

使用指南

N32WB03x 固件升级使用指南

简介

本文档将介绍N32WB03X 固件升级的原理和应用例程,帮助用户了解固件升级的流程,便于快速开发。

N32WB03X固件升级功能特点:

- 支持串口升级固件。
- 支持蓝牙升级固件。
- 支持蓝牙MTU 20-244字节。
- 为保证安全, 蓝牙升级使用ECC数字签名校验固件合法性。
- 为加快升级的速度, 和支持版本可回退的功能, 我们提供蓝牙“双Bank”升级。
- 我们同时提供蓝牙“单Bank”升级, 满足用户程序占用空间较大的情况。
- 系统自动选择“双Bank”升级还是“单Bank”升级, 用户无需判断。
- 升级速度可以通过手机APP调节, 以便兼容各厂商手机蓝牙。

目录

简介	1
1 实例演示	4
1.1 JLINK烧录演示	4
1.2 串口升级演示	6
1.3 蓝牙“双BANK”升级演示	7
1.4 蓝牙“单BANK”升级演示	9
2 FLASH内存分布	12
3 数据结构	14
3.1 BOOTSETTING	14
3.2 INIT PACKET	15
3.3 DFU_SETTING	16
4 升级流程	17
4.1 串口升级流程	17
4.2 蓝牙双BANK升级流程	17
4.3 蓝牙单BANK升级流程	18
5 升级命令	19
5.1 串口升级命令	19
5.1.1 Usart dfu	20
5.1.2 Ping	20
5.1.3 Init packet	20
5.1.4 Packet header	20
5.1.5 Packet	21
5.1.6 Postvalidate	21
5.1.7 Activate&Reset	21
5.2 蓝牙升级命令	22
5.2.1 更新BLE连接间隙命令	25
5.2.2 更新BLE MTU命令	25
5.2.3 查询版本号和升级方式命令	25
5.2.4 创建发送dfu_setting和签名文件命令	26
5.2.5 创建发送固件数据命令	26
5.2.6 FLASH固件整体校验命令	27
5.2.7 激活分区表和复位命令	27
5.2.8 跳入ImageUpdate	27
5.3 错误代码	28
6 工具讲解	29
6.1 JLINK工具	29
6.2 NSUTIL工具	29
6.3 NSANDROIDUTIL工具	29
7 例程讲解	31

7.1 MASTERBOOT讲解	31
7.2 APPUSART讲解	33
7.3 APPOTA讲解	34
7.4 IMAGEUPDATE讲解	38
8 加密讲解	39
9 常见问题	40
9.1 运行《JLINKPROGRAMMING.BAT》批处理问题	40
10 版本历史	41
11 声明	42

NATIONS CONFIDENTIAL

1 实例演示

1.1 JLINK烧录演示

进入《N32WB03x_SDK\utilities\dfu\Image\JLINKProgrammingDemo》目录。

双击《JLINKProgramming.bat》文件，查看命令行窗口输出，如下图。

```
=====BootSetting.bin=====
00000000: 52 19 34 43 FF FF FF FF 00 40 00 01 B8 49 00 00 R.4C.....@...I..
00000010: F7 5C 36 95 01 00 00 00 01 00 00 00 FF FF FF FF .\6.....
00000020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..
00000030: 00 00 02 01 30 4A 00 00 3A 87 34 26 01 00 00 00 ....0J...:4&...
00000040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..
00000050: FF FF FF FF FF FF FF FF 00 C0 03 01 EC 36 00 00 .....6..
00000060: 1D D3 2D D9 01 00 00 00 FF FF FF FF FF FF FF FF ..-.....
00000070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..
00000080: AE 1C 41 A5 F4 35 DD 3D 89 C8 00 D8 0F 8D 2A C2 ..A..5.=.....*.
00000090: 63 3A 02 37 24 5D 2D DB F0 46 A1 6A 5E 43 26 44 c:;.7$]-..F.j^C&D
000000A0: 73 20 7D 16 86 EA 41 6B A3 8D 0D 60 DA 61 CD 98 s }...Ak...a..
000000B0: 53 D5 22 A5 14 6A EE 64 BB B4 7E 40 39 A6 B5 29 S."..j.d..@9..)
Bootsetting created successfully!
SEGGER J-Link Commander V6.32 (Compiled Apr 20 2018 17:25:19)
DLL version V6.32, compiled Apr 20 2018 17:25:02
```

第一步：NSUtil工具生成BootSetting.bin，命令行显示Bootsetting created successfully!说明文件生成成功。

```
Erasing device (N32WB031KEQ6-2)...
J-Link: Flash download: Total time needed: 1.521s
Erasing done.
```

第二步：JLink擦除芯片Flash。

```
Downloading file [Image\MasterBoot.bin]...
J-Link: Flash download: Bank 0 @ 0x01000000: 1 range affected (8192 bytes)
J-Link: Flash download: Total time needed: 0.620s (Prepare: 0.040s, Compare O.K.
```

第三步：烧录MasterBoot.bin到芯片Flash。

```
Downloading file [Image\Bootsetting.bin]...
J-Link: Flash download: Bank 0 @ 0x01000000: 1 range affected (4096 bytes)
J-Link: Flash download: Total time needed: 0.285s (Prepare: 0.037s, Compare O.K.
```

第四步：烧录Bootsetting.bin到芯片Flash。

```
Downloading file [Image\APP1.bin]...
J-Link: Flash download: Bank 0 @ 0x01000000: 1 range affected (20480 bytes)
J-Link: Flash download: Total time needed: 1.715s (Prepare: 0.045s, Compare O.K.
```

第五步：烧录APP1.bin到芯片Flash。

```
Downloading file [Image\APP2.bin]...  
J-Link: Flash download: Bank 0 @ 0x01000000: 1 range affected (20480 bytes)  
J-Link: Flash download: Total time needed: 1.703s (Prepare: 0.040s, Compare  
O.K.
```

第六步：烧录APP2.bin到芯片Flash。

```
Downloading file [Image\ImageUpdate.bin]...  
J-Link: Flash download: Bank 0 @ 0x01000000: 1 range affected (16384 bytes)  
J-Link: Flash download: Total time needed: 1.339s (Prepare: 0.054s, Compare  
O.K.
```

第七步：烧录ImageUpdate.bin到芯片Flash。

```
Reset delay: 0 ms  
Reset type NORMAL: Resets core & peripherals via SYSRESETREQ & VECTRESET bit.  
Reset: Halt core after reset via DEMCR.VC_CORERESSET.  
Reset: Reset device via AIRCR.SYSRESETREQ.
```

第八步：复位芯片。

JLINK依次执行，擦除，写入MasterBoot.bin, Bootsetting.bin, APP1.bin, APP2.bin, ImageUpdate.bin, 复位命令，完成批量烧录工作，如果烧录不了，可能是因为芯片进入DEEP SLEEP，可以尝试先执行《[JLINKProgramming.bat](#)》，再给芯片上电。

MasterBoot.bin由[MasterBoot Keil例程](#)生成，生成路径查看例程 keil → option for target → User → Run #2。

Bootsetting.bin由NSUtil python 工具生成，[bootsetting数据结构查看第三章第一节](#)。

APP1.bin和APP2.bin由[AppOTA例程](#)生成，生成路径查看例程 keil → option for target → User → Run #2。

ImageUpdate.bin由[ImageUpdater例程](#)生成，生成路径查看例程 keil → option for target → User → Run #2。

Bin文件的运行地址和功能描述在[第二章](#)有详细介绍。

文本工具打开《[JLINKProgramming.bat](#)》文件。

```

set JLink_path=..\..\JLink\JLink_V632\JLink.exe
set JLink_script_path=..\..\JLink\JLink_Script\download.jlink
set NSUtil_path=..\..\NSUtil\NSUtil.exe

::Creating bootsetting file
::bootsetting.bin path
set output_bootsetting=.\Image\bootsetting.bin
::bank1 parameters, nonoptional
set bank1_start_address=0x1004000
set bank1_version=0x00000001
set bank1_bin=.\Image\APP1.bin
set bank1_activation=yes
::bank2 parameters, optional
set bank2_start_address=0x1020000
set bank2_version=0x00000001
set bank2_bin=.\Image\APP2.bin
set bank2_activation=no
::ImageUpdate parameters, optional
set image_update_start_address=0x0103C000
set image_update_version=0x00000001
::set image_update_bin=.\Image\ImageUpdate.bin
set image_update_activation=no
::Public key, optional
::set public_key_file=.\keys\public_key.bin

```

用户可以修改bank1_activation=no, bank2_activation=yes, 上电启动bank2的程序, 可以通过串口接芯片PB1 (115200 N 8 1) 查看打印看效果, 也可以通过LED1和LED2闪烁频率看效果, APP1以100毫秒闪烁, APP2以500毫秒闪烁, 蓝牙搜索NATION广播名称。

用户还可以修改bank1_activation=no, image_update_activation=yes, 看串口打印效果, 或者蓝牙搜索ImageUpdate设备广播看效果。

需要注意的是, activation只有一个程序可以启用。

1.2 串口升级演示

进入《N32WB03x_SDK\utilities\dfu\Image\UartProgrammingDemo》目录。

双击《JLINKProgramming.bat》文件, 查看命令行窗口输出。

```

=====BootSetting.bin=====
00000000: 13 1F 64 9F FF FF FF FF 00 40 00 01 58 0E 00 00 ..d.....@..X...
00000010: 0D 1D BC 89 01 00 00 00 01 00 00 00 FF FF FF FF .....
00000020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000030: 00 00 02 01 58 0E 00 00 CF F8 F2 0C 01 00 00 00 ....X.....
00000040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000080: FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000090: FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
000000A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
000000B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
Bootsetting created successfully!

