GR551x Fast DFU Example Application

Version: 1.8

Release Date: 2022-02-20

Shenzhen Goodix Technology Co., Ltd.

Trademarks and Permissions

GOODiX and other Goodix trademarks are trademarks of Shenzhen Goodix Technology Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

Disclaimer

Information contained in this document is intended for your convenience only and is subject to change without prior notice. It is your responsibility to ensure its application complies with technical specifications.

Shenzhen Goodix Technology Co., Ltd. (hereafter referred to as “Goodix”) makes no representation or guarantee for this information, express or implied, oral or written, statutory or otherwise, including but not limited to representation or guarantee for its application, quality, performance, merchantability or fitness for a particular purpose. Goodix shall assume no responsibility for this information and relevant consequences arising out of the use of such information.

Without written consent of Goodix, it is prohibited to use Goodix products as critical components in any life support system. Under the protection of Goodix intellectual property rights, no license may be transferred implicitly or by any other means.

Shenzhen Goodix Technology Co., Ltd.

Headquarters: 2F. & 13F., Tower B, Tengfei Industrial Building, Futian Free Trade Zone, Shenzhen, China

TEL: +86-755-33338828 FAX: +86-755-33338099

Website: www.goodix.com

Preface

Purpose

This document introduces how to quickly implement Fast DFU on GR551x System-on-Chips (SoCs) with the debugging App GRToolbox (Android) by using Bluetooth Low Energy (Bluetooth LE) technology, to help users have better understanding of Fast DFU for GR551x SoCs.

Audience

This document is intended for:

- GR551x user
- GR551x developer
- GR551x tester
- Technical writer

Release Notes

This document is the fourth release of *GR551x Fast DFU Example Application*, corresponding to GR551x SoC series.

Revision History

Version	Date	Description
1.5	2020-08-30	Initial release
1.6	2020-12-15	Updated GRToolbox UI figures based on software update.
1.7	2021-08-09	Changed the section “Supported Development Platform” into “Preparation”.
1.8	2022-02-20	• Added the sections "Fast DFU Service and Characteristics" and "Fast DFU Communications Protocol". • Modified the file name of the example firmware based on Software Development Kit (SDK) changes.

Contents

1 Introduction

The Fast Device Firmware Update (Fast DFU) enables downloading the target firmware to a target device and enables the device to automatically check and update the firmware through Bluetooth transmission.

Compared with the over-the-air (OTA) technology which also enables downloading and updating the target firmware through Bluetooth transmission, GR551x Fast DFU greatly shortens the time for firmware update by providing Fast OTA Service which enables control, interaction, and various features of Fast DFU, as well as optimizing data read and write.

As a solution for wireless firmware update, Fast DFU features faster update process and better user experience. This document elaborates on principles and applications of Fast DFU for GR551x System-on-Chips (SoCs).

Before getting started, you can refer to the following documents.

Table 1-1 Reference documents

Name	Description
GR551x Developer Guide	Introduces GR551x Software Development Kit (SDK) and how to develop and debug applications based on the SDK.
Bluetooth Core Spec	Offers official Bluetooth standards and core specification from Bluetooth SIG.
J-Link/J-Trace User Guide	Provides J-Link operational instructions. Available at www.segger.com/downloads/jlink/UM08001_JLink.pdf.
Keil User Guide	Offers detailed Keil operational instructions. Available at www.keil.com/support/man/docs/uv4/.
GR551x Bluetooth Low Energy Stack User Guide	Introduces the BLE Protocol Stack supported by GR551x SoCs.

2 Fast DFU Overview

This chapter introduces fundamental concepts about GR551x Fast DFU.

