# 前言：

**对于stm32而言，使用stm32cube和keil合作的开发方式目前而言最为方便。而其中的代码、思想对快速开发十分有用，忽略底层的硬件流通，从寄存器的操作和hal库的结合开始开发，故从实践方向来总结stm32的各项功能与使用。**

**总结结构如下：**

**1.有关stm32cube软件的基本使用；**

**2.有关keil软件的基本的使用；**

**3.关于GPIO的基本操作**

**3.1led灯的点亮**

**3.2按键的输入检测**

**3.3按键触发外部中断**

**4.关于串口通讯的总结**

**4.1实现基础的数据接收传输**

**4.2实现中断接收和中断发送**

**4.3实现DMA方式接收与发送（数据定长）**

**4.4实现完善的DMA方式接收、DMA方式发送同时结合空闲中断，实现不定长数据传输**

**5.关于定时器**

**5.1滴答定时器实现us级的延时**

**5.2自带库HAL实现ms级的延时**

**5.3实现普通定时器控制led灯闪烁**

**5.4实现普通定时器输出PWM波形**

**6.关于DMA方式的ADC采样**

**剩余未涉及内容：  
1）I2C通讯协议**

**2）SPI通讯协议**

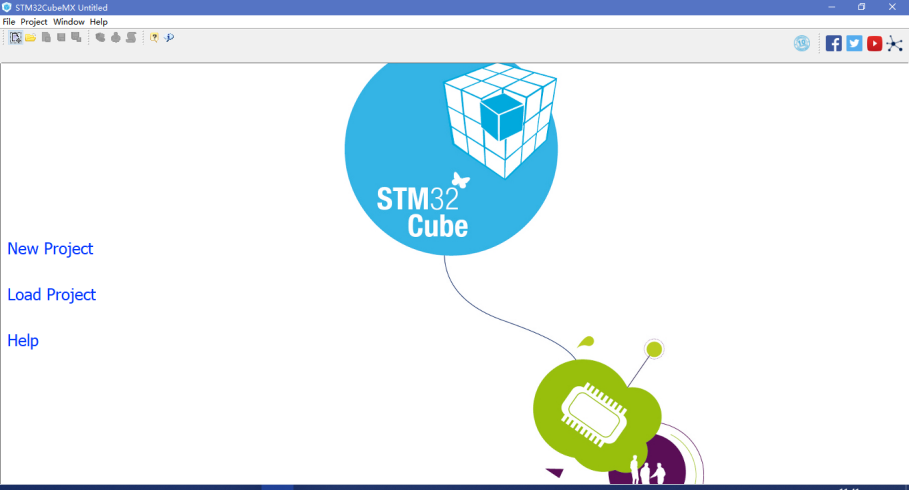
**3）CAN总线控制器**

**4）SD卡读取与写入**

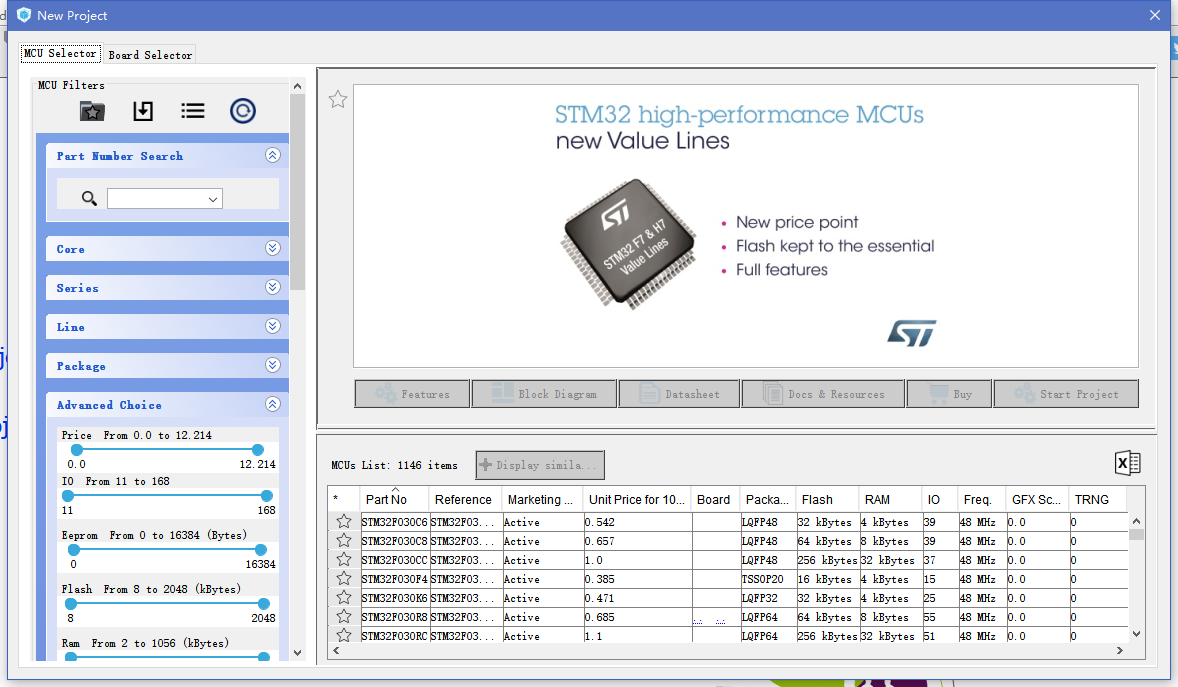
**5）以太网及LwIP协议栈移植**

**6）关于LCD、OLED、WIFI模块、摄像头等外设的驱动使用**

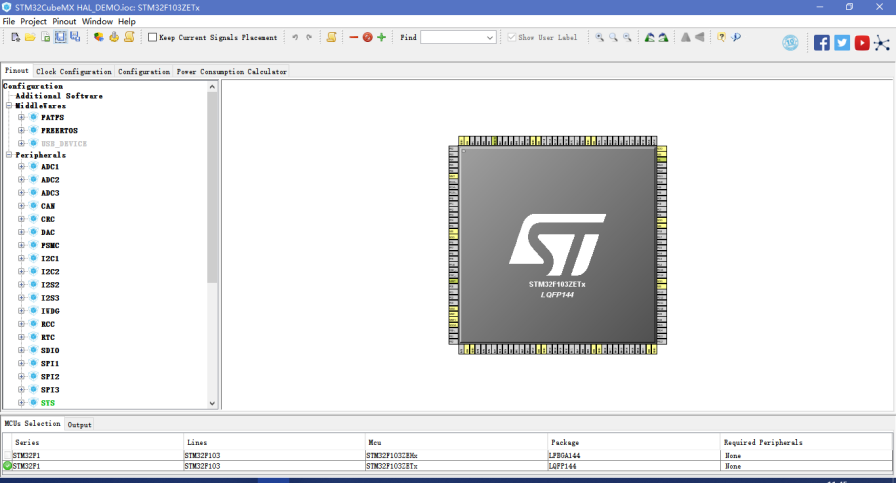
# 关于stm32cube



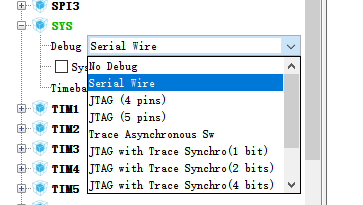
**初始界面**



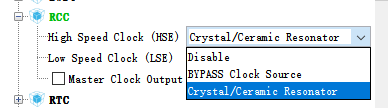
**芯片挑选界面**



**以上界面为芯片的引脚选择界面，右侧为芯片的封装图，左侧为芯片的功能选择。**

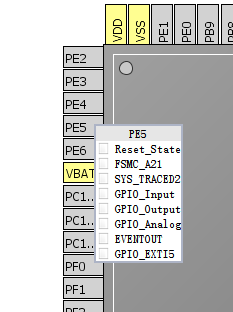
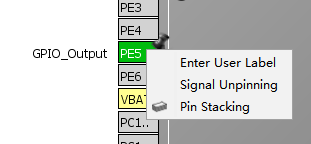


