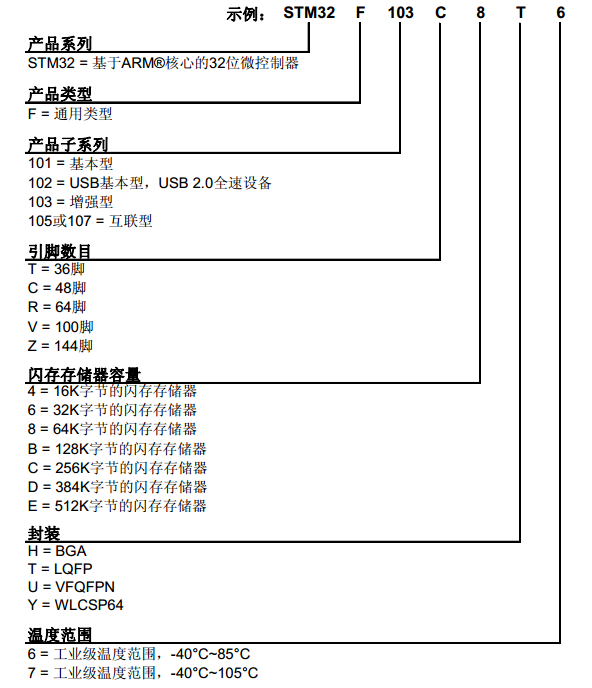
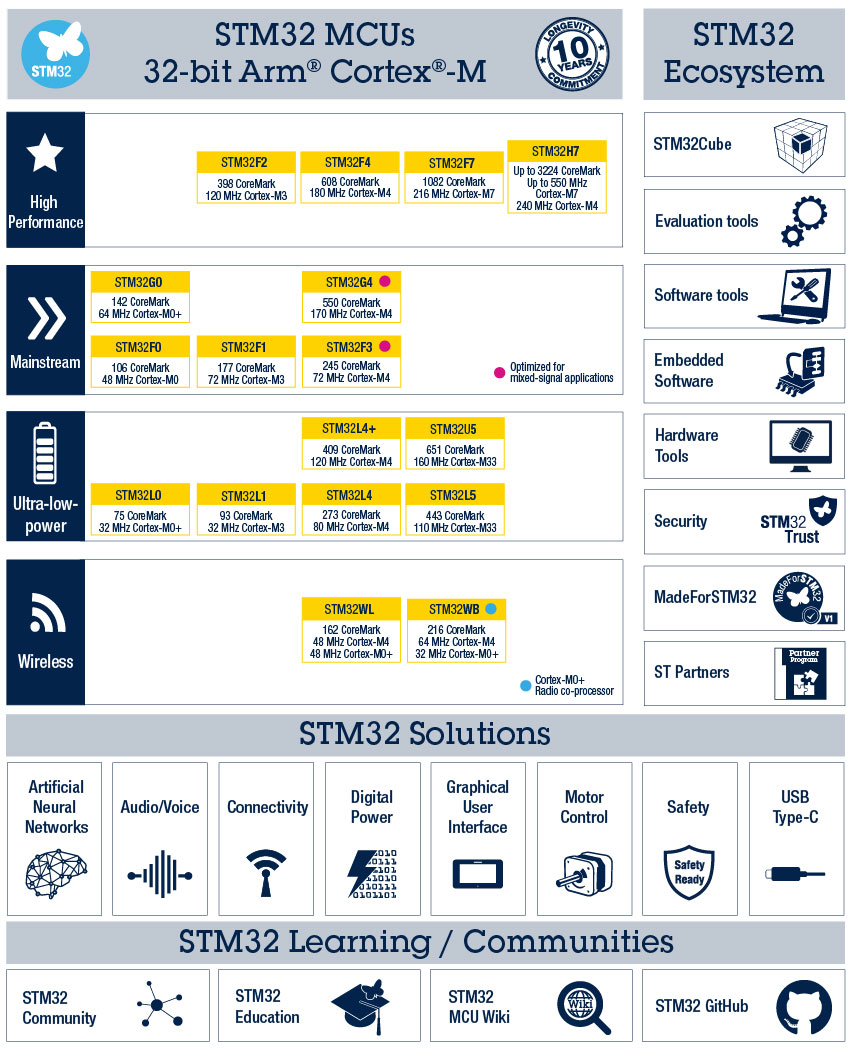
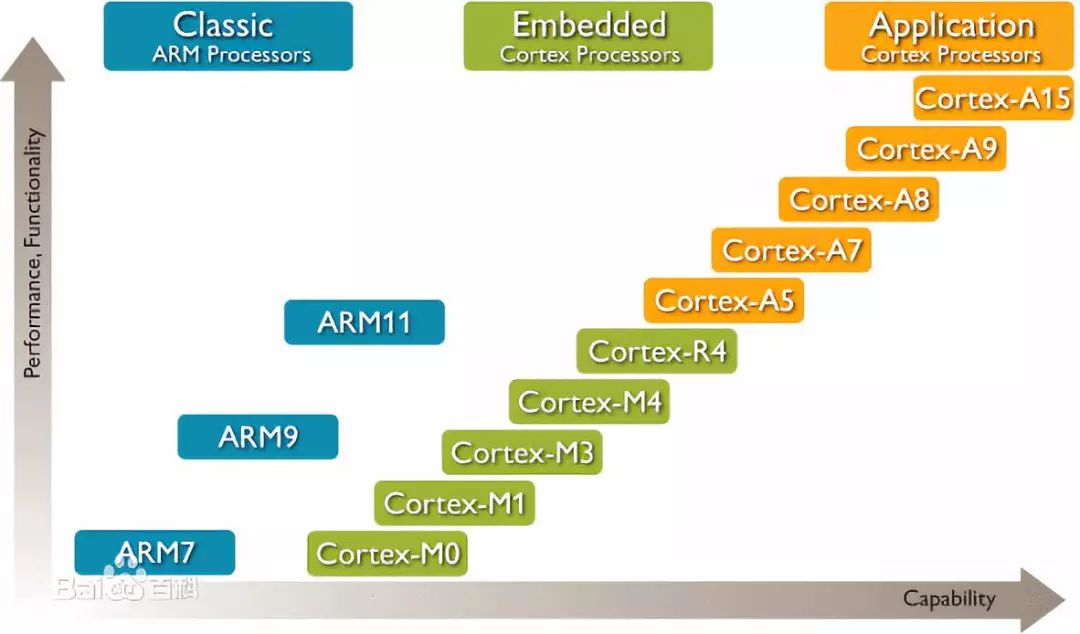
# 1. STM32简介

