ARC

外部中断

支持	www.armrunc.com http://armrunc.taobao.com

Change History

版本号	更改描述	作者	修改时间
1.0.0	第一版	armrunc	2012-3-6

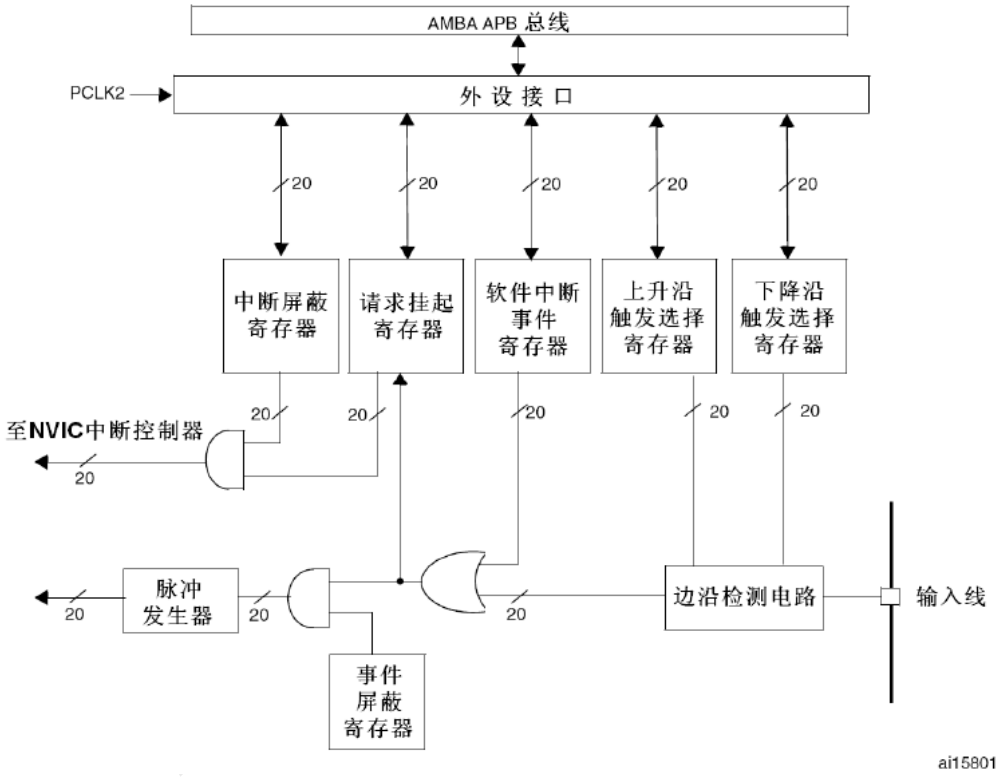
目录

第6章 外部中断

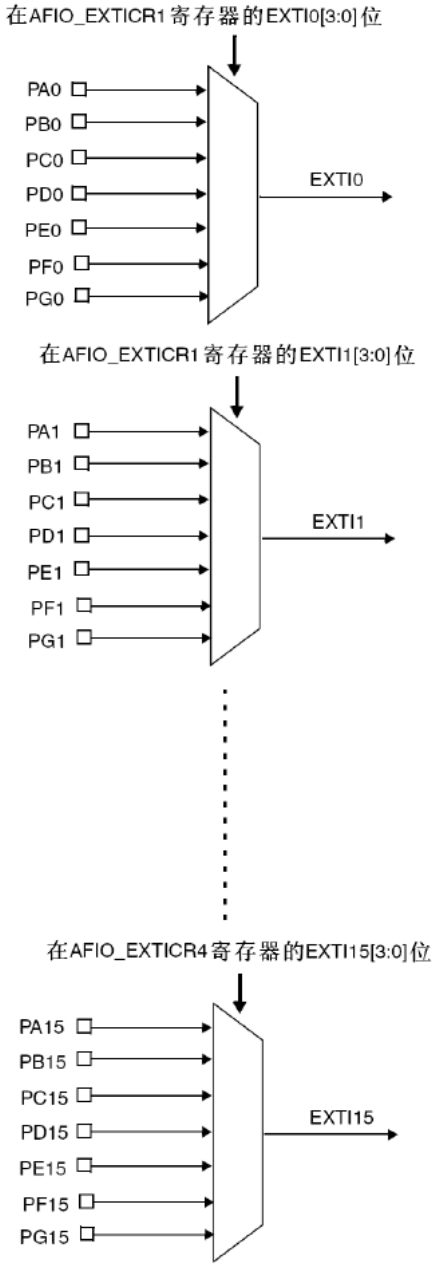
6.1 外部中断简介

外部中断/事件控制器含有 19 个沿检测器，用来产生中断请求。每个输入线可以独立地配置输入类型(脉冲或挂起)和对应的触发事件(上升沿或下降沿或者双边沿都触发)。每个输入线都可以独立地被屏蔽。挂起寄存器保持着状态线的中断请求。

