GR551x HRS RSCS Relay示例手册

版本： **1.8**

发布日期： **2020-12-15**

深圳市汇顶科技股份有限公司

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

本文档介绍如何使用和验证GR551x SDK中的HRS RSCS Relay示例，旨在帮助用户快速进行二次开发。

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
- 开发爱好者
- 文档工程师

版本说明

本文档为第5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	更新了文档中页脚的时间
1.5	2020-05-30	调整了“4 应用详解”章节中代码的缩进格式
1.6	2020-06-30	基于SDK刷新版本
1.7	2020-11-09	更新“3.4 测试验证”章节图3-2
1.8	2020-12-15	更新GRToolbox软件界面截图

版权所有 © 2020 深圳市汇顶科技股份有限公司

目录

前言 I

1 简介 1

2 Profile概述 2

3 初次运行 4

3.1 准备工作 4
3.2 硬件连接 4
3.3 下载固件 5
3.4 测试验证 5

4 应用详解 10

4.1 工程目录 10
4.2 实现流程及代码介绍 10

版权所有 © 2020 深圳市汇顶科技股份有限公司

1 简介

HRS RSCS Relay（Heart Rate Sensor & Running Speed and Cadence Sensor Relay）示例演示了如何将GR551x芯片应用于多角色（Peripheral和Central）、多连接的场景，实现了心率传感器和跑速与步频传感器中继设备功能。HRS RSCS Relay设备可同时作为采集器和传感器。

- 采集器

  作为GATT客户端，接收来自心率传感器、跑速与步频传感器的测量数据。

- 传感器

  作为GATT服务端，将接收测量数据发送至其他采集器设备（例如手机APP “GRToolbox”）。

本文将介绍如何使用和验证GR551x SDK中的HRS RSCS Relay示例。在进行操作前，建议参考以下文档。

表 1-1 文档参考

名称	描述
应用及自定义GR551x Sample Service	介绍实现自定义Service的相关知识
GR551x开发者指南	GR551x软硬件介绍、快速使用及资源总览
Bluetooth Core Spec v5.1	Bluetooth官方标准核心规范5.1： https://www.bluetooth.com/specifications/bluetooth-core-specification/
Bluetooth GATT Spec	Bluetooth Profile和Service的详细信息： www.bluetooth.com/specifications/gatt
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/

版权所有 © 2020 深圳市汇顶科技股份有限公司

2 Profile概述

HRS RSCS Relay示例实现的Profile如下：

- 标准Profile：Heart Rate Profile、Running Speed and Cadence Profile，由蓝牙技术联盟（Bluetooth SIG）定义。
- 自定义Profile：Goodix HRS RSCS Relay Control Point Profile，由Goodix自定义。

以GRToolbox作为HRS RSCS Relay的采集器为例，应用场景如图 2-1所示。

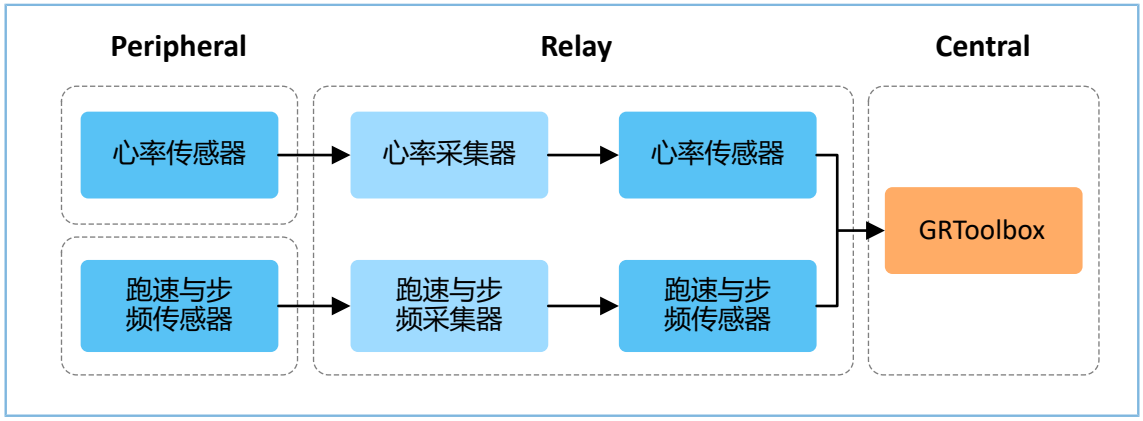

图 2-1 应用场景图

作为采集器，HRS RSCS Relay注册以下Profile：

- Heart Rate Client Profile：接收心率传感器的测量数据。
- Running Speed and Cadence Client Profile：接收跑速与步频传感器的测量数据。

