GR55xx Power Consumption Profile Example Application

Version: 2.0

Release Date: 2021-04-26

Shenzhen Goodix Technology Co., Ltd.

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 power consumption measurement 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 eighth release of *GR55xx Power Consumption Profile 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	Updated the logos in the headers.
1.6	2020-06-30	Updated the document version based on SDK changes.
1.7	2020-09-25	Added a note for removing the jumper caps on J5 before power consumption measurement and added the hardware layout of GR55xx SK Board in "Hardware Connection".
1.8	2020-11-09	Updated a figure in "Setting of Measurement Scenarios".
1.9	2020-12-15	Updated GRToolbox UI figure based on software update.
2.0	2021-04-26	Optimized descriptions in "Initial Operation" and "Application Details".

Contents

Preface I

1 Introduction 1

2 Profile Overview 2

3 Initial Operation 3

3.1 Supported Development Platform 3

3.2 Firmware Download 3

3.3 Test and Verification 3

4 Application Details 7

4.1 Running Procedures 7

4.2 Major Code 7

4.2.1 Power Management Configuration 7

4.2.2 Command Parsing, Execution, and Response 8

4.2.3 Enabling Notify 10

1 Introduction

GR55xx Power Consumption Profile (PCP) example allows users to set parameters in real time through mobile phones, to configure GR55xx power consumption measurement scenarios.

This document introduces how to use and verify a custom Goodix PCP 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 Power Consumption Service (PCS) is defined in PCP. It is customized by Goodix, the 128-bit vendor-specific UUID of which is A6ED0501-D344-460A-8075-B9E8EC90D71B, to transmit data and commands and receive responses.

PCS includes two characteristics:

- TX: Transmit data.
- Setting: Send commands to customize power consumption measurement scenarios and receive responses of command execution.

The characteristics are described in detail as follows.

Table 2-1 PCS characteristics

Characteristic	UUID	Type	Support	Security	Property
TX	A6ED0202-D344-460A-8075-B9E8EC90D71B	128 bits	Mandatory	None	Notify
Setting	A6ED0203-D344-460A-8075-B9E8EC90D71B	128 bits	Mandatory	None	Write, Indicate

3 Initial Operation

This chapter introduces how to run and verify the GR55xx PCP example.

Note:

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

3.1 Supported Development Platform

You can use and modify the PCP 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 PCP example is in `SDK_Folder\projects\ble\ble_peripheral\ ble_app_pcs`.

You can download *ble_app_pcs_fw.bin* to the GR55xx Starter Kit Board (GR55xx SK Board). For details, see *GProgrammer User Manual*.

Note:

- The *ble_app_pcs_fw.bin* is in `SDK_Folder\projects\ble\ble_peripheral\ble_app_pcs\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
Keysight 14585A	Power control and analysis software launched by Keysight
GRToolbox (Android)	A Bluetooth LE debugging tool. Available in `SDK_Folder\tools\GRToolbox`.

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.

For how to set up a hardware environment for test and verification, refer to "Environment Setup" in the specific GR55xx Sleep Mode and Power Consumption Measurement Application Note in use.

When the GR55xx SK Board downloaded with *ble_app_pcs_fw.bin* is powered on, the system enters Ultra Deep Sleep Mode. After reset, press **OK** on the board for more than 3 seconds to allow the system to initiate advertising, which lasts for 30 seconds. If the board is not connected to other devices, the system then enters Sleep Mode again due to advertising timeout; if the board is connected to other devices, the system also enters Sleep Mode when the board is in disconnected state, until the system wakes up. Press **OK** to wake the system up from Sleep Mode.

Note:

For more information about buttons on a GR55xx Board, see the user guide of the specific GR55xx Board in use.

The steps are provided as follows:

1. Establish connection and set measurement scenarios.

   Establish connection by using the mobile tool GRToolbox. Detailed steps are shown as follows:

   (1). Run GRToolbox, and select **Application** > **PCS**.

   (2). Tap **CONNECT** and then start scanning target devices. Discover a device with the advertising name **Goodix_Power** (the advertising name can be modified in the *user_app.c* file).