2.1 Device Roles

Two Fast DFU device roles are defined:

- Control device (the host): a device, such as a mobile phone, that sends update data to the target device
- Target device (the device): a device, such as a wristband, that receives update data from the control device

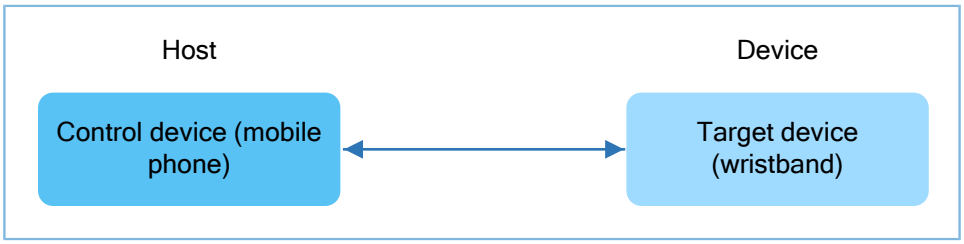

Figure 2-1 Device role

2.2 Interactions

To speed up data transmission, only firmware information/data/checksum interactions are supported between the host and the device.

The interaction procedures between the host and the device are shown in Figure 2-2.

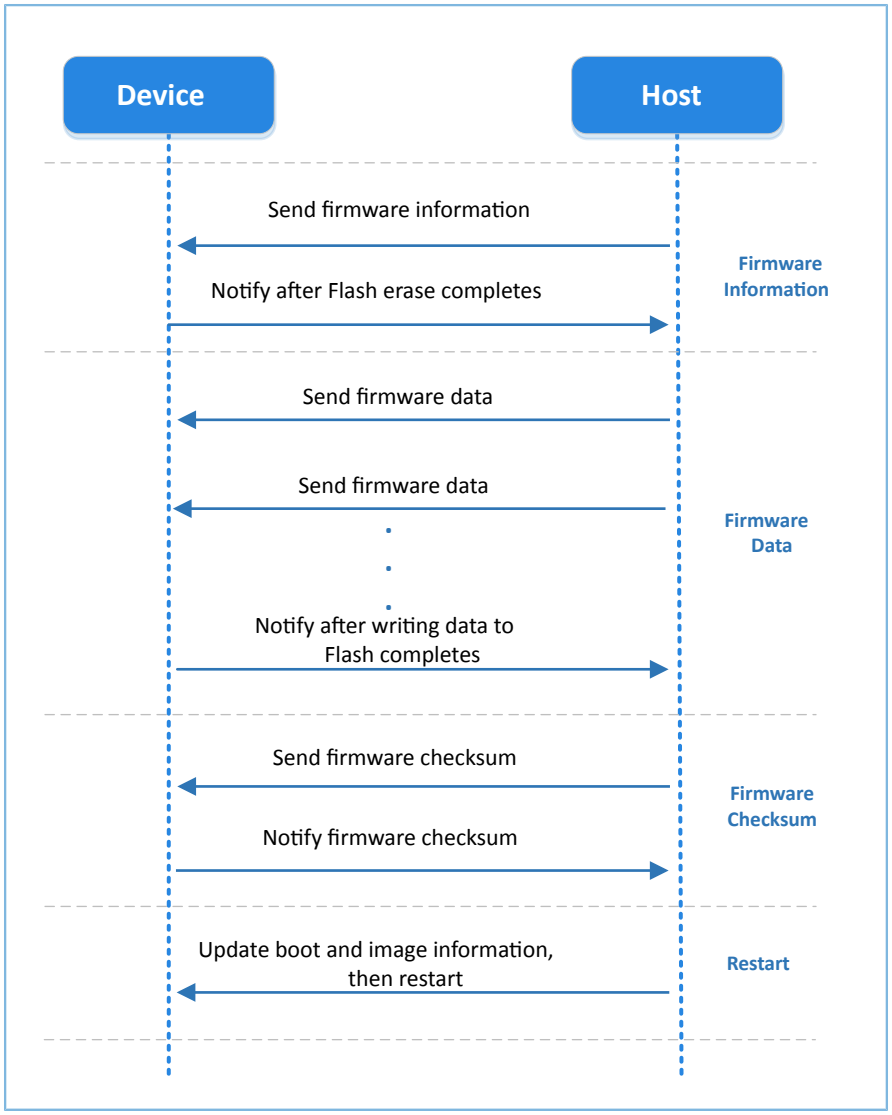

Figure 2-2 Fast DFU interaction procedures

1. Firmware information interaction: The host sends the target firmware information to the device; after receiving the information, the device erases the flash memory to be used for firmware update for one time and then notifies the host.
2. Firmware data interaction: The host sends the target firmware data to the device continuously; then the device caches the received data in a local ring buffer. When the data in the ring buffer reaches the set length, the device writes the data for one time to the flash. After writing all the firmware data to the flash, the device notifies the host.
3. Firmware checksum interaction: The host and the device send the calculated firmware checksums to each other for comparison.

If the checksums from the host and the device are consistent, the boot and image information will be updated and the device will restart.

2.3 Fast DFU Service and Characteristics

The UUID of Goodix Fast DFU Service is **a6ed0701-d344-460a-8075-b9e8ec90d71b**. Fast DFU characteristics are divided into 2 types, as listed in the table below.

Table 2-1 Fast DFU characteristics

Description	UUID	Property
Fast DFU Cmd Characteristic	a6ed0702-d344-460a-8075-b9e8ec90d71b	Write Notify without Response
Fast DFU Data Characteristic	a6ed0703-d344-460a-8075-b9e8ec90d71b	Write without Response

The role of each characteristic:

- Fast DFU Cmd Characteristic: Receive control commands from the host, such as a command to erase the flash for a device.
- Fast DFU Data Characteristic: Receive data from the host with a property of **Write without Response** to speed up data transmission.

2.4 Fast DFU Communications Protocol

The Fast DFU between the host and the device is based on Fast DFU communications protocols.

2.4.1 Data Frame Format

This section introduces the format of data frames sent by the host or responded by the device for Fast DFU based on Fast DFU communications protocols.

