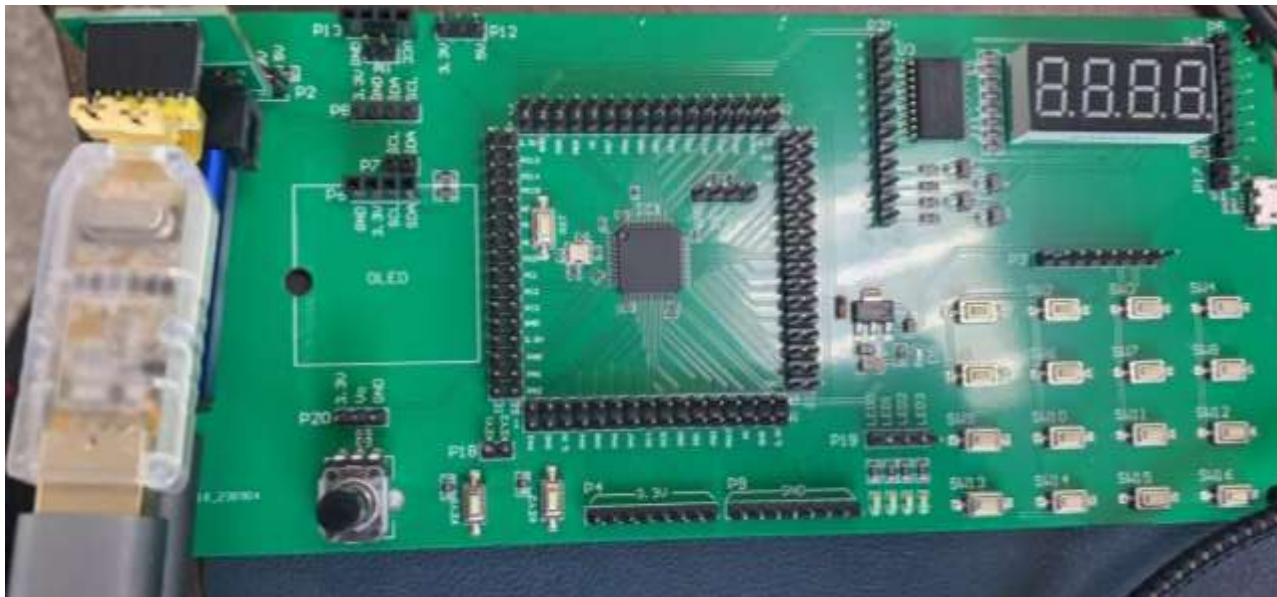


STM32F401RB开发板硬件资源介绍及STM32CubeIDE软件的安装与使用

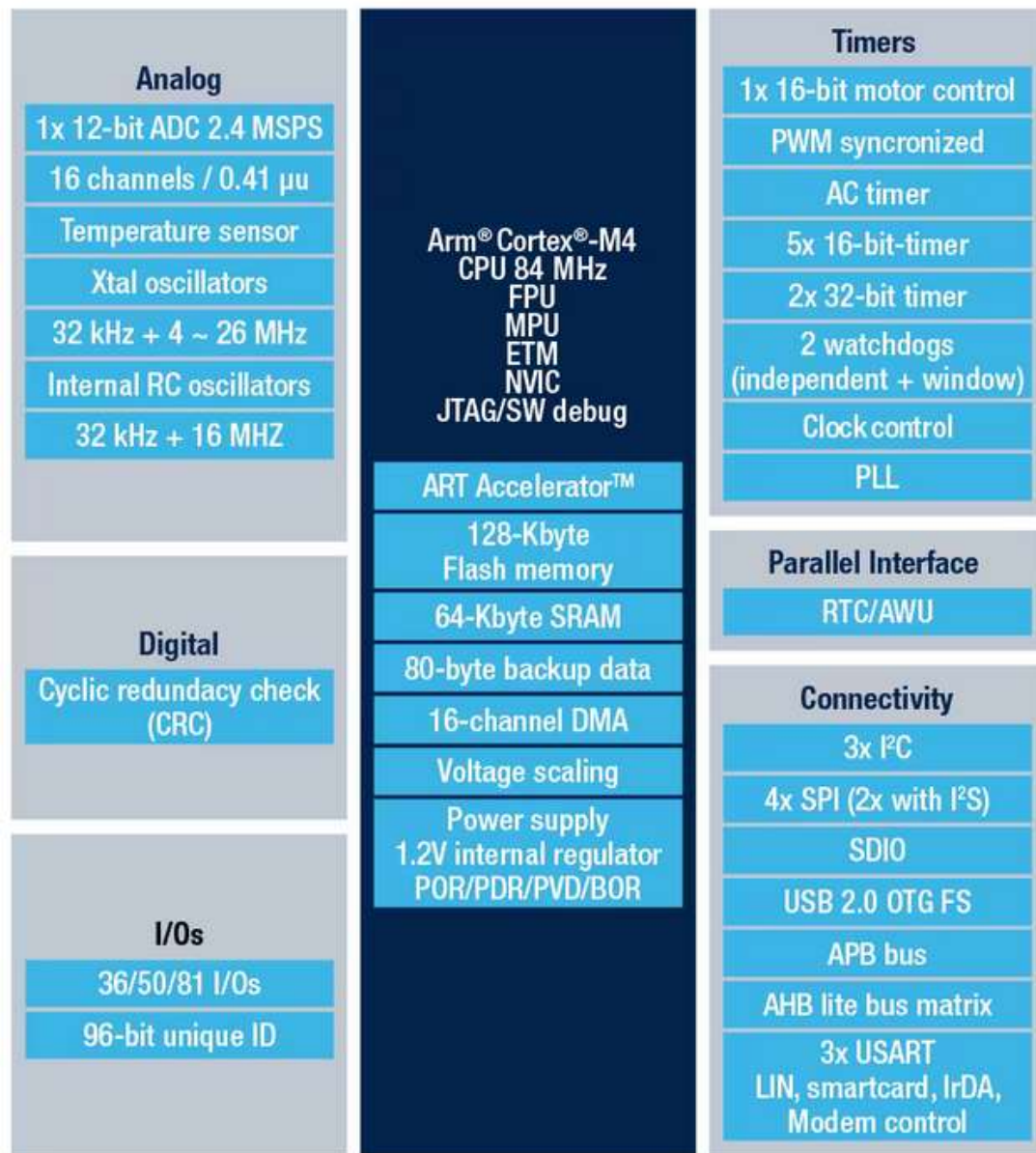




- 使用STM32F401RB微控制器，最高主频84MHz
- 开发板上集成了**调试器**功能，和**USB转串口**工具。由**USB口供电（5V）**。
- STM32F401RB微控制器的所有引脚均引出到排针。
- 具有**简单按键、矩阵键盘、4个LED灯、变阻器、OLED显示屏、4位七段数码管、超声波测距模块、串口通信**等功能，还具有3.3V和GND的接线排、电源指示灯、上拉电阻接线排，方便连接和测试。
- 每个功能使用前，都需要用线将MCU的引脚与功能模块的引脚连起来。



- STM32F401RB
- 32位架构，内部寄存器、数据及总线接口都是32位
 - ARM **Cortex-M4**内核
 - 三级流水线
 - 哈佛总线架构
 - 具有浮点运算单元
 - NVIC嵌套向量中断控制器
 - 具有一个12位ADC可以连接16个模拟通道
 - 具有多种定时器、计数器
 - 具有多种通信接口，如I2C、SPI、SDIO、UART等



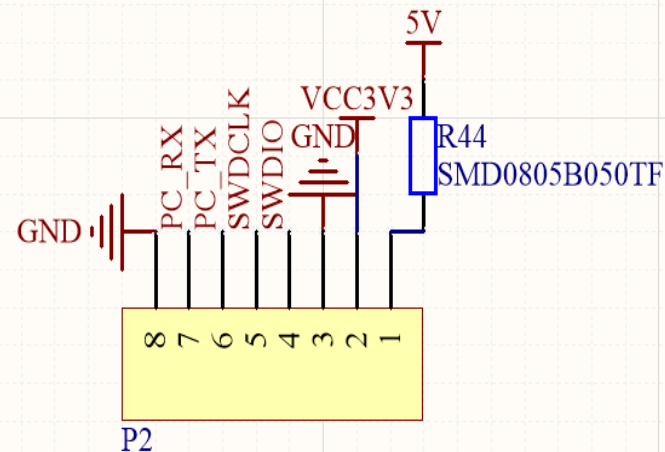
- 让MCU芯片工作起来:

- 供电
- 重置
- 时钟
- 调试器连接
- 简单按键
- LED



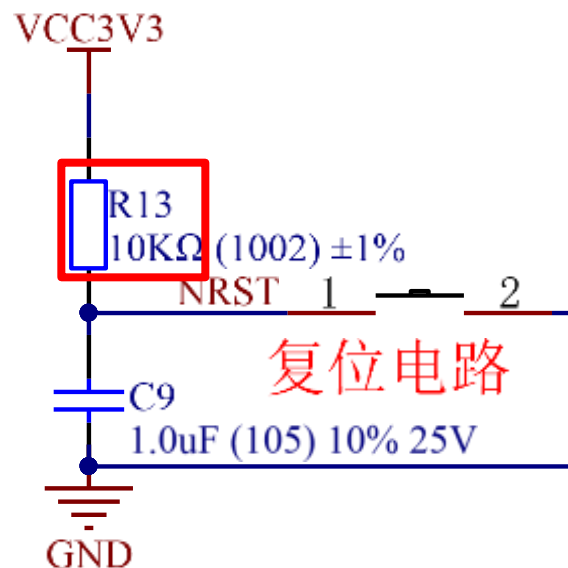
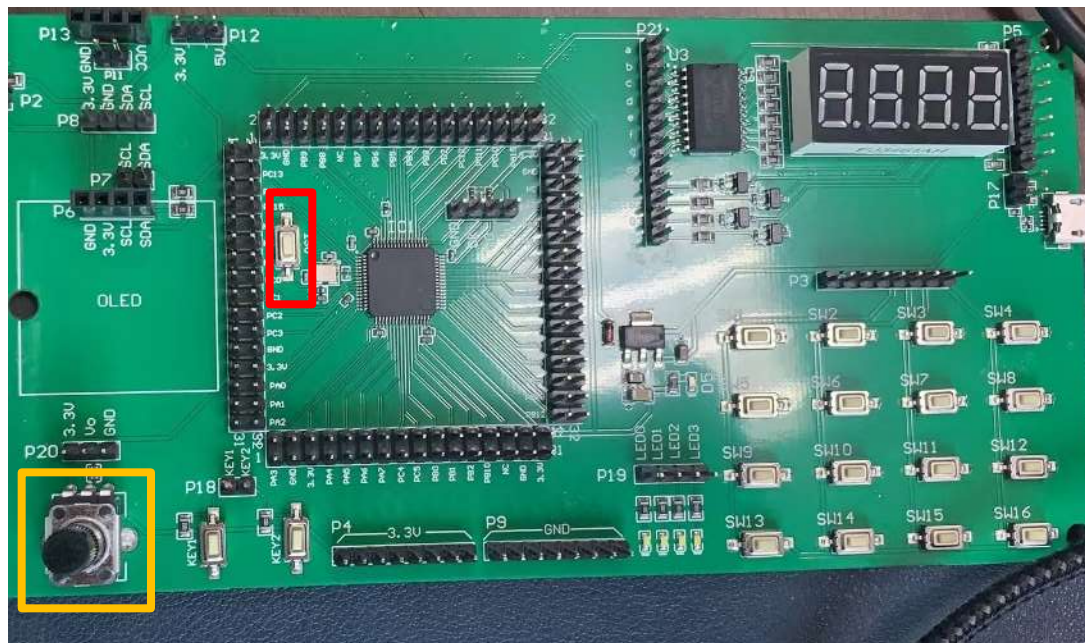
➤ 供电