作为传感器，HRS RSCS Relay注册以下Profile：

- Heart Rate Server Profile：将接收心率传感器的测量数据中继至GRToolbox。
- Running Speed and Cadence Server Profile：将接收跑速与步频传感器的测量数据中继至GRToolbox。
- Goodix HRS RSCS Relay Control Point Profile：接收来自GRToolbox的控制指令，并返回执行结果。

Goodix HRS RSCS Relay Control Point Profile 包括HRRCPS（HRS RSCS Relay Control Point Service），其专用128位UUID为A6ED0601-D344-460A-8075-B9E8EC90D71B。

HRRCPS包含两个特征：

- HRR Control Point Characteristic：接收Relay采集器的控制指令。
- HRR Control Point Response Characteristic：回应指令执行结果至Relay采集器。

Characteristic的具体描述如下表所示：

表 2-1 HRRCPS Characteristic

Characteristic	UUID	Type	Support	Security	Properties
HRR Control Point	A6ED0602-D344-460A-8075-B9E8EC90D71B	128 bits	Mandatory	None	Write

版权所有 © 2020 深圳市汇顶科技股份有限公司

Characteristic	UUID	Type	Support	Security	Properties
HRR Control Point Response	A6ED0603-D344-460A-8075-B9E8EC90D71B	128 bits	Mandatory	None	Indicate

版权所有 © 2020 深圳市汇顶科技股份有限公司

3 初次运行

本章介绍如何快速验证GR551x SDK中的HRS RSCS Relay示例。

说明:

SDK_Folder为GR551x SDK的根目录。

3.1 准备工作

运行应用工程示例之前，请完成以下准备工作。

- 硬件准备

表 3-1 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（共计3块）
数据线	Micro USB 2.0数据线

- 软件准备

表 3-2 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，下载网址：www.keil.com/download/product/
GRToolbox（Android）	GR551x BLE调试工具，位于SDK_Folder\tools\GRToolbox
GProgrammer（Windows）	GR551x Programming工具，位于SDK_Folder\tools\GProgrammer

3.2 硬件连接

验证HRS RSCS Relay示例需使用三块开发板，分别作为Relay设备、HRS传感器设备和RSCS传感器设备。三块开发板之间通过BLE无线连接。

GR5515 Starter Kit开发板与计算机之间使用Micro USB 2.0 数据线进行连接，如图 3-1所示。

版权所有 © 2020 深圳市汇顶科技股份有限公司

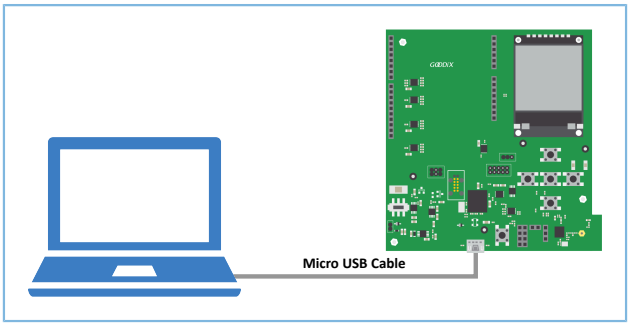

图 3-1 硬件连接示意图

3.3 下载固件

请依次下载*ble_app_hrs_rscs_relay_fw.bin*固件至Relay开发板，*ble_app_hrs_fw.bin*固件至HRS传感器开发板，*ble_app_rscs_fw.bin*固件至RSCS传感器开发板。

下载固件至开发板的具体操作方法，请参考《GProgrammer用户手册》。

说明:

- *ble_app_hrs_rscs_relay_fw.bin*位于`SDK_Folder\projects\ble\ble_multi_role\ble_app_hrs_rscs_relay\build`。
- *ble_app_hrs_fw.bin*位于`SDK_Folder\projects\ble\ble_peripheral\ble_app_hrs\build`。
- *ble_app_rscs_fw.bin*位于`SDK_Folder\projects\ble\ble_peripheral\ble_app_rscs\build`。

其中SDK_Folder为GR551x SDK的根目录。

3.4 测试验证

HRS RSCS Relay设备、HRS传感器设备及RSCS传感器设备准备完毕，即可进行HRS RSCS Relay示例的测试验证，具体步骤如下：

1. 扫描HRS RSCS Relay设备

   打开GRToolbox，选择“应用 > RELAY”。

版权所有 © 2020 深圳市汇顶科技股份有限公司

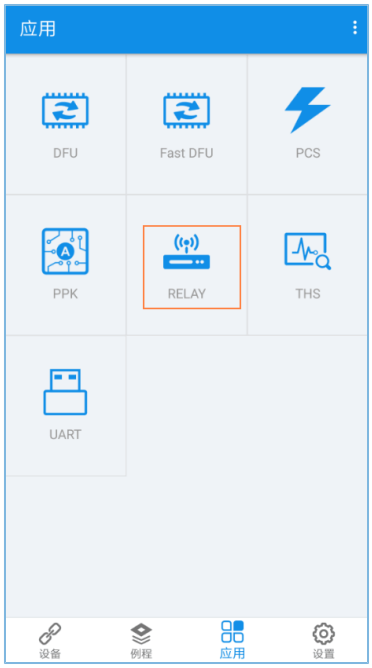

图 3-2 选择“RELAY”

手机扫描发现广播名为“Goodix_HRS_RSCS_RELAY”的设备（广播名可在*user_app.c*文件中修改），如图 3-3所示。

图 3-3 在手机端发现Goodix_HRS_RSCS_RELAY

说明:

设备全名若超出14个字符，名称中间将以省略号显示。

版权所有 © 2020 深圳市汇顶科技股份有限公司

2. 连接HRS RSCS Relay设备

选中并连接“Goodix_HRS_RSCS_RELAY”，进入HRS RSCS RELAY界面。

图 3-4 HRS RSCS RELAY界面

3. 连接传感器设备

点击连接 🔗 按钮，控制HRS RSCS Relay设备扫描、连接HRS和RSC传感器设备。Relay设备连接成功后如下图所示。

版权所有 © 2020 深圳市汇顶科技股份有限公司

图 3-5 连接传感器设备

4. 开启传感器通知

使用通知 按钮，控制HRS RSCS Relay设备通知HRS和RSC传感器上报采集的数据。

完成后即可接收HRS RSCS Relay设备中继出的心率、跑速与步频相关信息。

图 3-6 开启HRS传感器通知

版权所有 © 2020 深圳市汇顶科技股份有限公司

5. 读取HRS传感器位置

点击“获取”按钮，控制HRS RSCS Relay设备读取HRS传感器位置信息。

图 3-7 读取HRS传感器位置

若实际情况符合上述说明，则HRS RSCS Relay应用示例运行成功。

版权所有 © 2020 深圳市汇顶科技股份有限公司

4 应用详解

本章主要介绍HRS RSCS Relay示例的工程目录、实现流程以及主要代码说明。

4.1 工程目录

HRS RSCS Relay示例的源代码和工程文件位于`SDK_Folder\projects\ble\ble_multi_role\ble_app_hrs_rscs_relay`，其中工程文件位于Keil_5文件夹。

双击工程文件*ble_app_hrs_rscs_relay.uvprojx*，在Keil中查看HRS RSCS Relay示例的ble_app_hrs_rscs_relay工程目录结构，相关文件说明如表 4-1 所示。

表 4-1 ble_app_hrs_rscs_relay文件说明

Group	文件	描述
gr_profiles	ble_prf_utils.c	Profile相关操作工具
	hrs.c	Heart Rate Service实现
	hrs_c.c	Heart Rate Service客户端实现
	rscs.c	Running Speed and Cadence Service实现
	rscs_c.c	Running Speed and Cadence Service客户端实现
	hrrcps.c	Heart Rate &Running Speed and Cadence Service实现
user_callback	user_gap_callback.c	获取扫描、连接和断连事件
user_platform	user_periph_setup.c	App log、设备地址和电源管理模式的配置
user_app	main.c	main()入口函数
	user_app.c	HRS RSCS Relay应用Profile注册及逻辑处理

