



GR551x BLE应用示例用户手册

版本： **1.6**

发布日期： **2020-06-30**

版权所有 © 2020 深圳市汇顶科技股份有限公司。保留一切权利。

非经本公司书面许可，任何单位和个人不得对本手册内的任何部分擅自摘抄、复制、修改、翻译、传播，或将其全部或部分用于商业用途。

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

GR551x SDK提供丰富的BLE应用示例程序。本文档列举了部分示例程序，简单介绍了代码理解和测试验证步骤，便于开发者快速理解和使用这些示例开发应用程序。

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
- 开发爱好者
- 文档工程师

版本说明

本文档为第4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	新增“6.3 DFU示例”章节
1.5	2020-05-30	优化4.4和5.2章节中Current Time Profile描述
1.6	2020-06-30	基于SDK刷新版本

目录

前言.....	I
1 简介.....	1
2 环境搭建.....	2
2.1 准备工作.....	2
2.2 硬件连接.....	2
2.3 下载固件.....	3
2.4 串口设置.....	3
3 应用示例概述.....	4
4 BLE Peripheral示例.....	7
4.1 警报通知Client应用示例.....	7
4.1.1 代码理解.....	7
4.1.2 测试验证.....	7
4.2 信标应用示例.....	7
4.2.1 代码理解.....	7
4.2.2 测试验证.....	8
4.3 血压应用示例.....	8
4.3.1 代码理解.....	9
4.3.2 测试验证.....	9
4.4 当前时间Server应用示例.....	9
4.4.1 代码理解.....	10
4.4.2 测试验证.....	10
4.5 自行车速度与踏频应用示例.....	10
4.5.1 代码理解.....	10
4.5.2 测试验证.....	11
4.6 血糖应用示例.....	11
4.6.1 代码理解.....	11
4.6.2 测试验证.....	12
4.7 心率应用示例.....	12
4.7.1 代码理解.....	13
4.7.2 测试验证.....	13
4.8 心率多连接应用示例.....	13
4.8.1 代码理解.....	13
4.8.2 测试验证.....	13
4.9 电话警报通知Client应用示例.....	14
4.9.1 代码理解.....	14
4.9.2 测试验证.....	14

4.10 接近应用示例.....	14
4.10.1 代码理解.....	14
4.10.2 测试验证.....	15
4.11 跑速与步频应用示例.....	15
4.11.1 代码理解.....	16
4.11.2 测试验证.....	16
4.12 体温计应用示例.....	16
4.12.1 代码理解.....	17
4.12.2 测试验证.....	17
4.13 吞吐率Server应用示例.....	17
4.13.1 代码理解.....	17
4.13.2 测试验证.....	18
4.14 ANCS Client应用示例.....	18
4.15 FreeRTOS模板应用示例.....	18
4.16 BLE鼠标应用示例.....	18
4.17 功耗测试应用示例.....	18
4.18 模板应用示例.....	19
4.19 串口Server应用示例.....	19
4.20 AMS Client应用示例.....	19
4.21 微信AirSync应用示例.....	19
5 BLE Central示例.....	20
5.1 警报通知Server应用示例.....	20
5.1.1 代码理解.....	20
5.1.2 测试验证.....	20
5.2 当前时间Client应用示例.....	20
5.2.1 代码理解.....	20
5.2.2 测试验证.....	21
5.3 电话警报通知Server应用示例.....	21
5.3.1 代码理解.....	21
5.3.2 测试验证.....	21
5.4 心率Client应用.....	21
5.4.1 代码理解.....	21
5.4.2 测试验证.....	22
5.5 跑速与步频Client应用.....	22
5.5.1 代码理解.....	22
5.5.2 测试验证.....	22
5.6 串口Client应用.....	23
5.6.1 代码理解.....	23
5.6.2 测试验证.....	23
6 其它示例.....	24

6.1 BLE Basic示例..... 24

6.2 BLE Multi-Role示例..... 24

6.3 DFU示例..... 24

6.4 DTM示例..... 24

1 简介

GR551x SDK提供了丰富的BLE应用示例程序以帮助用户快速理解SDK API的用法和实现基于GR551x的BLE应用，包括：

- Bluetooth SIG GATT Profile Spec定义的Server、Client或Broadcaster示例。
- 自定义GATT Profile示例。
- FreeRTOS、Apple Notification Center Service（ANCS）等示例。

在了解和使用示例程序前，建议参考以下文档。

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
GR551x BLE Stack用户指南	介绍GR551x支持的蓝牙低功耗协议栈
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/
Bluetooth Core Spec v5.1	Bluetooth官方标准核心规范5.1： https://www.bluetooth.com/specifications/bluetooth-core-specification/

2 环境搭建

本章介绍如何快速搭建示例程序的运行环境。

 说明:

SDK_Folder为GR551x SDK的根目录。

2.1 准备工作

运行示例程序之前，请完成以下准备工作。

- 硬件准备

表 2-1 硬件准备

名称	描述
J-Link工具	SEGGER公司推出的JTAG仿真器，如需更多了解，请访问 www.segger.com/products/debug-probes/j-link/
开发板	GR5515 Starter Kit开发板
连接线	Micro USB 2.0串口连接线
Android手机	操作系统Android 4.4（KitKat）及以上版本的手机
iOS设备	支持BLE 4.0及以上的iOS设备，如iPhone 4s/iPad 3及其以上版本

- 软件准备

表 2-2 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/
GProgrammer（Windows）	GR551x Programming工具，位于SDK_Folder\tools\GProgrammer
GRUart（Windows）	GR551x串口调试工具，位于SDK_Folder\tools\GRUart
GRToolbox（Android）	GR551x BLE调试工具，位于SDK_Folder\tools\GRToolbox

