GR551x Power Mode and Power Consumption Measurement Application Note

Version: 2.1

Release Date: 2022-09-30

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: Floor 12-13, Phase B, Tengfei Industrial Building, Futian Free Trade Zone, Shenzhen, China

TEL: +86-755-33338828 Zip Code: 518000

Website: www.goodix.com

Preface

Purpose

This document introduces the power modes and power management modes of GR551x System-on-Chips (SoCs), with detailed example projects provided, to help users efficiently configure power modes of the system.

Audience

This document is intended for:

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

Release Notes

This document is the ninth release of *GR551x Power 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 jumpers 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.
2.0	2022-06-24	Optimized descriptions in the document.
2.1	2022-09-30	• Updated the code file name and path, as well as code in "Configuring an Example Project" and "Enabling Power Management Mode". • Deleted a redundant note in "Configuring Power Management Mode".

Contents

1 Introduction

This chapter introduces the power modes, wake-up sources in low power modes, and the process for switching between the modes of GR551x System-on-Chips (SoCs).

1.1 Power 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. Typical scenarios in this mode are described below:

  - The MCU subsystem (including Arm processor, 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

  This mode is implemented by Wait For Interrupt/Wait For Event (WFI/WFE) command built in the Arm processor. When the WFI/WFE command is called, the program counter (PC) points to the address of the WFI/WFE command; when an interrupt request (IRQ) or event occurs and CPU wakes up, the PC points to the next command and subsequent commands are executed. The system detects the use of peripherals/Bluetooth subsystem/Timer and decides whether to enter this mode.

- Sleep mode

  In this mode, a high-frequency clock source HFXO_32M stops working; MCU subsystem (except for Retention SRAM) and Bluetooth subsystem are powered off. Only the always-on (AON) domains are powered, to make sure data in Retention SRAM is not lost, and to power the modules or I/Os that can wake up the system from sleep mode: Bluetooth LE Timer, Sleep Timer, Real Time Calendar, and AON GPIOs. When no task is pending, the system can enter sleep mode.

The system automatically switches between the power modes, as shown in the figure below.

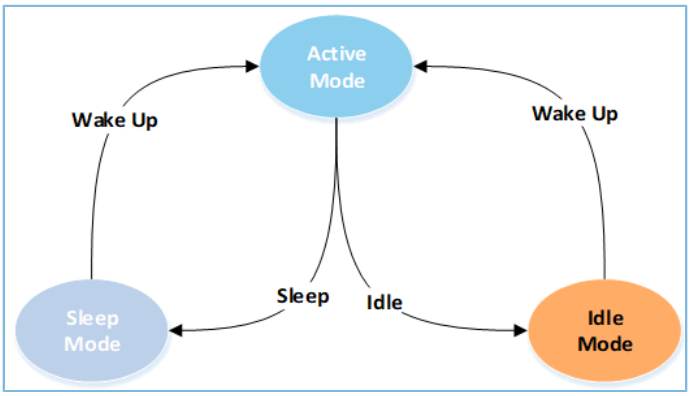

Figure 1-1 Mode switching mechanism

1.2 Wake-up Sources

Wake-up sources are used to wake up the system from low power modes: from idle mode to active mode, or from sleep mode to active mode.

Wake-up sources from idle mode to active mode:

- Reset
- NVIC Event
- Debug Event

Wake-up sources from sleep mode to active mode:

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

Users can call the following API in the GR551x Software Development Kit (SDK) to configure wake-up sources:

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

For more information about configuration of wake-up sources, see *GR551x Datasheet*.

