

文档版本	V1.0
发布日期	20210524

APT32S003 GPIO 应用指南

APTCHIP



目录

1 概述	1
2. 适用的硬件.....	1
3. 应用方案代码说明	1
3.1 GPIO 配置	1
3.2 GPIO 输出	4
3.3 GPIO 管脚状态	4
3.4 GPIO 上下拉	5
3.5 GPIO 开漏	6
3.6 GPIO 强驱动	7
3.7 GPIO AF 复用功能	7
3.8 GPIO 外部中断	8
4. 程序下载和运行	10

1 概述

本文介绍了在APT32S003中使用GPIO的应用范例。

2. 适用的硬件

该例程使用于 APT32S003 系列学习板

3. 应用方案代码说明

3.1 GPIO 配置

基于 APT32S003 完整的库文件系统，可以对 GPIO 进行配置。

- 硬件配置：

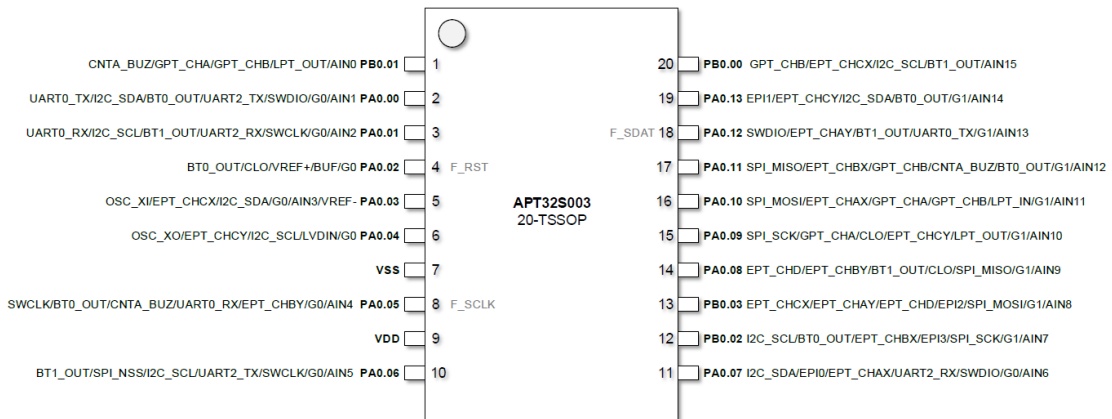


图 3.3.1 管脚定义图

SWD 管脚默认接口为 PA0.05 和 PA0.12，修改成其它 AF 功能，将丢失调试连接并无法再连接调试器

● GPIO 原理图:

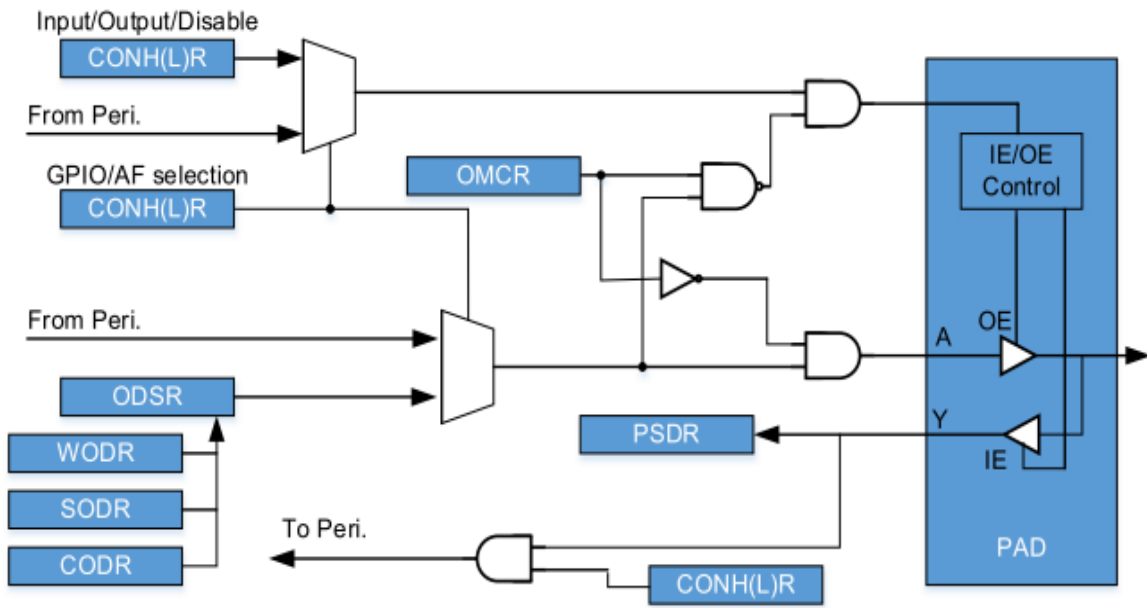


图 3.3.2 GPIO 框图

● 软件配置:

使用库函数，在 apt32f101_initialr.c 文件中进行初始化的配置

● 编程要点:

1. SYSCON_CONFIG();配置时钟函数
- 2.GPIO_Init(); 配置 GPIO 状态

```

/*****
//IO OUTPUT INPUT SET 1
//EntryParameter:GPIOx,GPIO_Pin(0~15),byte,Dir
//GPIOx:GPIOA0,GPIOB0
//GPIO_Pin:PIN_0~15
//byte:Lowbyte(PIN_0~7),Highbyte(PIN_8~15)
//Dir:0:output 1:input
//ReturnValue:NONE
*****/
void GPIO_Init(CSP_GPIO_T *GPIOx,uint8_t PinNum,GPIO_Dir_TypeDef Dir)
{
    uint32_t data_temp;
    uint8_t GPIO_Pin;
    if(PinNum<8)
    {
        switch (PinNum)
        {

```

```

case 0:data_temp=0xfffff0;GPIO_Pin=0;break;
case 1:data_temp=0xfffff0;GPIO_Pin=4;break;
case 2:data_temp=0xffff0ff;GPIO_Pin=8;break;
case 3:data_temp=0xffff0ff;GPIO_Pin=12;break;
case 4:data_temp=0xff0ffff;GPIO_Pin=16;break;
case 5:data_temp=0xf0ffff;GPIO_Pin=20;break;
case 6:data_temp=0xf0ffff;GPIO_Pin=24;break;
case 7:data_temp=0x0fffff;GPIO_Pin=28;break;
}
if (Dir)
{
    (GPIOx)->CONLR = ((GPIOx)->CONLR & data_temp) | 1<<GPIO_Pin;
}
else
{
    (GPIOx)->CONLR = ((GPIOx)->CONLR & data_temp) | 2<<GPIO_Pin;
}
}
else if (PinNum<16)
{
    switch (PinNum)
    {
        case 8:data_temp=0xfffff0;GPIO_Pin=0;break;
        case 9:data_temp=0xfffff0;GPIO_Pin=4;break;
        case 10:data_temp=0xffff0ff;GPIO_Pin=8;break;
        case 11:data_temp=0xffff0ff;GPIO_Pin=12;break;
        case 12:data_temp=0xff0ffff;GPIO_Pin=16;break;
        case 13:data_temp=0xf0ffff;GPIO_Pin=20;break;
        case 14:data_temp=0xf0ffff;GPIO_Pin=24;break;
        case 15:data_temp=0x0fffff;GPIO_Pin=28;break;
    }
}
if (Dir)
{
    (GPIOx)->CONHR = ((GPIOx)->CONHR & data_temp) | 1<<GPIO_Pin;
}
else
{
    (GPIOx)->CONHR = ((GPIOx)->CONHR & data_temp) | 2<<GPIO_Pin;
}
}
}
}

```

● 函数说明:



`void GPIO_Init(CSP_GPIO_T *GPIOx,uint8_t PinNum,GPIO_Dir_TypeDef Dir)`

3.2 GPIO 输出

```

/*****/
//Write GPIO
//EntryParameter:GPIOx,uint8_t bit
//GPIOx:GPIOA0,GPIOB0
//bit:0~15
//ReturnValue:VALUE
/*****/
void GPIO_Write_High(CSP_GPIO_T *GPIOx,uint8_t bit)
{
    (GPIOx)->SODR = (1ul<<bit);
}
void GPIO_Write_Low(CSP_GPIO_T *GPIOx,uint8_t bit)
{
    (GPIOx)->CODR = (1ul<<bit);
}

```

- 代码说明:

GPIO_Write_High(): ---- 用于 GPIO 口输出高

GPIO_Write_Low(): ---- 用于 GPIO 口输出低

- 函数参数:



`GPIO_Write_Low(CSP_GPIO_T *GPIOx,uint8_t bit)`

3.3 GPIO 管脚状态

```

/*****/
//READ PA IO STATUS
//EntryParameter:GPIOx,uint8_t bit
//GPIOx:GPIOA0,GPIOB0
//bit:0~15
//ReturnValue:VALUE

```

```

/*****
uint8_t GPIO_Read_Status(CSP_GPIO_T *GPIOx,uint8_t bit)
{
    uint8_t value = 0;
    uint32_t dat = 0;
    dat=((GPIOx)->PSDR)&(1<<bit);
    if (dat == (1<<bit))
    {
        value = 1;
    }
    return value;
}

```

● 代码说明:

GPIO_Read_Status(); ----用于读取 GPIO 的管脚状态

● 函数参数说明:



uint8_t GPIO_Read_Status(GPIOA0,0)

3.4 GPIO 上下拉

```

/*****
//Write GPIO pull high/low
//EntryParameter:GPIOx,uint8_t bit
//GPIOx:GPIOA0,GPIOB0
//bit:0~15
//Return Value:VALUE
/*****
void GPIO_PullHigh_Init(CSP_GPIO_T *GPIOx,uint8_t bit)
{
    (GPIOx)->PUDR = (((GPIOx)->PUDR) & ~(0x03<<(bit*2))) | (0x01<<(bit*2));
}
void GPIO_PullLow_Init(CSP_GPIO_T *GPIOx,uint8_t bit)
{
    (GPIOx)->PUDR = (((GPIOx)->PUDR) & ~(0x03<<(bit*2))) | (0x02<<(bit*2));
}
void GPIO_PullHighLow_DIS(CSP_GPIO_T *GPIOx,uint8_t bit)
{

```

```
(GPIOx)->PUDR = ((GPIOx)->PUDR) & ~(0x03<<(bit*2));
}
```

● 代码说明:

GPIO_PullHigh_Init(); ----用于 GPIO 上拉配置

GPIO_PullLow_Init(); ----用于 GPIO 下拉配置

GPIO_PullHighLow_DIS(); ----用于关闭 GPIO 上/下拉

● 函数参数说明:



GPIO_PullHigh_Init(GPIOA0,4);

3.5 GPIO 开漏

```
/*
 * *****
 * //Write GPIO open drain init
 * //EntryParameter:GPIOx,uint8_t bit
 * //GPIOx:GPIOA0,GPIOB0
 * //bit:0-15
 * //ReturnValue:VALUE
 * *****
 */
void GPIO_OpenDrain_EN(CSP_GPIO_T *GPIOx,uint8_t bit)
{
    (GPIOx)->OMCR = ((GPIOx)->OMCR) | (0x01<<bit);
}

void GPIO_OpenDrain_DIS(CSP_GPIO_T *GPIOx,uint8_t bit)
{
    (GPIOx)->OMCR = ((GPIOx)->OMCR) & ~(0x01<<bit);
}
}
```

● 代码说明:

GPIO_OpenDrain_EN();----用于设置 GPIO 开漏

GPIO_OpenDrain_DIS();----用于关闭 GPIO 开漏

3.6 GPIO 强驱动

标有(HS)符号的 IO 为大电流驱动口 (High Sink Current IO)，支持 120mA 的灌电流

S003 系列的大电流驱动口： PB0.2(HS) PB0.3(HS) PA0.8(HS) PA0.9(HS)

```

/*****/
//Write GPIO Drive Strength init
//EntryParameter:GPIOx,uint8_t bit
//GPIOx:GPIOA0,GPIOB0
//bit:0~15
//ReturnValue:VALUE
/*****/

void GPIO_DriveStrength_EN(CSP_GPIO_T *GPIOx,uint8_t bit)
{
    (GPIOx)->DSCR = ((GPIOx)->DSCR) | (0x01<<(bit*2));
}

void GPIO_DriveStrength_DIS(CSP_GPIO_T *GPIOx,uint8_t bit)
{
    (GPIOx)->DSCR = ((GPIOx)->DSCR) & ~(0x01<<(bit*2));
}
    
```

● 代码说明：

GPIO_DriveStrength_EN(); ---- 用于配置驱动强度

GPIO_DriveStrength_DIS();---- 用于关闭驱动强度

3.7 GPIO AF 复用功能

举例：把 PA0.0 设置为 UART_TX 功能,通过 APT32S003 系列数据手册，得知 PA0.0 管脚功能分配复用 AF1 是 UART0_TX。

Package	Pin Name										Default	TTL	Mode	Reset	Status
	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8	EXI					
20TSSOP															

