GR551x HID Mouse Example Application

Version: 1.6

Release Date: 2020-06-30

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 run Human Input Device (HID) Service and GR551x HID mouse example for the first time and its application details, to help users quickly get started with secondary development.

Audience

This document is intended for:

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

Release Notes

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

Revision History

Version	Date	Description
1.0	2019-12-08	Initial release, corresponding to GR551x_SDK_1.0.0
1.3	2020-03-16	Updated the release time in the footers.
1.5	2020-05-30	Modified the booting procedures of a GR551x mouse example in “Section 4.2.1 Boot”.
1.6	2020-06-30	Updated the document version based on SDK changes.

Contents

1 Introduction

Based on GR551x SDK and GR5515 Starter Kit Board (GR5515 SK Board), GR551x HID Mouse example implements a HID firmware example.

It is recommended to refer to the following documents and information before using and modifying a HID mouse example.

Table 1-1 Reference documents

Name	Description
GR551x Developer Guide	Introduces the software/hardware and quick start guide of GR551x SoCs.
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 http://www.keil.com/support/man/docs/uv4/.
Bluetooth Core Spec v5.1	Offers official Bluetooth standards and core specification (v5.1) from Bluetooth SIG. Available at https://www.bluetooth.com/specifications/bluetooth-core-specification/.
Bluetooth GATT Spec	Provides details about Bluetooth profiles and services. Available at http://www.bluetooth.com/specifications/gatt.
GProgrammer User Manual	Lists GProgrammer operational instructions including downloading firmware to and encrypting firmware on GR551x SoCs.

2 HID over GATT Profile (HOGP)

This chapter introduces the device roles, HID Service, and security requirements defined by HID over GATT Profile (HOGP).

2.1 Device Roles

HOGP defines two roles: HID Device and HID Host.

- HID Device

  The HID Device shall perform the GAP Peripheral role as a GATT Server. Common HID Devices include keyboards and mice.

  A HID Device shall contain at least an HID Service instance, a Battery Service (BAS) instance, a Device Information Service (DIS) instance, and optionally a Scan Parameters Service instance. A HID Device can contain one or more other types of GATT Service instances that do not serve as parts of HOGP.

  The ble_app_hids_mouse example used to implement the HID Device in a GR551x SDK contains a HID Service instance, a BAS instance, and a DIS instance.

- HID Host

  The HID Host, parsing the input data delivered by the HID Device, shall perform the GAP Central role as a GATT Client. The HID Host is responsible for scanning, connecting to, and configuring the HID Device. When the connection between the HID Device and HID Host is established, the HID Host can receive and read data from as well as write data to the HID Device. Common HID Host examples are Android phones.

2.2 HID Service

The HID Service presents data and associated formats of the HID Device (defined in *USB HID Specification*) to the HID Host.

The HID Service uses characteristics to access data on a HID Device. For details about the characteristics, see Table 2-1.

Table 2-1 HID Service characteristics

Characteristic		Requirement	Description
Protocol Mode		Mandatory for Boot Protocol Mode support	Expose the current protocol mode, or set the desired protocol mode.
Report	Input Report Type	Mandatory to support at least one Report Type if the Report characteristic is supported	Exchange data between HID Device and HID Host.
	Output Report Type		
	Feature Report Type		
Report Map		Mandatory	Define formatting information for the data transferred between HID Device and HID Host.
Boot Keyboard Input Report		Mandatory for keyboards	Transfer fixed format and length Input Report data in Boot Protocol Mode

Characteristic	Requirement	Description
Boot Keyboard Output Report	Mandatory for keyboards	Transfer fixed format and length Output Report data in Boot Protocol Mode
Boot Mouse Input Report	Mandatory for mice	Transfer fixed format and length Input Report data in Boot Protocol Mode
HID Information	Mandatory	Hold a set of values known as the HID Device's HID Attributes
HID Control Point	Mandatory	A control-point for switching Suspend status

2.3 Security Requirements

According to *Bluetooth Core Spec v5.1*, LE Security Mode 1 includes Security Level 2 and Security Level 3.

- Security Level 2: Encrypted Link required; MITM protection not necessary.
- Security Level 3: MITM-protected encrypted link required.

According to *HOGP Specification*, the HID Device shall support either Security Level 2 or 3.

- The Security Property of all characteristics supported by the HID Service shall be set to Security Mode 1 and either Security Level 2 or 3.
- It is recommended that all characteristics specified by Device Information Service, Scan Parameters Service, and BAS should be set to the same LE Security Mode and Security Level.

Users can set the security parameters for a GR551x mouse example by using gap_params_init(). For details, see "Section 4.3.3 Security Parameter Configuration".

