# 第15章 CubeMx软件使用

本章主要通过具体实例介绍CubeMx软件的使用，CubeMx可以自动生成底层硬件的配置代码，能够大幅提升编程效率，其前提是对STM32基本原理的理解，能够看懂基本配置代码，基本配置代码看不懂，遇到实际问题将寸步难行。标准库与HAL库95%左右代码相通，Keil中的操作也是通用的，熟悉标准库编程后，只需几天时间，就可以使用HAL库上手做项目。

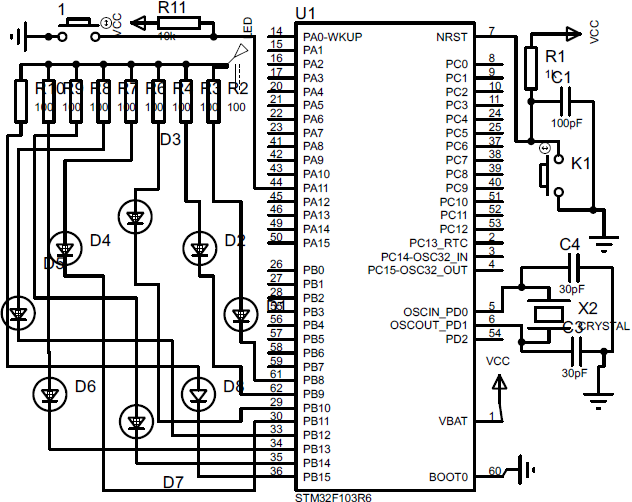
## 15.1实现功能要求

用LED灯组成环形圆圈，LED灯沿圆圈跑动，当按下按键之后，产生外部中断，所有LED灯闪烁三次，LED灯沿圆圈继续跑动。

采用CubeMX配置片内外设参数，自动生成驱动代码，使用HAL库编程实现以上功能。

## 15.2 硬件连接原理图

硬件连接电路图如图15-1所示，外部中断连接到端口PA11引脚，8个LED灯分别接到PB8~PB15上，需要配置GPIO端口PB8~PB15为输出模式，PA11为外部中断。

