对STM32基本知识的详细剖析

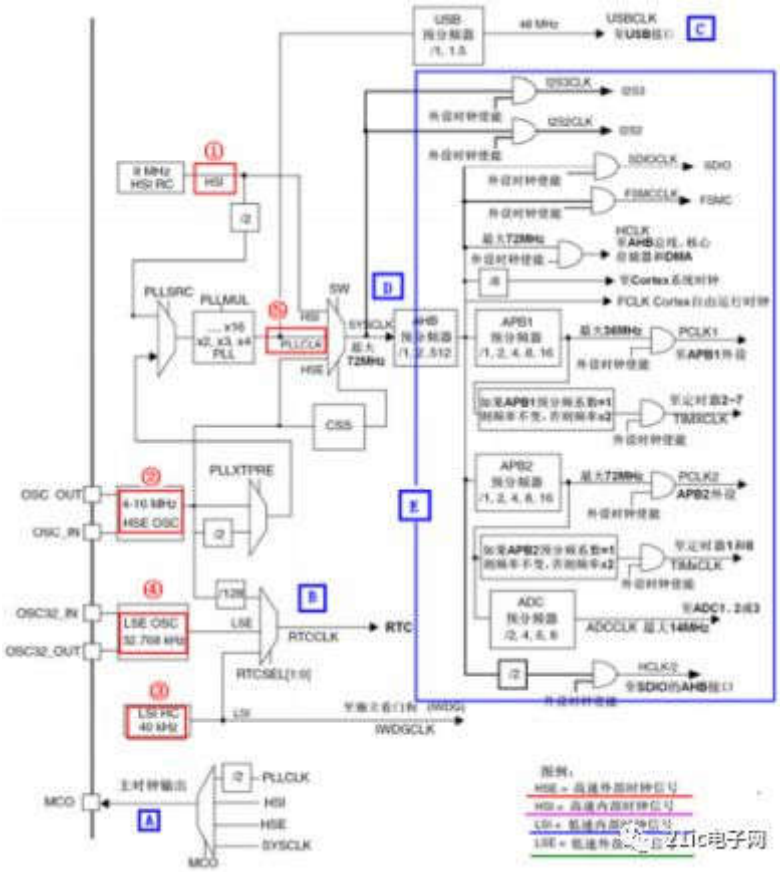
21ic电子网 发表于 2018-01-16 14:29:17

电子发烧友网 熊掌号
收听电子行业动态，抢先知晓半导体行业

STM32是一种功能比较强大的32位单片机，广泛应用于各种嵌入式设备中，由于它的普及性及丰富的资源，受到广大嵌入式开发者的喜欢，但要想学好用好STM32也并非易事，毕竟，相比8位、16位产品，STM32要复杂得多。

STM32的时钟

众所周知STM32有5个时钟源HSI、HSE、LSI、LSE、PLL，其实它只有四个，因为从下图中可以看到PLL都是由HSI或HSE提供的。



其中，高速时钟(HSE和HSI)提供给芯片主体的主时钟.低速时钟(LSE和LSI)只是提供给芯片中的RTC(实时时钟)及独立看门狗使用，图中可以看出高速时钟也可以提供给RTC。内部时钟是在芯片内部RC振荡器产生的，起振较快，所以时钟在芯片刚上电的时候，默认使用内部高速时钟。而外部时钟信号是由外部的晶振输入的，在精度和稳定性上都有很大优势，所以上电之后我们再通过软件配置，转而采用外部时钟信号.