```

从Bootsetting.bin可以看出, 芯片中只有bank1和bank2有程序, 且bank1程序激活。

APP1.bin和APP2.bin由AppUsart例程生成, 生成路径查看例程 keil → option for target → User → Run #2。

文本工具打开《UartFirmwareUpdate.bat》文件。

```

set NSUtil_path=..\..\NSUtil\NSUtil.exe
:: APP.bin is the update firmware
:: --app_start_address is the address of update firmware
:: --app_version is the version of update firmware
:: --serial_port is the serial port number
set app_bin=.\Image\APP2.bin
set app_start_address=0x1020000
set app_version=0x01020304
set serial_port=COM3

%NSUtil_path% ius serial %app_bin%
                                --app_start_address %app_start_address%
                                --app_version %app_version%
                                --serial_port %serial_port%

```

修改serial_port=自己电脑的串口号（通过查看设备管理器得到），USB转串口线接芯片PB6（芯片TX）和PB7（芯片RX），保存关闭。

双击《UartFirmwareUpdate.bat》文件，查看命令行窗口打印结果。

```

E:\Nations\NS7300\SOFTWARE\N32WB03x\N32WB03x_SDK\N32WB03x_SDK\utilities\dfu\Image\串口升级演示>echo off
Serial Image Update In Progress [-----] 0%
Serial Image Update In Progress [#####] 100%
3672 bytes Image Update Complete in 2.13s
请按任意键继续. . .

```

JLINK默认烧录bank1程序执行（PB0脚会被拉高），串口默认升级到bank2程序执行（PA6脚会被拉高），用户也可以修改app_bin=.\Image\APP1.bin，app_start_address=0x01004000，使串口默认升级bank1程序执行。

1.3 蓝牙“双Bank”升级演示

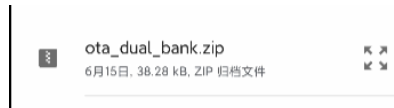
进入 N32WB03x_SDK\utilities\dfu\NSAndroidUtil\ 目录，手机安装 NSAndroidUtil.apk 文件。再进入 N32WB03x_SDK\utilities\dfu\Image\Dual Bank Update Demo\ 目录，双击《JLINKProgramming.bat》，烧录固件到芯片。

再进入N32WB03x_SDK\utilities\dfu\Image\Dual Bank Update Demo\Image\ 目录，将《ota_dual_bank.zip》文件拷贝到手机内部存储设备上。

点击连接，再点击选择NATIONS（MAC：66:55:44:77:22:99）设备



等待右上角蓝牙状态变成Connected，再点击文件，选择ota_dual_bank.zip文件。



升级包信息中可以查看到升级包的大小，点击升级，观察升级状态改变，进度条增加。



升级结束，蓝牙自动断开，升级状态显示升级耗时。



用户可以双击《GenerateUpdateImage.bat》，观察命令行窗口信息。


```

=====dfu_setting.dat=====
00000000: 19 E5 23 DF 00 40 00 01 B8 49 00 00 F7 5C 36 95 ..#..@...I... \6.
00000010: 02 00 00 00 00 00 02 01 30 4A 00 00 3A 87 34 26 .....0J...: 4&
00000020: 02 00 00 00 00 C0 03 01 EC 36 00 00 1D D3 2D D9 .....6....-.
00000030: 02 00 00 00 5C 2C 11 44 E8 A3 37 6C 2E D5 54 EE .... \. .D..71..T.
00000040: 2A 90 AA 07 08 7B 48 B2 8F 78 2E FA E8 E6 E3 1D *.... {H..x.....
00000050: 66 75 AA A3 54 C2 27 C2 EE ED 98 0A EC DB 1A 42 fu..T.'.....B
00000060: 79 36 32 D3 A8 7C 53 69 C2 15 FC E8 D4 F7 51 A1 y62.. |Si.....Q.
00000070: B2 73 EF 22 .s."
=====config.txt=====
APP1_START_ADDRESS : 0x01004000
APP1_VERSION : 0x00000002
APP1_SIZE : 0x000049b8
APP2_START_ADDRESS : 0x01020000
APP2_VERSION : 0x00000002
APP2_SIZE : 0x00004a30
IMAGE_UPDATE_START_ADDRESS : 0x0103c000
IMAGE_UPDATE_VERSION : 0x00000002
IMAGE_UPDATE_SIZE : 0x000036ec

```

该批处理文件会按照文件内配置的参数和Image文件夹下的bin文件，制作蓝牙升级包，命令行窗口同时显示dfu_setting.dat文件数据，和Config.txt文件数据。

1.4 蓝牙“单Bank”升级演示

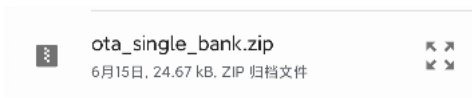
进入N32WB03x_SDK\utilities\dfu\NSAndroidUtil\目录，手机安装NSAndroidUtil.apk文件。

再进入 N32WB03x_SDK\utilities\dfu\Image\SingleBankProgrammingDemo\ 目录，双击《JLINKProgramming.bat》，烧录固件到芯片。

再进入N32WB03x_SDK\utilities\dfu\Image\SingleBankProgrammingDemo\Image\目录，将《ota_single_bank.zip》文件拷贝到手机内部存储设备上。

点击连接，再点击选择NATIONS（MAC：66:55:44:77:22:99）设备。

等待右上角蓝牙状态变成Connected，再点击文件，选择ota_single_bank.zip文件。



升级包信息中可以查看到升级包的大小，点击升级，观察升级状态改变，进度条增加，单bank升级蓝牙会断开重连一次，界面会弹出对话框要用户等待蓝牙自动重连。



蓝牙重连后升级继续，升级结束显示升级耗时。

NSAndroidUtil : V1.0 Connected

蓝牙名称过滤 断开

设备名称 : ImageUpdate 设备地址 : 66:55:44:77:22:9A
interval : 60ms latency : 0 timeout : 5000ms
MTU : 244

OTA固件升级

升级 升级状态
send image data...

文件 升级包信息
APP1_START_ADDRESS : 0x01004000
APP1_VERSION : 0x00000002
APP1_SIZE : 0x00004bcc
APP2_START_ADDRESS : 0xffffffff
APP2_VERSION : 0xffffffff
APP2_SIZE : 0xffffffff
IMAGE_UPDATE_START_ADDRESS : 0x0103c000
IMAGE_UPDATE_VERSION : 0x00000002
IMAGE_UPDATE_SIZE : 0x0000388c

Device Information Service

Manufacture :
Model :
Serial :
HW version :
FW version :
SW version :

NSAndroidUtil : V1.0 Disconnected

蓝牙名称过滤 连接

设备名称 : 设备地址 :
interval : latency : timeout :
MTU :

OTA固件升级

升级 升级状态
OTA Finish in 18.511s, PRN : 2048

文件 升级包信息
APP1_START_ADDRESS : 0x01004000
APP1_VERSION : 0x00000002
APP1_SIZE : 0x00004bcc
APP2_START_ADDRESS : 0xffffffff
APP2_VERSION : 0xffffffff
APP2_SIZE : 0xffffffff
IMAGE_UPDATE_START_ADDRESS : 0x0103c000
IMAGE_UPDATE_VERSION : 0x00000002
IMAGE_UPDATE_SIZE : 0x0000388c

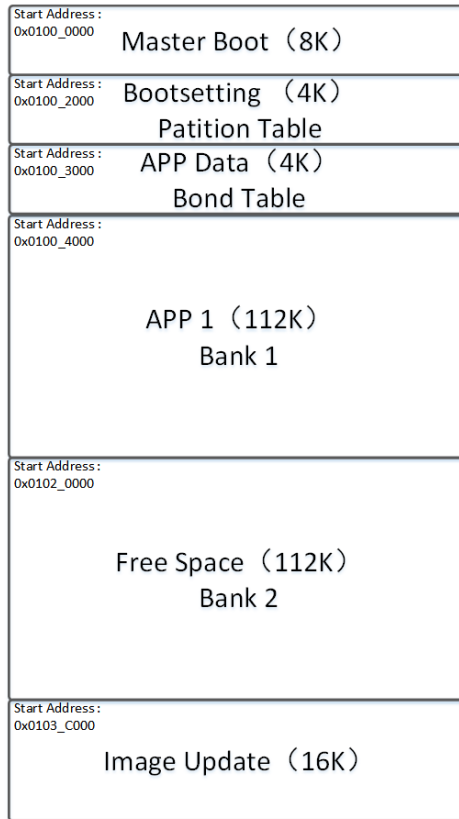
Device Information Service

Manufacture :
Model :
Serial :
HW version :
FW version :
SW version :