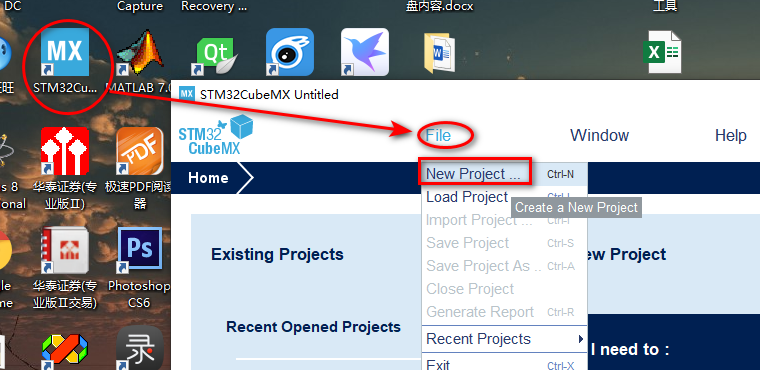


###### 图15-1 硬件电路图

## 15.3 CubeMX配置

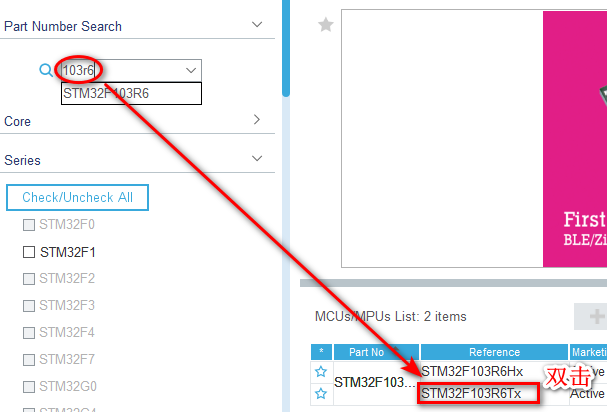
### 15.3.1 CubeMX新建工程及STM32选型

启动CubeMX软件，新建工程，如图15-2所示。



###### 图15-2 新建工程

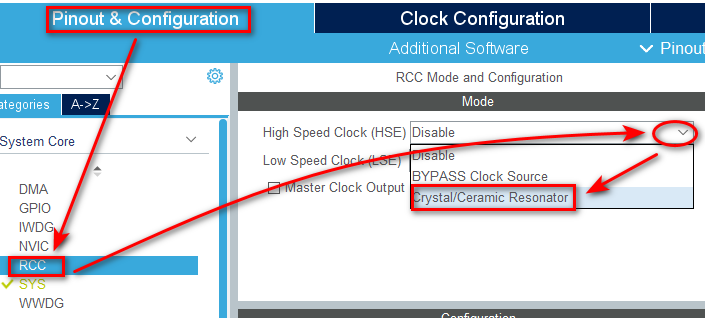
根据图15-1的STM32处理器，输入对应型号的关键字找到对应的芯片，如图15-3所示。第一次选择芯片后，会安装相关驱动，需要等待几分钟。选择对应的型号后双击即可。



###### 图15-3 选择芯片型号

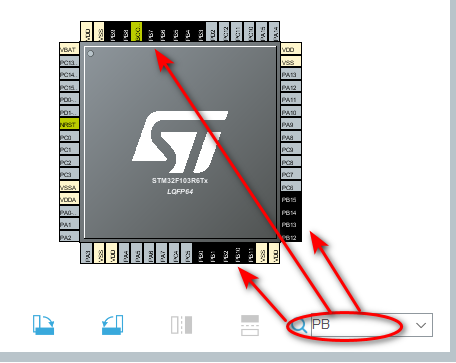
### 15.3.2 时钟源及端口配置

进入配置界面后，先配置时钟，如图15-4所示，此处选择外部晶振作为高速时钟。



###### 图15-4 选择晶振作为高速时钟

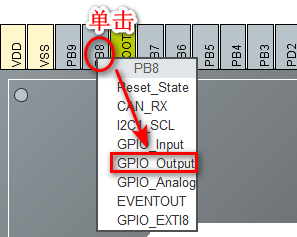
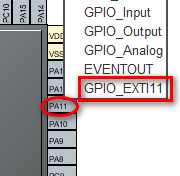
继续配置引脚，如图15-5所示，根据图15-1所用到的引脚PB8~15，右下角输入“PB”快速查找引脚。上滚鼠标滚轮，可以放大引脚图，左键单击拖拽鼠标可以移动引脚图。



###### 图15-5 快速查找引脚

通过放大、移动找到PB8引脚后，单击并选择输出模式，如图15-6所示。用同样的方法，将PB9~15也设置成输出模式。将PA11设置为中断模式，如图15-7所示。快捷菜单中 “Reset\_State”命令，表示取消设置。

