GR551x Sleep Mode and Power Consumption Measurement Application Note

Version: 1.9

Release Date: 2022-02-20

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 the Sleep Mode and example application details of GR551x System-on-Chips (SoCs), to help users configure the Sleep Mode.

Audience

This document is intended for:

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

Release Notes

This document is the seventh release of *GR551x Sleep Mode and Power Consumption Measurement Application Note*, corresponding to GR551x SoC series.

Revision History

Version	Date	Description
1.0	2019-12-08	Initial release
1.3	2020-03-16	Updated the contents in “Configuration of I/O Pins”.
1.5	2020-05-30	Optimized the descriptions in “Operating Modes” and “Configuration of I/O Pins”.
1.6	2020-06-30	Deleted the MSIO declaration in the code of “Configuration of I/O Pins”; optimized the descriptions and diagram in “Verify Sleep Process”; deleted the configuration descriptions about io_table in “Software Configuration”.
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 the SK Board in “Environment Setup”.
1.8	2021-08-09	Changed the section “Supported Development Platform” into “Preparation”.
1.9	2022-02-20	Modified the file name of the example firmware based on Software Development Kit (SDK) changes.

Contents

1 Introduction

This chapter introduces operating modes, wakeup sources, and the switch process between different modes of GR551x System-on-Chips (SoCs).

1.1 Operating Modes

GR551x SoCs operate in three modes: Active, Idle, and Sleep.

- Active Mode

  In this mode, the CPU of GR551x SoCs runs at full speed; users can configure and use peripherals on demand. Typical scenarios in this mode are described below:

  - The MCU Subsystem (including ARM processors, SRAM, and peripherals) remains in standby or active state.
  - The Bluetooth Subsystem (including RF Transceiver and Communication Core) remains in standby or active state.
  - The PMU Subsystem (including DC/DC, LDO, and RTC) remains in active state.

- Idle Mode

  The Idle Mode is configured by Wait For Interrupt/Wait For Event (WFI/WFE) command built in ARM processors. When this command is called, the PC pointer stops at the address of the WFI/WFE command; when an IRQ or event occurs in the system and CPU wakes up, the PC pointer points to the next command and continues implementing commands. Bluetooth tasks of GR551x SoCs are implemented based on interrupts. When the GR551x SoC is implementing Bluetooth tasks, users can configure the system to Idle Mode to reduce the power consumption; in the case of no Bluetooth tasks to be implemented, switch the Bluetooth Subsystem to POWER OFF Mode to further reduce power consumption.

- Sleep Mode

  When there are no tasks to be processed on GR551x SoCs, users can configure the system to WFI/WFE state; to further reduce the power consumption, you can configure it to Sleep Mode. In this mode, the XTAL32M clock stops running, and the following modules are powered off:

  - The MCU Subsystem (except for Retention SRAM)
  - The Bluetooth Subsystem

  In Sleep Mode, only Always-on (AON) Domain in the system is powered on, to prevent data stored in Retention SRAM from getting lost, and the modules with wakeup function are also powered on, including Bluetooth Low Energy (Bluetooth LE) Timer, Sleep Timer, Real Time Calendar, and AON GPIO.

