STM32 MCU Development

STM32 单片机开发

-- STM32CubeIDE 开发环境介绍

目录

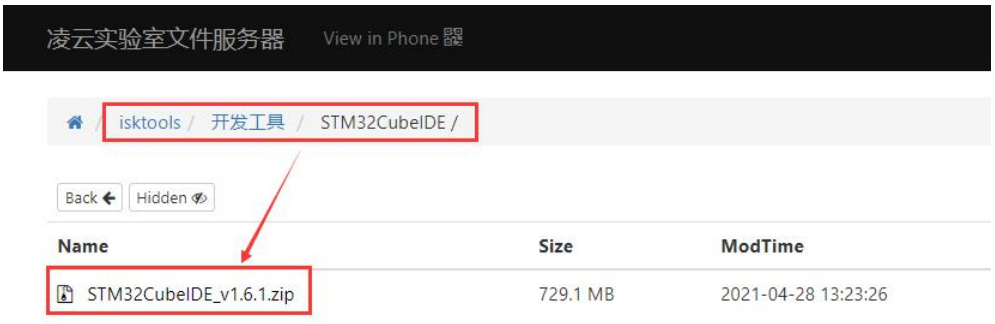
1. STM32CubeIDE 安装

1.1. STM32CubeIDE 简介

STM32CubeIDE 是 ST 意法半导体近几年来基于开源的 Eclipse 项目开发并大力推荐的 STM32 集成开发环境(IDE,Integrated Development Environment)，STM32CubeIDE 里集成了 STM32CubeMX，编译、调试工具等，这样我们可以直接在该环境下编译、烧录、调试代码。其中 STM32CubeMX 是 ST 推出的 MCU 图形化配置界面，通过该工具配置好 CPU 之后，可以基于 HAL 库自动生成 C 程序代码。

ST 为开发者开发提供一些外设操作的库，方便对外设进行控制。最早的是标准外设库（Standard Peripherals Library），后面新出的 HAL 库（Hardware Abstraction Layer）是 ST 公司力推的，该库将要取代之前的标准外设库，ST 当前最新的芯片已经不再提供 stdlib 库了。

STM32CubeIDE 开发工具可以在 ST 官方站点上下载，但 ST 官方站点访问非常慢，另外也需要注册才能下载，大家可以从凌云实验室官方站点下载，当前最新的版本为 STM32CubeIDE v1.6.1： http://studio.iot-yun.club:2211/isktools



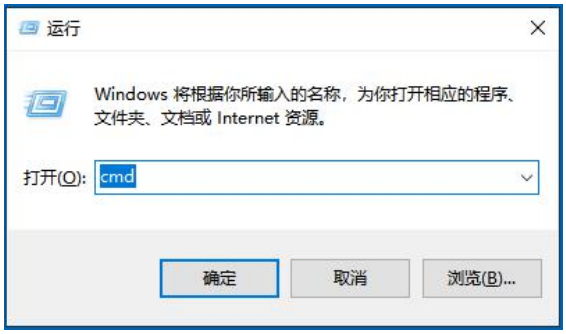
1.2. STM32CubeIDE 安装注意事项

因为 STM32CubeIDE 是基于 Eclipse 开发的，它对中文的支持非常不友好，所以在安装的过程中都不能使用中文。即大家要注意以下几点：

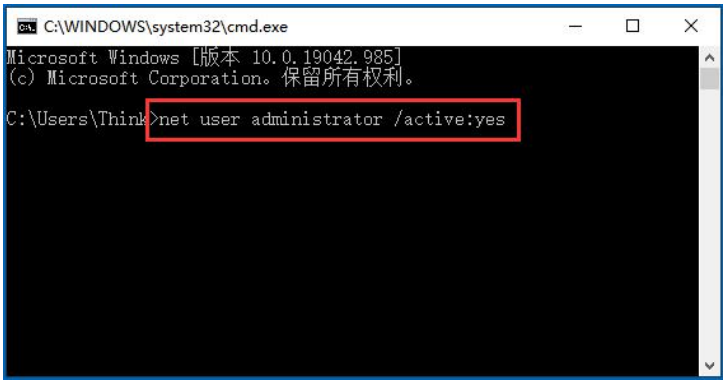
- 解压缩出来的安装文件存放路径不能包含中文，否则不能开始安装；
- STM32CubeIDE 的安装路径一定不能有中文，否则程序可能运行异常；
- STM32CubeIDE 创建的项目存放路径不能有中文，否则程序不能烧录；
- STM32CubeIDE 创建的项目存放路径不能是分区顶层(如 D:/、E:/)，否则不能编译；
- Win10 登录系统用户名不能为中文，否则项目开发时会出错；