2.4.1.1 Data Frame Sent by the Host

Table 2-2 Structure of command frame sent by the host

Frame Header	Operating Command	Data
4 bytes	1 byte	n byte(s)

- Frame header: the start of a frame; fixed to 0x474f4f44 for Fast DFU example
- Operating command: operating command code delivered by the host
- Data: delivered by the host according to operating commands. For details, refer to the “Data Sent from the Host” sections in “Section 2.4.2 Fast DFU Command Set”.

2.4.1.2 Data Frame Responded by the Device

Table 2-3 Structure of command frame responded by the device

Operating Command	Data
1 byte	n byte(s)

- Operating command: operating command code responded by the device
- Data: response from the device to operating commands delivered by the host. For details, refer to the "Response Data from the Device" sections in "Section 2.4.2 Fast DFU Command Set".

2.4.2 Fast DFU Command Set

Fast DFU commands are delivered by the host and received by the device, including commands for flash erasing, firmware writing done, and checksum obtaining.

2.4.2.1 Flash Erasing Command

2.4.2.1.1 Data Sent from the Host

Table 2-4 Format of sent data for the flash erasing command

Byte No.	Description	Valid Value	Remarks
0–3	Frame header	0x474f4f44	Fixed value
4	Command	0x01	Flash erasing command
5–8	Flash erasing address	4 bytes	Start address of the Flash area to be erased
9–12	Flash erasing space	4 bytes	Size of the Flash area to be erased

2.4.2.1.2 Response Data from the Device

Table 2-5 Format of response data for the flash erasing command

Byte No.	Description	Valid Value	Remarks
0	Command	0x01	Respond to the command after flash erasing completes.
1	Erasing status	• 0x00 • 0x01 • 0x02 • 0x03 • 0x04 • 0x05 • 0x06	• 0x00: Start address of the Flash area to be erased is not 4 KB aligned. • 0x01: Start erasing. • 0x02: in normal erasing operation • 0x03: Erasing completes and data can be delivered. • 0x04: The flash area to be erased overlaps the current running area. • 0x05: Erasing fails. • 0x06: invalid flash area to be erased
2–3	Number of erased pages	2 bytes	Number of erased pages

2.4.2.2 Firmware Writing Done Command

2.4.2.2.1 Data Sent from the Host

Table 2-6 Format of sent data for the firmware writing done command

Byte No.	Description	Valid Value	Remarks
0–3	Frame header	0x474f4f44	Fixed value
4	Command	0x02	Firmware writing done command

2.4.2.2.2 Response Data from the Device

Table 2-7 Format of response data for the firmware writing done command

Byte No.	Description	Valid Value	Remarks
0	Command	0x02	Respond to the command after firmware writing completes.

