generate d:\cc_test\private.key
```

如图在 d 盘的 cc_test 文件夹中会生成 private.key 文件

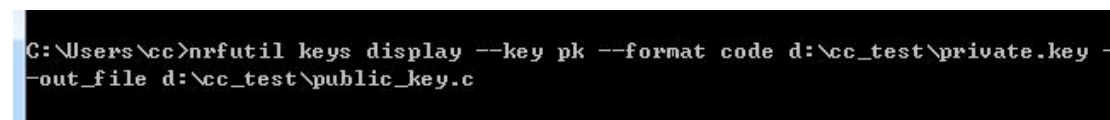


然后使用这个生成的文件来生成 public key

这里我们直接生成.c 文件

在 cmd 中输入:

```
nrfutil keys display --key pk --format code d:\cc_test\private.key --out_file d:\cc_test\public_key.c
```



输入完成之后文件夹中会出现相应的.c 文件了

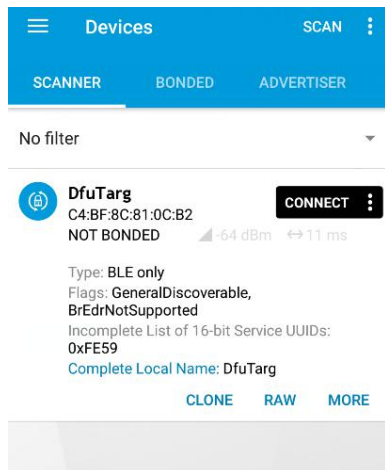


将这个文件改名为 dfu_public_key.c，将<bootloader 工程>\examples\dfu\bootloader_secure 这个路径下的 dfu_public_key.c 用新生成的替换掉。

此时再编译 bootloader，已经没有错误的提示了

```
compiling dfu_public_key.c...
linking...
Program Size: Code=20026 RO-data=2054 RW-data=180 ZI-data=19316
FromELF: creating hex file...
".\_build\nrf52832_xxaa_s132.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:02
```

这时在 build_目录下会生成的 bootloader 的 hex 文件，可以用 nRFgo Studio 下载 bootloader 了。注意，还是和以前步骤一样，先下载协议栈，再下载 bootloader。



Bootloader 可以正常工作之后,我们下一步就可以打包我们自己的 application 空中升级包了,上面安装的新版本的 nrfutil.exe 来进行 zip 包的打包。

一个 zip 包包含了我们要升级的 hex 文件和 init data 还有数字签名。官方可以 softdevice, bootloader 和 application 进行空中升级,在此只演示应用程序的空中升级。

1、准备好应用程序的 hex 文件,通常 hex 文件是编译之后出现在_build 文件夹中,默认 52832sdk 例程编译出来名称为 nrf52832_xxaa.hex。本次实验我们以串口透传例程作为空中升级的固件。使用的是 Ble_app_uart 例程

2、将前面步骤生成的 private.key 和 application.hex 放入同一个路径中,这里将文件同样放在放在 d:\cc_test 下



3、在该路径 shift+右键打开 DOS 命令输入以下内容:

```
nrfutil pkg generate --hw-version 52 --application-version 1 --application application.hex --sd-req 0x8C --key-file private.key cc_Dfu12.2.zip
```

注: 这个步骤就是向 zip 中添加 init data 和数字签名,官方定义的规则如下:

Acceptance rules for versions

Unless version validation is skipped, the `dfu_handle_prevalidate()` function applies the following acceptance rules to determine whether the image is accepted:

- **Hardware version:** If the hardware version that is specified in the init packet matches the hardware of the device, the image is accepted.
- **SoftDevice Firmware ID:** If one of the specified firmware IDs matches the ID of the current SoftDevice, the image is accepted.
- **Firmware version:** If the image contains a bootloader, the image is accepted if the new firmware version is greater than (>) the existing version of the bootloader. If the image contains an application, the image is accepted if the new firmware version is greater than or equal to (>=) the existing version of the bootloader.