2.2 硬件连接

使用Micro USB 2.0数据线连接GR5515 Starter Kit开发板与计算机。

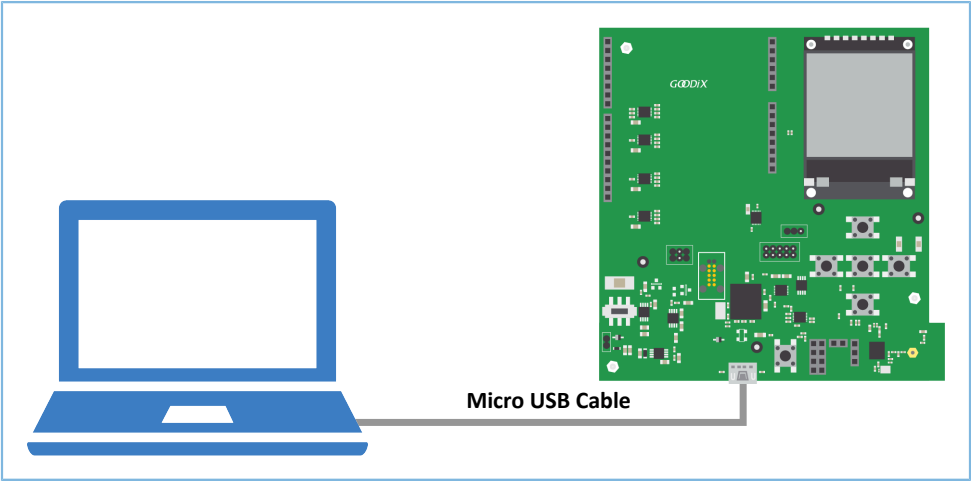


图 2-1 硬件连接示意图

2.3 下载固件

下载ble_app_template示例的ble_app_template_fw.bin固件至开发板。具体操作方法，请参考[《GProgrammer用户手册》](#)。

说明:

ble_app_template_fw.bin位于:

SDK_Folder\projects\ble\ble_peripheral\ble_app_template\build\, SDK_Folder是GR551x SDK的根目录。

2.4 串口设置

打开GRUart工具后，参照如表 2-3 参数设置串口。

表 2-3 GRUart串口配置参数

PortName	BaudRate	DataBits	Parity	StopBits	Flow Control
需根据实际选择	115200	8	None	1	不勾选

3 应用示例概述

BLE示例程序位于SDK_Folder\projects\ble\，具有以下共通性：

- 主函数main()

main()调用外设和BLE Stack的初始化函数，然后进入while(1){}主循环（Main Loop）。在该主循环内，开发者可以添加应用层事件调度代码（包括处理用户输入和GUI刷新等），比如[4.19 串口Server应用示例](#)；pwr_mgmt_schedule()必须在循环最后被调用，以实现自动功耗管理。

典型main.c文件内容如下：

```
STACK_HEAP_INIT(heaps_table);

static app_callback_t app_ble_callback =
{
    .app_ble_init_cmp_callback = ble_init_cmp_callback,
    .app_gap_callbacks = &app_gap_callbacks,
    .app_gatt_common_callback = &app_gatt_common_callback,
    .app_gattc_callback = &app_gattc_callback,
    .app_sec_callback = &app_sec_callback,
};

int main (void) {
    /*< Initialize user peripherals. */
    app_periph_init();
    /*< Initialize BLE Stack. */
    ble_stack_init(&app_ble_callback, &heaps_table);
    // Main Loop
    while(1)
    {
        app_log_flush();
#ifdef SK_GUI_ENABLE
        gui_refresh_schedule();
#endif
        pwr_mgmt_schedule();
    }
}
```

更多详细介绍请参考[《GR551x开发者指南》](#)“4.3.3.1 修改主函数”。

- 外设初始化函数app_periph_init()

配置APP Log模块后，设置功耗管理模式，开发者可根据需要进行其他必要硬件的初始化。

在产品开发过程中，为方便设置调试用的蓝牙设备地址（BD ADDR），可用SYS_SET_BD_ADDR()配置临时的蓝牙地址。示例代码如下：

```
/**@brief Bluetooth device address. */
static const uint8_t s_bd_addr[SYS_BD_ADDR_LEN] = {0x08, 0x00, 0xcf, 0x3e, 0xcb, 0xea};

static void app_log_assert_init(void)
```

```

{
    app_log_init_t log_init;
    log_init.filter.level = APP_LOG_LVL_DEBUG;
    log_init.fmt_set[APP_LOG_LVL_ERROR] = APP_LOG_FMT_ALL & (~APP_LOG_FMT_TAG);
    log_init.fmt_set[APP_LOG_LVL_WARNING] = APP_LOG_FMT_LVL;
    log_init.fmt_set[APP_LOG_LVL_INFO] = APP_LOG_FMT_LVL;
    log_init.fmt_set[APP_LOG_LVL_DEBUG] = APP_LOG_FMT_LVL;
    app_log_init(&log_init, bsp_uart_send, bsp_uart_flush);
    app_assert_default_cb_register();
}

void app_periph_init(void)
{
    SYS_SET_BD_ADDR(s_bd_addr);
    bsp_uart_init();
    app_log_assert_init();
    #if SK_GUI_ENABLE
        user_gui_init();
    #endif
    pwr_mgmt_mode_set(PMR_MGMT_SLEEP_MODE);
}

```

说明:

在产品的正式发布代码中请务必删除SYS_SET_BD_ADDR()。正式产品中的蓝牙地址存储在eFuse中，应用程序不需要再使用SYS_SET_BD_ADDR()在Flash中设置蓝牙地址。关于eFuse的介绍请参考 [《GR551x Datasheet》](#)；GProgrammer工具支持写入数据到eFuse中。

- 应用初始化Application Initialization

BLE Stack完成初始化后会调用类型为app_ble_init_cmp_cb_t的Callback函数。开发者可在该Callback函数中执行Application initialization，比如添加BLE Services、初始化GAP Parameters、设置Advertising Parameters并开始Advertising等。

