



GR55xx HRS RSCS Relay Example Application

Version: 1.9

Release Date: 2021-04-26

Copyright © 2021 Shenzhen Goodix Technology Co., Ltd. All rights reserved.

Any excerpt, backup, modification, translation, transmission or commercial use of this document or any portion of this document, in any form or by any means, without the prior written consent of Shenzhen Goodix Technology Co., Ltd is prohibited.

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 use and verify a Heart Rate Sensor & Running Speed and Cadence Sensor Relay (HRS RSCS Relay) example in the GR55xx SDK, to help users quickly get started with secondary development.

Audience

This document is intended for:

- GR55xx user
- GR55xx developer
- GR55xx tester
- Hobbyist developer
- Technical writer

Release Notes

This document is the seventh release of *GR55xx HRS RSCS Relay Example Application*, corresponding to GR55xx SoC series.

Revision History

Version	Date	Description
1.0	2019-12-08	Initial release
1.3	2020-03-16	Updated the release time in the footers.
1.5	2020-05-30	Adjusted the indentation of the code in “Application Details”.
1.6	2020-06-30	Updated the document version based on SDK changes.
1.7	2020-11-09	Updated figures in “Test and Verification”.
1.8	2020-12-15	Updated GRTtoolbox UI figure based on software update.
1.9	2021-04-26	Optimized descriptions in “Initial Operation” and “Application Details”.

Contents

Preface.....	I
1 Introduction.....	1
2 Profile Overview.....	2
3 Initial Operation.....	4
3.1 Supported Development Platform.....	4
3.2 Firmware Download.....	4
3.3 Test and Verification.....	4
4 Application Details.....	9
4.1 Running Procedures.....	9
4.2 Major Code.....	9
4.2.1 Receiving a Command from GRTtoolbox.....	9
4.2.2 Command to Connect to HRS Device.....	10
4.2.3 Command to Enable HRS Notification.....	10
4.2.4 Command to Obtain the HRS Device Location.....	11

1 Introduction

The Heart Rate Sensor & Running Speed and Cadence Sensor Relay (HRS RSCS Relay) example demonstrates how to apply GR55xx SoCs in scenarios with multi-roles (Peripheral and Central) and multi-connections, to enable functions of an HRS RSCS Relay device. The HRS RSCS Relay device can serve as both a collector and a sensor.

- **Collector**
As a GATT Client, the HRS RSCS Relay device receives measurement data from heart rate sensor as well as running speed and cadence sensor.
- **Sensor**
As a GATT Server, the HRS RSCS Relay device sends the received data to other collectors, such as GRToolbox (a Bluetooth LE debugging App for GR55xx SoCs).

This document introduces how to use and verify an HRS RSCS Relay example in the GR55xx SDK.

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

Table 1-1 Reference documents

Name	Description
GR55xx Sample Service Application and Customization	Introduces how to apply and customize Goodix Sample Service in developing Bluetooth LE applications based on GR55xx SDK.
Developer guide of the specific GR55xx SoC	Introduces the software/hardware and quick start guide of the specific GR55xx SoC in use.
Bluetooth Core Spec	Offers official Bluetooth standards and core specification from Bluetooth SIG.
Bluetooth GATT Spec	Provides details about Bluetooth profiles and services. Available at www.bluetooth.com/specifications/gatt .
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/ .

2 Profile Overview

The HRS RSCS Relay example implements the following profiles:

- Standard profiles: Heart Rate Profile as well as Running Speed and Cadence Profile, which are defined by Bluetooth SIG
- Custom profile: Goodix HRS RSCS Relay Control Point Profile, which is defined by Goodix

The application scenarios where GRToolbox is used as an HRS RSCS Relay collector are shown in [Figure 2-1](#).