**为下载以及调试的方法选择，根据选择确定引脚**



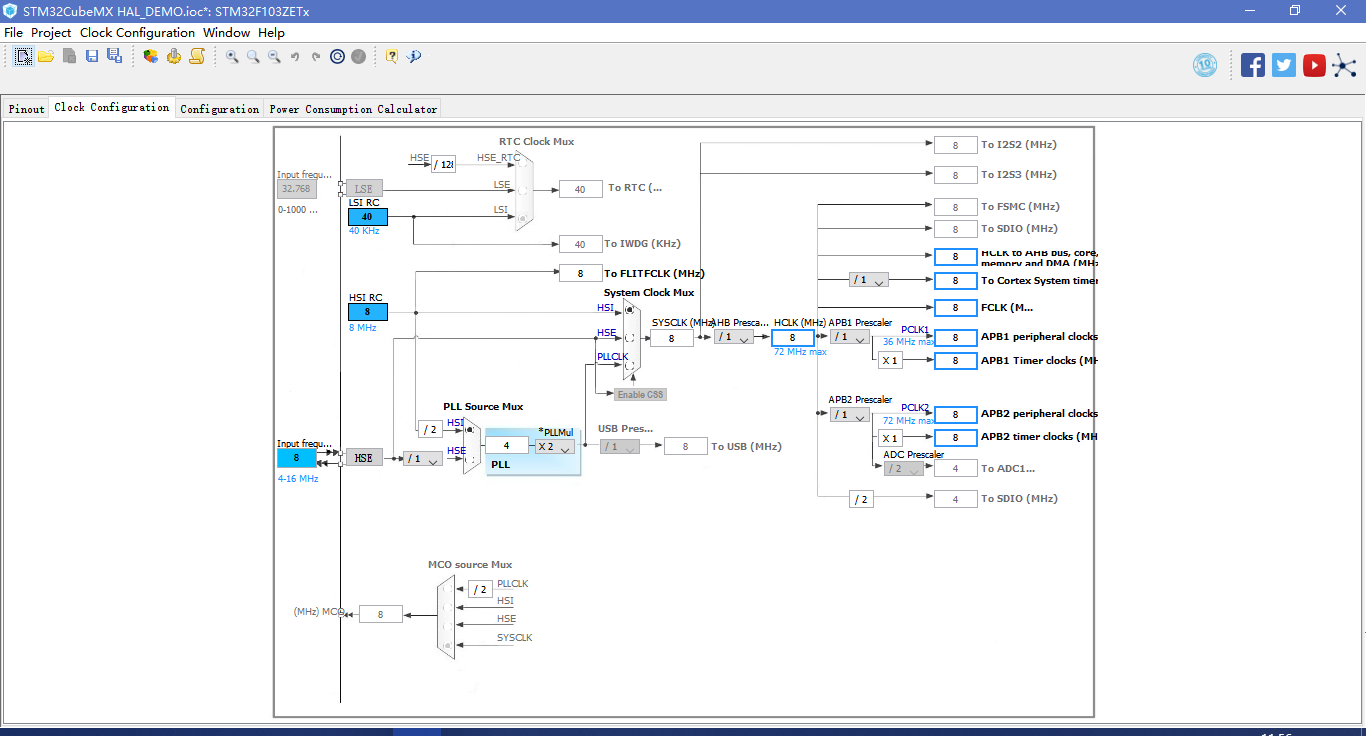
**为时钟的选择，选择常为外部高速时钟**

**以上两项配置，均为基本配置**

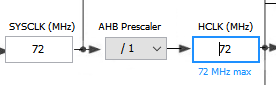
****

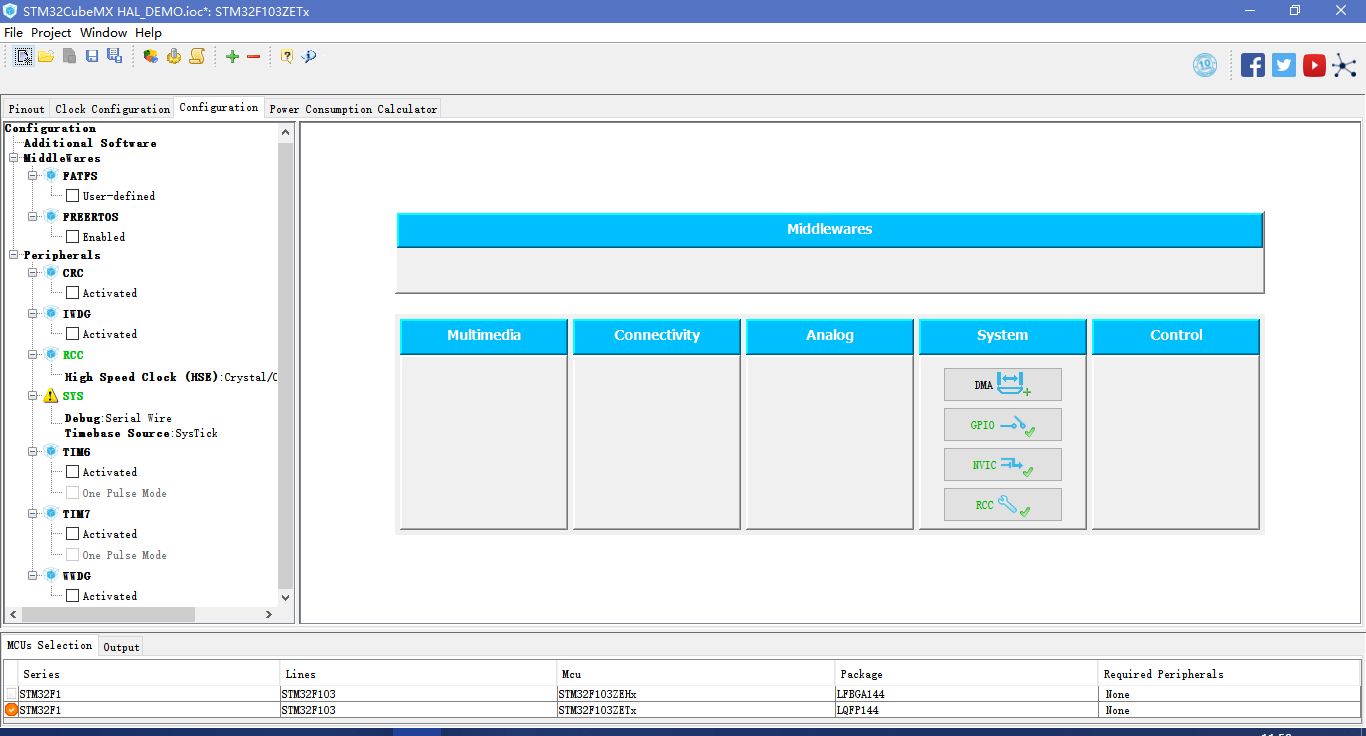
**当左键点击引脚时，出现左图，为选择功能界面，选择后右键引脚，可修改引脚名称**