在示例程序中，该Callback函数一般被实现在user_app.c中，由ble_stack_init()函数注册至SDK中。示例代码如下：

```

void ble_init_cmp_callback(void)
{
    sdk_err_t      error_code;
    gap_bdaddr_t   bd_addr;
    sdk_version_t  version;

    sys_sdk_verison_get(&version);
    APP_LOG_INFO("Goodix GR551x SDK V%d.%d (commit %d)", version.major, version.minor,
                version.commit_id);
    error_code = ble_gap_addr_get(&bd_addr);
    APP_ERROR_CHECK(error_code);

    APP_LOG_INFO("Local Board %02X:%02X:%02X:%02X:%02X:%02X",

```

```
        bd_addr.gap_addr.addr[5],
        bd_addr.gap_addr.addr[4],
        bd_addr.gap_addr.addr[3],
        bd_addr.gap_addr.addr[2],
        bd_addr.gap_addr.addr[1],
        bd_addr.gap_addr.addr[0]);
APP_LOG_INFO("HID Mouse example started.");

hw_simulator_init();

error_code = app_timer_create(&s_hw_simulator_timer_id, ATIMER_REPEAT,
                             hw_simulator_timer_handler);

APP_ERROR_CHECK(error_code);
error_code = app_timer_start(s_hw_simulator_timer_id, HW_SIM_UPDATE_INTERVAL, NULL);

APP_ERROR_CHECK(error_code);
services_init();
gap_params_init();
adv_params_init(true);
}
```

- **GATT Service Event Handlers**

应用程序可为每个GATT Service提供一个Event Handler，以便GATT Service将其内部的Event通知给应用程序。一个特定应用程序对某些GATT Service的Event不感兴趣，需要将该GATT Service的Event Handler函数指针赋值为NULL，不处理对应的事件。在示例程序中，Event Handler通常采用xxx_service_event_process的格式命名，它们一般被定义在user_app.c中。

- **BLE SDK Callback**

ble_stack_init()函数将BLE SDK Callback注册至SDK后，开发者可在这些Callback中处理来自SDK的BLE Event，比如广播停止、连接断开、连接参数更新等。这些Callback的实现均在目录Src\user_callback下的源文件中。更多详细介绍请参考[《GR551x开发者指南》](#)“4.3.3.2 实现BLE相关业务逻辑”。

- **Sensor Simulator**

一些GATT Profile的主要用途是传输传感器采集的数据。GR5515 Starter Kit开发板上没有集成传感器，所以BLE应用示例程序采用了Sensor Simulator来模拟传感器。GR551x SDK在SDK_Folder\components\libraries\sensorsim中提供了一个Sensor Simulator Module。

4 BLE Peripheral示例

本章主要介绍常见的BLE Peripheral应用示例。

4.1 警报通知Client应用示例

警报通知Client应用示例（Alert Notification Client）实现Alert Notification Profile（ANP）中Client角色，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_ans_c\。本示例在与Alert Notification Server设备建立连接后，可发现Alert Notification Service，并获取Server端Alert Notification，比如：来电、未接电话以及SMS/MMS等消息。

4.1.1 代码理解

在本示例与Alert Notification Server设备建立连接并成功发现Server端Alert Notification Service后，会先后调用ans_c_new_alert_notify_set()和ans_c_unread_alert_notify_set()使能Server端All New Alert Category Characteristic和All Unread Alert Category Characteristic Notification。

调用ans_c_sup_new_alert_cat_read()、ans_c_sup_unread_alert_cat_read()以及ans_c_ctrl_point_set()可获取Server端消息通知，并将其通过串口打印出来。

4.1.2 测试验证

使用Alert Notification Server设备和PC端串口工具GRUart可以测试警报通知Client示例应用。测试步骤如下：

1. 用GProgrammer下载ble_app_ans_c_fw.bin至开发板1作为Client设备。
2. 用GProgrammer下载ble_app_ans_fw.bin至开发板2作为Server设备。
3. PC端连接Client设备串口后，运行并设置GRUart。
4. 当Client设备与Server设备建立连接后，单击Client设备“OK”键获取Server端New Alert information和Unread Alert information。
5. 可在GRUart工具观察到对端Alert Notification信息。

4.2 信标应用示例

信标应用示例（Beacon）遵循Apple®的iBeacon协议，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_beacon\。

4.2.1 代码理解

iBeacon协议基于标准的低功耗蓝牙协议，其广播内容基于制造商指定广播数据类型。该广播数据定义在以下数组中：

```
static uint8_t s_adv_data_set[] =
{
    // Manufacturer specific adv data type
```

```
0x1A,  
BLE_GAP_AD_TYPE_MANU_SPECIFIC_DATA,  
// Company Identifier  
0x4C, 0x00,  
// Beacon Data Type  
0x02,  
// Data Length  
0x15,  
// UUID - Variable based on different use cases/applications  
0x01, 0x12, 0x23, 0x34,  
0x45, 0x56, 0x67, 0x78,  
0x89, 0x9a, 0xab, 0xbc,  
0xcd, 0xde, 0xef, 0xf0,  
// Major value for identifying Beacons  
0x00,  
0x01,  
// Minor value for identifying Beacons  
0x00,  
0x01,  
// The Beacon's measured RSSI at meter distance in dBm  
0xc3  
}
```

本Beacon应用在完成GAP参数和广播数据初始化之后，会启动一个周期为5秒的Timer来更新广播数据中的Major和Minor值。

4.2.2 测试验证

使用GRToolbox App可以测试Beacon应用，测试步骤如下：

1. 用GProgrammer下载`ble_app_beacon_fw.bin`至开发板。
2. 用GRToolbox App扫描周围的BLE设备，找到与GRUart中显示的蓝牙地址相同的iBeacon设备。
3. 在GRToolbox App上查看该设备广播数据中的Major和Minor值。

说明:

为方便开发者观察程序运行情况，Major和Minor值会随时间发生变化。这种变化并非iBeacon协议的规定。

4.3 血压应用示例

血压应用示例程序（Blood Pressure Sensor）实现Blood Pressure Profile（BLP）中Sensor角色，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_bps\。