3 Initial Operation

This chapter introduces how to run and verify a GR551x mouse example in a GR551x SDK.

Note:

SDK_Folder is the root directory of GR551x SDK.

3.1 Preparation

Perform the following tasks before running a HID mouse example.

- **Hardware preparation**

Table 3-1 Hardware preparation

Name	Description
Development board	GR5515 SK Board
Android phone	Phones supporting HOGP and running on Android 4.4 (KitKat) and later versions
Connection cable	Micro USB 2.0 serial cable

- **Software preparation**

Table 3-2 Software preparation

Name	Description
Windows	Windows 7 and later versions
J-Link driver	A J-Link driver. Available at www.segger.com/downloads/jlink/.
Keil MDK5	An integrated development environment (IDE). Available at www.keil.com/download/product/.
GProgrammer (Windows)	A GR551x programming tool. Available in `SDK_Folder\tools\GProgrammer`.

3.2 Hardware Connection

Connect the GR5515 SK Board to a PC with a Micro USB 2.0 cable. The connection diagram is as shown in the figure below:

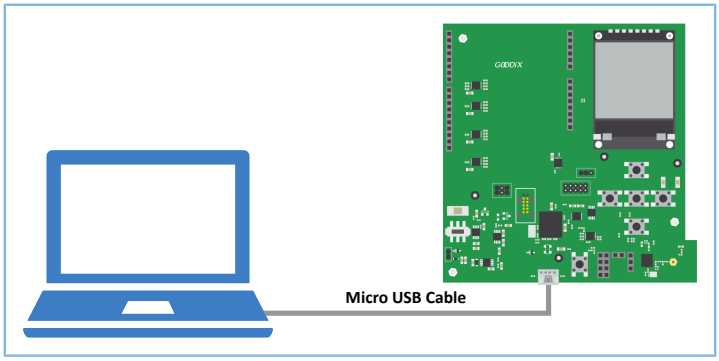

Figure 3-1 Hardware connection

3.3 Firmware Download

Download *ble_app_hids_mouse_fw.bin* to the GR5515 SK Board through GProgrammer. For specific operations, refer to *GProgrammer User Manual*.

Note:

The *ble_app_hids_mouse_fw.bin* is in `SDK_Folder\projects\ble\ble_peripheral\ble_app_hids_mouse\build\`.

3.4 Test and Verification

You can use an Android phone to test and verify a mouse example.

1. Press **RESET** on the GR5515 SK Board, and the board enters Advertising mode.
2. Open the Bluetooth setting interface on the phone, and turn **Bluetooth** on. Wait until the phone discovers **Goodix_Mouse**.
3. Tap **Goodix_Mouse** to connect it to the phone.
4. Enter **123456** into **Pin Code** in the pop-up dialog box.

After pairing, you can see the device named **Goodix_Mouse** under **Paired devices** on the phone, and the device shows as **Connected**. As shown in Figure 3-2, you can long press the **UP**, **DOWN**, **LEFT**, or **RIGHT** buttons on the GR5515 SK Board to move the mouse arrow.

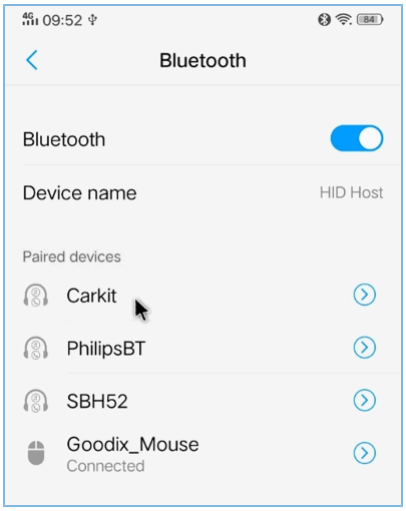

Figure 3-2 GR551x mouse arrow on an Android phone

In addition, you can use the mouse example in media play control scenarios. Press **UP** or **DOWN** twice on the GR5515 SK Board to increase or decrease the volume respectively. Press **OK** twice to stop or resume playing. Press **RIGHT** twice to switch to the next track, and press **LEFT** twice to switch to the previous track. It should be noted that media play control functions of the mouse example may be unavailable in some scenarios due to version limitations on Android operating systems.

4 Application Details

This chapter introduces the project directory, running procedures, and major code of the GR551x mouse example.

4.1 Project Directory

The source code and project file of the GR551x mouse example are in SDK_Folder\projects\ble\ble_peripheral\ble_app_hids_mouse, and project file is in the Keil_5 folder.

Double-click the project file, *ble_app_hids_mouse.uvprojx*, to check the ble_app_hids_mouse project directory structure of the mouse example in Keil. Related files are described in Table 4-1.