继续进一步配置GPIO端口，如图15-8所示，PB8默认选择高电平，LED灯灭，单击左侧“GPIO”后，会回到端口列表，对其余PB9~15端口采用相同的设置。

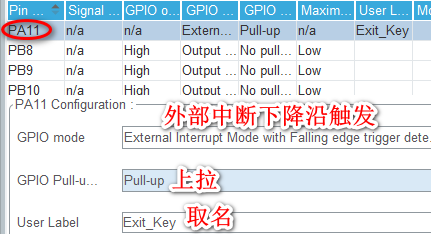
 

###### 图15-6 输出模式配置 图15-7 外部中断模式配置



###### 图15-8 输出配置

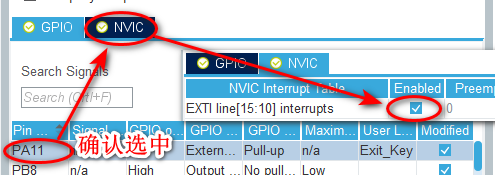
用类似的方法对PA11进行外部中断配置，默认上拉即高电平。



###### 图15-9 外部中断配置

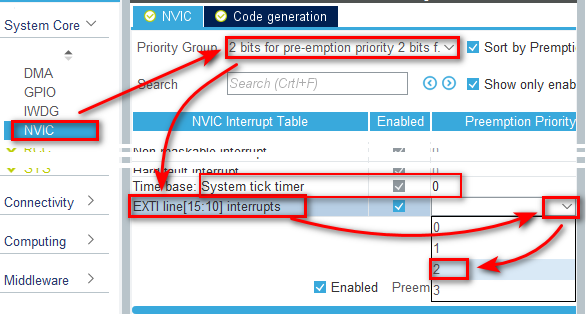
### 15.3.3 CubeMX中断配置

中断配置如图15-10所示，确认选中外部中断引脚后，再使能外部中断。



###### 图15-10 外部中断使能

进一步配置中断分组，抢占优先级，如图15-11所示，注意外部中断的抢占优先级要设置得比较低，此处设置为“2”。

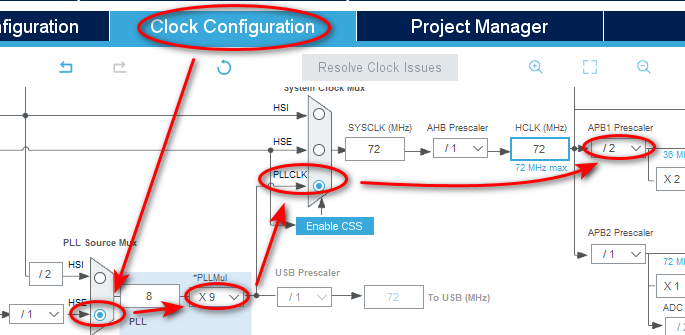


###### 图15-11 中断优先级配置

注意，图15-11中，在“EXTI line[15:10] interrrupts”的上方，“Time base: System tick timer”即systick默认抢占优先级为“0”。

### 15.3.4 CubeMX时钟树配置

在CubeMX中配置时钟树时，由于采用了可视化界面，时钟树上各路片内外设时钟一目了然，配置参数如图15-12所示。注意：能够选择“HSE”按钮的前提是要先将外部晶振作为时钟源（如图15-4所示的操作）。