本示例包括BLP规定的Sensor必须包含的两个GATT Service:

- Blood Pressure Service
- Device Information Service

还包含了一个BLP未规定的GATT Service。

- Battery Service

Blood Pressure Service支持以下三个Characteristic:

- Blood Pressure Measurement
- Blood Pressure Measurement Client Characteristic Configuration descriptor
- Blood Pressure Feature

SDK的Sensor Simulator Module模拟Blood Pressure Service和Battery Service中Characteristics值的持续变化。

4.3.1 代码理解

血压应用示例支持同时发出最多五个扩展广播（Extended Advertising）；支持最多与10个Collector同时连接并传输数据。

start_next_adv()函数负责判断是否开启一个新的广播。

- 当一个广播被开启后，start_next_adv()会被app_gap_adv_start_cb()调用；
- 当一个广播因为建立连接被停止后，start_next_adv()会被app_gap_adv_stop_cb()调用；
- 当一个连接断开时，start_next_adv()会被app_disconnect_handler()调用。

在已经建立10个连接之后，本应用不会再发出新的广播。

血压应用示例支持Just Work配对。完成配对绑定过程之后，Collector可以使能Sensor的Blood Pressure Measurement Indication。Blood Pressure Measurement Timer对应的Timeout Handler函数bps_sim_measurement()会完成Blood Pressure Measurement模拟数据的组包和发送。

4.3.2 测试验证

使用GRToolbox App可以测试血压应用示例。测试步骤如下：

1. 用GProgrammer下载ble_app_bps_fw.bin至开发板。
2. 在GRToolbox App的“例程 > BPM”模块中扫描并连接名为“Goodix_BLP”的设备。
3. 连接成功后，在BPM模块可观察到不断变化的血压数据：Systolic、Diastolic、Mean AP和Pulse Rate。

4.4 当前时间Server应用示例

当前时间应用示例（Current Time Server）实现Current Time Profile的Server角色，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_cts\。

本示例包含Current Time Service，支持以下三个Characteristic:

- Current Time
- Local Time Information
- Reference Time Information

4.4.1 代码理解

Current Time Server启动后，一个Timer会立刻开始运行。该Timer的Timeout Handler函数current_time_update()将持续调整Current Time。如果Current Time Client使能了Server的Current Time Characteristic Notification，Server会调用cts_cur_time_send()将Current Time的改变通知Client。Current Time Information Characteristic也可被Client设置。

4.4.2 测试验证

在测试验证中，此应用示例需与Current Time Client配合使用，测试验证过程参考[5.2 当前时间Client应用示例](#)。

4.5 自行车速度与踏频应用示例

自行车速度与踏频应用示例（Cycling Speed and Cadence Sensor）实现Cycling Speed and Cadence Profile（CSCP）中Sensor角色，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_cscs\。

本示例包含三个GATT service：

- Cycling Speed and Cadence Service（CSCP规定必备的GATT Service）
- Device Information Service（CSCP规定可选择实现的GATT Service）
- Battery Service（CSCP未规定的GATT Service）

CSCP支持以下六个Characteristic：

- CSC Measurement
- CSC Measurement Client Characteristic Configuration descriptor
- CSC Feature
- Sensor Location
- SC Control Point
- SC Control Point Client Characteristic Configuration Descriptor

SDK的Sensor Simulator Module模拟Cycling Speed and Cadence Service和Battery Service中Characteristic值的持续变化。

4.5.1 代码理解

Cycling Speed and Cadence Collector使能Sensor的CSC Measurement Characteristic Notification后，CSC Measurement timer会开始运行，定时执行其timeout handler函数csc_meas_timeout_handler()模拟产生持续变化的CSC Measurement characteristic value：

- Cumulative Wheel Revolutions
- Last Wheel Event Time

- Cumulative Crank Revolutions
- Last Crank Event Time

然后，调用Cycling Speed and Cadence Service API: `csc_measurement_send()`来发送CSC Measurement characteristic value至Cycling Speed and Cadence Collector。

4.5.2 测试验证

使用GRToolbox App可以测试自行车速度与踏频应用示例。测试步骤如下：

1. 用GProgrammer下载`ble_app_cscs_fw.bin`至开发板。
2. 在GRToolbox App的“例程 > CSC”模块中扫描并连接名为“Goodix_CSCS”的设备。
3. 连接成功后，在CSC模块可观察到自行车速度与踏频的数据变化

4.6 血糖应用示例

血糖应用示例（Glucose Sensor）实现Glucose Profile（GLP）中Sensor角色，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_gls\。

本示例包括GLP规定的Sensor必须包含的两个GATT Service：

- Glucose Service
- Device Information Service

还包含一个GLP未规定的GATT Service

- Battery Service

Glucose Service支持以下五个Characteristic：

- Glucose Measurement
- Glucose Measurement Client Characteristic Configuration descriptor
- Glucose Feature
- Record Access Control Point
- Record Access Control Point Client Characteristic Configuration Descriptor

SDK的Sensor Simulator Module模拟Battery Service中Characteristic值的持续变化。Glucose Service中Glucose Measurement Characteristic Value是通过开发板“OK”键控制其创建的。

4.6.1 代码理解

在Glucose Collector使能Sensor的Glucose Measurement Characteristic Notification后，如果Record Access Control Point Characteristic被Collector写入Glucose Database相关指令，会调用`gls_meas_val_send()`将Glucose Measurement Value发送至Collector，这些指令包括以下五种：

- Sequence Number

- Base Time & Time offset
- Glucose Concentration
- Sample Type
- Sample Location

单击开发板“OK”键后，key_click_event_handler()会调用glucose_measurement_excute()创建一条Glucose Sample Data并保存在Database中。