## 1.1 STM32命名规范







ARM既指ARM公司，也指ARM处理器内核。

A：Application，主要偏向于应用

R：Realtime，偏向于实时性

M：Microcontroller，微控制器

## 1.2 STM32F103C8T6特性

系列：主流系列STM32F1

内核：ARM Cortex-M3

主频：72MHz

RAM：20K（SRAM）

ROM：64K（Flash）

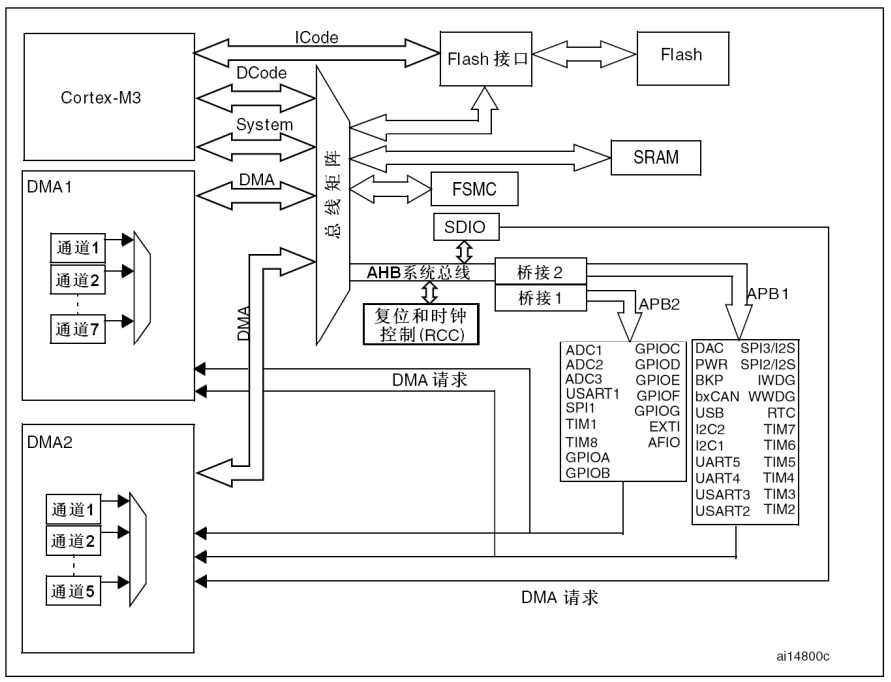
供电：2.0~3.6V（标准3.3V）

封装：LQFP48