- P2排针的2号脚，为板子提供3.3V电压，1号脚为板子提供5V电压。其中5V供电有一个自恢复保险丝R44。
- 3.3V电和5V电由调试器提供，因此只要将调试器的USB线插到电脑上，就会有电。



➤ 重置:

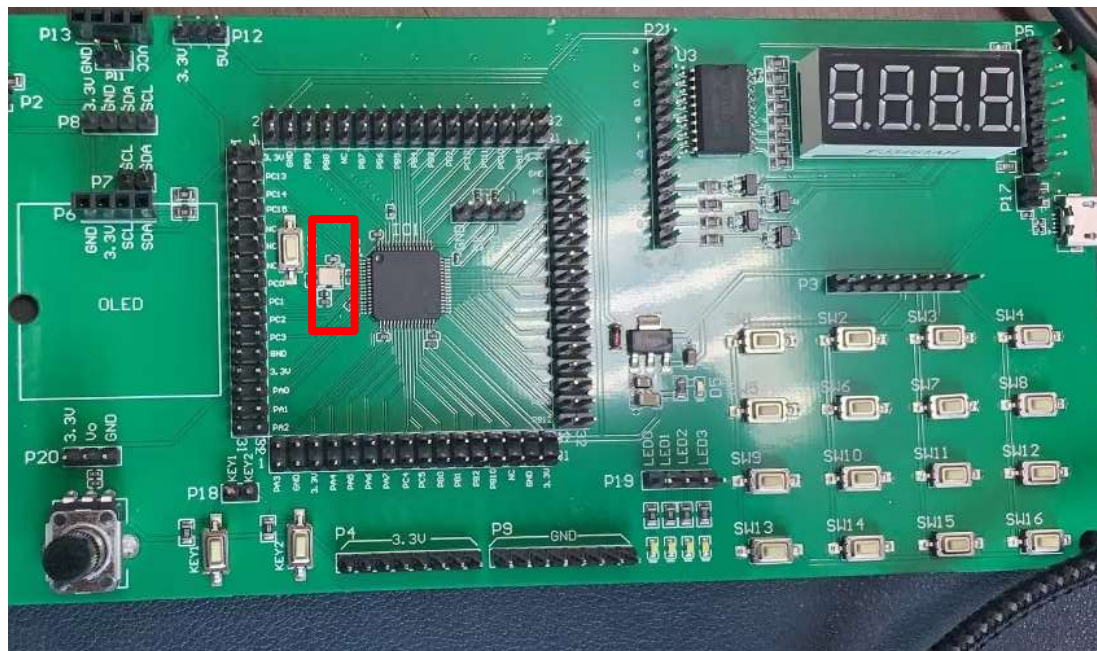
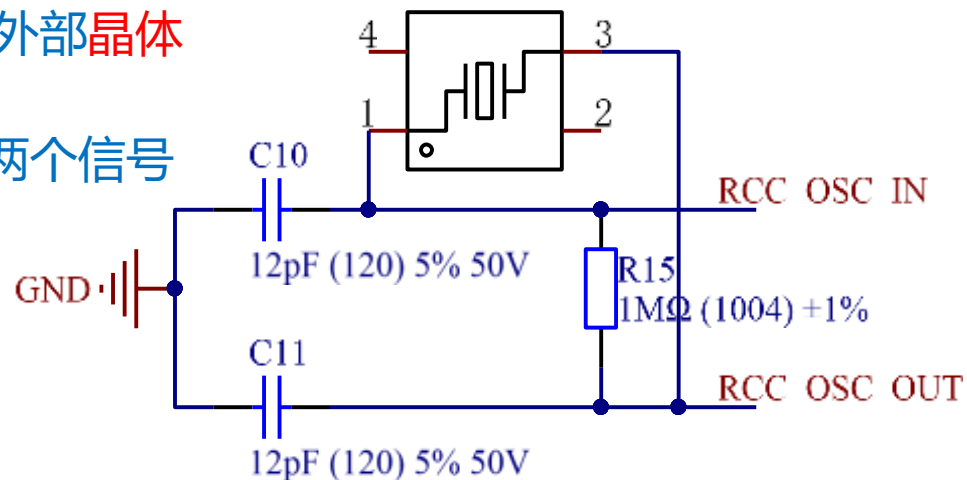
- MCU的NRST引脚经过一个上拉电阻连接到3.3V。
- 有一个复位键，按下后可以将NRST引脚与GND相连。
- 按下复位键后，MCU的NRST引脚为低电平，MCU复位。



NRST	PH1-OSC_OUT
8	NRST

➤ 时钟:

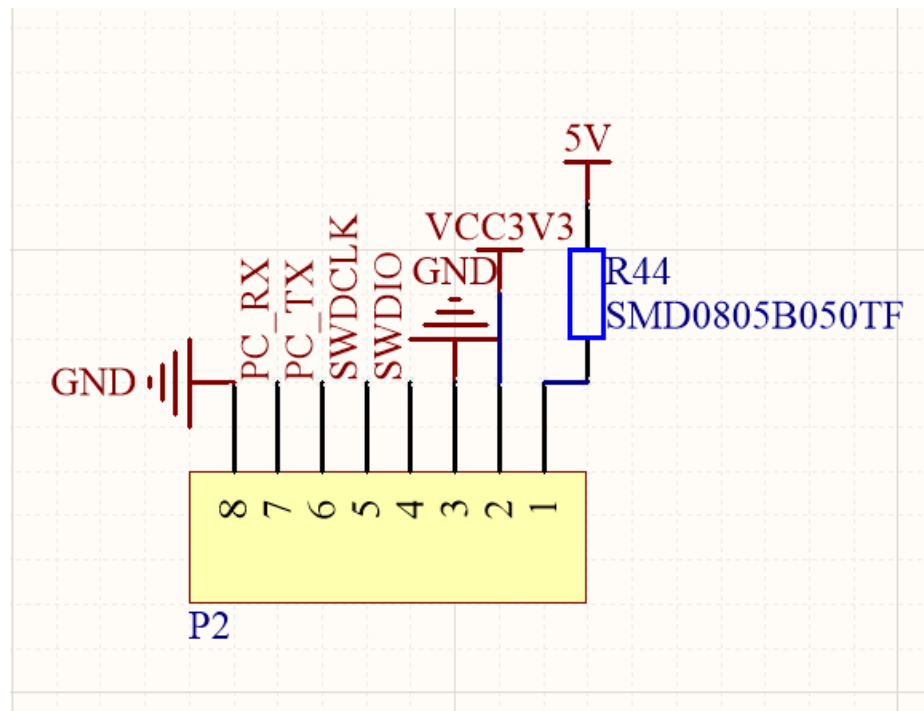
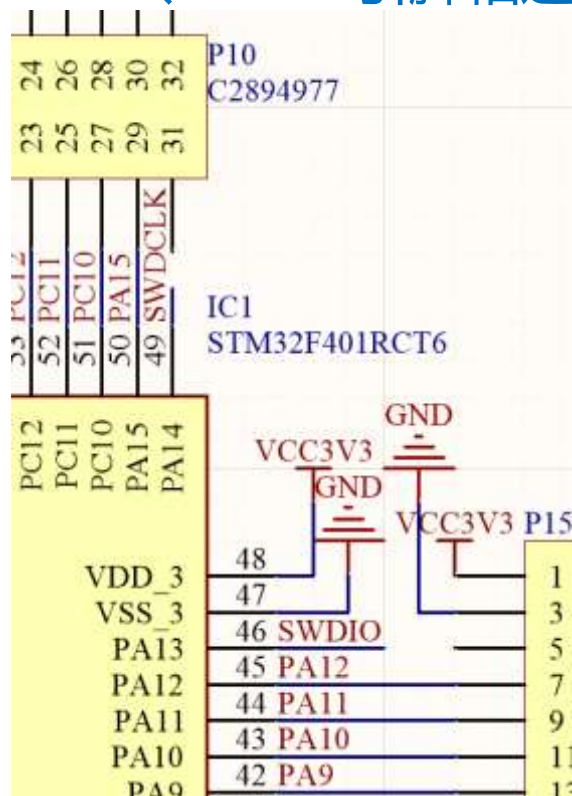
- MCU的PH0和PH1引脚用于连接外部晶体振荡器
- RCC_OSC_IN和RCC_OSC_OUT两个信号连接到了无源整体振荡器的两端



RCC_OSC_IN	5	PC15-OSC32_OUT
RCC_OSC_OUT	6	PH0-OSC_IN
		PH1-OSC_OUT

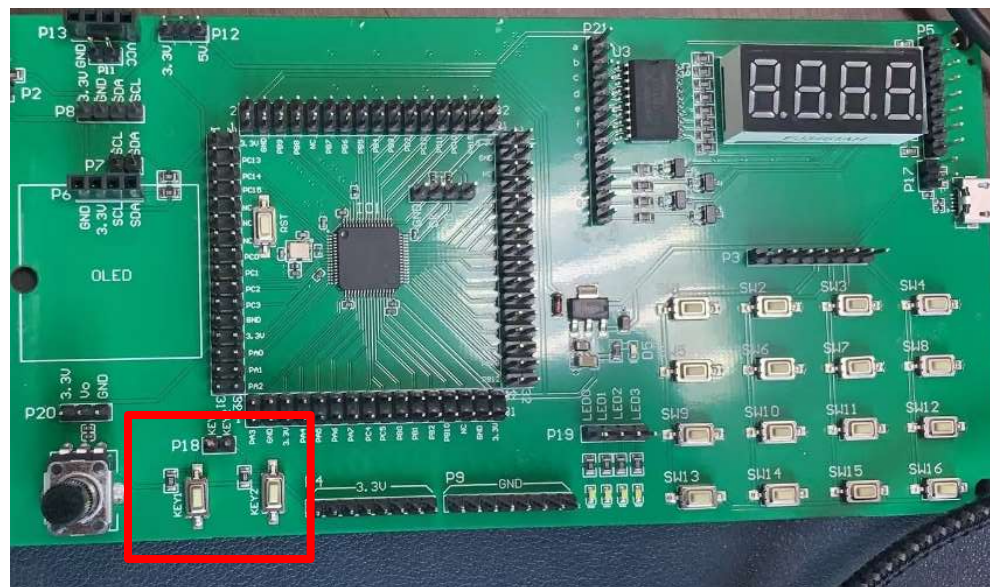
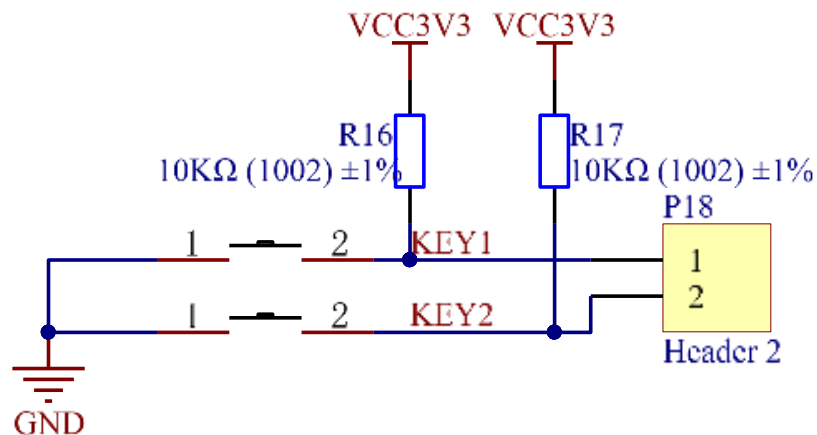
➤ 调试器连接