4.6.2 测试验证

使用GRToolbox App可以测试血糖应用示例。测试步骤如下：

1. 用GProgrammer下载ble_app_gls_fw.bin至开发板。
2. 用GRToolbox扫描到名为“Goodix_GLS”的设备，并选中“连接”。
3. 单击开发板“OK”键创建Glucose Measurement Characteristic Value。
4. 使能Glucose Measurement Notification后，向Record Access Control Point写入Database获取指令，即可收到Glucose Measurement Characteristic Value。

4.7 心率应用示例

心率应用示例（Heart Rate Sensor）实现Heart Rate Profile（HRP）中Sensor角色，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_hrs\。

本示例包括HRP规定的Sensor必须包含的两个GATT Service：

- Heart Rate Service
- Device Information Service

还包含一个HRP未规定的GATT Service

- Battery Service

Heart Rate Service支持以下四个Characteristic：

- Heart Rate Measurement
- Heart Rate Measurement Client Characteristic Configuration descriptor
- Body Sensor Location
- Heart Rate Control Point

SDK的Sensor Simulator Module模拟Heart Rate Service和Battery Service中Characteristic值的持续变化。

4.7.1 代码理解

Collector使能Sensor的Heart Rate Measurement Characteristic Notification后，Heart Rate Measurement Timer和RR Interval Measurement Timer开始运行。Heart Rate Measurement Timer的Timeout Handler函数heart_rate_meas_timeout_handler()模拟产生持续变化的Heart Rate Measurement characteristic value:

- Heart Rate
- Energy Expended
- Sensor Contact Detected

heart_rate_meas_timeout_handler()调用hrs_heart_rate_measurement_send()来发送Notification。Collector可接收到呈三角波变化的Heart Rate。Sensor Contact的状态不停地在Detected和Undetected之间变化。

RR Interval Measurement Timer 的Timeout Handler函数rr_interval_timeout_handler()模拟产生持续变化的RR Interval。rr_interval_timeout_handler()不发送Notification，只将测量到的RR Interval交给Heart Rate Service存储。

4.7.2 测试验证

使用GRToolbox App可以测试心率应用示例。测试步骤如下:

1. 用GProgrammer下载ble_app_hrs_fw.bin至开发板。
2. 在GRToolbox App的“例程 > HRS”模块中扫描并连接名为“Goodix_HRM”的设备。
3. 连接成功后，在HRS模块可观察到Heart Rate值和曲线、RR Interval和Location Finger。

4.8 心率多连接应用示例

心率多连接应用示例（Heart Rate Sensor Multi Link）对Heart Rate Profile（HRP）中Sensor角色的实现与心率应用示例相同，但可被多个Heart Rate Collector连接获取心率数据，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_hrs_multi_link\。

4.8.1 代码理解

本应用启动后会开启一个广播，直至当其被Heart Rate Collector连接，然后调用multi_advertising_start()再次开启一个广播。当本应用建立链路达到最大值（可配置）时，将不再开启新的广播。

4.8.2 测试验证

使用GRToolbox App可以测试心率多连接应用示例。测试步骤如下:

1. 用GProgrammer下载ble_app_hrs_multi_link_fw.bin至开发板。
2. 在GRToolbox App的“例程 > HRS”模块中扫描并连接名为“GR_HRM_MLINK”的设备。
3. 连接成功后，在HRS模块可观察到Heart Rate变化曲线和Contact Detected状态变化。
4. 用另一部手机重复第 2、3 步骤，建立多条链路，均可观察到Heart Rate变化曲线和Contact Detected状态变化。

4.9 电话警报通知Client应用示例

电话警报通知Client应用示例（Phone Alert Status Client）实现Phone Alert Status Profile（PASP）中Client角色，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_pass_c\。本示例在与Phone Alert Status Server设备建立连接、发现Phone Alert Status Service后，可获取来自Server设备的警报状态和铃声设置。

4.9.1 代码理解

当本应用与Phone Alert Status Server设备建立连接并成功发现Server端Alert Notification Service后，先后调用pass_c_alert_status_notify_set()和pass_c_ringer_set_notify_set()使能Alert Status和Ring Set的Notification。调用pass_c_ctrl_point_set()可对Server端的铃声进行设置。

4.9.2 测试验证

与Phone Alert Status Server设备配合测试Phone Alert Status Client应用示例。测试步骤如下：

1. 用GProgrammer下载ble_app_pass_c_fw.bin至开发板1作为Client设备。
2. 用GProgrammer下载ble_app_pass_fw.bin至开发板2作为Server设备。
3. 将Client设备串口连接至PC端，打开并设置GRUart。
4. 当Client设备与Server设备建立连接后，单击Client设备“OK”键可设置Server设备为“Silent”模式；双击可设置Server设备为“Mute Once”模式；长按可取消Server设备的“Silent”模式。
5. 当Client设备收到来自Server设备的“Alert Status”信息时，将通过串口打印出来。可在GRUart观察到具体信息，包括Ringer State、Vibrate State和Display State。

4.10 接近应用示例

接近应用示例（Proximity Reporter）实现Proximity Profile（PXP）中Reporter角色，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_pxp_reporter\。

本应用包括PXP规定Reporter必须包含的三个GATT Service:

- Link Loss Service
- Immediate Alert Service
- TX Power Service

4.10.1 代码理解

根据PXP Spec的要求，本应用在启动后的30秒内会发出Fast Connection Advertising；在30秒后会发出Reduced Power Advertising。user_gap_callback.c中的app_gap_op_cmp_evt_cb()会调用app_adv_stopped_handler()来启动Reduced Power Advertising。

Monitor向Reporter发起连接请求后，会收到来自Reporter的配对请求。user_sm_callback.c中的app_sec_rcv_enc_req_cb()负责处理来自BLE Stack的配对和加密请求。在该函数的TK_REQ处理分支中，语句tk = 123456设置了配对Pin Code为“123456”。

