GR551x固件升级指南

版本： **1.8**

发布日期： **2021-08-09**

深圳市汇顶科技股份有限公司

商标声明

GOODiX 和其他汇顶商标均为深圳市汇顶科技股份有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人持有。

免责声明

本文档中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。

深圳市汇顶科技股份有限公司（以下简称“GOODIX”）对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。GOODIX对因这些信息及使用这些信息而引起的后果不承担任何责任。

未经GOODIX书面批准，不得将GOODIX的产品用作生命维持系统中的关键组件。在GOODIX知识产权保护下，不得暗中或以其他方式转让任何许可证。

深圳市汇顶科技股份有限公司

总部地址：深圳市福田保税区腾飞工业大厦B座2层、13层

电话：+86-755-33338828　　传真：+86-755-33338099

网址：www.goodix.com

前言

编写目的

本文档将介绍GR551x固件升级的原理和应用，以帮助用户深入了解GR551x的固件升级模式，便于快速开发。

读者对象

本文适用于以下读者：

- GR551x用户
- GR551x开发人员
- GR551x测试人员
- 文档工程师

版本说明

本文档为第6次发布，对应的产品系列为GR551x。

修订记录

版本	日期	修订内容
1.0	2019-12-08	首次发布
1.3	2020-03-16	更新文档页脚版本时间
1.5	2020-05-30	更新“DFU存储”章节中Flash Boot_Info的结构图，优化“DFU方式”章节描述
1.6	2020-06-30	更新“配置外部Flash命令”章节中主机端发送的数据内容
1.7	2020-12-25	• 更新“操作步骤”和“裸机状态下应用固件DFU时升级固件失败”章节内容 • 新增“使用OS的应用固件OTA DFU时出现异常”章节
1.8	2021-08-09	更新“准备工作”章节

目录

1 简介

设备固件升级（Device Firmware Update，DFU）是一种引导加载机制，用于更新目标设备的固件，以修复产品缺陷，丰富产品功能。

本文档主要介绍了GR551x的固件升级方式、原理，以及DFU功能的启用和测试。

在进行操作前，可参考以下文档。

表 1-1 文档参考

名称	描述
GR551x开发者指南	GR551x软硬件介绍、快速使用及资源总览
Bluetooth Core Spec	Bluetooth官方标准核心规范
J-Link用户指南	J-Link使用说明：www.segger.com/downloads/jlink/UM08001_JLink.pdf
Keil用户指南	Keil详细操作说明：www.keil.com/support/man/docs/uv4/
GR551x OTA示例手册	GR551x固件空中升级示例手册
GR551x BLE Stack用户指南	介绍了GR551x低功耗蓝牙协议栈各层的基本功能

2 GR551x DFU

本章介绍了GR551x DFU的基础概念，如DFU方式、设备角色、固件格式等。

2.1 DFU方式

GR551x支持以下两种DFU方式：

- 在应用固件中进行升级：运行应用固件即可实现目标固件下载；待目标固件下载完成后，系统跳转至目标固件运行。升级过程中，手机端不断开蓝牙连接就能直接升级固件，用户体验佳。
- 跳转Boot进行空中升级（Over The Air，OTA）：升级固件时，系统需从应用程序跳转至Boot程序，由Boot程序进行目标固件下载；下载完成后，系统跳转至目标固件运行。升级过程中，手机端需断开蓝牙连接，然后通过蓝牙重连Boot固件。与应用固件中进行升级相比，此方式更能充分利用Flash空间。

说明:

- 若采用在应用固件中升级方式，需提前规划当前应用固件、目标固件地址，且满足：将目标固件的Code Load Address和当前应用固件的Code Load Address设置为不同。
- 在应用固件中升级方式支持“拷贝升级模式”，即将目标固件先升级到其他未使用Flash地址，完成后再拷贝到实际运行地址处，详细操作可参考《GR551x OTA示例手册》“连接开发板并升级固件”章节。