2 FLASH内存分布

N32WB03x芯片FLASH地址范围 0x0100_0000 - 0x0103_FFFF 可用空间 256K字节。

程序和数据FLASH分布如下图：



名称	N32WB03X FLASH地址	说明
Master Boot (程序)	0x0100_0000 – 0x0100_1FFF (8K)	上电后程序入口； 读取分区表，程序跳转； 串口升级功能；
Bootsetting (数据) (Patition Table)	0x0100_2000 - 0x0100_2FFF (4K)	FLASH分区表；
APP Data (数据) (Bond Table) (User Data)	0x0100_3000 - 0x0100_3FFF (4K)	APP数据存储区； 存储绑定列表（5个设备约500字节）； 剩余空间存储用户自定义数据；
APP 1 (程序) (Bank 1)	0x0100_4000 - 0x0101_FFFF (112K)	用户程序1存储区域；
Free Space/APP2 (程序) (Bank 2)	0x0102_0000 - 0x0103_BFFF (112K)	预留空间； 或者用户程序2存储区域；
Image Update (程序)	0x0103_C000 - 0x0103_FFFF (16K)	预留空间； 或者“单BANK”升级使用；

MasterBoot程序提供程序跳转入口功能，和串口升级功能。

Bootsetting数据提供程序跳转，和升级共享数据。

APP Data用户数据存储区域，如果需要更大空间，建议使用外接存储器，或者修改FLASH内存分布（具体修改方法请联系技术支持）。

APP1程序为用户程序。

APP2程序为用户编译在Bank2的用户程序。

ImageUpdate程序为“单bank”升级使用程序。

NATIONS CONFIDENTIAL

3 数据结构

3.1 Bootsetting

大小 (字节)	名称	说明
4	Bootsetting CRC	Bootsetting以下数据校验值
4	MasterBoot Force Update	强制MasterBoot串口升级 1: 串口升级 默认: 0xFFFFFFFF
4*10	Bank 1 分区	4字节start address程序起始地址 4字节size程序大小 4字节crc程序校验值 4字节version程序版本号 4字节activation程序激活码: 1激活, 其它 4*5字节reserve保留
4*10	Bank 2 分区	4字节start address程序起始地址 4字节size程序大小 4字节crc程序校验值 4字节version程序版本号 4字节activation程序激活码: 1激活, 其它 4*5字节reserve保留
4*10	Image Update 分区	4字节start address程序起始地址 4字节size程序大小 4字节crc程序校验值 4字节version程序版本号 4字节activation程序激活码: 1激活, 其它 4*5字节reserve保留

64	公开密钥	用于ECC签名验证。
----	------	------------

Bootsetting crc = CRC32 (MasterBoot Force Update + Bank 1 分区 + Bank 2 分区 + Image Update 分区 + 公开密钥)

MasterBoot Force Update: MasterBoot程序判断这个变量，决定是否进入串口升级模式。

Bank 1 分区: 记录APP1程序的起始地址，程序大小，程序CRC32校验值，以及程序的激活状态。

Bank 2 分区: 记录APP2程序的起始地址，程序大小，程序CRC32校验值，以及程序的激活状态。

Image Update 分区: 记录Image Update程序的起始地址，程序大小，程序CRC32校验值，以及程序的激活状态。

公开密钥: NSUTIL工具生成，升级验签使用。

reserve: 保留字段，用于扩展。

3.2 init packet

串口升级Init packet使用

大小 (字节)	名称	说明
4	CRC	数据校验值
4	App_start_address	新固件首地址
4	App_size	新固件大小 (单位字节)
4	App_crc	新固件校验值
4	App_version	新固件版本
4*10	Reserve	保留

3.3 dfu_setting

NSUTIL.exe制作升级包自动生成，蓝牙升级中使用，用于对升级固件的验签，和升级固件的完整性校验。

大小（字节）	名称	说明
4	CRC	DFU_SETTING数据校验值
4*4	APP 1 参数	4字节start address程序起始地址 4字节size程序大小 4字节crc程序校验值 4字节version程序版本号
4*4	APP 2 参数	4字节start address程序起始地址 4字节size程序大小 4字节crc程序校验值 4字节version程序版本号
4*4	Image Update 参数	4字节start address程序起始地址 4字节size程序大小 4字节crc程序校验值 4字节version程序版本号
64	Signature	HASH数据+私钥生成签名文件

$CRC = CRC32 (APP\ 1\ 参数 + APP\ 2\ 参数 + Image\ Update\ 参数 + Signature)$

APP 1 参数：升级固件APP1程序的起始地址，程序大小，程序CRC，以及程序版本号。

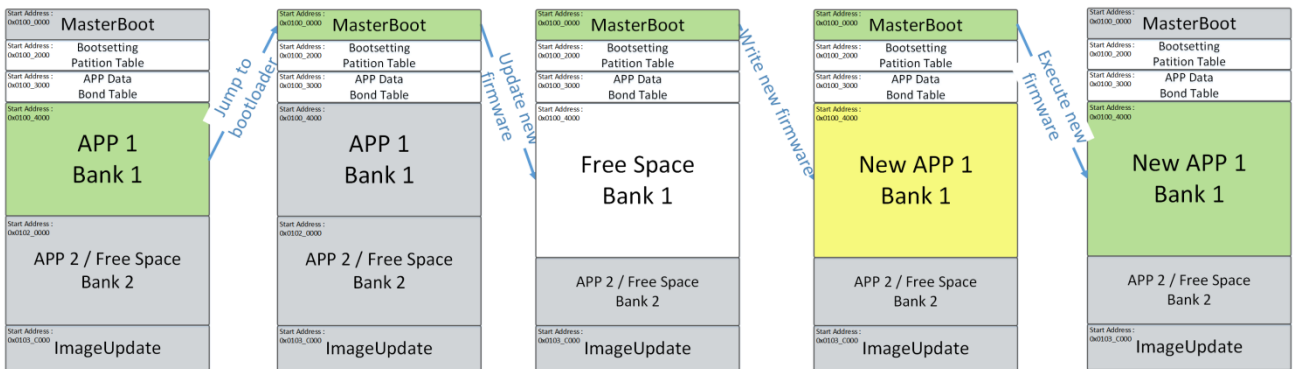
APP 2 参数：升级固件APP1程序的起始地址，程序大小，程序CRC，以及程序版本号。

Image Update 参数：升级固件Image Update程序的起始地址，程序大小，程序CRC，以及程序版本号。

$Signature = ECC_ECDSA_SHA256_NIST256P (APP\ 1\ 参数 + APP\ 2\ 参数 + Image\ Update\ 参数)$

4 升级流程

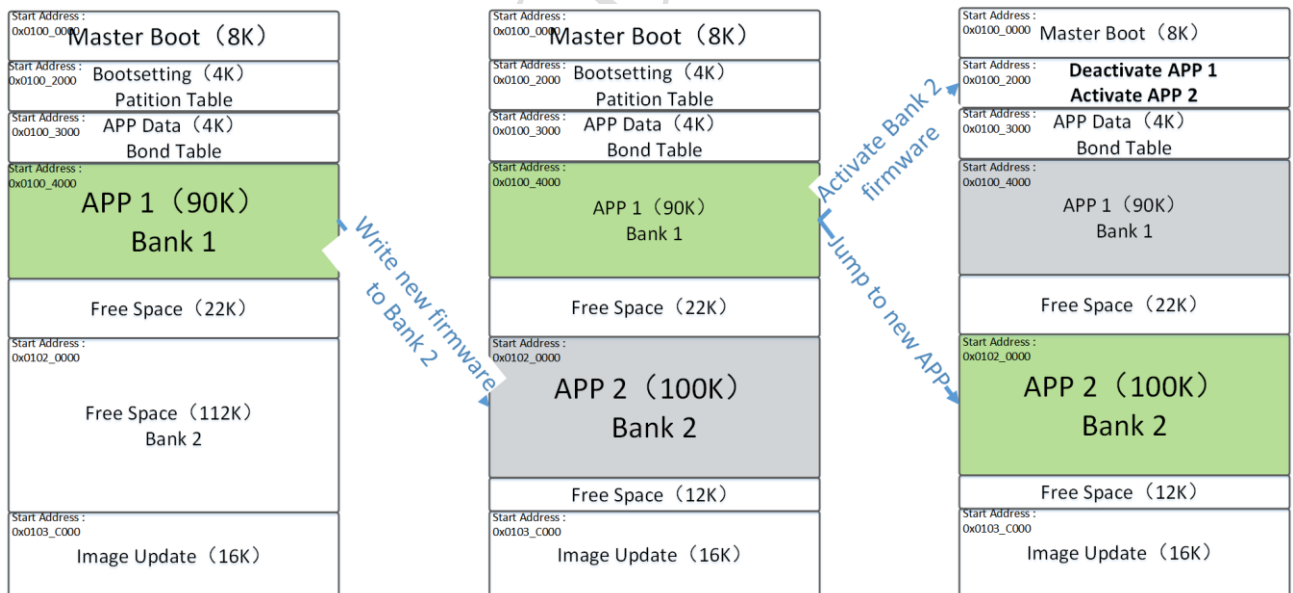
4.1 串口升级流程



用户的应用程序将bootsetting数据中的MasterBootForceUpdate变量置位，软件复位，进入MasterBoot程序。MasterBoot程序检查MasterBoot Force Update被置位，进入串口升级流程。