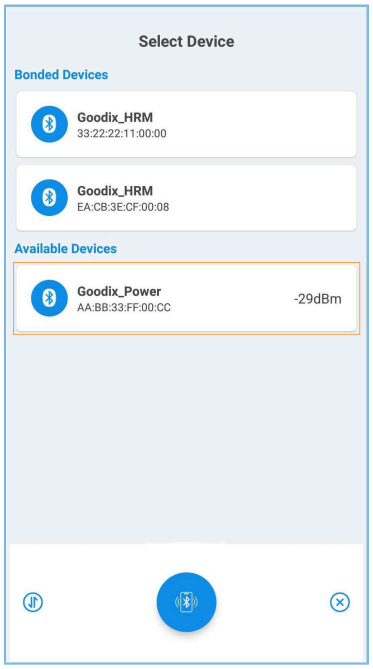
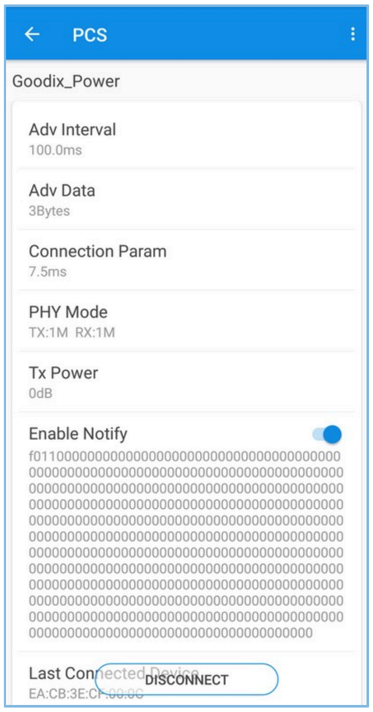


Figure 3-1 Discovering **Goodix_Power**

   (3). Tap **Goodix_Power** to establish connection, and then enter the setting page to set related power consumption measurement scenarios, including **Adv Interval**, **Adv Data**, **Connection Param**, **PHY Mode**, **Tx Power**, and **Enable Notify**, as shown in the figure below. Settings of **Adv Interval**, **Adv Data**, and **Tx Power** are valid only if advertising is restarted after the current connection is broken; settings of **Connection Param** and **PHY Mode** are valid only for the current connection. If no advertising name or service UUID is discovered, tap **Last Connected Device** to search devices based on the previous device MAC address.



PCS

Goodix_Power

Adv Interval
100.0ms

Adv Data
3Bytes

Connection Param
7.5ms

PHY Mode
TX:1M RX:1M

Tx Power
0dB

Enable Notify
f011000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
000000000000000000000000000000000000

Last Connected Device
EA:CB:3E:CF:00:0C

DISCONNECT

Figure 3-2 Setting page of power consumption measurement scenarios

(4). If the connection is broken, press **OK** on the GR55xx SK Board, to allow the device to re-initiate advertising with the configured data length and connection interval.

2. Measure GR55xx power consumption.

After measurement scenarios are set, measure GR55xx power consumption in different scenarios with Keysight installed on the PC.

Scenario 1: advertising state at an interval of 1s

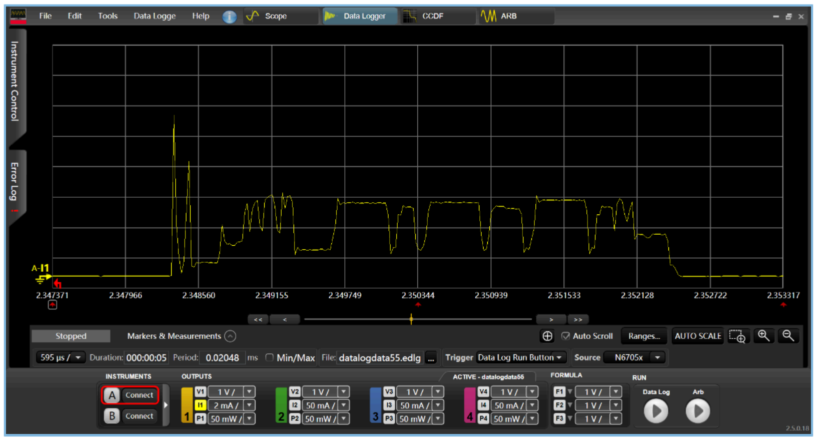


Figure 3-3 Power consumption measurement scenario 1

Scenario 2: connecting state at an interval of 200 ms

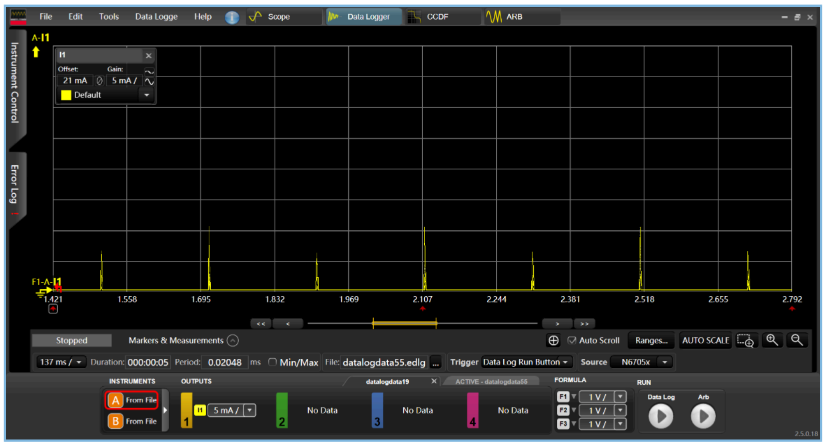


Figure 3-4 Power consumption measurement scenario 2

Users can set other power consumption measurement scenarios on demand.

