User Guide

User Guide for CMT453x Firmware Update

Introduction

This document introduces the principles and application examples of firmware upgrades for the CMT453x, helping users understand the upgrade process to enable faster development.

Key Features:

- Supports firmware upgrades via UART.
- Supports firmware upgrades via Bluetooth.
- Bluetooth MTU supports 20–244 bytes.
- ECC digital signature verification is used in Bluetooth upgrades to ensure firmware authenticity and security.
- Supports dual-bank upgrade via Bluetooth, enabling rollback capability and faster upgrade speeds.
- Offers single-bank Bluetooth upgrade for applications occupying large flash space.
- The system automatically selects between dual-bank and single-bank upgrades—no user intervention required.
- Upgrade speed is adjustable via mobile app to ensure compatibility with different smartphone Bluetooth implementations.

Contents

1 Demonstration

1.1 Demonstration of JLINK Programming

Enter the directory .*CMT453x_SDK\utilities\dfu\Image\JLINKProgrammingDemo*

Double click the script file *JLINKProgramming*.bat to view the Command Prompt information, as shown below.

Step 1: *BootSetting.bin* is generated by HPUtil tool, and "Bootsetting created successfully!" is displayed in the command line, indicating that the file is generated successfully.

```
=======================BootSetting.bin============================
00000000: 66 01 47 D3 FF FF FF FF  00 40 00 01 60 54 00 00  f.G......@..`T..
00000010: C5 16 ED C4 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 FF FF  ................
00000030: 00 00 02 01 2C 55 00 00  C0 68 7B 9D 01 00 00 00  ....,U...h{.....
00000040: FF 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 24 39 00 00  ............$9..
00000060: A2 44 D9 6D 01 00 00 00  FF FF FF FF FF FF FF FF  .D.m............
00000070: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000080: AE 1C 41 A5 F4 35 DD 3D  89 C8 00 D8 0F 8D 2A C2  ..A..5.=......*.
00000090: 63 3A 02 37 24 5D 2D DB  F0 46 A1 6A 5E 43 26 44  c:.7$]-..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
```

Step 2: JLink detect and connect to the device

```
Connecting to target via SWD
Found SW-DP with ID 0x0BB11477
Scanning AP map to find all available APs
AP[1]: Stopped AP scan as end of AP map has been reached
AP[0]: AHB-AP (IDR: 0x04770021)
Iterating through AP map to find AHB-AP to use
AP[0]: Core found
AP[0]: AHB-AP ROM base: 0xE00FF000
CPUID register: 0x410CC200. Implementer code: 0x41 (ARM)
Found Cortex-M0 r0p0, Little endian.
FPUnit: 4 code (BP) slots and 0 literal slots
CoreSight components:
ROMTbl[0] @ E00FF000
ROMTbl[0][0]: E000E000, CID: B105E00D, PID: 000BB008 SCS
ROMTbl[0][1]: E0001000, CID: B105E00D, PID: 000BB00A DWT
ROMTbl[0][2]: E0002000, CID: B105E00D, PID: 000BB00B FPB
Cortex-M0 identified.
Reset delay: 0 ms
Reset type NORMAL: Resets core & peripherals via SYSRESETREQ & VECTRESET bit.
Reset: Halt core after reset via DEMCR.VC_CORERESET.
Reset: Reset device via AIRCR.SYSRESETREQ.
```

Step 3: JLink erases the chip's Flash.

```
Erasing device (CMT4531)...
J-Link: Flash download: Total time needed: 1.278s
Restore: 0.006s)
Erasing done.
```

Step 4: Program *MasterBoot.bin* to the chip's 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: 1.109s (Prepare: 0.051s, Compare: 0.138s,
Restore: 0.021s)
O.K.
```

Step 5: Program *Bootsetting.bin* to the chip's 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: 0.790s (Prepare: 0.049s, Compare: 0.139s,
Restore: 0.023s)
O.K.
```

Step 6: Program *APP1.bin* to the chip's Flash.

```
Downloading file [Image\APP1.bin]...
J-Link: Flash download: Bank 0 @ 0x01000000: 1 range affected (24576 bytes)
J-Link: Flash download: Total time needed: 2.449s (Prepare: 0.052s, Compare: 0.138s,
Restore: 0.022s)
O.K.
```

Step 7: Program *APP2.bin* to the chip's Flash.

```
Downloading file [Image\APP2.bin]...
J-Link: Flash download: Bank 0 @ 0x01000000: 1 range affected (24576 bytes)
J-Link: Flash download: Total time needed: 2.509s (Prepare: 0.053s, Compare: 0.138s,
Restore: 0.022s)
O.K.
```

Step 8: Program *ImageUpdate.bin* to the chip's 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: 1.849s (Prepare: 0.049s, Compare: 0.139s,
Restore: 0.023s)
O.K.
```

Step 9: Reset the chip.

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

JLINK successively executes, erases, and writes in *MasterBoot.bin*, *Bootsetting.bin*, *APP1.bin*, *APP2.bin*, *ImageUpdate.bin*, reset command and completes programming in batch. Programming failure, if any, may be caused by the chip in DEEP SLEEP. We can try to execute the *JLINKProgramming.bat* first before power up the chip.

MasterBoot.bin

It is generated by the MasterBoot Keil project. After compilation, the system will automatically copy the generated firmware to the *dfu\image* directory. For details, refer to the project settings in Keil: Options for Target → User → Run #2.

Bootsetting.bin

Bootsetting.bin is generated by the HPUtil python tool. Refer to Section 1 of Chapter 3 for the bootsetting data structure.

APP1.bin & APP2.bin

They are generated by the AppOTA example project. However, the start addresses for APP1.bin and APP2.bin need to be adjusted accordingly. Their start addresses are 0x1004000 and 0x1020000, respectively. For details, please refer to the Flash memory map. The output path can be found in the Keil settings: Options for Target → User → Run #2.

ImageUpdate.bin

It is generated by the ImageUpdater Keil Project. For details, refer to the project settings in Keil: Options for Target → User → Run #2.

Chapter 2 provides detailed description of the different bin executing address and function of Bin file.

Open the *JLINKProgramming.bat* file using any text editor.

```
set JLink_path=.\JLink\JLink_V632\JLink.exe
set JLink_script_path=.\JLink\JLink_Script\download.jlink
set HPUtil_path=.\HPUtil\HPUtil.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
::ImageUpdate parameters, optional
set public_key_file=.\keys\public_key.bin
```

Users can modify bank1_activation=no, bank2_ Activation=yes, power on and start the bank2 program, and can view the printing output through the serial port connecting to the chip PB1 (115200 N 8 1), or through the blinking frequency of LED1 and LED2. APP1 and APP2 flash in 100 and 500 milliseconds, respectively. Users can search for the Bluetooth device with *HOPERF_APPx* (where x is 1 or 2).

Users can also modify bank1_ activation=no, image_ update_ Activation=yes, check the serial port printing output, or search for the Bluetooth device with *ImageUpdate* device broadcast.

It should be noted that only one program can enable activation.

1.2 Demonstration of UART Firmware Update

Enter the directory *CMT453x_SDK\utilities\dfu\Image\UartProgrammingDemo*

By double clicking the script file *JLINKProgramming*.bat, users can view the Command Prompt information.

Bootsetting.bin reveals that in the chip, only bank1 and bank2 have programs and bank1 program is activated.

```
========================BootSetting.bin==============================
00000000: F0 53 32 7C FF FF FF FF  00 40 00 01 9C 0F 00 00  .S2|.....@......
00000010: FC E8 69 E6 01 00 00 00  01 00 00 00 FF FF FF FF  ..i.............
00000020: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000030: 00 00 02 01 9C 0F 00 00  E3 47 01 4E 01 00 00 00  .........G.N....
00000040: FF FF 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 FF FF  ................
00000060: FF FF 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 FF FF  ................
00000080: FF FF 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 FF FF  ................
000000A0: FF FF 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 FF FF  ................
Bootsetting created successfully!
```

Run the *JLINKProgramming.bat* script to sequentially program MasterBoot.bin, Bootsetting.bin, APP1.bin, and APP2.bin to the device via JLink.

APP1.bin and APP2.bin are generated by the AppUsart example. The output path can be found in the Keil settings: Options for Target → User → Run #2.