4.2 实现流程及代码介绍

HRS RSCS Relay示例开始运行后，将依次完成初始化外设和BLE协议栈、添加Profiles、开启广播以及等待连接。

说明:

HRS RSCS Relay示例的主要逻辑代码位于：

- Keil工程目录树中，user_app下的*user_app.c*文件。
- Keil工程目录树中，user_callback下的*user_gap_callback.c*文件

HRS RSCS Relay示例经过GRToolbox扫描、连接之后，实现流程如图 4-1所示：

版权所有 © 2020 深圳市汇顶科技股份有限公司

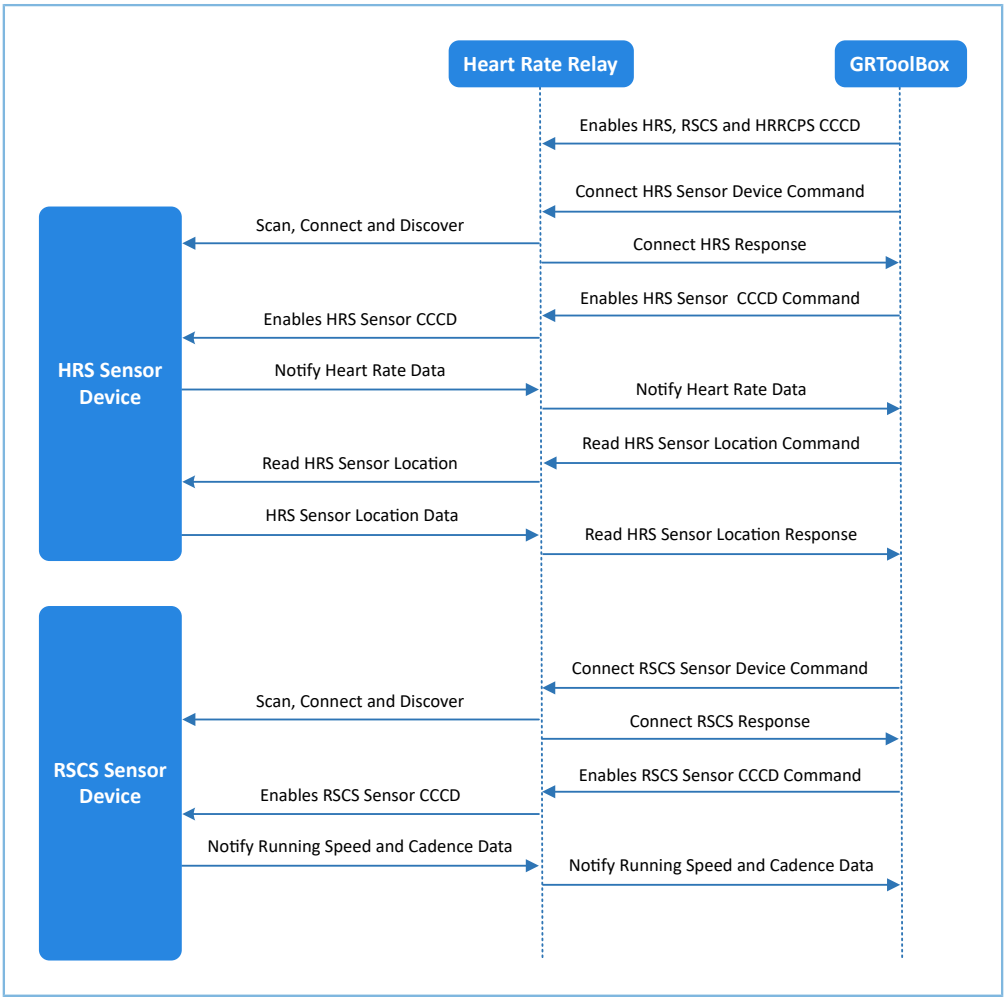

图 4-1 实现流程图

下文以HRS传感器为例，详细介绍GRToolbox、中继设备以及心率传感器之间的交互过程及主要代码。

- 接收来自GRToolbox的指令

  当HRR Control Point Characteristic Value接收到GRToolbox的控制指令数据时，会解析出相应的事件上报至应用层，并执行相应的指令，代码片段如下：

```
static void hrrcps_evt_process(hrrcps_evt_t *p_evt)
{
    sdk_err_t   error_code;

    if (p_evt->conn_idx == s_conn_idx_collector)
    {
        switch (p_evt->evt_type)
        {
            case HRRCPS_EVT_CTRL_PT_IND_ENABLE:
                APP_LOG_DEBUG("HRR Control Point Indication is enabled.");
                break;
            case HRRCPS_EVT_CTRL_PT_IND_DISABLE:
                APP_LOG_DEBUG("HRR Control Point Indication is disabled.");
                break;
```

版权所有 © 2020 深圳市汇顶科技股份有限公司

```
            case HRRCPS_EVT_SCAN_HRS:
                if (NO_ACTIVE_STATE != g_hrs_active_state)
                {
                    hrrcps_op_error_handler(HRRCPS_CTRL_PT_SCAN_HRS);
                }
                error_code = ble_gap_scan_start();
                if (error_code != SDK_SUCCESS)
                {
                    hrrcps_op_error_handler(HRRCPS_CTRL_PT_SCAN_HRS);
                }
                g_hrs_active_state = SCAN_DEV_STATE;
                APP_LOG_DEBUG("Start scanning, target device: HRS.");
                break;
            case HRRCPS_EVT_SCAN_RSCS:
                if (NO_ACTIVE_STATE != g_rscs_active_state)
                {
                    hrrcps_op_error_handler(HRRCPS_CTRL_PT_SCAN_RSCS);
                }
                error_code = ble_gap_scan_start();
                if (error_code != SDK_SUCCESS)
                {
                    hrrcps_op_error_handler(HRRCPS_CTRL_PT_SCAN_RSCS);
                }
                g_rscs_active_state = SCAN_DEV_STATE;
                APP_LOG_DEBUG("Start scanning, target device: RSCS.");
                break;
             case HRRCPS_EVT_ENABLE_HRS_NTF:
                error_code = hrs_c_heart_rate_meas_notify_set
                                                    (s_conn_idx_hrs_c, true);
                if (error_code != SDK_SUCCESS)
                {
                    hrrcps_op_error_handler(HRRCPS_CTRL_PT_HRS_NTF_ENABLE);
                }
                s_user_write_id = USER_WR_HRS_NTF_EN;
                APP_LOG_DEBUG("Enable HRS notification.");
                break;
            case HRRCPS_EVT_DISABLE_HRS_NTF:
                error_code = hrs_c_heart_rate_meas_notify_set
                                                   (s_conn_idx_hrs_c, false);
                if (error_code != SDK_SUCCESS)
                {
                    hrrcps_op_error_handler(HRRCPS_CTRL_PT_HRS_NTF_DISABLE);
                }
                s_user_write_id = USER_WR_HRS_NTF_DIS;
                APP_LOG_DEBUG("Disable HRS notification.");
                break;
            case HRRCPS_EVT_ENABLE_RSCS_NTF:
                error_code = rscs_c_rsc_meas_notify_set
                                                   (s_conn_idx_rscs_c, true);
```