2.2 DFU设备角色

DFU设备角色包括：

- 控制设备（主机端）：向目标设备发送升级数据，比如手机。
- 目标设备（设备端）：接收来自控制设备的升级数据，比如手环。

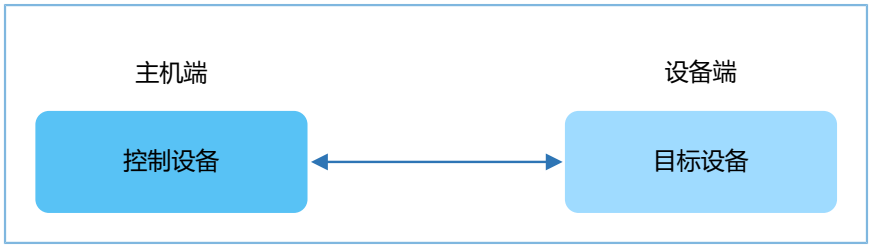

图 2-1 DFU设备角色

2.3 DFU固件格式

DFU传输的固件是bin格式，包括非加密固件bin和加密固件bin。

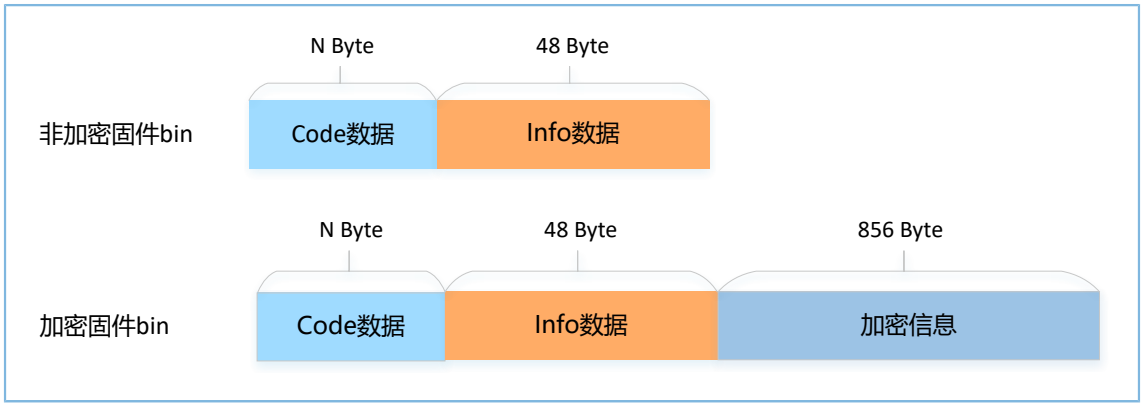

图 2-2 固件bin的数据格式

数据格式各字段说明如下：

- Code数据：固件本身数据，需要16字节对齐，N表示长度可变。
- Info数据：固件的描述信息。
- 加密信息：将非加密固件加密为加密固件所使用到的加密数据信息。

Info数据的格式（小端模式）如下：

表 2-1 Info数据的格式

Byte	Field	Description		
0 - 1	pattern	芯片数据标识，值为0x4744		
2 - 3	version	版本信息		
4 - 7	bin_size	Code数据的长度（Byte）		boot info（24B）
8 - 11	check_sum	Code数据位的校验和		
12 - 15	load_addr	Code数据存储的起始地址		
16 - 19	run_addr	Code数据的起始运行地址		
20 - 23	xqspi_xip_cmd	SPI访问模式		
24 - 27	boot config	位域	[0:3]：Clock Speed [4]：Code Copy Mode [5:7]：System Clock [8]：Check Image [9]：Boot Delay Time [10:31]：Reserved	
28 - 39	comments	固件描述信息		
40 - 47	reserved	用于16字节对齐，值为0x00		