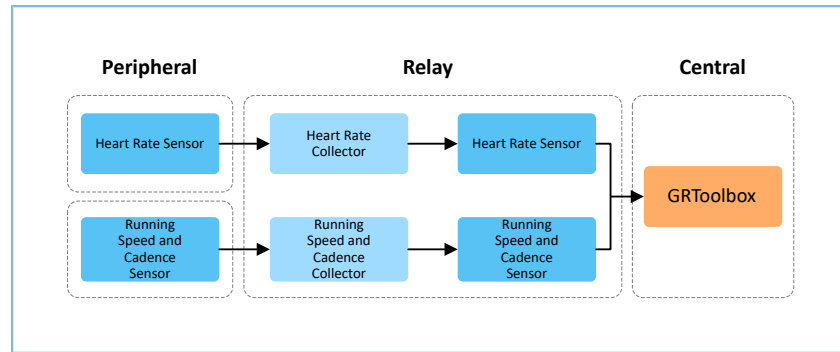


Figure 2-1 Application scenarios

HRS RSCS Relay device registers the following profiles when it is used as a collector:

- Heart Rate Client Profile: Receive measurement data from a heart rate sensor.
- Running Speed and Cadence Client Profile: Receive measurement data from a running speed and cadence sensor.

HRS RSCS Relay device registers the following profiles when it is used as a sensor:

- Heart Rate Server Profile: Relay the received data from a heart rate sensor to GRToolbox.
- Running Speed and Cadence Server Profile: Relay the received data from a running speed and cadence sensor to GRToolbox.
- Goodix HRS RSCS Relay Control Point Profile: Receive control commands from GRToolbox and returns execution outcomes.

Goodix HRS RSCS Relay Control Point Profile includes HRS RSCS Relay Control Point Service (HRRPCS), with a 128-bit vendor-specific UUID of A6ED0601-D344-460A-8075-B9E8EC90D71B.

HRRPCS has the following characteristics:

- HRR Control Point characteristic: Receive control commands from the HRS RSCS Relay collector.
- HRR Control Point Response characteristic: Return execution outcomes to the HRS RSCS Relay collector.

These characteristics are described in detail as follows:

Table 2-1 HRRPCS characteristics

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

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

3 Initial Operation

This chapter introduces how to quickly verify an HRS RSCS Relay example in the GR55xx SDK.

Note:

SDK_Folder is the root directory of the GR55xx SDK in use.

3.1 Supported Development Platform

You can use and modify the HRS RSCS Relay example on the following platform.

Table 3-1 Supported development platform

Hardware Platform	Development Board Model
GR551x development kit	GR5515-SK-BASIC

3.2 Firmware Download

The source code of the HRS RSCS Relay example is in SDK_Folder\projects\ble\ble_multi_role\ ble_app_hrs_rscs_relay.

You can download *ble_app_hrs_rscs_relay_fw.bin*, *ble_app_hrs_fw.bin*, and *ble_app_rscs_fw.bin* to three GR55xx Starter Kit Boards (serving as the Relay device, the HRS device, and the RSCS device respectively) through GProgrammer. For details, see [GProgrammer User Manual](#).

Note:

- The *ble_app_hrs_rscs_relay_fw.bin* is in SDK_Folder\projects\ble\ble_multi_role\ble_app_hrs_rscs_relay\build.
- The *ble_app_hrs_fw.bin* is in SDK_Folder\projects\ble\ble_peripheral\ble_app_hrs\build.
- The *ble_app_rscs_fw.bin* is in SDK_Folder\projects\ble\ble_peripheral\ble_app_rscs\build.
- You can find GProgrammer in SDK_Folder\tools\GProgrammer.

3.3 Test and Verification

The hardware and software required for test and verification are listed in [Table 3-2](#).

Table 3-2 Hardware and software resources

Name	Description
GRTtoolbox (Android)	A Bluetooth LE debugging tool. Available in SDK_Folder\tools\GRTtoolbox.

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.

When the HRS RSCS Relay device, the HRS device, the RSCS device, and GRToolbox are ready, test and verify the HRS RSCS Relay example. Steps are described as follows:

1. Scan the HRS RSCS Relay device.

Run GRToolbox, and select **Application > RELAY**.

Start scanning and discover a device with the advertising name **Goodix_HRS_RSCS_RELAY** (the advertising name can be modified in *user_app.c*), as shown in [Figure 3-1](#).

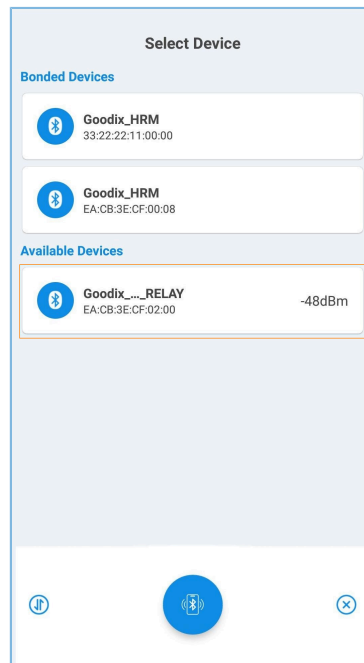


Figure 3-1 Discovering Goodix_HRS_RSCS_RELAY on GRToolbox

Note:

If the length of the device name exceeds 14 characters, the middle part of the device name is replaced with an ellipsis.

