3.1 Light up LED

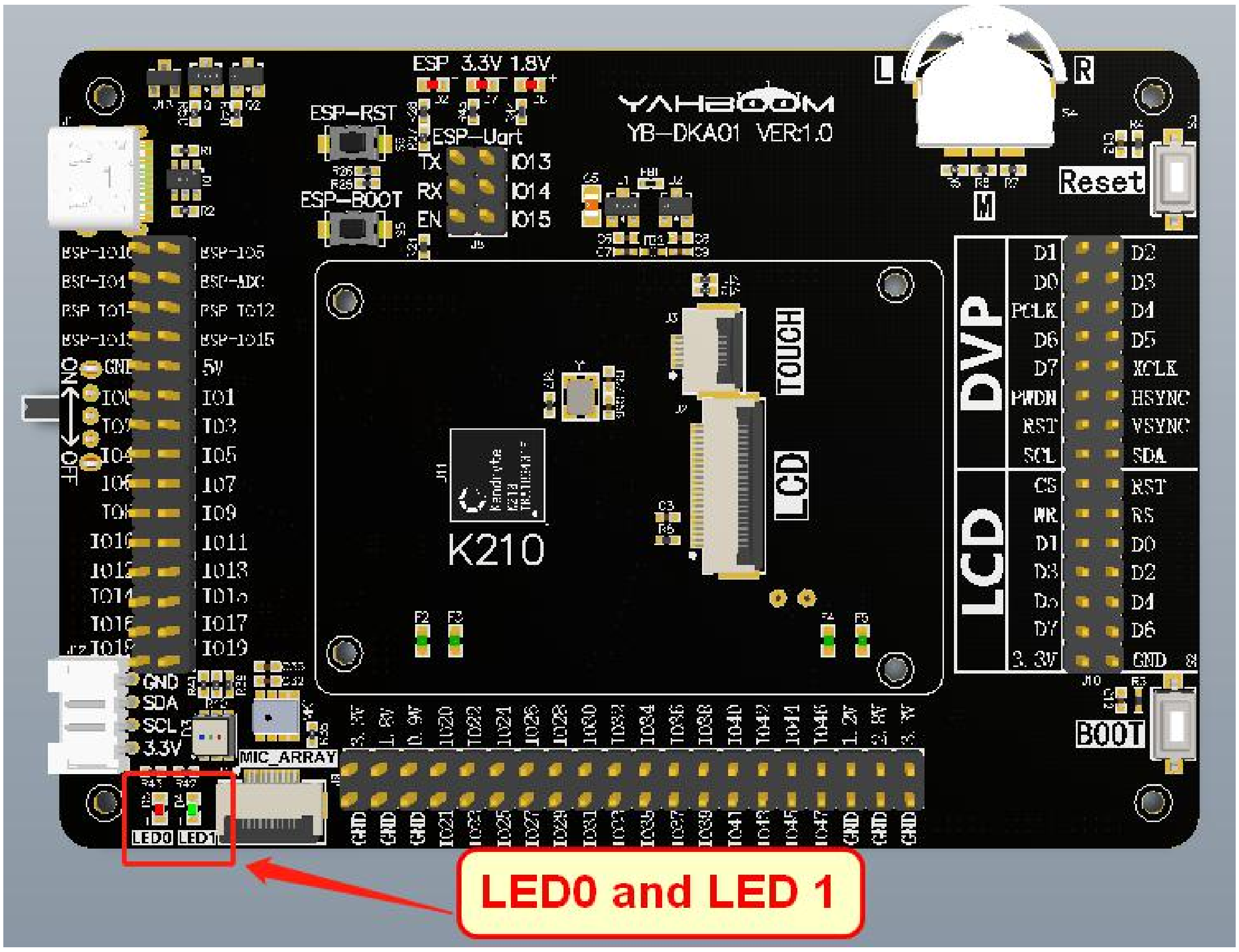
1. Experiment purpose

In this lesson mainly learns the most basic functions of K210, FPIOA pin mapping and light up LED lights.

2.Experiment preparation

2.1 components

LED0 and LED1. As shown below.