版权所有 © 2020 深圳市汇顶科技股份有限公司

```
                if (error_code != SDK_SUCCESS)
                {
                    hrrcps_op_error_handler(HRRCPS_CTRL_PT_RSCS_NTF_ENABLE);
                }
                s_user_write_id = USER_WR_RSCS_NTF_EN;
                APP_LOG_DEBUG("Enable RSCS notification.");
                break;
            case HRRCPS_EVT_DISABLE_RSCS_NTF:
                error_code = rscs_c_rsc_meas_notify_set
                                                    (s_conn_idx_rscs_c, false);
                if (error_code != SDK_SUCCESS)
                {
                    hrrcps_op_error_handler(HRRCPS_CTRL_PT_RSCS_NTF_DISABLE);
                }
                s_user_write_id = USER_WR_RSCS_NTF_DIS;
                APP_LOG_DEBUG("Disable RSCS notification.");
                break;
            case HRRCPS_EVT_HRS_SENSOR_LOC_READ:
                error_code = hrs_c_sensor_loc_read(s_conn_idx_hrs_c);
                if (error_code != SDK_SUCCESS)
                {
                    hrrcps_op_error_handler(HRRCPS_CTRL_PT_HRS_SEN_LOC_READ);
                }
                APP_LOG_DEBUG("Read HRS sensor location.");
                break;
            case HRRCPS_EVT_RSCS_SENSOR_LOC_READ:
                error_code = rscs_c_sensor_loc_read(s_conn_idx_rscs_c);
                if (error_code != SDK_SUCCESS)
                {
                    hrrcps_op_error_handler(HRRCPS_CTRL_PT_RSCS_SEN_LOC_READ);
                }
                APP_LOG_DEBUG("Read RSCS sensor location.");
                break;
            default:
                break;
        }
    }
}
```

- 连接HRS传感器指令

  HRRCPS解析该指令并以“HRRCPS_EVT_SCAN_HRS”事件上报至应用层，然后开启扫描、过滤HRS传感器设备。

扫描到目标设备后，HRS RSCS Relay设备会依次停止扫描、连接HRS传感器设备，并发现Heart Rate Service，相关描述下。

1. 停止扫描（通过判断广播数据中是否含有HRS UUID，以确定是否为HRS传感器），代码片段如下。

```
void app_adv_report_handler(const uint8_t *p_data, uint16_t length,
                            const gap_bdaddr_t *p_bdaddr)
```

版权所有 © 2020 深圳市汇顶科技股份有限公司

```
{
    sdk_err_t error_code;

    if (SCAN_DEV_STATE == g_hrs_active_state)
    {
        if (target_srvc_uuid_find(p_data, length, BLE_ATT_SVC_HEART_RATE))
        {
            memcpy(&s_hrs_target_addr, p_bdaddr, sizeof(gap_bdaddr_t));
            error_code = ble_gap_scan_stop();
            APP_ERROR_CHECK(error_code);
            APP_LOG_DEBUG("Stop scanning for Connecting HRS procedure.");
            g_hrs_active_state = CONN_UNDERWAY_STATE;
            return;
        }
    }

    if (SCAN_DEV_STATE == g_rscs_active_state)
    {
        If (target_srvc_uuid_find(p_data, length,
            BLE_ATT_SVC_RUNNING_SPEED_CADENCE))
        {
            memcpy(&s_rscs_target_addr, p_bdaddr, sizeof(gap_bdaddr_t));
            error_code = ble_gap_scan_stop();
            APP_ERROR_CHECK(error_code);
            APP_LOG_DEBUG("Stop scanning for Connecting RSCS procedure.");
            g_rscs_active_state = CONN_UNDERWAY_STATE;
            return;
        }
    }
}
```

2. 连接HRS传感器设备。

```
void app_scan_stop_handler(void)
{
    if (CONN_UNDERWAY_STATE == g_hrs_active_state)
    {
        s_gap_connect_param.peer_addr.addr_type = s_hrs_target_addr.addr_type;
        s_gap_connect_param.peer_addr.gap_addr  = s_hrs_target_addr.gap_addr;
        ble_gap_ext_connect(BLE_GAP_OWN_ADDR_STATIC, &s_gap_connect_param);
    }

    if (CONN_UNDERWAY_STATE == g_rscs_active_state)
    {
        s_gap_connect_param.peer_addr.addr_type = s_rscs_target_addr.addr_type;
        s_gap_connect_param.peer_addr.gap_addr  = s_rscs_target_addr.gap_addr;
        ble_gap_ext_connect(BLE_GAP_OWN_ADDR_STATIC, &s_gap_connect_param);
    }
}
```