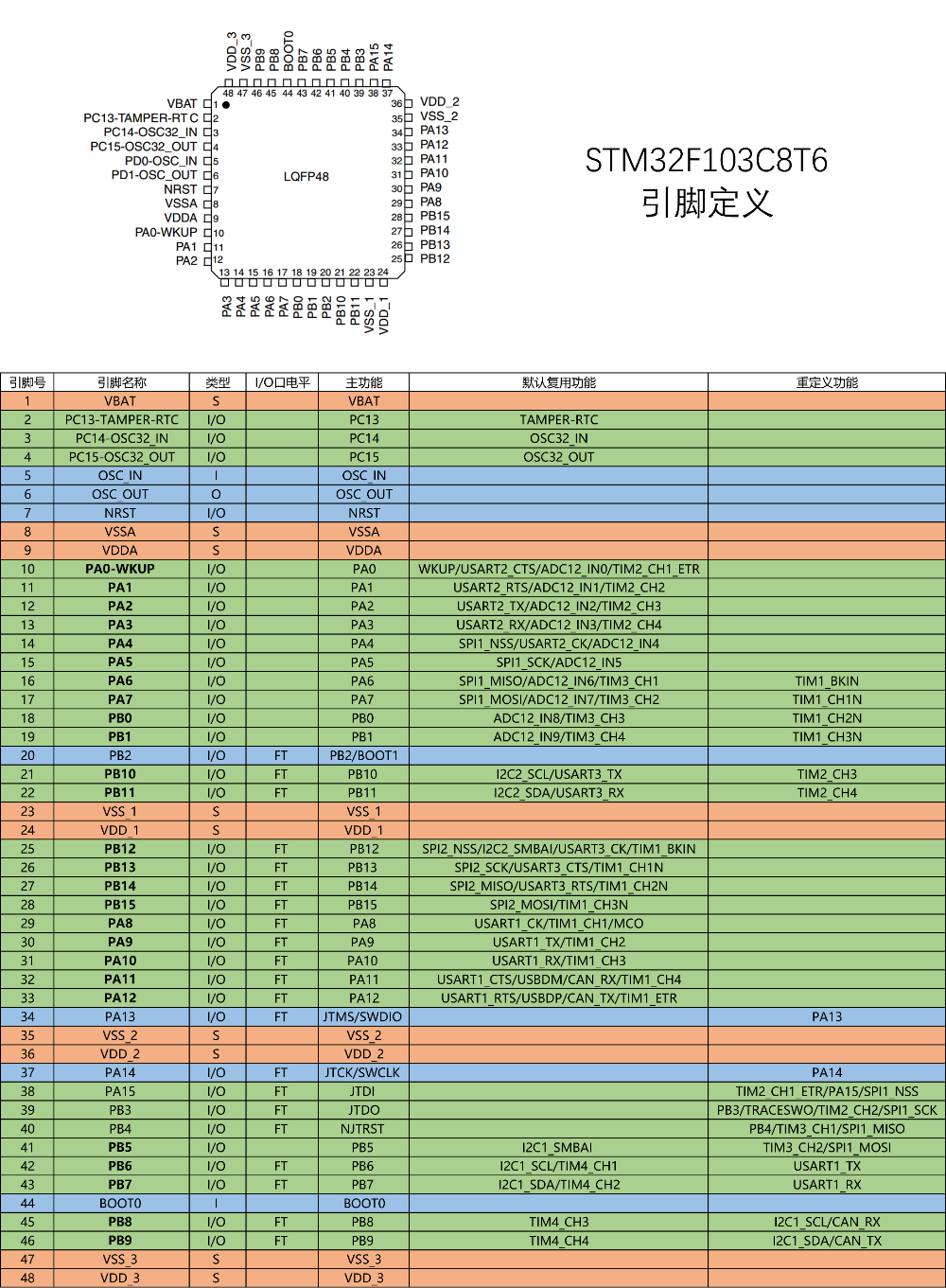
## 1.3 片上资源/外设



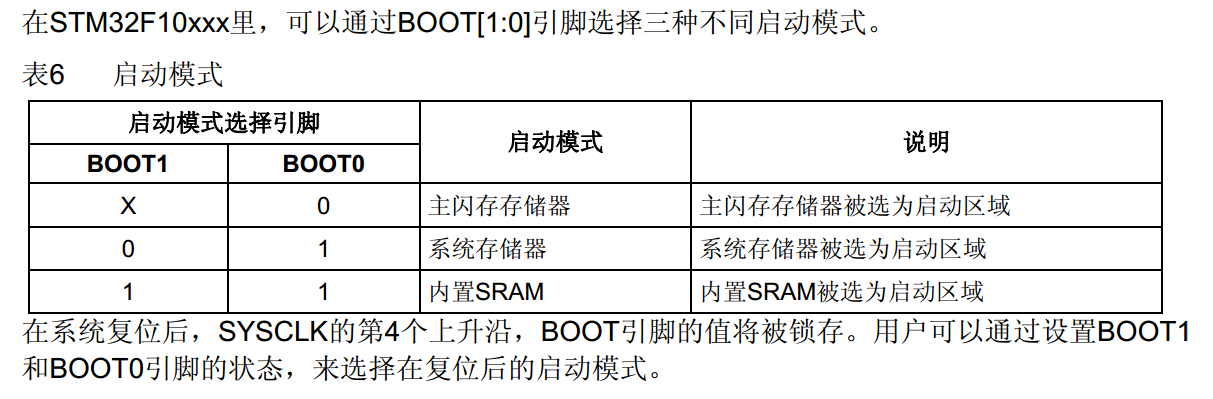
## 1.4 系统架构



## 1.5 引脚定义



## 1.6 启动配置

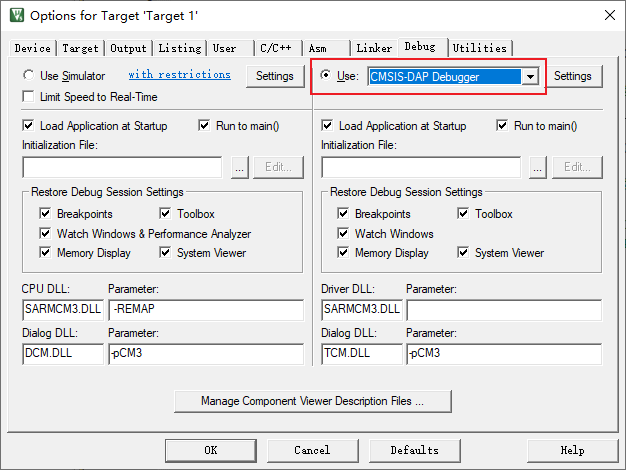


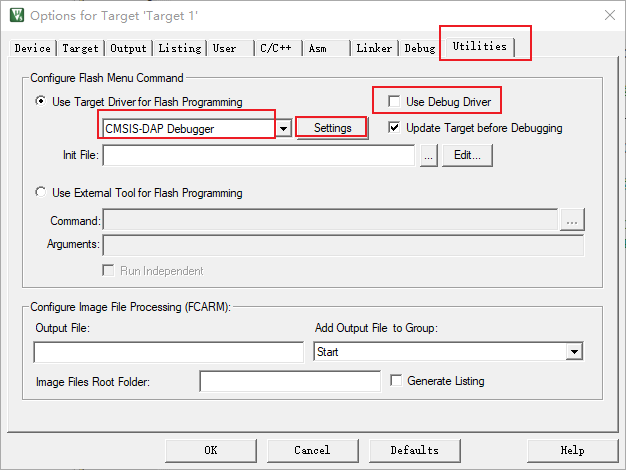
(1)最常用配置：BOOT0设为0，正常的执行Flash闪存里面的程序，使用STLINK、DAPlink进行下载调试