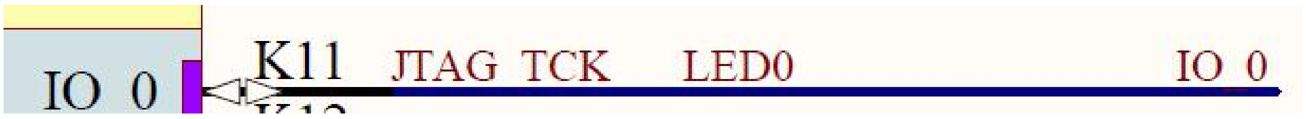
2.2 Component characteristics

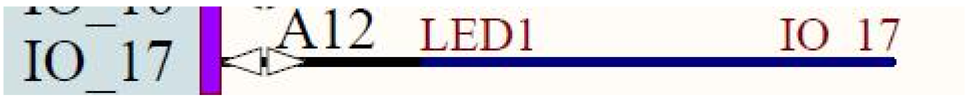
LED0 is a red light, LED1 is a green light.

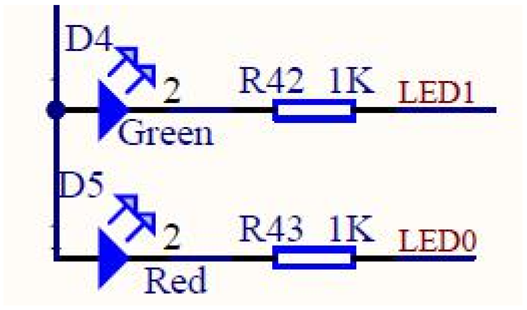
Both LED lights will be on at low level and will be off at high level.

2.3 Hardware connection

LED0 and LED1 have been soldered on the K210 development board by default. LED0 is connected to IO0, and LED1 is connected to IO17.







2.4 SDK API function

Header file is gpio.h

There are 8 general-purpose GPIO, they are use the same interrupt source. They can be configured to input and output signals, or trigger IO port total interrupt and edge trigger and level trigger. Each IO can be assigned to one of the 48 pins on FPIOA.

We will provide the following ports for users.

- gpio_init: GPIO port initialization
- gpio_set_drive_mode: set GPIO port input or output mode
- gpio_set_pin: set GPIO pin level high/low
- gpio_get_pin: read GPIO pin level

2.5 What is FPIOA?

FPIOA (Field Programmable IO Array) . It allows users to map 255 internal functions to 48 free IOs on the periphery of the chip:

- Support IO programmable function selection
- Support IO output 8 kinds of drive capability selection
- Support IO internal pull-up resistor selection
- Support IO internal pull-down resistor selection
- Supports IO input internal schmitt trigger settings
- Support IO output slope controlling
- Support internal input logic level setting