高速外部时钟(HSE)：以外部晶振作时钟源，晶振频率可取范围为4~16MHz，我们一般采用8MHz的晶振。

高速内部时钟(HSI)： 由内部RC振荡器产生，频率为8MHz，但不稳定。

低速外部时钟(LSE)：以外部晶振作时钟源，主要提供给实时时钟模块，所以一般采用32.768KHz。

低速内部时钟(LSI)：由内部RC振荡器产生，也主要提供给实时时钟模块，频率大约为40KHz。

OSC_OUT和OSC_IN开始，这两个引脚分别接到外部晶振8MHz,第一个分频器PLLXTPRE，遇到开关PLLSRC(PLL entry clock source)，我们可以选择其输出，输出为外部高速时钟(HSE)或是内部高速时钟(HSI)。这里选择输出为HSE，接着遇到锁相环PLL，具有倍频作用，在这里我们可以输入倍频因子PLLMUL，要是想超频，就得在这个寄存器上做手脚啦。经过PLL的时钟称为PLLCLK。倍频因子我们设定为9倍频，也就是说，经过PLL之后，我们的时钟从原来8MHz的 HSE变为72MHz的PLLCLK。紧接着又遇到了一个开关SW，经过这个开关之后就是STM32的系统时钟(SYSCLK)了。通过这个开关，可以切换SYSCLK的时钟源，可以选择为HSI、PLLCLK、HSE。我们选择为PLLCLK时钟，所以SYSCLK就为72MHz了。PLLCLK在输入到SW前，还流向了USB预分频器，这个分频器输出为USB外设的时钟(USBCLK)。回到SYSCLK，SYSCLK经过AHB预分频器，分频后再输入到其它外设。如输出到称为HCLK、FCLK的时钟，还直接输出到SDIO外设的SDIOCLK时钟、存储器控制器FSMC的FSMCCLK时钟，和作为APB1、APB2的预分频器的输入端。GPIO外设是挂载在APB2总线上的， APB2的时钟是APB2预分频器的输出，而APB2预分频器的时钟来源是AHB预分频器。因此，把APB2预分频器设置为不分频，那么我们就可以得到GPIO外设的时钟也等于HCLK，为72MHz了。

SYSCLK：系统时钟，STM32大部分器件的时钟来源。主要由AHB预分频器分配到各个部件。

HCLK:由AHB预分频器直接输出得到，它是高速总线AHB的时钟信号，提供给存储器，DMA及cortex内核，是cortex内核运行的时钟，cpu主频就是这个信号，它的大小与STM32运算速度，数据存取速度密切相关。

FCLK：同样由AHB预分频器输出得到，是内核的“自由运行时钟”。“自由”表现在它不来自时钟 HCLK，因此在HCLK时钟停止时 FCLK 也继续运行。它的存在，可以保证在处理器休眠时，也能够采样和到中断和跟踪休眠事件 ，它与HCLK互相同步。