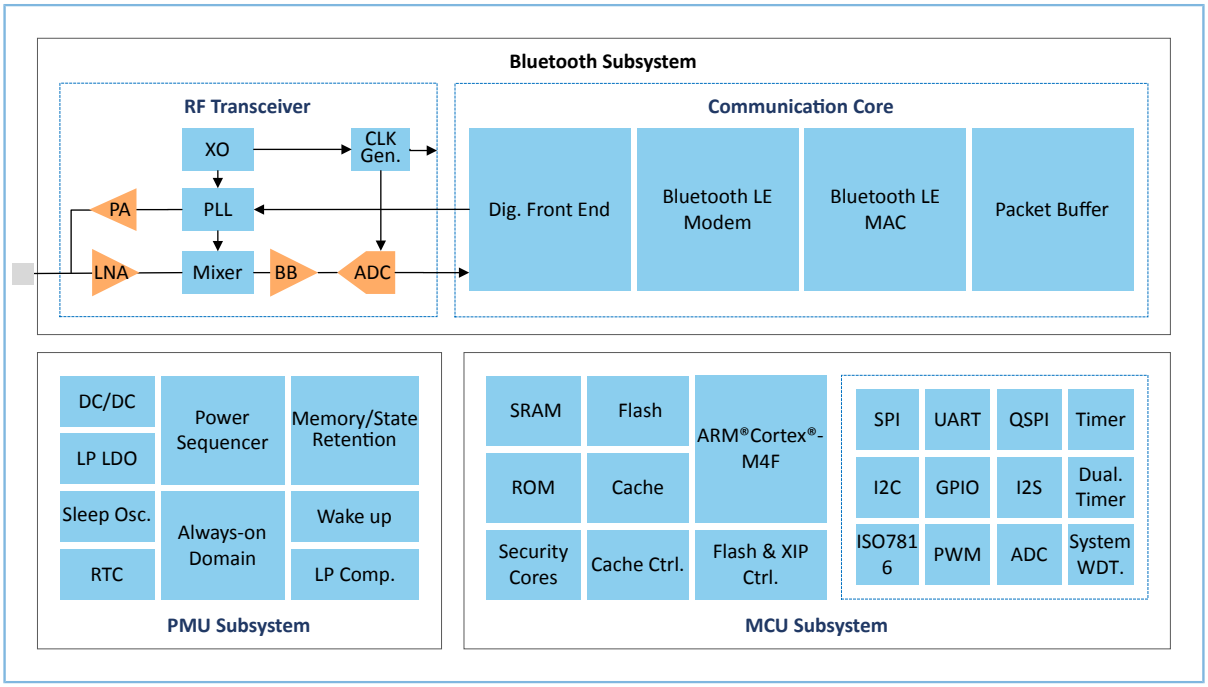

Figure 1-1 GR551x system architecture

The figure below shows the switching of system modes, which is completed automatically according to the current state.

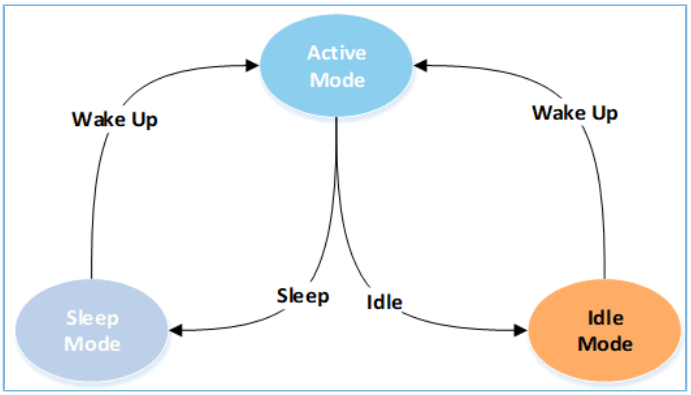

Figure 1-2 System mode switching

For detailed description and configuration of the system mode, refer to "Chapter 3 Mode Configuration".

1.2 Wakeup Sources

Wakeup sources can wake GR551x SoCs up from low power modes, including from Idle Mode to Active Mode, or from Sleep Mode to Active Mode.

Idle Mode: ARM processors go into Idle Mode and remain the state after executing a WFE or WFI command. Wakeup sources from Idle Mode to Active Mode are:

- Reset

- NVIC Event
- Debug Event

Sleep Mode: When the system is in Sleep Mode, only the modules located in Always-on Domain can wake up the system and enable it to enter the warm boot process. Wakeup sources from Sleep Mode to Active Mode are:

- Bluetooth LE Timer
- Sleep Timer
- Real Time Calendar
- AON GPIO
- LPCOMP
- Reset

Users can call the following API in an Software Development Kit (SDK) to configure wakeup sources:

```
void pwr_mgmt_wakeup_source_setup(uint32_t wakeup_source);
```

For more information about configuration of wakeup sources, see *GR551x Datasheet*.

1.3 Sleep Control Process

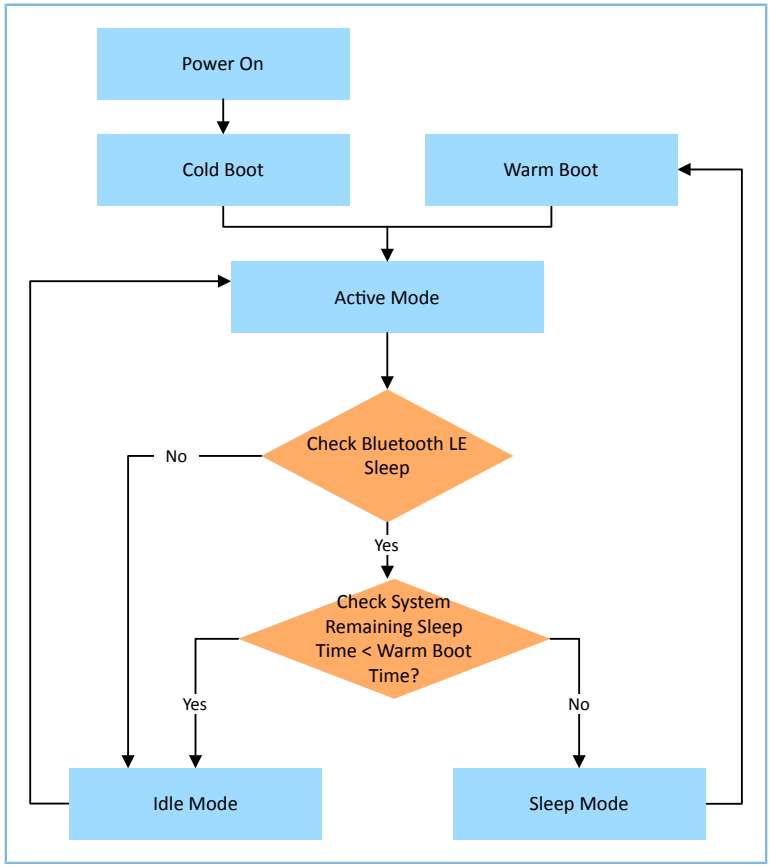

Figure 1-3 Sleep policy

A typical sleep control process (for example, users use the Bluetooth LE Subsystem and peripherals at the same time) is described below:

1. The system enters the cold boot process after it is powered on.
2. Once entering Active Mode, the system runs at full speed with high power consumption. In this mode, GR551x SoCs are able to execute any task, such as processing Bluetooth tasks or reading data from peripherals.
3. If there is no task to be processed, the system goes into the process to check Bluetooth LE Sleep state. At this time, the system will go into Idle Mode or Sleep Mode based on internal logical judgment of Bluetooth LE Core.
4. If the system is in Bluetooth LE Idle state (the Bluetooth LE module may be ready for automatic transmitting/receiving, and other peripherals are in Idle state), no processing by software is required; the Bluetooth Subsystem remains in power-on state to ensure the Bluetooth LE functionality, and the system goes into Idle Mode from Bluetooth LE Idle state. Interrupt events and debug operations can wake the system up quickly from Idle Mode to Active Mode.
5. If the system is in Bluetooth LE Sleep state, the Bluetooth Subsystem is powered off to further reduce power consumption. At this moment, if the remaining sleep time (the interval between the sleep decision time and the wakeup time of a timer; the timer can be Bluetooth LE Timer, Sleep Timer, or RTC) is found to be shorter than the warm boot time, the system goes into Idle Mode. If longer, the system goes into Sleep Mode. When the system wakes up from Sleep Mode by a wakeup source, it enters the warm boot process.

Note:

Boot time and high boot current are required for the system to enter the warm boot process from Sleep Mode. Therefore, if the sleep time is short (less than 2 ms), the power consumption reduction is unobvious. In consideration of overall system performance, if the sleep time is long, the system requires low average power consumption in Sleep Mode.

1.4 Reference Documents

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

Table 1-1 Reference documents

Name	Description
GR551x Developer Guide	Introduces GR551x SDK and how to develop and debug applications based on the SDK.
Keil User Guide	Offers detailed Keil operational instructions. Available at www.keil.com/support/man/docs/uv4/.
Bluetooth GATT Spec	Provides details about Bluetooth profiles and services. Available at www.bluetooth.com/specifications/gatt.
GR551x Datasheet	Provides overview, pinout, memory, PMU, clock, peripherals, security cores, communications subsystems, and packaging information of GR551x SoCs.

2 Demo Test and Verification

This chapter introduces how to use and verify an example project supporting GR551x Sleep Mode; all the example projects in SDK_Folder\projects\ble\ble_peripheral\ support Sleep Mode.

Note:

SDK_Folder is the root directory of GR551x SDK.

2.1 Preparation

Perform the following tasks before applying the example.

- **Hardware preparation**

Table 2-1 Hardware preparation

Name	Description
Development board	GR5515 Starter Kit Board (SK Board)
Connection cable	Micro USB 2.0 cable
Logic analyzer	Saleae Logic Analyzer and Keysight Power Analyzer

- **Software preparation**

Table 2-2 Software preparation

Name	Description
Windows	Windows 7/Windows 10
J-Link driver	A J-Link driver. Available at www.segger.com/downloads/jlink/.
Keil MDK5	An integrated development environment (IDE). MDK-ARM Version 5.20 or later is required. Available at www.keil.com/download/product/.

2.2 Verify Sleep Process

In Sleep Mode, when a Bluetooth LE Timer interrupt arrives, the system wakes up and goes into the warm boot process. Users can add GPIO initialization modules (such as GPIO Toggle) to warm boot process code, so that GPIO outputs corresponding messages when the system is in Sleep Mode or in the warm boot process, to verify whether the system works well when it switches between Sleep Mode and the warm boot process.

Details of steps for sleep process verification are provided in following sections.

2.2.1 Configure an Example Project

The example project, ble_app_pcs, is taken as a configuration example in this document.

Note:

The system is in Ultra Deep Sleep Mode when ble_app_pcs starts running. After reset, press **OK** on the board for more than 3 seconds to allow the system to initiate advertising, which lasts for 30 seconds.

Users can configure the system to Ultra Deep Sleep Mode by calling pwr_mgmt_ultra_sleep() at the SDK layer. In this mode, power domains in PMU subsystem are powered off (except for AON Domain). So is the Retention SRAM. Therefore, if the system wakes up by AON GPIO or Sleep Timer, the program executes the cold boot process.

1. Open the project, ble_app_pcs, in Keil; path: SDK_Folder\projects\ble\ble_peripheral\ble_app_pcs\Keil_5\ble_app_pcs.uvprojx.
2. Add GPIO initialization function to the code (main_init function) of the warm boot process.