3. Experimental principle

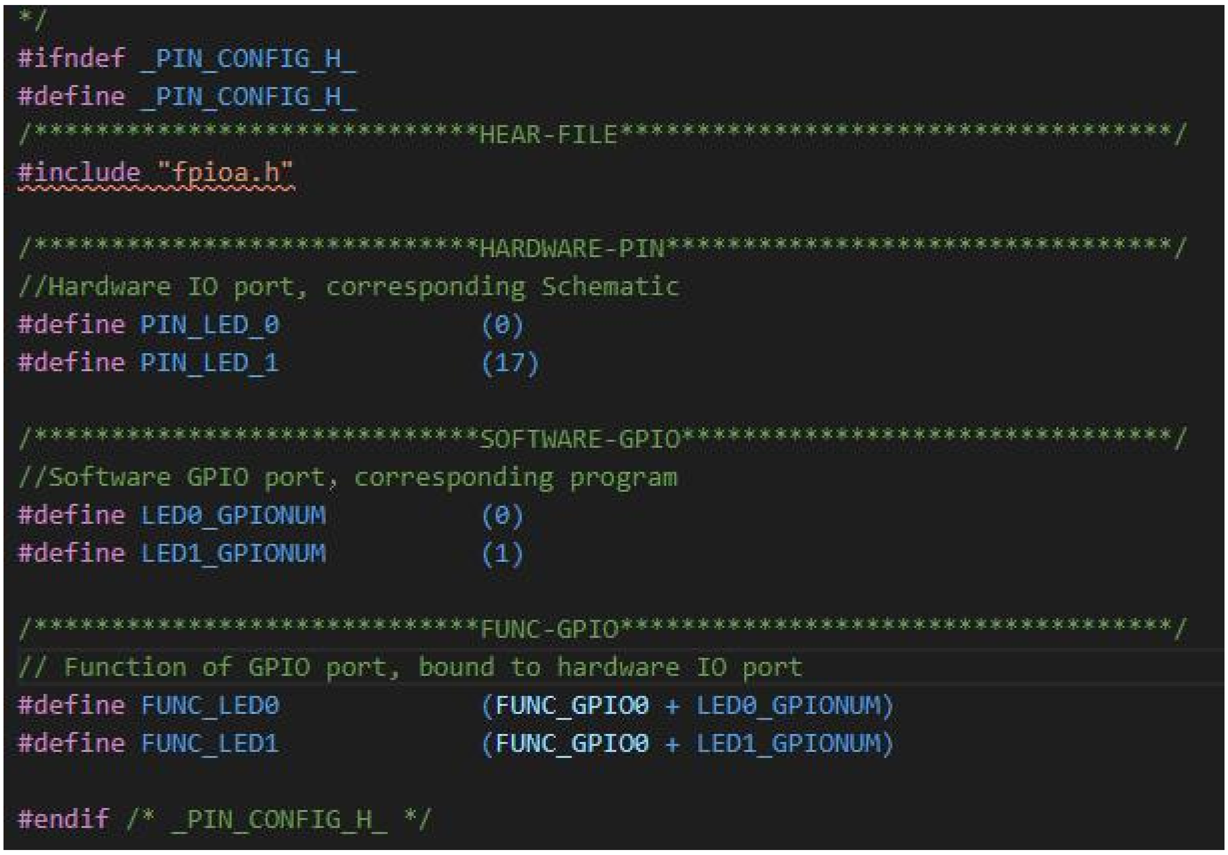
LED (Light Emitting Diode), also known as light emitting diode, is a solid-state semiconductor device that can convert electrical energy into visible light. It can directly convert electricity into light. The inside of the LED is a semiconductor chip, one end of the chip is the negative pole, and the other end is connected to the positive pole of the power supply. The entire chip is encapsulated by epoxy resin. We only need to input positive voltage to the positive pole, and make negative pole connect to GND form a loop to light up the LED.

The semiconductor wafer is composed of two parts. One part is a P-type semiconductor, in which holes dominate. Other end is an N-type semiconductor, mainly electrons inside. When these two semiconductors are connected, a P-N junction is formed between them. When the current acts on the chip through the wire, the electrons will be pushed to the P area. The electrons and holes will recombine in the P area. Then, emit energy in the form of photons. This is the principle of LED light emission. The wavelength of light is the color of LED, which is determined by the material forming the P-N junction.

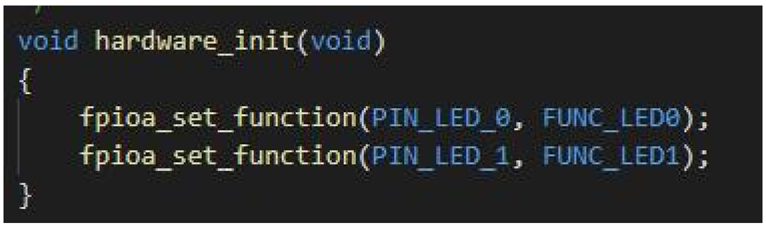
4. Experiment procedure

4.1 According to the above hardware connection pin diagram, K210 hardware pins and software functions use FPIOA mapping relationship.

!Note: All the operations in the program are software pins, so you need to map the hardware pins to software GPIO functions. Then, you can directly operate the software GPIO.