**注意：当引脚被占用时，会导致部分功能变黄、变红。**

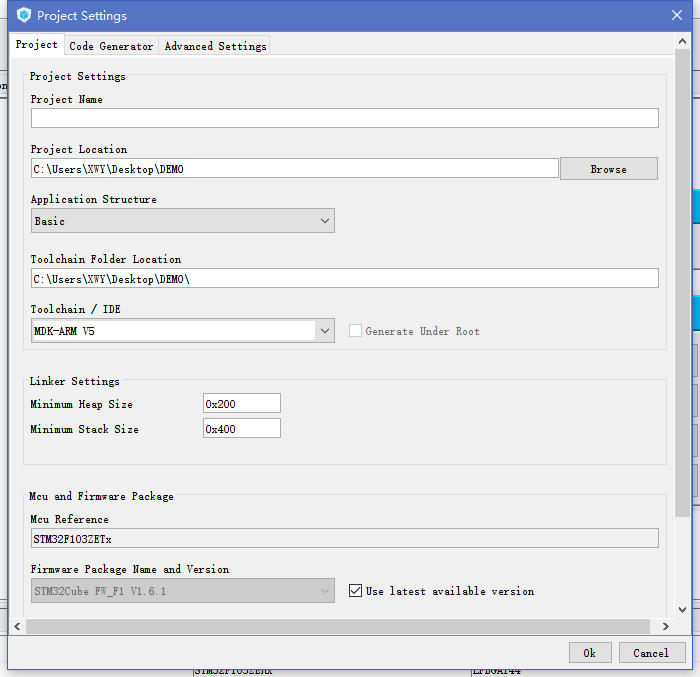


**以上为时钟树界面**

**常将HCLK设为最大即可**

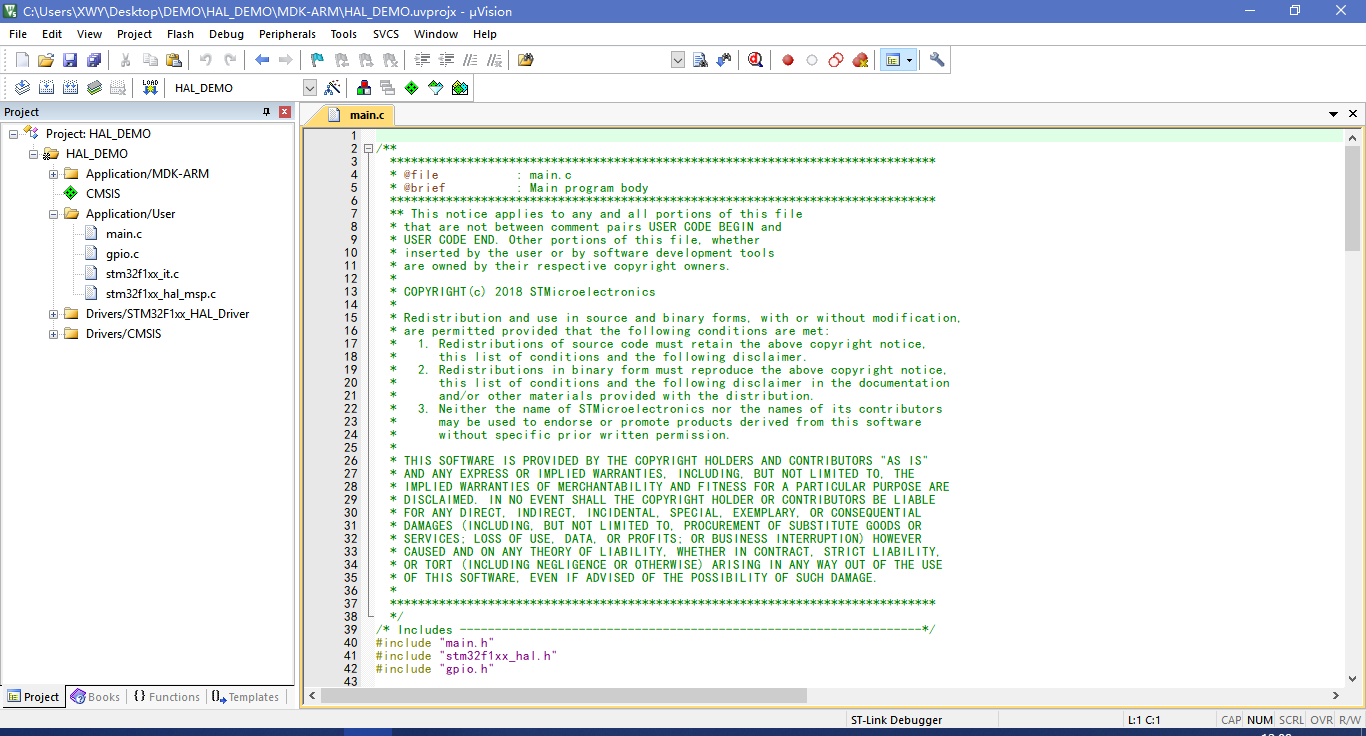


**以上为引脚的信息配置界面，可设置中断、DMA、输入输出设置等等**

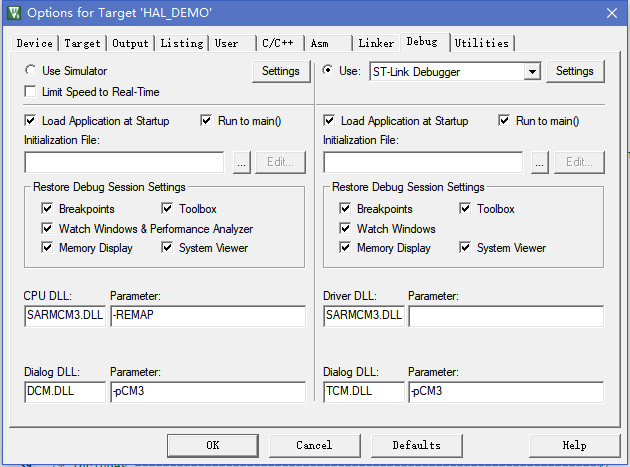
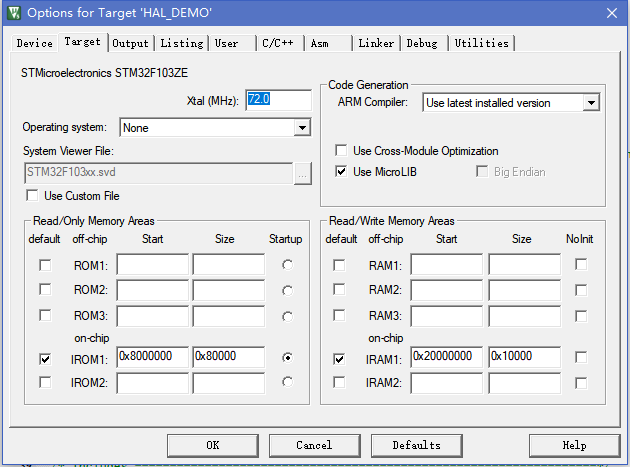


**最后，生成代码前要修改项目设置，确保生成期望的文件以及期望的位置。**

# 二、关于keil



**主界面如上，需要注意是下方的工程设置**



**基本的路径包含、配置，cube已经配置好了，需要修改的只是DEBUG以及下载方式。