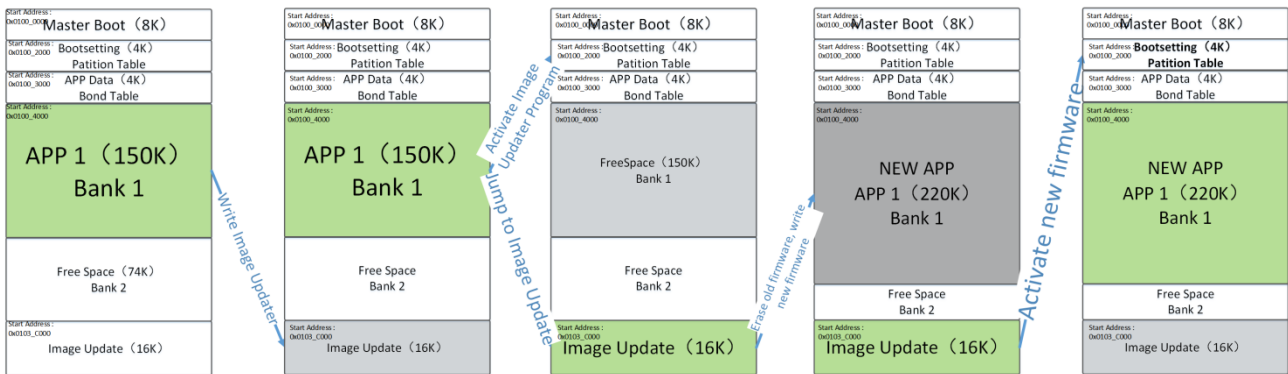
MasterBoot通过串口接收新固件，擦除bank 1区域原有固件，并写入新固件，全部固件写完，激活新固件，最后跳入新固件执行。

4.2 蓝牙双bank升级流程



“双bank”升级采用APP1升级APP2，或者APP2升级APP1的方法，升级速度更快，更稳定，缺点：用户程序只能使用一半FLASH区域，制作升级包时需要多生成一个bank2的bin文件。

4.3 蓝牙单bank升级流程

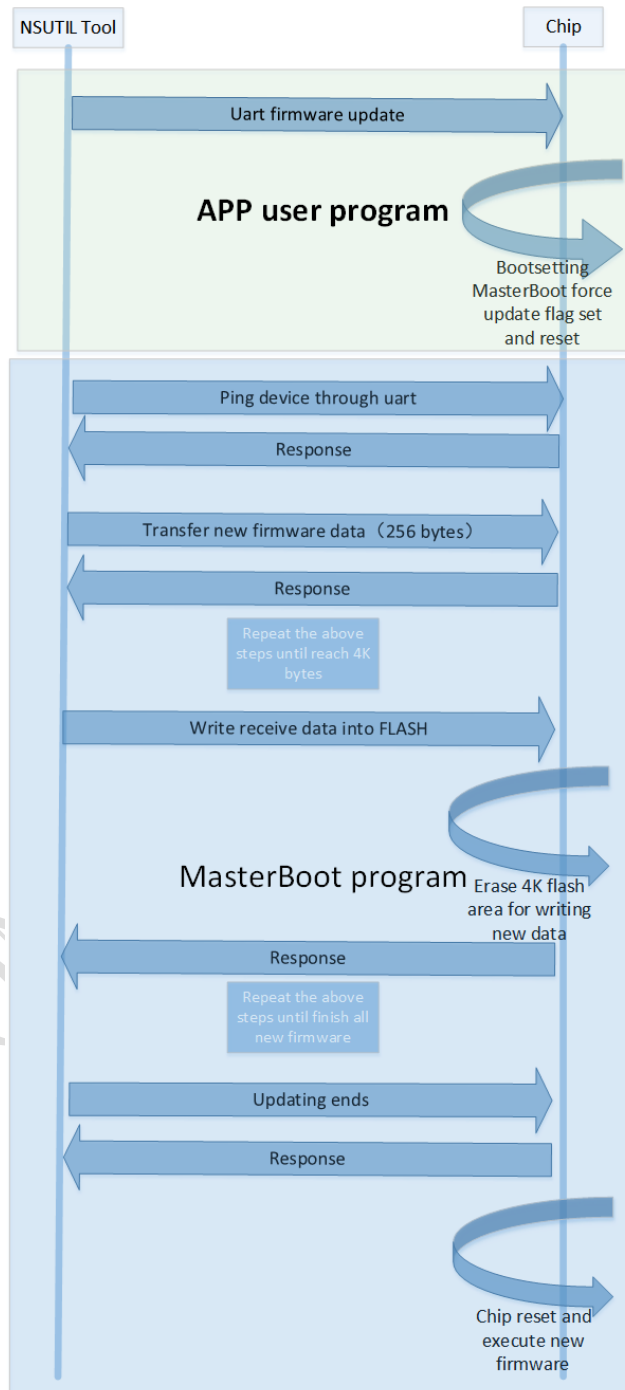


“单bank”升级采用ImageUpdate程序升级APP1程序，优点：用户程序可以使用全部FLASH区域，缺点：升级速度慢，因为蓝牙需要断开重连导致连接不稳定，以及升级失败没有备份。

NATIONS CONFIDENTIAL

5 升级命令

5.1 串口升级命令



5.1.1 Usart dfu

PC上位机命令芯片进入串口升级模式，芯片会复位进入MasterBoot串口升级模式。

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(PC->芯片)
命令号	1	0x07	强制进入串口升级
参数	3	0x01,0x02,0x03	防止误判

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(芯片->PC)
响应号	1	0x07	

5.1.2 Ping

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(PC->芯片)
命令号	1	0x01	PC上位机尝试与芯片通信

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(芯片->PC)
响应号	1	0x01	

5.1.3 Init packet

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(PC->芯片)
命令号	1	0x02	整个新固件头文件
INIT PACKET	Init packet 大小		详情见: init packet

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(芯片->PC)
响应号	1	0x02	
错误码	1		详情见: 错误代码

5.1.4 Packet header

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(PC->芯片)
命令号	1	0x03	一包数据头文件
OFFSET	4		当前升级文件偏移位置
SIZE	4		将要发送升级数据大小
CRC	4		将要发送升级包数据校验

			值
--	--	--	---

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(芯片->PC)
响应号	1	0x03	
错误码	1		详情见：错误代码

5.1.5 Packet

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(PC->芯片)
命令号	1	0x04	升级包数据
数据	<=256-3		按照包头偏移的升级包数据

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(芯片->PC)
响应号	1	0x04	
错误码	1		详情见：错误代码

5.1.6 Postvalidate

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(PC->芯片)
命令号	1	0x05	校验接收数据

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(芯片->PC)
响应号	1	0x05	
错误码	1		详情见：错误代码