2.4.2.3 Checksum Obtaining Command

2.4.2.3.1 Data Sent from the Host

Table 2-8 Format of sent data for the checksum obtaining command

Byte No.	Description	Valid Value	Remarks
0–3	Frame header	0x474f4f44	Fixed value
4	Command	0x03	Checksum obtaining command
5–8	Checksum	4 bytes	Firmware checksum

2.4.2.3.2 Response Data from the Device

Table 2-9 Format of response data for the checksum obtaining command

Byte No.	Description	Valid Value	Remarks
0	Command	0x03	Respond to the command after obtaining checksum.
1–4	Checksum	4 bytes	Firmware checksum obtained by the device

2.4.2.4 Boot Information Setting Command

2.4.2.4.1 Data Sent from the Host

Table 2-10 Format of sent data for the boot information setting command

Byte No.	Description	Valid Value	Remarks
0–3	Frame header	0x474f4f44	Fixed value

Byte No.	Description	Valid Value	Remarks
4	Command	0x04	Boot information setting command
5–28/ 5–44	Boot information	24 bytes (Boot Info)/ 40 bytes (Boot and Image Info)	• Boot Info: Update the boot information only. • Boot and Image Info: Update both the boot and image information.

2.4.2.4.2 Response Data from the Device

After boot and image information update completes, the device will restart without response.

2.4.2.5 DFU Flash Type Setting Command

2.4.2.5.1 Data Sent from the Host

Table 2-11 Format of sent data for the DFU flash type setting command

Byte No.	Description	Valid Value	Remarks
0–3	Frame header	0x474f4f44	Fixed value
4	Command	0x05	DFU flash type (internal/external) setting command
5	Flash type	0x00/0x01	• 0x00: internal flash • 0x01: external flash

2.4.2.5.2 Response Data from the Device

Table 2-12 Format of response data for the DFU flash type setting command

Byte No.	Description	Valid Value	Remarks
0	Command	0x05	Respond to the command after flash type is set.

2.4.2.6 Copying Firmware for Fast DFU Command

2.4.2.6.1 Data Sent from the Host

Table 2-13 Format of sent data for the copying firmware for Fast DFU command

Byte No.	Description	Valid Value	Remarks
0–3	Frame header	0x474f4f44	Fixed value
4	Command	0x08	Boot information setting command
5–44	Boot information	40 bytes (Boot and Image Info)	Update both the boot and image information.

2.4.2.6.2 Response Data from the Device

After boot and image information update completes, the device will restart without response.

2.4.2.7 DFU Version Obtaining Command

2.4.2.7.1 Data Sent from the Host

Table 2-14 Format of sent data for the DFU version obtaining command

Byte No.	Description	Valid Value	Remarks
0–3	Frame header	0x474f4f44	Fixed value
4	Command	0x0b	DFU version obtaining command

2.4.2.7.2 Response Data from the Device

Table 2-15 Format of response data for the DFU version obtaining command

Byte No.	Description	Valid Value	Remarks
0	Command	0x0b	Respond to the command after obtaining DFU version.
1	Version	1 byte	DFU version obtained by the device

3 Enablement of Fast DFU

This chapter introduces how to enable Fast DFU for GR551x SoCs.

The Fast DFU functionality is encapsulated in the Fast DFU module (available in SDK_Folder\components\libraries\fast_dfu). To use Fast DFU in applications, you only need to call related APIs. Details are provided below by taking ble_dfu_fast as an example.