2. Connect to the HRS RSCS Relay device.

Select **Goodix_HRS_RSCS_RELAY** to establish connection, and enter the HRS RSCS RELAY interface.

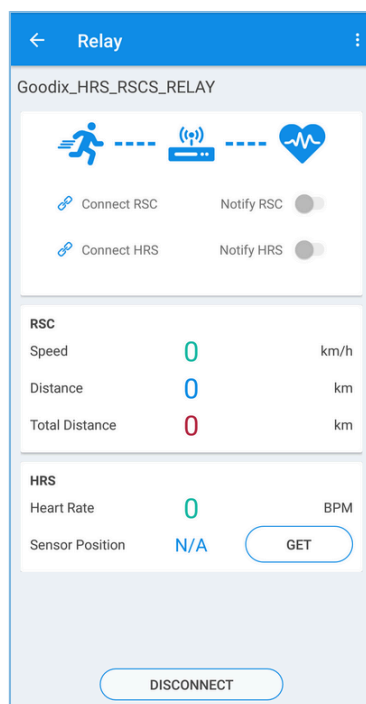



Figure 3-2 HRS RSCS RELAY interface

3. Connect to sensor devices.

Tap  to enable the HRS RSCS Relay device to scan and connect to the HRS and RSC devices. The interface below is shown after the Relay device is connected to the HRS and RSC devices.

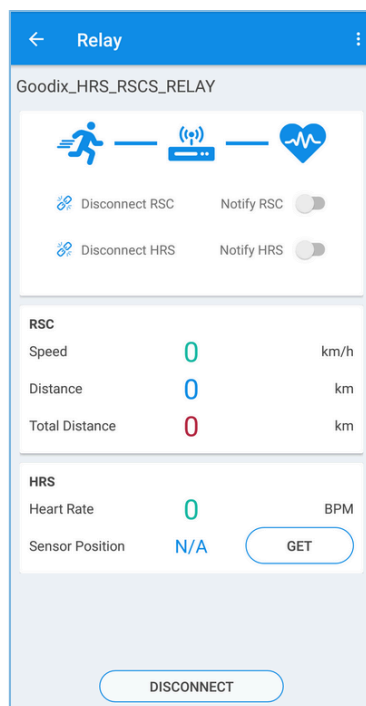



Figure 3-3 Connecting to the HRS and RSC devices

4. Enable sensor notifications.

Tap  to enable the HRS RSCS Relay device to notify the HRS and RSC devices to report measurement data.

This allows the phone to receive heart rate, running speed, and cadence information relayed from the HRS RSCS Relay device.

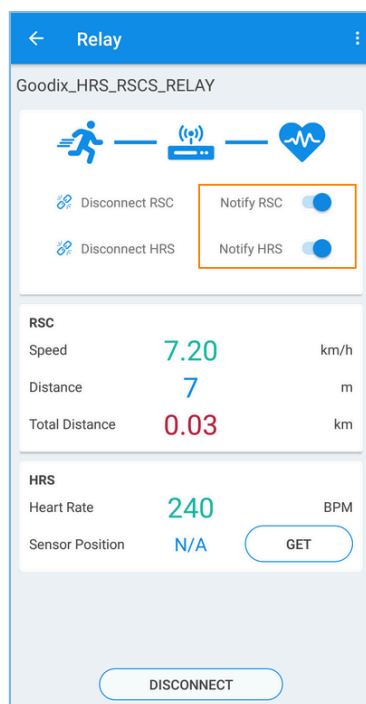


Figure 3-4 Enabling HRS and RSC notifications

5. Read the HRS device location.

Tap **GET** to enable the HRS RSCS Relay device to read the HRS device location.