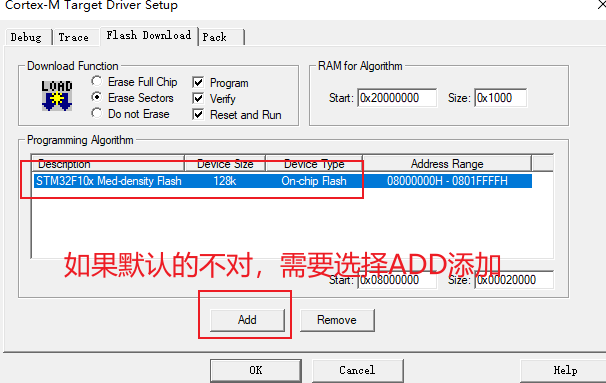
(2)串口下载模式：BOOT0设为1，BOOT1设为0，系统存储器(存放Bootloader程序，接收串口的数据，然后刷新到主闪存中)

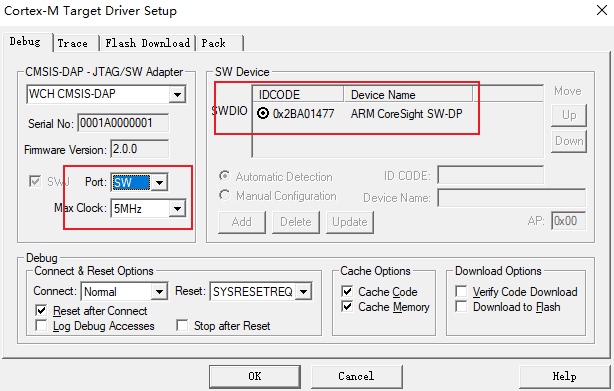
(3)最后一种模式，主要用于程序调试，用的比较少

## 1.7 DAPLink烧录

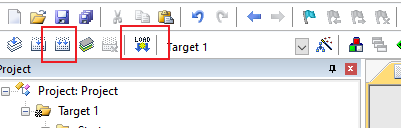








每次下载程序，需要先rebuild编译，在下载！



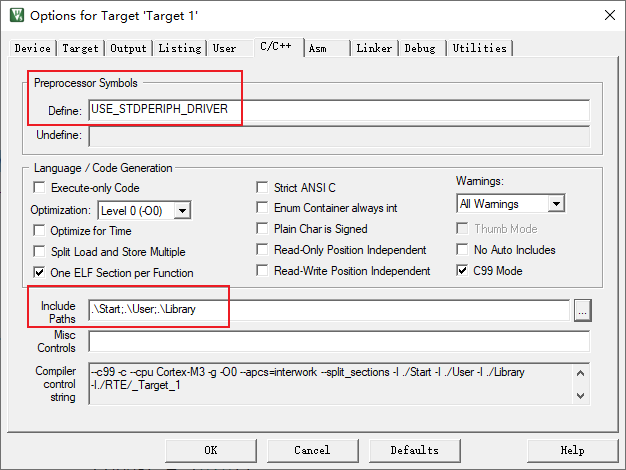
# 2. STM32工程模板

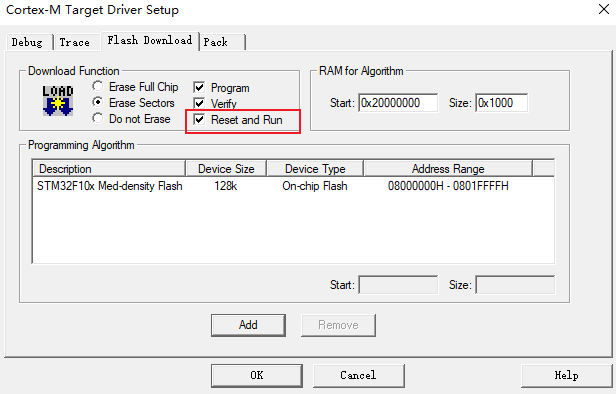
## 2.1 Keil配置工作模板

使用ST库函数的开发方式进行寄存器配置。

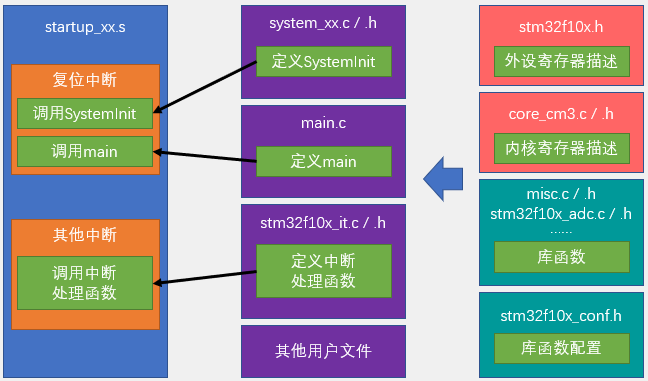
新建工程步骤：

* 建立工程文件夹，Keil中新建工程，选择型号
* 工程文件夹里建立Start、Library、User等文件夹，复制固件库里面的文件到工程文件夹
* 工程里对应建立Start、Library、User等同名称的分组，然后将文件夹内的文件添加到工程分组里
* 工程选项，C/C++，Include Paths内声明所有包含头文件的文件夹
* 工程选项，C/C++，Define内定义USE\_STDPERIPH\_DRIVER (标准外设驱动字符串，使用库函数开发时，这个是必须添加的)
* 工程选项，Debug，下拉列表选择对应调试器，Settings，Flash Download里勾选Reset and Run