Note:

SDK_Folder is the root directory of GR551x SDK.

3.1 ble_dfu_fast Project

The ble_dfu_fast example is in SDK_Folder\projects\ble\dfu\ble_dfu_fast, and the project file is in the Keil_5 folder.

Double-click the project file, *ble_dfu_fast.uvprojx*, to view the project directory structure of ble_dfu_fast in Keil. For related files, see Table 3-1.

Table 3-1 File description of ble_dfu_fast

Group	File	Description
gr_libraries	fast_dfu.c	Initializes Fast DFU Service, including protocol control and process scheduling for Fast DFU.
gr_profiles	ble_prf_utils.c	Contains profile-related operational tools.
	fast_otas.c	Implements Fast OTA Service.
user_callback	user_gap_callback.c	Implements GAP callbacks, such as connection, disconnection, and GAP parameter update.
	user_gatt_common_callback.c	Implements GATT common callbacks, such as MTU exchange.
user_platform	user_periph_setup.c	Configures the serial port parameters, device address, power management mode, and Fast DFU.
user_app	main.c	Contains the main() function.
	user_app.c	Implements Fast OTA profile registration and logical processing.

3.2 Steps

Follow the steps below to apply Fast DFU in applications.

1. Initialize the Fast DFU functionality.

   Path: user_platform\user_periph_setup.c under the project directory

   Name: app_periph_init();

   Call fast_dfu_init() in this function to initialize Fast DFU.

```
void app_periph_init(void)
{
```

```
    SET_BD_ADDR(BD_ADDR_NVDS_TAG, BD_ADDR_LENGTH, s_bd_addr);
    bsp_uart_init();
    qspi_flash_init();
    fast_dfu_init(&qspi_flash_api, &dfu_pro_call);
    pwr_mgmt_mode_set(PMR_MGMT_ACTIVE_MODE);
    APP_LOG_DEBUG("DFU FAST DEMO START");
}
```

Table 3-2 Input parameter description for fast_dfu_init()

Parameter	Description	Value
p_dfu_func	Whether to program the external flash. To program the external flash, it is required to input the corresponding interface pointer for the external flash operation (read/write/erase).	• Yes: The external flash operation API implemented by users is assigned to this parameter. • No: **NULL** is assigned to this parameter.
p_state_callback	Enable/Disable monitoring on update status at the application layer.	• Yes: The Fast DFU status handling callback function implemented by users is assigned to this parameter. • No: **NULL** is assigned to this parameter.

Note:

- In the example code, the external Flash API **qspi_flash_api** is assigned to the input parameter **p_dfu_func** of fast_dfu_init(), and the Fast DFU status handling callback **dfu_pro_call** is assigned to **p_state_callback**.
- To use the external flash, ONCE_WRITE_DATA_LEN in *fast_dfu.h* shall be less than 4096 bytes.

2. Initialize the Fast DFU service.

   Path: `user_app\user_app.c` under the project directory

   Name: services_init();

   Call fast_dfu_service_init() in this function to register the Fast DFU service.

```
static void services_init(void)
{
    fast_dfu_service_init();
}
```

3. Call fast_dfu_schedule() in the while(1) { } loop of the main() function to schedule the Fast DFU functionality.

   Path: `user_app\main.c` under the project directory

   Name: main();

```
int main(void)
{
    // Initialize user peripherals.
    app_periph_init();

    // Initialize BLE Stack.
    ble_stack_init(&s_app_ble_callback, &heaps_table);
```

```
    // loop
    While (1)
    {
        fast_dfu_schedule();
        pwr_mgmt_schedule();
    }
}
```

4 Test and Verification

The GR551x SDK provides a Fast DFU example. This chapter introduces how to test and verify the Fast DFU functionality by using a GR5515 Starter Kit Board (SK Board) and an Android phone.

4.1 Preparation

Perform the following tasks before the test.