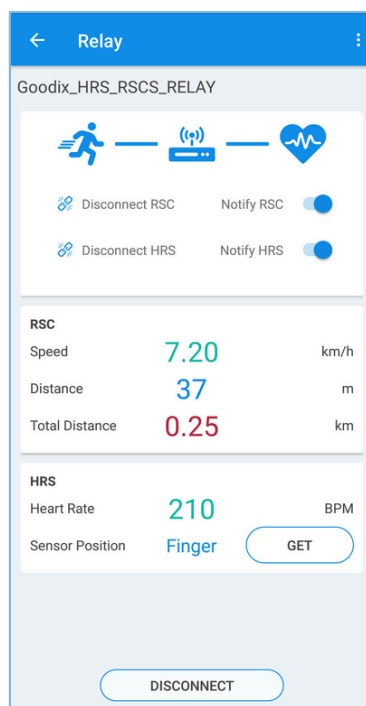


Figure 3-5 Reading the HRS device location

If GRToolbox displays information as shown above, the HRS RSCS Relay example runs successfully.

4 Application Details

This chapter introduces the running procedures and major code of the HRS RSCS Relay example.

4.1 Running Procedures

When the HRS RSCS Relay example starts running, it performs operations such as initializing peripherals and BLE Protocol Stack, adding profiles, and enabling advertising. The running procedures of the HRS RSCS Relay example after GRTtoolbox discovers the advertisement and connection is established are shown in [Figure 4-1](#):

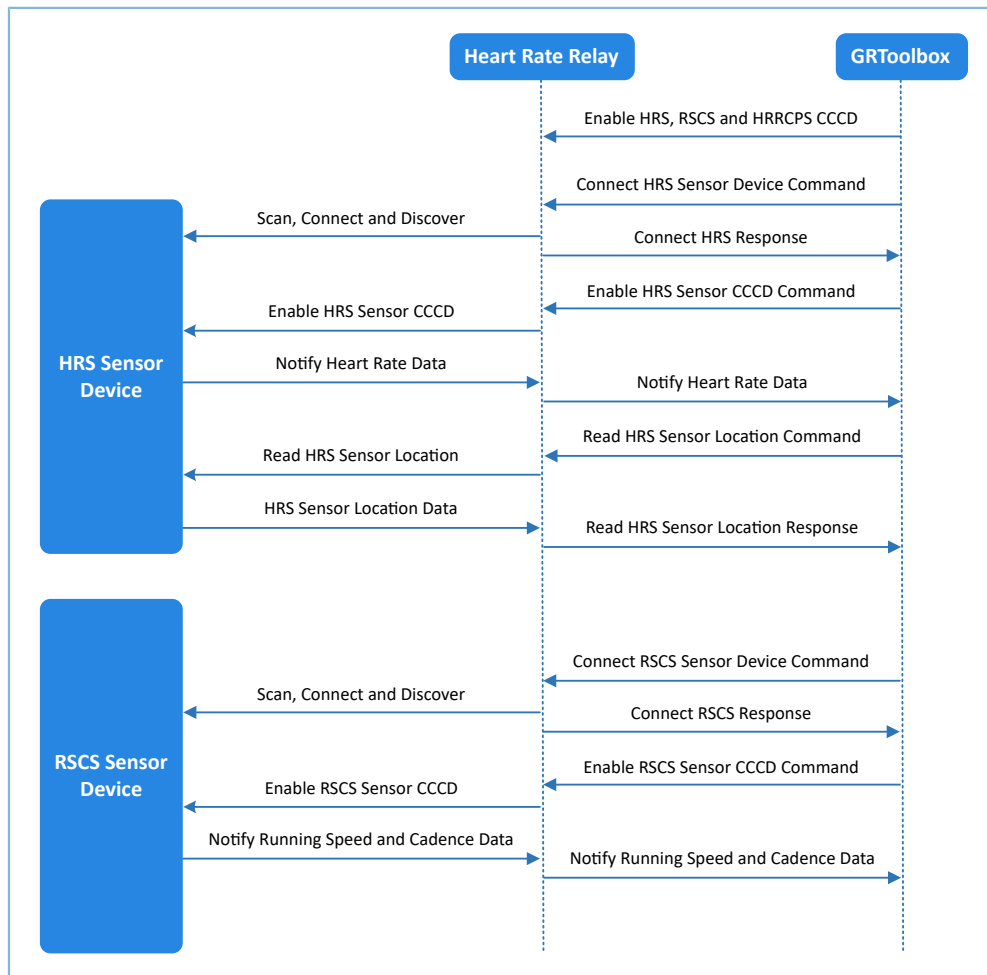


Figure 4-1 Running procedures

4.2 Major Code

In the following parts, the HRS device is taken as an example to introduce major code related to interactions between GRTtoolbox, Relay device, and HRS device.