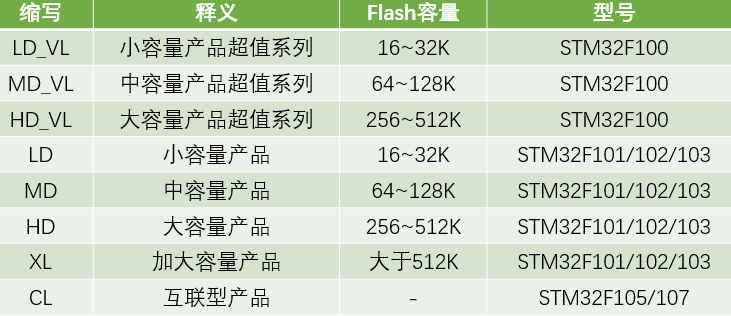


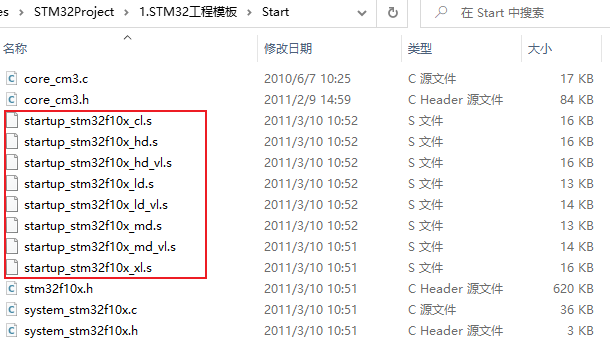


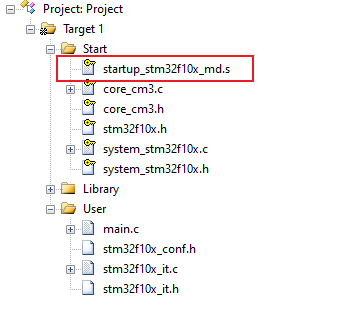
工程架构：

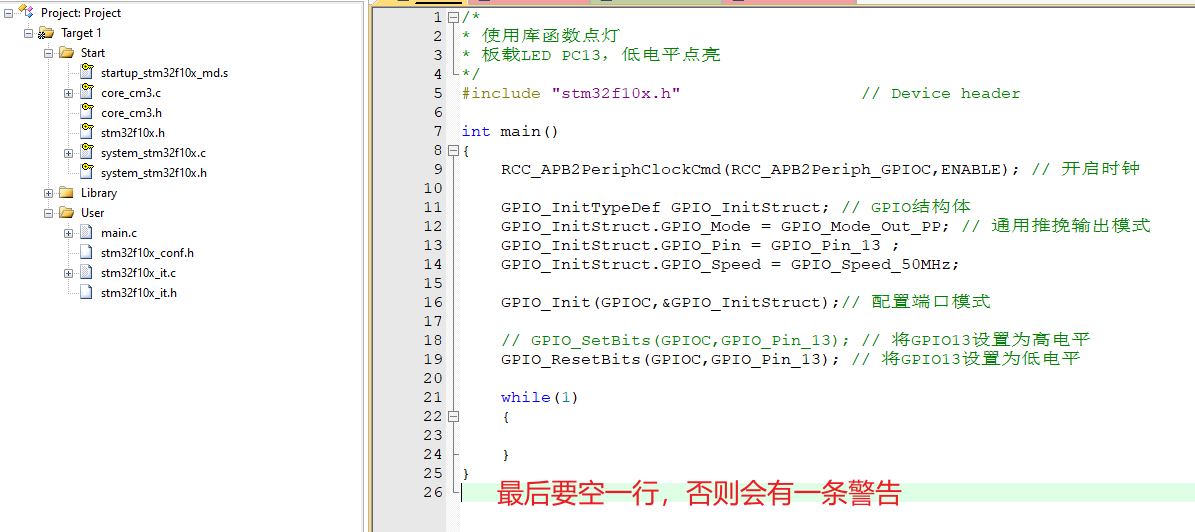


根据不同的产品型号，选择不同的文件



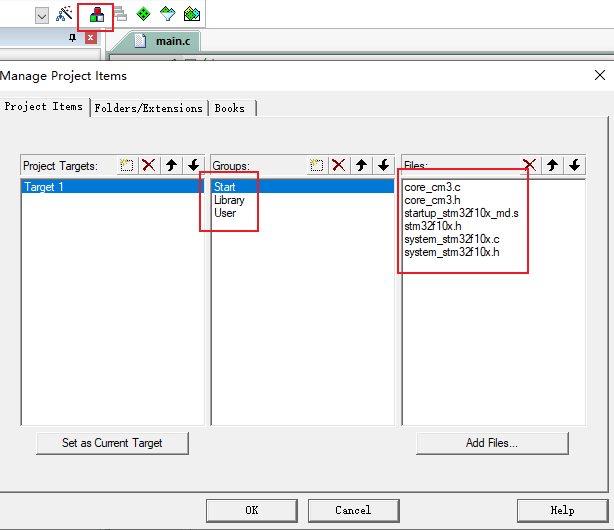




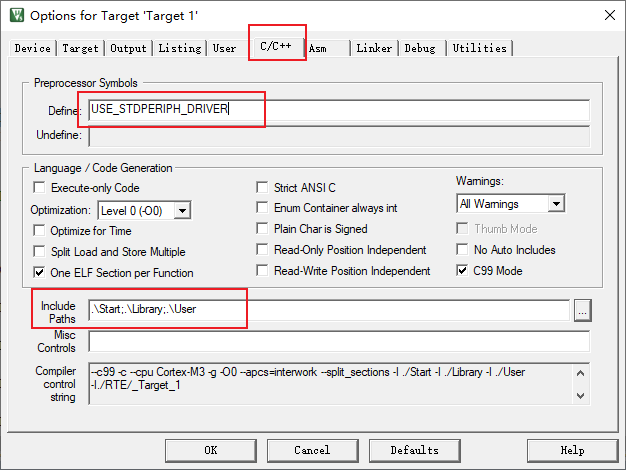