如果系统用户名为中文，则可以在 Windows 命令行启用 Administrator 用户，并使用该用户来安装和开发学习。具体方法如下：

1. Win+R 组合键弹出“运行”对话框输入 cmd 并按回车，进入 Windows 命令行：



2. Windows 命令行下执行命令 net user administrator /active:yes 命令使能

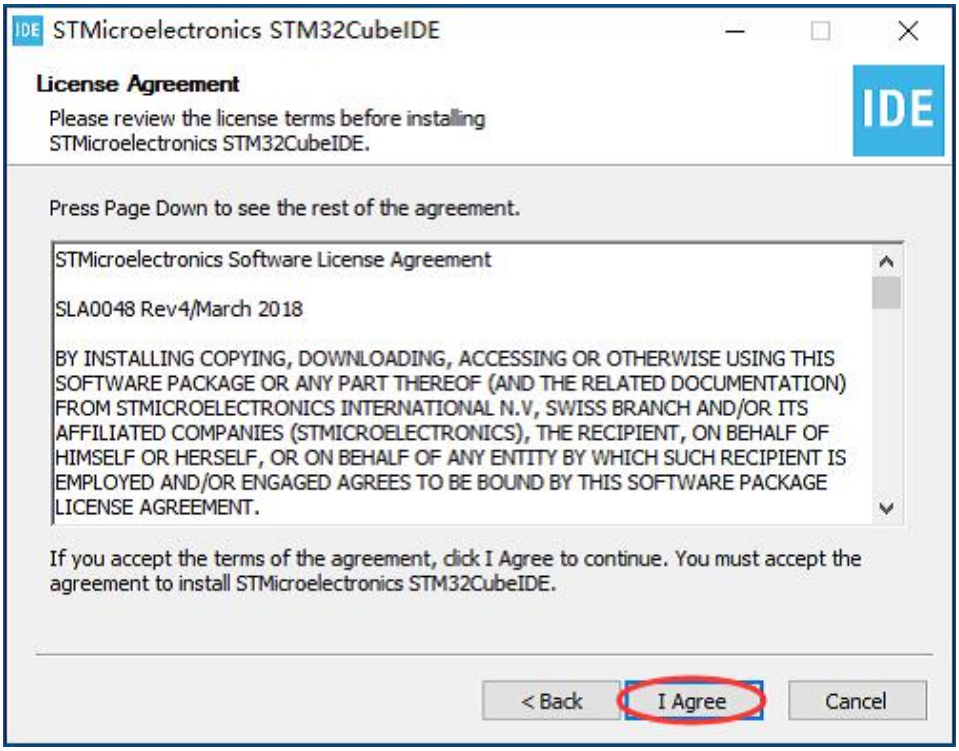


3. 注销或重启 Windows 系统，并点击 Administrator 用户登录

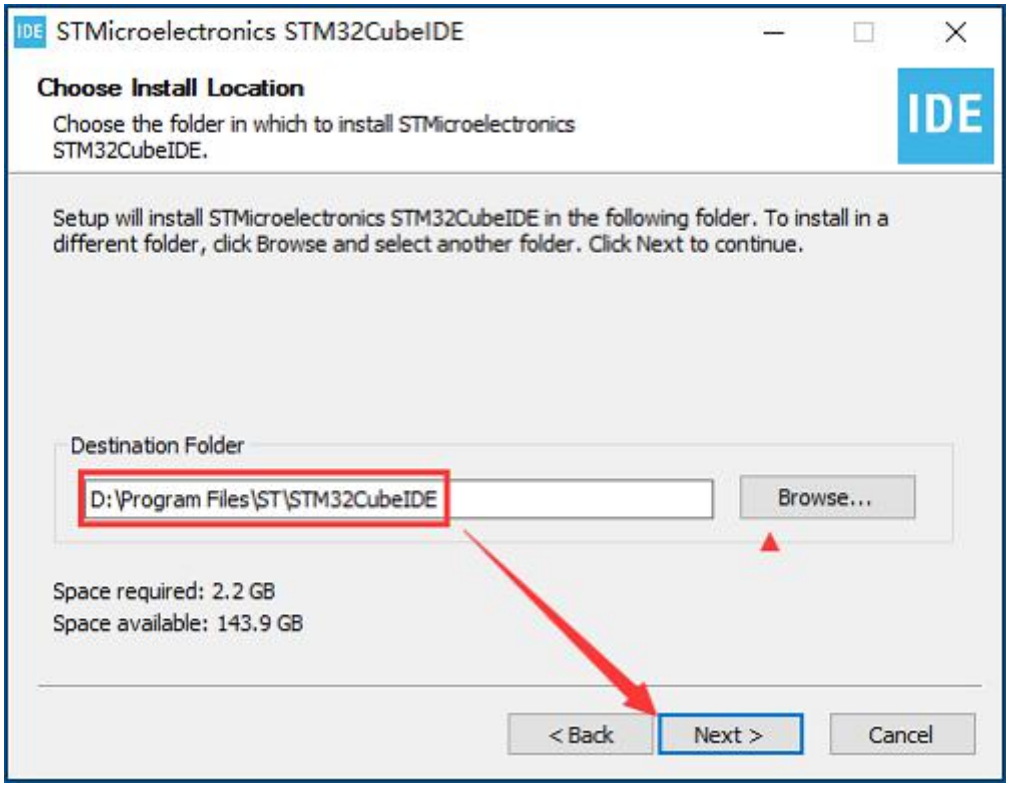
1.3. STM32CubeIDE 安装

1. 点击运行 STM32CubeIDE 的安装程序(注意安装文件存放路径不能存在中文，建议拷贝或解压到桌面上安装)：

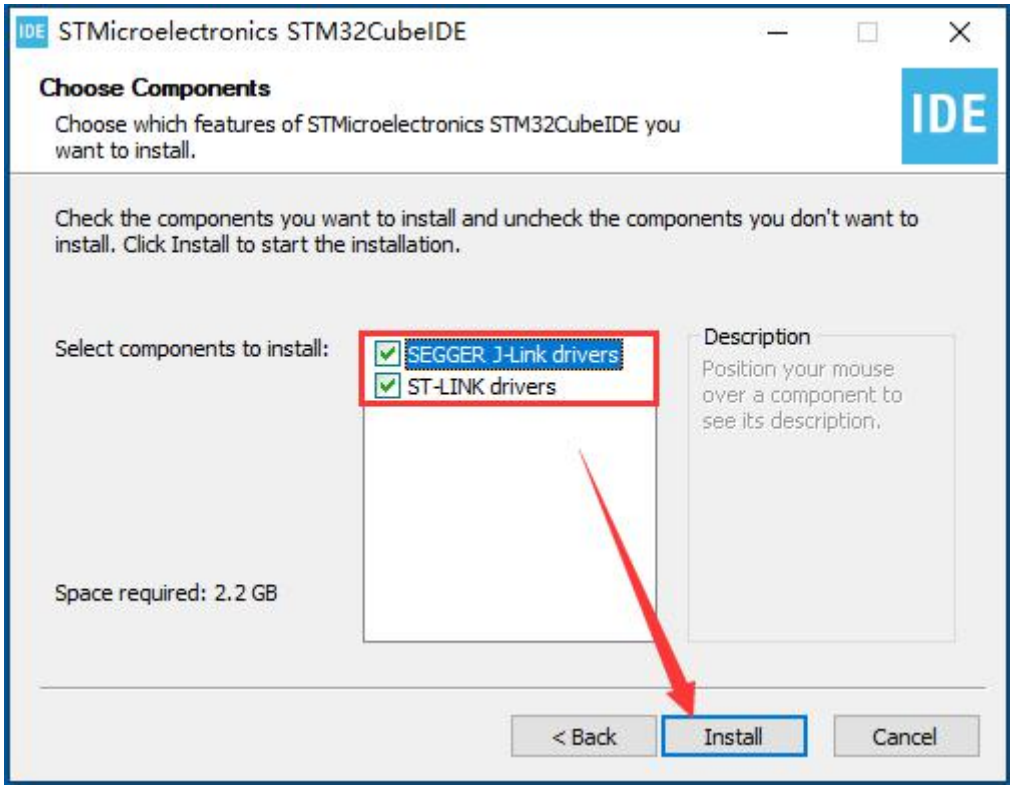
2. 接受安装许可协议



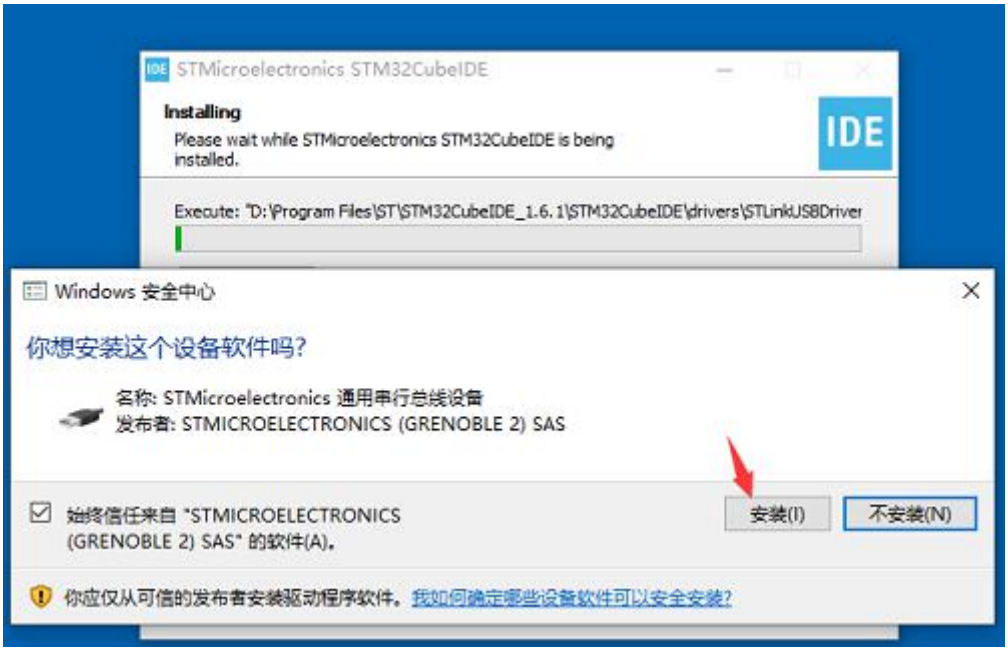
3. 选择合适的安装路径(注意安装路径不能存在中文)



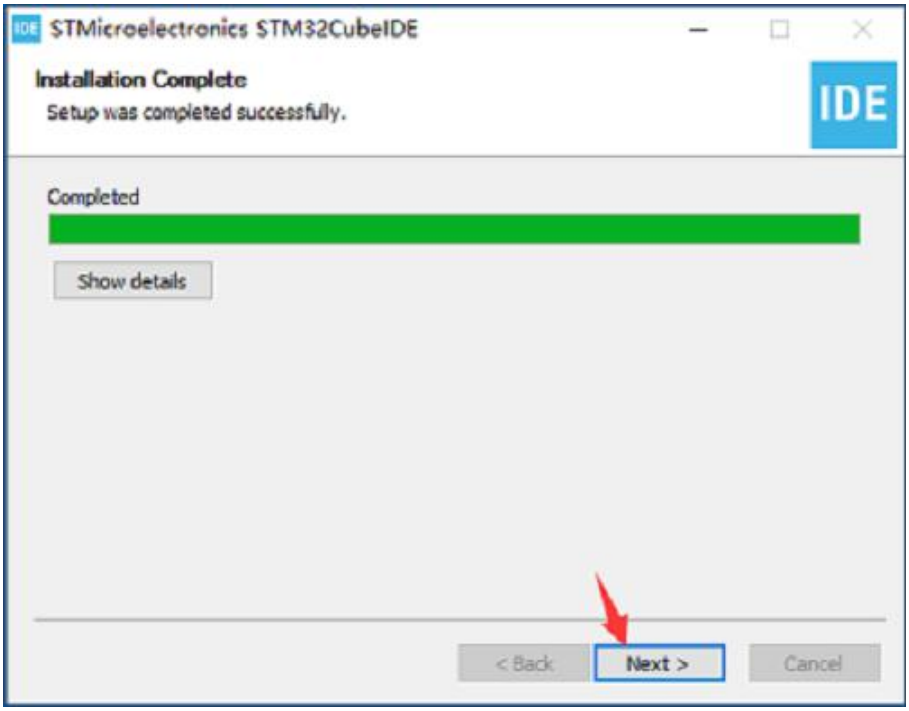
4. 选中两个开发过程中所需的调试软件驱动

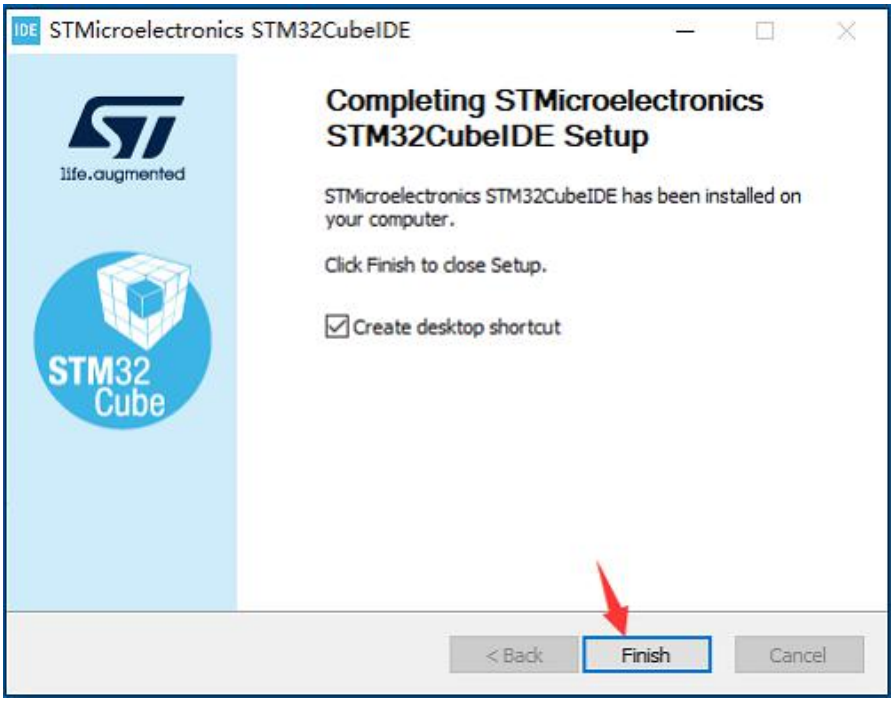


5. 开始安装并同意 ST-Link 的软件安装



6. 安装完成

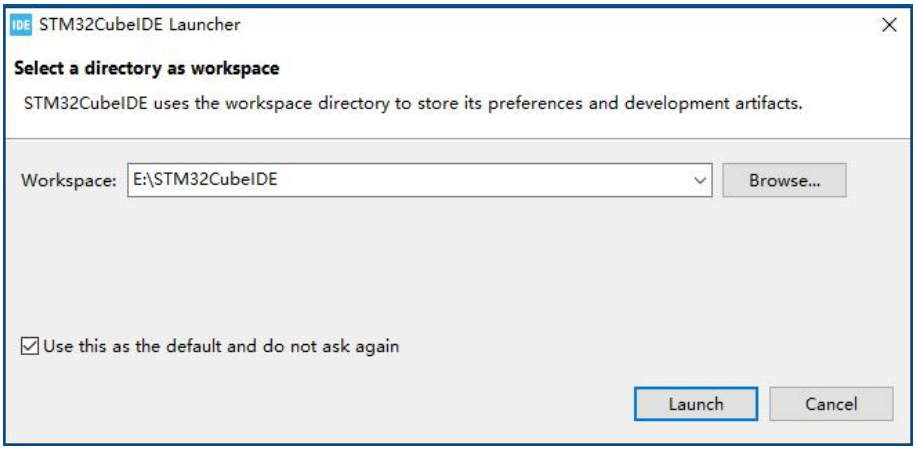




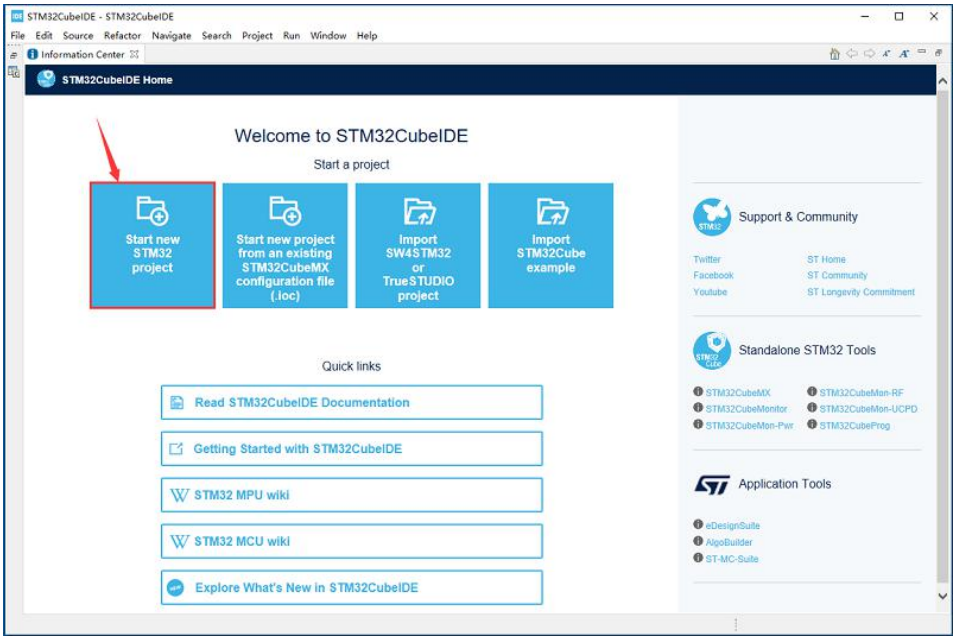
点击 Finish 按钮安装完成！

1.4. STM32CubeIDE 运行

首次运行 STM32CubeIDE 程序，需要配置新创建项目文件和代码的存放路径。(注意该路径不能有中文，也不能为系统分区顶层目录，如 D:\、E:\等)