```
Downloading file [Image\MasterBoot.bin]...
J-Link: Flash download: Bank 0 @ 0x01000000: 1 range affected (8192 bytes)
J-Link: Flash download: Total time needed: 1.113s (Prepare: 0.046s, Compare: 0.139s,
ore: 0.022s)
O.K.

Downloading file [Image\Bootsetting.bin]...
J-Link: Flash download: Bank 0 @ 0x01000000: 1 range affected (4096 bytes)
J-Link: Flash download: Total time needed: 0.789s (Prepare: 0.051s, Compare: 0.138s,
ore: 0.022s)
O.K.

Downloading file [Image\APP1.bin]...
J-Link: Flash download: Bank 0 @ 0x01000000: 1 range affected (4096 bytes)
J-Link: Flash download: Total time needed: 0.946s (Prepare: 0.048s, Compare: 0.139s,
ore: 0.023s)
O.K.

Downloading file [Image\APP2.bin]...
J-Link: Flash download: Bank 0 @ 0x01000000: 1 range affected (4096 bytes)
J-Link: Flash download: Total time needed: 0.937s (Prepare: 0.047s, Compare: 0.138s,
ore: 0.021s)
O.K.
```

Open the *UartFirmwareUpdate.bat* file using a text editor.

```
set HPUtil_path=..\..\HPUtil\HPUtil.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=COM5
set serial_baudrate=115200



%HPUtil_path% ius serial %app_bin%                                          ^
                          --app_start_address %app_start_address%          ^
                          --app_version %app_version%                      ^
                          --serial_port %serial_port%                       ^
                          --serial_baudrate %serial_baudrate%              ^
                          --force_update false
```

Modify serial_port= to the correct COM port number (which can be found in Device Manager). Connect the USB-to-Serial cable to PB6 (chip TX) and PB7 (chip RX). Save and close the file.
Then double-click the *UartFirmwareUpdate.bat* file and check the output in the command line window.

```
D:\SDKs\CMT\CMT453x_SDK_v1.3.1_alpha\CMT453x_SDK_v1.3.1\utilities\dfu\Image\UartProgrammingDemo>echo off
baudrate --> 115200
Serial Image Update In Progress  [##################################]  100%
 3996 bytes Image Update Complete in 2.06s
```

By default, JLINK programs and executes the Bank1 application (PB0 pin will be pulled high), and the UART upgrade defaults to Bank2 execution (PA6 pin will be pulled high).
Users can also modify app_bin=.\Image\APP1.bin and app_start_address=0x01004000 to make the UART upgrade default to executing the Bank1 application.

1.3 Demonstration of BLE “Dual Bank” Update

Go to the *CMT453x_SDK\utilities\dfu\HPAndroidUtil* directory and install the *HPAndroidUtil.apk* file on your smartphone.
Navigate to *CMT453x_SDK\utilities\dfu\Image\DualBankUpdateDemo* and double-click *JLINKProgramming.bat* to program the firmware into the chip.
Next, go to the *CMT453x_SDK\utilities\dfu\Image\DualBankUpdateDemo\Image* directory and copy the *ota_dual_bank.zip* file to your phone’s internal storage.
Tap 'Connect' in the app, then select the device named HOPERF_APP1.

Wait until the Bluetooth in the upper right corner is Connected, then click the file and select *ota_ dual_ Bank.zip*.

We can view the size of update package in the update package information. After we click Update, the update status changes and the progress bar increases.

As the update finishes, the Bluetooth will automatically disconnect, and the update status displays the update time consumed.

The Bluetooth upgrade package used during the upgrade process, *ota_dual_bank.zip*, can be generated by double-clicking *GenerateUpdateImage.bat*. You can observe the details in the command line window.

```
========================dfu_setting.dat=============================
00000000: 41 68 05 13 00 40 00 01  FC 6A 00 00 62 72 6D 14  Ah...@...j..brm.
00000010: 02 00 00 00 00 00 02 01  C8 6B 00 00 D3 91 E2 E6  .........k......
00000020: 02 00 00 00 00 C0 03 01  18 39 00 00 58 B7 27 2F  .........9..X.'/
00000030: 02 00 00 00 C0 CA EE 5A  79 70 69 0E 29 80 B3 3A  .......Zypi.)..:
00000040: 85 6C 3A 63 F1 34 6B 85  3B 9F 74 AC BE 33 F3 4D  .l:c.4k.;.t..3.M
00000050: 23 D7 36 5B 0A B8 8D 0A  C1 55 DD 9B D0 75 FE B5  #.6[.....U...u..
00000060: CB 14 69 0F 0C 4E 13 17  5B 34 06 47 00 19 D9 26  ..i..N..[4.G...&
00000070: 3A 36 40 9A                                      :6@.
========================config.txt=============================
APP1_START_ADDRESS : 0x01004000
APP1_VERSION : 0x00000002
APP1_SIZE : 0x00006afc
APP2_START_ADDRESS : 0x01020000
APP2_VERSION : 0x00000002
APP2_SIZE : 0x00006bc8
IMAGE_UPDATE_START_ADDRESS : 0x0103c000
IMAGE_UPDATE_VERSION : 0x00000002
IMAGE_UPDATE_SIZE : 0x00003918
```

This batch file creates the Bluetooth upgrade package based on the parameters configured within the file and the .bin

files in the Image folder. The command line window will also display the contents of the *dfu_setting.dat* and *Config.txt* files.

1.4 Demonstration of BLE “Single Bank” update

Enter the directory *CMT453x_SDK\utilities\dfu\HPAndroidUtil*, and install the *HPAndroidUtil.apk* file on the phone. Then enter the directory *CMT453x_SDK\utilities\dfu\Image\SingleBankProgrammingDemo*, double-click *JLINKProgramming.bat* to flash the firmware to the chip.

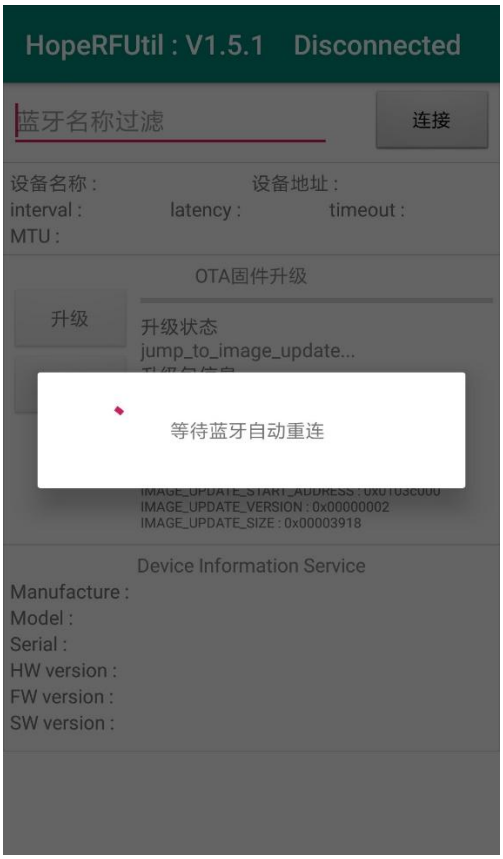
Next, go to the directory *CMT453x_SDK\utilities\dfu\Image\SingleBankProgrammingDemo\Image*, and copy the ota_single_bank.zip file to the phone’s internal storage.

Click "Connect," then click to select the HOPERF_APP1 device.

Wait until the Bluetooth status in the upper right corner changes to "Connected," then click "File" and select the ota_single_bank.zip file. The upgrade package information will show the size of the package.

Click “Upgrade,” observe the upgrade status changes and the progress bar updates.

During the single-bank upgrade process, Bluetooth will disconnect and reconnect once. A dialog box will pop up on the screen asking the user to wait for Bluetooth to automatically reconnect.

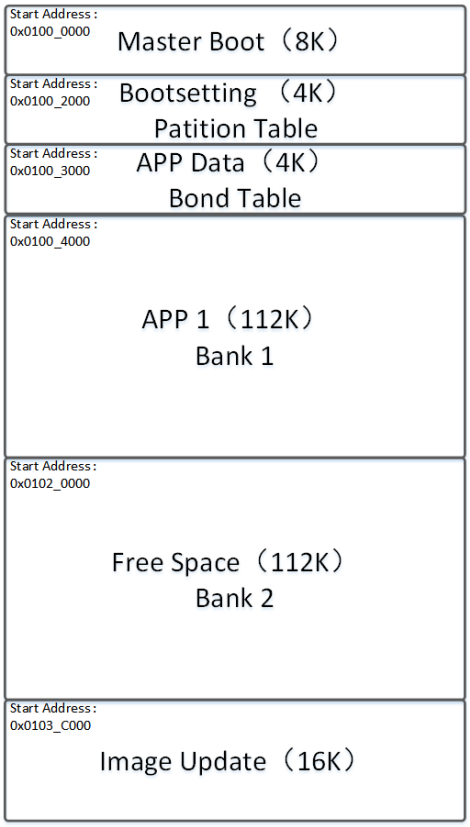


After Bluetooth reconnects, the upgrade continues. When the upgrade finishes, the time taken for the upgrade is displayed.

2 FLASH Memory Map

The FLASH address range of the CMT453x chip is 0x0100_0000 to 0x0103_FFFF, with an available space of 256 KB.

The distribution of program and data FLASH is shown in the following diagram:



Name	CMT453x FLASH address	Description
Master Boot (program)	0x0100_0000 - 0x0100_1FFF (8K)	Program entry after power on; Read the partition table and jump the program; Serial port update function;
Bootsetting (data) (Patition Table)	0x0100_2000 - 0x0100_2FFF (4K)	FLASH partition table;
APP Data (data) (Bond Table) (User Data)	0x0100_3000 - 0x0100_3FFF (4K)	APP data storage area; Storage binding list (about 500 bytes for 5 devices); The remaining space stores user-defined data;
APP 1 (program) (Bank 1)	0x0100_4000 - 0x0101_FFFF (112K)	User program 1 storage area;
Free Space/APP2 (program) (Bank 2)	0x0102_0000 - 0x0103_BFFF (112K)	Reserved space; or user program 2 storage area;

Image Update (program)	0x0103_C000 - 0x0103_FFFF (16K)	Reserved space; or update “single BANK”;

The MasterBoot program provides program jump entry and serial port update.

Bootsetting data provides program jump and updates shared data.

If more space is required by the APP Data in storage area, it is recommended to use external storage or modify the FLASH memory distribution (users are welcome to contact technical support for specific modification methods).

APP1 is a user program.

APP2 is a user program compiled on Bank2.

ImageUpdate is a program for “single bank” update. ImageUpdate is a minimal program that supports Bluetooth OTA. It receives the updated firmware sent from the host and writes it to the flash area where the application resides to complete the program update.

3 Data Structure

3.1 Bootsetting

Size（Bytes）	Name	Description
4	Bootsetting CRC	Bootsetting data check value
4	MasterBoot Force Update	Forced serial port update for MasterBoot 1: serial port update By default: 0xFFFFFFFF
4*10	Bank 1 partition	4 bytes: start address of program 4 bytes: size of program 4 bytes: crc of program 4 bytes: version of program 4 bytes: activation code of program: 1 activation, others 4*5 bytes: reserve
4*10	Bank 2 partition	4 bytes: start address of program 4 bytes: size of program 4 bytes: crc of program 4 bytes: version of program 4 bytes: activation code of program: 1 activation, others 4*5 bytes: reserve
4*10	Image Update partition	4 bytes: start address of program 4 bytes: size of program 4 bytes: crc of program 4 bytes: version of program 4 bytes: activation code of program: 1 activation, others 4*5 bytes: reserve

64	Public key	Used for ECC signature verification.

Bootsetting crc=CRC32 (MasterBoot Force Update+Bank 1 partition+Bank 2 partition+Image Update partition+public key)

MasterBoot Force Update: MasterBoot program decides whether to enter the serial port update mode by judging this variable.

Bank 1 partition: record the start address, program size, CRC32 value, and activation status of APP1 program.

Bank 2 partition: record the start address, program size, CRC32 value, and activation status of APP2 program.

Image Update partition: record the start address, program size, CRC32 value, and activation status of Image Update program.

Public key: generated by the HPUTIL tool, and used for updating signature verification.

Reserve: reserved field for extension.

In the code, the structure is defined as follows:

```
typedef struct
{
    uint32_t start_address;
    uint32_t size;
    uint32_t crc;
    uint32_t version;
    uint32_t activation;
    uint32_t reserve[5];
}HP_Bank_t;

typedef struct
{
    uint32_t crc;
    uint32_t master_boot_force_update;
    HP_Bank_t app1;
    HP_Bank_t app2;
    HP_Bank_t ImageUpdate;
    uint8_t public_key[64];
}HP_Bootsetting_t;
```

3.2 init packet

Serial port update for Init packet

Size (Byte)	Name	Description
4	CRC	Data check value
4	App_start_address	First address of new firmware
4	App_size	Size of new firmware (in bytes)
4	App_crc	Check value of new firmware
4	App_version	Version of new firmware
4*10	Reserve	Reserve

In the code, the structure is defined as follows:

```
typedef struct
{
    uint32_t crc;
    uint32_t app_start_address;
    uint32_t app_size;
    uint32_t app_crc;
    uint32_t app_version;
    uint32_t reserve[10];
}_init_pkt;
```

3.3 dfu_setting

Automatically generated by HPUTIL.exe when creating the upgrade package.
Used during Bluetooth upgrades for signature verification of the upgrade firmware and for integrity checking of the upgrade firmware.

Size (Byte)	Name	Description
4	CRC	DFU_SETTING data check value
4*4	APP 1 parameter	4 bytes: start address of program 4 bytes: size of program 4 bytes: crc of program 4 bytes: version of program
4*4	APP 2 parameter	4 bytes: start address of program 4 bytes: size of program 4 bytes: crc of program 4 bytes: version of program
4*4	Image Update parameter	4 bytes: start address of program 4 bytes: size of program 4 bytes: crc of program 4 bytes: version of program
64	Signature	HASH data+signature file based on private key

CRC=CRC32 (APP 1 parameter+APP 2 parameter+Image Update parameter+Signature)

APP 1 parameter: start address, size, CRC, and version number of update firmware APP1 program.

APP 2 parameter: start address, size, CRC, and version number of update firmware APP2 program.

Image Update parameters: starting address, size, CRC, and version number of the Image Update program for firmware update.

Signature = ECC_ ECDSA_ SHA256_ NIST256P (APP 1 parameter+APP 2 parameter+Image Update parameter)

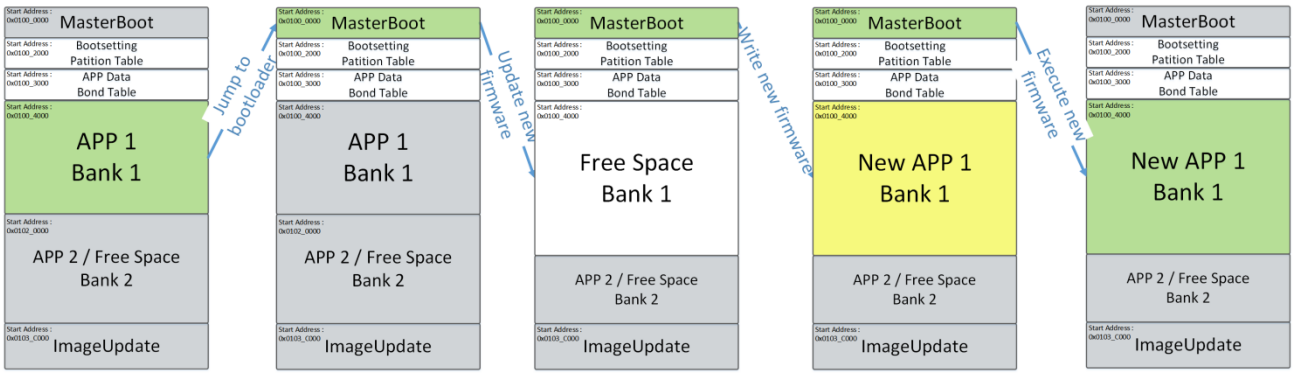
In the code, the structure is defined as follows:

```
typedef struct{
    uint32_t start_address;
    uint32_t size;
    uint32_t crc;
    uint32_t version;
}Dfu_setting_bank_t;
```