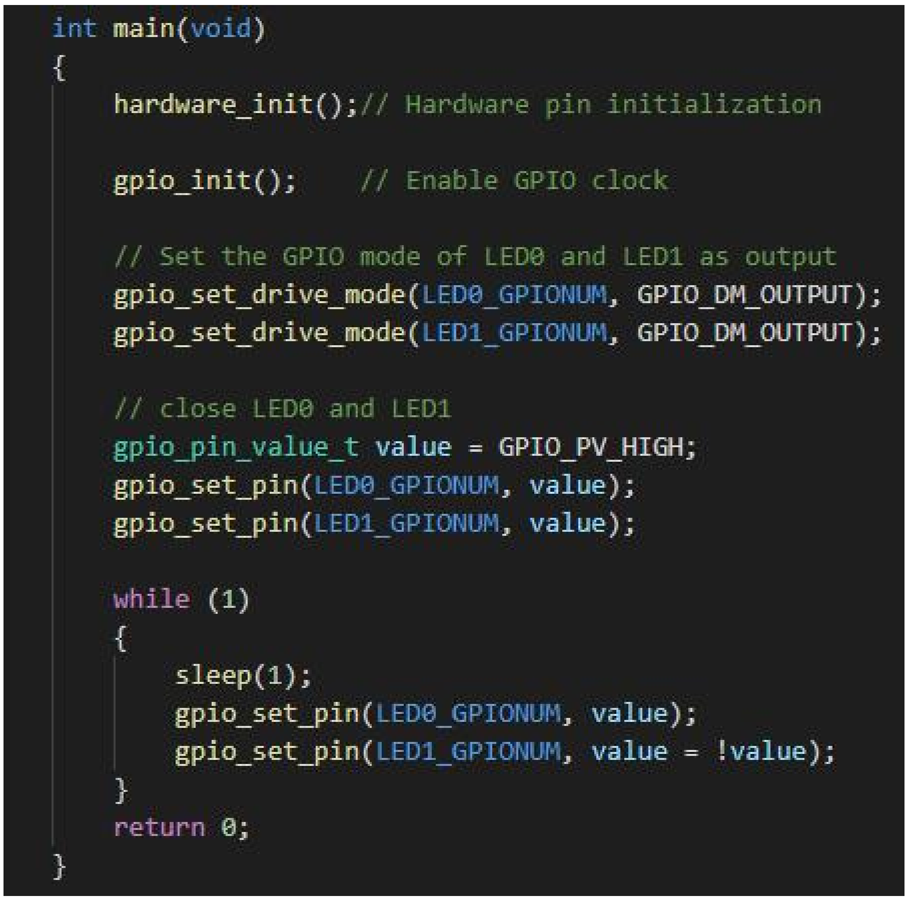
```
*/
#ifndef _PIN_CONFIG_H_
#define _PIN_CONFIG_H_
/*****************************HEAR-FILE************************************/
#include "fpioa.h"

/*****************************HARDWARE-PIN*********************************/
//Hardware IO port, corresponding Schematic
#define PIN_LED_0             (0)
#define PIN_LED_1             (17)

/*****************************SOFTWARE-GPIO********************************/
//Software GPIO port, corresponding program
#define LED0_GPIONUM          (0)
#define LED1_GPIONUM          (1)

/*****************************FUNC-GPIO************************************/
// Function of GPIO port, bound to hardware IO port
#define FUNC_LED0             (FUNC_GPIO0 + LED0_GPIONUM)
#define FUNC_LED1             (FUNC_GPIO0 + LED1_GPIONUM)

#endif /* _PIN_CONFIG_H_ */
```



```
void hardware_init(void)
{
    fpioa_set_function(PIN_LED_0, FUNC_LED0);
    fpioa_set_function(PIN_LED_1, FUNC_LED1);
}
```

4.2 The main function is the entry function of the K210 chip. All programs are executed from here. First, initialize the hardware pins. Then, enable the GPIO clock. Next, set LED0 and LED1 to output mode, and then set the level of LED0 and LED1 to high Level, indicating the off state.

Finally, in the while loop, change the value every second, which can make LED0 and LED1 light up alternately.



```c
int main(void)
{
    hardware_init();// Hardware pin initialization

    gpio_init();    // Enable GPIO clock

    // Set the GPIO mode of LED0 and LED1 as output
    gpio_set_drive_mode(LED0_GPIONUM, GPIO_DM_OUTPUT);
    gpio_set_drive_mode(LED1_GPIONUM, GPIO_DM_OUTPUT);

    // close LED0 and LED1
    gpio_pin_value_t value = GPIO_PV_HIGH;
    gpio_set_pin(LED0_GPIONUM, value);
    gpio_set_pin(LED1_GPIONUM, value);

    while (1)
    {
        sleep(1);
        gpio_set_pin(LED0_GPIONUM, value);
        gpio_set_pin(LED1_GPIONUM, value = !value);
    }
    return 0;
}
```

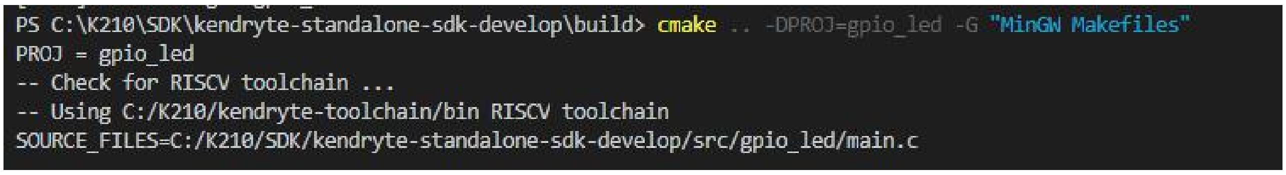
4.3 Compile and debug, burn and run

Copy the gpio_led folder to the src directory in the SDK.