4 Application Details

This chapter introduces the running procedures and major code of the PCP example.

4.1 Running Procedures

When the board downloaded with PCP example is powered on, the system initializes peripherals and power management, BLE Protocol Stack, and PCS successively. The main procedures are shown in the figure below:

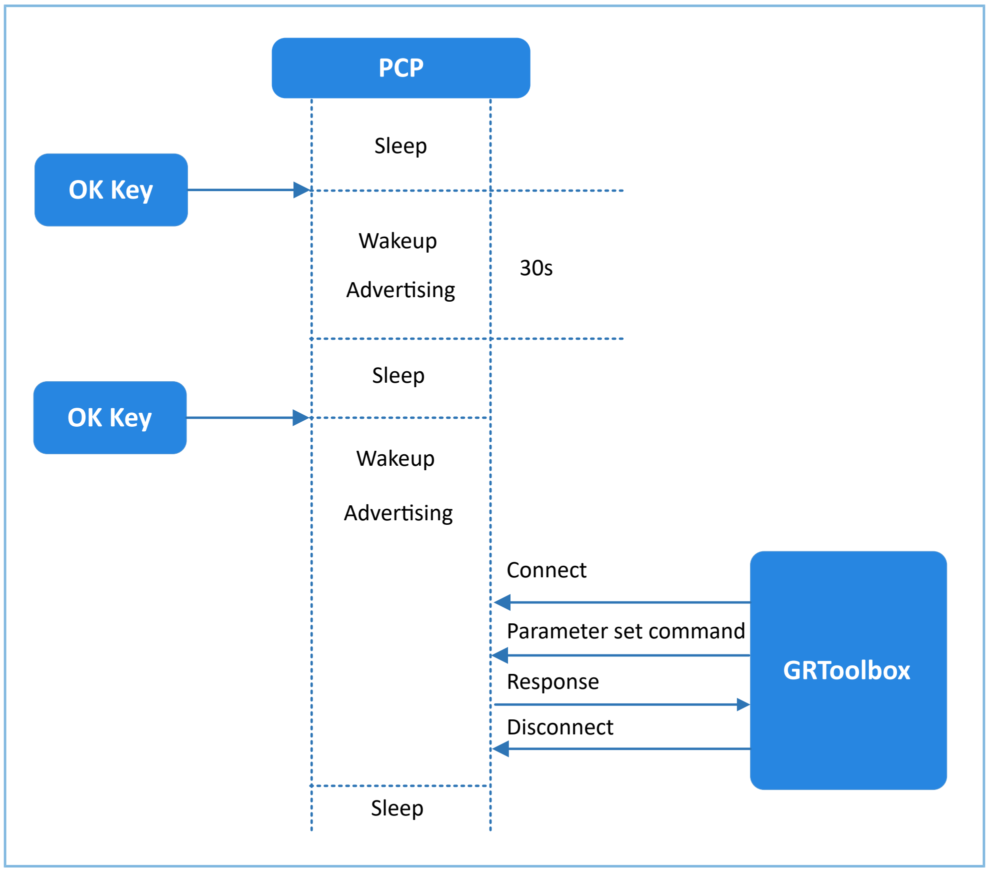


Figure 4-1 Running procedures

4.2 Major Code

The following parts elaborate on major code related to interactions between the GR55xx SK Board and GRToolbox.

4.2.1 Power Management Configuration

1. Configure the power management mode of the GR55xx SoC to Sleep Mode (PMR_MGMT_SLEEP_MODE), and configure an external wakeup source (on-board **OK** button) to wake up the GR55xx SoC and enable advertising.

   Path: `user_platform\user_periph_setup.c` under the project directory

   Name: wkup_key_init(), app_periph_init();

```
static void wkup_key_init(void)
{
    ...
    s_gpiote_param.pin = KEY_OK_PIN;
    ...

    app_gpiote_init(&s_gpiote_param, 1);
}
```

```
void app_periph_init(void)
{
    ……
    wkup_key_init();
    pwr_mgmt_mode_set(PMR_MGMT_SLEEP_MODE);
}
```

2. The following function determines whether the system enters Ultra Deep Sleep Mode or implements normal task logics after booting. It is called in the main() function.