PCLK1：外设时钟，由APB1预分频器输出得到，最大频率为36MHz，提供给挂载在APB1总线上的外设，APB1总线上的外设如下：

RCC_APB1Periph_TIM2 TIM2时钟

RCC_APB1Periph_TIM3 TIM3时钟

RCC_APB1Periph_TIM4 TIM4时钟

RCC_APB1Periph_WWDG WWDG时钟

RCC_APB1Periph_SPI2 SPI2时钟

RCC_APB1Periph_USART2 USART2时钟

RCC_APB1Periph_USART3 USART3时钟

RCC_APB1Periph_I2C1 I2C1时钟

RCC_APB1Periph_I2C2 I2C2时钟

RCC_APB1Periph_USB USB时钟

RCC_APB1Periph_CAN CAN时钟

RCC_APB1Periph_BKP BKP时钟

RCC_APB1Periph_PWR PWR时钟

RCC_APB1Periph_ALL 全部APB1外设时钟

PCLK2：外设时钟，由APB2预分频器输出得到，最大频率可为72MHz，提供给挂载在APB2总线上的外设，APB2总线上的外设如下：

RCC_APB2Periph_AFIO 功能复用IO时钟

RCC_APB2Periph_GPIOA GPIOA时钟

RCC_APB2Periph_GPIOB GPIOB时钟

RCC_APB2Periph_GPIOC GPIOC时钟

RCC_APB2Periph_GPIOD GPIOD时钟

RCC_APB2Periph_GPIOE GPIOE时钟

RCC_APB2Periph_ADC1 ADC1时钟

RCC_APB2Periph_ADC2 ADC2时钟

RCC_APB2Periph_TIM1 TIM1时钟

RCC_APB2Periph_SPI1 SPI1时钟

RCC_APB2Periph_USART1 USART1时钟

RCC_APB2Periph_ALL 全部APB2外设时钟

STM32的几种输入模式

STM32有4种输入模式：

1）模拟输入 GPIO_AIN：用于AD转换

2）浮空输入 GPIO_IN_FLOATING：引脚处于浮空模式，电平状态是不确定的。外部信号输入什么，IO口就是什么状态。

3）上拉输入 GPIO_IPU：防止IO口出现不确定的状态，比如，当IO口悬空时，就会通过内部的上拉电阻将该点钳位在高电平。

4）下拉输入 GPIO_IPD：功能与上拉电阻类似，防止IO口出现不确定的状态，比如，当IO口悬空时，就会通过内部的下拉电阻将该点钳位在低电平。

STM32中空的I/O管脚是高电平还是低电平取决于具体情况。

1、IO端口复位后处于浮空状态，也就是其电平状态由外围电路决定。

2、STM32上电复位瞬间I/O口的电平状态默认是浮空输入，因此是高阻。做到低功耗。

3、STM32的IO管脚配置口默认为浮空输入，把选择权留给用户，这是一个很大的优势：一方面浮空输入确保不会出现用户不希望的默认电平(此时电平取决于用户的外围电路);另一方面降低了功耗，因为不管是上拉还是下拉都会有电流消耗。从另一个角度来看，不管I/O管脚

的默认配置如何，还是需要在输出的管脚外加上拉或下拉，这是为了保证芯片上电期间和复位时输出的管脚始终处于已知的电平。

4、在没有任何操作的情况下，STM32通用推挽输出模式的引脚默认低电平，也就是有电的状态。所以在配置的时候通常会先把引脚的电平设置拉高，让电路不产生电流。有电到没电这一过程也就是引脚电平从低到高的过程。

5、STM32的I/O管脚有两种：TTL和CMOS，所有管脚都兼容TTL和CMOS电平。也就是说从输入识别电压上看，所有管脚不管是TTL管脚还是CMOS管脚都可以识别TTL或CMOS电平。

STM32的中断系统

在STM32中，中断数量大大增加，而且中断的设置也更加复杂。

1 基本概念

ARM Coetex-M3内核共支持256个中断，其中16个内部中断，240个外部中断和可编程的256级中断优先级的设置。STM32目前支持的中断共84个（16个内部+68个外部），还有16级可编程的中断优先级的设置，仅使用中断优先级设置8bit中的高4位。

STM32可支持68个中断通道，已经固定分配给相应的外部设备，每个中断通道都具备自己的中断优先级控制字节PRI_n(8位，但是STM32中只使用4位，高4位有效)，每4个通道的8位中断优先级控制字构成一个32位的优先级寄存器。68个通道的优先级控制字至少构成17个32位的优先级寄存器。

4bit的中断优先级可以分成2组，从高位看，前面定义的是抢占式优先级，后面是响应优先级。按照这种分组，4bit一共可以分成5组

第0组：所有4bit用于指定响应优先级；

第1组：最高1位用于指定抢占式优先级，后面3位用于指定响应优先级；

第2组：最高2位用于指定抢占式优先级，后面2位用于指定响应优先级；

第3组：最高3位用于指定抢占式优先级，后面1位用于指定响应优先级；

第4组：所有4位用于指定抢占式优先级。

所谓抢占式优先级和响应优先级，他们之间的关系是：具有高抢占式优先级的中断可以在具有低抢占式优先级的中断处理过程中被响应，即中断嵌套。

当两个中断源的抢占式优先级相同时，这两个中断将没有嵌套关系，当一个中断到来后，如果正在处理另一个中断，这个后到来的中断就要等到前一个中断处理完之后才能被处理。如果这两个中断同时到达，则中断控制器根据他们的响应优先级高低来决定先处理哪一个；如果他们的抢占式优先级和响应优先级都相等，则根据他们在中断表中的排位顺序决定先处理哪一个。每一个中断源都必须定义2个优先级。