```
44 #if defined ( NRF51 ) && !defined(NRF_DFU_HW_VERSION)
45     #define NRF_DFU_HW_VERSION (51)
46 #elif defined ( NRF52 ) && !defined(NRF_DFU_HW_VERSION)
47     #define NRF_DFU_HW_VERSION (52)
48 #else
49     #error No target set for HW version.
50 #endif
```



```
C:\Windows\system32\cmd.exe

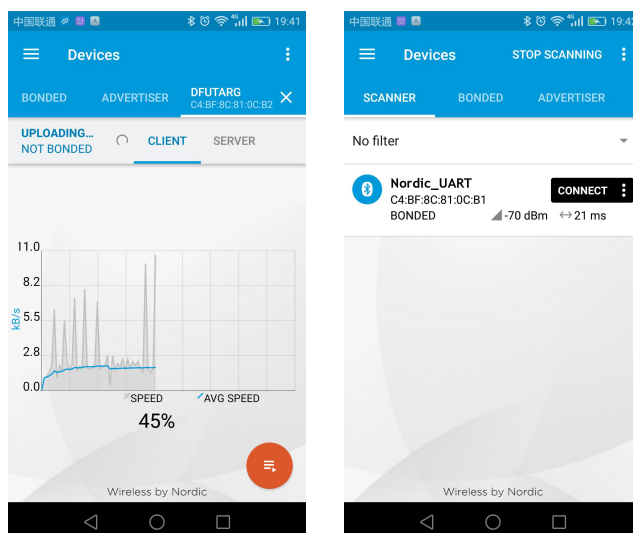
D:\cc_test>nrfutil pkg generate --hw-version 52 --application-version 1 --application application.hex --sd-req 0x8C --key-file private.key cc_Dfu12.2.zip
Zip created at cc_Dfu12.2.zip

D:\cc_test>
```

此时该目录下可以发现 zip 文件已经生成了



将这个 zip 包放进手机中,使用 nrftoolbox(推荐使用 v2.0.0 以上)或者 nrf connect 可以进行空中升级,这里我打开的是安卓版本 nrf connect4.9(需使用 v4.8.0 以上),连接上了之后选择右上角的 DFU,选择刚才生成的 zip 包。升级完成后可以看到此时蓝牙广播为 ble_app_uart 例程的广播名,bootloader 功能正常。

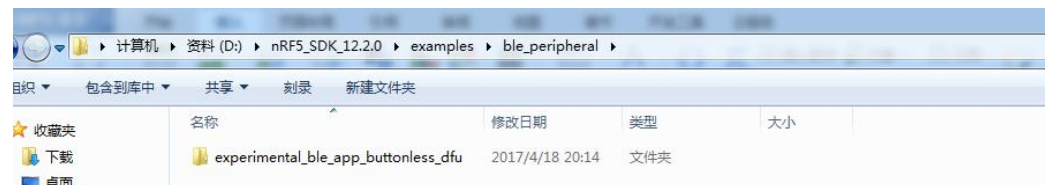


总结:由此可见 SDK12 中 bootloader 对比更新之前的 DFU bootloader 打包和环境的安装上麻烦了很多。但由于 zip 包中加入了 key 文件,需要与 bootloader 中的 key 相对应才能空中升级,别人在不知道 bootloader 中的 key 的情况下,是不能对你的设备进行空中升级的,所以此次更新比以前版本的 bootloader 更安全了。

Bootloader 能正常进行 BLE 空中升级，那我们只需要在应用程序 application 中，控制程序跳转进入到 bootloader 里面就可以完成空中升级了。

在以前版本的 SDK 中，进入 bootloader 的方式一般都是长按着 boot 按键(在官方 DK 中为 Botton4)，然后复位进入到 bootloader 中，但这样如果产品中没有按钮或者其他物理按键，就会非常不方便(以前我都是直接添加 DFU 服务进入应用程序中，也可以不需要按键)

在 SDK12 中有这样一个例程，能帮助使用者在 application 程序中不需要按键就能跳转进入 bootloader，例程在<sdk 目录>\examples\ble_peripheral\experimental_ble_app_buttonless_dfu

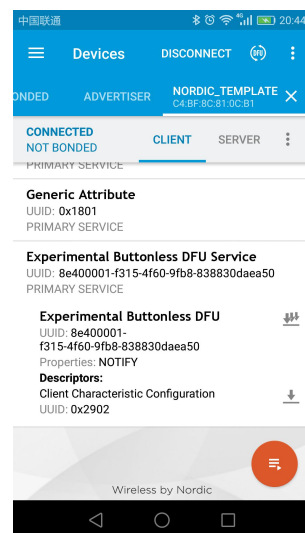


根据官方文档的介绍，可以看出，实现的方式非常简单，只要向该 DFU control point 特征值中写入 0x01，设备便断开连接，上报 BLE_DFU_EVT_ENTERING_BOOTLOADER 事件并进入 bootloader 中。

When notification of DFU control point characteristic is enabled, and the value 0x01 is written to it, the device will disconnect, send the event `BLE_DFU_EVT_ENTERING_BOOTLOADER`, and enter the bootloader.

与旧版本的 SDK 空中升级步骤一样，首先使用 nRFgo Studio 将协议栈下载到设备中，然后下载 bootloader，最后下载应用程序 application，程序下载完成之后能发现蓝牙广播，名称为“Nordic_Template”

然而我们在连接上之后，发现 DFU control point 并没有写的属性，无法写入 0x01



▲ As a Nordic employee I can confirm that this is a bug in the experimental buttonless DFU example. As correctly observed by @keton, the characteristic does not have the property enabled, i.e. the following line is missing.

Bjørn Spockeli
Nordic employee
5892 ●5 ●7
answered Sep 15 '16

jr
1 ●1 ●1
updated Nov 8 '16

