TG7120B

GPIO Application Note

Version 0.2

Author:

Security: Public

Date: 2020.9

目录

图表目录

1 简介

GPIO，全称 General-Purpose Input/Output（通用输入输出），是一种软件运行期间能够动态配置和控制的通用引脚。
不同型号的芯片支持的 GPIO 数量有差异。GPIO 上电默认属性如下表：

QFN32	Default mode	Default IN_OUT	IRQ/ Wakeup	FULLMUX	ANA	KSCAN
P02	SWD_IO	IN	√	√		mk_in[1]
P03	SWD_CLK	IN	√	√		mk_out[1]
P07	GPIO	IN	√	√		mk_in[10]
P09	GPIO	IN	√	√		mk_out[4]
P10	GPIO	IN	√	√		mk_in[4]
P11	GPIO	IN	√	√	√	mk_out[11]
P14	GPIO	IN	√	√	√	mk_out[2]
P15	GPIO	IN	√	√	√	mk_in[2]
XTAL16M_I	XTALI(ANA)	ANA			√	
XTAL16M_O	XTALO(ANA)	ANA			√	
P18	GPIO	IN	√	√	√	mk_in[5]
P20	GPIO	IN	√	√	√	mk_out[5]
P34	GPIO	IN	√	√		

表 1：GPIO 上电默认属性配置

1.1 P2/P3

P2(SDW_IO)、P3(SDW_CLK)可以接调试器，接调试器调试程序时，这两个 IO 口的非调试功能会受影响。
因此使用 P2、P3 时，硬件不要接调试器。

1.2 GPIO 模式

GPIO 模式是最常用的模式，可配置为输出并输出高低电平，可以配置为输入读取外部的高低电平。
当配置为输入时，支持中断和唤醒。

1.2.1 GPIO 输出

配置相应 GPIO 方向寄存器为输出，向输出寄存器写 1 或 0，即可输出高或低电平。

1.2.2 GPIO 输入

配置相应 GPIO 方向寄存器为输入，读取输入寄
存器的值，即可获取当前 GPIO 的电平状态。
如使用中断，需要打开 GPIO 中断使能功能，并配置中断产生条件。
如使用唤醒，需要打开 GPIO 唤醒使能功能，并配置唤醒产生条件。

1.2.3 GPIO retention

当 GPIO 做输出时，可配置 retention 功能。retention 默认是关闭的。
retention 打开时，系统休眠时，GPIO 的输出特性和输出值保持不表。retention 关闭时，系统休眠时，GPIO 会恢复默认输入态。
比如，P00 运行时配置为 GPIO 输出且输出 1，如系统进入休眠时候，该 GPIO 会变为输入态，此时不会输出 1。如果想让该 GPIO 在休眠时也保持输出 1 这种状态，那么需要在休眠前配置该 GPIO 的 retention 功能。

1.2.4 GPIO 上下拉电阻

每个 GPIO 支持四种上下拉电阻配置：

- 浮空:高阻态。
- 强上拉:上拉到 AVDD33，高电平，驱动电流大。上拉电阻 150kΩ 欧姆。
- 弱上拉:上拉到 AVDD33，高电平，驱动电流小。上拉电阻 1MΩ 欧姆。
- 下拉:下拉到地，低电平，下拉电阻 100kΩ 欧姆。

上下拉电阻硬件默认值：

- P03：下拉。
- 其他 GPIO：浮空。

1.2.5 中断和唤醒

所有 GPIO 支持中断和唤醒。
中断支持电平触发和边沿触发，唤醒支持边沿触发。
注意事项：GPIO 做唤醒源，不支持片外电阻，只能用片内电阻。

1.3 GPIO FULLMUX 模式

所有 GPIO 都支持 GPIO FULLMUX 功能，可根据应用 GPIO 配置为 UART，I2C、PWM 等功能。比如，在烧录模式下，UART 用到的就是 P9(tx)，P10(rx)。这里的 P9、P10 就是 GPIO FULLMUX 功能。在应用程序中，我们可以将其他 GPIO 复用为 UART 功能，也可以将 P9、P10 复用为 PWM 等其他功能。GPIO 和复用关系可以灵活配置的。

1.4 ANALOG 模式

只有 P11、P14、P15、P18、P20 支持模拟功能。

- adc:采集引脚上的电压，单端支持的引脚有 P11、P14、P15、P20，差分支持的引脚有 P20P15。

2 GPIO 典型应用

2.1 GPIO 输出

配置对应 GPIO 方向寄存器(swporta_ddr)为输出，设置输出寄存器(swporta_dr)为 0 或 1。
驱动有对应的 API，直接调用即可。
比如：将 P11 设置为输出，并不停输出 0 和 1。

```
/* define dev */
gpio_dev_t led;

void hal_gpio_output(void)
{
    int ret = -1;

    drv_pinmux_config(P11, PIN_FUNC_GPIO);
    /* gpio port config */
    led.port = P11;

    /* set as output mode */
    led.config = OUTPUT_PUSH_PULL;

    /* configure GPIO with the given settings */
    ret = hal_gpio_init(&led);
    if (ret != 0) {
        printf("gpio init error !\n");
    }

    /* output high */
    hal_gpio_output_high(&led);

    /* output low */
    hal_gpio_output_low(&led);

    /* toggle output */
    hal_gpio_output_toggle(&led);
    /* sleep 1000ms */
    aos_msleep(1000);
```

```
    ret = hal_gpio_finalize(&led);
    if (ret != 0) {
        printf("hal_gpio_finalize error !\n");
    }
    printf("gpio output test ok\r\n");
}
```