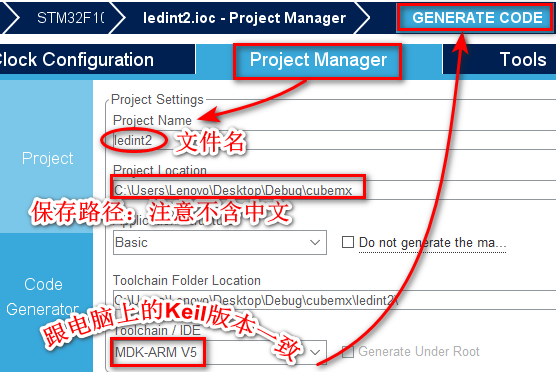


###### 图15-12 时钟树配置

按照图15-12所示配置完时钟后，重点注意APB1和APB2的时钟频率，大部分片内外设挂靠在该时钟分支。

### 15.3.5 工程输出配置

在生成代码之前需要进行文件名、输出目录、生成类型等配置，如图15-13所示，配置完相关输出设置后选择“GENERATE CODE”生成代码。



###### 图15-13 工程输出配置

等待一会儿，会成功生成代码，可以打开工程所在的文件夹，也可以点击打开生成的工程，如图15-14所示。



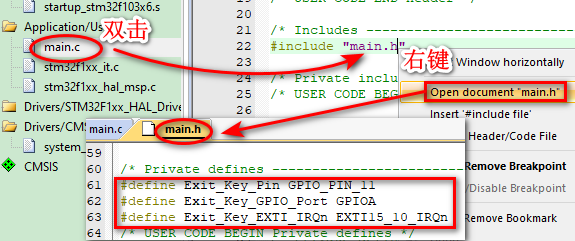
###### 图15-14 打开文件夹或工程

CubeMX配置的参数有很多，其余参数采用默认设置即可。配置操作一般有多种方法，本例程只使用了比较通用的一种方法，读者在学习中也可以尝试其他操作方法。

## 15.4 CubeMX生成的工程简介

CubeMX自动生成的工程跟标准库所建工程基本上区别不大，主要有如下几点。

**（1）自动定义常量**： 进入main.c文件，如图15-15所示，打开main.h文件，查看CubeMX自动生成的常量定义，即图15-9所示的引脚名“Exit\_Key”。

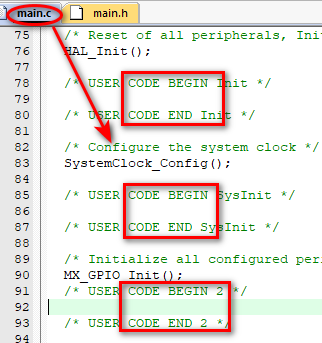
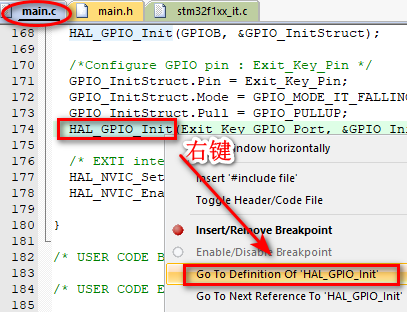


###### 图15-15 查看工程main.c代码

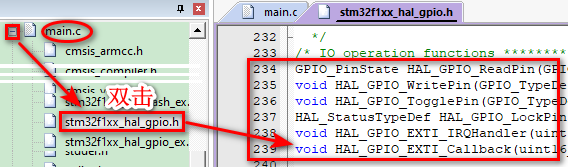
**（2）用户代码区**：注意main.c中很多注释 “USER CODE BEGIN…”及“USER CODE END…”，如图15-16所示，这些注释成对出现，为用户代码编写区。当用户配置CubeMX出现错误，需要重新生成代码后， “…BEGIN…”和“…END…”之间的代码不会改变，其他代码会改变或者被删除。

**（3）HAL库函数查看**：对工程进行编译连接，发现没有任何问题“0 Error(s), 0 Warning(s).”。CubeMX已经自动生成了系统时钟配置“SystemClock\_Config(void)”和端口初始化“MX\_GPIO\_Init(void)”函数代码，进入端口初始化代码查看注释说明，如图15-17所示。其操作过程与标准库方法一致，配置参数的代码与标准库也基本一致，当然HAL函数库里所有函数前面多了关键字“HAL\_”。

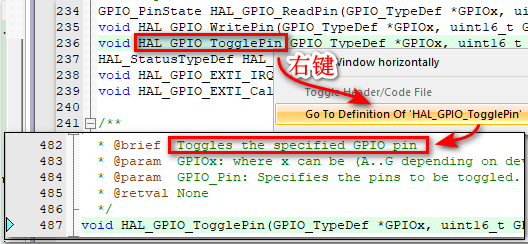
查看HAL库提供的GPIO相关函数，如图15-18所示，可以右键查看其注释，了解其功能。例如，查看“HAL\_GPIO\_TogglePin( )”函数的功能，如图15-19所示，该函数表示对引脚取反。若英文不熟悉的话，可以通过百度搜索一下，或者通过翻译软件翻译一下。英文注释每次遇到了需要读懂，通过学习几个实例之后，就很容易看懂。在实际的工程项目中，会用到其他公司的芯片，相关的技术资料会非常少，可能只有英文资料及代码实例。