```
char_md.char_props.write = 1;
```

edit flag offensive link

add a comment

这里官方给出了解释：这个程序还处于实验阶段，难免会有些 BUG，只要将 `char_md.char_props.write = 1;` 添加进入程序即可。

```

1  sdk_config.h 2  main.c 3  ble_dfu.c
81  memset(&cccd_md, 0, sizeof(cccd_md));
82
83  BLE_GAP_CONN_SEC_MODE_SET_OPEN(&cccd_md.read_perm);
84  BLE_GAP_CONN_SEC_MODE_SET_OPEN(&cccd_md.write_perm);
85
86  cccd_md.vloc = BLE_GATTS_VLOC_STACK;
87
88  memset(&char_md, 0, sizeof(char_md));
89
90  char_md.char_props.notify = 1;
91  char_md.char_props.write = 1;
92  char_md.p_char_user_desc = NULL;
93  char_md.p_char_pf = NULL;
94  char_md.p_user_desc_md = NULL;
95  char_md.p_cccd_md = &cccd_md;
96  char_md.p_sccd_md = NULL;
97
98  ble_uuid.type = p_dfu->uuid_type;
99  ble_uuid.uuid = 0x0001;
100
101  memset(&attr_md, 0, sizeof(attr_md));
102
103  BLE_GAP_CONN_SEC_MODE_SET_OPEN(&attr_md.read_perm);
104  BLE_GAP_CONN_SEC_MODE_SET_OPEN(&attr_md.write_perm);

```

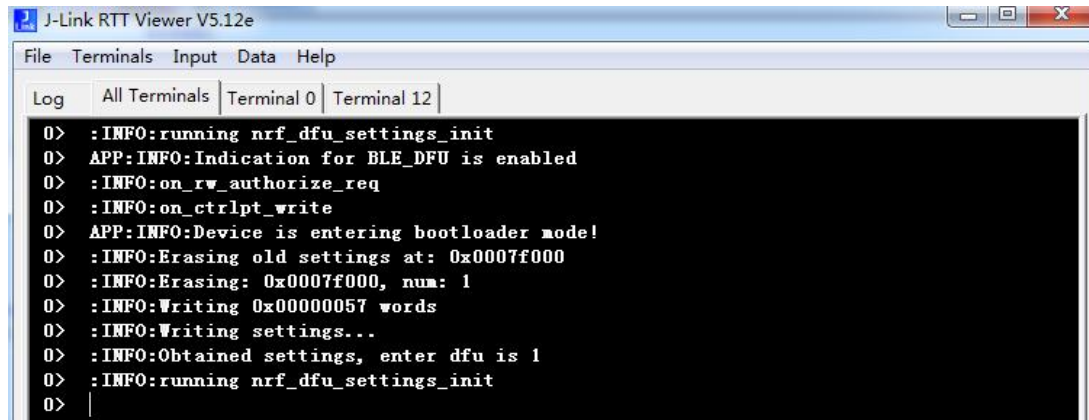
重新编译下载之后再次打开 APP，这回可以向设备写入信息了，向设备中写入 0x01，果然蓝牙断开连接了。

