ESP8266 GPIO 使用说明

Version 0.4

Espressif Systems IOT Team

免责申明和版权公告

本文中的信息，包括供参考的URL地址，如有变更，恕不另行通知。

文档"按现状"提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档不负任何责任，包括使用本文档内信息产生的侵犯任何专利权行为的责任。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。

Wi-Fi联盟成员标志归Wi-Fi联盟所有。

文中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。

Table of Contents

1. 概述

ESP8266共有16个通用IO，管脚的位置和管脚的分别为：

GPIO NO.	pin NO.	pin name
GPIO0	pin15	GPIO0_U
GPIO1	pin26	U0TXD_U
GPIO2	pin14	GPIO2_U
GPIO3	pin25	U0RXD_U
GPIO4	pin16	GPIO4_U
GPIO5	pin24	GPIO5_U
GPIO6	pin21	SD_CLK_U
GPIO7	pin22	SD_DATA0_U
GPIO8	pin23	SD_DATA1_U
GPIO9	pin18	SD_DATA2_U
GPIO10	pin19	SD_DATA3_U
GPIO11	pin20	SD_CMD_U
GPIO12	pin10	MTDI_U
GPIO13	pin12	MTCK_U
GPIO14	pin9	MTMS_U
GPIO15	pin13	MTDO_U

其中，在四线(QUAD)模式flash下，有六个io口用于flash通讯。

在两线(DUAL)模式flash下，有四个IO口用于与flash通讯。

本说明如果对照以下资料阅读，会更有助理解：

GPIO寄存器表：ESP8266_GPIO_Register.xlsx

与引脚功能复用表：ESP8266_Pin_List.xlsx

2. GPIO寄存器说明

2.1. GPIO功能选择寄存器

下面以8266的MTDI为例，说明GPIO功能的选择。

功能选择寄存器PERIPHS_IO_MUX_MTDI_U(不同的GPIO，该寄存器不同)

PIN_FUNC_SELECT(PERIPHS_IO_MUX_MTDI_U,FUNC_GPIO12);

此处的FUNC_GPIO12=3。

不同的PIN脚，配置不同。

配置的时候，请参考ESP8266_Pin_List_xxxxxx.xlsx表格，在该表格的Digital Die Pin List 页中可以查到通用的GPIO以及复用功能，在Reg页可以查阅到GPIO功能选择相关的寄存器。

Digital Die Pin List该页中的FUNCTION下拉选择项就是功能的配置选项。需要注意的是，如果需要配置为FUNCTION3，应该往寄存器对应的位中写2，如果需要配置为FUNCTION2，应该往寄存器对应的位中写1，以此类推。

2.2. GPIO输出寄存器

a) 输出使能寄存器GPIO_ENABLE_W1TS

bit[15:0]输出使能位(可读写):

若对应的位被置1，表示该IO的输出被使能。Bit[15:0]对应16个GPIO的输出使能位。

b) 输出禁用寄存器GPIO_ENABLE_W1TC

bit[15:0]输出禁用位(可读写):

若对应的位被置1，表示该IO的输出被禁用。Bit[15:0]对应16个GPIO的输出禁用位。

c) 输出使能状态寄存器GPIO_ENABLE

Bit[15:0]输出使能状态位(可读写):

该寄存器的bit[15:0]的值，反映的是对应的PIN脚输出使能状态。

GPIO_ENABLE的bit[15:0]通过给GPIO_ENABLE_W1TS的bit[15:0]和GPIO_ENABLE_W1TC的bit[15:0]写值来控制。例如GPIO_ENABLE_W1TS的bit[0]置1，则GPIO_ENABLE的bit[0]=1。

GPIO_ENABLE_W1TC的bit[1]置1，则GPIO_ENABLE的bit[1]=0。

d）输出低电平寄存器GPIO_OUT_W1TC

bit[15:0]输出低电平位(只写寄存器):

若对应的位被置1，表示该IO的输出为低电平（同时需要使能输出）。Bit[15:0]对应16个GPIO的输出状态。

备注：如果需要将该PIN配置为高电平，需要配置GPIO_OUT_W1TS寄存器。

e）输出高电平寄存器GPIO_OUT_W1TS

bit[15:0]输出高电平位(只写寄存器):

若对应的位被置1，表示该IO的输出为高电平（同时需要使能输出）。Bit[15:0]对应16个GPIO的输出状态。

备注：如果需要将该PIN配置为低电平，需要配置GPIO_OUT_W1TC寄存器。

f)输出状态寄存器GPIO_OUT

bit[15:0]输出状态位(读写寄存器):