- 将USB口插到电脑上，就可以将电脑与调试器连接起来
- 使用**SWDCLK**和**SWDIO**两个引脚进行调试，调试器的这两个引脚与MCU的**PA13**、**PA14**引脚相连，用于调试。



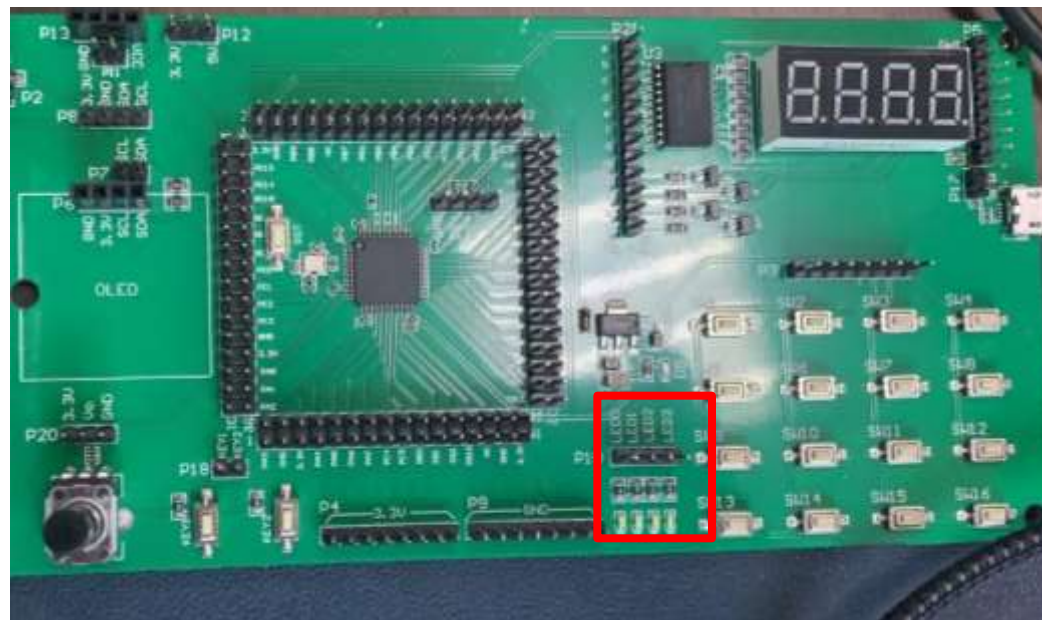
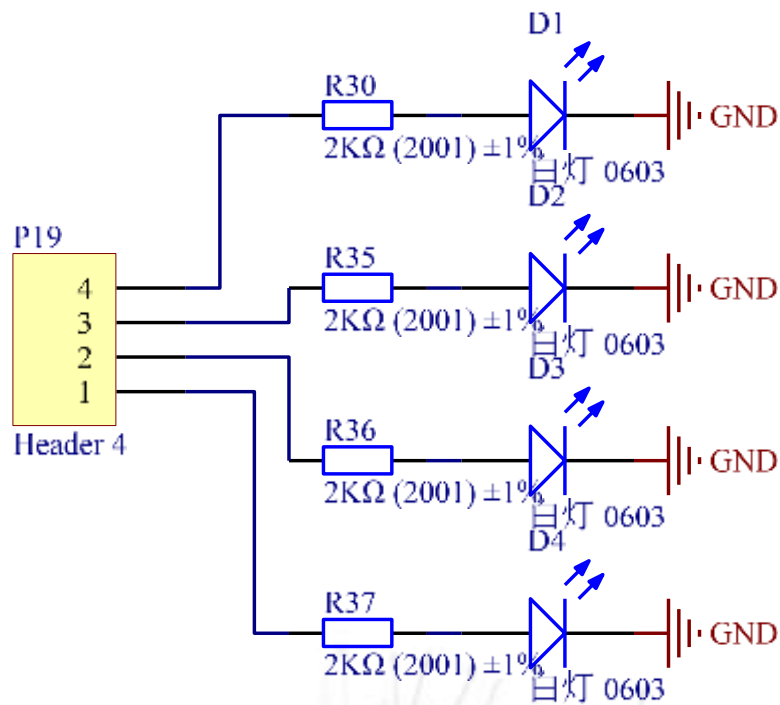
➤ 简单按键

- 简单按键的一端接地，另一端经过一个上拉电阻接到3.3V。
- 按键没有按下时，KEY1的电平是3.3V。
- 按键按下后，KEY1的电平就是0V。
- 要与MCU的引脚连接后才能使用



➤ LED灯:

- LED灯是一个发光二极管。
- 其阴极接地，阳极经过一个电阻然后接到控制信号。
- 当控制信号为3.3V时，灯亮。
- 根据LED灯的压降和电阻确定LED灯的电流。



STM32CubeIDE软件的 安装



➤ STM32CubeIDE软件：ST提供的集成开发环境

➤ 安装要求：

- 安装程序的路径中不可有中文和特殊字符，建议新建一个管理员账户，用户名必须为纯英文，不要有空格和特殊字符。新建管理员账户的方法见[链接](#)。
- 先上网
- 关闭杀毒软件
- 安装到C盘，确保C盘有足够的空间 (>10GB)

安装程序下载地址（校内网）：

<http://10.193.37.86:5000/d/s/v5Qll3XLLHFsSivOXBt5h7N6PyUQJYKd/Es1SXKYhewzilqlvtN3SajqKPc1mpl85-Y7ugi6lWvwo>

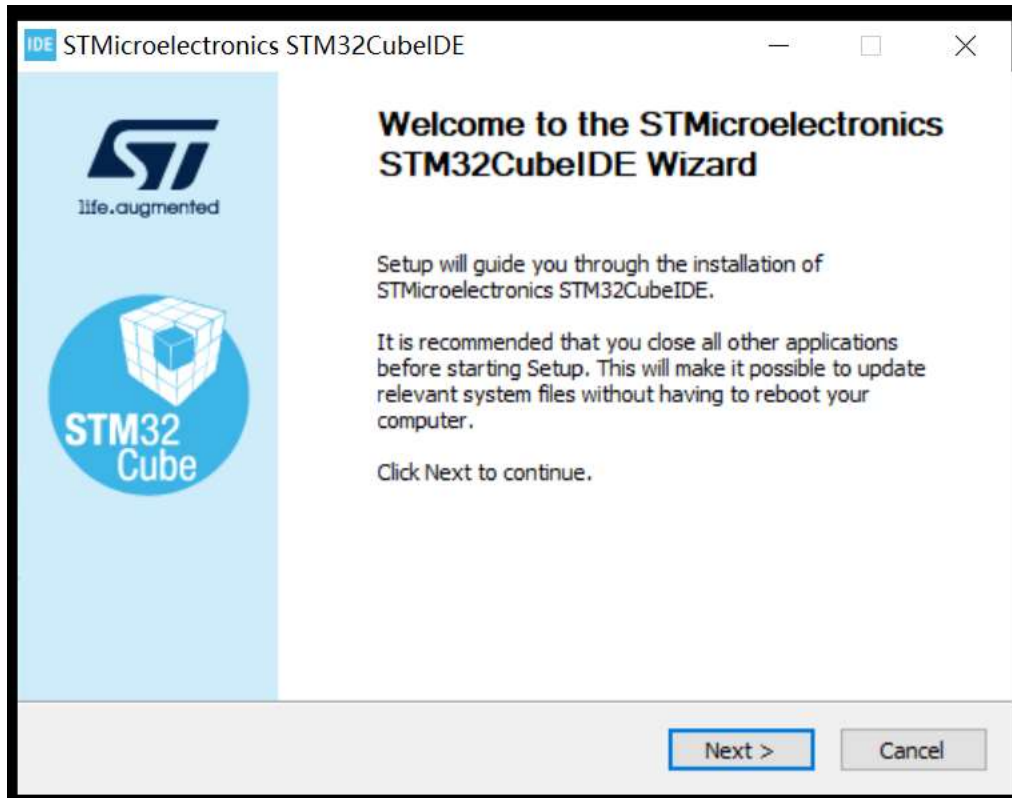
安装程序下载地址（校外网）：

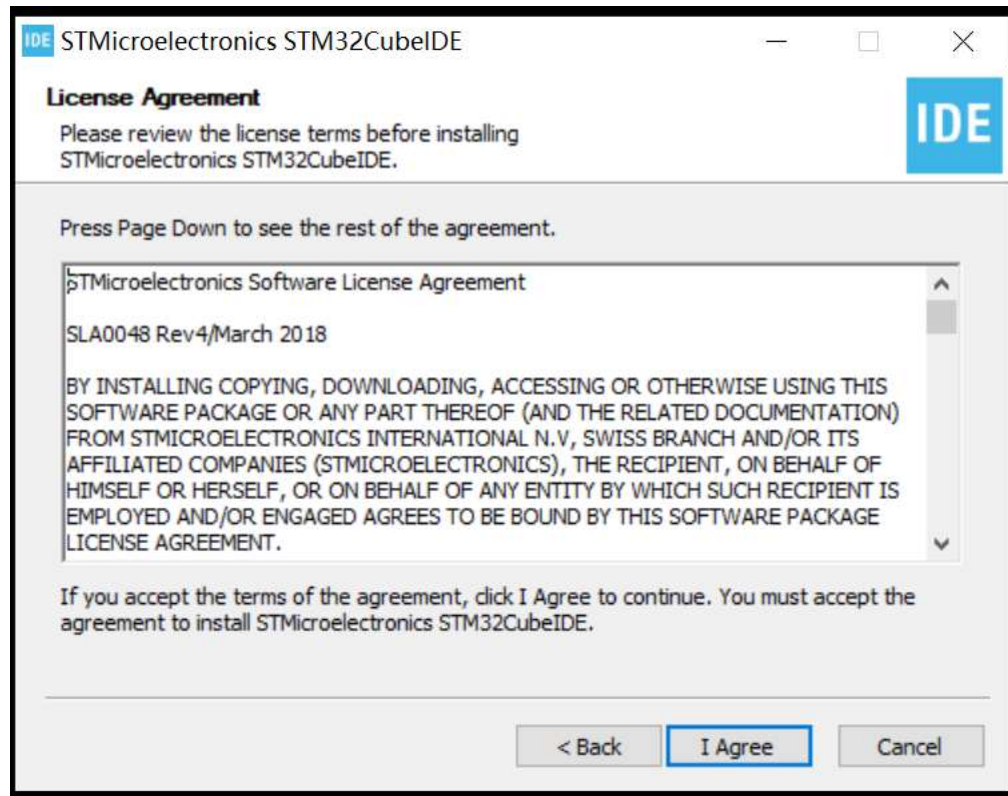
<http://ee.shenzhuo.vip:36644/d/s/v5Qll3XLLHFsSivOXBt5h7N6PyUQJYKd/Es1SXKYhewzilqlvtN3SajqKPc1mpl85-Y7ugi6lWvwo>