有几点需要注意的是：

1）如果指定的抢占式优先级别或响应优先级别超出了选定的优先级分组所限定的范围，将可能得到意想不到的结果；

2）抢占式优先级别相同的中断源之间没有嵌套关系；

3）如果某个中断源被指定为某个抢占式优先级别，又没有其它中断源处于同一个抢占式优先级别，则可以为这个中断源指定任意有效的响应优先级别。

2 GPIO外部中断

STM32中，每一个GPIO都可以触发一个外部中断，但是，GPIO的中断是以组位一个单位的，同组间的外部中断同一时间只能使用一个。比如说，PA0，PB0，PC0，PD0，PE0，PF0，PG0这些为1组，如果我们使用PA0作为外部中断源，那么别的就不能够再使用了，在此情况下，我们只能使用类似于PB1，PC2这种末端序号不同的外部中断源。每一组使用一个中断标志EXTIx。EXTI0 – EXTI4这5个外部中断有着自己的单独的中断响应函数，EXTI5-9共用一个中断响应函数，EXTI10-15共用一个中断响应函数。

对于中断的控制，STM32有一个专用的管理机构：NVIC。对于NVIC的详细解释，可以参考《ARM Cortex-M3权威指南》，Joseph Yiu著，宋岩译，北京航空航天大学出版社出版，第8章NVIC与中断控制。中断的使能，挂起，优先级，活动等等部都是NVIC在管理的。因为我学习STM32重点在于如何开发程序，所以内部的一些东西，在此我就不详细说明了，有感兴趣的可以参看上面提到的那本数。

STM32外部中断使用实例

其实上面那些基本概念和知识只是对STM32的中断系统有一个大概的认识，用程序说话将会更能够加深如何使用中断。**使用外部中断的基本步骤如下：**

1. 设置好相应的时钟；
2. 设置相应的中断；
3. IO口初始化；
4. 把相应的IO口设置为中断线路（要在设置外部中断之前）并初始化；
5. 在选择的中断通道的响应函数中中断函数。

由于我用的奋斗开发板没有引出相应的芯片引脚，所以只能用按键来触发相应的中断。根据原理图，K1/K2/K3连接的是PC5/PC2/PC3，因此我将用EXTI5/EXTI2/EXTI3三个外部中断。PB5/PD6/PD3分别连接了三个LED灯。中断的效果是按下按键，相应的LED灯将会被点亮。

1. 设置相应的时钟

首先需要打开GPIOB、GPIOC和GPIOE（因为按键另外一端连接的是PE口）。然后由于是要用于触发中断，所以还需要打开GPIO复用的时钟。相应的函数在GPIO的学习笔记中有了详细了解释。详细代码如下：

```
void RCC_cfg()
{
    //打开PE PD PC PB端口时钟，并且打开复用时钟
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE| RCC_APB2Periph_GPIOC | RCC_APB2Periph_GPIOD | RCC_APB2Periph_GPIOB |RCC_APB2Periph_AFIO, ENABLE);
}
```

设置相应的时钟所需要的RCC函数在stm32f10x_rcc.c中，所以要在工程中添加此文件。

2. 设置好相应的中断

设置相应的中断实际上就是设置NVIC，在STM32的固件库中有一个结构体NVIC_InitTypeDef，里面有相应的标志位设置，然后再用NVIC_Init()函数进行初始化。详细代码如下：

```
void NVIC_cfg()
{
    NVIC_InitTypeDefNVIC_InitStructure;
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);                    //选择中断分组2
NVIC_InitStructure.NVIC_IRQChannel= EXTI2_IRQChannel;     //选择中断通道2
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority= 0; //抢占式中断优先级设置为0
    NVIC_InitStructure.NVIC_IRQChannelSubPriority= 0;        //响应式中断优先级设置为0
    NVIC_InitStructure.NVIC_IRQChannelCmd=ENABLE;                    //使能中断
    NVIC_Init(&NVIC_InitStructure);
    NVIC_InitStructure.NVIC_IRQChannel=EXTI3_IRQChannel;          //选择中断通道3
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority= 1; //抢占式中断优先级设置为1
    NVIC_InitStructure.NVIC_IRQChannelSubPriority= 1;  //响应式中断优先级设置为1
    NVIC_InitStructure.NVIC_IRQChannelCmd=ENABLE;                    //使能中断
    NVIC_Init(&NVIC_InitStructure);
    NVIC_InitStructure.NVIC_IRQChannel= EXTI9_5_IRQChannel;  //选择中断通道5
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority= 2; //抢占式中断优先级设置为2
    NVIC_InitStructure.NVIC_IRQChannelSubPriority= 2;  //响应式中断优先级设置为2
    NVIC_InitStructure.NVIC_IRQChannelCmd=ENABLE;                    //使能中断
    NVIC_Init(&NVIC_InitStructure);
}
```