2.3.1 DFU存储

固件信息存储在GR551x Flash的Img_info区域，起始地址为0x01000040，长度为400个字节，最多可存储10个固件信息。

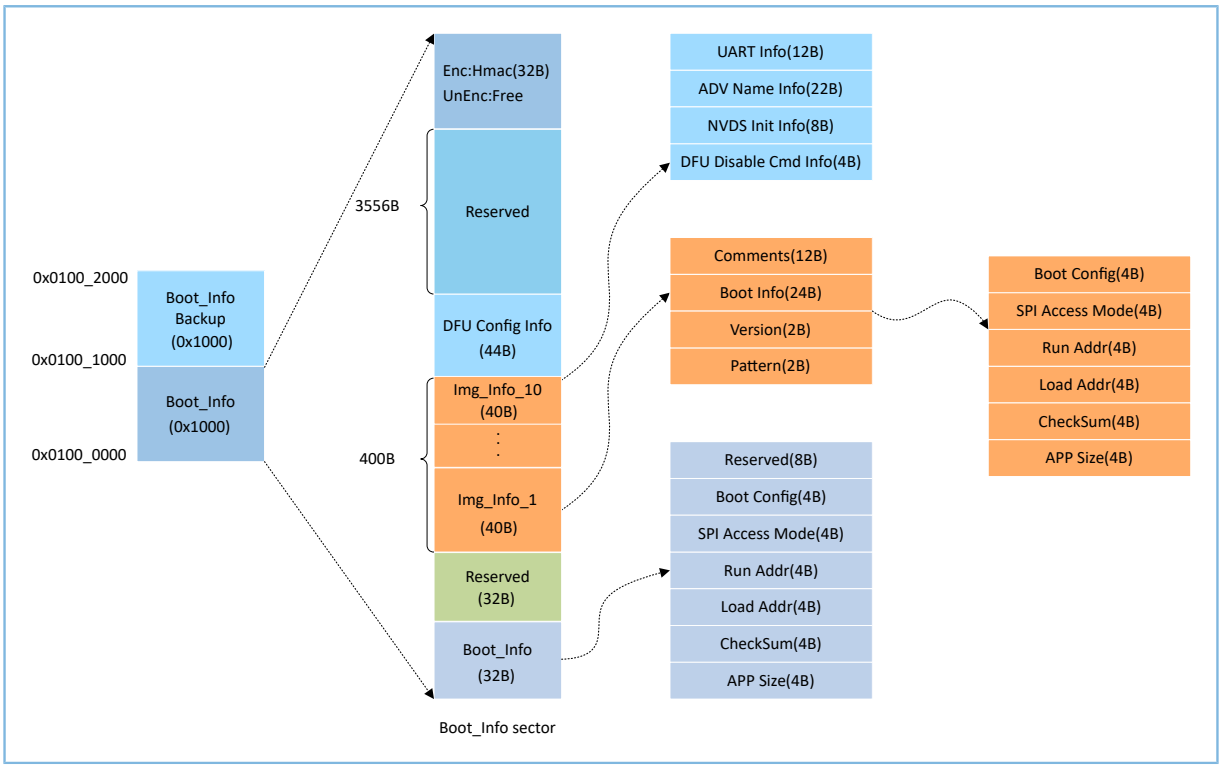

图 2-3 Flash中Boot_Info的结构

2.4 DFU通信协议

主机端和设备端基于DFU通信协议进行固件升级。

2.4.1 基础帧定义

基础帧定义了通信中最底层的数据包结构。应用数据包协议建立在基础帧之上，位于基础帧的“数据”域。如果基础帧长度超过链路通信的最大载荷，主机端需要将其分成片段发送。设备端在收到正确的帧头与数据长度后，开始处理数据。

2.4.2 帧结构定义

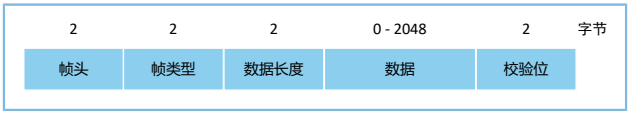

图 2-4 帧结构

- 帧头：标识帧的开始，以字符‘G’和‘D’的ASCII码值0x47和0x44表示。
- 帧类型：用于区别“数据”域中的数据类型。
- 数据长度：“数据”域的长度值。

- 数据：数据长度可变，最长为2048字节。
- 校验位：帧类型、应答、数据长度、数据的16位校验和。

2.4.3 字节顺序定义

基础帧的数据域采用小端模式，即低字节在前、高字节在后。

2.5 DFU命令集

DFU命令由主机端发送，设备端接收。

表 2-2 DFU命令说明

命令	命令代码	描述
Program Start	0x0023	在向设备端下载程序时，主机端发送Image Info信息。
Program Flash	0x0024	从指定的地址开始一次最多向Flash写入1024个字节，此指令可指定写Flash的方式（擦除写/不擦除写）。
Program End	0x0025	主机端发送此指令告知设备端编程数据已经发送完成。数据段第一个字节为Reset Flag，用于指示设备端编程数据发送完成后，是否需要复位并执行当前应用程序。
Config external flash	0x002A	配置外部Flash。
Get flash information	0x002B	获取Flash信息。

2.5.1 Program Start命令

主机端使用此命令将Image Info信息（即不含Reserve字段的Info数据）发送给设备端，长度为40字节。

设备端收到Image Info信息后，进行判断：

- 如果是对内部Flash进行编程，则需校验编程数据的Code Load Address是否为内部Flash地址。
- 如果是对外部Flash进行编程，则需指定编程地址和编程数据大小。

2.5.1.1 主机端发送数据

表 2-3 Program Start发送数据格式

字节序号	描述		有效值	说明
0 - 1	帧头		0x4744	以字符‘G’和‘D’的ASCII码值0x47和0x44表示
2 - 3	帧类型		0x0023	Program Start命令
4 - 5	数据长度		0x0029/0x0009	• 若Program的对象为固件，则数据域为41个字节，包括1个字节的Flash类型和长度为40个字节Image Info信息 • 若Program的对象为数据，则数据域为9个字节，包括1个字节的Flash类型、4个字节起始地址和4个字节数据内容
6	数据	Flash类型	0x00/0x01	• 0x00：内部Flash • 0x01：外部Flash

字节序号	描述		有效值	说明
7 - 14或 7 - 46		Flash写入数据	每字节的取值范围为：0x00 - 0xFF	数据的内容
15 - 16或 47 - 48	校验和		0x0000 - 0xFFFF	帧类型、数据长度和数据域的校验和（16 bits）

2.5.1.2 设备端回应数据

表 2-4 Program Start回应数据格式

字节序号	描述	有效值	说明
0 - 1	帧头	0x4744	以字符‘G’和‘D’的ASCII码值0x47和0x44表示
2 - 3	帧类型	0x0023	Program Start命令
4 - 5	数据长度	0x0001	数据域内容的长度
6	应答	0x01/0x02	• 0x01：成功 • 0x02：失败
7 - 8	校验和	0x0000 - 0xFFFF	帧类型、数据长度和应答域的校验和（16 bits）