- **Hardware preparation**

Table 4-1 Hardware preparation

Name	Description
Development board	GR5515 Starter Kit Board (SK Board)
Android phone	A mobile phone running on Android 5.0 (KitKat) or later
Connection cable	Micro USB 2.0 cable

- **Software preparation**

Table 4-2 Software preparation

Name	Description
Windows	Windows 7 or later
J-Link driver	A J-Link driver. Available at www.segger.com/downloads/jlink/ .
Keil MDK5	An integrated development environment (IDE). MDK-ARM Version 5.20 or later is required. Available at www.keil.com/download/product/.
GRToolbox (Android)	A Bluetooth LE debugging tool. Available in SDK_Folder\tools\GRToolbox.
GProgrammer (Windows)	A programming tool. Available in SDK_Folder\tools\GProgrammer.

4.2 Downloading Fast DFU Firmware to SK Board

The GR551x SDK contains the Fast DFU example and precompiled firmware. To perform the Fast DFU test, first download the Fast DFU firmware to the SK Board, and then connect the Android phone with the SK Board through GRToolbox to fast update the target firmware.

Download the Fast DFU firmware *ble_dfu_fast.bin* to the SK Board. For details, see *GProgrammer User Manual*.

Note:

- The load address and run address of *ble_dfu_fast.bin* are 0x01002000.
- *ble_dfu_fast.bin* is in SDK_Folder\projects\ble\dfu\ble_dfu_fast\build\.

Turn on Bluetooth on the mobile phone and run GRToolbox. If a device named as **Goodix_Fast_DFU** is discovered, the firmware demo runs normally.

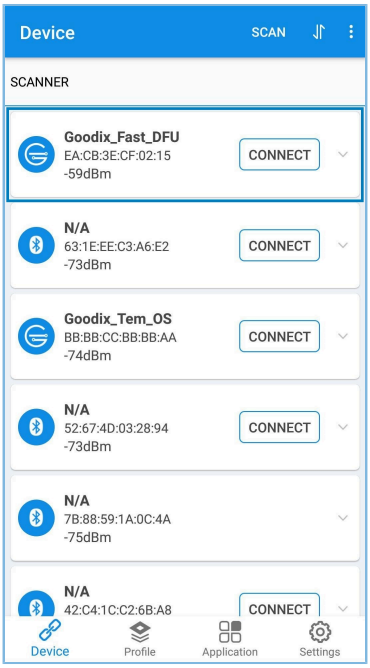

Figure 4-1 Discovering **Goodix_Fast_DFU**

Note:

Screenshots of GRToolbox in this document are for reference only, to help users better understand the software operation. In the case of interface differences due to version changes, the interface of GRToolbox in practice shall prevail.

Goodix_Fast_DFU refers to the device advertising name defined in *ble_dfu_fast.bin*.

4.3 Fast DFU Through GRToolbox on the Mobile Phone

This section introduces how to perform Fast DFU by using the target firmware *ble_tem_dfu.bin* through GRToolbox on the mobile phone.

Note:

- The load address and run address of *ble_tem_dfu.bin* are 0x01040000, which shall not conflict with those of *ble_dfu_fast.bin*.
- The *ble_tem_dfu.bin* is in: SDK_`Folder\projects\ble\ble_peripheral\ble_app_template_dfu\build\`.

---

1. Copy the target firmware to the mobile phone.

   Copy *ble_tem_dfu.bin* to `\Goodix\SaveData` in the root directory of the mobile phone.

2. Load the target firmware.

   (1). Run GRToolbox (Android). Tap **Application** and then select **Fast DFU** to enter the Fast DFU interface.

(2). Tap **SELECT** in the **Update** area and then load *ble_tem_dfu.bin* which is copied in Step 1. When the file is loaded to GRToolbox successfully, information of the *ble_tem_dfu.bin* file for update is displayed, as shown in Figure 4-4.