由于有3个中断，因此根据前文所述，需要有3个bit来指定抢占优先级，所以选择第2组。又由于EXTI5-9共用一个中断响应函数，所以EXTI5选择的中断通道是EXTI9_5_IRQChannel，详细信息可以在头文件中查询得到。用到的NVIC相关的库函数在stm32f10x_nivc.c中，需要将此文件复制并添加到工程中。具体位置可以查看关于GPIO的笔记。这段代码编译起来没有任何问题，但是在链接的时候就会报错，需要把STM32F10xR.LIB加入工程中，具体位置在...\Keil\ARM\RV31\LIB\ST\STM32F10xR.LIB。

3. IO口初始化

```
void IO_cfg()
{
    GPIO_InitTypeDefGPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_2;                    //选择引脚2
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;          //输出频率最大50MHz
   GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;              //带上拉电阻输出
    GPIO_Init(GPIOE,&GPIO_InitStructure);
    GPIO_ResetBits(GPIOE,GPIO_Pin_2);                    //将PE.2引脚设置为低电平输出
    GPIO_InitStructure.GPIO_Pin= GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_5; //选择引脚2 3 5
    GPIO_InitStructure.GPIO_Mode= GPIO_Mode_IN_FLOATING; //选择输入模式为浮空输入
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;          //输出频率最大50MHz
    GPIO_Init(GPIOC,&GPIO_InitStructure);                    //设置PC.2/PC.3/PC.5
    GPIO_InitStructure.GPIO_Pin= GPIO_Pin_3 |GPIO_Pin_6;              //选择引脚3 6
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;          //输出频率最大50MHz
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;              //带上拉电阻输出
    GPIO_Init(GPIOD,&GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_5;                    //选择引脚5
```

```
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;          //输出频率最大50MHz
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;                  //带上拉电阻输出
    GPIO_Init(GPIOB,&GPIO_InitStructure);
}
```

其中连接外部中断的引脚需要设置为输入状态，而连接LED的引脚需要设置为输出状态，初始化PE.2是为了使得按键的另外一端输出低电平。GPIO中的函数在stm32f10x_gpio.c中。

4. 把相应的IO口设置为中断线路

由于GPIO并不是专用的中断引脚，因此在用GPIO来触发外部中断的时候需要设置将GPIO相应的引脚和中断线连接起来，具体代码如下：

```
void EXTI_cfg()
{
    EXTI_InitTypeDefEXTI_InitStructure;
    //清空中断标志
    EXTI_ClearITPendingBit(EXTI_Line2);
    EXTI_ClearITPendingBit(EXTI_Line3);
    EXTI_ClearITPendingBit(EXTI_Line5);
    //选择中断管脚PC.2 PC.3 PC.5
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOC,GPIO_PinSource2);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOC,GPIO_PinSource3);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOC,GPIO_PinSource5);
    EXTI_InitStructure.EXTI_Line= EXTI_Line2 | EXTI_Line3 | EXTI_Line5; //选择中断线路2 3 5
    EXTI_InitStructure.EXTI_Mode= EXTI_Mode_Interrupt; //设置为中断请求，非事件请求
    EXTI_InitStructure.EXTI_Trigger= EXTI_Trigger_Rising_Falling; //设置中断触发方式为上下降沿触发
    EXTI_InitStructure.EXTI_LineCmd=ENABLE;                  //外部中断使能
    EXTI_Init(&EXTI_InitStructure);
}
```

EXTI_cfg中需要调用到的函数都在stm32f10x_exti.c。

5. 写中断响应函数

STM32不像C51单片机那样，可以用过interrupt关键字来定义中断响应函数，STM32的中断响应函数接口存在中断向量表中，是由启动代码给出的。默认的中断响应函数在stm32f10x_it.c中。因此我们需要把这个文件加入到工程中来。