```

21:02:47.253 Connected to C4:BF:8C:81:0C:B1
21:02:48.546 Services discovered
21:02:50.389 Data written to descr.
00002902-0000-1000-8000-00805f
9b34fb, value: (0x) 01-00
21:02:50.389 "Notifications enabled" sent
21:02:53.964 Data written to 8e400001-
f315-4f60-9fb8-838830daea50,
value: (0x) 01
21:02:53.964 "Enter bootloader" sent
21:02:53.970 Notification received from
8e400001-
f315-4f60-9fb8-838830daea50,
value: (0x) 20-01-01
21:02:53.970 "Response for: Enter bootloader
Status: Success" received
21:02:55.158 [Server] Device disconnected
21:02:55.166 Disconnected

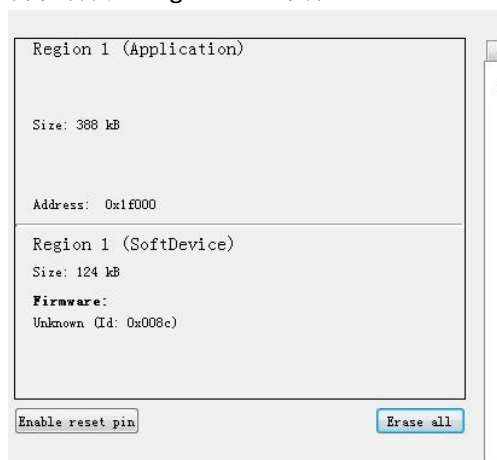
```

然而设备并没有进入到 bootloader 模式，广播名称依然为“Nordic_Template”，LOG 打印信息



```
J-Link RTT Viewer V5.12e
File Terminals Input Data Help
Log All Terminals Terminal 0 Terminal 12
0> :INFO:running nrf_dfu_settings_init
0> APP:INFO:Indication for BLE_DFU is enabled
0> :INFO:on_rw_authorize_req
0> :INFO:on_ctrlpt_write
0> APP:INFO:Device is entering bootloader mode!
0> :INFO:Erasing old settings at: 0x0007f000
0> :INFO:Erasing: 0x0007f000, num: 1
0> :INFO:Writing 0x00000057 words
0> :INFO:Writing settings...
0> :INFO:Obtained settings, enter dfu is 1
0> :INFO:running nrf_dfu_settings_init
0> |
```

再次打开 nRFgo Studio 发现 bootloader 已经被擦除掉了。



根据官方人员的解释，这是另一个 BUG

Its another bug in the buttonless DFU example. It reserves a codepage at 0x7E000 for MBR parameters, which is not necessary and should not been added. The nRF52 flash algorithm in Keil will see this and perform a sector erase from 0x7F000 to 0x1F000 before flashing the application.

▲ 1 The IROM and IRAM settings for an application using the S132 v3.0.0 with no bootloader are:

Bjørn Spockeli
Nordic employee
5892 ●5 ●7
answered Oct 27 '16
updated Oct 28 '16

✓

```

IROM1 Start: 0x1F000 Size: 0x61000
IRAM1 Start: 0x20002128 Size: 0xDEd8

```

The Secure Bootloader start address is `0x78000` and `0x1F000` + `0x61000` = `0x80000`. Thus, you have to alter the application IROM settings so that the application does not overwrite the bootloader, i.e.

```

IROM1 Start: 0x1F000 Size: 0x59000
IRAM1 Start: 0x20002128 Size: 0xDEd8

```

If you're using GCC you have to alter the application linker script in the armgcc folder, i.e.

```

MEMORY
{
  FLASH (rx) : ORIGIN = 0x1f000, LENGTH = 0x59000
  RAM (rwx) : ORIGIN = 0x20002128, LENGTH = 0xded8
}

```

根据提示修改 ROM 大小

on-chip

☒ IROM1: ☐

☐ IROM2: ☐

将 nrf_dfu_settings.c 中此部分代码去掉如图:


```

43  /*
44
45  #if defined ( NRF52 )
46
47  #if defined ( __CC_ARM )
48
49      uint8_t m_mbr_params_page[CODE_PAGE_SIZE]      __attribute__((at(NRF_MBR_PARAMS_PAGE_ADDRESS))) __attr
50
51  #elif defined ( __GNUC__ )
52
53      uint8_t m_mbr_params_page[CODE_PAGE_SIZE]      __attribute__((section(".mbrParamsPage")));
54
55  #elif defined ( __ICCARM__ )
56
57      __no_init uint8_t m_mbr_params_page[CODE_PAGE_SIZE] @ NRF_MBR_PARAMS_PAGE_ADDRESS;
58
59  #else
60
61      #error Not a valid compiler/linker for m_mbr_params_page placement.
62
63  #endif
64
65
66  #if defined ( __CC_ARM )
67      uint32_t m_uicr_mbr_params_page_address __attribute__((at(NRF_UICR_MBR_PARAMS_PAGE_ADDRESS)))
68          = NRF_MBR_PARAMS_PAGE_ADDRESS;
69
70  #elif defined ( __GNUC__ )
71      volatile uint32_t m_uicr_mbr_params_page_address __attribute__((section(".uicrMbrParamsPageAddress")
72          = NRF_MBR_PARAMS_PAGE_ADDRESS;
73  #elif defined ( __ICCARM__ )
74
75      __root const uint32_t m_uicr_mbr_params_page_address @ NRF_UICR_MBR_PARAMS_PAGE_ADDRESS
76          = NRF_MBR_PARAMS_PAGE_ADDRESS;
77
78  #else
79
80      #error Not a valid compiler/linker for m_mbr_params_page placement.
81
82  #endif
83
84  #endif // defined ( NRF52 )
85  */
86  nrf_dfu_settings_t s_dfu_settings;

```

编译下载之后发现不会把 bootloader 擦除掉了，但是程序还是在运行 bootloader 模式，下载了程序没有跳转到应用程序开始执行。以前 SDK 版本下载之后程序要从 application 开始执行的话，可以在 bootloader 中设置 APP 有效标志。

▲ In order to flash a combined SoftDevice, Bootloader and Application, where the device jumps to the application without having to perform a OTA update to set the valid application flag, you have to set the `BANK_VALID_APP` flag in `bootloader_settings.c`, i.e. change the line

Bjørn Spockeli
Nordic employee
5892 ● 5 ● 7
answered Feb 8 '17

```
uint8_t m_boot_settings[CODE_PAGE_SIZE] __attribute__((at(BOOTLOADER_SETTINGS_ADDRESS))) __attribute
```

但是 SDK12 和以前的 SDK 不一样，它需要生成 setting 文件，烧录的时候将程序和 setting 文件合并起来，方法如下：

使用前面安装的 nrfutil 来生成 setting 文件，将最新编译出来的 application.hex 文件复制出来放到统一的文件夹中，这里我还是放在 d:\cc_test 文件夹下。

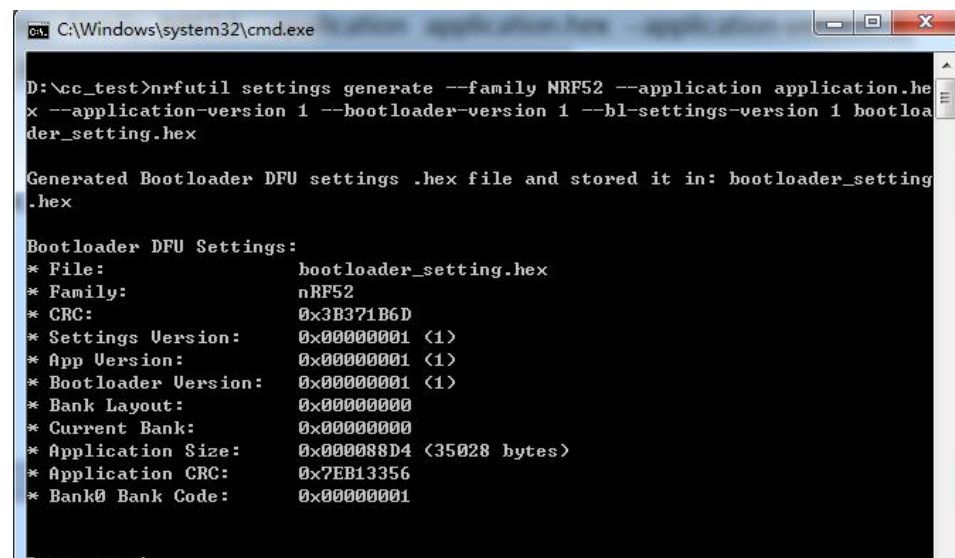
在该目录下邮件+shift 打开 DOS 命令，输入以下内容：