下面是 STM32CubeIDE 软件的欢迎页，点击 Start New STM32 Project 即可开始创建 STM32 项目了。



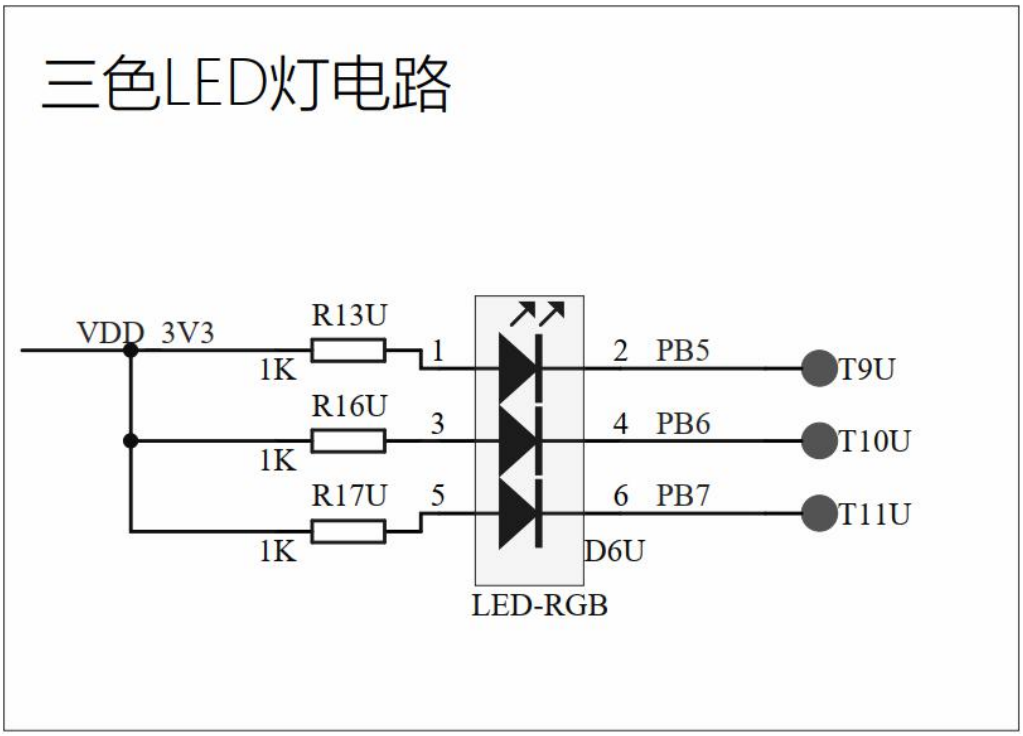
2. STM32CubeIDE 开发实例

接下来我们以 BearKE1 开发板上的流水灯实现为例，详细讲解使用 STM32CubeIDE 进行开发的流程。

2.1. LED 灯工作原理分析

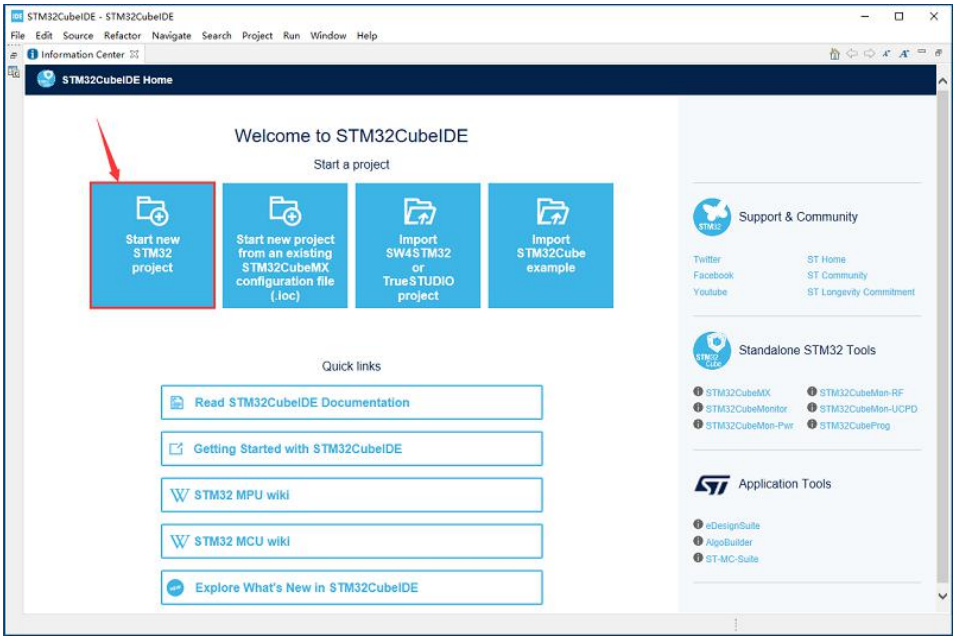
在 BearKE1 开发板上有一个 RGB 三色灯，他们分别连接到了 STM32 CPU 的 PB6、PB7、PB5 这三个管脚上，从下面的原理图上我们可以看出： CPU 的三个管脚如果输出低电平，那 Led 灯就亮了；而如果输出高电平，则三个 Led 就灭了。

接下来我们的主要工作就是通过编程控制 CPU，让这个三个管脚输出高低电平从而控制 LED 灯的亮灭。

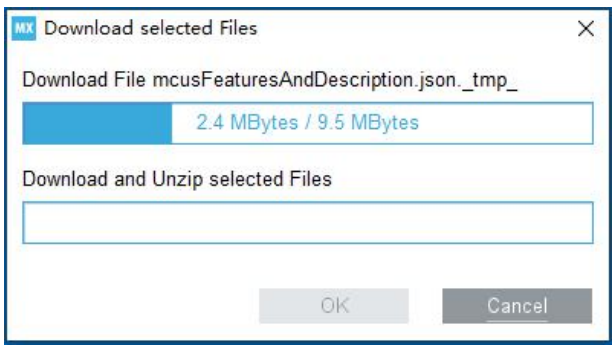


2.2. 创建 Led 流水灯项目

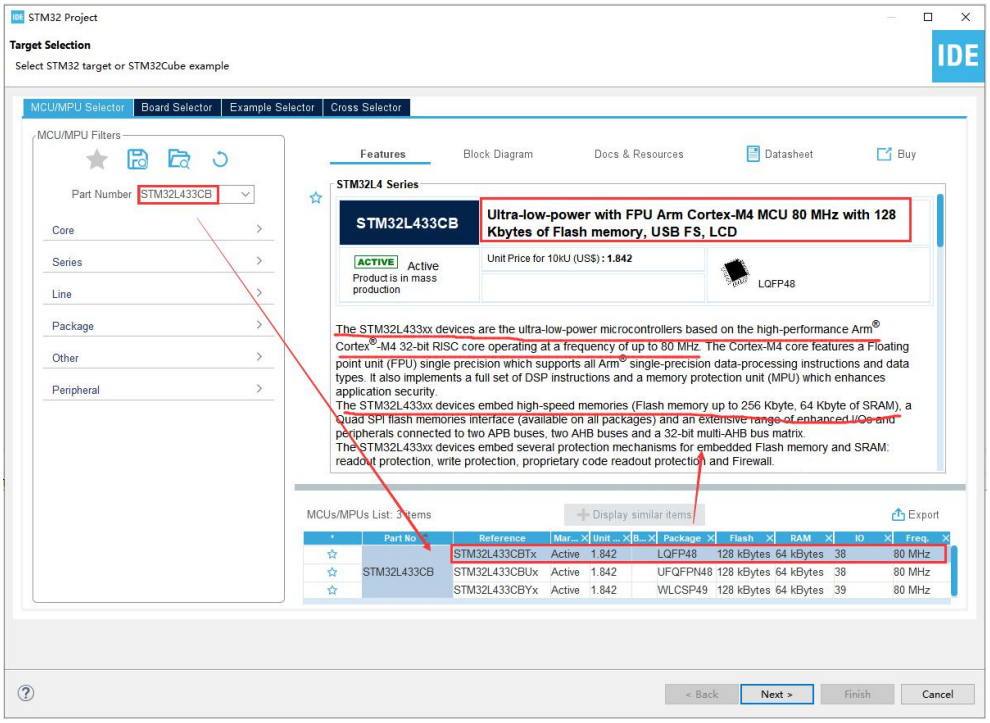
1. 使用 STM32CubeIDE 开始创建一个新的项目



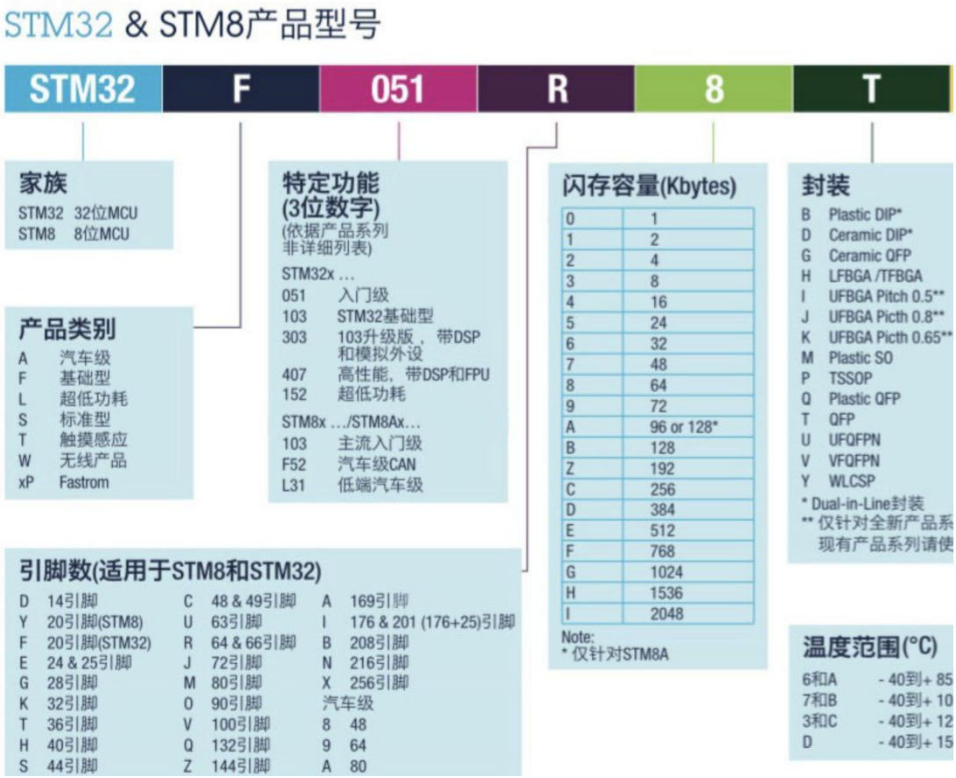
2. 如果首次使用或长久没使用则会提示软件更新，点击确认下载更新：



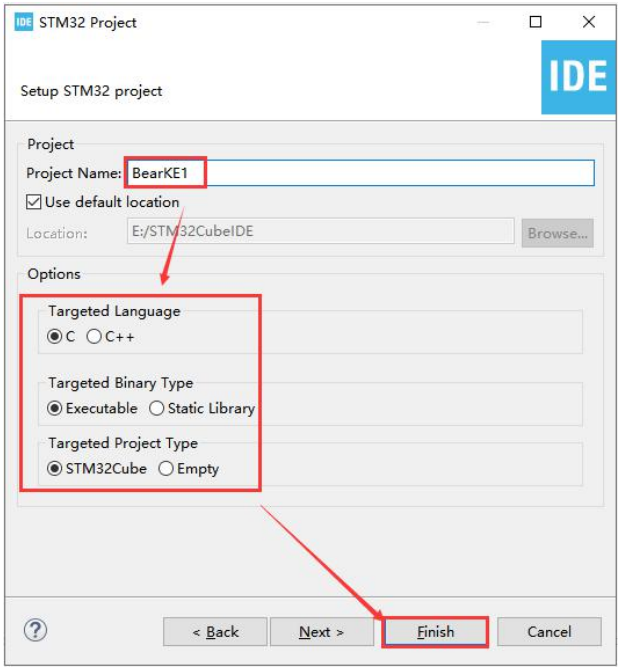
3. 更新完成后，在 Part Number 里输入开发板或产品所使用的 CPU 型号来查询相应 CPU，然后选中 CPU 的具体型号并点击 Next 开始创建项目。BearKE1 开发板使用的 CPU 型号为 STM32L433CBT6，这里我们输入 STM32L433CB 即可。