在这个文件中，我们发现，很多函数都是只有一个函数名，并没有函数体。我们找到EXTI2_IRQHandler()这个函数，这就是EXTI2中断响应的函数。我的目标是将LED灯点亮，所以函数体其实很简单：

```
voidEXTI2_IRQHandler(void)
{
    //点亮LED灯
    GPIO_SetBits(GPIOD,GPIO_Pin_6);
    //清空中断标志位，防止持续进入中断
    EXTI_ClearITPendingBit(EXTI_Line2);
}
voidEXTI3_IRQHandler(void)
{
    GPIO_SetBits(GPIOD,GPIO_Pin_3);
    EXTI_ClearITPendingBit(EXTI_Line3);
}
voidEXTI9_5_IRQHandler(void)
{
    GPIO_SetBits(GPIOB,GPIO_Pin_5);
    EXTI_ClearITPendingBit(EXTI_Line5);
}
```

由于EXTI5-9是共用一个中断响应函数，因此所有的EXTI5 – EXTI9的响应函数都写在这个里面。

6. 写主函数

```
#include"stm32f10x_lib.h"
void RCC_cfg();
void IO_cfg();
void EXTI_cfg();
void NVIC_cfg();
int main()
{
    RCC_cfg();
```

```
    IO_cfg();
    NVIC_cfg();
    EXTI_cfg();
    while(1);
}
```

