阅读排行榜

推荐排行榜

最新评论

1. Re:ESA2GJK1DH1K基础篇: 阿里云物联网平台: 使用阿里云物联网平台提供的自定义Topic通信控制(ESP8266,TCP透传指令)
代码在哪里下载？

--题哦咯

在GPIO设置为输出的状态下读取GPIO电平

1,控制PD3 输出高低电平,并打印其引脚状态(把以下程序直接拷贝到自己工程运行)

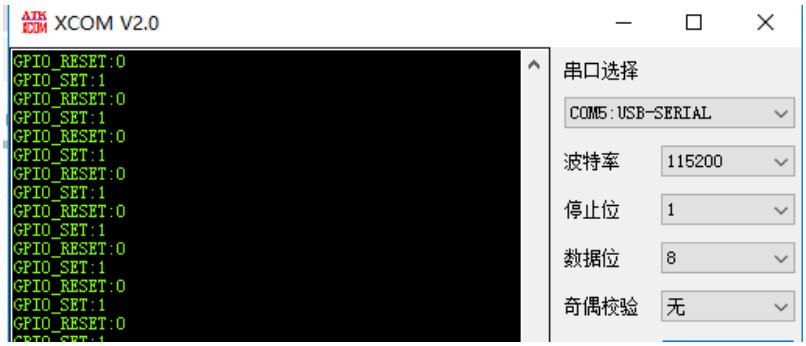
```
#include "debug.h"
#include "ch32v30x.h"



#define  GPIO_PORT         (GPIOD)
#define  GPIO_PIN          (GPIO_Pin_3)

#define  GPIO_SET    (GPIO_SetBits(GPIO_PORT, GPIO_PIN))    //输出高电平
#define  GPIO_RESET  (GPIO_ResetBits(GPIO_PORT, GPIO_PIN)) //输出低电平
#define  GPIO_INPUT  (GPIO_ReadOutputDataBit(GPIO_PORT, GPIO_PIN)) //
#define  GPIO_TOGGLE (GPIO_WriteBit(GPIO_PORT, GPIO_PIN, 1-GPIO_INPUT

#define  GPIO_RCC_ENADLE (RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD
/**
* @brief  init
* @param
* @param  None
* @param  None
* @retval None
* @example
**/
void gpio_init(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;
    GPIO_RCC_ENADLE;   //启动GPIO的时钟线,让时钟进去以驱动其GPIO
```

2. Re:ESP8266 SDK开发: 综合篇-C#上位机串口通信控制ESP8266
认真拜阅读您的程序 收获很大 ;但我发现一处问题 就是您在串口接收函数中 没有将空闲标志清零 导致程序最终并没有通过空闲中断来处理 而是每隔10ms处理一次

--觉代疯骚

```
    GPIO_InitStructure.GPIO_Pin = GPIO_PIN;//pin
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; //推挽输出(最大驱动
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;//频率越高,切换GPI
    GPIO_Init(GPIO_PORT, &GPIO_InitStructure);
}
int main(void)
{
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
    USART_Printf_Init(115200);
    Delay_Init();
    gpio_init();
    while(1)
    {
        GPIO_SET;//设置GPIO输出高电平
        printf("GPIO_SET:%d\r\n", GPIO_INPUT);//打印GPIO电平状态
        Delay_Ms(500);
        GPIO_RESET;//设置GPIO输出低电平
        printf("GPIO_RESET:%d\r\n", GPIO_INPUT);//打印GPIO电平状态
        Delay_Ms(500);
    }
}
```