5.1.7 Activate&Reset

名称	大小 (字节)	数值	说明
串口头	1	0xAA	(PC->芯片)
命令号	1	0x06	激活新固件，软件复位

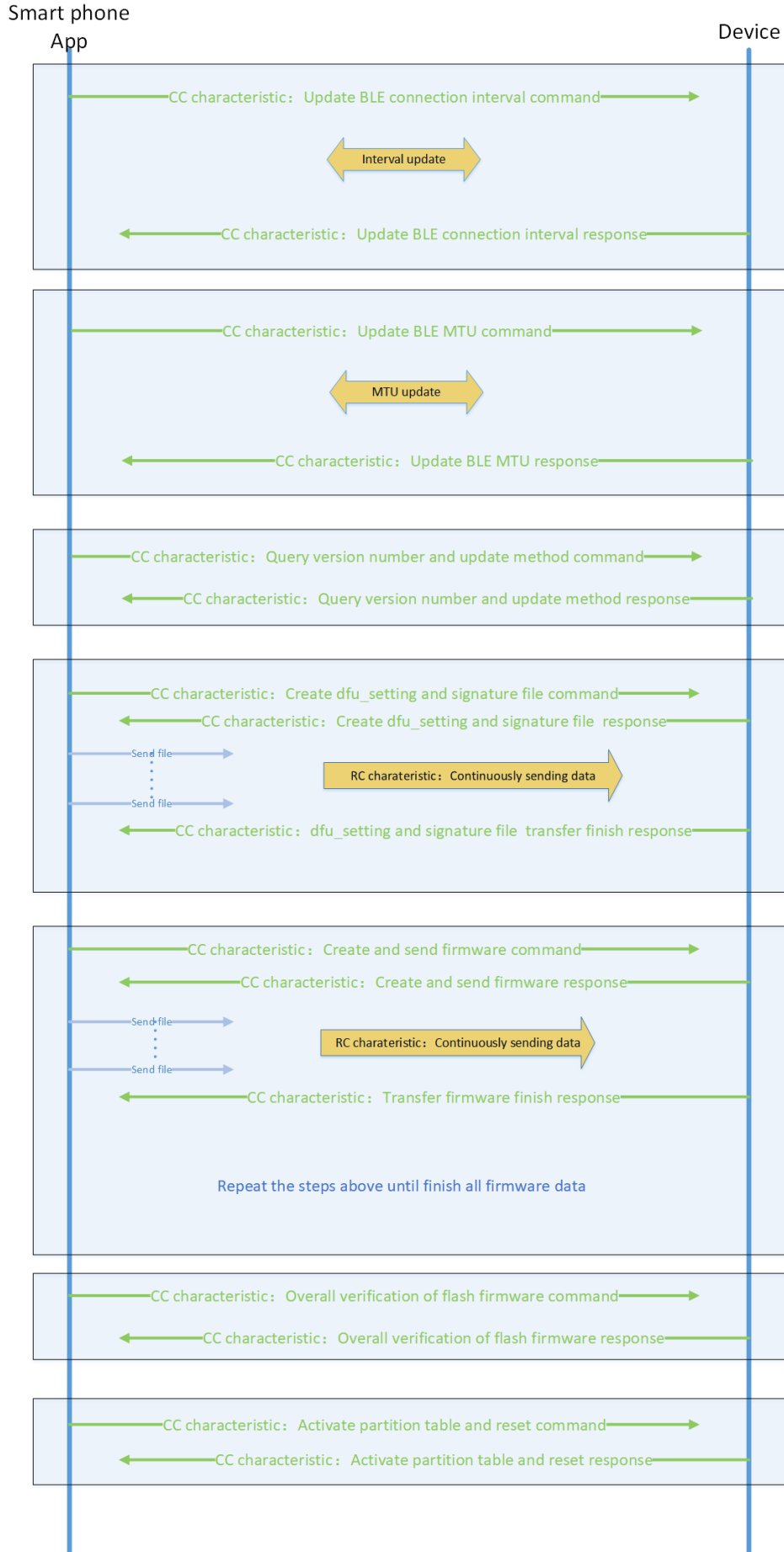
名称	大小 (字节)	数值	说明
串口头	1	0xAA	(芯片->PC)
响应号	1	0x06	
错误码	1		详情见：错误代码

5.2 蓝牙升级命令

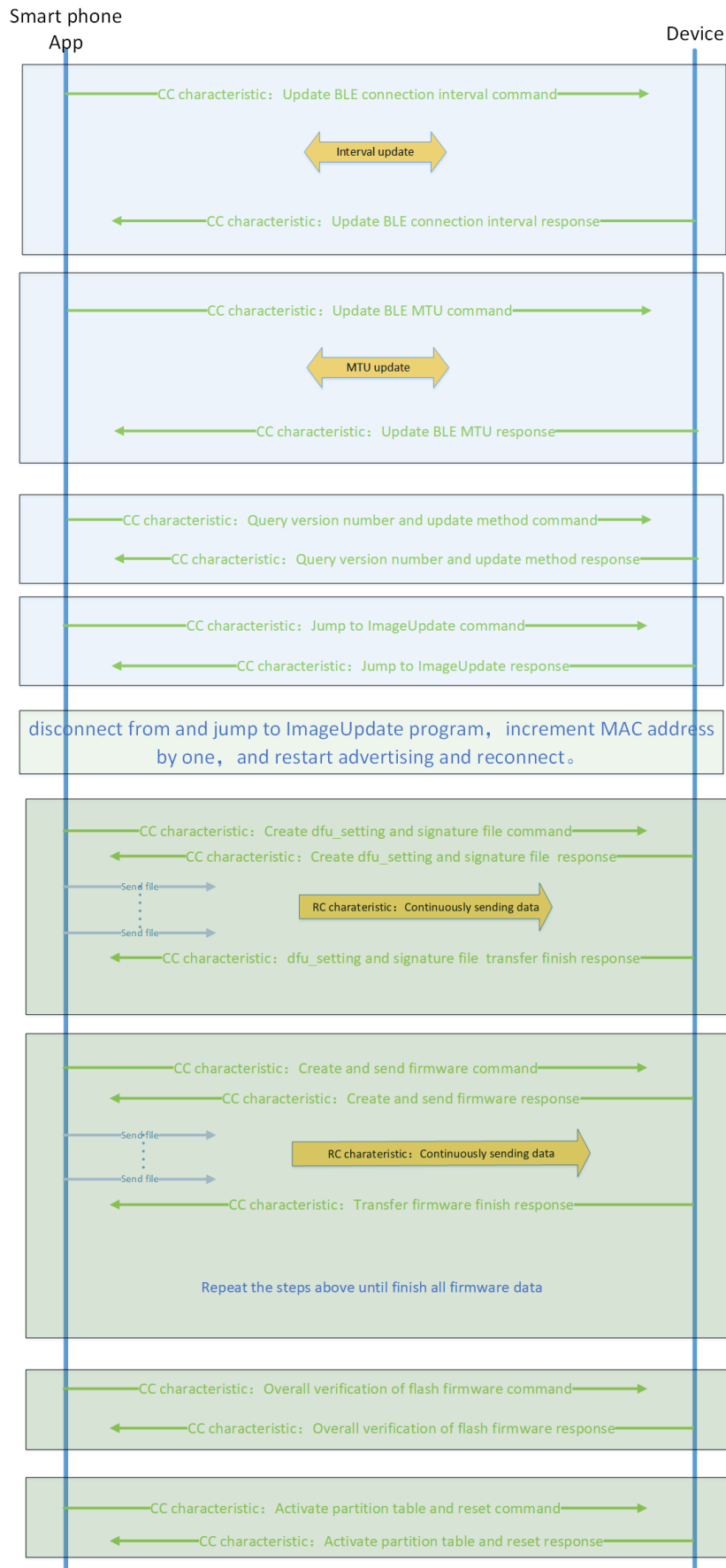
- 手机端通过CC特征发起命令，通过CC特征接收响应。
- 升级包数据流通过RC特征发送给设备。
- 服务和特征

类型	名称	UUID (16进制)	属性	MTU	功能说明
服务	IUS(Image Update Service)	11-11-11-11-11-11-11-11-11-11-11-11-00-01-11-11	Primary		固件升级服务
特征	RC(Receive Characteristic)	11-11-11-11-11-11-11-11-11-11-11-11-00-02-11-11	Write Without Response	20-244	固件接收特征
特征	CC(Command Characteristic)	11-11-11-11-11-11-11-11-11-11-11-11-00-03-11-11	Notiry, Write	20	命令收发特征

- 蓝牙双bank升级



● 单bank升级



5.2.1 更新BLE连接间隙命令

- 升级时减小BLE连接间隙，加快BLE传输速度，手机将连接间隙参数发给从设备，由从设备发起连接间隙参数更新命令。
- 原因是：不确定Android和IOS是否开放了连接间隙参数更新的上层命令。
- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
命令号	1	1	（手机->设备）
最小连接间隙	2		单位： 1.25 MS
最大连接间隙	2		单位： 1.25 MS
SLAVE LATENCY	2		从设备少响应个数
连接超时	2		单位： 10 MS

- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
响应号	1	1	（设备->手机）
错误码	1		详情见：错误代码

5.2.2 更新BLE MTU命令

- 加大RC特征MTU，手机将新MTU大小发给从设备，由从设备发起MTU更新命令。
- 原因是：不确定IOS是否开放了MTU更新的上层命令。
- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
命令号	1	2	（手机->设备）
新MTU大小	2		根据手机型号定义

- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
响应号	1	2	（设备->手机）
错误码	1		详情见：错误代码

5.2.3 查询版本号和升级方式命令

- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
命令号	1	3	（手机->设备）
新APP1固件大小	4		单位（字节）
新APP1固件大小	4		单位（字节）
新IMAGE UPDATE固件大小	4		单位（字节）
新IMAGE UPDATE固	4		

件版本			
-----	--	--	--

- 命令格式：（CC特征）

名称	大小（字节）	数值		说明
响应号	1	3		（设备->手机）
APP1版本号	4			分区表Bank 1 Version
APP2版本号	4			分区表Bank 2 Version
IMAGE UPDATE版本号	4			分区表IMAGE UPDATE Version
升级方式	1	value	meaning	设备读取分区表，计算剩余空间是否能容纳新固件，如果能容纳选择双Bank升级，如果不能容纳选择单Bank升级。
		1	选择APP1	
		2	选择APP2	
		3	选择IMAGE UPDATE	
		4	跳入IMAGE UPDATE	

5.2.4 创建发送dfu_setting和签名文件命令

- 设备端对接收到的dfu_setting文件进行签名验证，判断签名文件是否合法。
- 升级结束后设备端可以使用dfu_setting中的分区表更新自己本地分区表。
- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
命令号	1	4	（手机->设备）
新DFU SETTING大小	4		设备端通过RC特征接收完该大小数据后，通过CC特征通知手机接收完毕。