2.5.2 Program Flash命令

主机端使用此命令将数据写入设备端Flash（内部或外部Flash）的有效地址。在收到此命令后，设备端解析待写入数据的起始地址、长度和内容。如果该起始地址有效，设备端将从该地址开始写入数据（数据格式如表 2-5 所示）并返回ACK，否则将返回NACK。

2.5.2.1 主机端发送数据

表 2-5 Program Flash发送数据格式

字节序号	描述		有效值	说明
0 - 1	帧头		0x4744	以字符‘G’和‘D’的ASCII码值0x47和0x44表示
2 - 3	帧类型		0x0024	Program Flash命令
4 - 5	数据长度		0x0007 - 0x0800	数据域内容的长度
6	数据	Program类型	0x00/0x01/0x02/ 0x10/0x11/0x12	• 0x00：对内部Flash指定地址的页进行擦除后存储。 • 0x01：对内部Flash按照Program Start命令下发的Image Info信息进行Flash存储。 • 0x02：对内部Flash直接调用Flash write接口写入数据。 • 0x10：对外部Flash指定地址的页进行擦除后存储。 • 0x11：对外部Flash按照Program Start命令下发的Image Info信息进行Flash存储。 • 0x12：对外部Flash直接调用Flash write接口写入数据。

字节序号	描述		有效值	说明
7 - 10		起始地址	每字节的取值范围为0x00 - 0xFF	设备端Flash有效地址
11 - 12		Flash写入数据长度	0x0000 - 0x00FF	Flash写入数据长度最大值为1024个字节
13 - N		Flash写入数据	每字节的取值范围为0x00 - 0xFF	写入Flash的数据内容
N+1 - N+2	校验和		0x0000 - 0xFFFF	帧类型、数据长度和数据域的校验和（16 bits）

说明:

表 2-5 中的N表示数据域的长度可变，N取值为：14 ～ 1036。

2.5.2.2 设备端回应数据

表 2-6 Program Flash回应数据格式

字节序号	描述	有效值	说明
0 - 1	帧头	0x4744	以字符‘G’和‘D’的ASCII码值0x47和0x44表示
2 - 3	帧类型	0x0024	Program Flash命令
4 - 5	数据长度	0x0001	应答1字节
6	应答	0x01/0x02	• 0x01：成功 • 0x02：失败
7 - 8	校验和	0x0000 - 0xFFFF	帧类型、数据长度和应答域的校验和（16 bits）

2.5.3 Program End命令

主机端使用此命令告知设备端编程数据已发送完成，数据段包含设备端下一次Reset后的启动地址，以及复位类型标记（Reset Flag）。复位类型标记用于指示设备端接收Program End命令后是否立即执行下载的程序。

2.5.3.1 主机端发送数据

表 2-7 Program End发送数据格式

字节序号	描述	有效值	说明
0 - 1	帧头	0x4744	以字符‘G’和‘D’的ASCII码值0x47和0x44表示
2 - 3	帧类型	0x0025	Program End命令
4 - 5	数据长度	0x0005	数据域内容的长度

字节序号	描述		有效值	说明
6	数据	复位类型标记	0x00/0x01/0x02/0x12	• 0x00：仅将固件Info信息存储到Flash的Img Info区域。 • 0x01：将固件Info信息存储到Flash的Img Info和Boot区域，并在Reset后运行编程固件。 • 0x02：下载数据到内部Flash时，不会对Img Info和Boot区域进行操作。 • 0x12：下载数据到外部Flash。
7 - 10		编程文件的校验和	每字节的取值范围为：0x00 - 0xFF	bin文件的校验和
11 - 12	校验和		0x0000 - 0xFFFF	帧类型、数据长度和数据域的校验和（16 bits）

2.5.3.2 设备端回应数据

表 2-8 Program End回应数据格式

字节序号	描述	有效值	说明
0 - 1	帧头	0x4744	以字符‘G’和‘D’的ASCII码值0x47和0x44表示
2 - 3	帧类型	0x0025	Program End命令
4 - 5	数据长度	0x0001	应答1字节
6	应答	0x01/0x02	• 0x01：成功 • 0x02：失败
7 - 8	校验和	0x0000 - 0xFFFF	帧类型、数据长度和应答域的校验和（16 bits）

2.5.4 操作System Configuration区域数据命令

主机端使用此命令操作设备端System Configuration区域数据，包括读取、更新数据。System Configuration区域的地址，请参考2.3.1 DFU存储。

2.5.4.1 主机端发送数据

表 2-9 操作System Configuration区域的发送数据格式

字节序号	描述		有效值	说明
0 - 1	帧头		0x4744	以字符‘G’和‘D’的ASCII码值0x47和0x44表示
2 - 3	帧类型		0x0027	操作System Configuration区域数据命令
4 - 5	数据长度		0x0007 - 0x0407	数据域内容的长度
6	数据	操作命令	0x00/0x01	• 0x00：读取System Configuration区域数据。 • 0x01：更新System Configuration区域数据。
7 - 10		起始地址	0x01000000 - 0x01002000	地址范围必须是System Configuration区域内有效地址
11 - 12		数据长度	0x0000 - 0x0400	读取或更新数据的内容长度