2.使用数据线连接开发板



设置PA0为输入上拉状态, 读取PA0状态

```
#include "debug.h"
#include "ch32v30x.h"


#define  GPIO_PORT         (GPIOA)
#define  GPIO_PIN          (GPIO_Pin_0)

#define  GPIO_INPUT   (GPIO_ReadInputDataBit(GPIO_PORT, GPIO_PIN)) //读

#define  GPIO_RCC_ENADLE  (RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA
/**
* @brief  init
* @param
* @param  None
* @param  None
* @retval None
* @example
**/
void gpio_init(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;
    GPIO_RCC_ENADLE;    //启动GPIO的时钟线,让时钟进去以驱动其GPIO
    GPIO_InitStructure.GPIO_Pin = GPIO_PIN;//pin
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; //上拉输入
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;//设不设置都可以
    GPIO_Init(GPIO_PORT, &GPIO_InitStructure);
}
int main(void)
{
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
    USART_Printf_Init(115200);
    Delay_Init();
    gpio_init();
    while(1)
    {
        printf("GPIO_INPUT:%d\r\n", GPIO_INPUT);//打印GPIO电平状态
        Delay_Ms(500);
    }
}
```

注意:设置为输入状态时, 需要使用 GPIO_ReadInputDataBit 函数获取

main.c | ch32v30x_gpio.c | ch32v30x_gpio.h

```c

#include "debug.h"
#include "ch32v30x.h"


#define  GPIO_PORT         (GPIOA)
#define  GPIO_PIN          (GPIO_Pin_0)

#define  GPIO_INPUT  (GPIO_ReadInputDataBit(GPIO_PORT, GPIO_PIN)) //获取输入电平

#define  GPIO_RCC_ENADLE (RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE)) //打开时钟线
/**
* @brief  init
* @param
* @param  None
* @param  None
* @retval None
* @example
**/
void gpio_init(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;
    GPIO_RCC_ENADLE;   //启动GPIO的时钟线,让时钟进去以驱动其GPIO
    GPIO_InitStructure.GPIO_Pin = GPIO_PIN;//pin
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; //上拉输入
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;//设不设置都可以
    GPIO_Init(GPIO_PORT, &GPIO_InitStructure);
}
int main(void)
{
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
    USART_Printf_Init(115200);
    Delay_Init();
    gpio_init();
    while(1)
    {
        printf("GPIO_INPUT:%d\r\n", GPIO_INPUT);//打印GPIO电平状态
        Delay_Ms(500);
    }
}
```

把PA0口接低电平可以看到打印

```
GPIO_INPUT:1
GPIO_INPUT:1
GPIO_INPUT:1
GPIO_INPUT:1
GPIO_INPUT:1
GPIO_INPUT:1
GPIO_INPUT:1
GPIO_INPUT:1
GPIO_INPUT:1
GPIO_INPUT:0
GPIO_INPUT:0
GPIO_INPUT:0
GPIO_INPUT:0
GPIO_INPUT:0
GPIO_INPUT:0
GPIO_INPUT:0
```

关于中断优先级

在51单片机中只有一种优先等级,默认是(外部中断0, 定时器0, 外部中断1, 定时器1, 串口中断), 优先等级左面最高

当来了外部中断和定时器0中断的时候,优先处理外部中断0;

当定时器0执行的时候来了外部中断0,那么回去执行外部中断0, 然后再接着回来执行定时器0

对于CH32V307单片机, 中断有抢占式和响应式两种优先级(数越小越优先)

抢占式是指中断嵌套; 响应式是指中断同时来先执行谁;

抢占式等级相同,谁的响应式高先执行谁; 抢占式等级不同,谁的抢占式等级高先执行谁

假设有两个GPIO中断 PA0 和 PA1

PA0 的抢占式优先等级设置为 0; 响应式优先等级设置为 2;

PA1 的抢占式优先等级设置为 0; 响应式优先等级设置为 1;

假设正在执行的PA0中断, 现在来了PA1中断, 因为二者抢占式等级一样,所以等执行完了PA0 再执行PA1

假设PA0和PA1同时来了中断, 因为PA1的响应式比PA0高,所以先PA1 再执行PA0

PA0 的抢占式优先等级设置为 1; 响应式优先等级设置为 1;

PA1 的抢占式优先等级设置为 0; 响应式优先等级设置为 2;

假设正在执行的PA0中断, 现在来了PA1中断, 因为PA1抢占式比PA0高,所以会去执行PA1,然后再回来执行PA0

假设PA0和PA1同时来了中断, 因为PA1抢占式比PA0高,所以会去执行PA1,然后再回来执行PA0

所以记住上面的一句话就可以:

抢占式等级相同,谁的响应式高先执行谁; 抢占式等级不同,谁的抢占式等级高先执行谁

设置PA0为下降沿中断

1,把以下程序拷贝到工程,并下载到开发板

```
#include "debug.h"
#include "ch32v30x.h"

/*在中断里面调用的变量需要使用 volatile 修饰*/
volatile uint8_t gpio_interrupt_flag=0;

/**
* @brief  init
* @param  None
* @retval None
```

```
* @example
**/
void gpio_init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0};
    EXTI_InitTypeDef EXTI_InitStructure = {0};
    NVIC_InitTypeDef NVIC_InitStructure = {0};
    /*打开挂载GPIO总线APB2的复用时钟(GPIO除了输入输出的其它功能都叫做复用功能,
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO | RCC_APB2Periph_GPIOA
    /*设置GPIO*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;//pin
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; //上拉输入
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    /*设置PA0作为中断线的GPIO引脚*/
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource0);
    /*设置GPIO中断*/
    EXTI_InitStructure.EXTI_Line = EXTI_Line0;//中断线0
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;//中断模式
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;//下降沿触发
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;//使能
    EXTI_Init(&EXTI_InitStructure);
    /*配置中断优先等级*/
    NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;//外部中断0
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;//抢占式优
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 2;//响应式优先级
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;//使能
    NVIC_Init(&NVIC_InitStructure);
}

/**
* @brief  中断函数
**/
__attribute__((interrupt("WCH-Interrupt-fast"))) //中断函数前加这上这句,
void EXTI0_IRQHandler(void)
{
  if(EXTI_GetITStatus(EXTI_Line0)!=RESET)//产生中断
  {
      gpio_interrupt_flag=1;
      EXTI_ClearITPendingBit(EXTI_Line0);//清除中断标志
  }
}

int main(void)
{
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//设置优先级分组为2
    USART_Printf_Init(115200);
    Delay_Init();
    gpio_init();
    while(1)
    {
        if (gpio_interrupt_flag==1) {//有中断产生
            gpio_interrupt_flag=0;
            printf("gpio_interrupt_flag\r\n");
        }
    }
}
```

每次把PA0接到低电平串口就会打印

2,程序说明

1,关于优先级分组

前面有提到单片机有抢占式和响应式优先级，优先级是由一个四位的二进制数保存的,

四位的二进制数共有 2^4 = 16种, 也就是 0000 - 1111 , 就是有 0 - 15 等级

但是呢这是只存在一种优先等级的情况下, 可以有0 - 15 等级

上面的四位共同代表了抢占式和响应式优先级; 具体怎么分抢占式和响应式各有多少个;