该寄存器的[15:0]的值，反映的是对应PIN脚输出的状态。

GPIO_OUT的bit[15:0]是由GPIO_OUT_W1TS的bit[15:0]和GPIO_OUT_W1TC的bit[15:0]共同决定的。

例如：GPIO_OUT_W1TS的 Bit[1]=1,那么GPIO_OUT[1]=1。GPIO_OUT_W1TC的Bit[2]=1,那么GPIO_OUT[2]=0。

2.3. GPIO输入寄存器

输入状态寄存器 GPIO_IN

bit[15:0]输入状态位(可读写):

若对应的位为1，表示该IO的引脚状态为高电平，若对应的位为0表示该IO的引脚状态低电平 。Bit[15:0]对应16个GPIO的输入状态位。

备注：GPIO的输入检测功能，是缺省设置,无需使能。

2.4. GPIO中断寄存器

a) 中断类型寄存器 GPIO_PIN12(不同的GPIO该寄存器不同)

bit[9:7](可读写):

0:该GPIO的中断禁用

1:上升沿触发中断

2:下降沿触发中断

3:双沿触发中断

4:低电平

5:高电平

b) 中断状态寄存器GPIO_STATUS

Bit[15:0](可读写):

若对应的位被置1，表示该IO中断发生。Bit[15:0]对应16个GPIO。

c)清中断寄存器 GPIO_STATUS_W1TC

Bit[15:0](可读写):

向对应的位写1，对应的GPIO的中断状态就会被清除。

2.5. GPIO16对应接口

与其他IO口不同，GPIO16(XPD_DCDC)不属于通用GPIO模块，它属于RTC模块，可以用来在深度睡眠时候唤醒整个芯片，可以配置为输入或者输出模式，但无法触发IO中断。使用接口如下:

a). gpio16_output_conf(void)

将GPIO16配置为输出模式。

b). gpio16_output_set(uint8 value)

从GPIO16输出高电平/低电平，需要先配置为输出模式。

c). gpio16_input_conf(void)

将GPIO16配置为输入模式。

d). gpio16_input_get(void)

读取GPIO16的输入电平状态，需要先配置为输入模式。

3. 参数配置

在参数配置过程中，给出3个应用场景。用户可以此应用场景为例，配置其他的GPIO。

应用场景:

1) 配置MTDI输出高电平，并使能其上拉。

2) 配置MTDI为输入模式，并获取其电平状态。

3) 配置MTDI为下降沿触发中断。

3.1. 应用场景1参数配置

Step 1) 配置MTDI为GPIO模式

PIN_FUNC_SELECT(PERIPHS_IO_MUX_MTDI_U,FUNC_GPIO12);

该语句的作用是向PERIPHS_IO_MUX_MTDI_U寄存器的第4位、第5位写1。PERIPHS_IO_MUX_MTDI_U寄存器的第4位和第5位置1表示将MTDI配置为GPIO功能。PERIPHS_IO_MUX_MTDI_U寄存器，请参阅GPIO寄存器章节。

Step 2) 配置MTDI输出高电平

GPIO_OUTPUT_SET(GPIO_ID_PIN(12), 1);

该语句有两个作用:

- 向GPIO_ENABLE_W1TS的寄存器第12位写1，该位置1表示使能MTDI的输出功能。

- 向GPIO_OUT_W1TS的寄存器第12位写1，该位置1表示将MTDI输出为高电平。

备注：需要MTDI配置输出低电平，将该函数的第二个参数设置为0即可。

GPIO_OUTPUT_SET(GPIO_ID_PIN(12), 0);

该语句有两个作用：

- 向GPIO_ENABLE_W1TS的寄存器第12位写1，该位置1表示使能MTDI的输出功能。
- 向GPIO_OUT_W1TC的寄存器第12位写1，该位置1表示将MTDI输出为低电平。

Step 3）使能MTDI上拉

PIN_PULLUP_EN(PERIPHS_IO_MUX_MTDI_U);

该语句作用是向PERIPHS_IO_MUX_MTDI_U的第7位写1。该位置1表示使能MTDI的上拉功能。

备注：如果需要关闭MTDI的上拉功能，请使用如下语句

PIN_PULLUP_DIS(PERIPHS_IO_MUX_MTDI_U);