```
typedef struct{

    uint32_t crc;
    Dfu_setting_bank_t app1;
    Dfu_setting_bank_t app2;
    Dfu_setting_bank_t image_update;
    uint8_t signature[64];
}Dfu_setting_t;
```

4 Update Process

4.1 UART Firmware update process

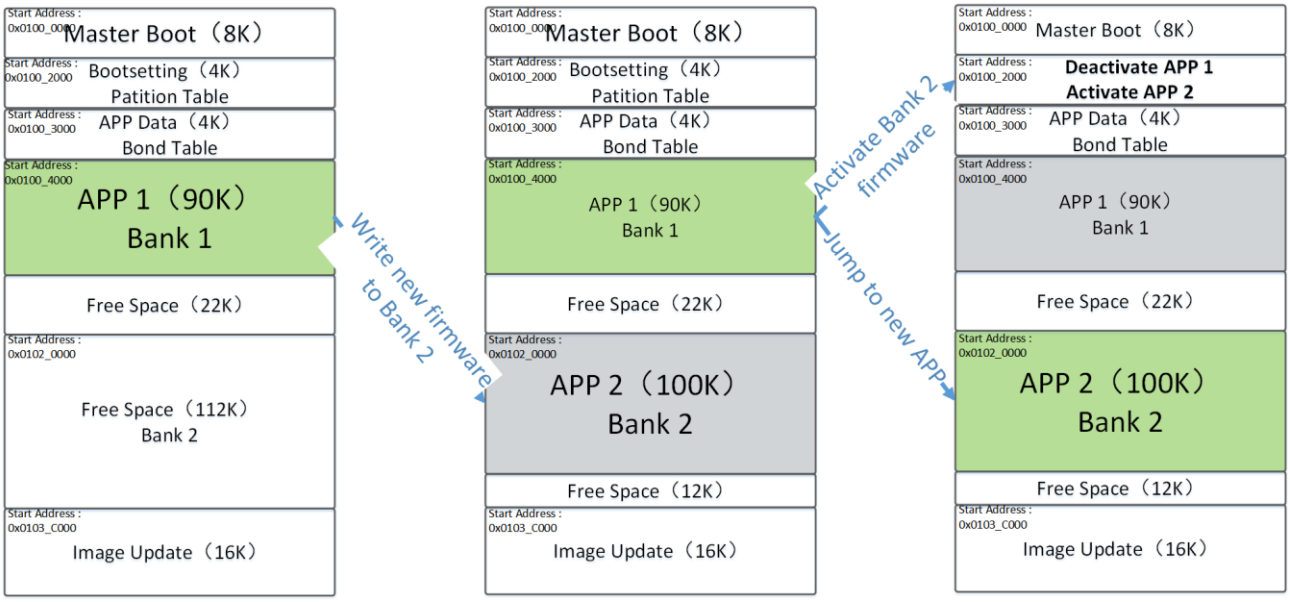


The user's application program sets the MasterBootForceUpdate variable in the bootsetting data, resets the software, and enables the MasterBoot program.

Set the MasterBoot Force Update variable and activate the serial port update process.

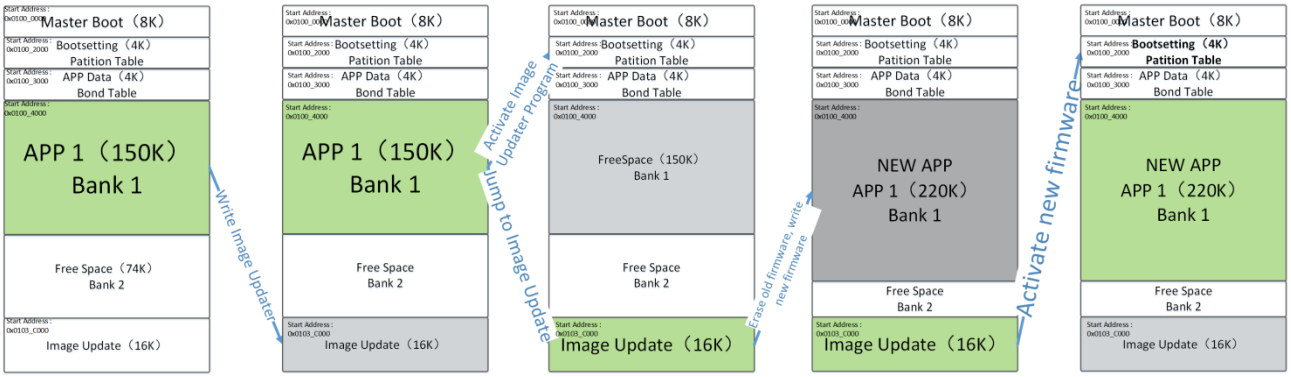
The MasterBoot receives new firmware through serial port, erases the original firmware in the bank 1 area, writes in the new firmware, and afterwards activates them, and finally jumps to new firmware for execution.

4.2 Update process of Bluetooth dual bank



The “dual bank” update is implemented by updating APP1 with APP2, or vice versa. It features faster update speed and higher stability. However, it also faces the disadvantage that the user program can only use half of the FLASH area. An additional bin file of bank2 needs to be generated when we make the update package.

4.3 Update process of Bluetooth single bank



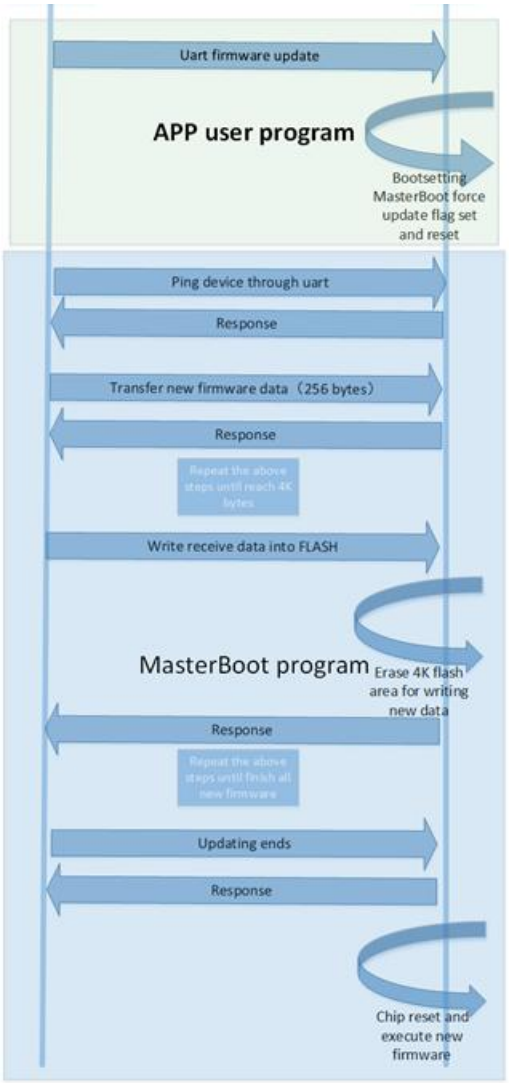
"single bank" update is implemented by updating APP1 program with ImageUpdate program. It has the advantage that the user program can use all FLASH regions, but it also faces the disadvantage of slow update speed, because Bluetooth will disconnection then re-connect causing possible unstable connection and there is no backup when the update fails.

5 Update Command

5.1 Commands for UART Firmware update

The following is the flowchart for serial port upgrade:

1. First, the PC host sends the “USART DFU” command to the device to request it to enter Masterboot serial upgrade mode. Masterboot may also enter serial upgrade mode by other methods, such as detecting an external button on the device. After the slave device receives the “USART DFU” command from the host, it sets the “Master Boot Force Update” flag in Bootsetting to 1 and resets the chip.
2. After the chip resets, it will execute the Master Boot code segment. Since the “Master Boot Force Update” flag is set to 1, the program waits for the host to send commands through the serial port to perform the next upgrade steps.



Below are the commands involved in the serial port upgrade process.

5.1.1 UART DFU

PC upper computer instructs the chip to enter serial port update mode, and the chip will be reset to enter the MasterBoot serial port update mode.

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(PC->chip)
Command number	1	0x07	Forced serial port update
Parameter	3	0x01,0x02,0x03	Prevent false judgment

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(Chip->PC)
Response number	1	0x07	

5.1.2 Ping

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(PC->chip)
Command number	1	0x01	PC upper computer attempts to communicate with chip

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(Chip->PC)
Response number	1	0x01	

5.1.3 Init packet

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(PC->chip)
Command number	1	0x02	Entire new firmware header file
INIT PACKET	Init packet size		Detailed in: init packet

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(Chip->PC)
Response number	1	0x02	
Error code	1		Detailed in: error code

5.1.4 Packet header

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(PC->chip)
Command number	1	0x03	A package of header files
OFFSET	4		Offset of the current update file

SIZE	4		Size of update data to be sent
CRC	4		Data check value of the update package to be sent

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(Chip->PC)
Response number	1	0x03	
Error code	1		Detailed in: error code

5.1.5 Packet

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(PC->chip)
Command number	1	0x04	Update Package Data
Data	<=256-3		Update package data offset based on package header

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(Chip->PC)
Response number	1	0x04	
Error code	1		Detailed in: error code

5.1.6 Postvalidate

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(PC->chip)
Command number	1	0x05	Verify the received data

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(Chip->PC)
Response number	1	0x05	
Error code	1		Detailed in: error code

5.1.7 Activate&Reset

Name	Size (Byte)	Value	Description
Packet header	1	0xAA	(PC->chip)
Command number	1	0x06	Activate new firmware and reset software

Name	Size (Byte)	Value	Description