字节序号	描述		有效值	说明
13 - N		更新数据	每字节的取值范围为： 0x00 - 0xFF	如果是读取数据命令，则没有更新数据域
N+1 - N+2	校验和		0x0000 - 0xFFFF	帧类型、数据长度和数据域的校验和（16 bits）

说明:

表 2-9 中的N表示数据域的长度可变：

- 若是更新数据命令，则N取值为：14 ～ 1036。
- 若是读取数据命令，则数据域为固定长度：7字节。

2.5.4.2 设备端回应数据

表 2-10 操作System Configuration区域的回应数据格式

字节序号	描述		有效值	说明
0 - 1	帧头		0x4744	以字符‘G’和‘D’的ASCII码值0x47和0x44表示
2 - 3	帧类型		0x0027	操作System Configuration区域数据命令
4 - 5	数据长度		0x0002 - 0x0402	数据域内容的长度
6	数据	应答	0x01/0x02	• 0x01：成功 • 0x02：失败
7		操作命令	0x00/0x10 0x01/0x11	• 0x00：读取System Configuration，芯片为非加密。 • 0x10：读取System Configuration，芯片为加密。 • 0x01：更新System Configuration，芯片为非加密。 • 0x11：更新System Configuration，芯片为加密。
8 - 11		System Configuration区域起始地址	0x01000000 - 0x01002000	如果是更新System Configuration区域命令，则没有此数据域。
12 - 13		System Configuration区域数据长度	0x0000 - 0x0400	
14 - N		System Configuration区域的数据内容	每字节的取值范围为： 0x00 - 0xFF	
N+1 - N+2	校验和		0x0000 - 0xFFFF	帧类型、数据长度和数据域的校验和（16 bits）

2.5.5 配置外部Flash命令

主机端使用此命令配置设备端外部Flash SPI接口。

2.5.5.1 主机端发送数据

表 2-11 配置外部Flash的发送数据格式

节序号	描述		有效值	说明
0 - 1	帧头		0x4744	以字符’G’和’D’的ASCII码值0x47和0x44表示
2 - 3	帧类型		0x002A	配置外部Flash命令
4 - 5	数据长度		0x0014	数据内容长度
6	数据	外部Flash类型	0x01/0x02	• 0x01：SPI Flash • 0x02：QSPI Flash
7 - 9		CS IO TYPE	00 - 03	GPIO类型选择
		CS PIN	00 - 31	GPIO PIN选择
		CS IO MUX	00 - 09	PIN MUX选择
10 - 12		CLK IO TYPE	00 - 03	GPIO类型选择
		CLK PIN	00 - 31	GPIO PIN选择
		CLK IO MUX	00 - 09	PIN MUX选择
13 - 15		MOSI(IO0) IO TYPE	00 - 03	GPIO类型选择
		MOSI(IO0) PIN	00 - 31	GPIO PIN选择
		MOSI(IO0) IO MUX	00 - 09	PIN MUX选择
16 - 18		MOSI(IO1) IO TYPE	00 - 03	GPIO类型选择
		MOSI(IO1) PIN	00 - 31	GPIO PIN选择
		MOSI(IO1) IO MUX	00 - 09	PIN MUX选择
19 - 21		IO2 IO TYPE	00 - 03	GPIO类型选择，仅QSPI有效
		IO2 PIN	00 - 31	GPIO PIN选择，仅QSPI有效
		IO2 IO MUX	00 - 09	PIN MUX选择，仅QSPI有效
22 - 24		IO3 IO TYPE	00 - 03	GPIO类型选择，仅QSPI有效
		IO3 PIN	00 - 31	GPIO PIN选择，仅QSPI有效
		IO3 IO MUX	00 - 09	PIN MUX选择，仅QSPI有效
25		QSPI ID	00 - 02	QSPI Module ID，仅QSPI有效
26 - 27	校验和		0x0000 - 0xFFFF	帧类型、数据长度和数据域的校验和（16 bits）

2.5.5.2 设备端回应数据

表 2-12 配置外部Flash的回应数据格式

字节序号	描述	有效值	说明
0 - 1	帧头	0x4744	以字符‘G’和‘D’的ASCII码值0x47和0x44表示
2 - 3	帧类型	0x002A	初始化外部Flash命令
4 - 5	数据长度	0x0001	应答1字节
6	应答	0x01/0x02	• 0x01：成功 • 0x02：失败
7 - 8	校验和	0x0000 - 0xFFFF	帧类型、数据长度和应答域的校验和（16 bits）

2.5.6 获取Flash信息

主机端使用此命令获取内外部Flash信息，包括Flash ID和Flash Size。外部Flash Size通过SFDP（Serial Flash Discoverable Parameters）协议获取。凡是支持SFDP协议的Flash芯片，都可通过此命令获取Flash Size。