4.10.2 测试验证

1. 用GProgrammer下载ble_app_pxp_reporter_fw.bin至开发板。
2. 将Reporter设备串口连接至PC端，打开并设置GRUart。
3. 用GRToolbox App扫描到名为“Goodix_PXP”的设备，并发起连接请求。
4. 在手机Pin Code中输入“123456”完成配对和绑定。
5. 将手机远离开发板，直到手机显示与开发板的连接已断开。
6. 在GRUart的Log窗口中将会观察到Link Loss Alert Message。

4.11 跑速与步频应用示例

跑速与步频应用示例（Running Speed and Cadence Sensor）实现Running Speed and Cadence Profile（RSCP）中Sensor角色，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_rscs\。

本示例包括以下服务：

- Running Speed and Cadence Service（RSCP规定的Sensor必备GATT Service）
- Device Information Service（RSCP规定可选择实现的GATT Service）
- Battery Service（RSCP未规定的GATT Service）

Running Speed and Cadence Service支持以下六个Characteristic：

- RSC Measurement
- RSC Measurement Client Characteristic Configuration descriptor
- RSC Feature
- Sensor Location
- SC Control Point
- SC Control Point Client Characteristic Configuration Descriptor

SDK的Sensor Simulator Module模拟Running Speed and Cadence Service和Battery Service中Characteristic值的持续变化。

4.11.1 代码理解

Running Speed and Cadence Collector使能Sensor的RSC Measurement Characteristic Notification后，RSC Measurement Timer会开始运行，其定时调用Timeout Handler函数rsc_meas_update()模拟产生持续变化的RSC Measurement Characteristic value:

- Instantaneous Speed
- Instantaneous Cadence
- Instantaneous Stride Length
- Total Distance
- Walking or Running Status

然后，调用Running Speed and Cadence Service API: rsc_measurement_send()来发送RSC Measurement Characteristic value至Running Speed and Cadence Collector。

4.11.2 测试验证

使用GRToolbox App可以测试跑速与步频应用示例。测试步骤如下:

1. 用GProgrammer下载ble_app_rscs_fw.bin至开发板。
2. 在GRToolbox App的“例程 > RSC”模块中扫描并连接名为“Goodix_RSCS”的设备。
3. 连接成功后，在RSC模块可观察到跑速与步频的数据变化。

4.12 体温计应用示例

体温计应用示例（Thermometer Sensor）实现Health Thermometer Profile（HTP）中Thermometer角色，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_hts\。

本示例包括HTP规定的Thermometer必须包含的两个GATT Service:

- Health Thermometer Service
- Device Information Service

还包含一个HTP未规定的GATT Service

- Battery Service

Health Thermometer Service支持以下八个Characteristic:

- Temperature Measurement
- Temperature Measurement Client Characteristic Configuration descriptor
- Temperature Type
- Intermediate Temperature
- Intermediate Temperature Client Characteristic Configuration descriptor

- Measurement Interval
- Measurement Interval Client Characteristic Configuration descriptor
- Measurement Interval Valid Range descriptor

SDK的Sensor Simulator Module模拟Health Thermometer Service和Battery Service中Characteristic值的持续变化。

4.12.1 代码理解

`gap_params_init()`使能了配对，并且调用`ble_gap_privacy_params_set()`开启了隐私模式，即每900秒会更新一次随机地址。本应用会持续发出Advertising，直到有Collector发起连接。

当与Collector连接后，Battery Level Timer就会启动，开始模拟Battery Level变化。直到Collector使能了Thermometer的Temperature Measurement Indication或Intermediate Temperature Measurement Notification后，Temperature Level Timer才会启动，之后，`hts_meas_timeout_handler()`会被周期性调用以模拟Temperature Level的变化。

当连接断开后，`app_disconnect_handler()`会停止Battery Level Timer和Temperature Level Timer。

4.12.2 测试验证

使用GRToolbox App可以测试体温计应用示例。测试步骤如下：

1. 用GProgrammer下载`ble_app_hts_fw.bin`至开发板。
2. 在GRToolbox App的“例程 > HTS”模块中扫描并连接名为“Goodix-HTS”的设备。
3. 连接成功后，在HTS模块可观察到Temperature不断变化，Time以2s为间隔增加。

4.13 吞吐率Server应用示例

吞吐率应用示例（Throughput Server）实现Goodix Throughput Profile规定的Server，其源代码位于`SDK_Folder\projects\ble\ble_peripheral\ble_app_throughput\`。本应用包括Goodix自定义的Throughput Service。它演示了测试BLE数据传输的吞吐率。本应用不支持通过本地物理串口输出吞吐率以及吞吐率相关参数。开发者可以使用GRToolbox App的Throughput功能模块，查看实时的吞吐率和配置吞吐率相关参数。

4.13.1 代码理解

`user_app.c`中的`gap_params_init()`函数设置了影响吞吐率的初始的Preferred Data Length，MTU和PHY参数。这些参数在连接建立后，也可以使用GRToolbox App进行修改。`throughput.c`中的`ths_event_process()`函数会在`THS_EVT_SETTINGS_CHANGED`处理分支中响应GRToolbox App对这些参数的改变。

`throughput.c`中的`ths_event_process()`函数会在`THS_EVT_DATA_RECEIVED`处理分支中统计接收到的数据总量。发送数据是通过在`ths_event_process()`函数中处理`THS_EVT_TOGGLE_SET`和`THS_EVT_DATA_SENT`完成的。如果`THS_EVT_TOGGLE_SET`所带的参数为`THS_TOGGLE_STATE_ON`，则调用`ths_send_data()`开

始发送第一包数据。当收到THS_EVT_DATA_SENT则开始发送下一包数据。如果收到所带参数为THS_TOGGLE_STATE_OFF的THS_EVT_TOGGLE_SET，则停止发送数据包。

