非阻塞式程序(拟freeRTOS)

2025年3月16日 21:41

- 阻塞：执行某段程序时，CPU因为需要等待延时或者等待某个信号而被迫处于暂停状态一段时间，程序执行时间较长或者时间不定
- 非阻塞：执行某段程序时，CPU不会等待，程序很快执行结束

```c
uint8_t Key_GetNum(void)
{
    uint8_t KeyNum = 0;
    if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_1) == 0)
    {
        Delay_ms(20);
        while (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_1) == 0);
        Delay_ms(20);
        KeyNum = 1;
    }
    if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_11) == 0)
    {
        Delay_ms(20);
        while (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_11) == 0);
        Delay_ms(20);
        KeyNum = 2;
    }
    return KeyNum;
}
```

```c
void LED1_Flash(void)
{
    GPIO_ResetBits(GPIOA, GPIO_Pin_1);
    Delay_ms(500);
    GPIO_SetBits(GPIOA, GPIO_Pin_1);
    Delay_ms(500);
}

void LED1_ON(void)
{
    GPIO_ResetBits(GPIOA, GPIO_Pin_1);
}

void LED1_OFF(void)
{
    GPIO_SetBits(GPIOA, GPIO_Pin_1);
}
```

- 定时中断，每隔20ms读取一次本次引脚值和上次引脚值
- 判断，如果本次是1，上次是0，则表示按键按下且当前处于刚松手的状态
- 置键码标志位，向主程序报告此事件


```c
void Key_Tick(void)
{
    static uint8_t Count;
    static uint8_t CurrState, PrevState;

    Count ++;
    if (Count >= 20)
    {
        Count = 0;

        PrevState = CurrState;
        CurrState = Key_GetState();

        if (CurrState == 0 && PrevState != 0)
        {
            Key_Num = PrevState;
        }
    }
}
```

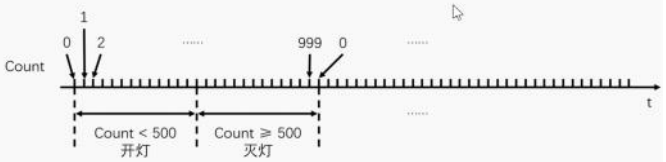
```c
void TIM2_IRQHandler(void)
{
    if (TIM_GetITStatus(TIM2, TIM_IT_Update) == SET)
    {
        Key_Tick();
        TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
    }
}
```

```c
void Key_Init(void)
{
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);

    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1 | GPIO_Pin_11;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
}

uint8_t Key_GetNum(void)
{
    uint8_t Temp;
    Temp = Key_Num;
    Key_Num = 0;
    return Temp;
}
```

```c
uint8_t Key_GetNum(void)
{
    uint8_t Temp;
    if (Key_Num)
    {
        Temp = Key_Num;
        Key_Num = 0;
        return Temp;
    }
    return 0;
}
```

注：UP实践发现，这样的写法会有bug，小概率会导致某些次按键按下无响应
因为如果中断正好发生在Temp=Key_Num和Key_Num=0这两句之间
并且中断里置了键码标志位
那么中断退出后，会立刻执行Key_Num=0，这样此次按键事件就会被忽略
所以，建议把代码修改为上图所示的样子
多加一个if判断，如果Key_Num不为0，再执行清零并返回的操作

- 定时中断，每隔1ms计次变量自增
- 计次变量计到周期值时，归零
- 判断，如果计次变量小于一个比较值，开灯，否则，关灯




```c
uint8_t LED1_Mode;

uint16_t LED1_Count;

void LED_Tick(void)
{
    if (LED1_Mode == 0)
    {
        LED1_OFF();
    }
    else if (LED1_Mode == 1)
    {
        LED1_Count ++;
        LED1_Count %= 1000;

        if (LED1_Count < 500)
        {
            LED1_ON();
        }
        else
        {
            LED1_OFF();
        }
    }
}
```

```c
while (1)
{
    KeyNum = Key_GetNum();

    if (KeyNum == 1)
    {
        FlashFlag = !FlashFlag;
    }

    if (FlashFlag)
    {
        LED1_SetMode(1);
    }
    else
    {
        LED1_SetMode(0);
    }

    OLED_ShowNum(1, 1, i ++, 5);
}
```

```c
void LED_Tick(void)
{
    if (LED1_Mode == 0)
    {
        LED1_OFF();
    }
    else if (LED1_Mode == 1)
    {
        LED1_Count ++;
        LED1_Count %= 1000;

        if (LED1_Count < 500)
        {
            LED1_ON();
        }
        else
        {
            LED1_OFF();
        }
    }
}
```

```c
        FlashFlag = !FlashFlag;
    }

    if (FlashFlag)
    {
        LED1_SetMode(1);
    }
    else
    {
        LED1_SetMode(0);
    }

    OLED_ShowNum(1, 1, i ++, 5);
}
```