- 新Bootsetting和签名文件接收完毕命令
- 设备端需要对Bootsetting进行CRC32完整性校验，另外需要对电子签名进行合法性检验。
- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
响应号	1	4	（设备->手机）
错误码	1		详情见：错误代码

5.2.5 创建发送固件数据命令

- 手机端通知设备端RC特征将要接收到的数据偏移值，数据大小，以及数据CRC校验值。

- 设备端RC特征接收完毕后，通过CC特征通知手机数据是否成功接收。
- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
命令号	1	5	（手机->设备）
固件数据偏移地址	4		
固件数据传输大小	4		小于等于2048
固件数据校验CRC	4		

- 固件数据接收完毕命令
- 接收完毕后，设备端需要对接收到的数据进行CRC验证，验证通过写入FLASH，如果达到4K偏移地址，需要对FLASH向后擦除4K。
- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
响应号	1	5	（设备->手机）
错误码	1		详情见：错误代码

5.2.6 FLASH固件整体校验命令

- 设备端对拷贝的固件进行CRC校验，与新分区表中的固件CRC进行对比，返回结果给手机。
- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
命令号	1	6	（手机->设备）

- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
响应号	1	6	（设备->手机）
错误码	1		详情见：错误代码

5.2.7 激活分区表和复位命令

- 设备端修改本地分区表对应的固件为激活状态，响应手机命令，再执行软件复位。
- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
命令号	1	7	（手机->设备）

- 命令格式：（CC特征）

名称	大小（字节）	数值	说明
响应号	1	7	（设备->手机）
错误码	1		详情见：错误代码

5.2.8 跳入ImageUpdate

- 命令格式：（CC特征）

名称	大小 (字节)	数值	说明
命令号	1	8	(手机->设备)

- 命令格式: (CC特征)

名称	大小 (字节)	数值	说明
响应号	1	8	(设备->手机)
错误码	1		详情见: 错误代码

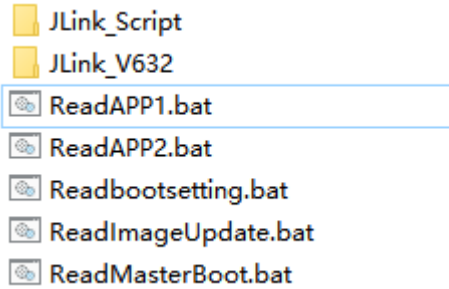
5.3 错误代码

数值	说明
0	成功
1	参数错误
2	CRC错误
3	电子签名错误

6 工具讲解

6.1 JLINK工具

进入N32WB03x_SDK\utilities\dfu\JLink\目录



JLink_V632文件夹中有SEGGER公司制作的JLINK工具，用户需要安装JLINK驱动才能使用。

JLink_Script文件夹中有JLink脚本文件，用户可以直接双击《ReadXX.bat》文件读取芯片FLASH中的数据。

JLink还有芯片FLASH擦除和编程功能，前面章节中使用的《JLINKProgramming.bat》文件就是实现擦除和编程功能。

6.2 NSUTIL工具

进入N32WB03x_SDK\utilities\dfu\NSUtil\文件夹。

- NSUtil.exe Windows平台执行程序。
- Source文件夹存放NSUtil.exe 的python源码。
- Python代码量少，开发时间短。

此工具实现的功能有：

- 制作[bootsetting.bin](#)文件
- 串口升级PC上位机
- 蓝牙升级包打包工具
- ECC密钥生成，数字签名工具

6.3 NSANDROIDUTIL工具

进入N32WB03x_SDK\utilities\dfu\NSAndroidUtil\文件夹。

- NSAndroidUtil.apk 安卓安装包。

此工具实现功能：

- 蓝牙OTA升级。

NATIONS CONFIDENTIAL

7 例程讲解

进入N32WB03x_SDK\projects\n32wb03x_EVAL\dfu\目录。

```

└─ app_ota
└─ app_usart
└─ common
└─ image_update
└─ masterboot

```

7.1 MasterBoot讲解

```

int main(void)
{
    masterboot();

    *(uint32_t*)0x40007014 = 0x0000080F;
    *(uint32_t*)0x40007020 = 0x00020018;

    PWR->VTOR_REG = 0x81000000;//set vector table
    RCC->CFG &= ~1;//set hsi as system clock
    while((RCC->CFG & (1<<2)));//wait for hsi as system clock

    dfu_leds_config();
    dfu_led_on(LED1_GPIO_PORT, LED_GPIO1_PIN);
    dfu_led_on(LED2_GPIO_PORT, LED_GPIO2_PIN);

    NS_SCHED_INIT(256, 16);
    ns_dfu_serial_init();

    while(1)
    {
        app_sched_execute();
        __WFE();
        __SEV();
        __WFE();
    }
}

```

- Masterboot(): 读取bootsetting分区表，直接跳入被激活且校验完整的固件。
- 点亮两个灯示意。
- 初始化简单调度。
- 初始化串口。
- 等待串口中断接收数据。

```
static void sched_evt(void * p_event_data, uint16_t event_size)
{
    switch(*(uint8_t *)p_event_data)
    {
        case SCHED_EVT_RX_DATA:
            if(m_buffer[0] == 0xAA)
            {
                switch(m_buffer[1]){
                    case DFU_SERIAL_CMD_Ping:
                        dfu_serial_cmd_ping();
                    }break;
                    case DFU_SERIAL_CMD_InitPkt:
                        dfu_serial_cmd_init_pkt();
                    }break;
                    case DFU_SERIAL_CMD_Pkt_header:
                        dfu_serial_cmd_pkt_header();
                    }break;
                    case DFU_SERIAL_CMD_Pkt:
                        dfu_serial_cmd_pkt();
                    }break;
                    case DFU_SERIAL_CMD_PostValidate:
                        dfu_serial_cmd_postvalidate();
                    }break;
                    case DFU_SERIAL_CMD_ActivateReset:
                        dfu_serial_cmd_activate_reset();
                    }break;
                    case DFU_SERIAL_CMD_JumpToMasterBoot:
                        dfu_serial_cmd_jump_to_master_boot();
                    }break;
                }
            }break;
    }
}
```

- 串口命令解析，处理，响应上位机。

7.2 AppUsart讲解

- 串口升级APP 1程序。

```

1 int main(void)
2 {
3     *(uint32_t*)0x40007014 = 0x0000080F;
4     *(uint32_t*)0x40007020 = 0x00020018;
5
6     PWR->VTOR_REG = CURRENT_APP_START_ADDRESS | 0x80000000;
7
8     dfu_leds_config();
9     if(CURRENT_APP_START_ADDRESS == NS_APP1_START_ADDRESS) {
10        dfu_led_on(LED1_GPIO_PORT, LED_GPIO1_PIN);
11    }else if(CURRENT_APP_START_ADDRESS == NS_APP2_START_ADDRESS) {
12        dfu_led_on(LED2_GPIO_PORT, LED_GPIO2_PIN);
13    }
14
15    NS_SCHED_INIT(256, 16);
16
17    dfu_flash_init();
18    dfu_usart1_interrupt_config();
19    dfu_usart1_enable();
20    while(1)
21    {
22        app_sched_execute();
23        __WFE();
24        __SEV();
25        __WFI();
26    }
27 }

```

- 判断当前处于bank 1还是bank 2中，bank1则亮LED1，bank2则亮LED2。
- 初始化串口。
- 等待串口中断接收数据。

```

static void sched_evt(void * p_event_data, uint16_t event_size)
{
    switch(*(uint8_t *)p_event_data)
    {
        case SCHED_EVT_RX_DATA:{
            if(m_buffer[0] == 0xAA)
            {
                switch(m_buffer[1]){
                    case DFU_SERIAL_CMD_JumpToMasterBoot:{
                        if(m_buffer[2] == 0x01 && m_buffer[3] == 0x02 && m_buffer[4] == 0x03)
                        {
                            uint8_t cmd[] = {0xAA,DFU_SERIAL_CMD_JumpToMasterBoot,0};
                            serial_send_data(cmd, sizeof(cmd));

                            if(ns_dfu_boot_force_usart_dfu() == false){
                                uint8_t cmd[] = {0xAA,DFU_SERIAL_CMD_JumpToMasterBoot,2};
                                serial_send_data(cmd, sizeof(cmd));
                            }
                        }
                        }else
                        {
                            uint8_t cmd[] = {0xAA,DFU_SERIAL_CMD_JumpToMasterBoot,1};
                            serial_send_data(cmd, sizeof(cmd));
                        }
                    }break;
                }
            }break;
        }
    }
}

```

- 处理串口命令，响应PC上位机，写入强制串口升级标志，服务跳入MasterBoot程序。

7.3 AppOTA讲解

```

249 /**
250  * @brief ble initialization
251  * @param
252  * @return
253  * @note
254  */
255 void app_ble_init(void)
256 {
257     struct ns_stack_cfg_t app_handler;
258     app_handler.ble_msg_handler = app_ble_msg_handler;
259     app_handler.user_msg_handler = app_user_msg_handler;
260     //initialization ble stack
261     ns_ble_stack_init(&app_handler);
262
263     app_ble_gap_params_init();
264     app_ble_sec_init();
265     app_ble_adv_init();
266     app_ble_prf_init();
267     //start adv
268     ns_ble_adv_start();
269 }