版权所有 © 2020 深圳市汇顶科技股份有限公司

3. 发现Heart Rate Service。

```
void app_connected_handler(uint8_t conn_idx,
                           const gap_conn_cmp_t *p_conn_param)
{
    if (GAP_LL_ROLE_MASTER == p_conn_param->ll_role)
    {
        if (CONN_UNDERWAY_STATE == g_hrs_active_state)
        {
            APP_LOG_DEBUG("Connected to HRS, IDX:%d.",conn_idx);

            s_conn_idx_hrs_c    = conn_idx;
            g_hrs_active_state = CONNECTED_STATE;

            hrs_c_disc_srvc_start(s_conn_idx_hrs_c);
            ble_srv_disc_proc_state_set(HRS_DISC_PROC_ID,
                                        BLE_SRV_DISC_UNDERWAY);

            APP_LOG_DEBUG("Start discovery HRS service.");
        }

        if (CONN_UNDERWAY_STATE == g_rscs_active_state)
        {
            APP_LOG_DEBUG("Connected to RSCS, IDX:%d.", conn_idx);

            s_conn_idx_rscs_c   = conn_idx;
            g_rscs_active_state = CONNECTED_STATE;

            rscs_c_disc_srvc_start(s_conn_idx_rscs_c);;
            ble_srv_disc_proc_state_set(RSCS_DISC_PROC_ID,
                                        BLE_SRV_DISC_UNDERWAY);

            APP_LOG_DEBUG("Start discovery RSCS service.");
        }
    }

    if (GAP_LL_ROLE_SLAVE == p_conn_param->ll_role)
    {
        APP_LOG_DEBUG("Connected to Collector, IDX:%d.", conn_idx);
        s_conn_idx_collector = conn_idx;
    }
}
```

- 开启HRS通知指令。

  HRRCPS解析该指令并以“HRRCPS_EVT_HRS_SENSOR_LOC_READ”事件上报至应用层，开启HRS传感器通知，然后将接收到心率数据中继至GRToolbox。

```
static void hrs_c_evt_process(hrs_c_evt_t *p_evt)
{
```

版权所有 © 2020 深圳市汇顶科技股份有限公司

```
    ...
    switch (p_evt->evt_type)
    {
        ...
        case HRS_C_EVT_HR_MEAS_VAL_RECEIVE:
            for (rr_intervals_idx = 0; rr_intervals_idx <
               p_evt->value.hr_meas_buff.rr_intervals_num; rr_intervals_idx++ )
            {
             hrs_rr_interval_add(
                     p_evt->value.hr_meas_buff.rr_intervals[rr_intervals_idx]);
            }
            hrs_sensor_contact_detected_update
                        (p_evt->value.hr_meas_buff.is_sensor_contact_detected);

            hrs_heart_rate_measurement_send(s_conn_idx_collector,
                                            p_evt->value.hr_meas_buff.hr_value,
                                  p_evt->value.hr_meas_buff.energy_expended);
            break;
        ...
        default:
            break;
    }
}
```

- 获取HRS传感器设备位置的指令。

  HRRCPS解析该指令并以“HRRCPS_EVT_HRS_SENSOR_LOC_READ”事件上报至应用层，然后读取HRS传感器位置，并将读取数据中继至GRToolbox。

```
static void hrs_c_evt_process(hrs_c_evt_t *p_evt)
{
    ...
    switch (p_evt->evt_type)
    {
        ...
        case HRS_C_EVT_SENSOR_LOC_READ_RSP:
            APP_LOG_DEBUG("HRS sensor location is got.");
            hrs_sensor_location_set((hrs_sensor_loc_t)p_evt->value.sensor_loc);
            rsp_val.cmd_id       = HRRCPS_CTRL_PT_HRS_SEN_LOC_READ;
            rsp_val.rsp_id       = HRRCPS_RSP_ID_OK;
            rsp_val.is_inc_prama = true;
            rsp_val.rsp_param    = p_evt->value.sensor_loc;
            error_code = hrrcps_ctrl_pt_rsp_send(s_conn_idx_collector,
                                                 &rsp_val);
            APP_ERROR_CHECK(error_code);
            break;
        default:
            break;
    }
}
```

版权所有 © 2020 深圳市汇顶科技股份有限公司

说明:

GRToolbox控制HRS RSCS Relay设备与RSCS传感器设备交互流程，与上述流程类似，此处不再赘述。

版权所有 © 2020 深圳市汇顶科技股份有限公司