main函数前是函数声明，main函数函数体中都是调用初始化配置函数，然后进入死循环，等待中断响应。

stm32 时钟 中断

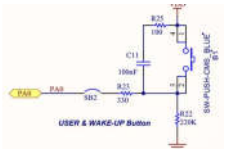
107次阅读 2018-03-13

求助stm32的IAP超级终端的升级问题！

116次阅读 2018-03-13

（纯干货）使用STM32测量频率和占空比的几种方法

81次阅读 2018-03-13



用STM32测量频率和占空比的几种方法

652次阅读 2018-03-13

stm32

83次阅读 2018-03-12

基于STM32的无线视频传输系统

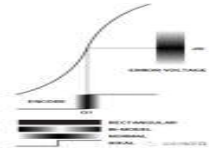
151次阅读 2018-03-12

手把手教你写单片机定时器中断程序

166次阅读 2018-03-12

【零基础学习STM32】第七讲：WWDG看门狗实验——复位ARM

113次阅读 2018-03-12



时钟抖动的4大根本原因及3种查看途径

151次阅读 2018-03-12

MM32F103C8T6资料

66次阅读 2018-03-10

485+STM32+SIM800C

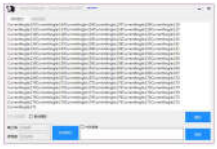
202次阅读 2018-03-10

哪位大神能帮忙看下stm8 time4的问题，烧录后进不了中断

161次阅读 2018-03-09

stm32使用中出现的警告

29次阅读 2018-03-09



基于STM32的三轴数字罗盘HMC5883L模块的测试方案

166次阅读 2018-03-08

意法半导体的新STM32让物联网设备快速连接云服务

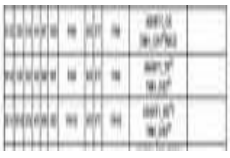
198次阅读 2018-03-02

一文了解stm32使用DMA模块的相关操作

2908次阅读 2018-02-27

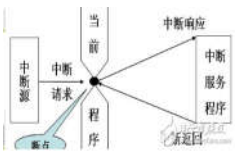
ATxmage入门应用之时钟模块和GPIO模块的介绍

398次阅读 2018-02-27



关于STM32的 一个TIM1 的PWM程序和PWM简单使用

252次阅读 2018-02-24



解析单片机中断处理过程、中断返回、中断撤除

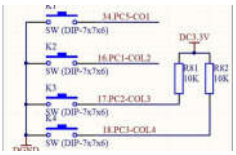
1370次阅读 2018-02-23

STM32定时器产生PWM彻底应用

303次阅读 2018-02-11

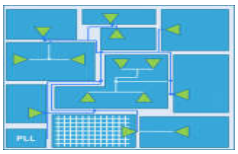
STM32基础PWM输出

162次阅读 2018-02-10



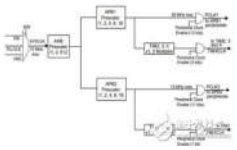
STM32的GPIO输入编程实例之读取按键状态

631次阅读 2018-02-10



LUCT工具主要特性及其使用教程

397次阅读 2018-02-10

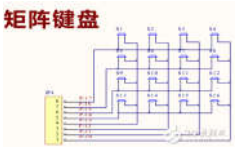


详细介绍定时器和定时器中断

914次阅读 2018-02-09

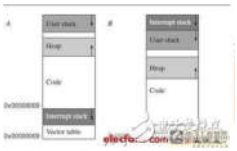
stm32寄存器版矩阵键盘库函数（附详细注释）

115次阅读 2018-02-09



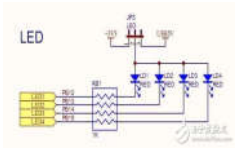
stm32矩阵键盘原理图及程序介绍

752次阅读 2018-02-09



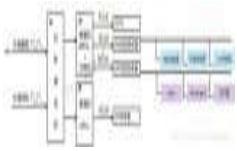
对ARM异常中断的集中情况进行总结，并给出了一些解决方法

912次阅读 2018-02-08



STM32的GPIO输出编程实例之点亮三色LED

1615次阅读 2018-02-07



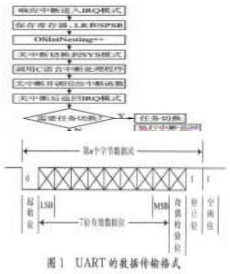
介绍arm9时钟与定时器

491次阅读 2018-02-07

ARM S3C4510B系统的异常中断机制解析

184次阅读 2018-02-03

基于uC/OSII和ARM7 中断机制的IRQ中断响应机制改进及优化解决方案



基于μC/OSII和ARM7中断机制的IRQ中断响应机制改进及优化解决方案

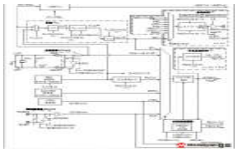
149次阅读 2018-02-03

串口通信协议stm32

195次阅读 2018-02-01

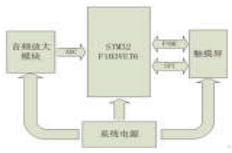
Microchip PIC16F1619实验：PIC16的双速启动教程

419次阅读 2018-02-01



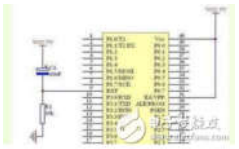
MCU的基础--时钟系统学习教程

153次阅读 2018-02-01



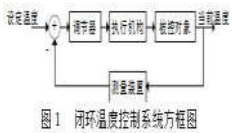
基于STM32实现孤立词语音识别系统

309次阅读 2018-01-31



针对初级工程师经常犯的一个小错误，做了针对性的纠正

703次阅读 2018-01-30



一款基于stm32的毕业设计方案

465次阅读 2018-01-29

基于stm32的电子秤方案大全（二款stm32的电子秤设计方案）

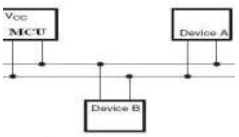
282次阅读 2018-01-29

详细介绍下与时钟相关的命令

483次阅读 2018-01-27

基于stm32的8m晶振不起振的原因解析

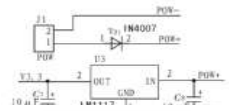
1467次阅读 2018-01-26



提供多主机功能，STM32的I2C通信简析

2854次阅读 2018-01-26

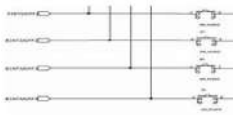
基于STM32的高性能低功耗的中文人机界面系统



322次阅读 2018-01-25

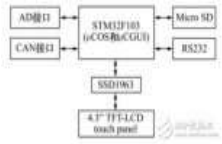
对于ROM与RAM的深度解析

874次阅读 2018-01-25



每一个GPIO如何配置成一个外部中断触发源

1857次阅读 2018-01-25



基于STM32F103的水动力测控系统设计的解解决方案

256次阅读 2018-01-25

介绍如何通过意法的STM32 MCU实现用DMA完成多通道的AD采样功能

1279次阅读 2018-01-24

高集成度编程工具软件STM32代码烧录编程实战

1734次阅读 2018-01-23

了解时钟基础知识是成为数字设计的软件工程师最基础的部分

626次阅读 2018-01-22