双击打开STM32CubeIDE安装程序

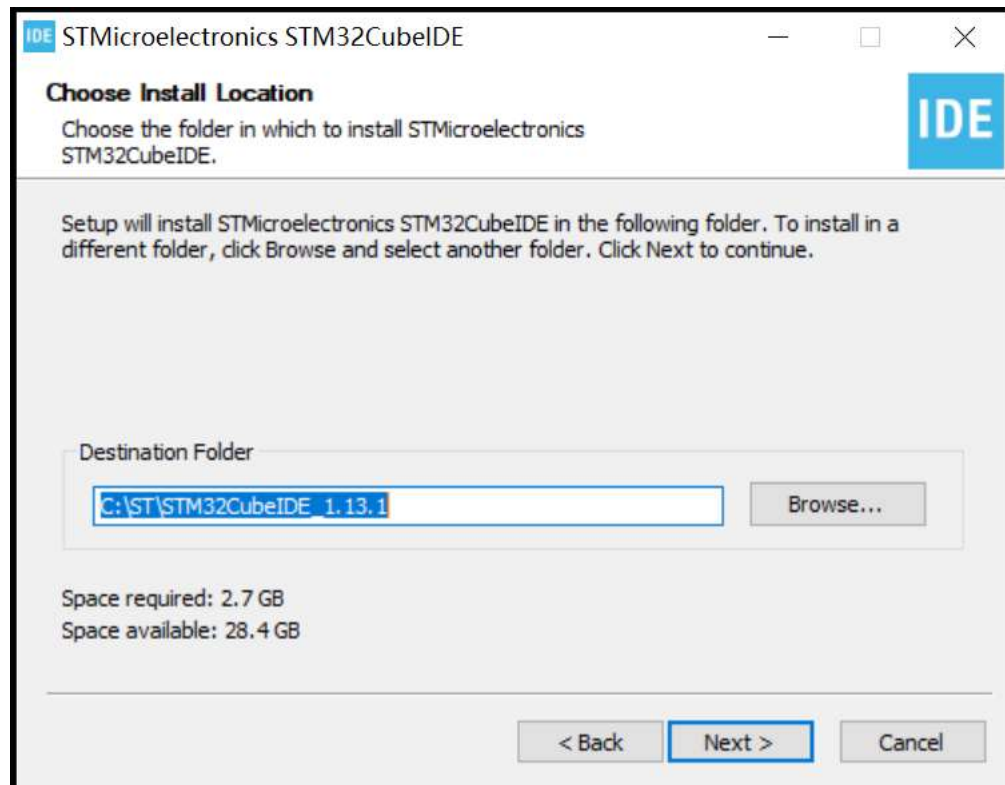




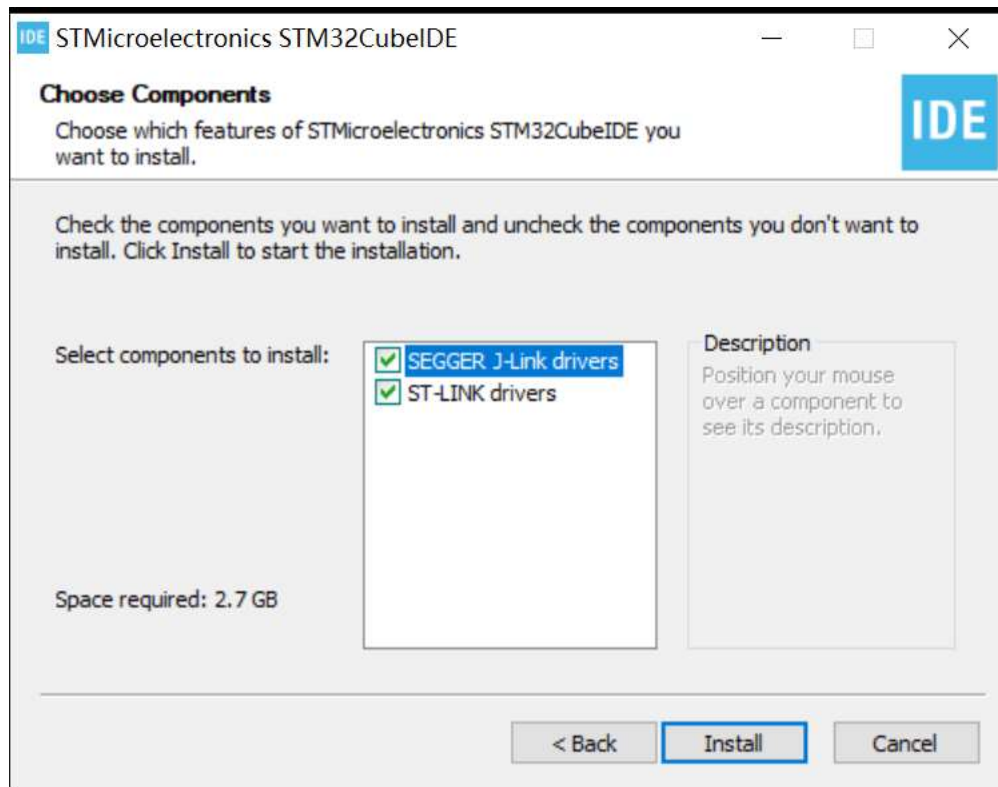


点I Agree

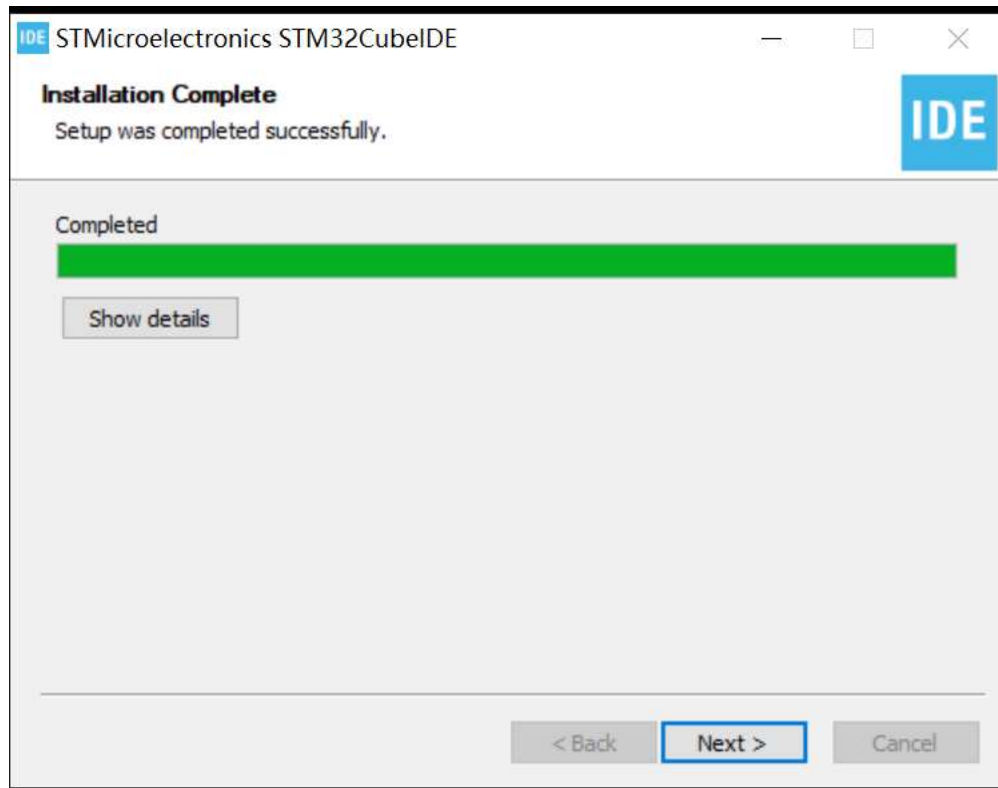




选择默认安装目录，点Next

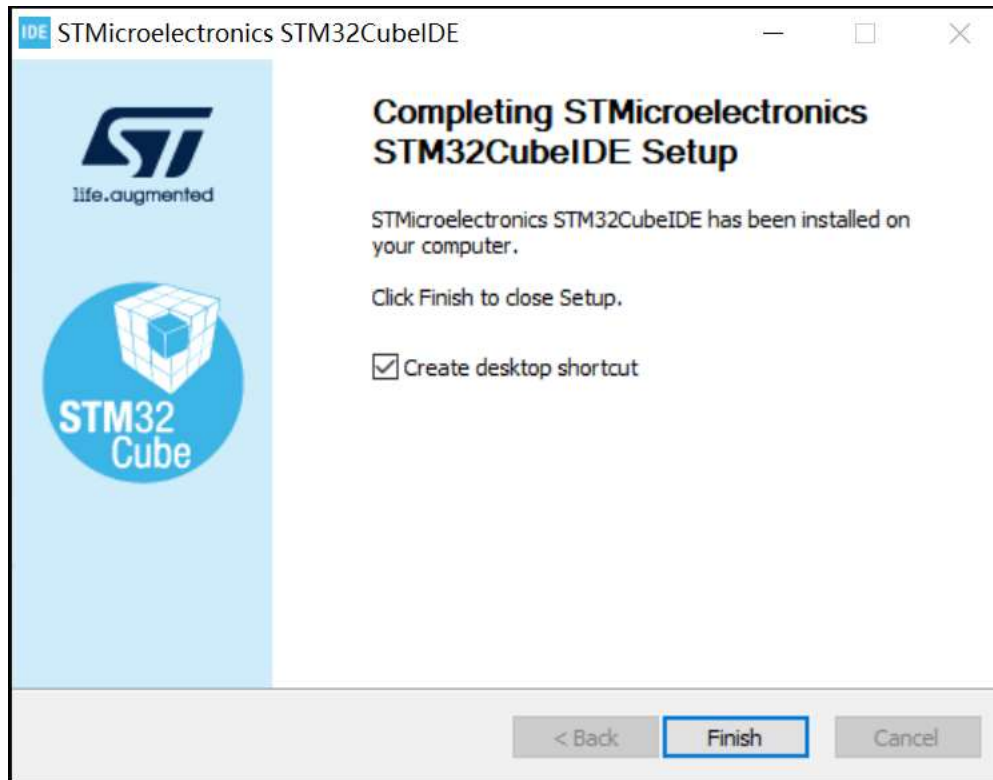


勾选以上两个选项，点Install



安装完成后，点**Next**





点Finish



STM32CubeIDE软件的使用

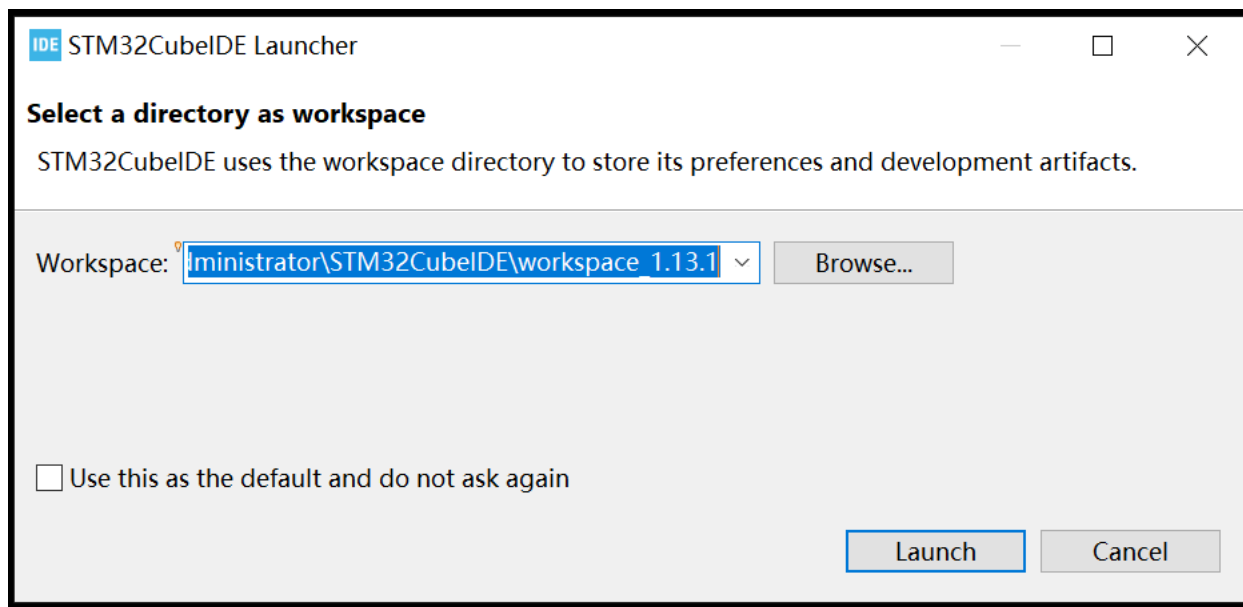




双击图标，打开STM32CubeIDE。

选择工作空间目录。

工作空间是存储程序工程及其代码的文件夹，地址采用全英文字符。



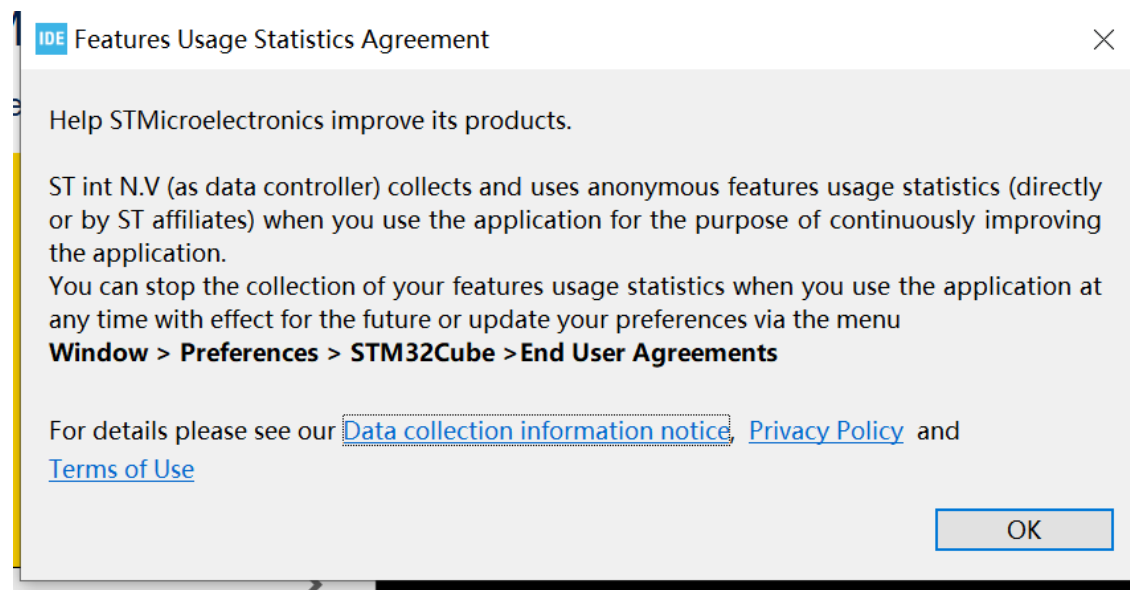
点击Launch



双击图标，打开STM32CubeIDE。

选择工作空间目录。

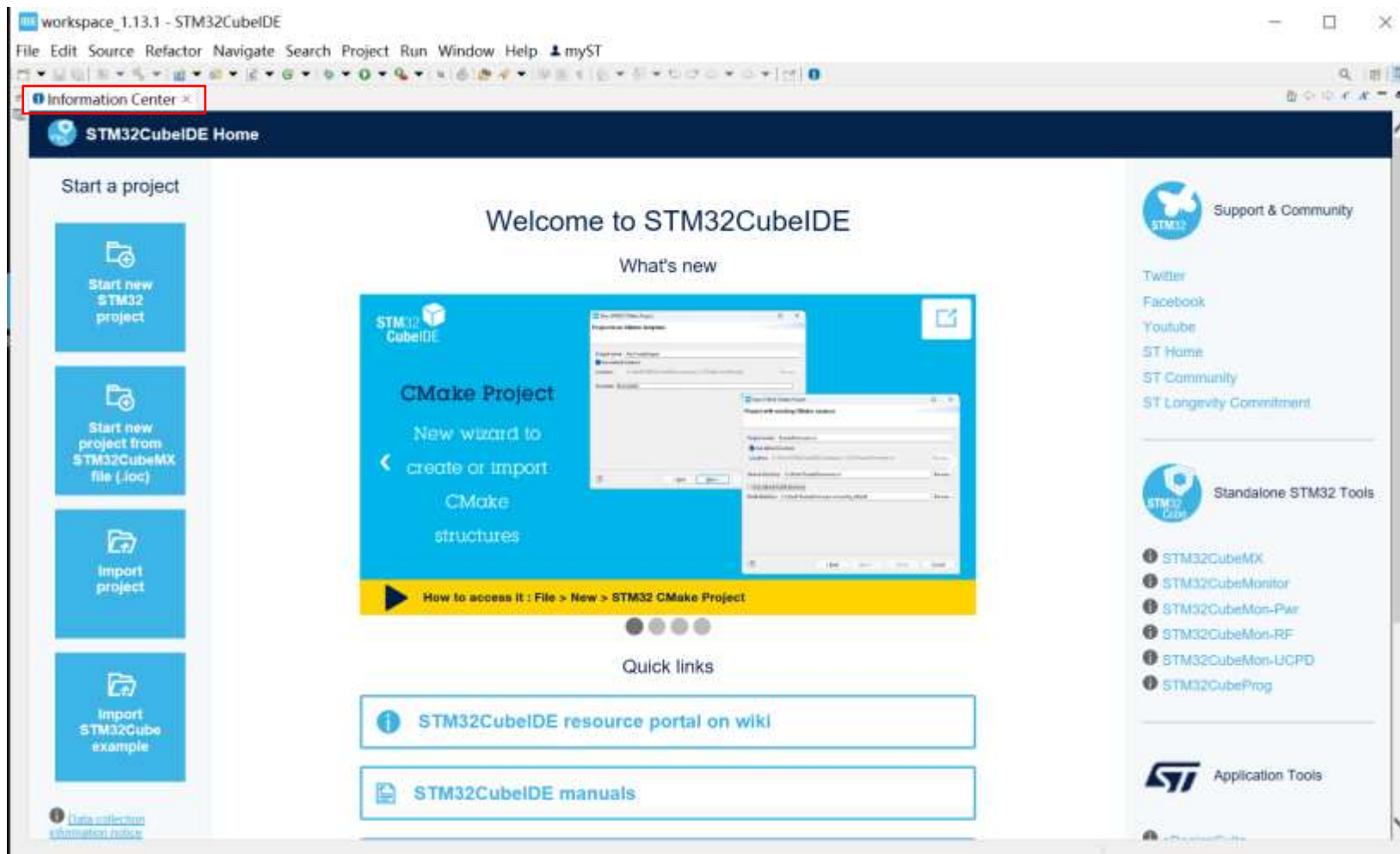
工作空间是存储程序工程及其代码的文件夹，地址采用全英文字符。



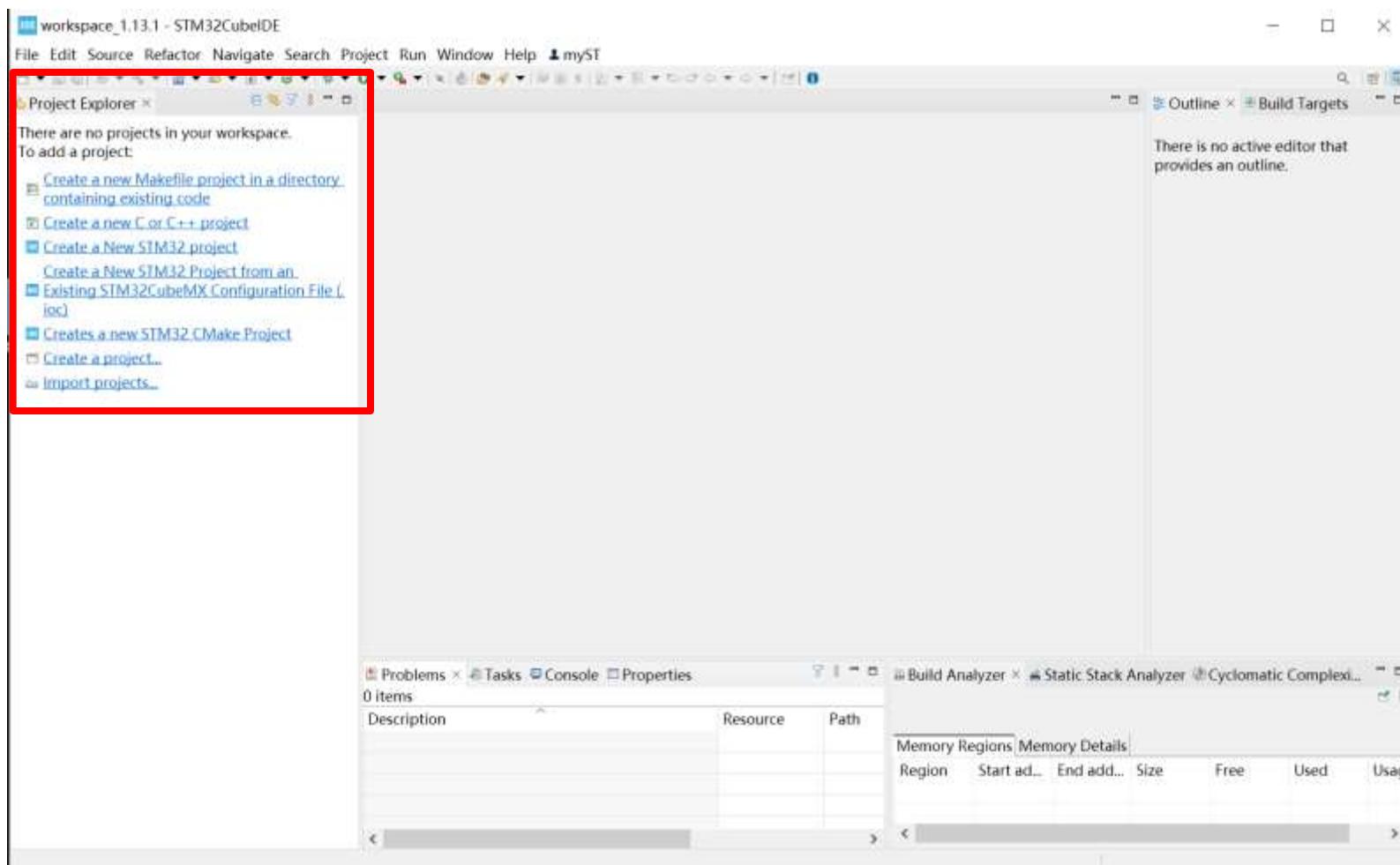
第一次打开时，如果弹出对话框点OK



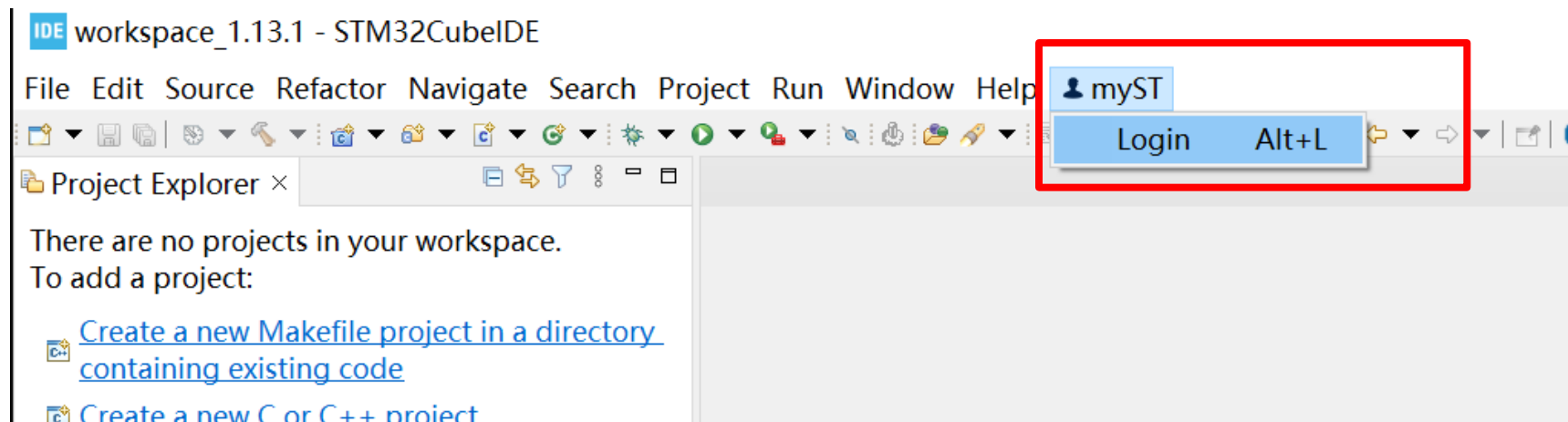
第一次打开IDE后，关闭Information Center选项卡



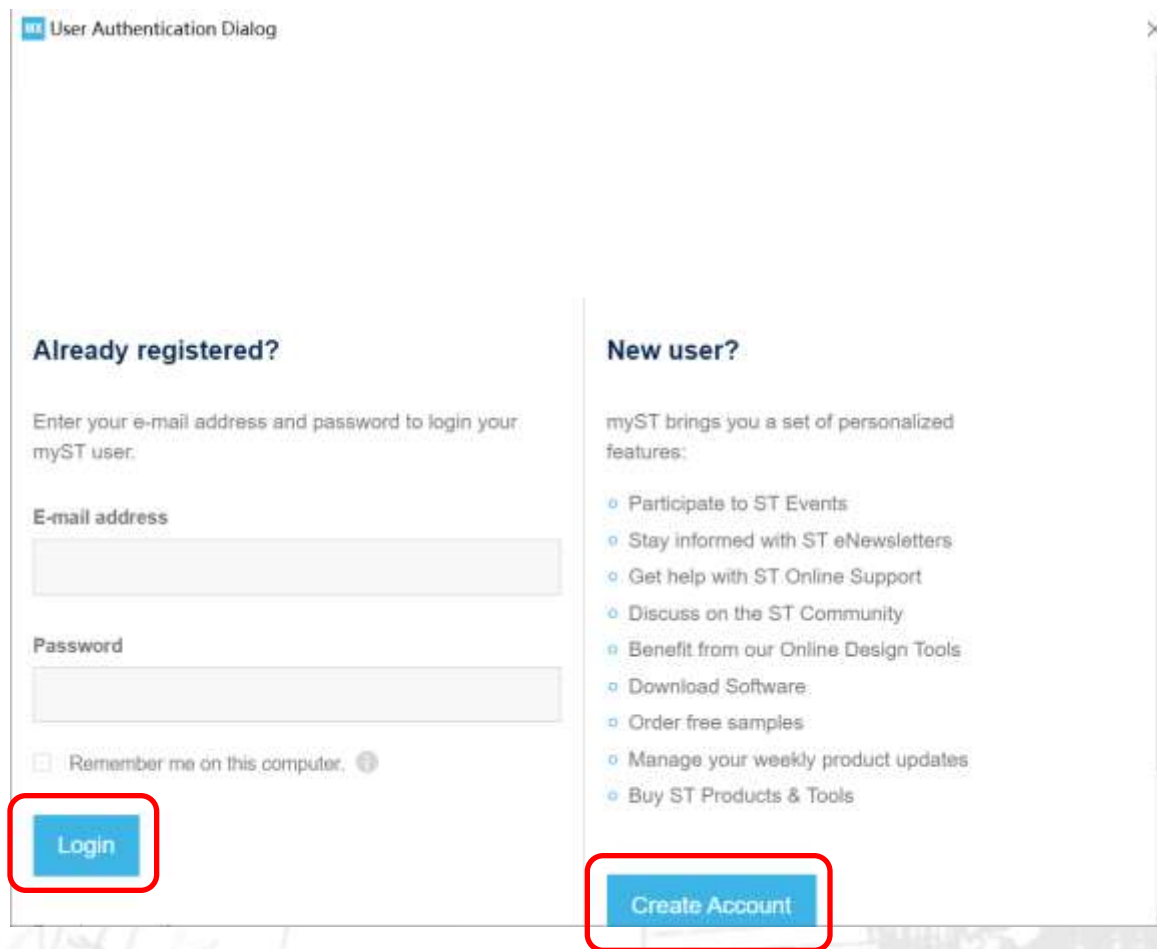