```
nrfutil settings generate --family NRF52 --application application.hex --application-version 1
--bootloader-version 1 --bl-settings-version 1 bootloader_setting.hex
```

说明：

--bl_settings-version: SDK12 和 SDK13 只能填 1

--application-version 和--bootloader-version:为用户选择的 application 和 bootloader 最初的版本，这里我都填 1。



```
C:\Windows\system32\cmd.exe

D:\cc_test>nrfutil settings generate --family NRF52 --application application.hex
x --application-version 1 --bootloader-version 1 --hl-settings-version 1 bootloa
der_setting.hex

Generated Bootloader DFU settings .hex file and stored it in: bootloader_setting
.hex

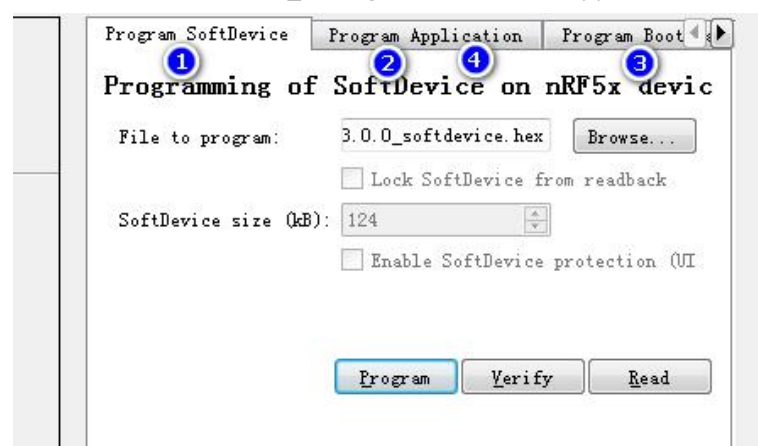
Bootloader DFU Settings:
* File:                bootloader_setting.hex
* Family:              nRF52
* CRC:                 0x3B371B6D
* Settings Version:    0x00000001 <1>
* App Version:         0x00000001 <1>
* Bootloader Version:  0x00000001 <1>
* Bank Layout:         0x00000000
* Current Bank:        0x00000000
* Application Size:     0x000088D4 (35028 bytes)
* Application CRC:     0x7EB13356
* Bank0 Bank Code:    0x00000001
```

此时会在该目录下生成 bootloader_setting.hex 文件



使用 nRFgo Studio 下载程序步骤:

softdevice-->bootloader_setting-->bootloader-->application



下载完成以后可以发现程序直接运行应用程序了。

总结：在我们的应用程序中只要加入 app_buttonless_dfu 的服务，以及事件处理，对上面的内容进行修改之后，向服务的特征中写入 0x01 便可以使设备进入 bootloader 模式了。

附录：

1、直接在 bootloader 的 keil 工程中直接下载 bootloader 的方法

在官网中下载 nRF5x-Command-Line-Tools，我电脑系统是 win7 64 位的版本

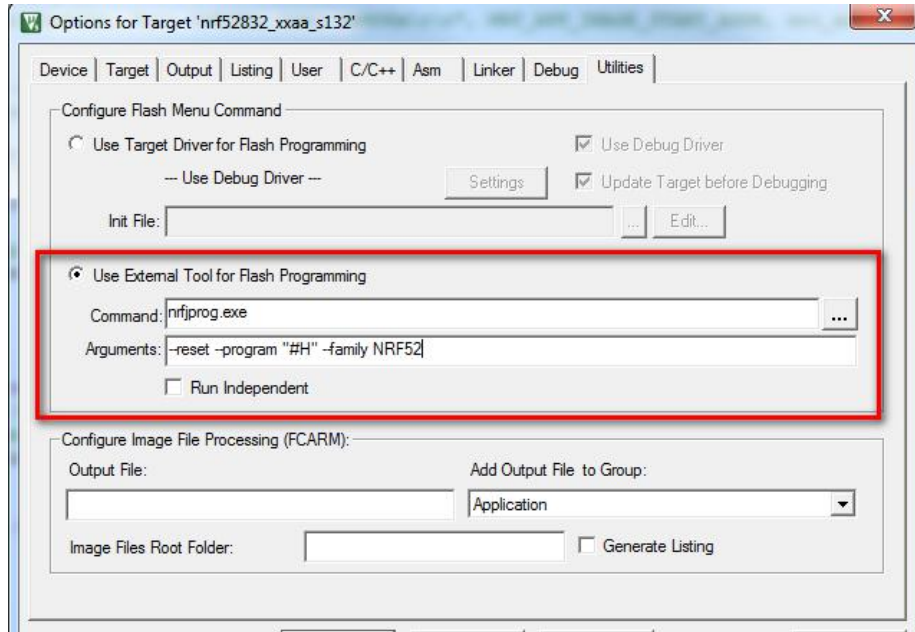
nRF5x- Command-Line- Tools-Win32	nRF5x toolset installer for Windows (JLinkARM, JLink CDC, nRFjprog, and mergehex)	9.4.0
--	--	-------

以前有安装过这个工具的请升级到最新版本(目前为 9.4.0)

安装完成之后，将这个路径添加到系统环境变量中



打开 bootloader 的 keil 工程，在 keil 中设置



复制以下内容到 Arguments:

--reset --program "#H" --family NRF52

设置之后下载了协议栈之后就可以在 keil 中直接下载 bootloader 了