**

# 实例一

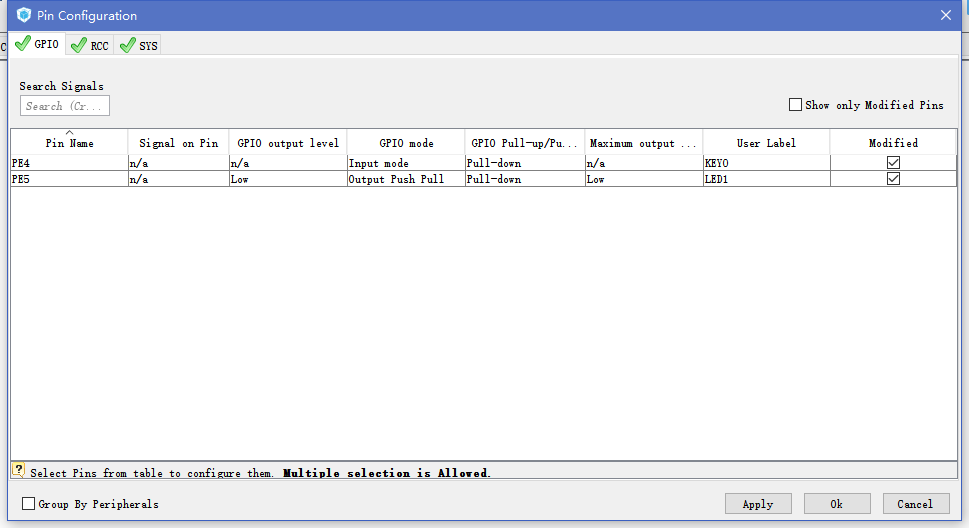
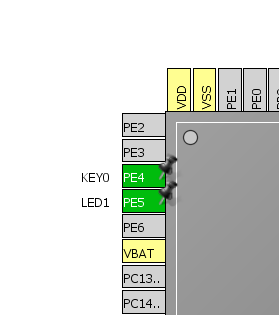
**目的：实现LED灯初始常亮，按键按下LED灯熄灭，松开LED恢复常亮。**

**参考：**

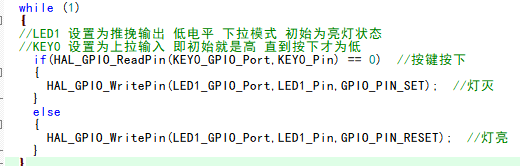
[**https://blog.csdn.net/jinpeng\_cumt/article/details/53019207**](https://blog.csdn.net/jinpeng_cumt/article/details/53019207) **（关于输入输出模式的介绍）**