Table 4-1 File description of ble_app_hids_mouse

Group	File	Description
gr_profiles	hids.c	This file helps implement HID Service.
	bas.c	This file helps implement BAS.
	dis.c	This file helps implement DIS.
user_callback	user_xxx_callback.c	This file helps implement Bluetooth low energy SDK callback functions.
user_platform	user_periph_setup.c	This file provides an entry for user peripheral initialization and implements button process functions.
user_app	main.c	This file contains the main() function.
	user_app.c	This file helps implement logics of Bluetooth advertising and connection at the user application layer.
	user_mouse.c	This file initializes HOGP-specified GATT Service and delivery of mouse data.

4.2 Running Procedures

The running procedures of a GR551x mouse example can be divided into three phases: boot, connection to HID Host, and disconnection from HID Host.

4.2.1 Boot

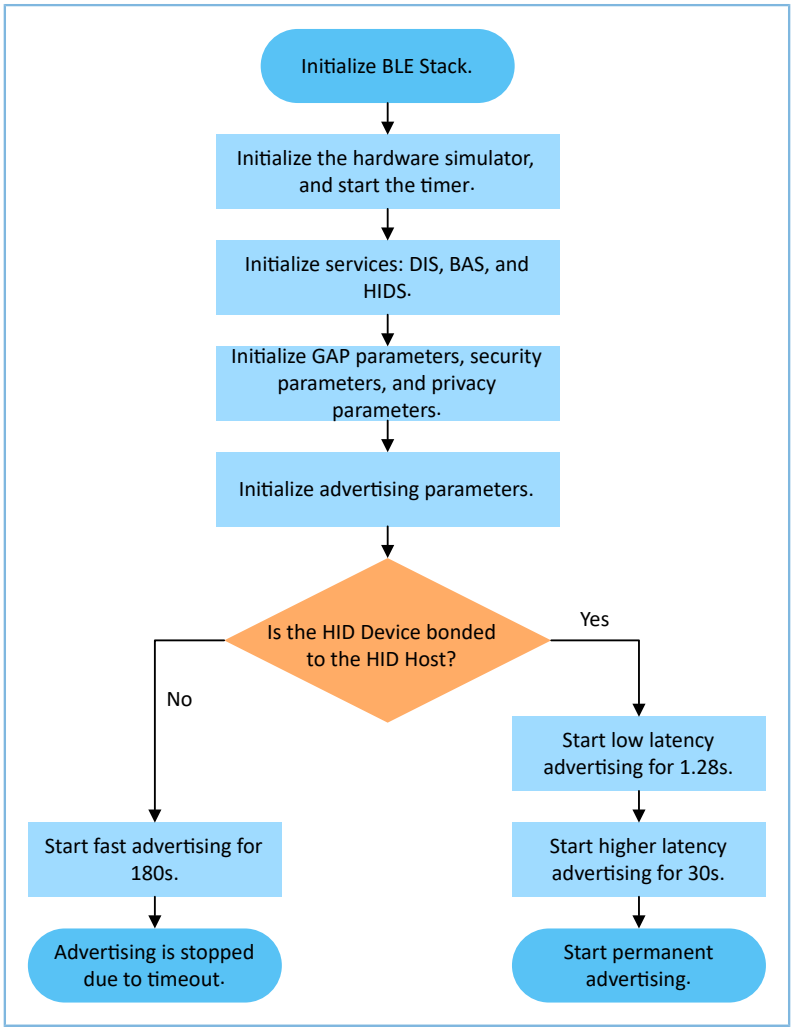

Figure 4-1 Booting procedures of a GR551x mouse example

During booting, the relation (bond or unbonded) between HID Device and HID Host affects the advertising parameters. For details, see “Chapter 5 Connection Establishment” in *HOGP Specification*.

Note:

If the value of bool erase_bond in void adv_params_init(bool erase_bond) is true, the bond information is erased after each reboot.

4.2.2 Connection to HID Host

When the HID Host successfully connects to, pairs with, and bonds to a HID Device, the HID Report Notification of the HID Device is enabled.

The key_continue_press_task() function of the GR551x mouse example receives press events from the Board Support Package (BSP) layer and calls hids_input_rep_send() to transmit the mouse data to the HID Host.

4.2.3 Disconnection from HID Host

When the HID Device is disconnected from the HID Host, the BLE Protocol Stack notifies the disconnection event to the mouse example by using gap_cb_fun_t::app_gap_disconnect_cb(). The app_disconnected_handler() function enables the HID Device to restart advertising based on the disconnection reason.