Packet header	1	0xAA	(Chip->PC)
Response number	1	0x06	
Error code	1		Detailed in: error code

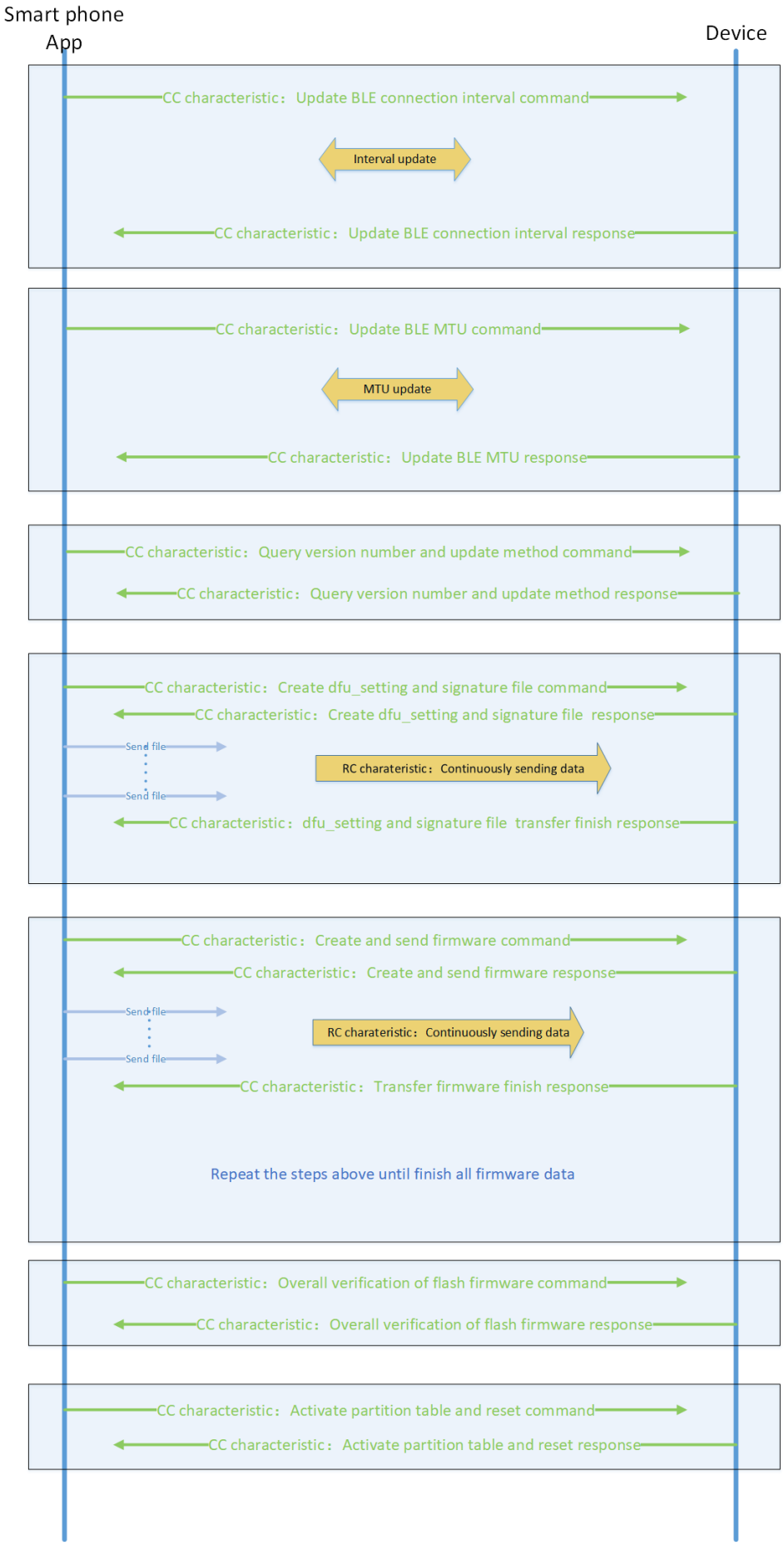
5.2 Commands for Bluetooth update

The mobile device sends commands via the **Command Characteristic (CC)** and also receives responses through the same CC characteristic.
The upgrade package data stream is sent to the device via the **RC characteristic**.

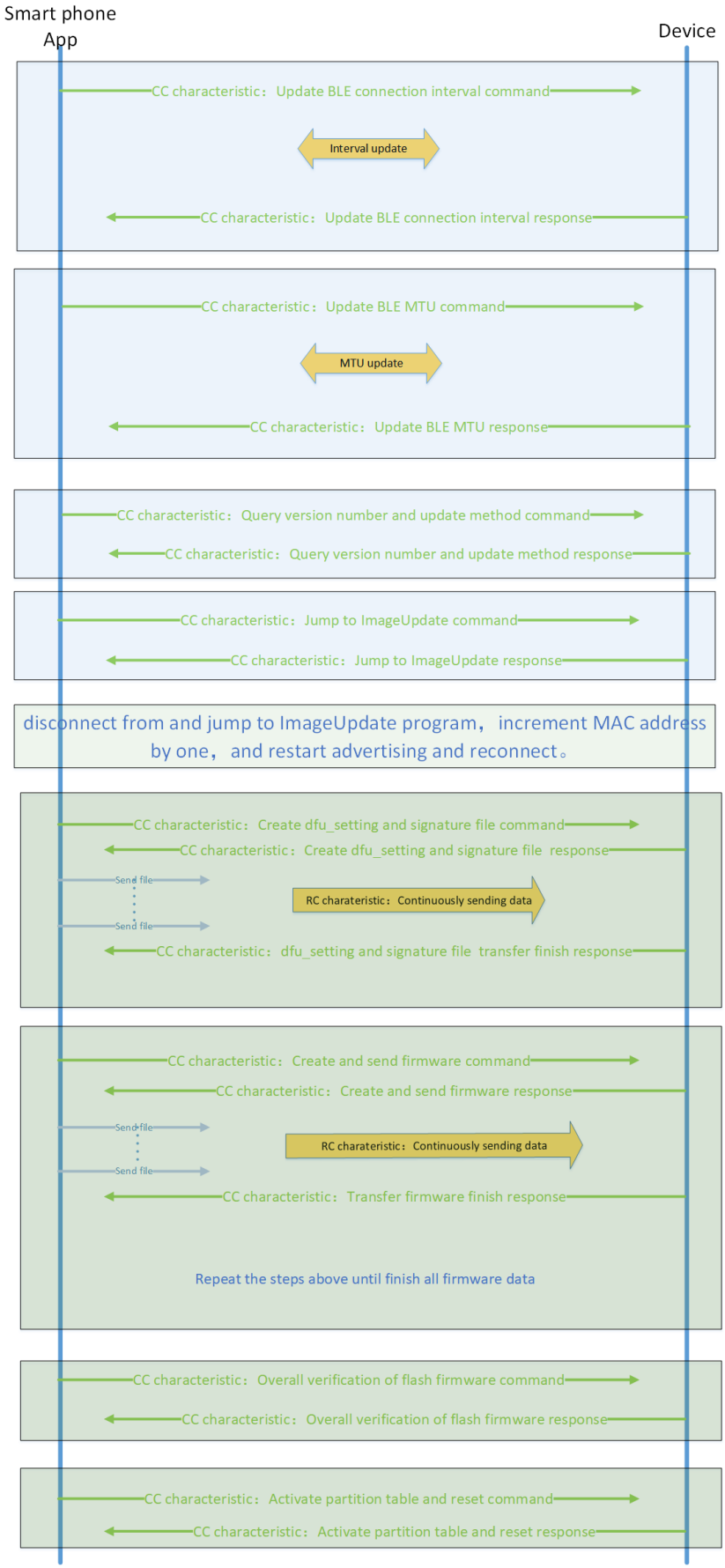
- Service and Characteristic

Type	Name	UUID (Hexadecimal)	Attribute	MTU	Function Description
Service	IUS (Image Update Service)	11-11-11-11-11-11-11-11-11-11-11-11-00-01-11-11	Primary		Firmware update service
Charact eristic	RC (Receive Characteristic)	11-11-11-11-11-11-11-11-11-11-11-11-00-02-11-11	Write Without Response	20-244	Firmware reception characteristics
Charact eristic	CC (Command Characteristic)	11-11-11-11-11-11-11-11-11-11-11-11-00-03-11-11	Notify, Write	20	Command receiving and sending characteristics

- Bluetooth dual bank update


Smart phone
App
Device
CC characteristic：Update BLE connection interval command
Interval update
CC characteristic：Update BLE connection interval response
CC characteristic：Update BLE MTU command
MTU update
CC characteristic：Update BLE MTU response
CC characteristic：Query version number and update method command
CC characteristic：Query version number and update method response
CC characteristic：Create dfu_setting and signature file command
CC characteristic：Create dfu_setting and signature file response
Send file
Send file
RC charateristic：Continuously sending data
CC characteristic：dfu_setting and signature file transfer finish response
CC characteristic：Create and send firmware command
CC characteristic：Create and send firmware response
Send file
Send file
RC charateristic：Continuously sending data
CC characteristic：Transfer firmware finish response
Repeat the steps above until finish all firmware data
CC characteristic：Overall verification of flash firmware command
CC characteristic：Overall verification of flash firmware response
CC characteristic：Activate partition table and reset command
CC characteristic：Activate partition table and reset response

- Single bank update



5.2.1 Update BLE connection interval

- During update, BLE connection interval is reduced and BLE transmission speed is increased. The mobile phone sends the connection interval parameters to the slave device which in turn initiates the command of updating connection interval parameters.
- Reason: it is uncertain whether Android and IOS have enabled the upper layer command of updating connection interval parameters.
- Command format: (CC)

Name	Size (Byte)	Value	Description
Command number	1	1	(Mobile ->Device)
Minimum connection interval	2		Unit: 1.25 MS
Maximum connection interval	2		Unit: 1.25 MS
SLAVE LATENCY	2		Reduced number of responses from slave devices
Connection timed out	2		Unit: 10 MS

- Command format: (CC)

Name	Size (Byte)	Value	Description
Response number	1	1	(Device->Mobile)
Error code	1		Detailed in: error code

5.2.2 Update BLE MTU

- Increase the MTU with RC. The mobile phone sends new MTU size to the slave device which in turn initiates the command of MTU update.
- Reason: it is uncertain whether the IOS has enabled the upper layer command of MTU update.
- Command format: (CC)

Name	Size (Byte)	Value	Description
Command number	1	2	(Mobile->Device)
Size of new MTU	2		Defined by phone model

- Command format: (CC)

Name	Size (Byte)	Value	Description
Response number	1	2	(Device->Mobile)
Error code	1		Detailed in: error code

5.2.3 Query version number and update method

- Command format: (CC)

Name	Size (Byte)	Value	Description
Command number	1	3	(Mobile->Device)
Size of the new APP1 firmware	4		Unit (byte)
Size of the new APP1 firmware	4		Unit (byte)
Size of the new IMAGE UPDATE firmware	4		Unit (byte)
Version of the new IMAGE UPDATE firmware	4		

- Command format: (CC)

Name	Size (Byte)	Value	Description
Response number	1	3	(Device->Mobile)
Version of APP1	4		Partition table Bank 1 Version
Version of APP2	4		Partition table Bank 2 Version
Version of IMAGE UPDATE	4		Partition table IMAGE UPDATE Version
Update mode	1	value: meaning 1: Select APP1 2: Select APP2 3: Select IMAGE UPDATE 4: Jump to IMAGE UPDATE	Device reads partition table, and calculates whether the remaining space can accommodate the new firmware. If yes, select dual bank update, otherwise select single bank update.

5.2.4 Create the dfu_Setting and signature file

- Perform signature verification on the file dfu_setting received at the device side to determine whether the signature file is legal.