2.5.6.1 主机端发送数据

表 2-13 获取Flash信息的发送数据格式

字节序号	描述	有效值	说明
0 - 1	帧头	0x4744	以字符‘G’和‘D’的ASCII码值0x47和0x44表示
2 - 3	帧类型	0x002B	获取Flash ID值
4 - 5	数据长度	0x0001	长度为1字节
6	Flash类型	0x00/0x01	• 0x00：内部Flash • 0x01：外部Flash
7 - 8	校验和	0x0000 - 0xFFFF	帧类型、数据长度和数据域的校验和（16 bits）

2.5.6.2 设备端回应数据

表 2-14 获取Flash信息的回应数据格式

字节序号	描述			有效值	说明
0 - 1	帧头			0x4744	以字符‘G’和‘D’的ASCII码值0x47和0x44表示
2 - 3	帧类型			0x002B	获取Flash ID值
4 - 5	数据长度			0x0009	应答9字节
6	数据	应答		0x01/0x02	• 0x01：成功 • 0x02：失败
7 - 14		Flash信息	Flash ID	每字节的取值范围为：0x00 - 0xFF	Flash ID值（4字节）
			Flash Size	每字节的取值范围为：0x00 - 0xFF	Flash Size值（4字节）

字节序号	描述	有效值	说明
15 - 16	校验和	0x0000 - 0xFFFF	帧类型、数据长度和数据域的校验和（16 bits）

3 启用DFU功能

本章将介绍如何启用GR551x芯片的DFU功能。

3.1 在应用固件中添加DFU功能

DFU功能封装在SDK_Folder\components\libraries\dfu_port模块中。用户只需调用相应的API接口，就能在应用中添加DFU功能。下面以ble_dfu_boot工程为例进行说明。

说明:

SDK_Folder为GR551x SDK的根目录。

3.1.1 ble_dfu_boot工程

示例ble_dfu_boot的keil工程文件位于目录：SDK_Folder\projects\ble\dfu\ble_dfu_boot，其中工程文件位于Keil_5文件夹。

双击打开*ble_dfu_boot.uvprojx*工程文件，在Keil中查看ble_dfu_boot示例的工程目录结构，相关文件说明如表 3-1 所示。

表 3-1 ble_dfu_boot文件说明

Group	文件	描述
gr_profiles	ble_prf_utils.c	Profile相关操作工具
	ota.s	OTA Service实现
user_callback	user_gap_callback.c	GAP Callback实现，如连接、断连、GAP参数更新等
	user_gatt_common_callback.c	GATT Common Callback实现，如MTU交换
user_platform	user_periph_setup.c	串口设备地址、电源管理模式、DFU的配置
user_app	main.c	main()入口函数
	user_app.c	OTA Profile注册及逻辑处理

3.1.2 操作步骤

1. 配置DFU升级的接口，用户可按需选择BLE升级或串口升级方式。
   - 选择BLE无线的升级方式

     路径：工程目录下user\user_app.c

     名称：services_init();

     在该函数中调用dfu_service_init接口，初始化用于进行升级的BLE服务。

```
static void services_init(void)
{
    dfu_service_init(NULL);
```

```
}
```

路径：工程目录下user_callback\user_gatt_common_callback.c

名称：app_gatt_mtu_exchange_cb();

在该函数中调用dfu_ble_set_mtu_size接口，以配置BLE升级方式所需的MTU值。

```
static void app_gatt_mtu_exchange_cb(uint8_t conn_idx, uint8_t status, uint16_t mtu)
{
    if(BLE_SUCCESS == status)
    {
       dfu_ble_set_mtu_size(mtu);
    }
}
```

路径：工程目录下platform\user_periph_setup.c

名称：app_periph_init();

在该函数中调用dfu_port_init接口，初始化DFU模块。

```
void app_periph_init(void)
{
    SYS_SET_BD_ADDR(s_bd_addr);
    bsp_uart_init();
    app_log_assert_init();
    pwr_mgmt_mode_set(PMR_MGMT_SLEEP_MODE);
    dfu_port_init(NULL,NULL);
}
```

- 选择串口升级的方式

  路径：工程目录下platform\user_periph_setup.c

  名称：app_uart_evt_handler();

  在APP_UART_EVT_RX_DATA事件中调用app_uart_receive_async和dfu_uart_receive_data_process接口，以实现串口升级方式。

```
void app_uart_evt_handler(app_uart_evt_t * p_evt)
{
    switch(p_evt->type)
    {
        case APP_UART_EVT_TX_CPLT:
            break;
        case APP_UART_EVT_RX_DATA:
            app_uart_receive_async(APP_UART_ID_0, uart_rx_data, UART_RX_SIZE);
            dfu_uart_receive_data_process(uart_rx_data, p_evt->data.size);
            break;
        case APP_UART_EVT_ERROR:
            break;
        default:break;
```

```
    }
```

路径：工程目录下platform\user_periph_setup.c

名称：app_periph_init();