   Users can use GPIO to output information, which shall be the GPIO not used in current application project, such as AON_GPIO3.

```
void main_init(void)
{
    uint32_t boot_flag = get_wakeup_flag();
    if ( COLD_BOOT == boot_flag )
    {
        extern void __main(void);
        __main();
    }
    else
    {
        ll_aon_gpio_set_pin_mode(LL_AON_GPIO_PIN_3, LL_AON_GPIO_MODE_OUTPUT);
        ll_aon_gpio_disable_it(LL_AON_GPIO_PIN_3);
        ll_aon_gpio_toggle_pin(LL_AON_GPIO_PIN_3);
        ll_aon_gpio_toggle_pin(LL_AON_GPIO_PIN_3);

        pwr_mgmt_warm_boot();
        while (1);
    }
    // Never execute here
}
```

Note:

Name: *system_gr55xx.c*

Path: SDK_Folder\toolchain\gr55xx\source\

3. Enable the sleep function: Modify the parameter value of the pwr_mgmt_mode_set() function to PMR_MGMT_SLEEP_MODE, which indicates entering the Sleep Mode.

```
void app_periph_init(void)
{
    SYS_SET_BD_ADDR(s_bd_addr);
    ble_rf_tx_mode_set(BLE_RF_TX_MODE_ULP_MODE);
    ble_rf_match_circuit_set(BLE_RF_MATCH_CIRCUIT_25OHM);
    wkup_key_init();
    pwr_mgmt_mode_set(PMR_MGMT_SLEEP_MODE);
}
```

Note:

Name: *user_periph_setup.c*

Path: SDK_Folder\projects\ble\ble_peripheral\ble_app_pcs\Src\platform\

2.2.2 Firmware Programming

The source code of the ble_app_pcs example is in SDK_Folder\projects\ble\ble_peripheral\ble_app_pcs.

You can download *ble_app_pcs.bin* to the SK Board through GProgrammer. For details, see *GProgrammer User Manual*.

Note:

- The *ble_app_pcs.bin* is in SDK_Folder\projects\ble\ble_peripheralble_app_pcs\build\.
- You can find GProgrammer in SDK_Folder\tools\GProgrammer.

2.2.3 Verify Output Waveform

Use a logic analyzer to capture the output waveform of a GPIO interface (AON_GPIO3).

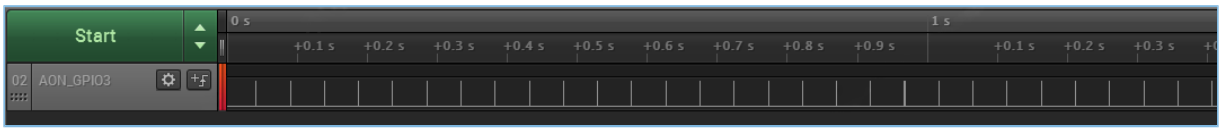

Figure 2-1 Capturing the output waveform of the GPIO interface with a logic analyzer

AON_GPIO3 outputs periodic impulse waves (see Figure 2-1), indicating Sleep Mode and the warm boot process of the system coordinate smoothly with each other. The interval between two impulses indicates the system is in Sleep Mode. When a Bluetooth LE Timer interrupt arrives, the system wakes up immediately and goes into the warm boot process. In the warm boot process, GPIO is initialized and AON_GPIO3 is toggled to a high level.

Note:

When the system is in Sleep Mode, J-Link is disconnected and users are unable to debug online. It is recommended to output external information by means of GPIO to debug the Sleep Mode function.

3 Mode Configuration

This chapter introduces how to configure Active Mode, Idle Mode, and Sleep Mode of GR551x SoCs.

3.1 Configure Sleep Mode

Users can configure the Sleep Mode flexibly according to the sleep policy, to ensure fast response and low power consumption of the system.

Take the project, ble_app_pcs, as an example; its main() function basically meets demands for general low-power applications. Users can control the sleep policy by pwr_mgmt_schedule() function. The code snippet of the main() function is as follows.

```
int main(void)
{
    // Initialize user peripherals.
    app_periph_init();

    // Initialize ble stack.
ble_stack_init(&s_app_ble_callback, &heaps_table);

    if (is_enter_ultra_deep_sleep())
    {
        pwr_mgmt_ultra_sleep(0);
    }

    // Loop
    while (1)
    {
        pwr_mgmt_schedule();
    }
}
```

Note:

Name: *main.c*

Path: SDK_Folder\projects\ble\ble_peripheral\ble_app_pcs\Src\user\

3.2 Enable Sleep Mode

Users can configure Active Mode, Idle Mode, and Sleep Mode of the system by using the pwr_mgmt_mode_set() function.

```
/**
*****************************************************************
 * @brief  Set the specified sleep mode. When the setting is completed, the system will
 *         automatically enter the specified sleep mode through the strategy.
 * @param[in] pm_mode : sleep level
 * @retval : void
*****************************************************************
 */
 void pwr_mgmt_mode_set(pwr_mgmt_mode_t pm_mode);
```

Note:

Name: *gr55xx_pwr.h*

Path: `SDK_Folder\components\sdk`

The parameter pm_mode of pwr_mgmt_mode_set() function is used to set Sleep Mode of the system.