**步骤：**

**1.cube设置**



**2.keil中只需在while（1）中写如下即可**



**知识点：**

**1.输出模式：推挽和开漏 只有开漏要外接上拉电阻**

**2.输入模式：上拉 下拉 浮空**

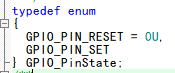
**3.HAL库有关gpio的相关函数**

**void HAL\_GPIO\_Init(GPIO\_TypeDef \*GPIOx, GPIO\_InitTypeDef \*GPIO\_Init)**

**GPIO\_PinState HAL\_GPIO\_ReadPin(GPIO\_TypeDef \*GPIOx, uint16\_t GPIO\_Pin)**

**void HAL\_GPIO\_TogglePin(GPIO\_TypeDef \*GPIOx, uint16\_t GPIO\_Pin)**

**void HAL\_GPIO\_WritePin(GPIO\_TypeDef \*GPIOx, uint16\_t GPIO\_Pin, GPIO\_PinState PinState)**



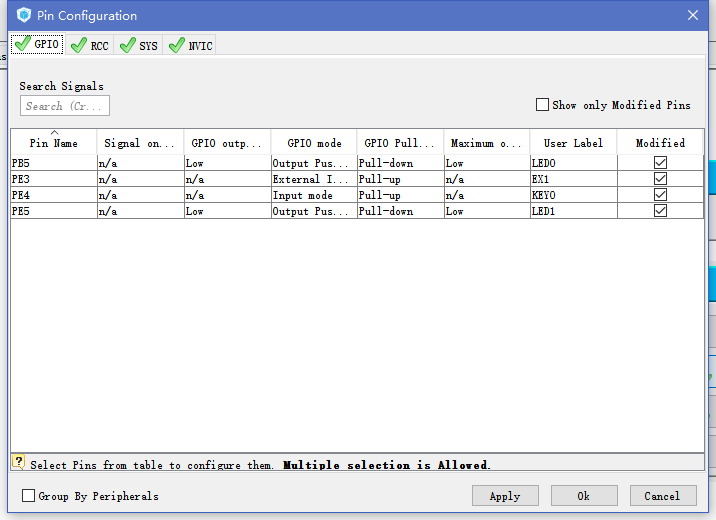
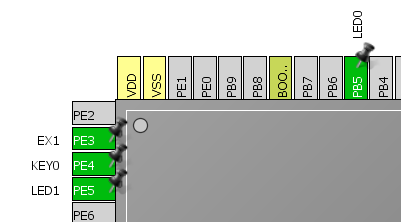
# 实例二

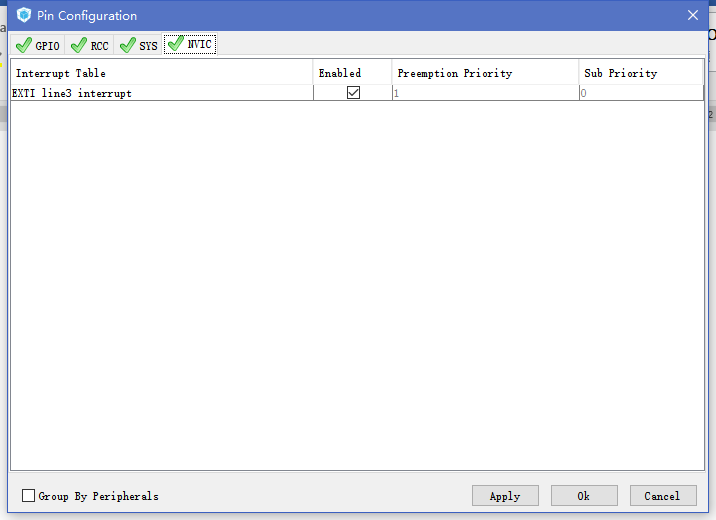
**目的：**

**实现每按键一次触发外部中断控制led灯翻转状态 PE3 - KEY1 PB5 - LED0**

**步骤：**

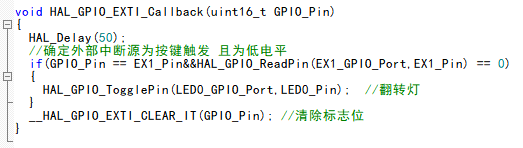
**1. cube操作：**





**2.keil程序：**

**main文件中无须添加程序，只需要在gpio.c文件中添加外部中断处理函数的实现。**



**知识点：**

1. **外部中断的上升沿触发或下降沿触发最好检测中断源的同时检测该引脚的电平高低，可减少误触发的情况；**
2. **延时的引入和引脚电平的判断都是为了更为准确的判断中断是否真的触发**
3. **有关外部中断的函数**

**void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin)**

**void HAL\_GPIO\_EXTI\_IRQHandler(uint16\_t GPIO\_Pin)**

**第2个为外部中断处理函数，触发外部中断自动进入第二个函数，而第二个函数的实现包含了第一个函数同时清除了标志位，我们需要做的是在第一个函数中编写逻辑程序即可。**

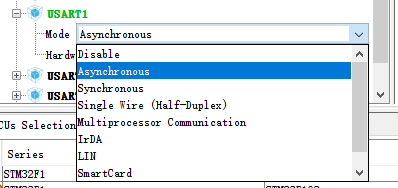
# 实例三

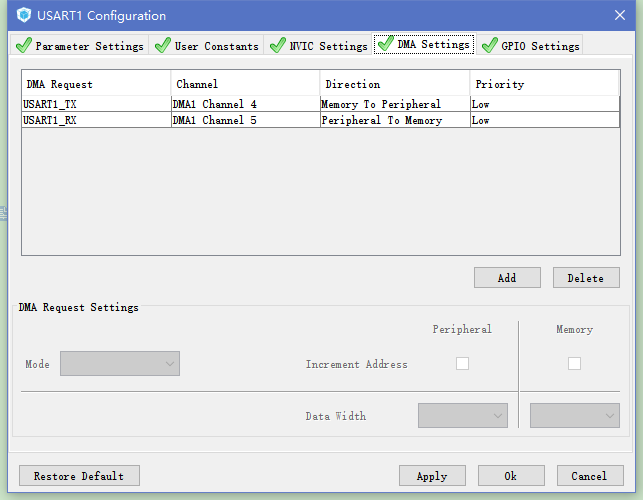
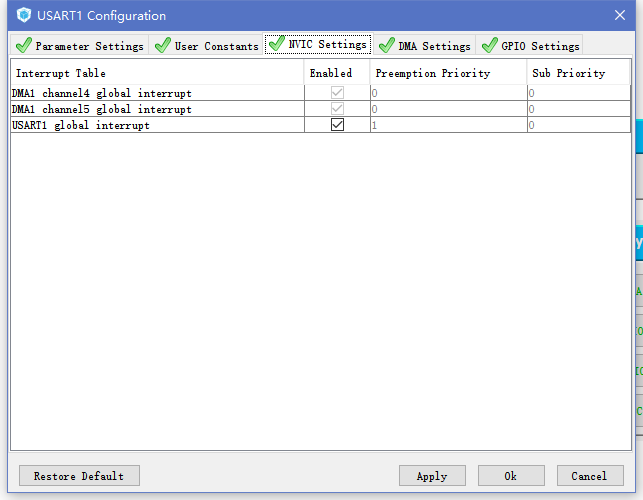
**目的：**