4.13.2 测试验证

使用GRToolbox App可以测试Throughput应用示例。测试步骤如下：

1. 用GProgrammer下载ble_app_throughput_fw.bin至开发板。
2. 用GRToolbox App扫描到名为“Goodix_THS”的设备，并发起连接请求。
3. 在GRToolbox App的“应用 > THS”功能模块中，可以进行Data Length、MTU、PHY、TX Power和CI参数配置。
4. 在GRToolbox App的THS模块中，使用Toggle On使开发板发送数据。
5. 在GRToolbox App的THS模块中，可以查看到数据从开发板传输到手机的实时吞吐率曲线，使用“Toggle Off”使开发板停止发送数据。

在GRToolbox App的THS模块中，将传输模式设置为Master Write模式，App开始向开发板传输数据，同时显示实时吞吐率曲线。

4.14 ANCS Client应用示例

ANCS Client应用示例（ANCS Client）实现Apple Notification Center Service的Client，演示了GR551x平台通过ANCS实现与IOS设备的通讯和交互。其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_ancs_c。具体使用方法请参考[《GR551x ANCS Profile示例手册》](#)。

4.15 FreeRTOS模板应用示例

FreeRTOS模板应用示例（FreeRTOS Template）采用FreeRTOS进行多任务调度，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_template_freertos。具体使用方法请参考[《GR551x FreeRTOS示例手册》](#)。

4.16 BLE鼠标应用示例

BLE鼠标应用示例（HID Mouse）实现符合HID Over GATT Profile的鼠标示例。其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_hids_mouse。具体使用方法请参考[《GR551x鼠标示例手册》](#)。

4.17 功耗测试应用示例

功耗测试应用示例（Power Consumption）实现Goodix自定义的Power Consumption Profile，可以测试GR5515 Starter Kit开发板的功耗。其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_pcs。具体使用方法请参考[《GR551x Power Consumption Profile示例手册》](#)。

4.18 模板应用示例

模板应用示例（Template）实现包含自定义Goodix Sample Service的Server角色，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_template。具体使用方法请参考[《应用及自定义GR551x Sample Service》](#)。

4.19 串口Server应用示例

串口Server应用示例（UART Server）实现Goodix Serial Port Profile（SPP），可以帮助用户快速开发基于数据透传的应用。其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_uart。具体使用方法请参考[《GR551x Serial Port Profile示例手册》](#)。

4.20 AMS Client应用示例

AMS Client应用示例（AMS Client）实现Apple Media Service Profile的Client角色，演示了GR551x平台通过AMS实现与iOS设备媒体的通讯和交互。其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_ams_c。具体使用方法请参考[《GR551x AMS Profile示例手册》](#)。

4.21 微信AirSync应用示例

GR551x微信AirSync应用示例（GR551x WeChat）实现WeChat Service的Server，演示了GR551x平台基于AirSync协议通过WeChat Service与微信客户端的通讯和交互，其源代码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_wechat。具体使用方法请参考[《GR551x微信AirSync应用说明》](#)。

5 BLE Central示例

5.1 警报通知Server应用示例

警报通知Server应用示例（Alert Notification Server）实现Alert Notification Profile（ANP）中Server角色，其源代码位于SDK_Folder\projects\ble\ble_central\ble_app_ans\。


本示例实现的Alert Notification Service支持以下五个Characteristic：

- Supported New Alert Category
- New Alert
- Supported Unread Alert Category
- Unread Alert Status
- Alert Notification Control Point

5.1.1 代码理解

本应用启动后，可通过开发板“OK”键模拟产生New Email和Missed Call等Unread Alert，分别调用ans_unread_alert_send()和ans_new_alert_send()将Unread Alert和New Alert信息发送至Alert Notification Client，Alert Notification数据包含以下内容：

- Alert ID
- Alert Number

 说明：

串口输出的Log中并没有包含具体的Alert ID和Alert Number。

5.1.2 测试验证

测试验证步骤参考[4.1 警报通知Client应用示例](#)。

5.2 当前时间Client应用示例

当前时间Client应用示例（Current Time Client）程序实现Current Time Profile的Client角色，其源代码位于SDK_Folder\projects\ble\ble_central\ble_app_cts_c\。本示例在与Current Time Server设备建立连接并发现Current Time Service后，可对Current Time Server设备的当前时间相关信息进行设置和获取。

5.2.1 代码理解

当本应用与Current Time Server设备建立连接并成功发现对端Current Time Service后，会调用cts_c_cur_time_notify_set()使能Server端Current Time characteristic Notification。通过开发板“OK”键可控制调用cts_c_cur_time_read()、cts_c_loc_time_info_read()和cts_c_ref_time_info_read()以获取对端Current time、Local time information和Reference time information，并通过串口打印出相关信息。

5.2.2 测试验证

使用Current Time Server设备和PC端串口工具GRUart可以测试Current Time Client应用示例。测试步骤如下：

1. 用GProgrammer下载`ble_app_cts_c_fw.bin`至开发板1作为Client设备。
2. 用GProgrammer下载`ble_app_cts_fw.bin`至开发板2作为Server设备。
3. 将Client设备串口连接至PC端，打开并设置GRUart。
4. 当Client设备与Server设备建立连接后，先后单击Client设备“OK”键可依次获取到对端Current time, Local time information和Reference time information。
5. 可在GRUart观察到Current time信息。

5.3 电话警报通知Server应用示例

电话警报通知Server应用示例（Phone Alert Status Server）实现Phone Alert Status Profile（PASP）中Server角色，其源代码位于`SDK_Folder\projects\ble\ble_central\ble_app_pass\`。

本示例实现的Phone Alert Status Service支持以下三个Characteristic：

- Alert Status
- Ringer Setting
- Ringer Control Point

5.3.1 代码理解

在Phone Alert Status Client使能Server的Alert Status和Ringer Setting characteristic Notification后，如果Ringer mode和Alert Status设置发生了改变，会分别调用`pass_ringer_setting_set()`和`pass_alert_status_set()`将当前的Ringer Setting和Alert Status信息发送至对Phone Alert Status Client。