Table 3-1 Values of pwr_mgmt_mode_t

Value of pm_mode	Description
PMR_MGMT_ACTIVE_MODE	The system goes into Active Mode and remains in this state, and runs at full speed.
PMR_MGMT_IDLE_MODE	The system is permitted to go into Idle Mode and switch between Active Mode and Idle Mode. • If there is any task to be processed, the system remains in Active Mode. • If not, the system remains in Idle Mode, waits to be woken up when an interrupt arrives, and then enters Active Mode.
PMR_MGMT_SLEEP_MODE	The system is permitted to go into Sleep Mode and switch between Active Mode, Idle Mode, and Sleep Mode according to the sleep policy. • If there is any task to be processed, the system remains in Active Mode. • If not, the system goes into Idle Mode or Sleep Mode based on predictable sleep time, which can be predicted by both AON Timer and Bluetooth LE Timer.

3.2.1 Configure Active Mode

Set the value of pm_mode of pwr_mgmt_mode_set() function to PMR_MGMT_ACTIVE_MODE, to keep the system in Active Mode. Unused memory blocks are configured automatically during system initialization. Users can also control the power supply and clocks of unused memory blocks, to reduce power consumption in Active Mode.

3.2.2 Configure Idle Mode

Set the value of pm_mode of pwr_mgmt_mode_set() function to PMR_MGMT_IDLE_MODE, so as to allow the system to switch between Active Mode and Idle Mode, depending on whether any task is to be processed. In the main() function of the example project, the pwr_mgmt_schedule() function automatically disables corresponding modules according to the sleep policy. For example, if no Bluetooth task is to be processed, the ble_core_sleep() function will power off the Bluetooth Subsystem, and enable the MCU Subsystem to automatically go into Idle Mode after it completes current tasks.

3.2.3 Configure Sleep Mode

Generally, the warm boot model of the system shall by applied in the Sleep Mode.

- Sleep Mode

  Set the value of pm_mode of pwr_mgmt_mode_set() function to PMR_MGMT_SLEEP_MODE, so as to enable the system to switch between Active Mode, Idle Mode, and Sleep Mode according to the sleep policy. In the

main() function of the example project, the pwr_mgmt_schedule() function automatically powers all the power domains off (except for Always-on Domain), and sets part of RAM to Retention RAM to lower power consumption according to the sleep policy.

- Warm boot

  The driver interface supports recovery from warm boot, and the warm boot process requires no user involvement.

Note:

Name: *system_gr55xx.c*

Path: `SDK_Folder\toolchain\gr55xx\source\`

---

4 Configuration of I/O Pins

4.1 Configuration Principle

The GR551x SoCs provide on-chip programmable pull-up/pull-down resistors. The resistors can replace external ones to save space and cost, and can also keep any unused GPIO pins from floating, to avoid excess current to flow from the VDDIO supply and/or unexpected behavior.

Note:

The GPIO pin cannot have both a pull-up and a pull-down resistors enabled simultaneously. The value of the resistor is approximately 100 kΩ. The default state upon power-on of a GR551x SoC is for a pull-down resistor to be enabled on all GPIO pins.

Follow the principles below to enable/disable on-chip pull-up/pull-down resistors of GR551x SoCs:

1. If a GPIO pin is in floating state and any external interference voltage is on the pin, enable the internal pull-up/pull-down resistors.
2. If a GPIO pin is unused, enable the internal pull-down resistor.
3. If a GPIO pin is configured as an input and is pulled up/down by an external device, disable the internal pull-up/pull-down resistors.
4. If a GPIO pin is configured as an input and the connected external device is in high impedance state, enable the internal pull-up/pull-down resistors.
5. If a GPIO pin is configured as an output, disable the internal pull-up/pull-down resistors.

The principle descriptions and circuit diagrams are provided below.

1. If a GPIO pin is in floating state, any external interference voltage on the pin may lead to breakover of the input gate, resulting in excess current flow from the VDDIO supply through the input gate. Sometimes this current can be large and could cause unexpected problems with the chip operation, leading to the GPIO pin being in floating state.

   The simplified schematic of a GPIO pin on a GR551x SoC is shown in Figure 4-1.

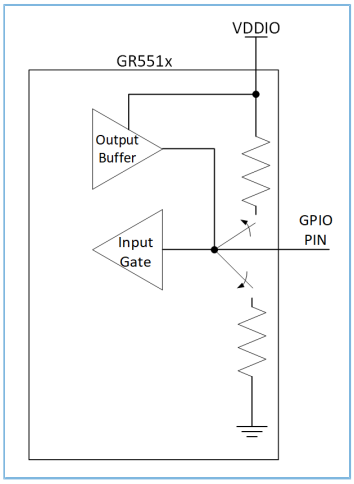

Figure 4-1 GPIO pin schematic diagram

2. In general, if a GPIO pin is configured as an input and connected to an external drive, the pull-up/pull-down resistors shall be disabled; otherwise, VDDIO current leakage may occur.