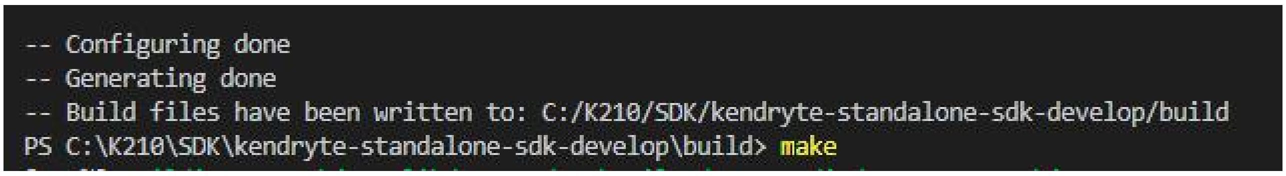
Then, enter the build directory and run the following command to compile.

cmake .. -DPROJ=gpio_led -G "MinGW Makefiles"

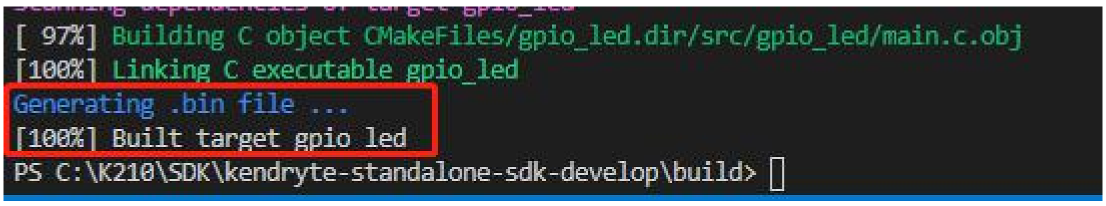
make



```
PS C:\K210\SDK\kendryte-standalone-sdk-develop\build> cmake .. -DPROJ=gpio_led -G "MinGW Makefiles"
PROJ = gpio_led
-- Check for RISCV toolchain ...
-- Using C:/K210/kendryte-toolchain/bin RISCV toolchain
SOURCE_FILES=C:/K210/SDK/kendryte-standalone-sdk-develop/src/gpio_led/main.c
```



```
-- Configuring done
-- Generating done
-- Build files have been written to: C:/K210/SDK/kendryte-standalone-sdk-develop/build
PS C:\K210\SDK\kendryte-standalone-sdk-develop\build> make
```

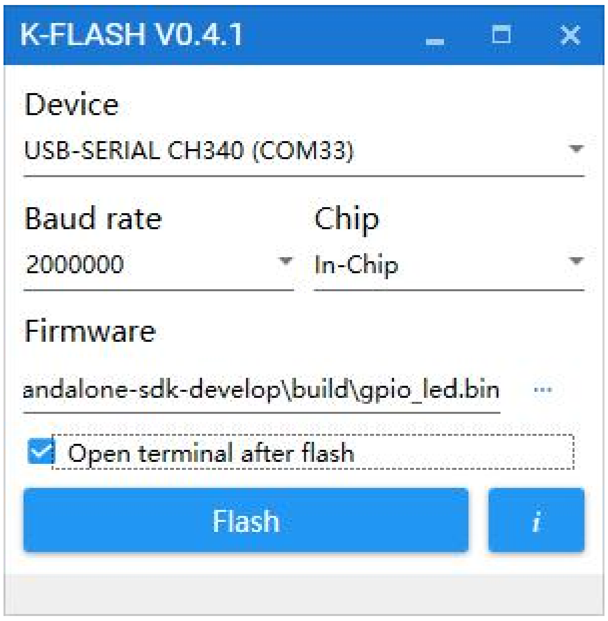


```
[ 97%] Building C object CMakeFiles/gpio_led.dir/src/gpio_led/main.c.obj
[100%] Linking C executable gpio_led
Generating .bin file ...
[100%] Built target gpio_led
PS C:\K210\SDK\kendryte-standalone-sdk-develop\build>
```

After the compilation is complete, the gpio_led.bin file will be generated in the build folder.
We need to use the type-C data cable to connect the computer and the K210 development board.
Open kflash, select the corresponding device, and then burn the gpio_led.bin file to the K210 development board.