外部中断控制器的框图如下：



其外部中断/事件映射如下：



另外三个 EXTI 线的连接方式如下：

- EXTI 线 16 连接到 PVD 输入
- EXTI 线 17 连接到 RTC 闹钟事件
- EXTI 线 18 连接到 USB 唤醒事件

6.2 外部中断应用实例 ----- 捕捉 GPIO 下降沿

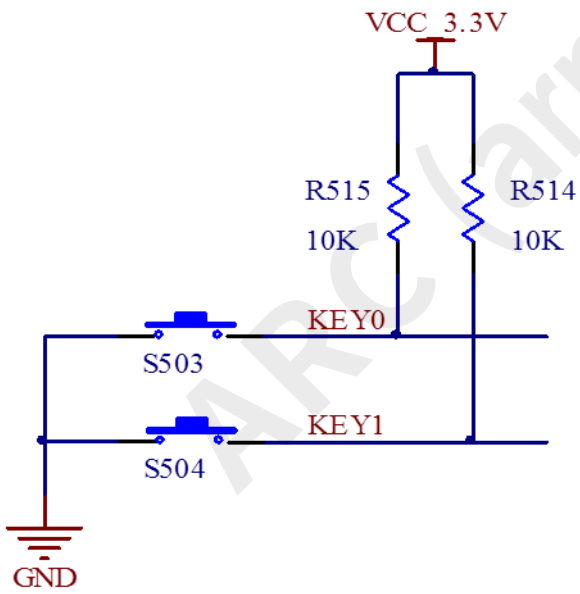
6.2.1 实例描述

本实例将外部中断线 2 和外部中断线 3 配置为下降沿触发输入，然后将 PC2 和 PC3 分别映射到外部中断线 2 和外部中断线 3，PC2 和 PC3 硬件外部有上拉电阻和按键，在按键没有按下时，输入检测为高电平，一旦按键按下，输入会检测到下降沿，在中断线处理函数 2 和 3 内会反转 LED。

6.2.2 硬件设计

该实例用到了 PC2 和 PC3，其在外部用了 10K 上拉电阻，在其被配置为悬浮输入时，检测的状态为高电平。

KEY0	PC2
KEY1	PC3



6.2.3 软件设计

本实例首先初始化 LED，然后再初始化按键口 PC2 和 PC3，将 PC2 和 PC3 配置为悬浮输入，然后分别映射到外部中断线 2 和外部中断线 3.

该实例的主要代码如下:

文件 Button_EXTI_main.c：

```
/**
  * @brief  Main program, EXTI example.
  * @param  None
  * @retval None
  */
int main(void)
{
    ARC_LED_Init();
    ARC_Button_Init();
    while (1)
    {
    }
}
```

文件 ARC_Button_EXTI.c：

```
/**
  * @brief  Initialize Button_EXTI.
  * @param  None
  * @retval None
  */
void ARC_Button_Init()
{
    ARC_Button_RCC_Init();
    ARC_Button_GPIO_Init();
    ARC_Button_NVIC_Init();
    ARC_Button_EXTI_Init();
}
```

文件 ARC_RCC.c：

```
/**
  * @brief  Configures Button clocks.
  * @param  None
  * @retval None
  */
void ARC_Button_RCC_Init(void)
{
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
}
```

文件 ARC_GPIO.c：

```
/**
  * @brief  Configures Button GPIO ports.
  * @param  None
```

```
   * @retval None
   */

/*
            ------------------
            | button0 | PC2 |
            ------------------
            | button1 | PC3 |
            ------------------
*/
void ARC_Button_GPIO_Init()
{
      GPIO_InitTypeDef GPIO_InitStructure;
      /* Configure the Button0 pin */
      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_Init(GPIOC, &GPIO_InitStructure);

      /* Configure the Button1 pin */
      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
      GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

文件 ARC_NVIC_API.c：

```
/**
   * @brief   Initialize NVIC of push button.
   * @param   None
   * @retval None
   */

void ARC_Button_NVIC_Init(void)
{
      NVIC_InitTypeDef NVIC_InitStructure;

      NVIC_InitStructure.NVIC_IRQChannel = EXTI2_IRQn;
      NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x0F;
      NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x0F;
      NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
      NVIC_Init(&NVIC_InitStructure);

      NVIC_InitStructure.NVIC_IRQChannel = EXTI3_IRQn;
      NVIC_Init(&NVIC_InitStructure);
```

```
}
```

文件 ARC_EXTI.c：

```
/**
  * @brief  Initialize EXTI of push button.
  * @param  None
  * @retval None
  */
void ARC_Button_EXTI_Init(void)
{
    EXTI_InitTypeDef EXTI_InitStructure;

    /* Connect Button EXTI Line to Button GPIO Pin */
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource2);

    /* Configure Button EXTI line */
    EXTI_InitStructure.EXTI_Line = EXTI_Line2;
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTI_InitStructure);

    /* Connect Button EXTI Line to Button GPIO Pin */
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource3);

    /* Configure Button EXTI line */
    EXTI_InitStructure.EXTI_Line = EXTI_Line3;
    EXTI_Init(&EXTI_InitStructure);
}
```

文件 stm32f10x_it.c

```
/**
  * @brief  This function handles External interrupt Line 2 request.
  * @param  None
  * @retval None
  */
void EXTI2_IRQHandler(void)
{
    if(EXTI_GetITStatus(EXTI_Line2) != RESET)
    {
        /* Toggle LED1 and LED2 */
        ARC_LED_Toggle(0);
        ARC_LED_Toggle(1);
        /* Clear the Key Button EXTI line pending bit */
```

```
            EXTI_ClearITPendingBit(EXTI_Line2);
        }
}

/**
  * @brief  This function handles External interrupt Line 3 request.
  * @param  None
  * @retval None
  */
void EXTI3_IRQHandler(void)
{
        if(EXTI_GetITStatus(EXTI_Line3) != RESET)
        {
            /* Toggle LED1 and LED2 */
            ARC_LED_Toggle(0);
            ARC_LED_Toggle(1);
            /* Clear the Key Button EXTI line pending bit */
            EXTI_ClearITPendingBit(EXTI_Line3);
        }
}
```