1.3 Power Mode Switching Process

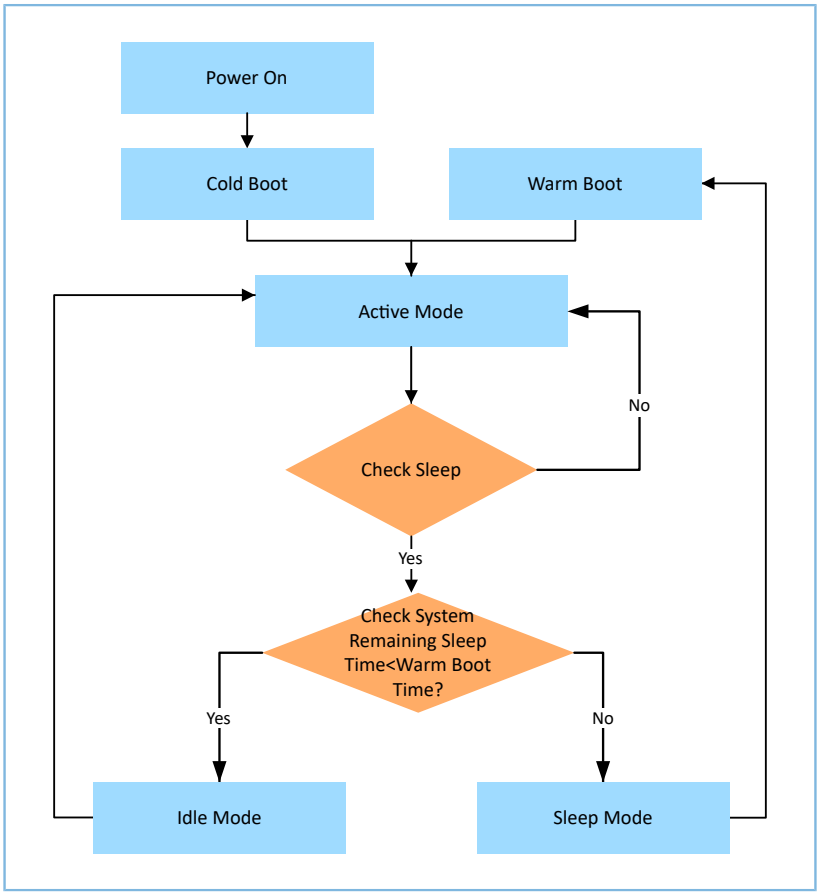

Figure 1-2 Power mode switching process

A typical power mode switching process is as follows (suppose Bluetooth LE subsystem and peripherals are used at the same time):

1. The system enters the cold boot process after power-on.
2. Enter active mode.
3. When no task is pending, the system starts to check whether the system is idle (Check Sleep in the process diagram above)
4. If the result is Yes, detect the remaining idle time (for example, the due time for Bluetooth LE Timer, Sleep Timer, and RTC). If the due time is shorter than warm boot process, the system enters idle mode; if the due time is longer than warm boot process, the system enters sleep mode, and waits to be woken up by a wake-up source.
5. If the result is No, the system stays in active mode.

Note:

Boot time and high boot current are required for the system to enter the warm boot process from sleep mode. When sleep time is relatively short (for example, shorter than 2s), the power consumption reduced is hardly noticeable. Therefore, to ensure overall performance, the system enters sleep mode only when sleep time is longer, so that the average power consumption can be reduced.

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 https://www.keil.com/support/man/docs/uv4/.
Bluetooth GATT Spec	Provides details about Bluetooth profiles and services. Available at https://www.bluetooth.com/specifications/gatt.
GR551x Datasheet	Provides overview, pinout, memory, PMU, clock, peripherals, security cores, communication 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 https://www.segger.com/downloads/jlink/.
Keil MDK5	An integrated development environment (IDE). MDK-ARM Version 5.20 or later is required. Available at https://www.keil.com/download/product/.

2.2 Verification of Sleep Mode Wake-up Process

In sleep mode, when a Bluetooth LE Timer interrupt occurs, the system is woken up and goes into the warm boot process. Users can add a GPIO initialization module (such as GPIO toggling) to the warm boot process code, so that GPIO outputs corresponding messages when the system is in sleep mode or in the warm boot process, to check whether the system works well when it switches from sleep mode to warm boot.

Details of verifying the process are provided in following sections.

2.2.1 Configuring an Example Project

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

Note:

- Ultra deep sleep mode is a type of deep sleep mode, in which all power domains (except for AON domains) are powered off, including Retention SRAM. After system wake-up with AON GPIO or Sleep Timer, cold boot will be initiated. To enter ultra deep sleep mode, users shall call pwr_mgmt_ultra_sleep() in the SDK.
- Run ble_app_pcs, and the system will call pwr_mgmt_ultra_sleep() to enter ultra deep sleep mode. AON GPIO will be set as the wake-up source. To wake up the system from ultra deep sleep mode, press **OK** on the board for 3 seconds or more, and an advertising will be initiated and last for 30 seconds.