**使用USART1（PA9、PA10）以DMA的方式完成数据的接收与发送，达成效果，开机发送一段数据到PC端，然后发送5个数据，返回5个数据。**

**步骤：**

**1.cube设置**



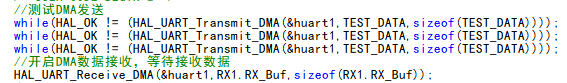


**如上图，开启usart引脚，开启串口中断、为RX TX添加DMA通道，记得连续接收数据要把其DMA模式改为轮询模式**

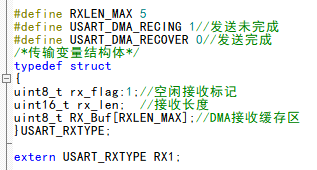
**2.keil程序编写**

**(1)Main文件**



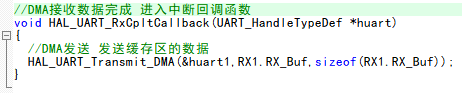


**(2)Usart.h文件**



**(3)Usart.c文件**





**知识点：**

1. **有关DMA的通道选择和中断优先级的设置均在cube中设置。**
2. **使用DMA进行串口发送时，用**

**HAL\_StatusTypeDef HAL\_UART\_Transmit\_DMA(UART\_HandleTypeDef \*huart, uint8\_t \*pData, uint16\_t Size)**

**函数可直接进行数据传输，但是要结合while(HAL\_OK != .....)来综合传输数据，当传输数据完成后，进入以下中断服务函数HAL\_UART\_TxCpltCallback**

1. **使用函数**

**HAL\_StatusTypeDef HAL\_UART\_Receive\_DMA(UART\_HandleTypeDef \*huart, uint8\_t \*pData, uint16\_t Size)**

**该函数启动接收状态，等待接收固定长度的字节存储到\*pData数组中。**

1. **当接收到期望长度数据时，进入中断服务函数HAL\_UART\_RxCpltCallback**
2. **DMA发送不可以连续重复发送，要么加延时要么加while(),否则会导致卡死**
3. **DMA接收的数据长度不满期望长度时，不会进行任何中断服务函数，所以需要上位机端自动补全至所需长度**