```
Build Output
nrfjprog.exe --reset --program "F:\cc_test\SDK12_Boot\exa
Parsing hex file.
Reading flash area to program to guarantee it is erased.
Checking that the area to write is not protected.
Programing device.
Applying system reset.
Run.
```

2、合并文件烧录

在生产的时候我们不希望单个文件烧录，我们希望通过合并文件烧录的方式来烧录程序，能节省烧录的时间和步骤。我们可以使用 nrf 工具将 softdevice, bootloader 和 application 合并起来，这里我们依然需要使用附录 1 说的 nRF5x-Command-Line-Tools 工具。

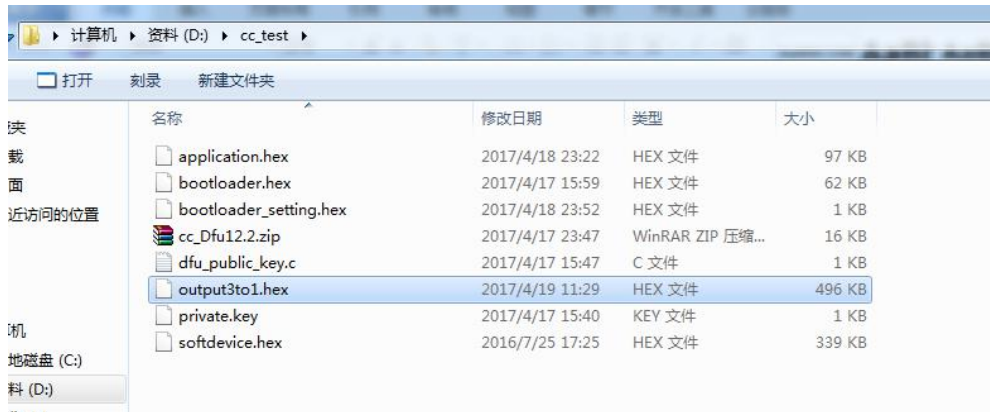
将 softdevice, bootloader 和 application 文件全部放在同一个文件夹下，我还是放在 d:\cc_test 中，右键+shift 打开 DOS 命令，输入以下内容：

可以将

mergehex --merge softdevice.hex bootloader.hex application.hex --output output3to1.hex

```
C:\Windows\system32\cmd.exe
D:\cc_test>mergehex --merge softdevice.hex bootloader.hex application.hex --outp
ut output3to1.hex
Parsing input hex files.
Merging files.
Storing merged file.
```

此时该目录下生成了一个 output3to1.hex 文件



直接使用 nRFgo studio 烧录这个 hex 就可以了, 若需要烧录后直接启动应用程序 application, 则只需要再将 bootloader_setting.hex 文件合并即可。

在 DOS 命令中输入以下内容:

```
mergehex --merge output3to1.hex bootloader_setting.hex --output output.hex
```

```
D:\cc_test>mergehex --merge output3to1.hex bootloader_setting.hex --output output3to1.hex
Parsing input hex files.
Merging files.
Storing merged file.
```

使用 nRFgo studio 烧录 output.hex 到 application program, 点击下载, 下载完成之后就可以看见程序从应用开始启动了。