!Tip:

Choose 'Open terminal after flash', so that after the firmware is burned, a terminal will pop up to view the debugging information.



K-FLASH V0.4.1

Device: USB-SERIAL CH340 (COM33)

Baud rate: 2000000

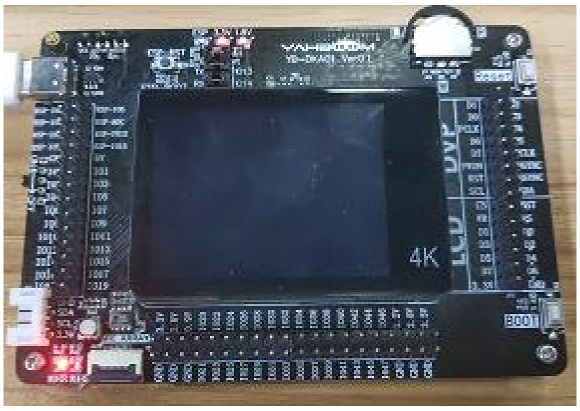
Chip: In-Chip

Firmware: andalone-sdk-develop\build\gpio_led.bin

[✓] Open terminal after flash

5. Experimental phenomenon

The two lights LED0 and LED1 light up alternately. Green light is on for 1 second --> green light is off --> red light is on for 1 second --> red light is off --> green light is on for 1 second. Keep the loop in this state.



6. Experiment summary

6.1 The K210 chip uses the FPIOA programmable array, so you need to map the hardware IO ports each time before using the hardware IO ports. And in programming, we will call software GPIO after software mapping.

6.2 K210 chip and other single-chip microcomputer chips are also run from the main function.

6.3 Before using GPIO, you need to set the input and output mode of GPIO.

6.4 The LED light is on at low level. It is on when the LED pin is set to low level, and it is off when it is set to high level.

Appendix -- API

Header file is gpio.h

1) gpio_init

Function: Initialize GPIO.

Function prototype: **int gpio_init(void)**

Return value:

Return value	Description
0	succeed
!0	fail

2) gpio_set_drive_mode

Function: Set GPIO drive mode

Function prototype: **void gpio_set_drive_mode(uint8_t pin, gpio_drive_mode_t mode)**

Parameter:

Parameter name	Description	Input/Output
pin	GPIO	Input
mode	GPIO drive mode	Input

Return value: No

3) gpio_set_pin

Function: set GPIO pin level high/low

Function prototype: **void gpio_set_pin(uint8_t pin, gpio_pin_value_t value)**

Parameter:

Parameter name	Description	Input/Output
pin	GPIO	Input
value	GPIO value	Input

Return value: No

4) gpio_get_pin

Function: Get the GPIO pin value.

Function prototype: **gpio_pin_value_t gpio_get_pin(uint8_t pin)**

Parameter:

Parameter name	Description	Input/Output
pin	GPIO	Input

Return value: Get the GPIO pin value.

Type of data

The related data types and data structures are defined as follows:

- gpio_drive_mode_t: GPIO drive mode.
- gpio_pin_value_t: GPIO value.

gpio_drive_mode_t

Description

GPIO drive mode.

Define

```
typedef enum _gpio_drive_mode
{
	GPIO_DM_INPUT,
	GPIO_DM_INPUT_PULL_DOWN,
	GPIO_DM_INPUT_PULL_UP,
	GPIO_DM_OUTPUT,
} gpio_drive_mode_t;
```

Member

Member name	Description
GPIO_DM_INPUT	Input
GPIO_DM_INPUT_PULL_DOWN	Input pull_down
GPIO_DM_INPUT_PULL_UP	Input pull_up
GPIO_DM_OUTPUT	Output

gpio_pin_value_t

Description: GPIO value.

Define

```
typedef enum _gpio_pin_value
{
	GPIO_PV_LOW,
	GPIO_PV_HIGH
} gpio_pin_value_t;
```

Member

Member name	Description
GPIO_PV_LOW	Low
GPIO_PV_HIGH	High