Note:

However, when the external drive enters high impedance state, the GR551x input remaining in floating state may cause excess current flow. Therefore, when the external drive enters high impedance state, enable corresponding pull-up/pull-down resistors.

---

(1). If a GPIO pin is being used as an input with the on-chip resistor enabled as a pull-up resistor and the external drive driving the pin is driving a low level on the pin, current will flow from the VDDIO supply through the pull-up resistor to ground (through the external device), as shown in Figure 4-2.

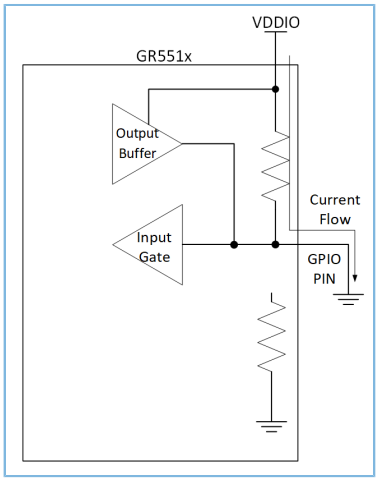

Figure 4-2 Current flow for a low driven input

The pull-up resistor is approximately 100 kΩ. If the VDDIO voltage is 3.3 V, the additional current flow from the VDDIO supply would be 33 µA just for this misconfigured GPIO pin. If there are additional misconfigured GPIO pins, each pin will also contribute an additional 33 µA.

(2). If a GPIO pin is configured as an input, the pull-down resistor is enabled and the external device is driving a high level on the GPIO pin, then no current will flow through the VDDIO supply, but it will flow through the VDDIO supply of the external device, as shown in Figure 4-3.

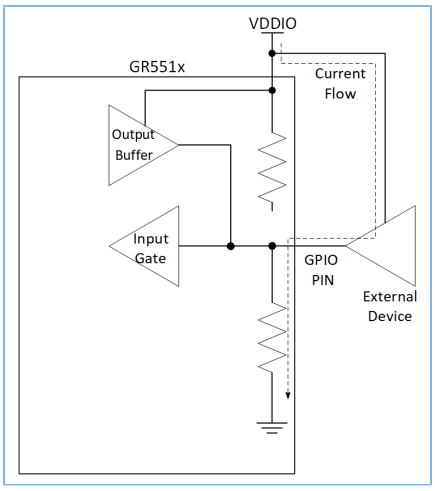

Figure 4-3 Current flow for a high driven input

3. If a GPIO pin is being used as an output, the pull-up/pull-down resistors should be disabled. Otherwise, there is a chance that electrical leakage may occur and excess current may flow through the enabled pull-up/pull-down resistors.

Note:

However, if a GPIO pin is configured as an output, and the internal pull-up/pull-down resistors are always disabled, then both the input to the GR551x input gate and the input to the external device will float, which shall be avoided. To avoid this situation, the pull-up/pull-down resistors shall be enabled during high impedance.

---

(1). If a GPIO pin is configured as an output driving high with a pull-down resistor enabled, current flows from the VDDIO supply to the pull-down resistor, as shown in Figure 4-4.

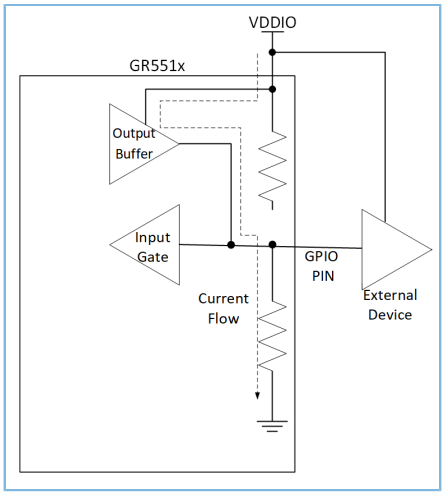

Figure 4-4 GPIO as output driving high

(2). If a GPIO pin is configured as an output driving low with a pull-up resistor enabled, the leaked current flows through the pull-up resistor and low-level output buffer to ground, as shown in Figure 4-5.

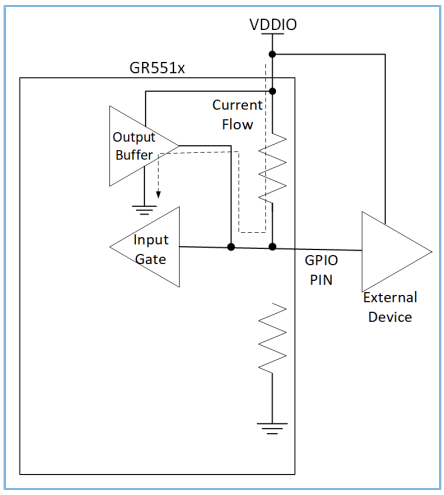

Figure 4-5 GPIO as output driving low

4.2 Software Configuration

After a GR551x SoC is powered on and initialized for the first time, initialization states of three types of I/O pins are described below:

- GPIO: input with the internal pull-down resistor enabled
- AON GPIO: input with the internal pull-down resistor enabled
- MSIO: analog input

To prevent electrical leakage due to uncertain I/O state and the subsequent higher power consumption of the system, users can select proper pull-up/pull-down resistors for peripheral applications on demand. For example:

- For common I/O pins that are unused, enable internal pull-down resistors based on Principle 2 in "Section 4.1 Configuration Principle".
- External pull-up resistors are available for the I2C module. Therefore, SCL/SDA pins shall be set to floating state based on Principle 3 in "Section 4.1 Configuration Principle".
- For the SPI/QSPI module, the initialization state of pins can be configured based on CPHA and CPOL in communications protocols. For example:
  - MODE0: Enable the pull-down resistor for the SCLK/CS pins.
  - MODE3: Enable the pull-up resistor for the SCLK/CS pins, and enable pull-down resistors for other pins.

5 Power Consumption Measurement

This section introduces how to measure power consumption with a power analyzer (Keysight N6705C DC Power Analyzer), and analyzes measurement results by using the SK Board as the device under test.

5.1 Measurement Principle

When a power analyzer is used to supply power (3.3 V) to a chip, the current consumed for running the chip flows through the power analyzer, and then the power analyzer records the current signal.

According to the formula:

P = U x I (P = Power consumption; U = Supply voltage; I = Current)

U refers to the constant voltage (3.3 V); the value of P is in direct proportion to that of I. Therefore, the current curve recorded by the power analyzer can be regarded as a power consumption curve.

5.2 Environment Setup

As shown in the figure below, connect the SK Board (with the firmware downloaded as described in "Section 2.2.2 Firmware Programming") to Keysight Power Analyzer (take Keysight N6705C as an example).

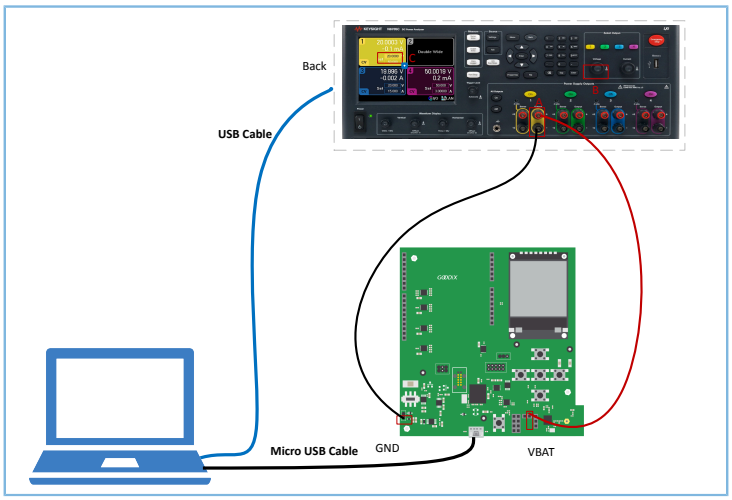

Figure 5-1 Connection between SK Board and Keysight Power Analyzer

1. Power on Keysight Power Analyzer, and turn the knob as shown at Location B in the figure above, to set the output channel voltage until the display at Location C shows 3.3 V.
2. As shown in the figure above, connect the positive pole ("+") at Location A of Keysight to VBAT pin (J10 Pin 2, as shown in the figure below) on the SK Board, to power the GR551x SoC on the board.