在该函数中进行串口初始化和DFU模块初始化（需要将串口发送接口uart_send_data挂载进DFU模块中），并调用UART异步接收接口。

```
void app_periph_init(void)
{
    SET_BD_ADDR(BD_ADDR_NVDS_TAG, BD_ADDR_LENGTH, s_bd_addr);
    bsp_uart_init();
    app_uart_receive_async(APP_UART_ID_0, uart_rx_data, UART_RX_SIZE);
    dfu_port_init(uart_send_data, &dfu_pro_call);
    pwr_mgmt_mode_set(PMR_MGMT_ACTIVE_MODE);
}
```

表 3-2 dfu_port_init接口的入参说明

参数	说明	取值
uart_send_data	是否需要实现串口升级功能	是：该参数赋值用户实现的串口发送接口 否：该参数赋值“NULL”
dfu_pro_call	是否需要在应用层监测升级状态	是：该参数赋值DFU状态处理callback 否：该参数赋值“NULL”

2. 在main函数的while(1)中调用dfu_schedule接口。

   路径：工程目录下user\main.c

   名称：main();

   在该函数中调用dfu_schedule接口，以实现dfu_schedule的调度。

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

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

    // loop
    while (1)
    {
        dfu_schedule();
    #if SK_GUI_ENABLE
        gui_refresh_schedule();
    #endif
        pwr_mgmt_schedule();
    }
```

```
}
```

说明:

若在有OS的应用中使用DFU，则建议专门为DFU创建一个task来调度dfu_schedule；同时为该任务分配的任务栈大小不能小于6 KB，否则会导致任务栈溢出。

3.2 跳转至Boot程序进行固件升级

如果用户采用跳转Boot进行空中升级的方式，需要将待升级的DFU Boot固件（如本示例的ble_dfu_boot）下载到GR551x芯片。升级应用跳转到DFU Boot，可参考ble_app_template_dfu示例。

ble_app_template_dfu工程的源代码和工程文件位于`SDK_Folder\projects\ble\ble_peripheral\ble_app_template_dfu`，其工程文件在Keil_5文件夹下。

双击打开工程文件*ble_app_template_dfu.uvprojx*，在Keil中编译该文件。DFU Boot固件Info数据将显示在Keil的log窗口。

路径：工程目录下`user\user_app.c`

名称：dfu_enter();

当应用接收到需要升级的指令时，在该函数中调用dfu_start_address接口，以传入DFU Boot固件的boot info信息。