###### 图15-16 用户代码编写区 图15-17查看端口初始化



###### 图15-18 查看HAL库的gpio函数

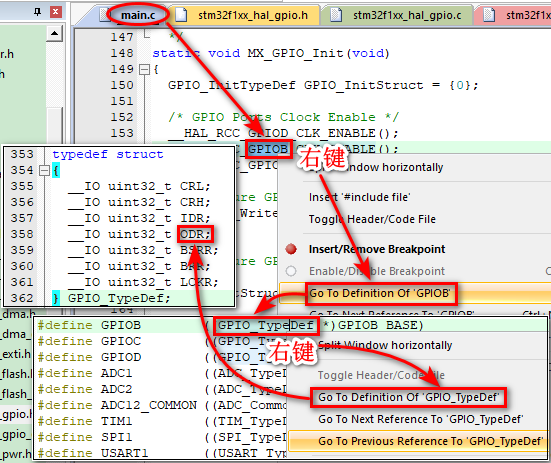


###### 图15-19 查看“HAL\_GPIO\_TogglePin( )”函数

**（4）GPIO端口整体赋值**：从图15-18中可以看出，HAL库中没有对GPIO端口的16个引脚整体赋值的函数，可以通过对GPIO的数据寄存器（ODR）进行赋值，追踪数据寄存器（ODR）的方法如图15-20所示。先选定“GPIOB”（选GPIOA、GPIOC或GPIOD均可），右键进入其定义，再选定“GPIO\_TypeDef”，再次右键进入其定义，可以查到GPIO的寄存器ODR。例如，对ODR寄存器的高8位赋值“1”，低8位赋值“0”代码如下：

GPIOB->ODR =0xff00; //0x表示十六进制

端口B的状态：PB15~ PB 8全部为“1”，PB7~ PB0全部为“0”。



###### 图15-20 追踪GPIO的数据寄存器

（5）中断：HAL库中断只需要重新定义回调函数并编程即可，大大简化了编程方式。追踪到回调函数如图15-21所示，回调函数定义如下：

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

回调函数的前面加上了\_\_weak 修饰符，用户可以在用户文件中重新定义一个同名函数，最终编译器编译的时候，会选择用户定义的函数，如果用户没有重新定义这个函数，那么编译器就会执行\_\_weak 声明的函数，并且编译器不会报错。

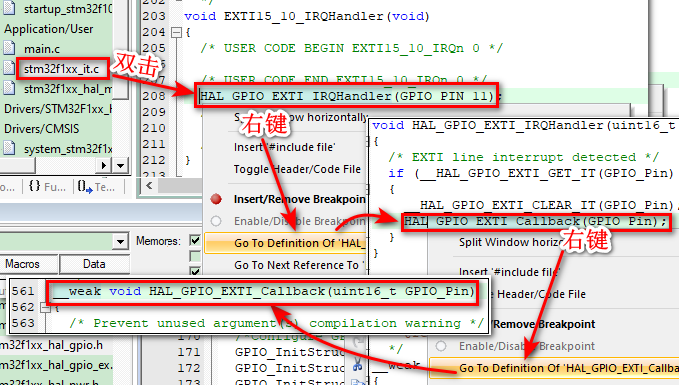
从追踪回调函数的过程可以看到，只要中断发生，就会调用回调函数，用户只需要复制该回调函数的头部（“\_\_weak”不复制），重新定义该函数即可。在定时器、串口等中断程序的编写中，均采用相同的方法。

## 15.5代码实现

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

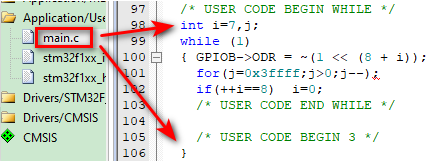
\* 实现功能：用LED灯组成环形圆圈，LED灯沿圆圈跑动，当按下按键之后，产生外部中断，所有LED灯闪烁三次，LED灯沿圆圈继续跑动。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/