```

- 配置蓝牙MAC地址。

- 配置蓝牙广播名称。
- 添加IUS服务。

```

18
19
20 void ns_dfu_ble_handler_cc(uint8_t const *input, uint8_t input_len, uint8_t *output, uint8_t *output_len)
21 {
22
23     switch(input[0]){
24
25         case OTA_CMD_CONN_PARAM_UPDATE:{
26             struct gapc_conn_param conn_param;
27             conn_param.intv_min = input[1]<<8 | input[2];
28             conn_param.intv_max = input[3]<<8 | input[4];
29             conn_param.latency = input[5]<<8 | input[6];
30             conn_param.time_out = input[7]<<8 | input[8];
31             app_env.manual_conn_param_update = 1;
32             app_update_param(&conn_param);
33             *output_len = 0;
34         }break;
35         case OTA_CMD_MTU_UPDATE:{
36             app_env.manual_mtu_update = 1;
37             app_mtu_set(input[1]<<8 | input[2]);
38             *output_len = 0;
39         }break;
40
41         case OTA_CMD_VERSION:{
42             rc_mtu_offset = 0;
43             memset(&m_ota_image,0,sizeof(m_ota_image));
44             uint32_t new_app1_size = input[1]<<24 | input[2]<<16 | input[3]<<8 | input[4];
45             uint32_t new_app2_size = input[5]<<24 | input[6]<<16 | input[7]<<8 | input[8];
46             uint32_t new_image_update_size = input[9]<<24 | input[10]<<16 | input[11]<<8 | input[12];
47             uint32_t new_image_update_version = input[13]<<24 | input[14]<<16 | input[15]<<8 | input[16];
48
49             #ifdef APPLICATION
50
51                 ota_selection = 0;
52
53             if(CURRENT_APP_START_ADDRESS == NS_APP1_START_ADDRESS){
54                 if(ns_bootsetting.appl.size > NS_APP1_DEFAULT_SIZE || new_app1_size > NS_APP1_DEFAULT_SIZE || new_
55                     if(ns_bootsetting.ImageUpdate.crc == dfu_crc32((uint8_t *)((uint32_t *)ns_bootsetting.ImageUpdat
56                         if(new_image_update_version > ns_bootsetting.ImageUpdate.version){
57

```

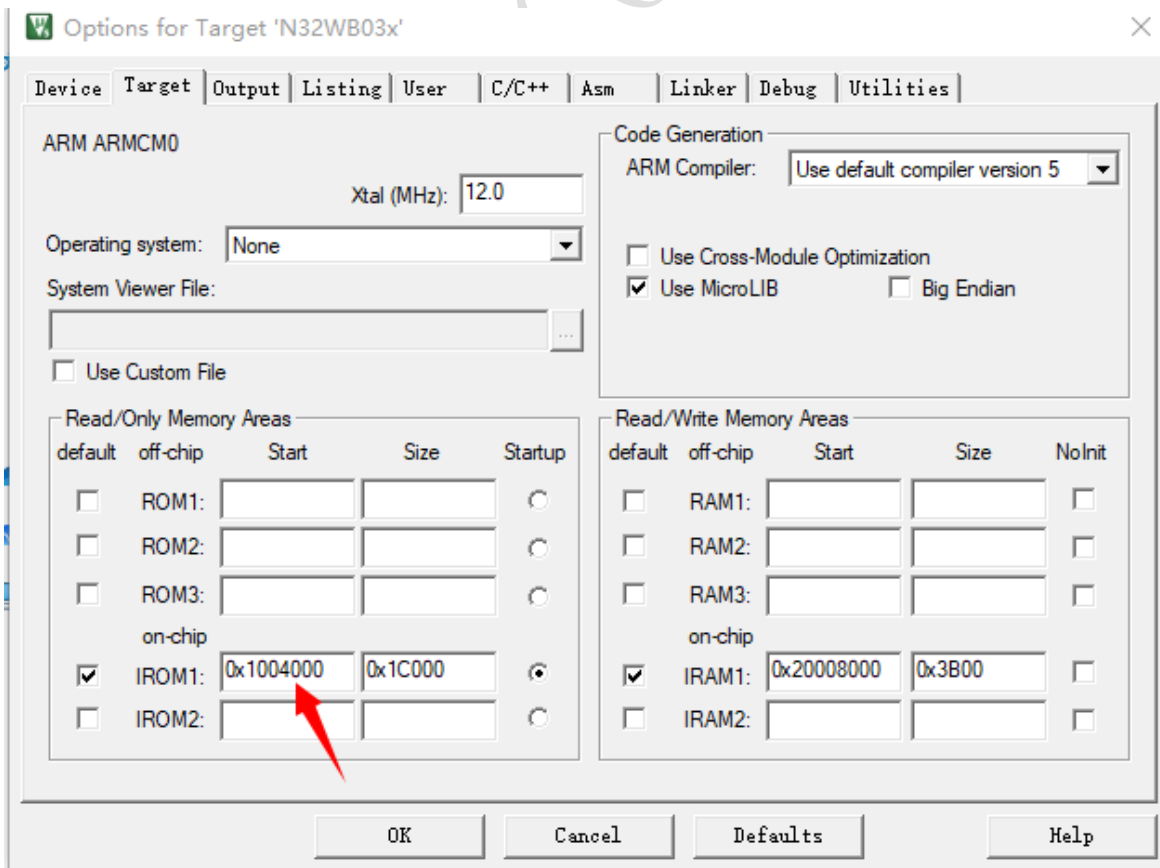
- 处理CC特征命令

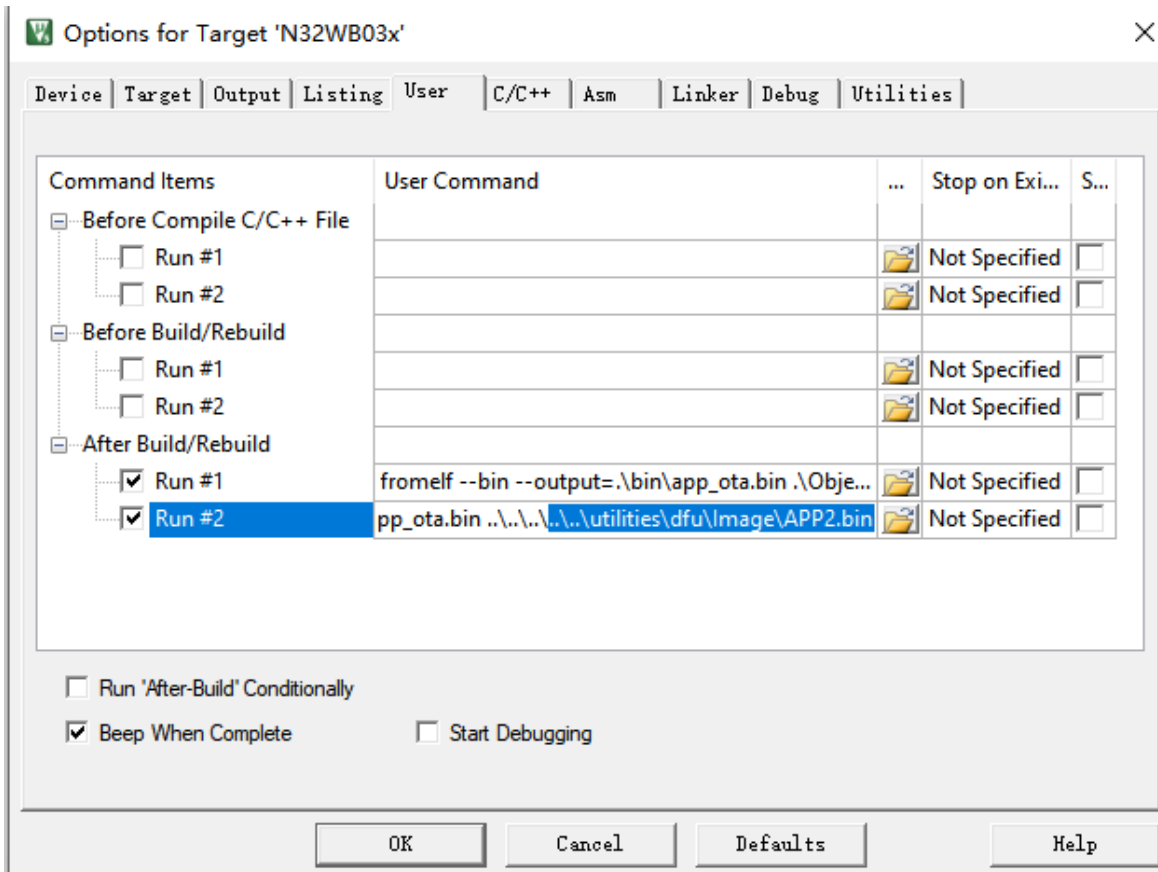
```

301
302
303
304 void ns_dfu_ble_handler_rc(uint8_t const *input, uint32_t input_len)
305 {
306     switch(m_rc_state) {
307
308     case OTA_RC_STATE_DFU_SETTING: {
309         m_rc_state = OTA_RC_STATE_NONE;
310         memcpy(&m_dfu_setting, input, sizeof(Dfu_setting_t));
311         uint32_t crc = dfu_crc32((uint8_t *)&m_dfu_setting.crc + 4, sizeof(Dfu_setting_t) - 4);
312         if(crc == m_dfu_setting.crc) {
313             uint8_t error = 0;
314
315             #if OTA_ECC_ECDSA_SHA256_ENABLE
316             uint8_t raw_data[sizeof(Dfu_setting_bank_t)*3];
317             memcpy(raw_data,&m_dfu_setting.appl,sizeof(Dfu_setting_bank_t));
318             memcpy(raw_data+sizeof(Dfu_setting_bank_t),&m_dfu_setting.app2,sizeof(Dfu_setting_bank_t));
319             memcpy(raw_data+sizeof(Dfu_setting_bank_t)*2,&m_dfu_setting.image_update,sizeof(Dfu_setting_bank_t));
320             uint8_t hash_digest[32];
321             if(ERROR_SUCCESS == ns_lib_ecc_hash_sha256(raw_data, sizeof(Dfu_setting_bank_t)*3, hash_digest)){
322                 if(ERROR_SUCCESS != ns_lib_ecc_ecdsa_verify(ns_bootsetting.public_key, hash_digest, 32, m_dfu_se
323                     error = 3;
324             }
325             }else{
326                 error = 3;
327             }
328             #endif
329
330             uint8_t response[2] = {OTA_CMD_CREATE_OTA_SETTING};
331             response[1] = error;
332             ns_ble_ius_app_cc_send(response,sizeof(response));
333         }else{
334             uint8_t response[2] = {OTA_CMD_CREATE_OTA_SETTING};
335             response[1] = 2;
336             ns_ble_ius_app_cc_send(response,sizeof(response));
337         }
338     }
339 }