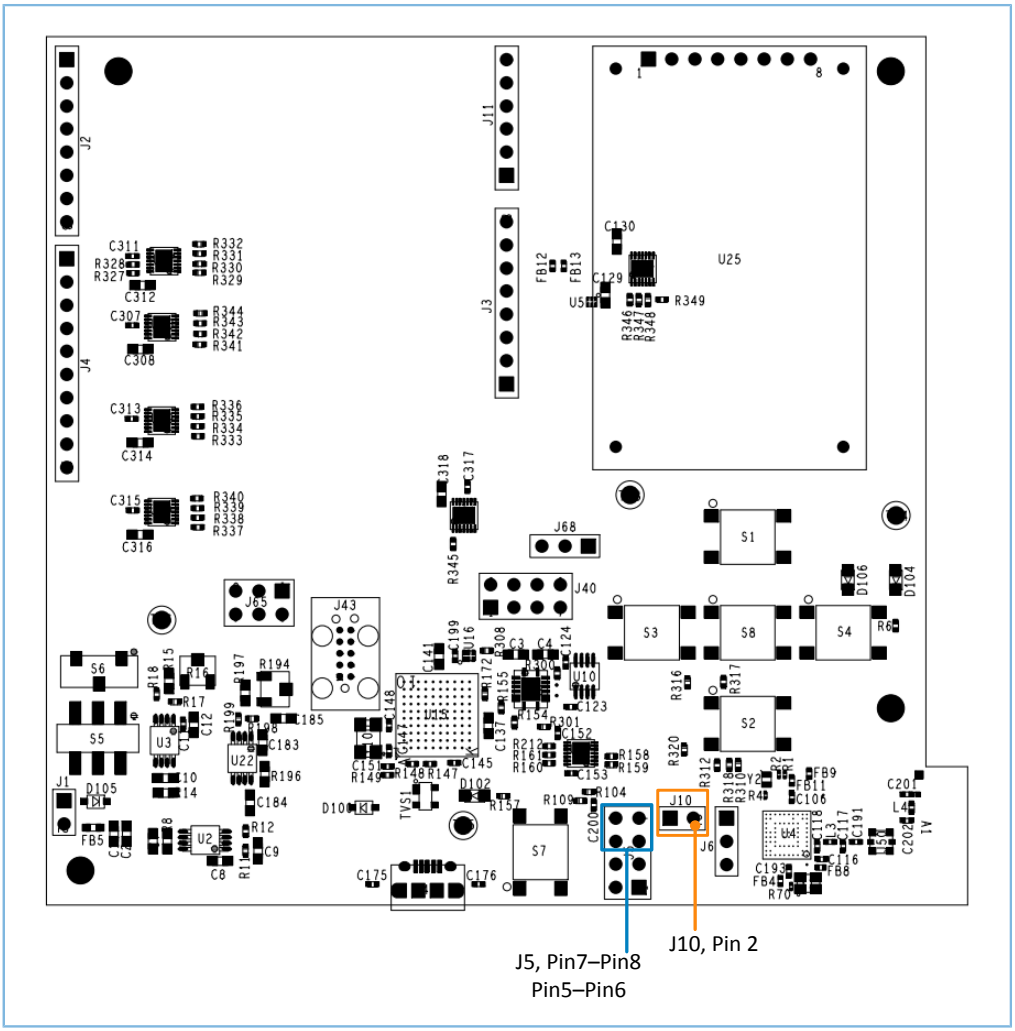

Figure 5-2 Hardware layout of SK Board (top view)

3. Connect the negative pole (“-”) at Location A of Keysight to GND pin on the SK Board, to form a circuit loop.
4. Connect the USB port on the SK Board to any USB port of the PC, to power the chip peripheral circuit on the board.
5. Connect Keysight Power Analyzer to the PC with a Micro USB cable.

Note:

Before power consumption measurement, remove the jumper caps on Pin 5–Pin 6 and Pin 7–Pin 8 of J5 (as shown in Figure 5-2), and remove the LCD in the upper-right corner on the board, to prevent abnormal measurements due to current leakage from VDDIO. For details, see “Section 4.1 Configuration Principle”.

5.3 Measurement

When a basic environment is established, perform power consumption test using KeySight Power Analyzer and bundled software on the PC.

The figure below shows the current waveform of a typical advertising task: After the system wakes up from Sleep Mode, it processes TX/RX tasks in Channel 37/38/39; when the tasks are finished, it goes into Sleep Mode again.

Figure 5-3 Power consumption test waveform of SK Board advertising task

The figure below shows the current waveform of a typical connection task, the current sequence of which is similar to that of the advertising task.

Figure 5-4 Power consumption test waveform of SK Board connection task

6 FAQ

This chapter lists possible problems with corresponding solutions when the system is in the warm boot process or in Sleep Mode.

6.1 Why Do Peripherals Operate Abnormally During Warm Boot?

- Description

  When the system is woken up by warm boot, peripherals cannot work normally.

- Analysis

  The peripheral module is a part of the MCU Subsystem Domain, which is powered off in Sleep Mode, causing the loss of previous register configurations.

- Solution

  It is recommended to use the App driver APIs. When the system is woken by warm boot, it automatically initializes the peripheral module. If the HAL driver APIs are used, users need to add peripheral initialization code in the hal_pm_resume function to finish initialization.

6.2 Why Is J-Link Disconnected?

- Description

  When Sleep Mode is enabled, the application program goes into Sleep Mode. At this point, J-Link is disconnected from the application program, resulting in debugging failure.

- Analysis

  In Sleep Mode, all power domains (except for Always-on Domain), including J-Link, are powered off by pmu_sleep. Therefore, J-Link is unable to stay connected, so that debugging of the Sleep Mode function fails.

- Solution

  In early development phases, it is recommended to develop application programs in Active Mode or Idle Mode. In later development phases, users only need to verify whether functions in Sleep Mode work normally.

6.3 Why Does J-Link Fail to Be Connected to PC?

- Description

  After the application with the Sleep Mode function enabled is downloaded to the SK Board, a failure occurs when users try to connect J-Link to the PC again.

- Analysis

  The runtime of the warm boot process is short, and the SK Board keeps in Sleep Mode for most of the time, causing J-Link unable to connect to and thus interact with the PC normally.

- Solution

  Configure the BOOT_LONG_TIME field in *custom_config.h*. After this field is enabled, the system adds delay time by 1 second in the cold boot process, so that users can connect to J-Link and download code to refresh firmware promptly during the added delay time.