## 2.2 创建工程步骤总结

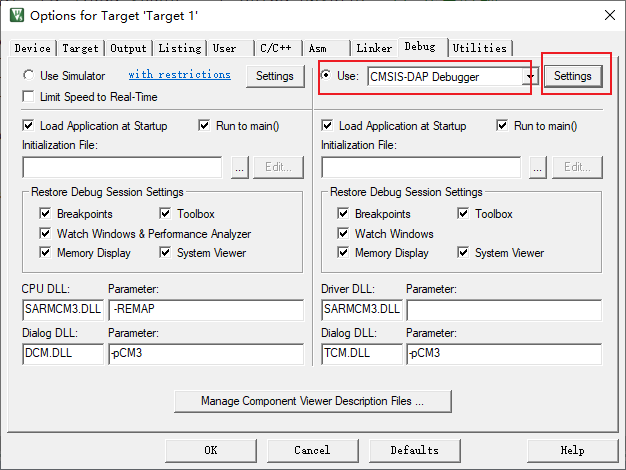
1. 快速创建Start、Library、User组，添加相关的文件

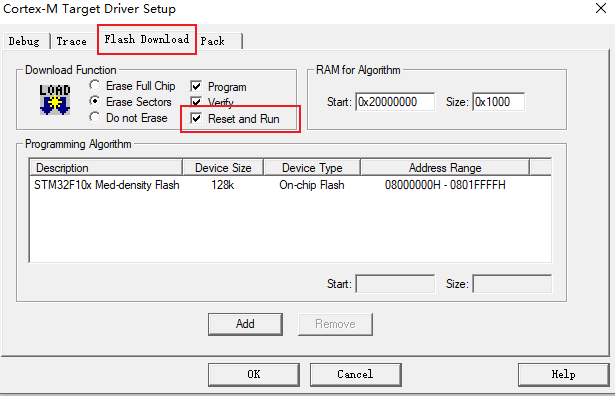


2.点击魔术棒按钮，添加第一步文件的路径，并添加这句话USE\_STDPERIPH\_DRIVER



3. 选择对应的调试器(DAPLink)

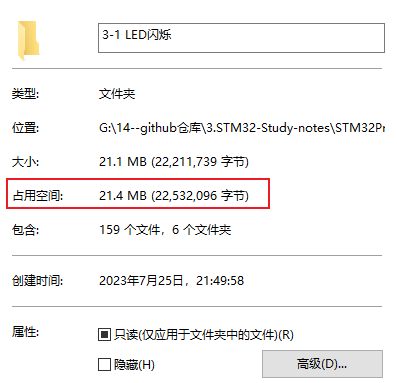
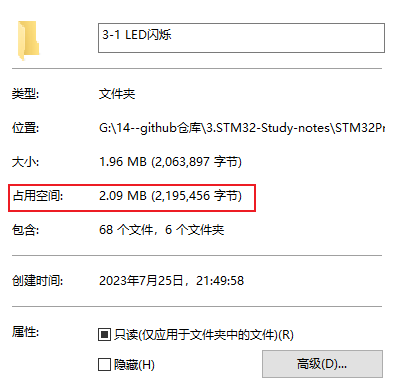