5.3.2 测试验证

测试验证步骤请参考[4.9 电话警报通知Client应用示例](#)。

5.4 心率Client应用

心率Client应用示例（Heart Rate Collector）实现Heart Rate Profile（HRP）中Collector角色，当它与Heart Rate Sensor建立连接后，可接收来自Sensor的心率检测及相关数据信息。其源代码位于`SDK_Folder\projects\ble\ble_central\ble_app_hrs_c\`。本示例支持了与Heart Rate Service、Device Information Service和Battery Service的交互。

5.4.1 代码理解

本应用启动后开启扫描，通过`hrs_srvc_uuid_find()`判定扫描到的广播数据是否含有Heart Rate Service UUID，若有则视为目标设备，并发起连接请求。

当建立连接之后，可通过开发板“OK”键交替使能或关闭Heart Rate Measurement Characteristic Notification和Battery Level Characteristic Notification。

当Heart Rate Measurement Characteristic Notification和Battery Level Characteristic Notification被使能后，接收到心率数据和电池电量数据分别通过hrs_c_evt_process()中HRS_C_EVT_HR_MEAS_VAL_RECEIVE事件和bas_c_evt_process()中BAS_C_EVT_BAT_LEVE_RECEIVE事件上报通知，并通过串口打印出相应的数据。

5.4.2 测试验证

1. 用GProgrammer下载ble_app_hrs_c_fw.bin至开发板。
2. 将Collector设备串口连接至PC端，打开并设置GRUart。
3. 串口输出启动、扫描、连接等日志。
4. 单击开发板“OK”键使能HRS notification、BAS notification。
5. 在GRUart显示来自Sensor的心率和电池电量的数据。

5.5 跑速与步频Client应用

跑速与步频Client应用示例（Running Speed and Cadence Collector）实现Running Speed and Cadence Profile（RSCP）中Collector角色，当它与Running Speed and Cadence Sensor建立连接后，可接收来自Sensor的跑速检测及相关数据信息。其源代码位于SDK_Folder\projects\ble\ble_central\ble_app_rscs_c\。本示例支持了对Running Speed and Cadence Service、Device Information Service和Battery Service的交互。

5.5.1 代码理解

本应用启动后开启扫描，通过rscs_srvc_uuid_find()判定扫描到的广播数据是否含有Running Speed and Cadence Service UUID，若有则视为目标设备，并发起连接请求。

当建立连接之后，可通过单击开发板“OK”键交替使能或关闭RSC Measurement Characteristic Notification和Battery Level Characteristic Notification。

当RSC Measurement Characteristic Notification和Battery Level Characteristic Notification被使能后，接收到跑速与步频数据和电池电量数据分别通过rscs_c_evt_process()中RSCS_C_EVT_RSC_MEAS_VAL_RECEIVE事件和bas_c_evt_process()中BAS_C_EVT_BAT_LEVE_RECEIVE事件上报通知，并通过串口打印出相应的数据。

5.5.2 测试验证

1. 用GProgrammer下载ble_app_rscs_c_fw.bin至开发板。
2. 将PC端连接至Collector设备串口后，运行并设置GRUart。
3. 串口输出启动、扫描、连接等日志。
4. 单击开发板“OK”键使能RSCS和BAS notification。
5. 在GRUart显示来自对端的跑速、步频和电池电量的数据。

5.6 串口Client应用

串口Client应用示例（UART Initiator）实现Goodix UART Profile中Initiator角色，其源代码位于SDK_Folder\projects\ble\ble_central\ble_app_uart_c\。当它与串口Server端（Acceptor角色）建立连接后，可进行双向数据传输，一方面接收本地串口输入数据通过BLE传输至对端；另一方面接收来自对端BLE数据传输至本地串口。

5.6.1 代码理解

本地物理串口和双向环形缓存的初始化工作分别在app_periph_init()和ble_init_cmp_callback()中完成。

在hal_uart_rx_cplt_callback()和hal_uart_dma_rx_tfr_callback()中接收到来自本地串口的数据，调用uart_to_ble_push()函数将串口数据写入UART至BLE方向的环形缓存（s_uart_rx_ring_buffer）。

在gus_c_evt_process()的GUS_C_EVT_PEER_DATA_RECEIVE事件中接收到来自对端BLE数据，调用ble_to_uart_push()函数将BLE数据写入BLE至UART方向的环形缓存（s_ble_rx_ring_buffer）。

双向传输的调度由uart_transport_schedule()函数完成，其在main.c的主循环中被调用。

5.6.2 测试验证

1. 用GProgrammer下载ble_app_gus_c_fw.bin至开发板。
2. 将Client设备串口连接至PC端，打开并设置GRUart。
3. 串口输出连接、使能对端CCCD等日志。
4. 通过GRUart写入数据至对端开发板，对端会接收相应的数据，并通过其串口输出。
5. 在GRUart显示来自对端的数据。

6 其它示例

6.1 BLE Basic示例

BLE基础示例介绍BLE协议栈的应用示例，其源代码位于SDK_Folder\projects\ble\ble_basic_example。具体使用方法，请参考[《GR551x BLE Stack用户手册》](#)。

6.2 BLE Multi-Role示例

BLE Multi Role应用示例实现Central角色和Peripheral角色，可以同时与多个设备保持连接。其源代码位于SDK_Folder\projects\ble\ble_multi_role\ble_app_hrs_rscs_relay。具体使用方法，请参考[《GR551x HRS RSCS Relay示例手册》](#)。

6.3 DFU示例

DFU应用示例演示了如何通过GRTtoolbox（Android）App对GR551x进行固件升级。其源代码位于SDK_Folder\projects\ble\dfu。具体使用方法，请参考[《GR551x OTA示例手册》](#)。

6.4 DTM示例

DTM应用示例演示了如何使用蓝牙测试仪对GR551x的射频性能进行DTM测试。其源代码位于SDK_Folder\projects\ble\dtm。具体使用方法，请参考[《GR551x DTM测试指南》](#)。