图 3.7.1 管脚功能定义

● GPIO 模式：

0h: GPD (GPIO Disabled)，当前 GPIO 输入输出禁止模式，即高阻态（默认模式）。

1h: GPI (GPIO Input)，当前 GPIO 设置为输入模式。

2h: GPO (GPIO Output), 当前 GPIO 设置为输出模式, 输入禁止。

3h: GPO (GPIO Output), 当前 GPIO 设置为输出模式, 输出监测使能 (输入 Buffer 使能)。

4h ~15h: AFx (x 从 ‘1’ 开始), 功能复用模式 (参见管脚配置)。

● **代码示例:**

GPIOA0->CONLR=(GPIOA0->CONLR&0X0FFFFFFF) | 0x00000004; //PA0.0->UART0_TX

3.8 GPIO 外部中断

S003 系列任何一个 GPIO 管脚都可以设置成外部中断源, 中断触发方式可由与 SYSCON EXI 相关的控制寄存器来进行设置。

```

void GPIO_CONFIG(void)
{
    GPIO_PullHigh_Init(GPIOB0,2);
    //----- EXI FUNTION -----/
    //EXI0_INT= EXI0/EXI16,EXI1_INT= EXI1/EXI17, EXI2_INT=EXI2-EXI3/EXI18/EXI19, EXI3_INT=EXI4-EXI9, EXI4_INT=EXI10-EXI15

    GPIO_IntGroup_Set(PB0,2,Selete_EXI_PIN2);           //EXI0 set PB0.2
    GPIOB0_EXI_Init(EXI2);                             //PB0.2 as input
    EXTI_trigger_CMD(ENABLE,EXI_PIN2,_EXIFT);         //ENABLE falling edge
    EXTI_interrupt_CMD(ENABLE,EXI_PIN2);              //enable EXI
    GPIO_EXTI_interrupt(GPIOB0,0b00000000000100);     //enable GPIOB02 as EXI
    EXI2_Int_Enable();                                 //EXI2-EXI3 INT Vector
    //EXI2_WakeUp_Enable(); }

    /*****
    //EXI2 Interrupt
    //EntryParameter:NONE
    //ReturnValue:NONE
    *****/

    void EXI2to3IntHandler(void)
    {
        // ISR content ...
        if ((SYSCON->EXIRS&EXI_PIN2)==EXI_PIN2)
        {
            SYSCON->EXICR = EXI_PIN2;
        }
        else if ((SYSCON->EXIRS&EXI_PIN3)==EXI_PIN3)
        {
            SYSCON->EXICR = EXI_PIN3;
        }
    }
}
    
```

```

}
}

```

● 代码说明:

GPIO_PullHigh_Init(); ----用于配置 GPIO 内部上拉

GPIO_IntGroup_Set(); ----用于配置外部中断组

GPIOB0_EXI_Init(); ----用于设置 GPIO 为输入

EXTI_trigger_CMD(); ----用于使能外部中断的触发方式

EXTI_interrupt_CMD(); ----用于使能 syscon 外部中断

EXI2_Int_Enable(); ----用于使能 GPIO 口外部中断

EXI2_WakeUp_Enable(); ----用于配置外部中断唤醒低功耗下

● 函数参数说明:



GPIO_IntGroup_Set(PB0,2,Selete_EXI_PIN2);



EXTI_trigger_CMD(ENABLE,EXI_PIN2, EXIFT);



EXTI_interrupt_CMD(ENABLE,EXI_PIN2);



```
GPIO_EXTI_interrupt(GPIOB0,0b00000000000100);
```

- 中断服务对应的外部中断组

EXI0 ----External interrupt GROUP0, GROUP16

EXI1 ----External interrupt GROUP1, GROUP17

EXI2 ---- External Interrupt GROUP2 ~ 3, GROUP18~19

EXI3 ---- External Interrupt GROUP4 ~ 9

EXI4 ---- External Interrupt GROUP10 ~ 15

4. 程序下载和运行

1. 将目标板与仿真器连接，分别为 VDD SCLK SWIO GND
2. 程序编译后仿真运行
3. 查看设置 GPIO 输出的高低电平