   Path: `user_platform\user_periph_setup.c` under the project directory

   Name: is_enter_ultra_deep_sleep();

```
bool is_enter_ultra_deep_sleep(void)
{
    if (APP_IO_PIN_RESET != app_io_read_pin(APP_IO_TYPE_AON, KEY_OK_PIN))
    {
        return true;
    }

    return false;
}
```

3. The main() function determines the procedure branches and low power management of the system after booting.

```
int main(void)
{
    app_periph_init();

    if (is_enter_ultra_deep_sleep())
    {
        pwr_mgmt_ultra_sleep(0);
    }
    ble_stack_init(&s_app_ble_callback, &heaps_table);
    while (1)
    {
        pwr_mgmt_schedule();
    }
}
```

4.2.2 Command Parsing, Execution, and Response

When Setting Characteristic Value receives a command from the peer device, it reports events and related information to the application layer. The pcs_param_parse() function then can be used to parse, execute, and respond to the command. This section introduces how to set the scenarios for power consumption measurement by taking the settings of advertising interval and advertising data as examples.

1. Set advertising interval.

   Set the advertising interval value, and respond to the peer device. This value takes effect on the next advertising. For specific code, see processing code of PCS_SETTING_TYPE_ADV_INTERVAL event in the pcs_param_parse() function.

   Path: `user_app\user_app.c` under the project directory

   Name: pcs_param_parse()

```
void pcs_param_parse(uint8_t conn_idx, uint8_t *p_data, uint16_t length)
{
    ...
    switch (p_data[0])
    {
        case PCS_SETTING_TYPE_ADV_INTERVAL:
            s_gap_adv_param.adv_intv_max = BUILD_U16(p_data[1], p_data[2]);
            s_gap_adv_param.adv_intv_min = BUILD_U16(p_data[1], p_data[2]);
            response[0] = PCS_SETTING_TYPE_ADV_INTERVAL;
            response[1] = PCS_SET_PARAM_SUCCESS;
            pcs_setting_reply(0, response, 2);
            break;

        ...

        default:
            break;
    }
}
```

2. Set advertising data.

   Set advertising data, and respond to the peer device based on execution results. The data length can be set to 3, 10, 17, 24, or 31 bytes, which takes effect on the next advertising. For specific code, see processing code of PCS_SETTING_TYPE_ADV_DATA event in the pcs_param_parse() function.

   Path: `user_app\user_app.c` under the project directory

   Name: pcs_param_parse()

Note:

You need to subtract 3-byte effective length when setting user advertising data by using ble_gap_adv_param_set(). This is because Advertising Type Flag occupies 3 bytes of the advertising data.

```
void pcs_param_parse(uint8_t conn_idx, uint8_t *p_data, uint16_t length)
{
    ...
        case PCS_SETTING_TYPE_ADV_DATA:
            response[0] = PCS_SETTING_TYPE_ADV_DATA;
            response[1] = PCS_SET_PARAM_SUCCESS;

            if (PCS_SET_ADV_DATA_3B == p_data[1])
            {
                s_adv_data_set.length = 0; // 3 byte for adv type
            }
            else if (PCS_SET_ADV_DATA_10B == p_data[1])
            {
                memcpy(s_adv_data_set.adv_data, s_adv_data_10b, 7);
                s_adv_data_set.length    = 7;      // 3 byte for adv type
            }
            ...
            pcs_setting_reply(0, response, 2);
            break;
            ...
}
```

4.2.3 Enabling Notify

If the peer device enables Notify (writing the value 0x0001 to CCCD), the example application starts notifying data after it receives PCS_EVT_TX_ENABLE event; if one data transmission is completed, the example application notifies the data again after it receives PCS_EVT_DATA_SENT, and stops notifying the data when it receives PCS_EVT_TX_DISABLE.

Path: `user_app\user_app.c` under the project directory

Name: pcs_service_event_process()

```
static void pcs_service_event_process(pcs_evt_t *p_evt)
{
    switch (p_evt->evt_type)
    {
        case PCS_EVT_TX_ENABLE:
            s_is_notify_enable = true;
            pcs_tx_data_notify();
            break;
        case PCS_EVT_TX_DATA_SENT:
            if (s_is_notify_enable)
            {
                s_notify_counter++;
                pcs_tx_data_notify();
            }
            break;
        ……
        default:
            break;
    }
}
```