4.2.1 Receiving a Command from GRTtoolbox

When the HRR Control Point characteristic receives control command data from GRTtoolbox, it parses the corresponding event and reports to the application layer, and executes the corresponding command.

Path: user_app\user_app.c under the project directory

Name: hrrcps_evt_process()

```
static void hrrcps_evt_process(hrrcps_evt_t *p_evt)
{
    ...
    switch (p_evt->evt_type)
    {
        case HRRCPES_EVT_CTRL_PT_IND_ENABLE:
            APP_LOG_DEBUG("HRR Control Point Indication is enabled.");
            break;

        case HRRCPES_EVT_CTRL_PT_IND_DISABLE:
            APP_LOG_DEBUG("HRR Control Point Indication is disabled.");
            break;

        ...
        default:
            break;
    }
    ...
}
```

4.2.2 Command to Connect to HRS Device

HRRCPES parses the command and reports the **HRRCPES_EVT_SCAN_HRS** event to the application layer; HRRCPES then starts scanning and searches for the HRS device. After discovering an advertisement, the Relay device judges whether the advertising device is the target device by using app_adv_report_handler() to check whether the advertising data contains HRS UUID. After the target device is discovered, the Relay device stops scanning, then establishes connection with the target device through app_scan_stop_handler(), and finally finds Heart Rate Service through app_connected_handler() (these functions are in user_app.c).

Path: user_app\user_app.c under the project directory

Name: hrrcps_evt_process()

```
static void hrrcps_evt_process(hrrcps_evt_t *p_evt)
{
    ...
    case HRRCPES_EVT_SCAN_HRS:
        ...
        error_code = ble_gap_scan_start();
        g_hrs_active_state = SCAN_DEV_STATE;
        break;
    ...
}
```

4.2.3 Command to Enable HRS Notification

HRRCPES parses the command and reports the **HRRCPES_EVT_ENABLE_HRS_NTF** event to the application layer; HRRCPES then enables HRS notification and relays the received heart rate data to GRTtoolbox.

Path: user_app\user_app.c under the project directory

Name: hrrcps_evt_process()

```
static void hrrcps_evt_process(hrrcps_evt_t *p_evt)
```

```
{
    ...
    case HRRCPES_EVT_ENABLE_HRS_NTF:
        error_code = hrs_c_heart_rate_meas_notify_set(s_conn_idx_hrs_c, true);
        s_user_write_id = USER_WR_HRS_NTF_EN;
        break;
    ...
}
```

Path: user_app\user_app.c under the project directory

Name: hrs_c_evt_process ()

```
static void hrs_c_evt_process(hrs_c_evt_t *p_evt)
{
    ...
    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;
    ...
}
```

4.2.4 Command to Obtain the HRS Device Location

HRRCPES parses the command and reports the **HRRCPES_EVT_HRS_SENSOR_LOC_READ** event to the application layer in BLE Protocol Stack; HRRCPES then reads the HRS device location and relays the data obtained to GRTtoolbox.

Path: user_app\user_app.c under the project directory

Name: hrrcps_evt_process()

```
static void hrrcps_evt_process(hrrcps_evt_t *p_evt)
{
    ...
    case HRRCPES_EVT_HRS_SENSOR_LOC_READ:
        error_code = hrs_c_sensor_loc_read(s_conn_idx_hrs_c);
        break;
    ...
}
```

Path: user_app\user_app.c under the project directory

Name: hrs_c_evt_process ()

```
static void hrs_c_evt_process(hrs_c_evt_t *p_evt)
{
    ...
    case HRS_C_EVT_SENSOR_LOC_READ_RSP:
        hrs_sensor_location_set((hrs_sensor_loc_t)p_evt->value.sensor_loc);
        rsp_val.cmd_id = HRRCPES_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;
...
}
```

Note:

You can use GRToolbox to control the interactions between the HRS RSCS Relay device and the RSCS device, which are similar to the procedures mentioned above, and therefore are not explained in this document.