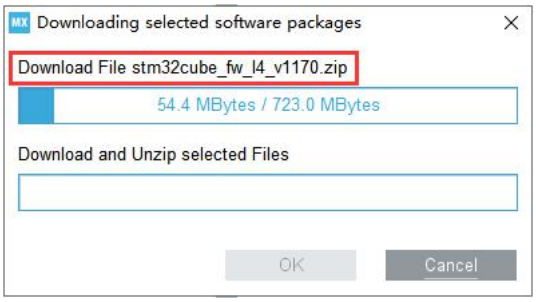
4. STM32 命名规则



5. 设置项目名称并确认项目源码保存路径：

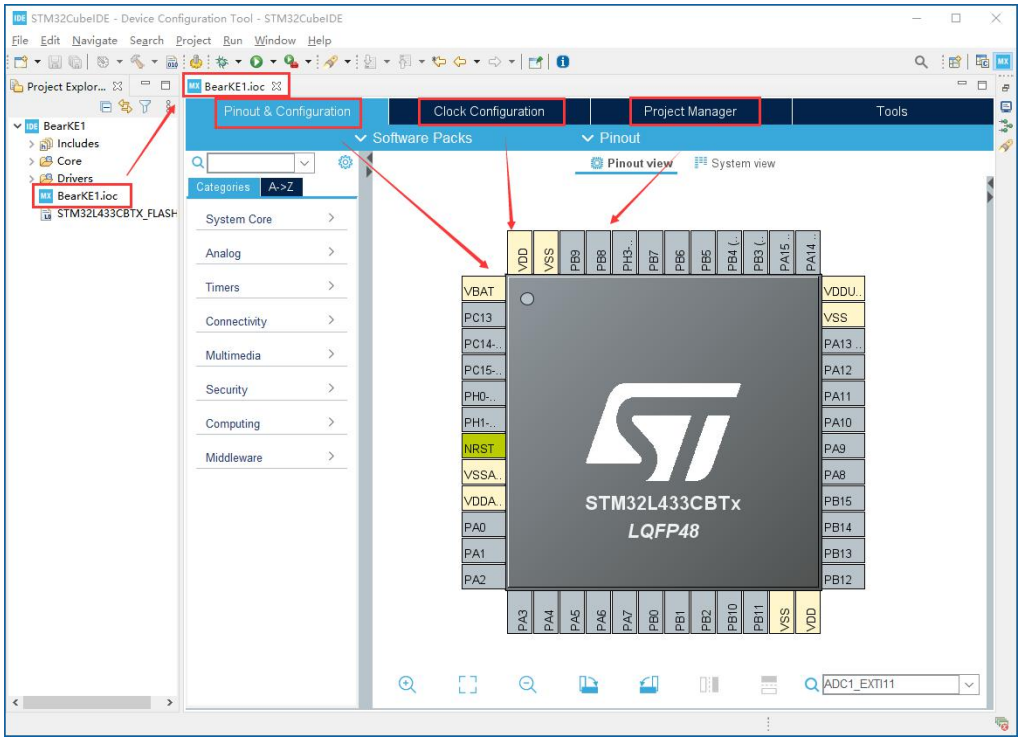


6. 如果首次使用或长时间未使用，则会自动更新 STM32 Hal 库源码：

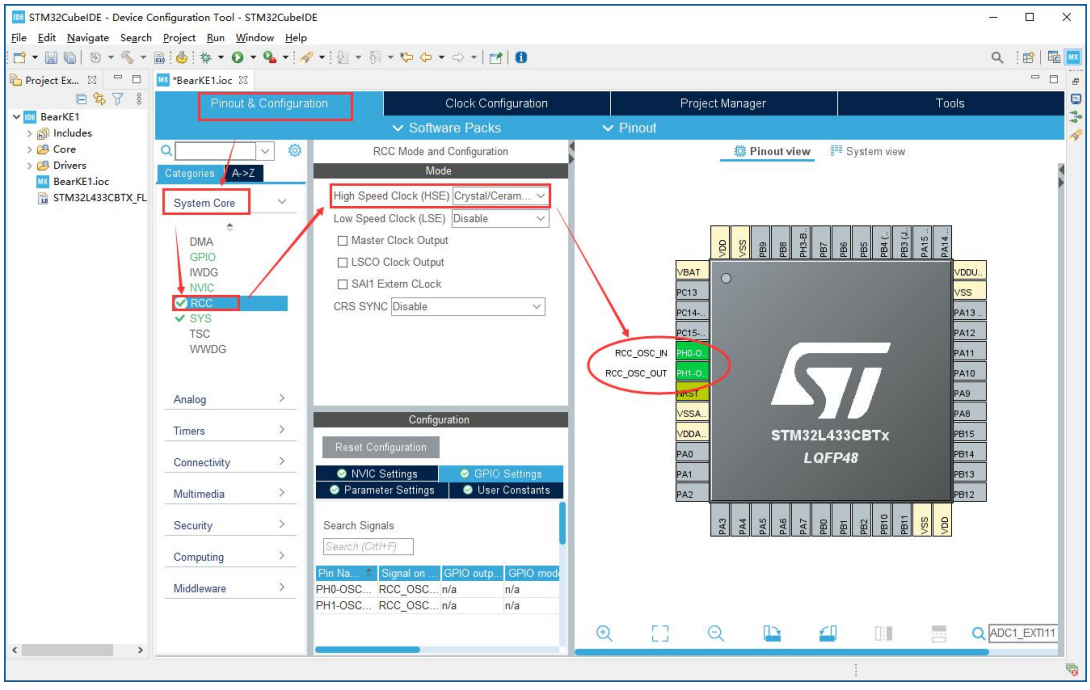


2.3. STM32CubeMX 配置

1. 项目创建好之后，默认将会进入到 STM32CubeMX 图形化配置界面，通过该界面可以对 CPU 进行配置：

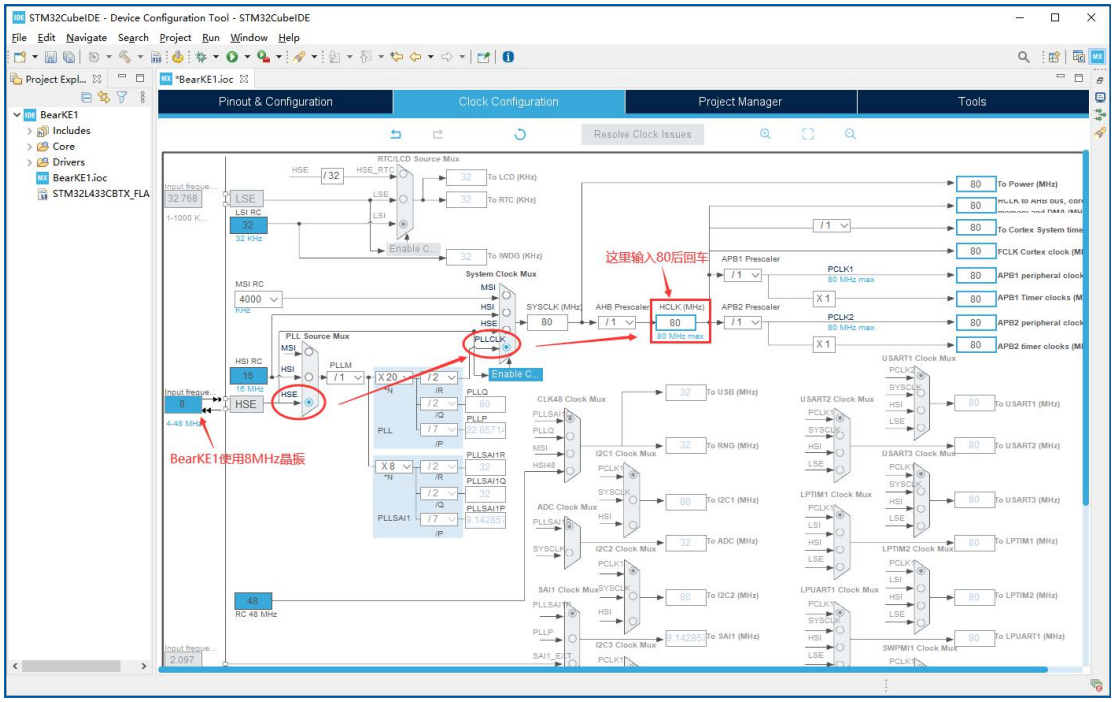


2. 首先配置使能 CPU 外部晶振

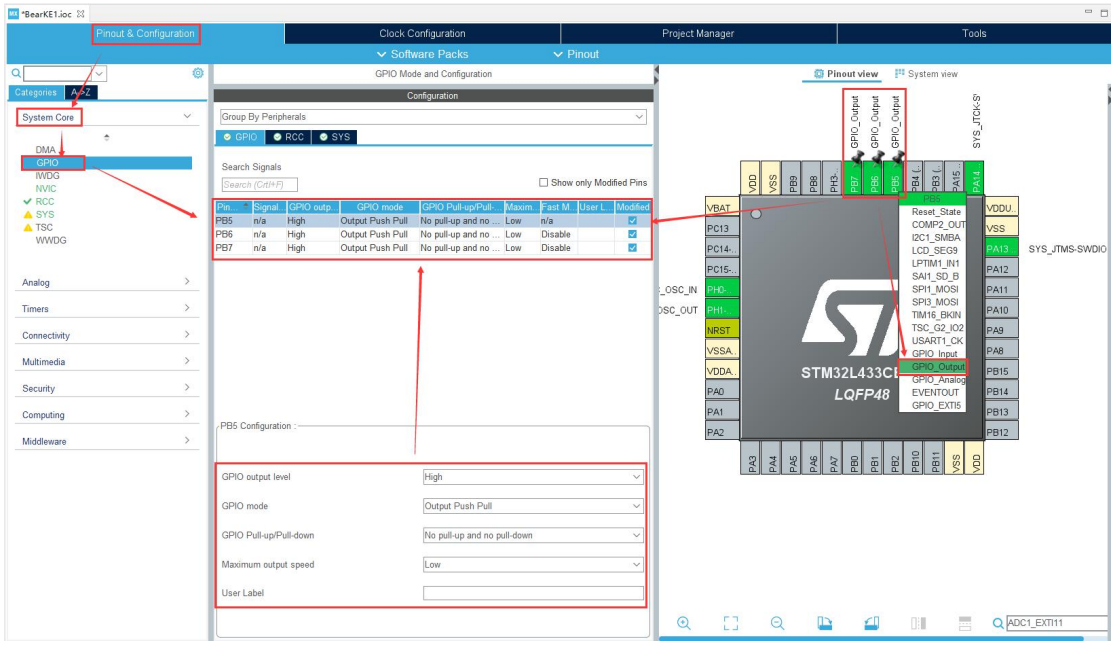


3. 配置 CPU 时钟树

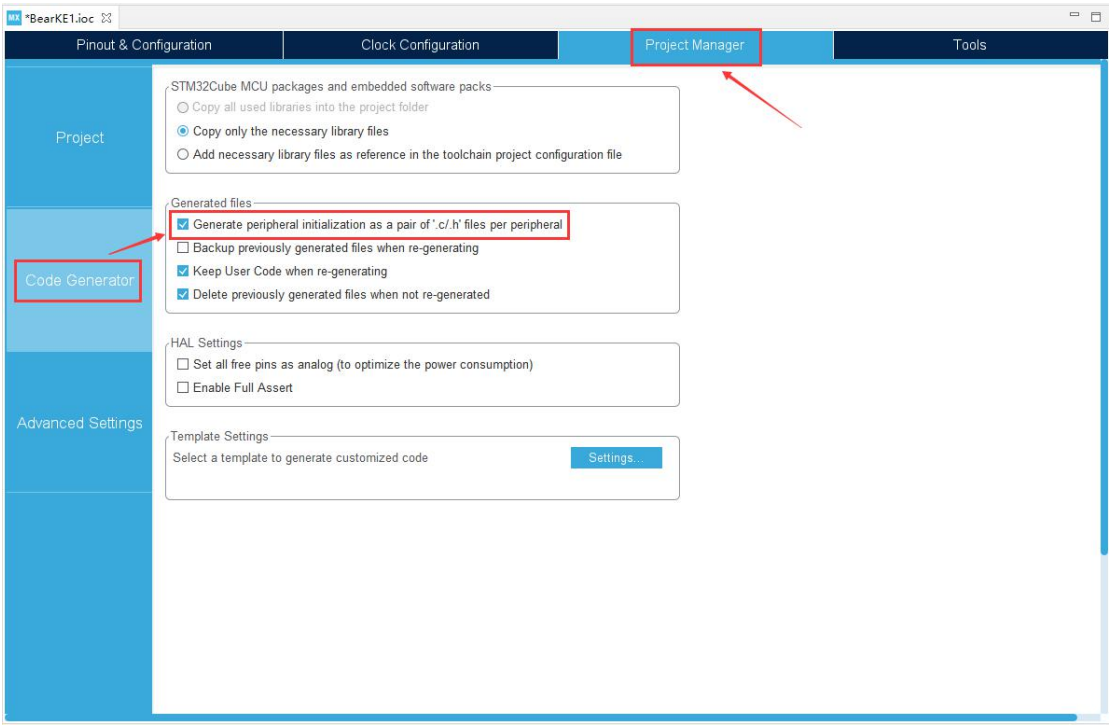
在 HCLK 配置里输入 80 后按回车，将会自动配置好 CPU 的时钟树：



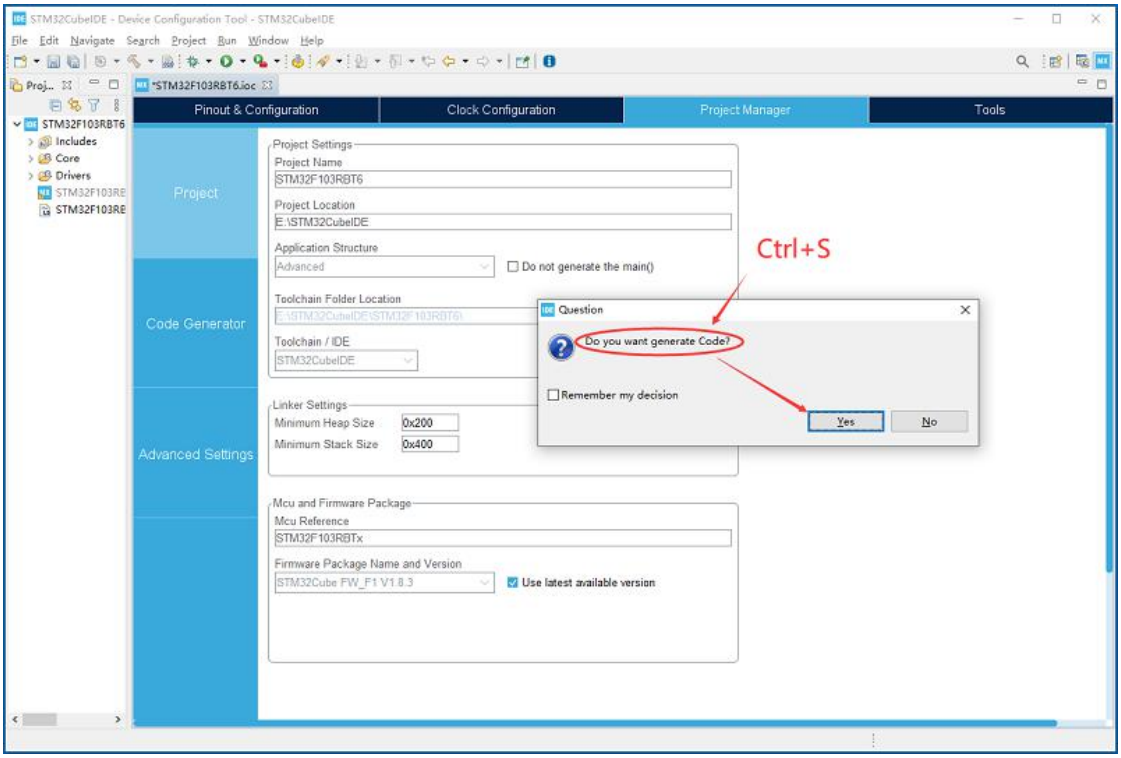
4. 配置 Led 连接的相应 CPU 管脚，设置默认电平为高电平（灯状态为灭）



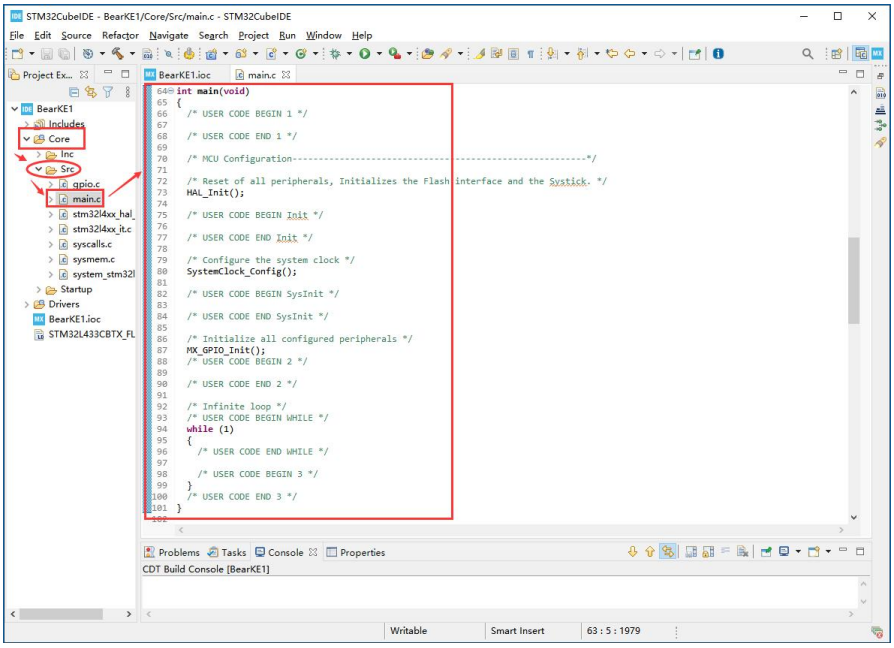
5. 配置代码生产的相关选项



6. 任意位置按 Ctrl+S 将开始自动生产代码



7. 下面是自动生成的代码



8. 下面是项目源码存储路径及目录结构

2.4. 程序修改及编译

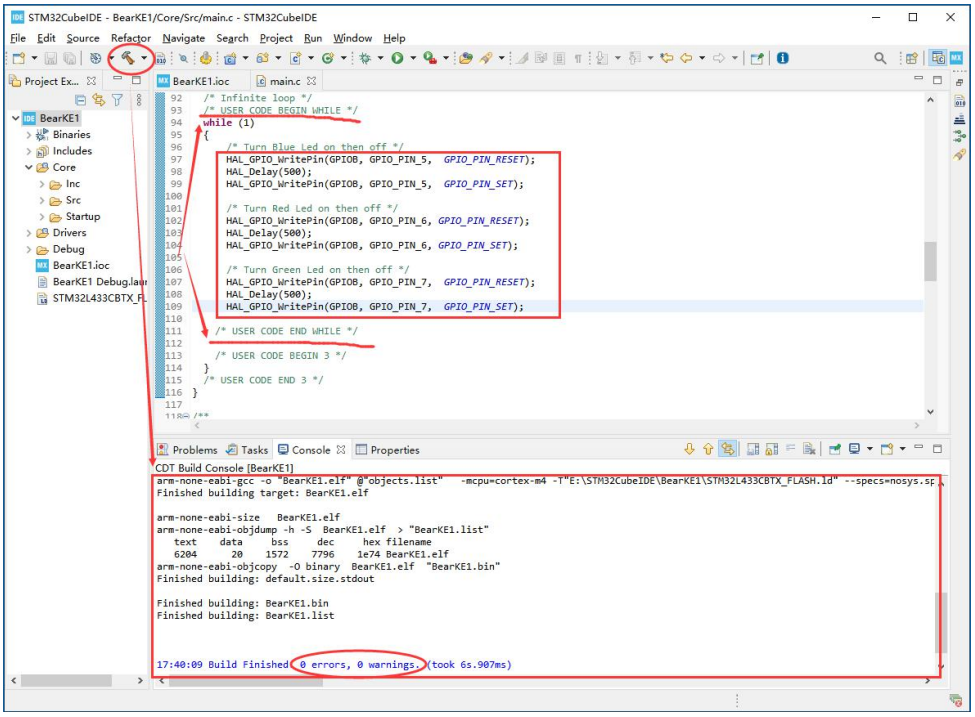
STM32CubeIDE 在添加代码时务必在 USER CODE BEGIN ... 和 USER CODE END ... 之间添加，否则 STM32CubeIDE 在重新生成代码时，会把我们添加的代码给删除。

1. 修改 main.c 中的 while 循环代码，添加 Led 流水灯的控制代码如下:

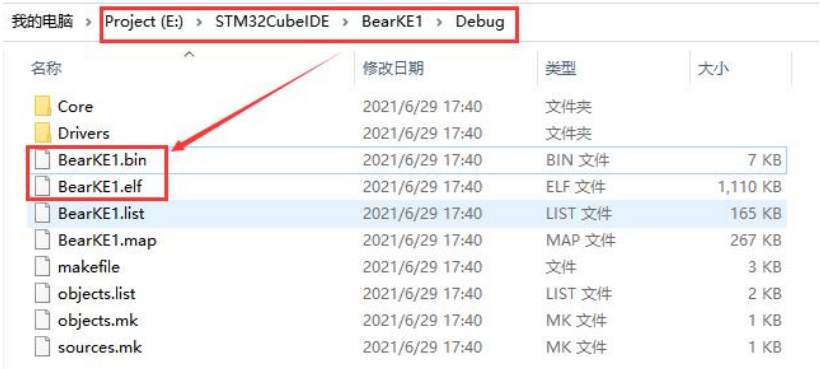
```
... ...
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* Turn Blue Led on then off */
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5,  GPIO_PIN_RESET);