就是下面的优先级分组来设置

假设设置优先级分组为 0 (NVIC_PriorityGroup_0) 那么就是没有抢占式,上面的四位全部作为响应式

那么咱在设置中断的时候,抢占式就不用设置了 , 响应式就是有(0-15) 选择

假设设置优先级分组为 1 (NVIC_PriorityGroup_1) 那么就是其中一位给抢占式,剩余3位作为响应式

那么咱在设置中断的时候,抢占式就是0 - 1 选择 , 响应式就是有(0-8) 选择

假设设置优先级分组为 2 (NVIC_PriorityGroup_2) 那么就是其中两位给抢占式, 其中两位作为响应式

那么咱在设置中断的时候,抢占式就是0 - 3 选择 , 响应式就是有(0-3) 选择

假设设置优先级分组为 3 (NVIC_PriorityGroup_3) 那么就是其中三位给抢占式, 其中一位作为响应式

那么咱在设置中断的时候,抢占式就是0 - 8 选择 , 响应式就是有(0-1) 选择

假设设置优先级分组为 4 (NVIC_PriorityGroup_4) 那么就是其中四位给抢占式, 没有响应式

那么咱在设置中断的时候,抢占式就是0 - 15 选择 , 响应式就不用设置了

main.c

```
55 {
56   if(EXTI_GetITStatus(EXTI_Line0)!=RESET)//产生中断
57   {
58       gpio_interrupt_flag=1;
59       EXTI_ClearITPendingBit(EXTI_Line0);//清除中断标志
60   }
61 }
62
63 int main(void)
64 {
65     NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//设置优先级分组为2
66     USART_Printf_Init(115200);
67     Delay_Init();
68     gpio_init();
69     while(1)
70     {
71         if (gpio_interrupt_flag==1) {//有中断产生
72             gpio_interrupt_flag=0;
73             printf("gpio_interrupt_flag\r\n");
74         }
75     }
76 }
```

```
/* Preemption_Priority_Group */
#define NVIC_PriorityGroup_0            ((uint32_t)0x00)
#define NVIC_PriorityGroup_1            ((uint32_t)0x01)
#define NVIC_PriorityGroup_2            ((uint32_t)0x02)
#define NVIC_PriorityGroup_3            ((uint32_t)0x03)
#define NVIC_PriorityGroup_4            ((uint32_t)0x04)
```

2,在中断里面赋值,在主轮训判断使用的变量需要使用 volatile 修饰

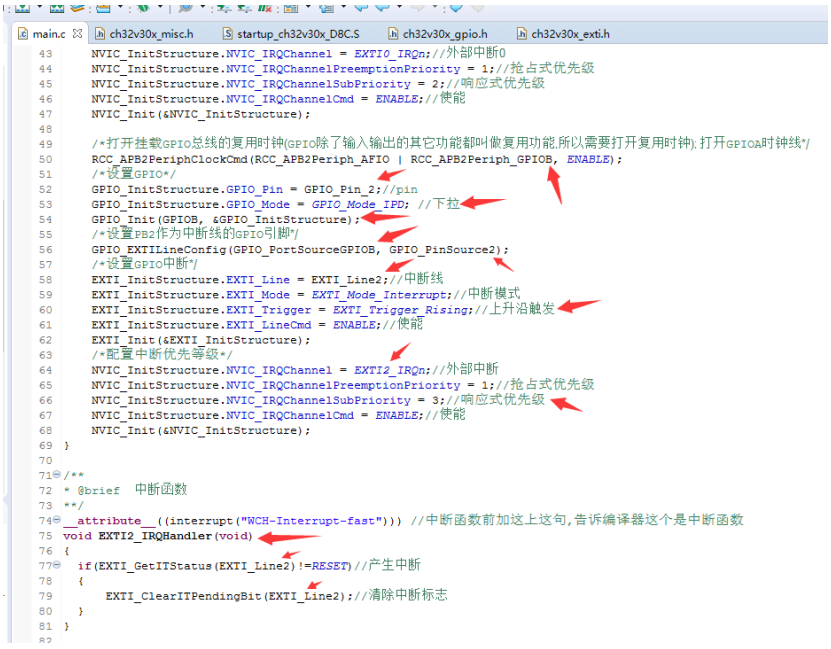
main.c ch32v30x_misc.h

```
7  * Copyright (c) 2021 Nanjing Qinheng Microelectronics Co.,
8  * SPDX-License-Identifier: Apache-2.0
9  ************************************************************
10
11 #include "debug.h"
12 #include "ch32v30x.h"
13
14 /*在中断里面调用的变量需要使用 volatile 修饰*/
15 volatile uint8_t gpio_interrupt_flag=0;
16
17 /**
```