- After the update, the device side can update its own local partition table by using the one in dfu_setting.

- Command format: (CC)

Name	Size (Byte)	Value	Description
Command number	1	4	(Mobile->Device)
Size of new DFU SETTING	4		After receiving the data of this size through the RC, the device side notifies the mobile phone of the completed reception through the CC.

- Command of receiving new Bootsetting and signature file
- It is required to, at device side, perform CRC32 integrity verification on Bootsetting, and legitimacy verification on electronic signatures.
- Command format: (CC)

Name	Size (Byte)	Value	Description
Response number	1	4	(Device->Mobile)
Error code	1		Detailed in: error code

5.2.5 Create and sending firmware data

- The mobile terminal notifies the device of the data offset value, data size, and data CRC check value that the RC will receive.
- After receiving RC, the device side notifies mobile phone through CC whether the data is successfully received.
- Command format: (CC)

Name	Size (Byte)	Value	Description
Command number	1	5	(Mobile->Device)
Offset address of firmware data	4		
Transfer size of firmware data	4		Less than or equal to 2048
Verification CRC of firmware data	4		

- Command of completing firmware data reception
- After reception, perform CRC verification on the received data at device side, verify and write in FLASH. If it reaches 4K offset address, it is required to erase 4K for FLASH backward.
- Command format: (CC)

Name	Size (Byte)	Value	Description
Response number	1	5	(Device->Mobile)
Error code	1		Detailed in: error code

5.2.6 Overall verification of FLASH firmware

- Perform CRC verification on the copied firmware at device side, compare it with the firmware CRC in the new partition table, and return the results to the mobile phone.
- Command format: (CC)

Name	Size (Byte)	Value	Description
Command number	1	6	(Mobile->Device)

- Command format: (CC)

Name	Size (Byte)	Value	Description
Response number	1	6	(Device->Mobile)
Error code	1		Detailed in: error code

5.2.7 Activate partition table and reset

- Modify the firmware corresponding to the local partition table to the active state at device side, respond to the command of mobile phone, and then execute software reset.
- Command format: (CC)

Name	Size (Byte)	Value	Description
Command number	1	7	(Mobile->Device)

- Command format: (CC)

Name	Size (Byte)	Value	Description
Response number	1	7	(Device->Mobile)
Error code	1		Detailed in: error code

5.2.8 Jump to ImageUpdate

- Command format: (CC)

Name	Size (Byte)	Value	Description
Command number	1	8	(Mobile->Device)

- Command format: (CC)

Name	Size (Byte)	Value	Description
Response number	1	8	(Device->Mobile)
Error code	1		Detailed in: error code

5.3 Error code

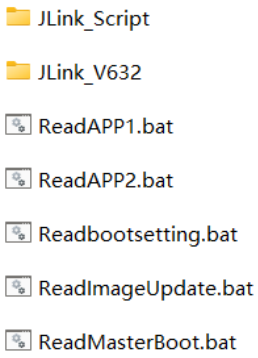
Value	Description

0	Success
1	Parameter error
2	CRC error
3	Electronic signature error

6 Tool Explanation

6.1 JLINK tool

Enter CMT453x_SDK\utilities\dfu\JLink\ directory



JLink_V632 folder contains a JLINK tool manufactured by SEGGER. Users need to install a JLINK driver before use.

JLink_Script folder contains a JLink script file. Users can read the data in the chip's Flash by double-clicking the file *Read XX.bat*.

JLink also supports chip FLASH erasure and programming, and the file *JLINK Download at one click.bat* used in the previous chapter is exactly used to implement such functions.

6.2 HPUTIL tool

Enter CMT453x_SDK\utilities\dfu\HPUtil\ directory

- HPUtil.exe Windows execution program

The functions implemented by this tool include:

- Generate bootsetting.bin file
- Serial port update via PC upper computer
- Packaging tool for Bluetooth update package
- Generate ECC key, digital signature tool

6.3 HPAndroidUtil tool

Enter CMT453x_SDK\utilities\dfu\HPAndroidUtil\ directory

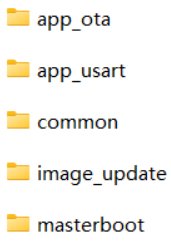
- AndroidUtil.apk Android installation package.

This tool implements the following functions:

- Bluetooth OTA update

7 Examples Explanation

Enter CMT453x_SDK_v1.X.X\projects\cmt453x_EVAL\ble_dfu directory



7.1 MasterBoot Explanation

MasterBoot is the first program executed after the system powers on. It determines whether to jump to **Bank1**, **Bank2**, or the **ImageUpdate** region to run the corresponding program by reading the information in the **Bootsetting** area, or it stays in **MasterBoot** to receive serial commands via UART and perform the upgrade process.
A code snippet of this program is as follows:

```
int main(void)
{
    masterboot();

    dfu_leds_config();
    dfu_led_on(LED1_GPIO_PORT, LED_GPIO1_PIN);
    dfu_led_on(LED2_GPIO_PORT, LED_GPIO2_PIN);
    HP_SCHED_INIT(256, 16);
    hp_dfu_serial_init();

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

```
static void masterboot(void)
{
    if(hp_bootsetting.crc == dfu_crc32((uint8_t *)&hp_bootsetting.crc + 4, sizeof(HP_Bootsetting_t) -
4))
    {
        if(hp_bootsetting.master_boot_force_update != HP_BOOTSETTING_MASTER_BOOT_FORCE_UPDATE_YES)
```

```
        {
            if(hp_bootsetting.app1.activation == HP_BOOTSETTING_ACTIVATION_YES &&
hp_bootsetting.app1.start_address == HP_APP1_START_ADDRESS)
            {
                if(hp_bootsetting.app1.crc == dfu_crc32((uint8_t *)((uint32_t
*)hp_bootsetting.app1.start_address), hp_bootsetting.app1.size))
                {
                    hp_dfu_boot_jump(hp_bootsetting.app1.start_address);
                }
            }
            //……
        }
    }
}
```

- Masterboot (): Read the bootsetting partition table and directly jump to the activated firmware with complete verification.
- Illuminate two lights for indication.
- Initialize simple scheduling.
- Initialize the serial port.
- Wait for the serial port to interrupt receiving data.

If no active application is detected in masterboot(), **MasterBoot** will wait for commands sent from the serial port. After receiving serial data, it will parse and process the commands and respond to the host accordingly.

```
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] == DFU_SERIAL_HEADER)
            {
                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;
  }
}
```

7.2 AppUsart Explanation

- UART DFU APP 1 program

```
int main(void)
{
    PWR->VTOR_REG = CURRENT_APP_START_ADDRESS | 0x80000000;

    dfu_leds_config();
    if(CURRENT_APP_START_ADDRESS == HP_APP1_START_ADDRESS){
        dfu_led_on(LED1_GPIO_PORT, LED_GPIO1_PIN);
    }else if(CURRENT_APP_START_ADDRESS == HP_APP2_START_ADDRESS){
        dfu_led_on(LED2_GPIO_PORT, LED_GPIO2_PIN);
    }

    HP_SCHED_INIT(256, 16);

    Qflash_Init();
    dfu_usart1_interrupt_config();
    dfu_usart1_enable();
    while(1)
    {
        app_sched_execute();
        __WFE();
        __SEV();
```

```
        __WFI();
    }
}
```

- Judge whether it is currently in bank 1 or bank 2. LED1 and LED2 will be ON respectively when it is in Bank 1 or bank 2
- Initialize the serial port.
- Wait for the UART interrupt for receiving data.
- Handle the serial port command, respond to the PC upper computer, write in forced serial port update flag, and the service jumps to the MasterBoot program.

```
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(hp_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;
    }
}
```

7.3 AppOTA Explanation

First, determine whether the current program is handling Bank1 or Bank2, and provide a corresponding LED indication.

```
int main(void)
{

    HP_LOG_INIT();

    if(CURRENT_APP_START_ADDRESS == HP_APP1_START_ADDRESS){
        HP_LOG_INFO("application 1 start new ...\r\n");
    }else if(CURRENT_APP_START_ADDRESS == HP_APP2_START_ADDRESS){
        HP_LOG_INFO("application 2 start new ...\r\n");
    }

    dfu_leds_config();

    for(uint8_t i=0;i<10;i++)
    {
        dfu_led_toggle(LED1_GPIO_PORT, LED_GPIO1_PIN);
        if(CURRENT_APP_START_ADDRESS == HP_APP1_START_ADDRESS){
            dfu_delay_ms(100);
        }else if(CURRENT_APP_START_ADDRESS == HP_APP2_START_ADDRESS){
            dfu_delay_ms(500);
        }
        dfu_led_toggle(LED2_GPIO_PORT, LED_GPIO2_PIN);
    }
    dfu_led_off(LED1_GPIO_PORT, LED_GPIO1_PIN);
    dfu_led_off(LED2_GPIO_PORT, LED_GPIO2_PIN);

    app_ble_init();

    while(1)
    {
        rwip_schedule();
        hp_sleep();
    }
}
```

Initialize the BLE stack:

- Configure MAC address of Bluetooth.
- Configure the name of Bluetooth broadcast.

- Add IUS service.

```
void app_ble_init(void)
{
    struct hp_stack_cfg_t app_handler = {0};
    app_handler.ble_msg_handler  = app_ble_msg_handler;
    app_handler.user_msg_handler = app_user_msg_handler;
    app_handler.lsc_cfg          = BLE_LSC_LSI_32000HZ;
    //initialization ble stack
    hp_ble_stack_init(&app_handler);

    app_ble_gap_params_init();
    app_ble_sec_init();
    app_ble_adv_init();
    app_ble_prf_init();
    //start adv
    hp_ble_adv_start();
}
```

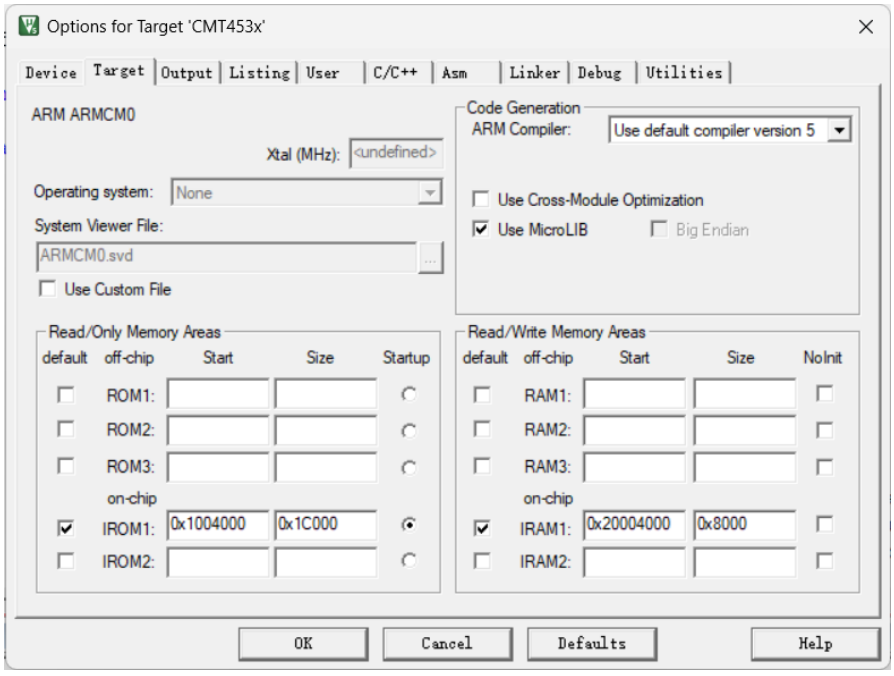
- Handle Command Characteristic (CC) commands

```
void hp_dfu_ble_handler_cc(uint8_t const *input, uint8_t input_len, uint8_t *output, uint8_t
*output_len)
{

    switch(input[0]){

        case OTA_CMD_CONN_PARAM_UPDATE:{
            struct gapc_conn_param conn_param;
            conn_param.intv_min = input[1]<<8 | input[2];
            conn_param.intv_max = input[3]<<8 | input[4];
            conn_param.latency = input[5]<<8 | input[6];
            conn_param.time_out = input[7]<<8 | input[8];
            app_env.manual_conn_param_update = 1;
            hp_ble_update_param(&conn_param);
            *output_len = 0;
        }break;
        case OTA_CMD_MTU_UPDATE:{
            app_env.manual_mtu_update = 1;
            hp_ble_mtu_set(input[1]<<8 | input[2]);
            *output_len = 0;
        }break;
        // ******
    }
}
```