    HAL_Delay(500);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5,  GPIO_PIN_SET);

    /* Turn Red Led on then off */
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_RESET);
    HAL_Delay(500);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_SET);

    /* Turn Green Led on then off */
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7,  GPIO_PIN_RESET);
    HAL_Delay(500);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7,  GPIO_PIN_SET);

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
... ...
```

2. 源码编译

点击工具栏上的锤子按钮将开始编译代码：



编译生成的可执行文件将会存放到项目的 Debug 文件夹下：



这里的可执行文件格式有两个：.bin 文件和.elf 文件。

- STM32CubeIDE 使用的是 GCC 编译，它编译生成的可执行文件格式默认为 ELF(Executable and linking format)格式，该格式是 Linux 系统下的常用可执行文件格式；

- bin 文件就是直接的二进制文件，内部没有地址标记，一般用编程器直接烧写到目标开发板上运行；

编译生成的可执行程序必须烧录到开发板上才能运行，STM32 CPU 支持两种烧录方式，一种是使用串口 ISP 烧录，另外一种是使用 ST-Link/J-link 等 ARM 调试器烧录，接下来我们将分别介绍这两种烧录方式。

3. STM32 串口 ISP 烧录

3.1. 串口 ISP 烧录烧录原理

在 ST 官网上下载的《STM32F103_RM0008.pdf》datasheet 3.4 节中有提到，STM32 一共有三种启动方式，其启动相关配置说明如下：

3.4 Boot configuration

In the STM32F10xxx, 3 different boot modes can be selected through BOOT[1:0] pins as shown in *Table 9*.