###### 图15-21 追踪中断回调函数

在main（）函数中编写的代码如图15-22所示，实现LED灯循环点亮，每点亮一个灯延时一会儿，可以根据实验现象适当调整延时时间。



###### 图15-22 main( )函数代码

重新定义中断回调函数如图15-23所示，实现每按一次按键，所有LED灯自动闪烁3次。中断回调函数编程时，需要注意以下几点说明：

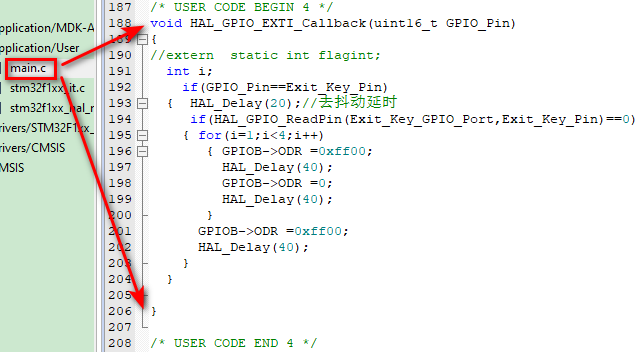
（1）延时函数：HAL\_Delay( )是毫秒级延时，不需要用户自己编写，直接调用即可，延时函数是通过systick中断改变其延时。由于systick中断抢占优先级为0（如图15-11所示），外部中断抢占优先级若设为0，则“HAL\_Delay(20)”延时将无法正确执行。

（2）回调函数中按键的判断：由于所有外部中断都调用同一个回调函数，当涉及多个外部中断时，回调函数编程时，需要读取引脚的状态，判断某按键是否按下，本例程判断语句如下：

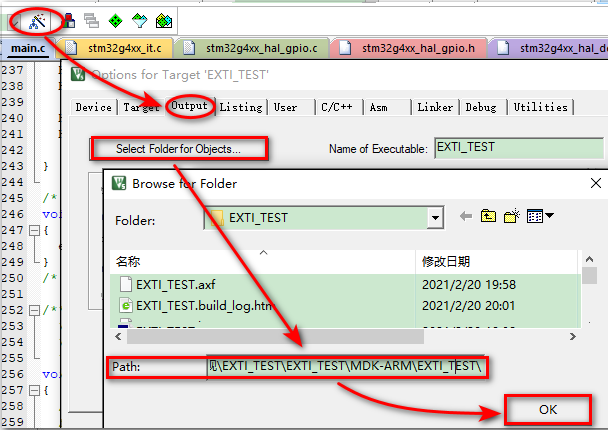
if(HAL\_GPIO\_ReadPin(Exit\_Key\_GPIO\_Port,Exit\_Key\_Pin)==0)

## 15.6程序调试

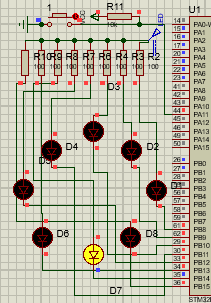
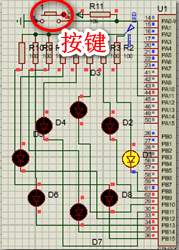
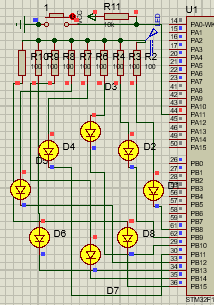
Keil中其他设置均采用默认值，直接编译链接生成hex文件，加载hex文件（文件位置如图15-24所示）到Proteus中观察实验现象，如图15-25所示。当我们按下按键后，会看到所有LED灯闪烁3次。



###### 图15-23 中断回调函数的编写



###### 图15-24 “hex”文档路径

###### 图15-25 Proteus仿真现象

采用CubeMX自动生成代码，编程效率会大幅提升，即使更换芯片型号，其他端口不变的话，只需在CubeMX中设置一下，重新生成代码后，应用程序的代码都不用修改。本例程只介绍了一些常用的配置操作，还有一些其他细节配置，遇到问题时，可以在网上随时查找相关技术资料。

### 习题

15-1 采用CubeMX自动生成代码，实现第9章定时器例程的功能。

15-2 采用CubeMX自动生成代码，实现第10章PWM调控的例程功能。

15-3 采用CubeMX自动生成代码，实现第11章串口的例程功能。