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 the 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_GPIO_3.

```
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);

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

Note:

Name: *gr_platform.c*

Path: SDK_Folder\platform\soc\common

Add `#include "gr55xx_ll_aon_gpio.h"` at the beginning of the file, to avoid compilation errors.

3. To set the power management mode to automatic sleep management mode, set the input parameter of pwr_mgmt_mode_set() to PMR_MGMT_SLEEP_MODE.

```
void app_periph_init(void)
{
    SYS_SET_BD_ADDR(s_bd_addr);
    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:

- *ble_app_pcs.bin* is in SDK_Folder\projects\ble\ble_peripheral\ble_app_pcs\Keil_5\Listings\.
- You can find GProgrammer in SDK_Folder\tools\GProgrammer.

2.2.3 Verification of Output Waveform

Use a logic analyzer to capture the output waveform of a GPIO (AON_GPIO_3).

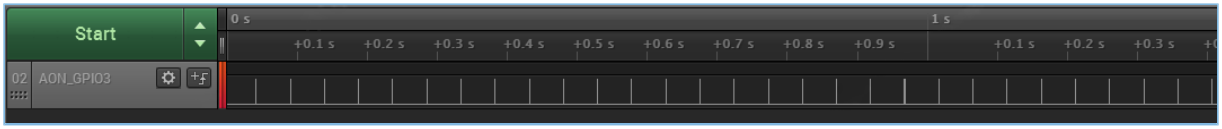

Figure 2-1 Capturing the output waveform of AON_GPIO_3 with a logic analyzer

AON_GPIO_3 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 occurs, the system wakes up immediately and goes into the warm boot process. In the warm boot process, GPIO is initialized and AON_GPIO_3 is toggled to high level.

Note:

When the system is in sleep mode, J-Link connection will be broken and users will be unable to debug online. It is recommended to output information through GPIOs to debug the sleep mode.

3 Power Management Mode

This chapter introduces how to set the system to automatic sleep management mode (PMR_MGMT_SLEEP_MODE), automatic idle management mode (PMR_MGMT_IDLE_MODE), and active management mode (PMR_MGMT_ACTIVE_MODE).

3.1 Implementing Power Management Mode

The main() function of a project (such as ble_app_pcs) is used for normal low-power application scenarios. Users can run pwr_mgmt_schedule() so that the system can automatically choose an appropriate power management mode. Code snippet of the main() function is as follows:

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

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

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

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

Note:

Name: *main.c*

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

3.2 Configuring Power Management Mode

Users can set the system to one of the power management modes by calling pwr_mgmt_mode_set().

```
/**
*********************************************************************
 * @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() is used to set the power management mode of the system.

Table 3-1 Values of pwr_mgmt_mode_t

Value of pm_mode	Description
PMR_MGMT_ACTIVE_MODE	Sets the system to active management mode. The system stays in active mode and runs at full speed.
PMR_MGMT_IDLE_MODE	Sets the system to automatic idle management mode. The system automatically switches between active mode and idle mode. • If there is task pending, the system chooses active mode. • If no, the system chooses idle mode, and will enter active mode after wake-up on an interrupt.
PMR_MGMT_SLEEP_MODE	Sets the system to automatic sleep management mode. The system automatically switches between active mode, idle mode, and sleep mode. • If there is task pending, the system chooses active mode. • If no, the system chooses idle mode or sleep mode based on the estimated sleep time with AON Timer or Bluetooth LE Timer.

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 reduce footprint and cost and can also keep any unused GPIO pins from being floating to avoid excess current leakage from VDDIO.

Note:

Avoid using both a pull-up resister and a pull-down resistor for the same GPIO. The resistance of each resistor is approximately 100 kΩ. After a GR551x SoC is powered on, pull-down resistors will be enabled for all GPIO pins by default.

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 VDDIO through the input gate. Sometimes, the current can be large and leads to errors in chip operation.

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

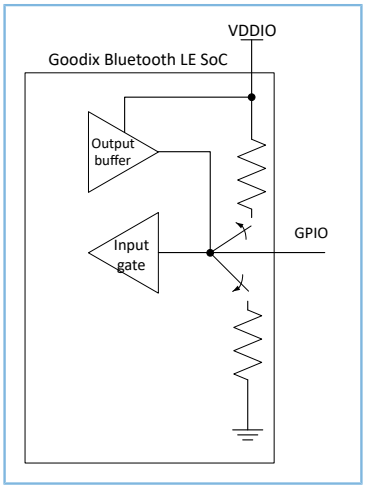

Figure 4-1 GPIO pin schematic diagram

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

Note:

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

---

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

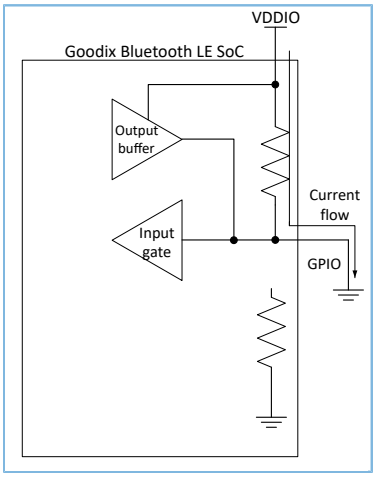

Figure 4-2 Current flow for a low driven input

The pull-up resistor is approximately 100 kΩ. When the VDDIO voltage is 3.3 V, if one GPIO pin is configured improperly, there will be excess current leakage (about 33 μA) from VDDIO; if multiple GPIO pins are configured improperly, each pin will have a leakage current of 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 there is no current leakage from VDDIO. However, current will be generated from VDDIO through the external device, as shown in Figure 4-3.