Table 9. Boot modes

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as boot space
0	1	System memory	System memory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space

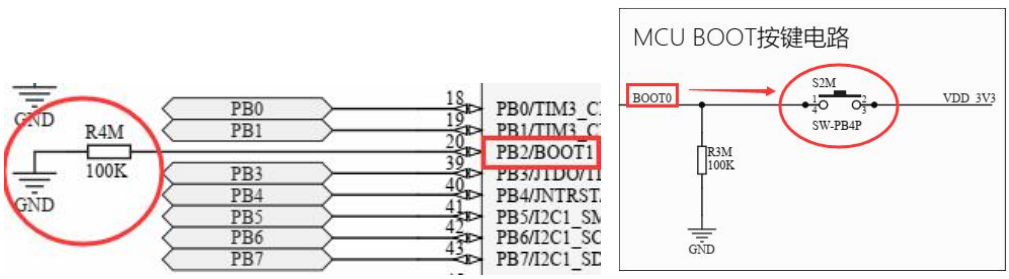
STM32 三种启动模式对应的存储介质均是芯片内置的，它们分别是：

① Main Falsh memory，即 STM32 CPU 内置的 64KB Flash。我们通过各种方式烧录（串口、ST-Link 等）的目的就是将程序烧录到这这片区域，这样系统上电时默认会从这里启动；

② System memory，这是 STM32 CPU 内置的一片 ROM，出厂后无法修改。STM32 在出厂时，由 ST 在这个区域内部预置了一段 BootLoader，也就是我们常说的 ISP 程序，通过该程序我们可以使用串口、ST-Link 等方式来烧录；

③ Embedded SRAM ，即 STM32 CPU 内置的 20KB SRAM，这个模式一般用于程序调试。假如我们在调试过程中只修改了代码中一个很小的不复，然后就需要重新擦除整个 Flash，比较的费时，可以考虑从这个模式启动代码，用于快速的程序调试，等程序调试完成后，再将程序下载到 Flash 中。不过这种模式我们很少用到。

另外，我们从 datasheet 的描述中也可以了解到，STM32 的启动模式由 CPU 的 Boot0 和 Boot1 两个管脚的电平决定。而在 BearKE1 的原理图中我们可以看到，Boot1 这个管脚被 R4M 这个电阻拉到了地上为低电平；而 Boot0 这个管脚则受 S2M 这个按键的控制，如果按键没有被按下，则默认是低电平，如果按键按下了则是高电平。



其中原理图上的 S2M 按键对应到开发板上的 MCU_BOOT 按键，这样：

- 如果 MCU_BOOT 被按下，则 BOOT0 管脚为高电平；此时 BOOT1 为 0，Boot0 为 1，即从 System memory 中的 ISP 程序启动，进入到串口烧录模式；

- 如果 MCU_BOOT 没有被按下，则 BOOT0 管脚为低电平；此时 BOOT1 为 0，Boot0 为 0，即从 Main Flash memory 启动，即 STM32 开发板从片内 Flash 中正常启动；

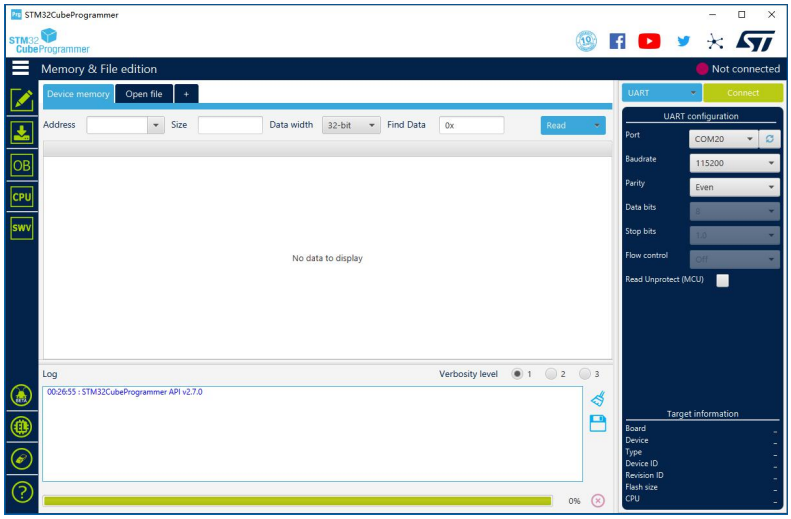
由此可知，如果我们要使用串口烧录的话，必须在启动或 CPU 复位时按下 MCU_BOOT 管脚方能进入到串口烧录模式。

3.2. STM32CubeProgrammer 安装

STM32CubeProgrammer 是 ST 公司新推出的一个高集成度的跨平台（支持 Windows、MacOS 和 Linux）编程烧录工具软件，它同时提供图形用户界面或命令行界面，另外也支持多种连接方式（JTAG、SWD、USB、UART 等）。

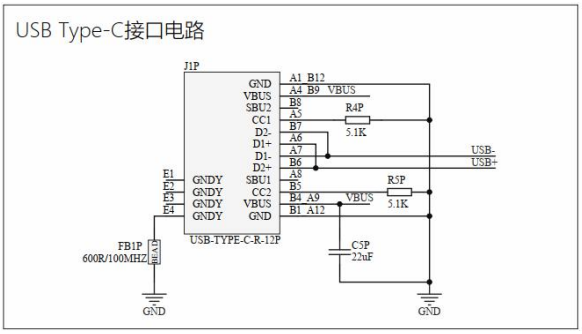
同样的该工具可以在 ST 官方站点上下载，但 ST 官方站点访问非常慢，另外也需要注册才能下载， 大家可以从凌云实验室官方站点下载： http://studio.iot-yun.club:2211/isktools

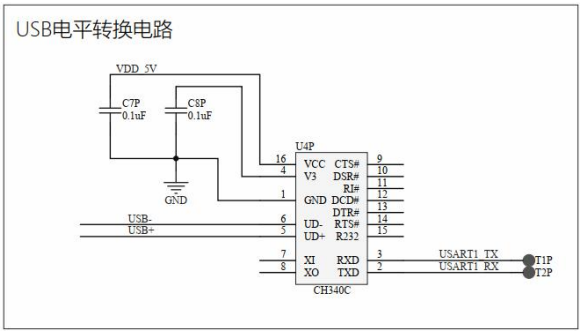
其安装过程比较简单，同 STM32CubeIDE 安装一样，所有路径都不能有中文，修改默认安装路径，然后其他所有采用默认配置安装即可。这里就不详细讲解该软件的安装了。下面是该软件的运行界面：



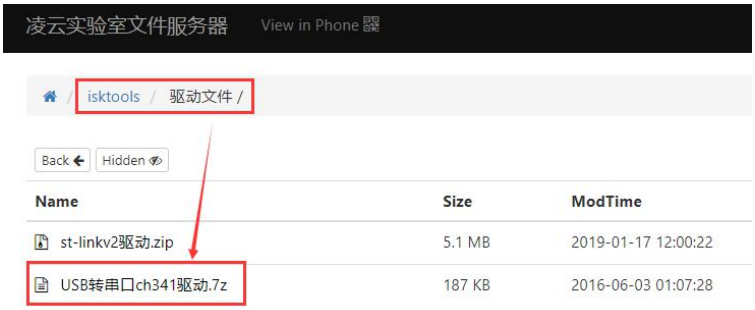
3.3. 串口 ISP 烧录

BearKE1 开发板上 TypeC USB 接口，除了给开发板提供电源以外，还使用 CH340 芯片提供了一个 USB 转串口设备，同时该 USB 转串口芯片连接到了 CPU 的 UART1 管脚上，这样我们就可以使用该接口来将 STM32CubeIDE 编译生成的可执行程序烧录到开发板上。下面是开发板上相关电路的原理图：



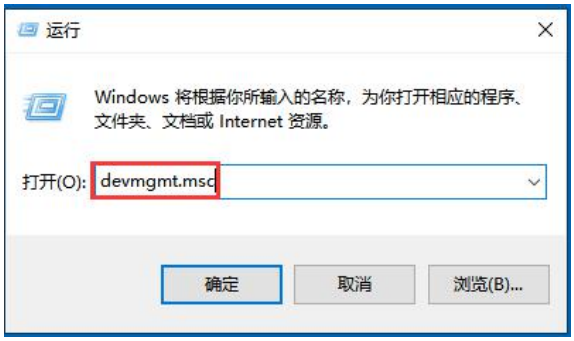


Windows 系统一般默认支持 CH340 USB 转串口芯片驱动，如果不支持的话，则可以从凌云实验室官网上下载并安装该设备驱动： http://studio.iot-yun.club:2211/isktools