假设设置PB2为上升沿中断

```c
#include "debug.h"
#include "ch32v30x.h"

/*在中断里面调用的变量需要使用 volatile 修饰*/
volatile uint8_t gpio_interrupt_flag=0;

/**
* @brief  init
* @param  None
* @retval None
* @example
**/
void gpio_init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0};
    EXTI_InitTypeDef EXTI_InitStructure = {0};
    NVIC_InitTypeDef NVIC_InitStructure = {0};
    /*打开挂载GPIO总线的复用时钟(GPIO除了输入输出的其它功能都叫做复用功能,所以需
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO | RCC_APB2Periph_GPIOA
    /*设置GPIO*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;//pin
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; //上拉输入
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    /*设置PA0作为中断线的GPIO引脚*/
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource0);
    /*设置GPIO中断*/
    EXTI_InitStructure.EXTI_Line = EXTI_Line0;//中断线0
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;//中断模式
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;//下降沿触发
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;//使能
    EXTI_Init(&EXTI_InitStructure);
    /*配置中断优先等级*/
    NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;//外部中断0
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;//抢占式优
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 2;//响应式优先级
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;//使能
    NVIC_Init(&NVIC_InitStructure);

    /*打开挂载GPIO总线的复用时钟(GPIO除了输入输出的其它功能都叫做复用功能,所以需
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO | RCC_APB2Periph_GPIOB
    /*设置GPIO*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;//pin
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD; //下拉
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    /*设置PB2作为中断线的GPIO引脚*/
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOB, GPIO_PinSource2);
    /*设置GPIO中断*/
    EXTI_InitStructure.EXTI_Line = EXTI_Line2;//中断线
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;//中断模式
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;//上升沿触发
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;//使能
    EXTI_Init(&EXTI_InitStructure);
    /*配置中断优先等级*/
    NVIC_InitStructure.NVIC_IRQChannel = EXTI2_IRQn;//外部中断
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;//抢占式优
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;//响应式优先级
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;//使能
    NVIC_Init(&NVIC_InitStructure);
}

/**
* @brief  中断函数
```

```
**/
__attribute__((interrupt("WCH-Interrupt-fast"))) //中断函数前加这上这句,
void EXTI2_IRQHandler(void)
{
  if(EXTI_GetITStatus(EXTI_Line2)!=RESET)//产生中断
  {
      EXTI_ClearITPendingBit(EXTI_Line2);//清除中断标志
  }
}

/**
* @brief  中断函数
**/
__attribute__((interrupt("WCH-Interrupt-fast"))) //中断函数前加这上这句,
void EXTI0_IRQHandler(void)
{
  if(EXTI_GetITStatus(EXTI_Line0)!=RESET)//产生中断
  {
      gpio_interrupt_flag=1;
      EXTI_ClearITPendingBit(EXTI_Line0);//清除中断标志
  }
}


int main(void)
{
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//设置优先级分组为2
    USART_Printf_Init(115200);
    Delay_Init();
    gpio_init();
    while(1)
    {
        if (gpio_interrupt_flag==1) {//有中断产生
            gpio_interrupt_flag=0;
            printf("gpio_interrupt_flag\r\n");
        }
    }
}
```



假设设置PB6,PB7为上升沿中断

外部中断5-9共用一个中断;

```
#include "debug.h"
#include "ch32v30x.h"

/**
* @brief  init
* @param  None
* @retval None
* @example
**/
void gpio_init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0};
    EXTI_InitTypeDef EXTI_InitStructure = {0};
    NVIC_InitTypeDef NVIC_InitStructure = {0};

    /*打开挂载GPIO总线的复用时钟(GPIO除了输入输出的其它功能都叫做复用功能,所以需
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO | RCC_APB2Periph_GPIOB
    /*设置GPIO*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;//pin
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD; //下拉
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    /*设置GPIO*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7;//pin
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD; //下拉
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    /*设置作为中断线的GPIO引脚*/
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOB, GPIO_PinSource6);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOB, GPIO_PinSource7);
    /*设置GPIO中断*/
    EXTI_InitStructure.EXTI_Line = EXTI_Line6;//中断线
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;//中断模式
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;//上升沿触发
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;//使能
    EXTI_Init(&EXTI_InitStructure);

    /*设置GPIO中断*/
    EXTI_InitStructure.EXTI_Line = EXTI_Line7;//中断线
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;//中断模式
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;//上升沿触发
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;//使能
    EXTI_Init(&EXTI_InitStructure);

    /*配置中断优先等级*/
    NVIC_InitStructure.NVIC_IRQChannel = EXTI9_5_IRQn;//外部中断
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;//抢占式
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;//响应式优先级
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;//使能
    NVIC_Init(&NVIC_InitStructure);
}
```

```c
/**
* @brief  中断函数
**/
__attribute__((interrupt("WCH-Interrupt-fast"))) //中断函数前加这上这句,
void EXTI9_5_IRQHandler(void)
{
    if(EXTI_GetITStatus(EXTI_Line6)!=RESET)//产生中断
    {
        printf("666666666666\r\n");
        EXTI_ClearITPendingBit(EXTI_Line6);//清除中断标志
    }
    else if(EXTI_GetITStatus(EXTI_Line7)!=RESET)//产生中断
    {
        printf("777777777777\r\n");
        EXTI_ClearITPendingBit(EXTI_Line7);//清除中断标志
    }
}


int main(void)
{
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//设置优先级分组为2
    USART_Printf_Init(115200);
    Delay_Init();
    gpio_init();
    while(1)
    {

    }
}
```