2.2 GPIO 输入

配置相应 GPIO 方向寄存器(swporta_ddr)为输入，读取输入寄存器(swporta_ddr)，即可获取当前 GPIO 的电平状态。

比如将 P11 配置为输入，并读取当前 GPIO 电平状态。

```
void hal_gpio_read(void)
{
    int ret = -1;

    drv_pinmux_config(P11, PIN_FUNC_GPIO);

    /* input pin config */
    button1.port = P11;

    /* set as pull_up mode */
    button1.config = INPUT_PULL_UP;

    /* configure GPIO with the given settings */
    ret = hal_gpio_init(&button1);
    if (ret != 0) {
        printf("gpio init error !\n");
    }

    uint32_t value = 0;
    /* gpio read */
    ret = hal_gpio_input_get(&button1, &value);
    if (ret != 0) {
        printf("gpio init error !\n");
    }
    ret = hal_gpio_finalize(&button1);
    if (ret != 0) {
        printf("hal_gpio_finalize error !\n");
    }
    printf("gpio read p11 :%d\r\n",value);
}
```

2.3 GPIO retention

配置 GPIO 为输出，当系统休眠后，GPIO 输出信息将会丢失。如果想在系统休眠后，仍保持 GPIO 输出状态并保持输出的高低电平，需要使用 GPIO retention 功能。GPIO 初始化时默认已使能 retention 功能。

2.4 GPIO 上下拉电阻

每个 GPIO 支持四种上下拉配置：悬空、强上拉、上拉、下拉。
比如：将 P11 配置为输入，配置强上拉，并读取当前 GPIO 电平状态。

```
void hal_gpio_read(void)
{
	int ret = -1;

	drv_pinmux_config(P11, PIN_FUNC_GPIO);

	/* input pin config */
	button1.port = P11;

	/* set as input mode */
	button1.config = INPUT_PULL_UP;

	/* configure GPIO with the given settings */
	ret = hal_gpio_init(&button1);
	if (ret != 0) {
		printf("gpio init error !\n");
	}

	phy_gpio_pull_set(P11, STRONG_PULL_UP); //FlOATING – 浮空  PULL_DOWN – 下拉

	uint32_t value = 0;
	/* gpio read */
	ret = hal_gpio_input_get(&button1, &value);
	if (ret != 0) {
		printf("gpio init error !\n");
	}
	ret = hal_gpio_finalize(&button1);
	if (ret != 0) {
		printf("hal_gpio_finalize error !\n");
	}
	printf("gpio read p11 :%d\r\n",value);
}
```

2.5 中断和唤醒

使用 GPIO 中断时，需要配置 GPIO 中断产生条件。中断产生后，GPIO 驱动会响应中断并调用用户配置的回调函数。
使用 GPIO 唤醒时，当系统进入休眠前，根据当前 GPIO 的电平状态设置唤醒系统的条件，当该条件产生时，系统唤醒并会调用用户配置的回调函数。
比如：将 P11 配置为输入，支持中断和唤醒，支持上升沿下降沿触发。

```
void button1_handler(void *arg)
{
    button1_pressed = 0;
}

void hal_gpio_intr(void)
{
    int ret = -1;

    drv_pinmux_config(P11, PIN_FUNC_GPIO);

    /* input pin config */
    button1.port = P11;

    button1.config = INPUT_PULL_UP;

    /* configure GPIO with the given settings */
    ret = hal_gpio_init(&button1);
    if (ret != 0) {
        printf("gpio init error !\n");
    }

    /* gpio interrupt config */
    ret = hal_gpio_enable_irq(&button1, IRQ_TRIGGER_FALLING_EDGE,
                                        button1_handler, NULL);
    if (ret != 0) {
        printf("gpio irq enable error !\n");
    }

    /* if button is pressed, print "button 1 is pressed !" */
    printf("button wait for press!\n");
    while(button1_pressed) {
        aos_msleep(1000);
    };
    if (button1_pressed == 0) {
        button1_pressed = 1;
```

```
        printf("button    is pressed !\n");
    }
    ret = hal_gpio_finalize(&button1);
    if (ret != 0) {
        printf("hal_gpio_finalize error !\n");
    }
    printf("p11 intr test ok\r\n");
}
```

2.6 GPIO FULLMUX 模式

使用 GPIO FULLMUX 时，配置 FULLMUX 功能并打开 FULLMUX 使能。不使用时，一定要关闭其 FULLMUX 使能。

比如将 P9、P10 复用为 UART，其中 P9 做 TX、P10 做 RX，波特率为 115200，使用 UART0。

```
#define CONSOLE_UART_IDX 0
#define CONSOLE_TXD                    P9
#define CONSOLE_RXD                    P10
#define CONSOLE_TXD_FUNC               FMUX_UART0_TX
#define CONSOLE_RXD_FUNC               FMUX_UART0_RX
......
drv_pinmux_config(CONSOLE_TXD, CONSOLE_TXD_FUNC);
drv_pinmux_config(CONSOLE_RXD, CONSOLE_RXD_FUNC);
......
console_init(CONSOLE_UART_IDX, 115200, 0);
......
```

2.7 ANALOG 模式

详见《TG7120B_ADC_Application_Note》