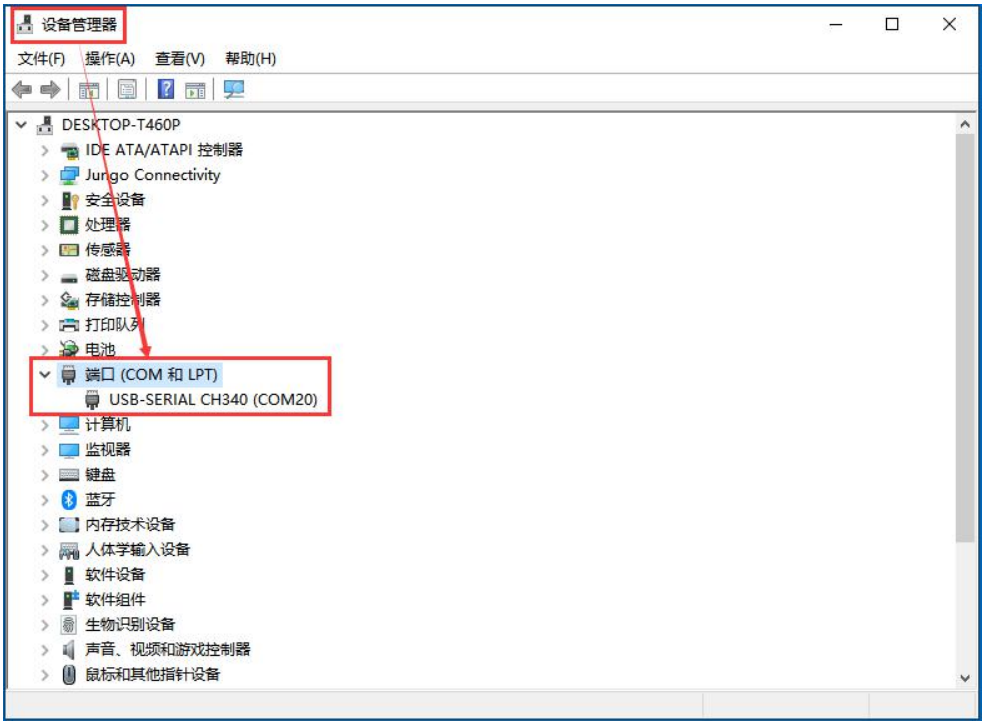


如果一切正常，使用 Type C USB 线连接开发板与 PC，然后给开发板上电之后，我们在设备管理器上应该可以看到一个新的串口设备，具体方法是：

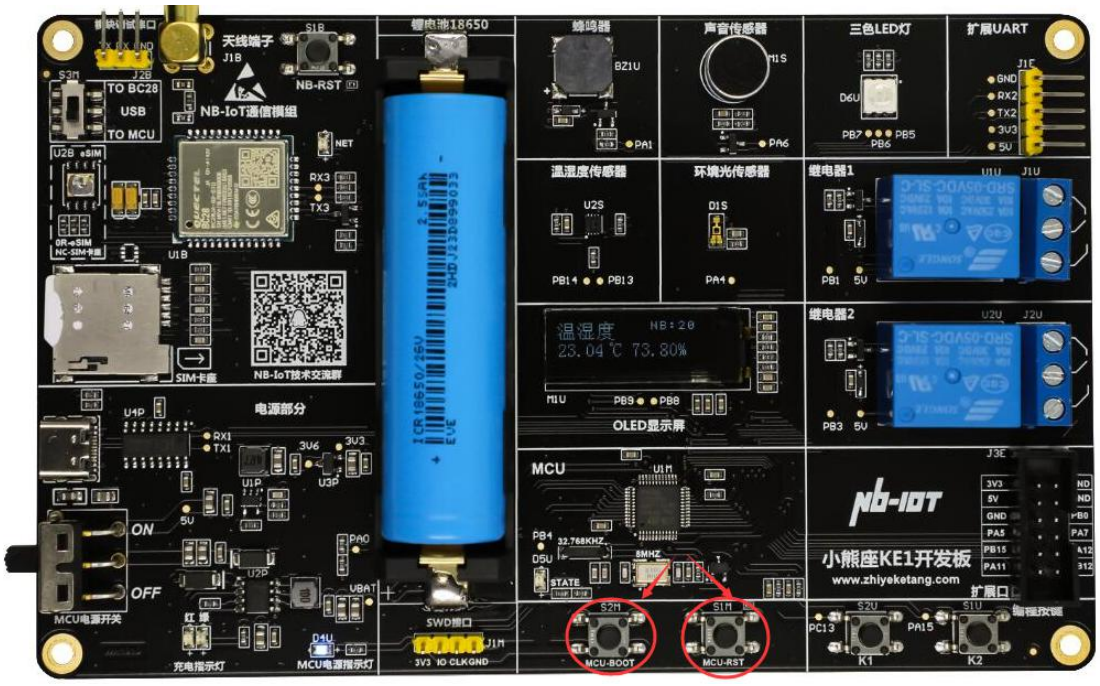
Win+R 快捷键打开运行，输入 devmgmt.msc 命令打开设备管理器：



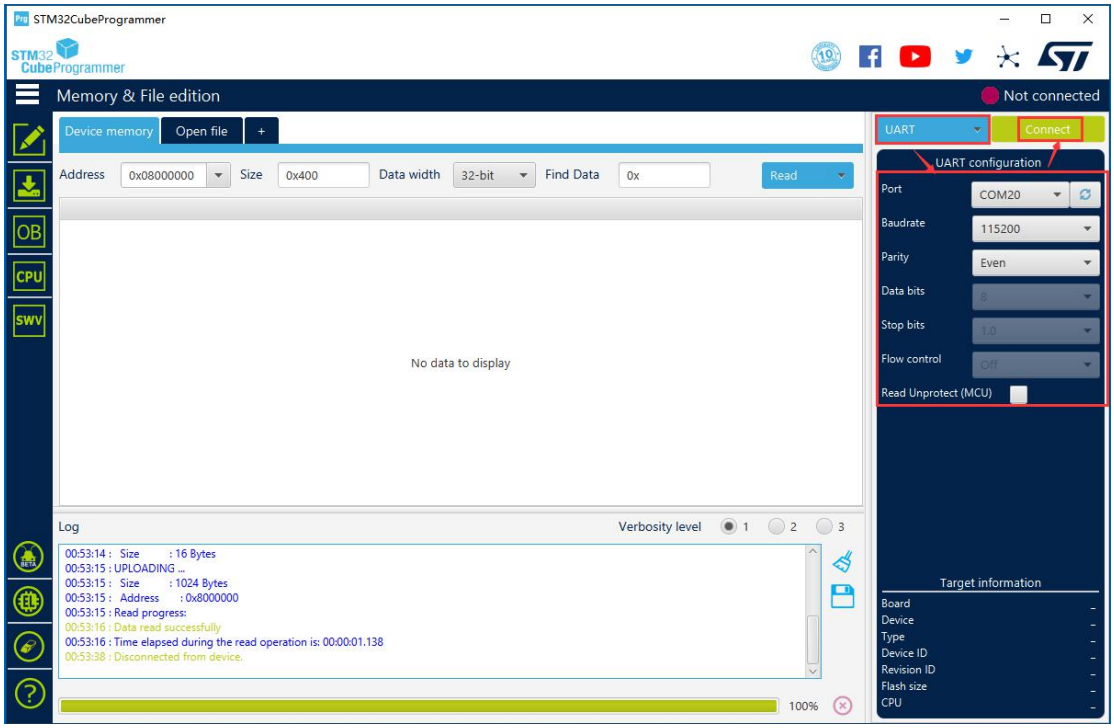
如果一切正常则在设备管理器中，我们可以看到一个新的 USB 转串口设备。否则的话，检查 USB 转串口驱动是否正常安装：



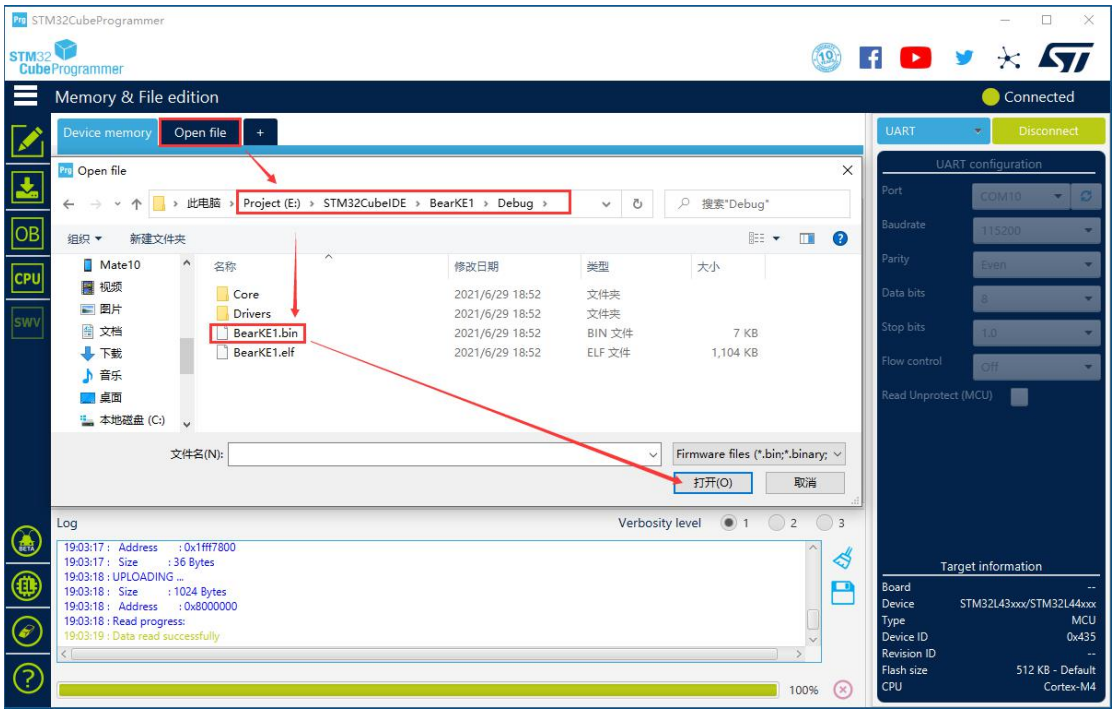
接下来，我们按住开发板的 MCU_BOOT 按键，同时再按一下 MCU_RST 按键让系统复位，再松开 MCU_BOOT 按键此时开发板将进入到 ISP 串口烧录模式。



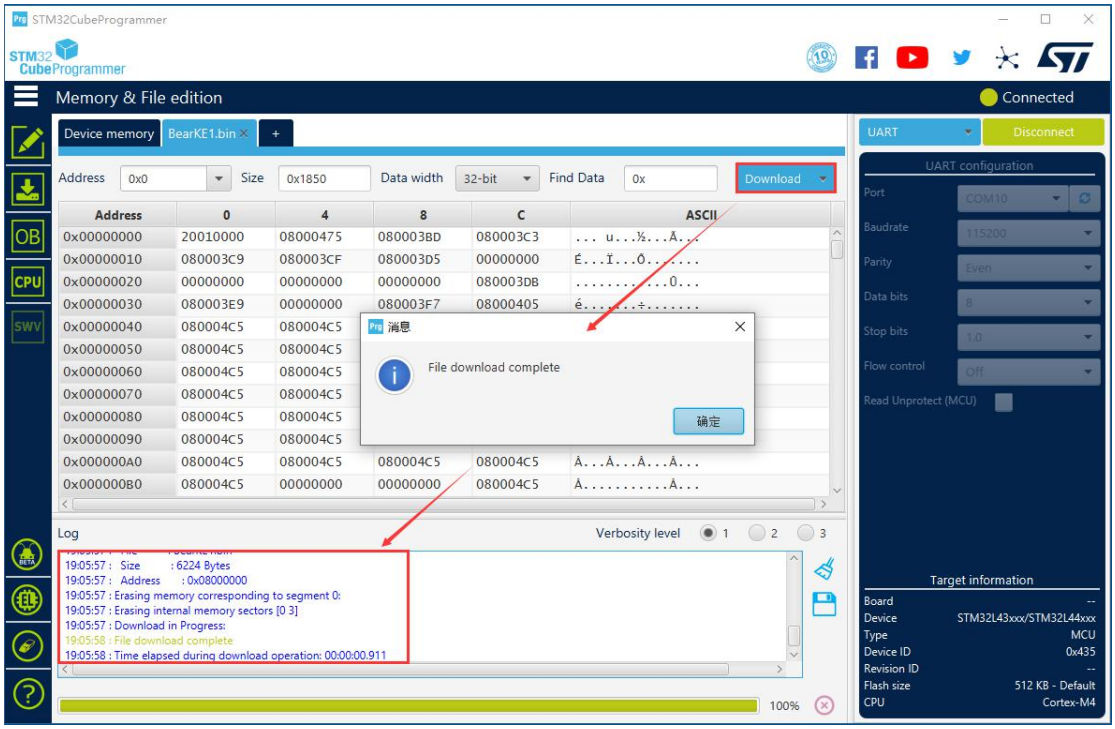
接下来我们运行 STM32CubeProgrammer 程序，选择相应的串口连接设备开始烧录：



正常连接到 STM32 开发板设备之后，点击“Open file”或旁边的 “+”按钮，选择之前编译生成的二进制文件，在这里.bin 和.elf 文件都可以烧录。