左侧会自动弹，Project Explorer选项卡



点击mySt->Login登录ST账号，如果没有ST账号可以点击Creat Account注册



如果注册过myST账号，则登录，如果没有注册过myST账号，则点击Create Account 注册。



The image shows a 'User Authentication Dialog' window with two main sections: 'Already registered?' and 'New user?'. The 'Already registered?' section includes a text prompt 'Enter your e-mail address and password to login your myST user.', followed by input fields for 'E-mail address' and 'Password'. Below these is a checkbox labeled 'Remember me on this computer, ⓘ'. A blue 'Login' button is at the bottom of this section. The 'New user?' section has a heading 'New user?' and a list of features: 'myST brings you a set of personalized features:'. The features are listed with blue circular icons: 'Participate to ST Events', 'Stay informed with ST eNewsletters', 'Get help with ST Online Support', 'Discuss on the ST Community', 'Benefit from our Online Design Tools', 'Download Software', 'Order free samples', 'Manage your weekly product updates', and 'Buy ST Products & Tools'. A blue 'Create Account' button is at the bottom of this section. Both the 'Login' and 'Create Account' buttons are highlighted with red rectangles.

User Authentication Dialog

Already registered?

Enter your e-mail address and password to login your myST user.

E-mail address

Password

☐ Remember me on this computer, ⓘ

Login

New user?

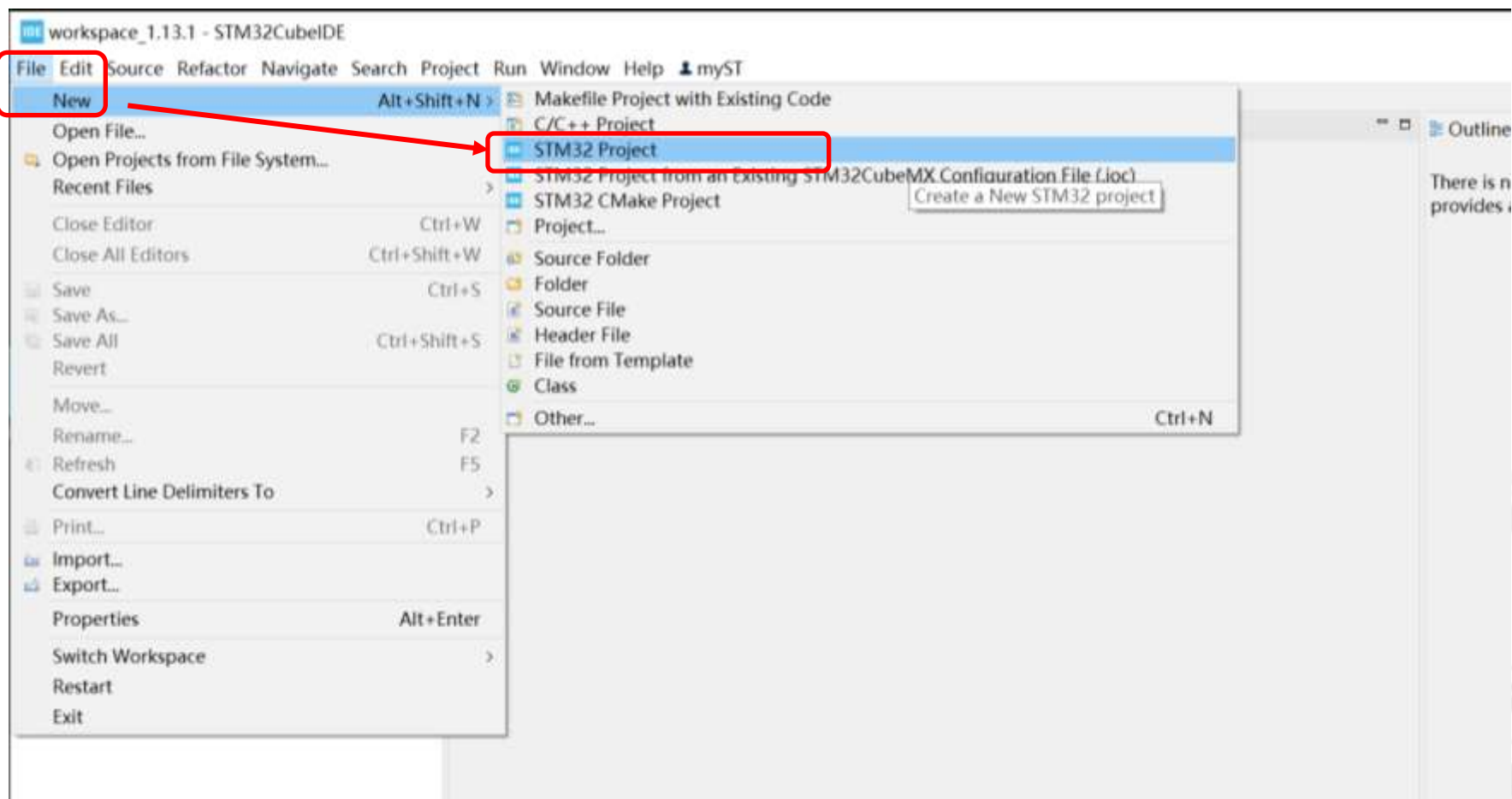
myST brings you a set of personalized features:

- Participate to ST Events
- Stay informed with ST eNewsletters
- Get help with ST Online Support
- Discuss on the ST Community
- Benefit from our Online Design Tools
- Download Software
- Order free samples
- Manage your weekly product updates
- Buy ST Products & Tools

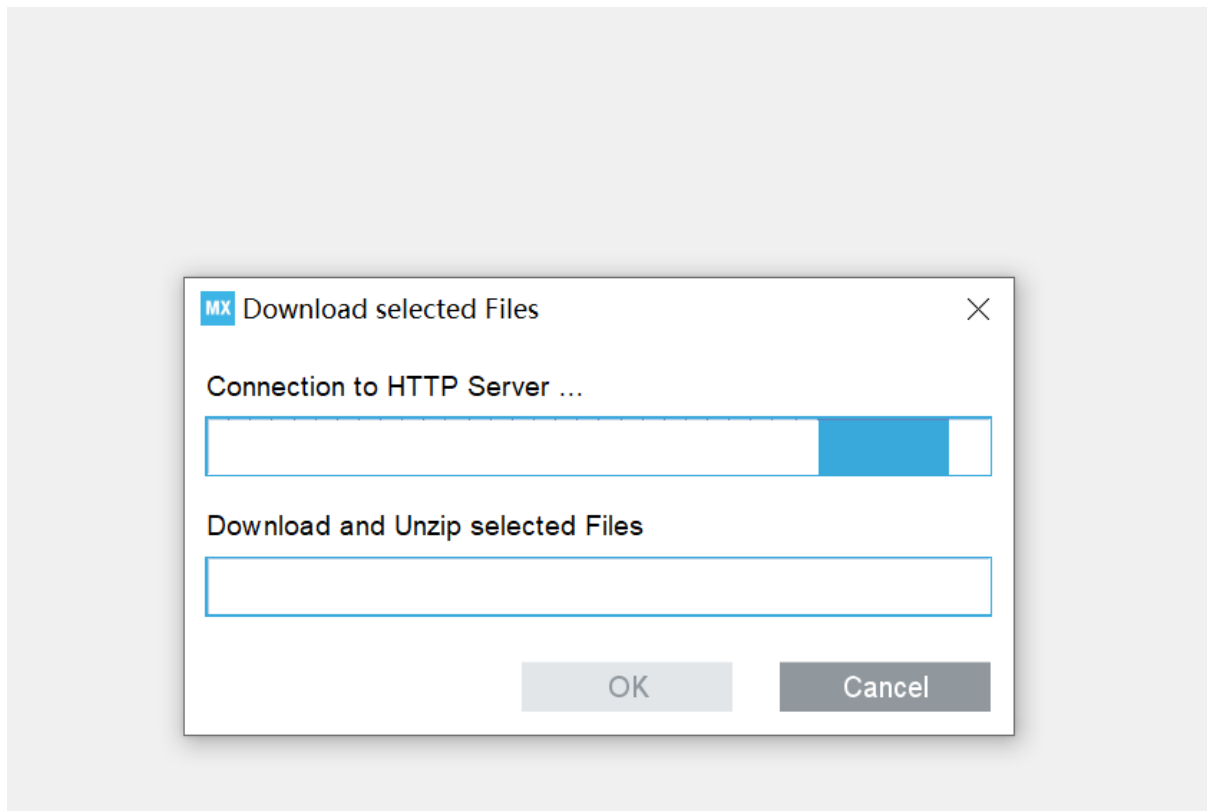
Create Account



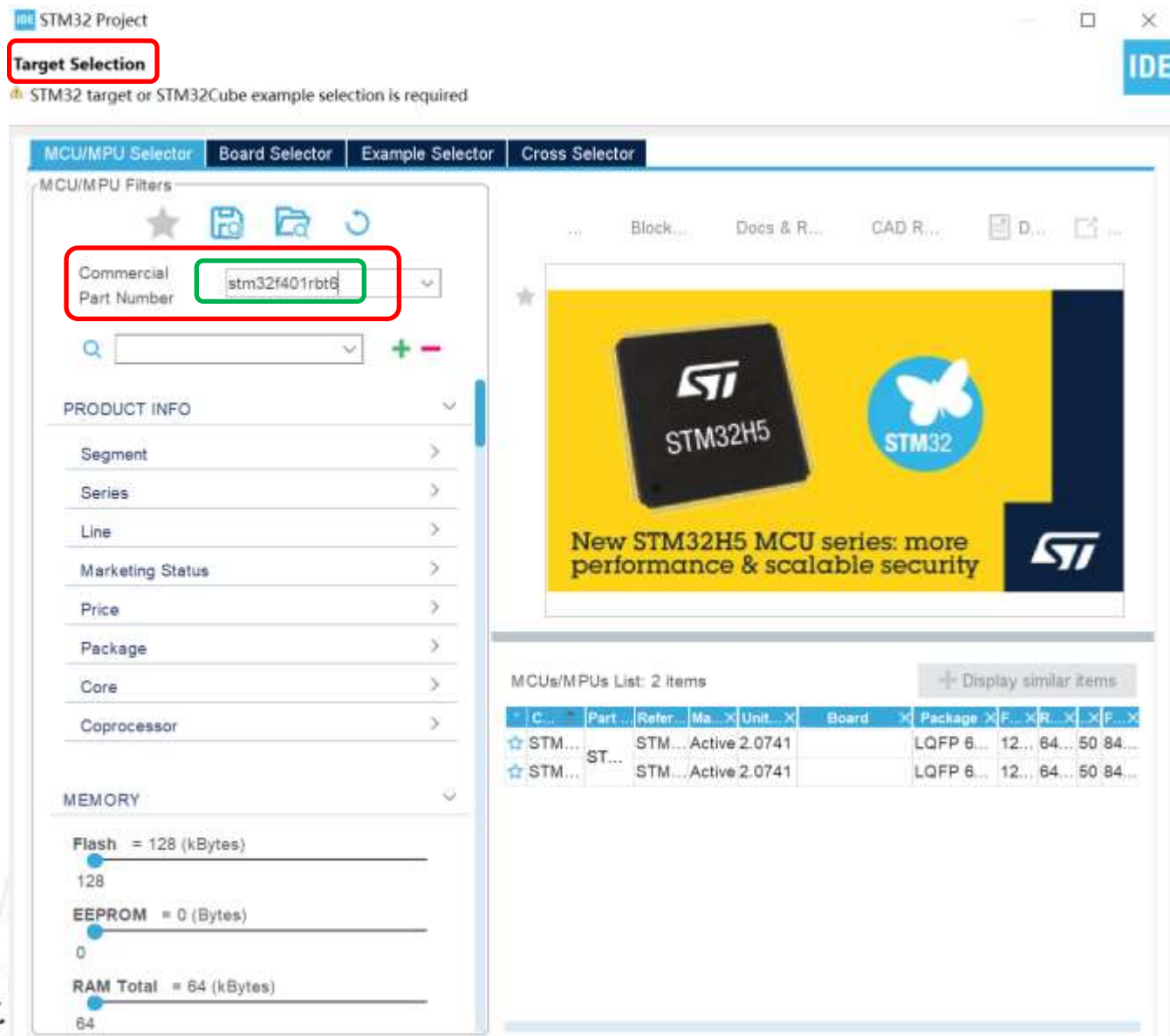
登录完成后，点击左上角的File->New->STM32 Project



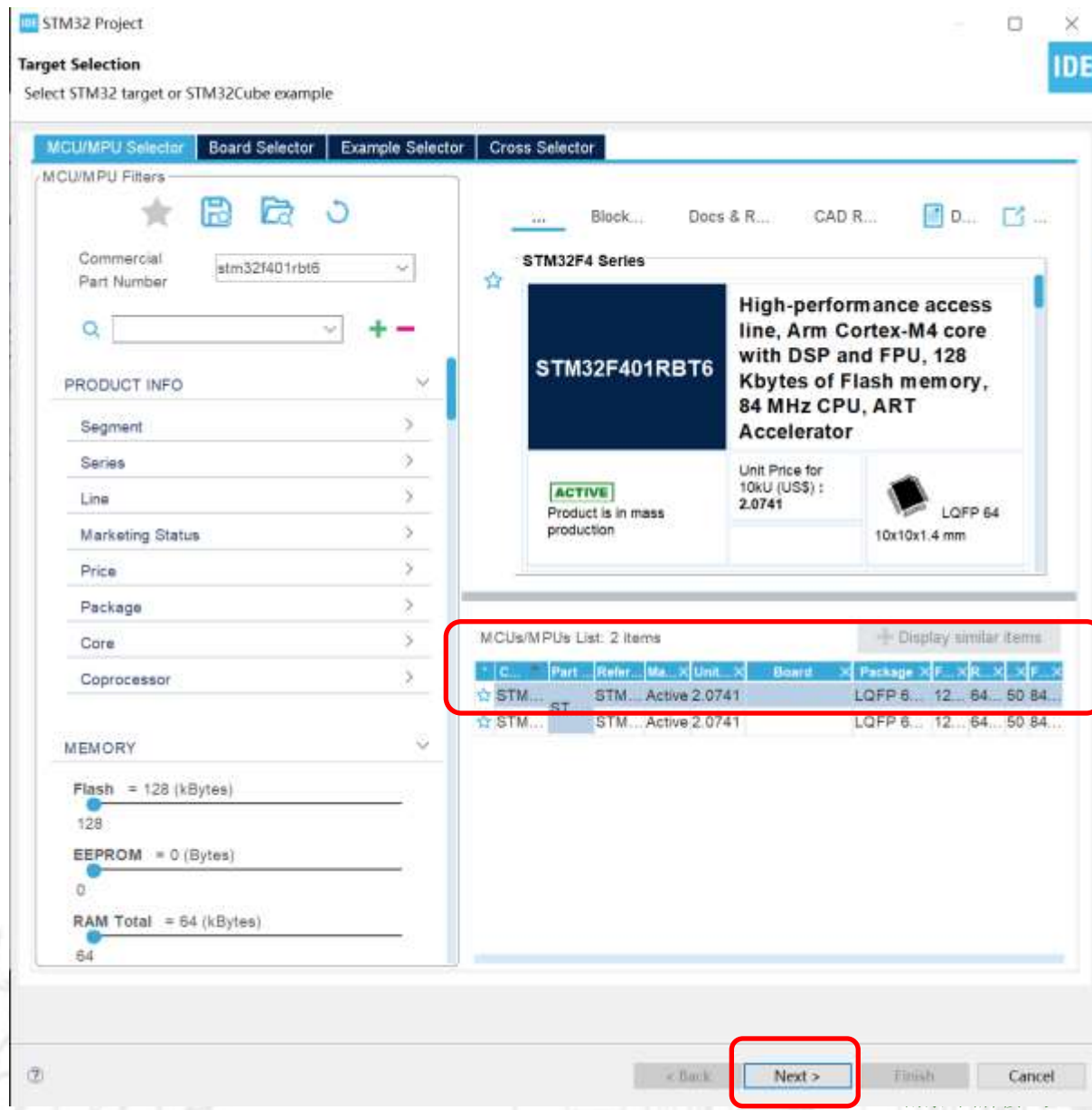
保持联网，等待下载完成



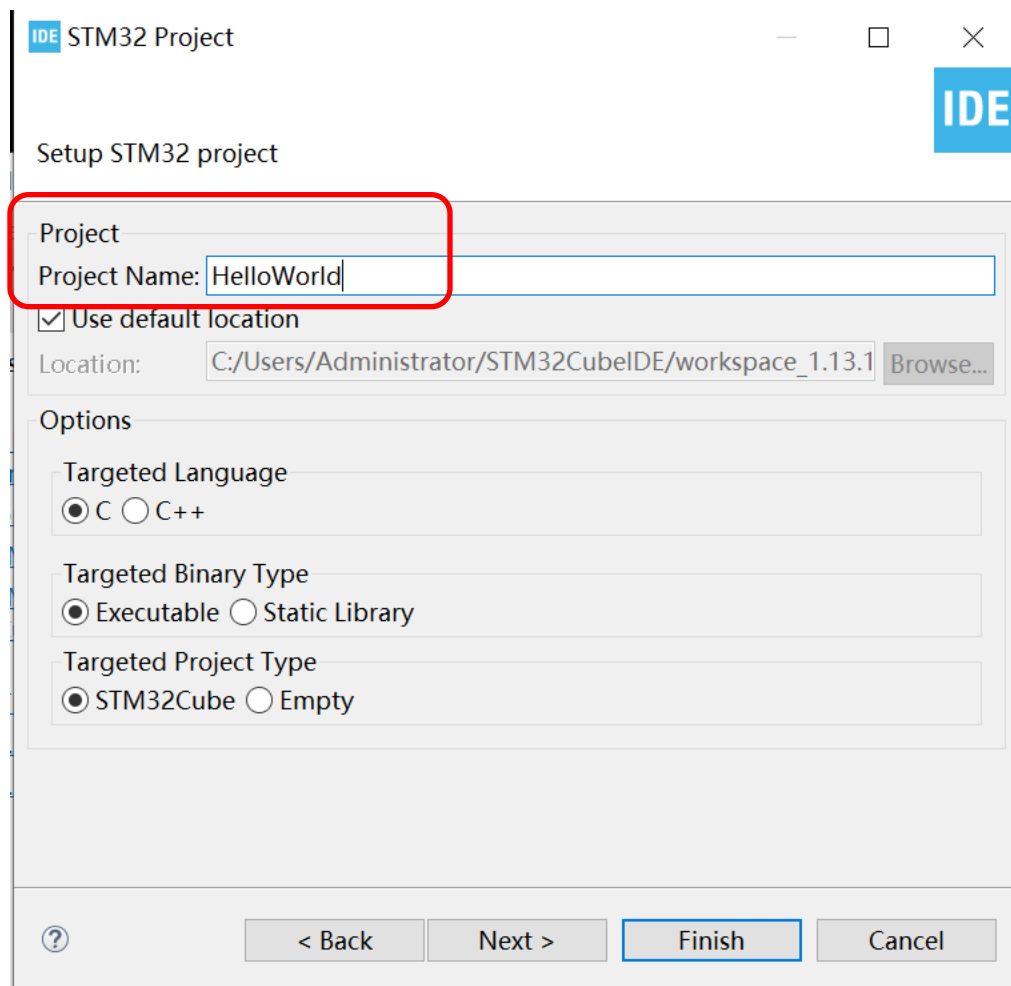
在弹出的Target Selection对话框里，左上角Commercial Part Number输入框里输入stm32f401rct6



鼠标左键点击选择右下角选项框里的第一条，然后点右下角的Next:



在弹出的对话框里，输入Project name，如“HelloWorld”，其他保持默认。



IDE STM32 Project

Setup STM32 project

Project

Project Name: HelloWorld

☒ Use default location

Location: C:/Users/Administrator/STM32CubeIDE/workspace_1.13.1 Browse...

Options

Targeted Language

☒ C ☐ C++

Targeted Binary Type