3.2. 应用场景2参数配置

Step 1）配置MTDI为GPIO模式

PIN_FUNC_SELECT(PERIPHS_IO_MUX_MTDI_U,FUNC_GPIO12);

该语句的作用是向PERIPHS_IO_MUX_MTDI_U寄存器的第4位、第5位写1。PERIPHS_IO_MUX_MTDI_U寄存器的第4位和第5位置1表示将MTDI配置为GPIO功能。

Step 2） 配置MTDI为输入模式

GPIO_DIS_OUTPUT(GPIO_ID_PIN(12));

Step 3）获取MTDI管脚的电平状态

Uint8 level=0;

level=GPIO_INPUT_GET(GPIO_ID_PIN(12));

GPIO_INPUT_GET(GPIO_ID_PIN(12))语句实际是获取GPIO_IN寄存器第12位的状态，该寄存器的值反映的是对应的管脚的输入电平（必须使能对应的管脚的输入功能，该寄存器的状态才有效）。

备注：

如果MTDI的电平为高电平，那么GPIO_INPUT_GET的返回值为1，level=1;

如果MTDI的电平为低电平，那么GPIO_INPUT_GET的返回值为0，level=0;

3.3. 应用场景3参数配置

```
typedef enum {
GPIO_PIN_INTR_DISABLE = 0,
GPIO_PIN_INTR_POSEDGE = 1,
GPIO_PIN_INTR_NEGEDGE = 2,
GPIO_PIN_INTR_ANYEGDE = 3,
GPIO_PIN_INTR_LOLEVEL = 4,
GPIO_PIN_INTR_HILEVEL = 5
} GPIO_INT_TYPE;
```

该结构体用来配置gpio的中断触发方式，该结构体在gpio.h中声明。

Step 1）配置MTDI为GPIO模式

```
PIN_FUNC_SELECT(PERIPHS_IO_MUX_MTDI_U,FUNC_GPIO12);
```

该语句的作用是向PERIPHS_IO_MUX_MTDI_U寄存器的第4位、第5位写1。PERIPHS_IO_MUX_MTDI_U寄存器的第4位和第5位置1表示将MTDI配置为GPIO功能。

Step 2）配置MTDI为输入模式

```
GPIO_DIS_OUTPUT(GPIO_ID_PIN(12));
```

Step 3）禁止所有的IO中断

```
ETS_GPIO_INTR_DISABLE();
```

Step 4）设置中断处理函数

```
ETS_GPIO_INTR_ATTACH(GPIO_INTERRUPT,NULL);
```

Step 5）配置MTDI为下降沿中断触发的方式

```
gpio_pin_intr_state_set(GPIO_ID_PIN(12),GPIO_PIN_INTR_NEGEDGE);
```

该语句的作用是向GPIO_PIN12的寄存器的[9:7]位写入0x02，向该位域内写入0x02,表示配置为下降沿中断。

备注：若需要禁用MTDI的中断功能，只需要将向GPIO_PIN12的寄存器的[9:7]位写入0x00即可。如需配置为其他的中断触发模式，请参阅GPIO寄存器章节。

Step6） 使能gpio中断

```
ETS_GPIO_INTR_ENABLE();
```

3.4. 中断函数处理流程说明

Step 1）清除该中断

Uint16 gpio_status=0;

gpio_status = GPIO_REG_READ(GPIO_STATUS);

GPIO_REG_WRITE(GPIO_STATUS_W1TC, gpio_status);

GPIO_STATUS和GPIO_STATUS_W1TC说明请参阅GPIO寄存器章节。

Step 2）判断是哪个IO触发的中断(当有多个IO都配置为中断方式时)

If(gpio_status==GPIO_Pin_12)

Step 3）如果是双沿中断，应该判断此次中断为上升沿中断还是下降沿中断。

if(!GPIO_INPUT_GET(GPIO_ID_PIN(12))) //如果MTDI此次触发方式为下降沿

3.5. 中断函数处理流程示例

```
void gpio_intr_handler()
{

    uint32 gpio_status = GPIO_REG_READ(GPIO_STATUS_ADDRESS);//read interrupt status
    uint8  level=0;
     GPIO_REG_WRITE(GPIO_STATUS_W1TC_ADDRESS, gpio_status);//clear interrupt mask
     if(gpio_status & (BIT(12))){  //judge whether interrupt source is gpio12
       if(GPIO_INPUT_GET(12)){   // if gpio 12 is high level

         }else{ // if gpio 12 is low level

      }
    }
     else{
     }

}
```