When the load address and the run address of the target firmware to be updated are the same as those of *ble_dfu_fast.bin*, users need to select **Copy Mode** and set **Copy Address** for the target firmware and **Start Address** of the flash to cache the target firmware, as shown in Figure 4-5.

Note:

- The **Start Address** of the flash shall be within the unoccupied flash space in the SoC and the flash space shall be larger than the target firmware size.
- Only SDK V1.6.06 or later supports updating the target firmware with the load address and run address same as those of *ble_dfu_fast.bin* by selecting **Copy Mode** in GRToolbox.

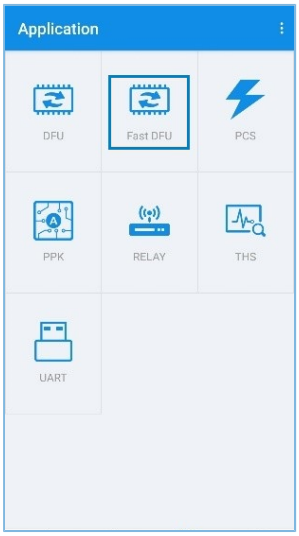

Figure 4-2 Selecting **Fast DFU**

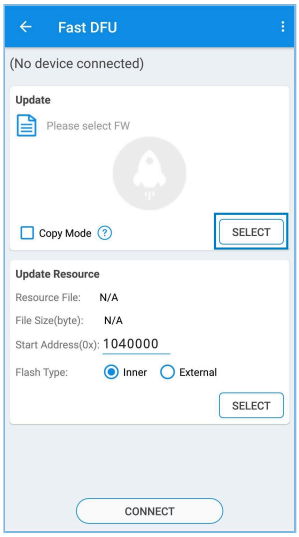

Figure 4-3 Selecting the target firmware

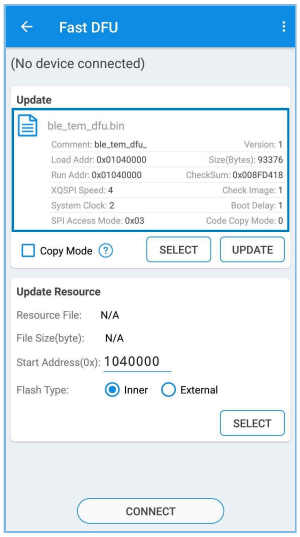

Figure 4-4 Viewing firmware information

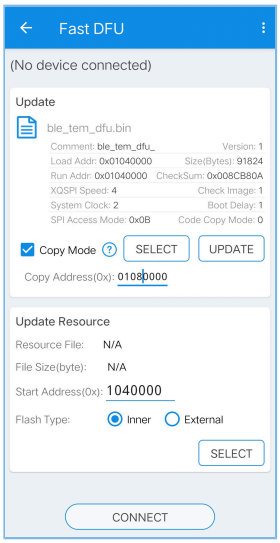

Figure 4-5 Selecting **Copy Mode**

3. Connect the mobile phone to the SK Board through Bluetooth.

Note:

Turn on Bluetooth on the mobile phone before connection.

Tap **CONNECT** and select **Goodix_Fast_DFU** from the device list (the advertising name of the SK Board downloaded with Fast DFU firmware; for details, see "Section 4.2 Downloading Fast DFU Firmware to SK Board") to establish a connection, as shown in Figure 4-6.

If a pop-up message **Connect Success** displays, connection between the mobile phone and the SK Board has been established, as shown in Figure 4-7.