4.小工具：keilkill，是一个批处理文件，可以把工程编译产生的中间文件都删掉



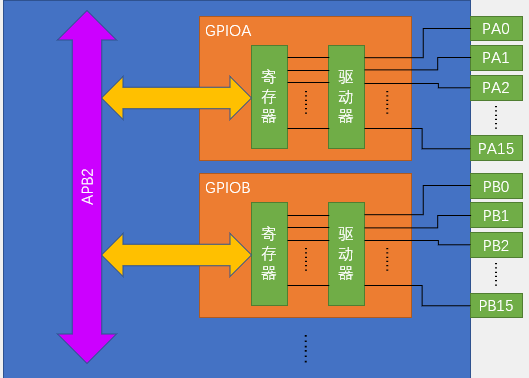
使用前后的对比：减小工程体积，之后可以添加到压缩包分享发送给别人

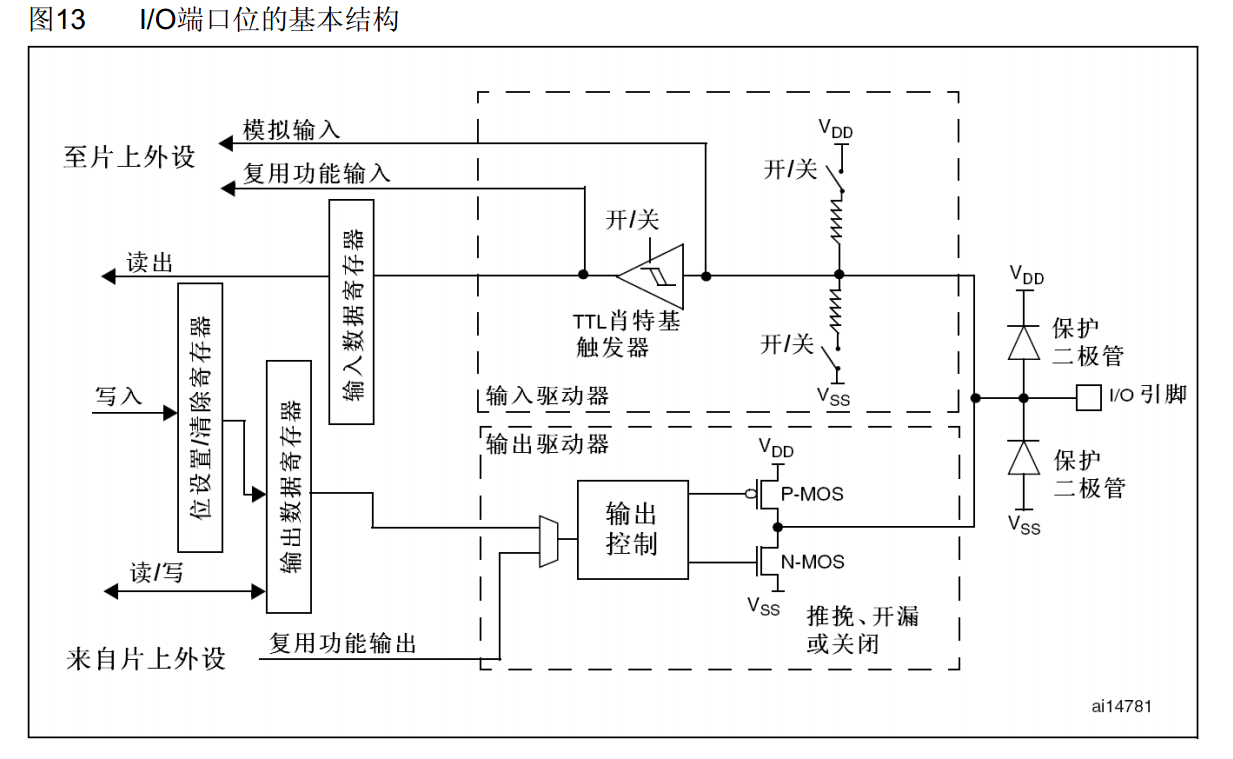
# 3. GPIO配置及应用

## 3.1 GPIO工作模式介绍

GPIO基本结构：

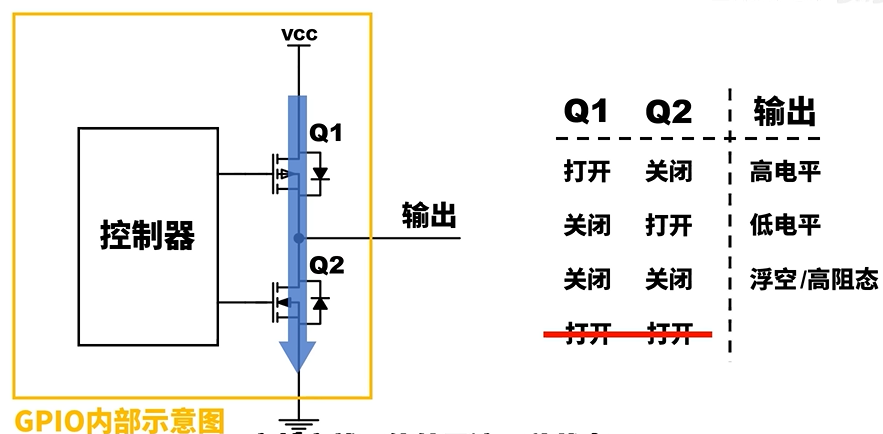


GPIO位结构：





单片机的GPIO配置为输出模式时，会有两种模式：推挽模式、开漏模式。



上下MOS同时导通时，发生短路，这种状态需要避免。

推完输出模式理解：GPIO输出高低电平时，电流的一个动作而已

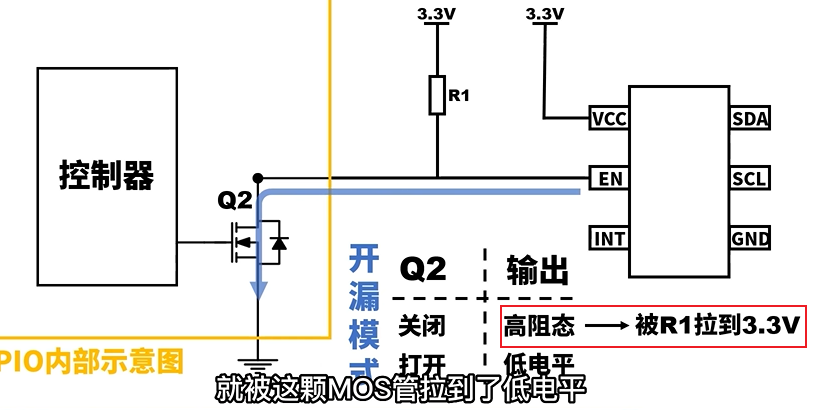
推：IO输出高电平时，把电流推出去

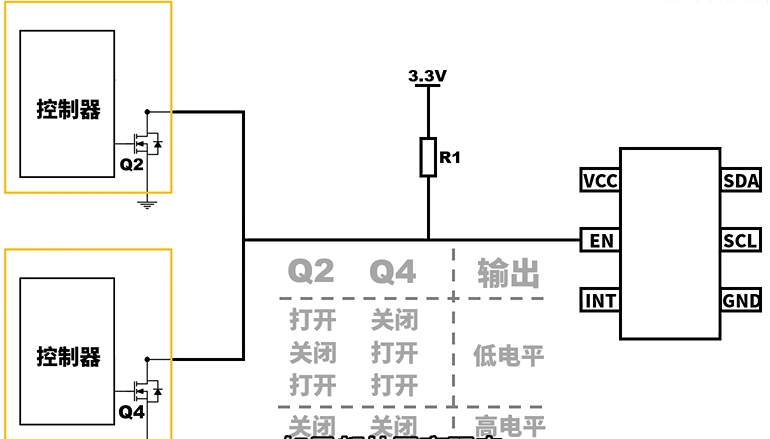
挽：IO输出低电平时，外界电流流进来，把电流挽回来。

开漏模式：低电平和高阻态(被外接上拉电阻拉到高电平)的组合，称为开漏输出。

需要在外面接一个上拉电阻，常用于几个GPIO控制一个输入。

开漏模式输出高电平相当于高阻态，没有驱动能力；低电平有驱动能力。





只要有任意一个GPIO输出低电平，enable就是低电平

如果都处于高阻态，enable就是高电平

## 3.2 LED闪烁



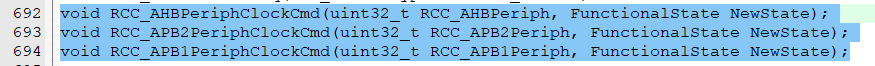
操作STM32的GPIO共需要3个步骤：

1.使用RCC开启GPIO的时钟

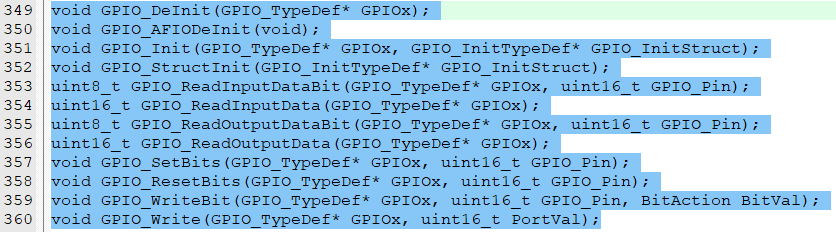
2.使用GPIO\_Init函数初始化GPIO

3.使用输出或者输入的函数控制GPIO口