```

- 处理RC特征命令和数据
- 制作升级包，需要用户修改keil的Options for Target中的IROM1和After Build Run #2 后的APP后的数字。下面是具体方法。





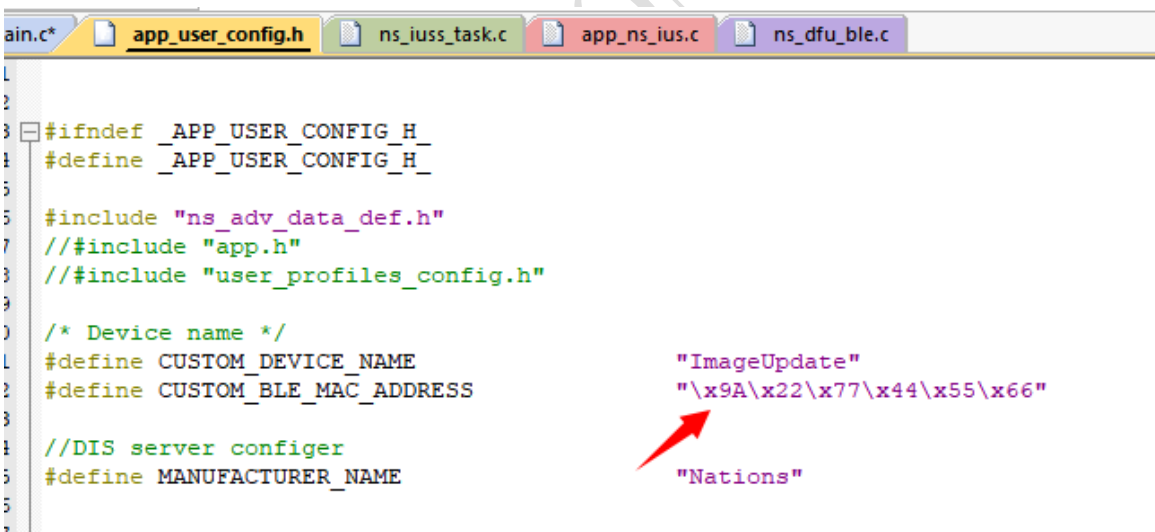
- IROM1: 0x1004000 对应Bank 1的APP 1程序地址； Run #2 修改成APP1（Image文件夹下对应生成APP1.bin文件）
- IROM1: 0x1020000 对应Bank 1的APP 1程序地址； Run #2 修改成APP2（Image文件夹下对应生成APP2.bin文件）

7.4 ImageUpdate讲解

```

3
4 int main(void)
5 {
6     *(uint32_t*)0x40007014 = 0x0000080F;
7     *(uint32_t*)0x40007020 = 0x00020018;
8
9     PWR->VTOR_REG = 0x00000000;
10    NS_BLE_STACK_INIT();
11
12
13    RCC->CFG &= ~1;
14    while((RCC->CFG & (1<<2)));
15
16    bootsetting_reset();
17
18    app_init();
19    prf_init(RWIP_INIT);
20
21    while(1)
22    {
23        rwip_schedule();
24    }
25 }
26

```



```

1
2
3 #ifndef _APP_USER_CONFIG_H_
4 #define _APP_USER_CONFIG_H_
5
6 #include "ns_adv_data_def.h"
7 // #include "app.h"
8 // #include "user_profiles_config.h"
9
10 /* Device name */
11 #define CUSTOM_DEVICE_NAME "ImageUpdate"
12 #define CUSTOM_BLE_MAC_ADDRESS "\x9A\x22\x77\x44\x55\x66"
13
14 //DIS server configer
15 #define MANUFACTURER_NAME "Nations"
16
17

```

- 蓝牙MAC地址最后一位加一。

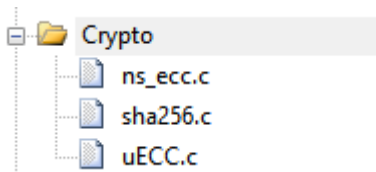
8 加密讲解

```
#define OTA_ECC_ECDSA_SHA256_ENABLE

#if OTA_ECC_ECDSA_SHA256_ENABLE
#include "ns_ecc.h"
#endif
```

图 1

- AppOTA和ImageUpdate程序中使能OTA_ECC_ECDSA_SHA256_ENABLE这个宏，使能升级验签功能。



- 项目包含这些文件，实现验签接口。

```
#if OTA_ECC_ECDSA_SHA256_ENABLE
uint8_t raw_data[sizeof(Dfu_setting_bank_t)*3];
memcpy(raw_data,&m_dfu_setting.appl,sizeof(Dfu_setting_bank_t));
memcpy(raw_data+sizeof(Dfu_setting_bank_t),&m_dfu_setting.app2,sizeof(Dfu_setting_bank_t));
memcpy(raw_data+sizeof(Dfu_setting_bank_t)*2,&m_dfu_setting.image_update,sizeof(Dfu_setting_bank_t));
uint8_t hash_digest[32];
if(ERROR_SUCCESS == ns_lib_ecc_hash_sha256(raw_data, sizeof(Dfu_setting_bank_t)*3, hash_digest)){
    if(ERROR_SUCCESS != ns_lib_ecc_ecdsa_verify(ns_bootsetting.public_key, hash_digest, 32, m_dfu_setting.signature)){
        error = 3;
    }
}else{
    error = 3;
}
#endif
```

- 当嵌入端接收到dfu_setting数据后使用以上方法进行验签。
- 制作升级包，使用ECC对固件CRC，固件大小，和固件其它参数进行签名加密，并保存在dfu_setting中，嵌入端收到dfu_setting后，使用自己已知的公有密钥进行验签，如果验签成功，才能开始真正升级（即擦除FLASH和写入FLASH）。
- 只对固件独有信息进行加密签名，这样加快嵌入端的验签速度，也能够较好的保护固件的升级。

9 常见问题

9.1 运行《JLINKProgramming.bat》批处理问题

WIN7上显示，api-ms-win-core-path-l1-1-0.dll丢失问题,将此dll文件放入 C:\Windows\System32 目录。
可以在其他正版的Windows电脑相同目录找到此dll文件或者咨询技术支持获取。

NATIONS CONFIDENTIAL

10 版本历史

日期	版本	修改
2021.07.29	V1.0	初始版本
2021.12.22	V1.1	根据SDK更新app_ota工程的代码截图，更新常见问题处理描述。
2022.12.28	V1.2	更新脚本为英文版本。

NATIONS CONFIDENTIAL

11 声明

国民技术股份有限公司（以下简称国民技术）保有在不事先通知而修改这份文档的权利。国民技术认为提供的信息是准确可信的。尽管这样，国民技术对文档中可能出现的错误不承担任何责任。在购买前请联系国民技术获取该器件说明的最新版本。对于使用该器件引起的专利纠纷及第三方侵权国民技术不承担任何责任。另外，国民技术的产品不建议应用于生命相关的设备和系统，在使用该器件中因为设备或系统运转失灵而导致的损失国民技术不承担任何责任。国民技术对本文档拥有版权等知识产权，受法律保护。未经国民技术许可，任何单位及个人不得以任何方式或理由对本文档进行使用、复制、修改、抄录、传播等。