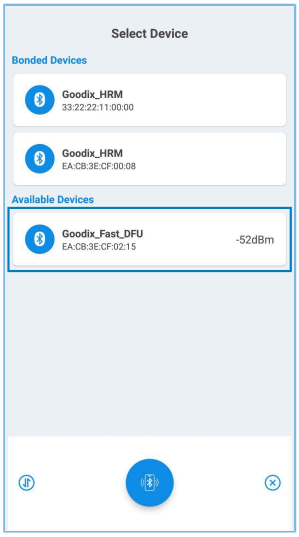

Figure 4-6 Connecting to **Goodix_Fast_DFU**

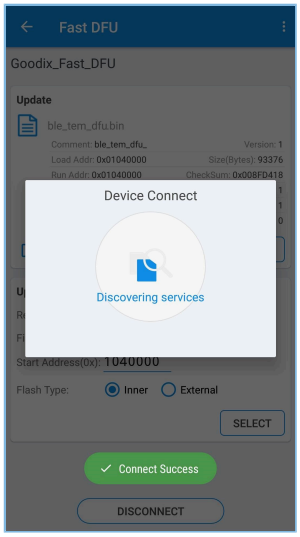

Figure 4-7 **Connect Success**

4. Tap **UPDATE** (Figure 4-8) to start updating.

   When the progress achieves 100%, the firmware update is completed.

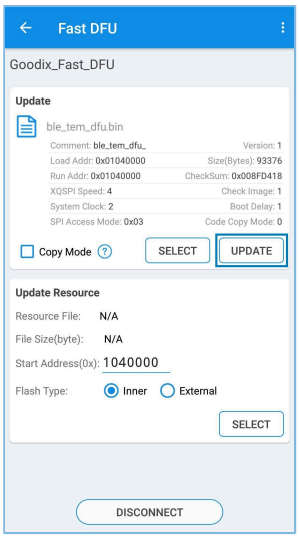

Figure 4-8 Tapping **UPDATE**

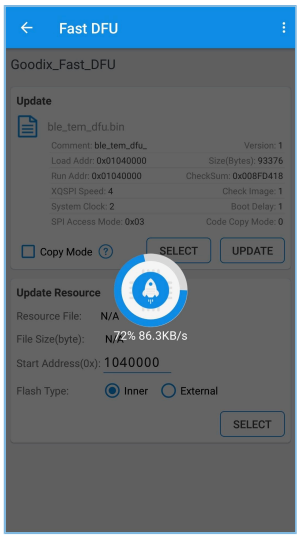

Figure 4-9 Updating

5. Check whether Fast DFU is completed successfully.

   The device automatically starts and runs the target firmware after the update is completed. If the target firmware (the updated firmware) runs successfully, it initiates an advertisement named as **Goodix_Tem_DFU**. Run GRToolbox and enter the **Device** interface. Pull to refresh the device list (or tap **SCAN** in the upper-right corner) to check whether **Goodix_Tem_DFU** is in the device list.

   If **Goodix_Tem_DFU** is in the device list (as shown in Figure 4-10), Fast DFU is completed successfully.

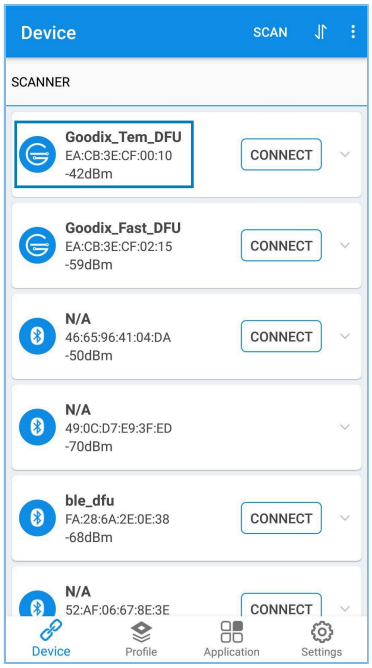

Figure 4-10 Discovering **Goodix_Tem_DFU**

5 FAQ

This chapter describes possible problems, reasons, and solutions during Fast DFU.

- Description

  Firmware update fails.

- Analysis

  1. For Fast DFU at 2.4 GHz with severe interference, the Bluetooth connection may be broken, resulting in update failure.
  2. The run address of the target firmware conflicts with that of the Fast DFU firmware, so an error occurs during update, resulting in update failure.

- Solution

  1. Make sure there is no severe interference in the operating environment of Fast DFU.
  2. Make sure the run address of the target firmware does not conflict with that of the Fast DFU firmware.