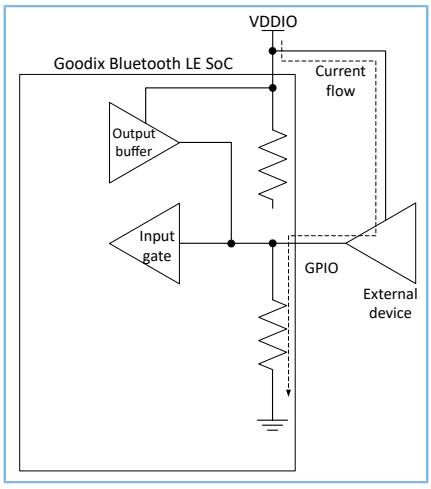

Figure 4-3 Current schematic for a high driven input

3. If a GPIO pin is 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, both the input to the GR551x input gate and the input to the external device will be floating. 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 VDDIO to the pull-down resistor, as shown in Figure 4-4.

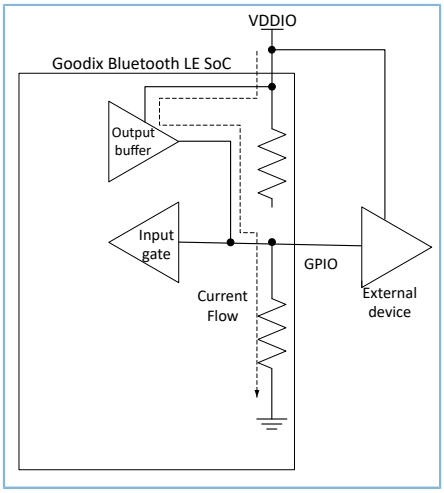

Figure 4-4 Current schematic for GPIO as output for high voltage level

(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.

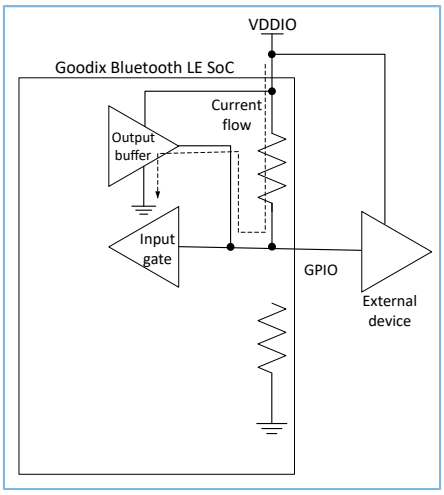

Figure 4-5 Current schematic for GPIO as output for low voltage level

4.2 Software Configuration

After a GR551x SoC is powered on and initialized, 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, 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".
- The initialization state of pins in the SPI/QSPI module can be configured based on CPHA and CPOL in communication 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 chapter introduces how to measure power consumption with a power analyzer (Keysight N6705C DC Power Analyzer) and analyze measurement results by using the SK Board as the device under test.

5.1 Measurement Principle

When a power analyzer is used to supply the SoC, the current consumed for running the SoC 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)

In the formula, U refers to the constant voltage; 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).

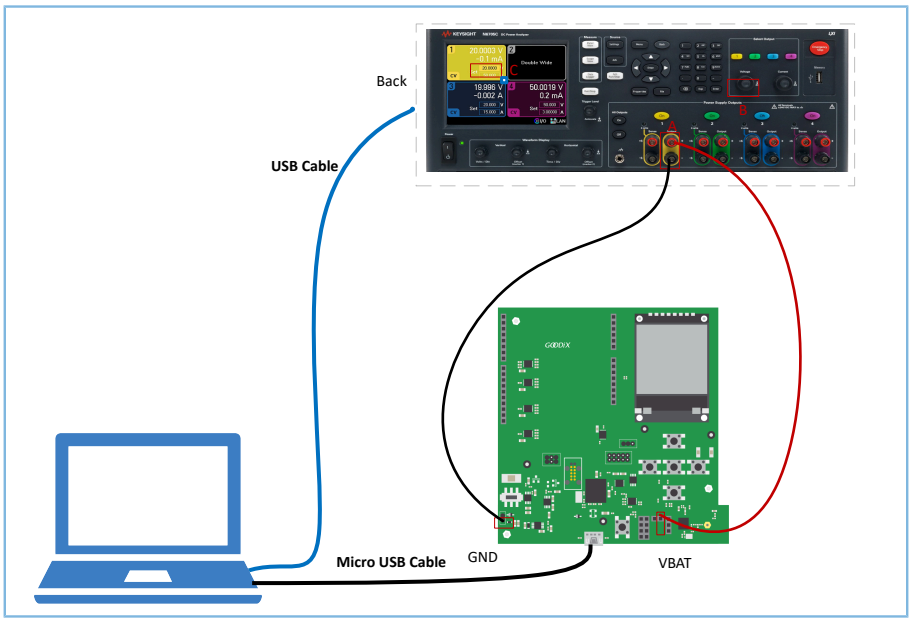

Figure 5-1 Connection between SK Board and Keysight Power Analyzer

1. Power on Keysight N6705C, 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 voltage at 3.3 V.
2. As shown in the figure above, connect the positive pole ("+") at Location A of Keysight N6705C to VBAT pin (J10 Pin 2, as shown in the figure below) on the SK Board, to power the GR551x SoC on the board.

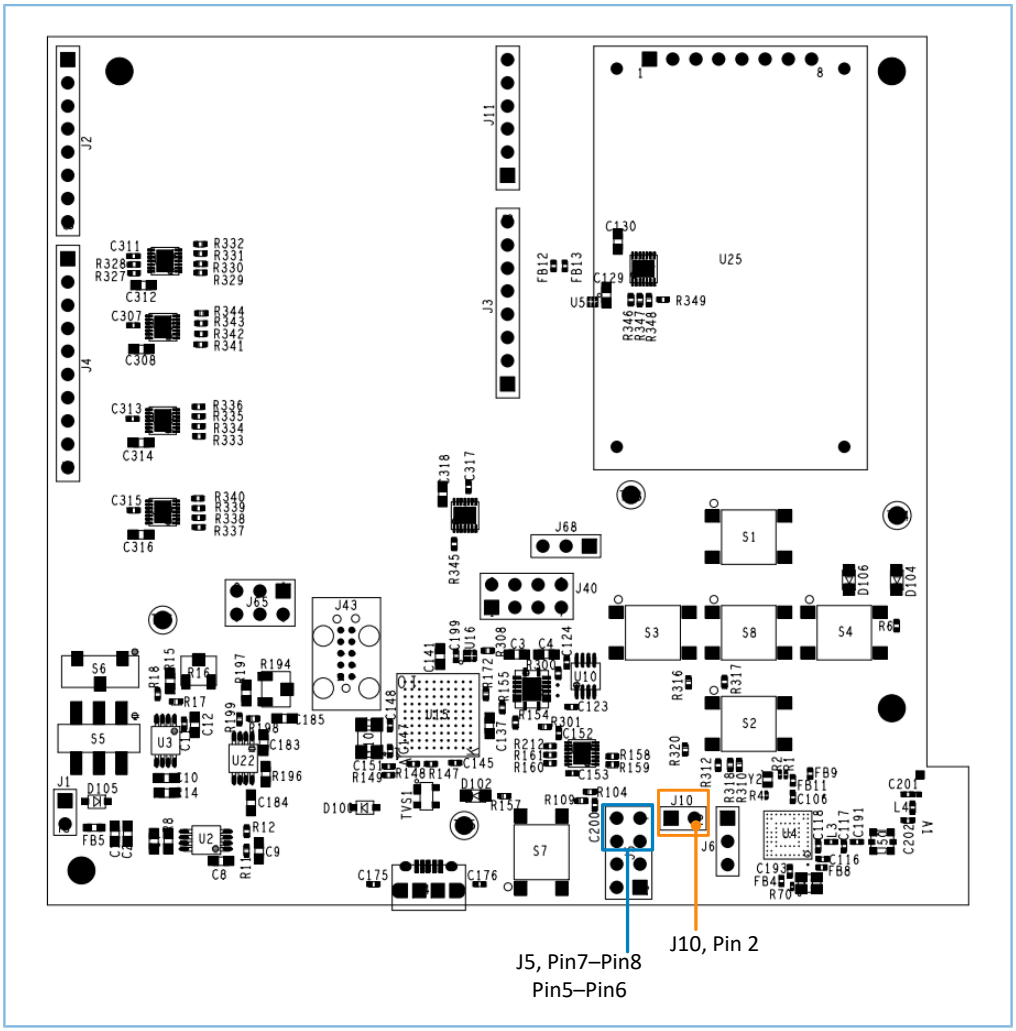

Figure 5-2 Hardware layout of SK Board (top view)

3. Connect the negative pole (“-”) at Location A of Keysight N6705C 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 N6705C to the PC with a Micro USB cable.

Note:

Before power consumption measurement, remove the jumpers 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, measure the power consumption using Keysight N6705C and a PC with relevant software.

The figure below shows the current waveform of a typical advertising task: After the system is woken up from sleep mode, it processes TX/RX tasks in channels 37/38/39; when the task is finished, it goes back to sleep mode.

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

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 for connection task of SK Board

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 Work Abnormally During Warm Boot?

- Description

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

- Analysis

  Peripheral modules are in the MCU subsystem, which is powered off in sleep mode, causing loss of register configurations.

- Solution

  It is recommended to use the APP driver APIs. When the system is woken up and enters warm boot, it automatically initializes peripheral modules. It is not recommended to use HAL driver APIs, because calling HAL driver APIs after system wake-up may result in operation failures due to the peripheral registers not being restored.

6.2 Why Is J-Link Disconnected?

- Description

  When automatic sleep management mode is enabled, the application goes into sleep mode. At this point, J-Link is disconnected from the application, resulting in debugging failure.

- Analysis

  In sleep mode, all power domains (except for AON domains), including J-Link, are powered off. That is why J-Link is unable to stay connected.

- Solution

  At the initial development stage, it is recommended to choose active management mode or automatic idle management mode. At later stages, users only need to verify whether the sleep functionality works normally.

6.3 Why Does Connection Through J-Link Fail?

- Description

  When I try to connect the SK Board to the PC through J-Link after downloading applications with automatic sleep mode enabled to the SK Board, the connection fails.

- Analysis

  The SK Board keeps in sleep mode for most of the time, and therefore J-Link cannot interact with the PC normally. That is why the connection fails.

- Solution

  Configure the BOOT_LONG_TIME field in *custom_config.h*. After this field is enabled, the system adds 500 ms delay time to the cold boot process, so that users can establish connection through J-Link and download code to refresh firmware promptly during this period.