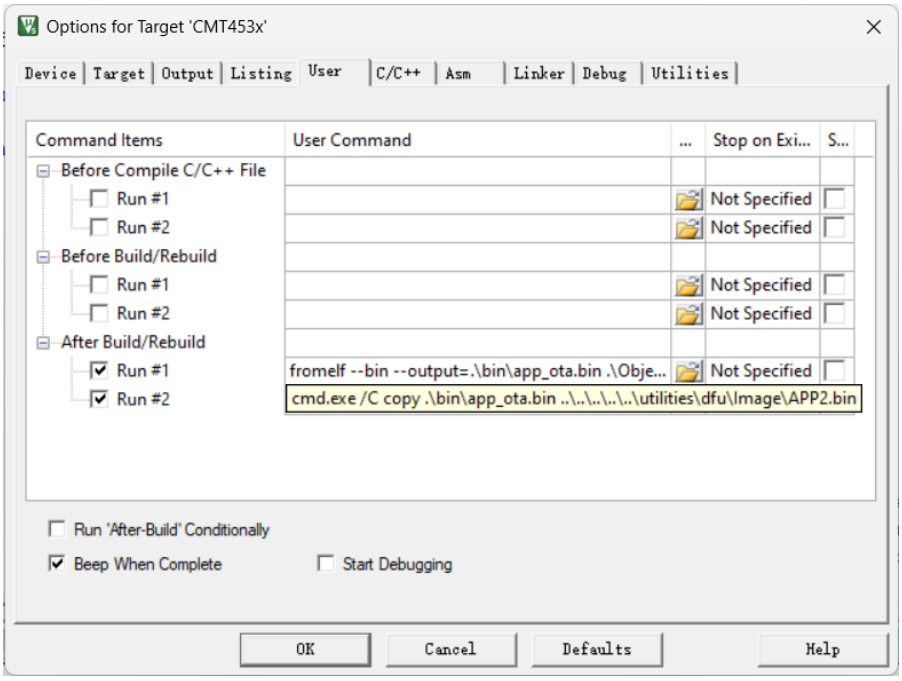
- Handle Receive Characteristic (RC) commands and data

```
void hp_dfu_ble_handler_rc(uint8_t const *input, uint32_t input_len)
{
    switch(m_rc_state){

        case OTA_RC_STATE_DFU_SETTING:{
            memcpy(m_buffer + rc_mtu_offset, input , input_len);
            rc_mtu_offset += input_len;
            if(rc_mtu_offset >= m_ota_setting_size){
                rc_mtu_offset = 0;
                m_rc_state = OTA_RC_STATE_NONE;
                memcpy(&m_dfu_setting, m_buffer, sizeof(Dfu_setting_t));
                uint32_t crc = dfu_crc32((uint8_t *)&m_dfu_setting.crc + 4, sizeof(Dfu_setting_t) - 4);
                if(crc == m_dfu_setting.crc){
                    uint8_t error = 0;
        // ******
    }
}
```

- To create the upgrade package, the user needs to modify IROM1 in Keil's Options for Target.
- If using DualBank mode for OTA, two upgrade firmware files are required: APP1.bin and APP2.bin.
- The start addresses of these programs are 0x0100_4000 and 0x0102_0000, corresponding to the Bank1 and Bank2 regions of the flash, respectively.
- The maximum size for both is set to 0x1C000.



After compilation, Keil will automatically copy the generated firmware to the .\utilities\dfu\Image directory and name it as APPx.bin.
When compiling the APP1 and APP2 projects, please modify the target firmware name accordingly.



7.4 ImageUpdate Explanation

ImageUpdate is used in the single-bank upgrade process. It is a minimal Bluetooth application designed to establish a connection with the mobile phone, receive commands and firmware updates from the phone, and write the received firmware directly into the address space of the existing application.

The program first calls bootsetting_reset() to set the bootsetting to default values. For example, if firmware exists in the Bank1 address region, bootsetting_reset() will set the Bank1 activation flag to 1.

This ensures that on the next reboot, the system will automatically jump from MasterBoot to Bank1 for execution.

```
int main(void)
{
    bootsetting_reset();

    app_init();
    prf_init(RWIP_INIT);

    while(1)
    {
        rwip_schedule();
```

```
    }
}
```

Use the Image Update server profile

```
/* Device name */
#define CUSTOM_DEVICE_NAME                  "ImageUpdate"
#define CUSTOM_BLE_MAC_ADDRESS              "\x8A\x22\x77\x44\x55\x66"

//Enable Image Update Server
#define CFG_APP_HP_IUS     1
```

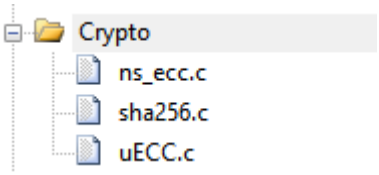
- For handling Bluetooth upgrade commands and receiving the upgrade firmware, please refer to the AppOTA explanation process.

8 Explanation of Encryption

Enable the macro OTA_ECC_ECDSA_SHA256_ENABLE in AppOTA and ImageUpdate program, so as to enables update and signature verification.

```
/* Private define ------------------------------------------------------------*/
#define OTA_ECC_ECDSA_SHA256_ENABLE                1

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

- The project includes these files to realize the verification interface.



- When receiving the dfu_setting data, the embedded side uses the below method for signature verification.

```
#if OTA_ECC_ECDSA_SHA256_ENABLE
uint8_t raw_data[sizeof(Dfu_setting_bank_t)*3];
memcpy(raw_data,&m_dfu_setting.app1,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 == hp_lib_ecc_hash_sha256(raw_data, sizeof(Dfu_setting_bank_t)*3, hash_digest)){
    if(ERROR_SUCCESS != hp_lib_ecc_ecdsa_verify(hp_bootsetting.public_key, hash_digest, 32,
m_dfu_setting.signature)){
        error = 3;
    }
}else{
    error = 3;
}
#endif
```

- Generate an update package, use ECC to sign and encrypt the firmware parameters, including CRC, size, and others, and save them in dfu_setting. After receiving the dfu_setting, the embedded side uses known public key to verify the signature. You can indeed start the update (that is, erasing and writing FLASH) when the signature verification is successful.
- Only the unique information of the firmware is encrypted and signed, so that the signature verification at the embedded side is accelerated and the firmware update is well protected.

9 Version History

Date	Version	Modification
2021.07.29	V1.0	Initial Version
2021.12.22	V1.1	Updated the app_ota project code screenshots based on the latest SDK. Updated the description for common issue handling.
2023.06.06	V1.2	Revised the document structure.

10 Notice

Liability Disclaimer

Shenzhen Hope Microelectronics Co., Ltd reserves the right to make changes without further notice to the product to improve reliability, function or design. Shenzhen Hope Microelectronics Co., Ltd does not assume any liability arising out of the application or use of any product or circuits described herein.

Life Support Applications

Shenzhen Hope Microelectronics Co., Ltd's products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Shenzhen Hope Microelectronics Co., Ltd customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Shenzhen Hope Microelectronics Co., Ltd for any damages resulting from such improper use or sale.

Contact Information

Shenzhen Hope Microelectronics Co., Ltd.

Address: 30th floor of 8th Building, C Zone, Vanke Cloud City, Xili Sub-district, Nanshan, Shenzhen, GD, P.R. China

Tel: +86-755-82973805

Post Code: 518055

Email: sales@hoperf.com

Website: www.hoperf.com