假设设置PB10上升沿中断, PB11下降沿中断

外部中断10-15共用一个中断;

```c
#include "debug.h"
#include "ch32v30x.h"

/**
* @brief  init
* @param  None
* @retval None
* @example
**/
void gpio_init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0};
    EXTI_InitTypeDef EXTI_InitStructure = {0};
    NVIC_InitTypeDef NVIC_InitStructure = {0};

    /*打开挂载GPIO总线的复用时钟(GPIO除了输入输出的其它功能都叫做复用功能,所以需
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO | RCC_APB2Periph_GPIOB
    /*设置GPIO*/
```

```
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;//pin
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD; //下拉
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    /*设置GPIO*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;//pin
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; //上拉
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    /*设置作为中断线的GPIO引脚*/
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOB, GPIO_PinSource10);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOB, GPIO_PinSource11);
    /*设置GPIO中断*/
    EXTI_InitStructure.EXTI_Line = EXTI_Line10;//中断线
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;//中断模式
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;//上升沿触发
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;//使能
    EXTI_Init(&EXTI_InitStructure);

    /*设置GPIO中断*/
    EXTI_InitStructure.EXTI_Line = EXTI_Line11;//中断线
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;//中断模式
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;//下降沿触发
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;//使能
    EXTI_Init(&EXTI_InitStructure);

    /*配置中断优先等级*/
    NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;//外部中断
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;//抢占式优
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;//响应式优先级
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;//使能
    NVIC_Init(&NVIC_InitStructure);
}


/**
* @brief  中断函数
**/
__attribute__((interrupt("WCH-Interrupt-fast"))) //中断函数前加这上这句,
void EXTI15_10_IRQHandler(void)
{
    if(EXTI_GetITStatus(EXTI_Line10)!=RESET)//产生中断
    {
        printf("00000000000000000\r\n");
        EXTI_ClearITPendingBit(EXTI_Line10);//清除中断标志
    }
    else if(EXTI_GetITStatus(EXTI_Line11)!=RESET)//产生中断
    {
        printf("1111111111\r\n");
        EXTI_ClearITPendingBit(EXTI_Line11);//清除中断标志
    }
}


int main(void)
{
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//设置优先级分组为2
    USART_Printf_Init(115200);
    Delay_Init();
    gpio_init();
    while(1)
    {
```

```
        }
    }
```

分类: CH32V307(WCH单片机)学习开发

好文要顶 关注我 收藏该文

杨奉武
关注 - 1
粉丝 - 779

0 0

« 上一篇： 100-CH32V307(WCH单片机)学习开发-GPIO输出高低电平
» 下一篇： 2-STM32+ESP8266+Air302远程升级篇(自建物联网平台)-STM32通过air302使用http下载程序文件,升级程序(单片机程序轮训检查更新)

posted on 2022-04-30 03:23 杨奉武 阅读(32) 评论(0) 编辑 收藏 举报

刷新评论 刷新页面 返回顶部

发表评论

编辑 预览

支持 Markdown

自动补全

提交评论 退出

[Ctrl+Enter快捷键提交]

· 资管巨头蜂拥抄底中概股：未来头部公司股价或将超预期上涨
» 更多新闻...