According to *HOGP Specification*, the HID Device should restart advertising if the connection is terminated due to link loss. To simplify tests, the GR551x mouse example restarts advertising when the disconnection reason is Remote User Terminated Connection. If the HID Device is bonded to the HID Host, the GR551x mouse example enters Low Latency Advertising, Higher Latency Advertising, and Permanent Advertising successively.

4.3 Major Code

The major code of the GR551x mouse example is in the following files under the Keil project directory:

- `user_app\user_app.c`
- `user_app\user_mouse.c`
- `user_callback\user_gap_callback.c`
- `user_callback\user_sm_callback.c`

4.3.1 Battery Simulator Initialization

Initialize the Battery Simulator by using the hw_simulator_init() function in *user_app.c*. The status of the Battery Simulator is updated by hw_simulator_timer_handler(), the timeout handler function of timers.

```
void ble_init_cmpl_callback(void)
{
    ...
    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);
    ...
}
```

When the initialization of the Battery Simulator is complete, the Battery Simulator begins to simulate the battery level changes.

```
static void hw_simulator_timer_handler(void *p_arg)
{
    uint8_t battery_level = (uint8_t)sensorsim_measure(&s_battery_sim_state,
                                                       &s_battery_sim_cfg);
    bas_batt_lvl_update(0, 0, battery_level);
}
```

Note:

The GR551x mouse example simulates the battery level changes by using APIs provided by the Sensor Simulator library. The source code of the Sensor Simulator library is stored in SDK_Folder\components\libraries\sensorsim\.

4.3.2 HID Service Initialization

Configure initialization parameters of HID Service by using hids_init() function in *user_mouse.c*.

The *user_mouse.c* file defines static const uint8_t rep_map_data[]. According to formats specified in *USB HID Specification*, static const uint8_t rep_map_data[] defines Report Map characteristics of mouse report and media key report.

Note:

The data length of static uint8_t rep_map_data[] should be equal to or shorter than the REPORT_MAP_MAX_SIZE defined in *hids.h*.

```
/**Limitation of length, as per Section 2.6.1 in HIDS Spec, V 1.0 */
#define REPORT_MAP_MAX_SIZE 512
```

If the NormallyConnectable bit of hids_init_t::hid_info::flags is set, s_normally_connectable should be set to true.

```
hid_info_flags = HIDS_INFO_FLAG_REMOTE_WAKE_MSK |
                 HIDS_INFO_FLAG_NORMALLY_CONNECTABLE_MSK;
s_normally_connectable = true;
```

4.3.3 Security Parameter Configuration

The gap_params_init() function in *user_app.c* sets the following security parameters in compliance with security requirements specified in *HOGP Specification*. For details, see “Section 2.3 Security Requirements”.

```
//set the default security parameters.
sec_param_t sec_param =
{
    .level = SEC_MODE1_LEVEL3,
    .io_cap = IO_DISPLAY_ONLY,
    .oob = false,
    .auth = AUTH_BOND | AUTH_MITM | AUTH_SEC_CON,
    .key_size = 16,
    .ikey_dist = KDIST_ALL,
    .rkey_dist = KDIST_ALL,
};
```

For details about sec_param_t setting, see "Chapter 4 Security Manager (SM)" in *GR551x Bluetooth Low Energy Stack User Guide*.

The gap_params_init() function uses the following function to enable privacy mode.

```
ble_gap_privacy_params_set(900, true);
```

In privacy mode, the GR551x mouse example generates a device address at an interval of 900s and does not respond to Connect requests from the bonded HID Host by using Public Address.

The app_sec_rcv_enc_req_cb() function in *user_sm_callback.c* is used to respond to encrypted pairing requests from the HID Host. In "Section 3.4 Test and Verification", the input pin code is the tk value set in the app_sec_rcv_enc_req_cb() function.

```
// user need to input the password
case TK_REQ:
    APP_LOG_INFO("Please Input pin code: 123456");
    cfm_enc.req_type = TK_REQ;
    cfm_enc.accept = true;
    tk = 123456;
    memset(cfm_enc.data.tk.key, 0, 16);
    cfm_enc.data.tk.key[0] = (uint8_t)((tk & 0x000000FF) >> 0);
    cfm_enc.data.tk.key[1] = (uint8_t)((tk & 0x0000FF00) >> 8);
    cfm_enc.data.tk.key[2] = (uint8_t)((tk & 0x00FF0000) >> 16);
    cfm_enc.data.tk.key[3] = (uint8_t)((tk & 0xFF000000) >> 24);
    break;
```

For more information about how to handle pairing and encryption requests, refer to "Section 4.2.1 Enable Bonding" in *GR551x Bluetooth Low Energy Stack User Guide*.