加载好烧录文件之后，点击“Download”按钮即可完成程序的烧录：



烧录成功之后，按 MCU_RST 复位按钮将会重启开发板，这时将会看到红绿蓝三个 Led 灯依次闪烁：

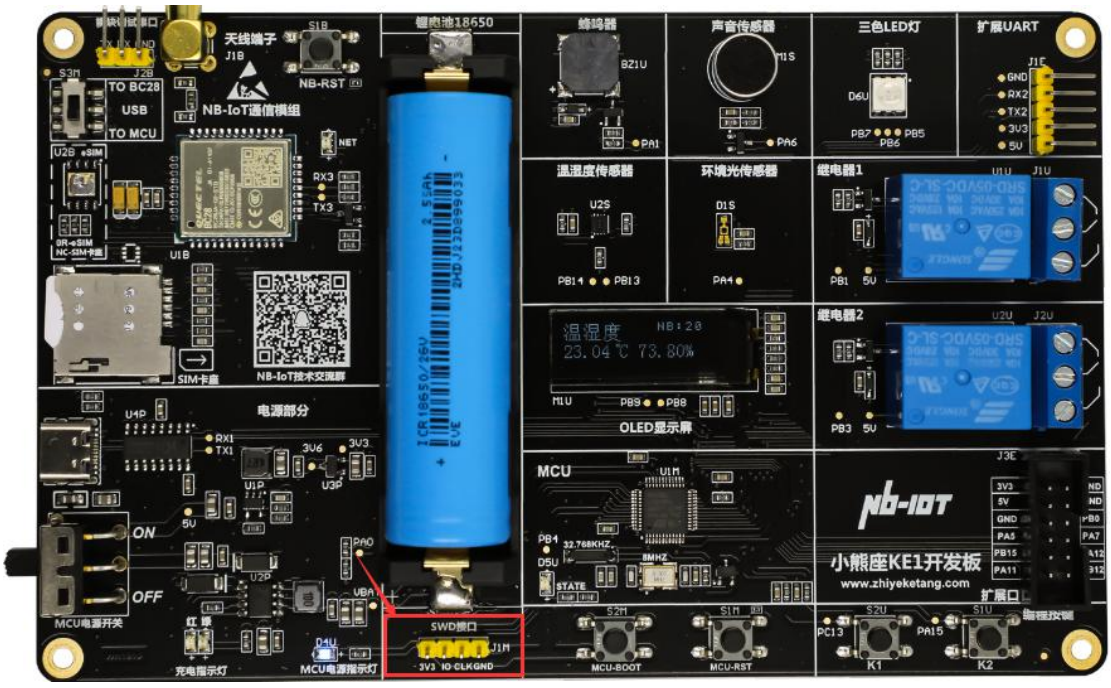
4. STM32 程序烧录与运行

4.1. ST-Link 介绍

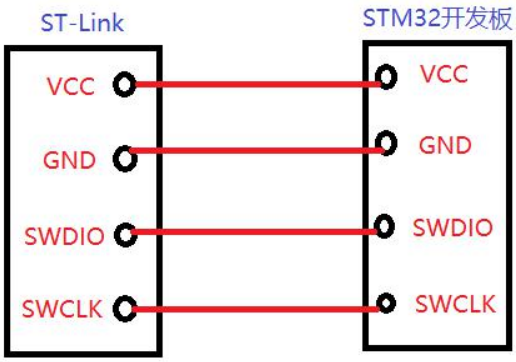
ST-Link V2 是 ST 意法半导体为评估开发 STM8 系列和 STM32 系列 MCU 而设计的集成仿真与下载为一体的开发工具（也是大家口中的烧录器），它具有 SWIM、JTAG / SWD 等通信接口。其中 STM8 系列通过 SWIM 接口与 ST-LINK /V2 连接，而 STM32 系列通过 JTAG / SWD 接口与 ST-LINK /V2 连接。

ST-Link V2 有下面两种物理接口形式，如果只是调试 STM32 则使用左侧 U 盘样式模块会比较多，而如果既要用到 STM8 又要用到 STM32 则使用右侧 ST 原厂 ST-Link 的会比较方便，接下来我们将介绍这两种烧录器连接 BearKE1 开发板进行烧录调试的方法。

下图是 BearKE1 开发板的功能示意图，其中左下角为 ST-Link 调试接口：



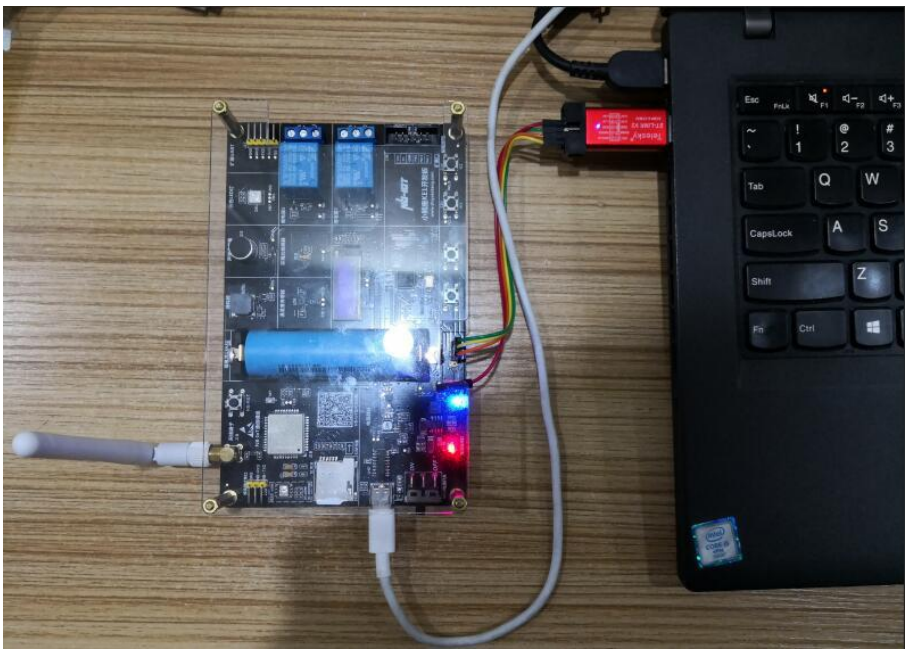
不管是哪种形式的 ST-Link 和 STM32 开发板，其接线的原理都如下图所示。如果不需要 ST-Link 供电的话，则可以不用连接 VCC：



1. U 盘样式 ST-Link 接线

首先介绍 U 盘样式的 ST-Link V2 的连线方式，下图是该调试器的 10 个针脚定义：

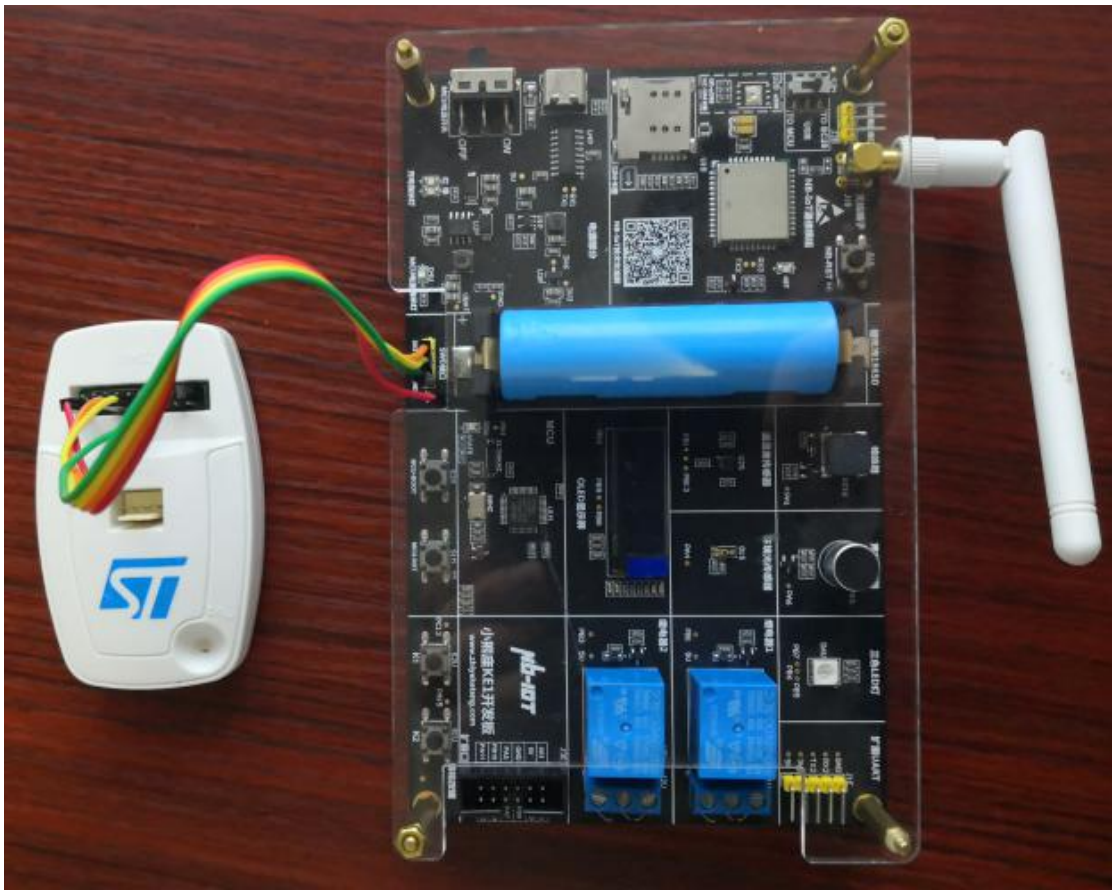
我们直接使用杜邦线连接，如下图所示：



2. 原厂 ST-Link 接线

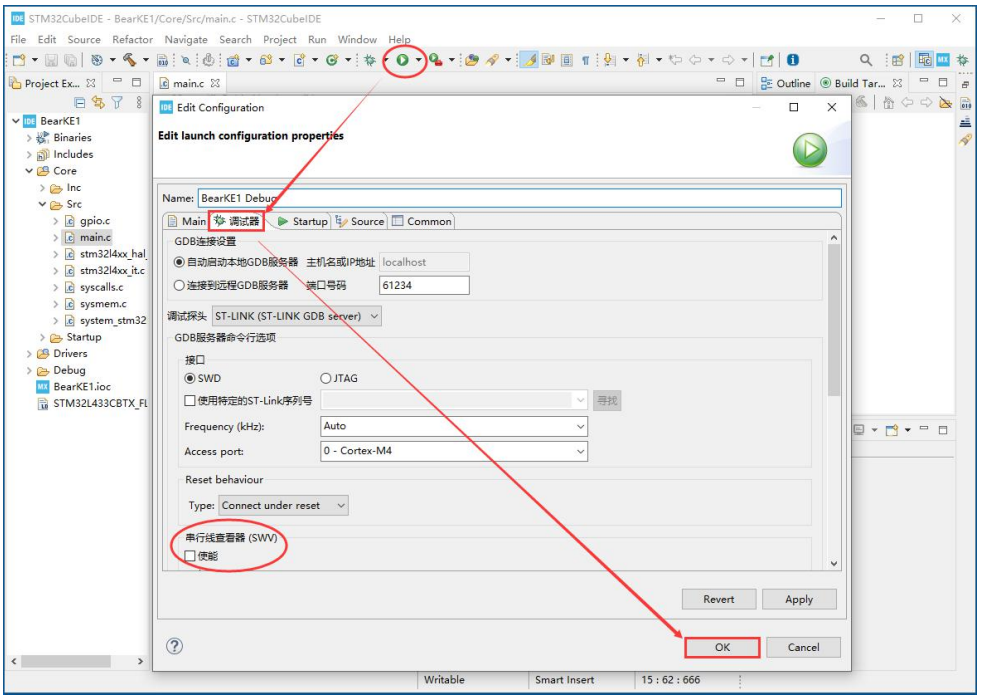
接下来介绍原厂 ST-Link V2 的连线方式，下图是该调试器的 10 个针脚定义：

我们直接使用杜邦线连接：



4.2. ST-Link 程序烧录

ST-Link 硬件连好并接入到 PC 上， 在 STM32CubeIDE 上按 调试按钮则开始烧录，如果首次运行的话，则需要先配置 ST-Link，这里直接使用默认的而配置即可，主要不能使能最下面的 SWV 模式。：



配置好并保存退出之后，再按一次即可开始烧录。烧录成功之后，按 MCU_RST 复位按钮将会重启开发板，这时将会看到红绿蓝三个 Led 灯依次闪烁。