```
static void dfu_enter(void)
{
    //use flash dfu boot
    boot_info_t boot_info =
    {
    .bin_size = 0x26cf0,
    .check_sum = 0xf04eff,
    .load_addr = 0x1002000,
    .run_addr = 0x1002000,
    .xqspi_xip_cmd = 0xeb,
    .xqspi_speed = 0x0,
    .code_copy_mode = 0x0,
    .system_clk = 0x0,
    .check_image = 0x0,
    .boot_delay = 0x1,
    .is_dap_boot = 0x1,
     };
    dfu_start_address(&boot_info);
}
```

4 GR551x OTA DFU

主机端可以通过无线传输方式为目标设备进行固件空中升级。采用BLE无线升级固件，需要设置OTA Profile和OTA Service。

4.1 BLE OTA Profile

OTA Profile是Goodix自定义的Profile，包括以下两种设备角色：

- 主机端：即中央（Central）设备，发送用户程序、用户固定数据（如：字库、图片数据和外部程序数据）。中央设备可以是电脑、手机等。
- 设备端：即外围（Peripheral）设备，接收用户程序、用户固定数据。外围设备可以是手环、心率计等。

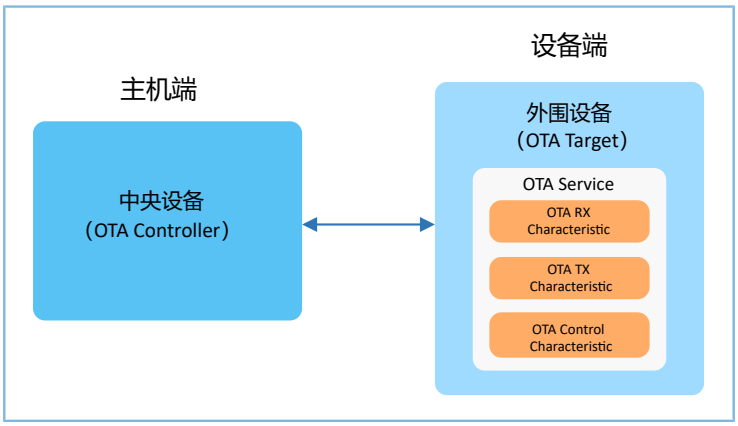

图 4-1 OTA Profile设备角色

4.2 BLE OTA Service

OTA Service是Goodix自定义的Service，它描述了设备固件更新所必需的信息。OTA Service是一个独立的Service，支持以下GATT行为：

- Write Characteristic Value
- Notifications
- Read Characteristic Descriptors
- Write Characteristic Descriptors

4.2.1 OTA Service 和OTA Characteristics

OTA Service的UUID为a6ed0401-d344-460a-8075-b9e8ec90d71b。OTA Characteristic包括3种特征，如表 4-1 所示。

表 4-1 OTA Characteristic说明

Description	UUID	Properties
OTA RX Characteristic	a6ed0402-d344-460a-8075-b9e8ec90d71b	Write without Response

Description	UUID	Properties
OTA TX Characteristic	a6ed0403-d344-460a-8075-b9e8ec90d71b	Notify, Indicate
OTA Control Characteristic	a6ed0404-d344-460a-8075-b9e8ec90d71b	Write without Response, Indicate

各特征的作用如下：

- OTA RX Characteristic：接收主机端下发的数据，属性为Write without Response，有利于提升数据传输速率。
- OTA TX Characteristic：将数据发送给主机端，属性为Notify。设备端上报给主机端的数据，都以Notify形式给出。
- OTA Control Characteristic：接收主机端下发的控制指令，比如使设备进入DFU程序。

4.3 OTA DFU测试

若需进行OTA DFU测试，详细操作请参考《GR551x OTA示例手册》。

5 串口DFU测试

本章将介绍使用开发板进行串口DFU测试。

5.1 准备工作

测试之前，需要完成以下准备工作。

- 硬件准备

表 5-1 硬件准备

名称	描述
开发板	GR5515 Starter Kit开发板（以下简称开发板）2块
数据线	Micro USB 2.0数据线

- 软件准备

表 5-2 软件准备

名称	描述
Windows	Windows 7/Windows 10操作系统
J-Link Driver	J-Link驱动程序，下载网址：www.segger.com/downloads/jlink/
Keil MDK5	IDE工具，支持MDK-ARM 5.20 及以上版本，下载网址：www.keil.com/download/product/
GProgrammer（Windows）	Programming工具，位于：`SDK_Folder\tools\GProgrammer`

5.2 测试验证

串口DFU测试包括：

- 使用GProgrammer工具进行DFU测试。
- 使用SDK中DFU Master和DFU Boot固件进行DFU测试。

5.2.1 GProgrammer工具测试

通过GProgrammer UART方式下载*ble_dfu_boot_fw.bin*固件至作为设备端的开发板B并运行，具体操作方法请参考《GProgrammer用户手册》。

说明:

*ble_dfu_boot_fw.bin*位于`SDK_Folder\projects\ble\dfu\ble_dfu_boot\build\`。

5.2.2 DFU Master和DFU Boot测试

在本测试中，开发板A作为DFU的主机端（DFU Master），开发板B作为设备端（DFU Boot）。

说明:

DFU Master示例工程的源代码和工程文件位于`SDK_Folder\projects\ble\dfu\dfu_master`，其*dfu_master.uvprojx*工程文件在Keil_5文件夹下。

DFU Boot固件*ble_dfu_boot_fw.bin*的工程目录，请参考5.2.1 GProgrammer工具测试。

使用SDK包中DFU Master和DFU Boot固件进行测试的步骤为：

1. 去掉开发板A和B上J5排针的5和6脚、7和8脚的短线帽。
2. 使用杜邦线将开发板A的J5 5脚和开发板B的J5 7脚相连，将开发板A的J5 7脚和开发板B的J5 5脚相连，使这两块开发板的串口相连接。
3. 使用GProgrammer分别下载dfu_master固件至开发板A、dfu_boot固件至开发板B。
4. 使用GProgrammer下载待升级的bin文件到开发板A。
5. 根据开发板A上的显示屏提示，操控开发板A上的按键进行串口升级。

说明:

- *ble_dfu_boot_fw.bin*（DFU Boot）的Load Address和Run Address为0x01002000。
- 为防止覆盖已下载的DFU Boot固件，用户需将待升级固件与DFU Boot的Load Address设置为不同，具体操作可参考《GR551x OTA示例手册》“创建目标升级固件”。

6 常见问题

本章列出了DFU过程中可能出现的问题和解决办法。

6.1 裸机状态下应用固件DFU时升级固件失败

- 问题描述

  使用BLE DFU时升级固件失败。

- 问题分析

  可能有多种原因。如在2.4 GHz干扰较严重的环境中使用BLE升级固件，蓝牙连接可能断开；或因配置不当导致升级固件失败。

- 处理方法

  1. 如果使用BLE升级，检查设备是否已经断开连接。
  2. 如果使用串口升级，检查主机端和设备端串口波特率是否一致。
  3. 检查升级固件地址和正在运行地址是否冲突。

6.2 使用OS的应用固件OTA DFU时出现异常

- 问题描述

  在使用FreeRTOS系统的情况下固件升级失败，出现其他task同样运行异常，或直接发生hardfault等异常现象。

- 问题分析

  除因6.1 裸机状态下应用固件DFU时升级固件失败中的原因外，也可能因任务栈大小分配不足导致任务栈溢出，OTA task任务栈至少需要分配6 KB。

- 处理方法

  为OTA task分配不小于6 KB的任务栈。