**当接收长度为1时，不能一次性发送所有数据，会导致数据中间丢失**

1. **最好为接收变量、发送变量设计一个结构体，包含标志位、缓存区等等，方便程序的编写**

# 实例四

**目的：**

**使用DMA的方式进行接收发送数据，使用空闲中断的方式进入中断处理函数，实现不定长数据的接收以及回显**

**步骤：**

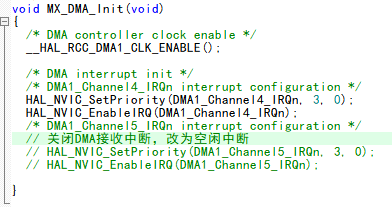
**1.cube设置**

**\* 注意DMA设置不可以设置为轮询的方式 使用普通模式即可**

**配置均与实例三一致**

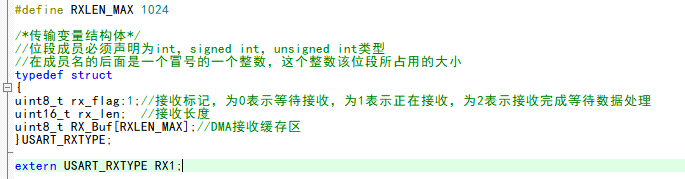
**2.Keil程序编写**

**(1) dma.c文件下**



**关闭DMA的接收中断，改用空闲的方式进入中断**

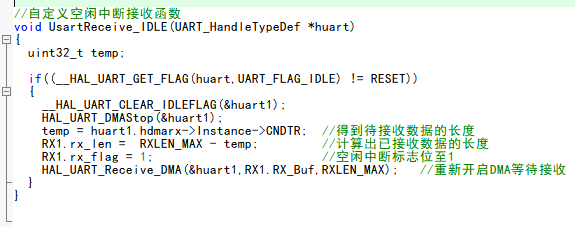
1. **usart.h文件下**



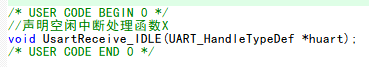
**声明传输变量结构体**

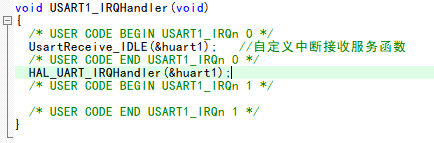
1. **usart.c文件下**





1. **stm32f1xx\_it.c文件下**





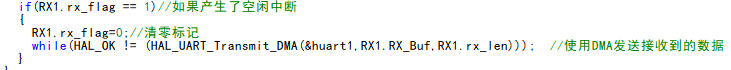
**在串口1的中断中加入自定义的服务中断，该中断由空闲中断自行触发**

1. **main文件下**

**while（1）函数前**



**while（1）函数里**

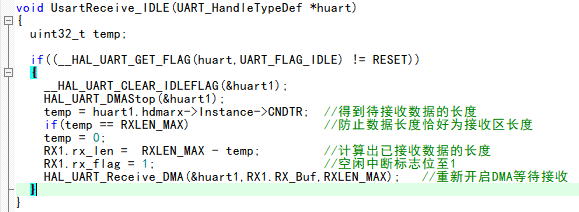


**知识点：  
1. 这个程序的逻辑为，关闭DMA的接收中断，改用空闲中断的方式触发，同时在中断处理函数中自行编写函数来处理：清除空闲中断标志位、更改接收标志位，重新打开DMA接收，在while(1)主函数中，根据标志位，将数据处理（此处为将数据发送回去）。**

**2. 可见DMA接收的方式进入中断次数少，比中断接收的方式效率高，所以接收常用DMA方式，而定常数据的接收可以使用原始的方式，但是不定长的数据可以使用空闲中断的方式。当然也可以将DMA的接收缓存区设置大些，这样方便数据的接收。**

**当数据超越接收区长度时，比如接收区长度为5，发送123456，会导致最后只有6返回，而当数据正好等于接收区长度时，会导致huart1.hdmarx->Instance->CNDTR为最大值，所以会导致计算出的rx\_len为0，导致无法正常计算数据长度。可采用以下方法：**

**方法一：**



**方法二：**

**增加接收缓存区的长度，设置得比预期的长度还长**

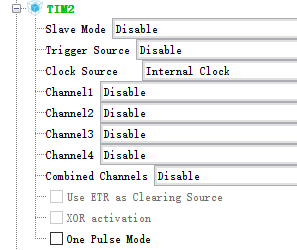
# 实例五

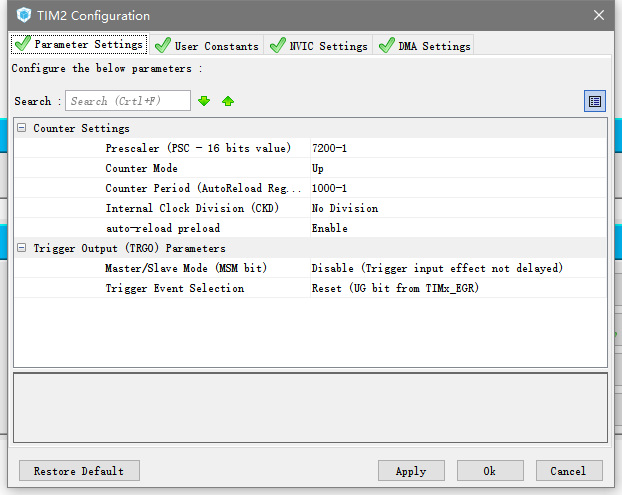
**目的：**

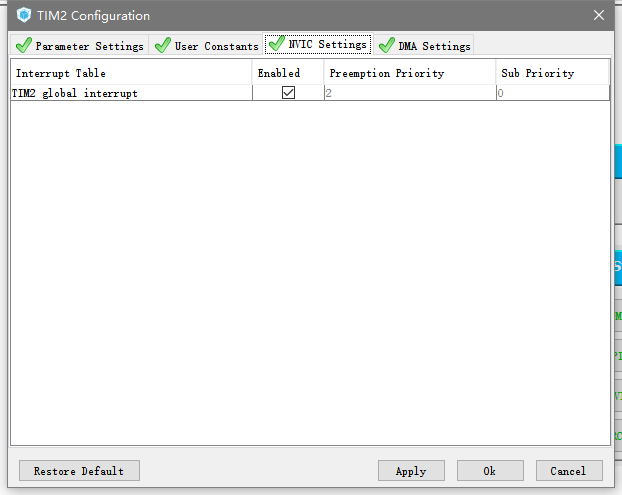
**使用定时器控制led灯按500ms的频率闪烁**

**步骤：**

1. **cube设置**







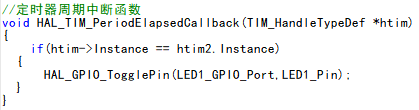
**注意：定时的时间计算是 1/时钟频率（72MHz） \* 预分频数（7200） \* 计数周期数（1000）为1s**

1. **keil程序编写**
2. **main文件中**

**while(1)之前**



1. **tim.c文件中**



**知识点：**

1. **有关延时 系统自带的可用hal\_delay()进行ms级别的延时，至于um级别的延时则需要使用滴答系统定时器、普通的定时器自行编写**
2. **注意开启中断的方式为HAL\_TIM\_Base\_Start\_IT();**