1.RCC(Reset Clock Controller)，复位与时钟控制器，常用RCC AHB外设时钟控制、RCC APB2外设时钟控制、RCC APB1外设时钟控制

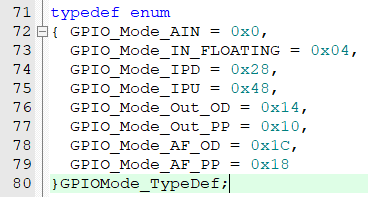


2.GPIO常用库函数



GPIO\_Init()函数：用结构体的参数来初始化GPIO口，需要先定义一个结构体变量，然后给结构体赋值，最后调用这个函数。

GPIO工作模式：



typedef enum

{ GPIO\_Mode\_AIN = 0x0, // Analog IN，模拟输入

GPIO\_Mode\_IN\_FLOATING = 0x04, // 浮空输入

GPIO\_Mode\_IPD = 0x28, // In Pull Down，下拉输入

GPIO\_Mode\_IPU = 0x48, // In Pull Up，上拉输入

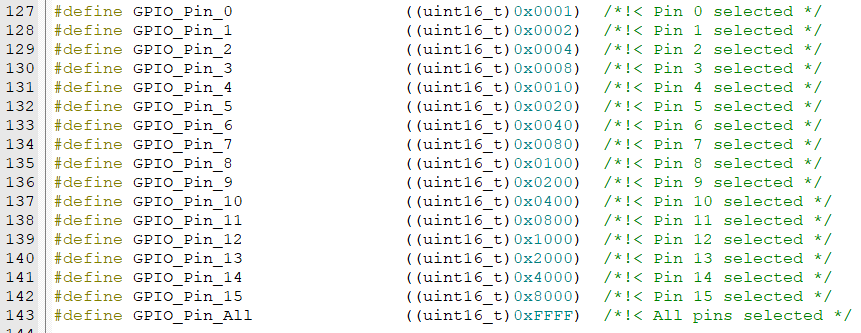
GPIO\_Mode\_Out\_OD = 0x14, // Out Open Drain，开漏输出

GPIO\_Mode\_Out\_PP = 0x10, // Out Push pull，推挽输出

GPIO\_Mode\_AF\_OD = 0x1C, // Atl Open Drain，复用开漏

GPIO\_Mode\_AF\_PP = 0x18 // Atl Push Pull，复用推挽

}GPIOMode\_TypeDef;



void GPIO\_SetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin); // 把指定的端口设置为高电平

void GPIO\_ResetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin); 设置GPIO输出低电平

# 4.OLED显示屏使用