☒ Executable ☐ Static Library

Targeted Project Type

☒ STM32Cube ☐ Empty

? < Back Next > Finish Cancel

点击Next



STM32 Project

Firmware Library Package Setup

Setup STM32 target's firmware

Target and Firmware Package

Target Reference:

STM32F401RBTx

Firmware Package Name and Version: STM32Cube FW_F4

V1.27.1

Firmware and Software Package Repository

Location:

C:\Users\Administrator\STM32Cube\Repository

See '[Firmware Updater](#)' for settings related to package installation

Code Generator Options

- ☐ Add necessary library files as reference in the toolchain project configuration file
- ☐ Copy all used libraries into the project folder
- ☒ Copy only the necessary library files



< Back

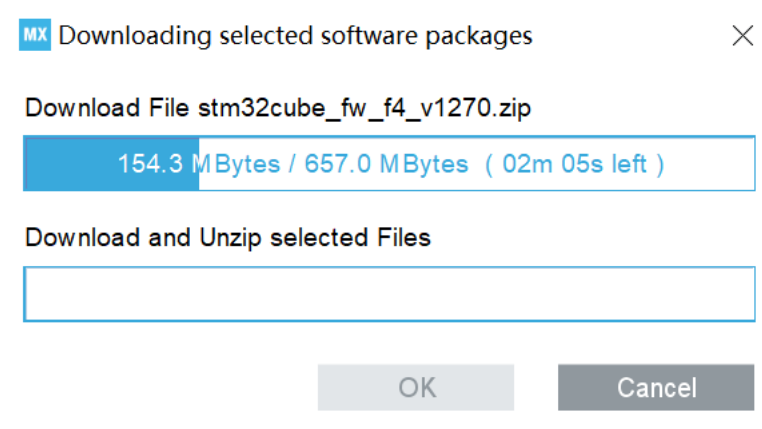
Next >

Finish

Cancel

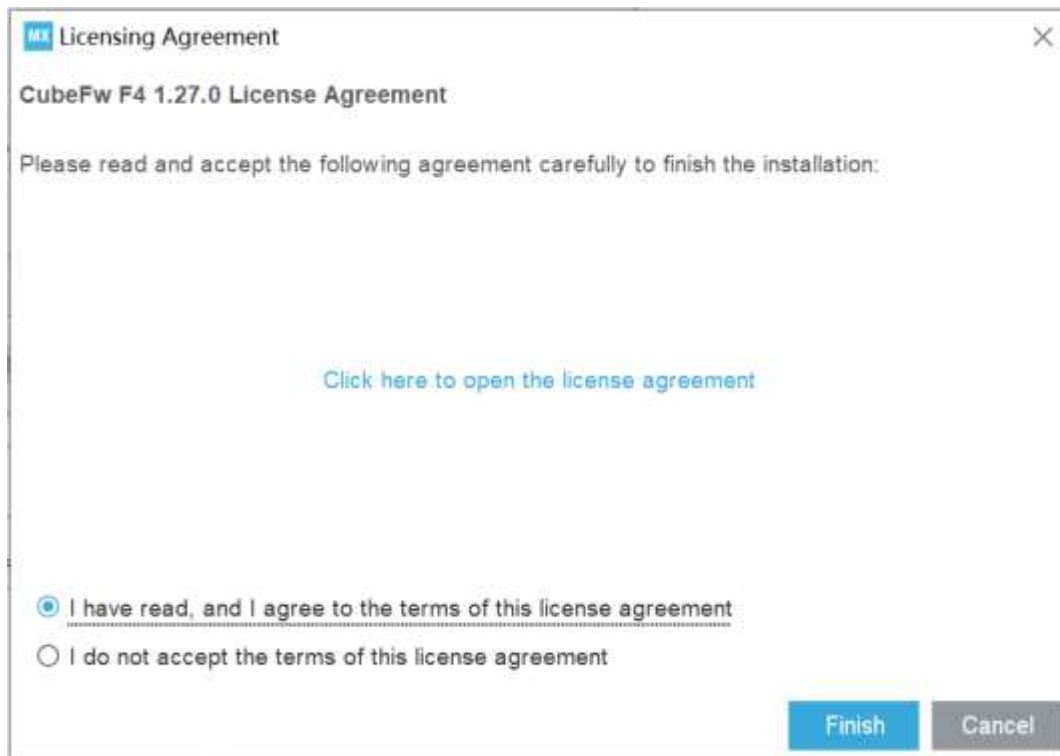
点击Finish





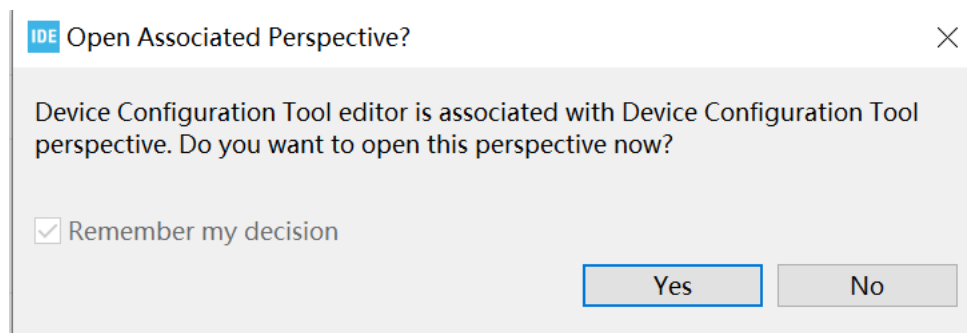
等待下载完成





如果弹出的对话框，选I have read ...，然后点击Finish

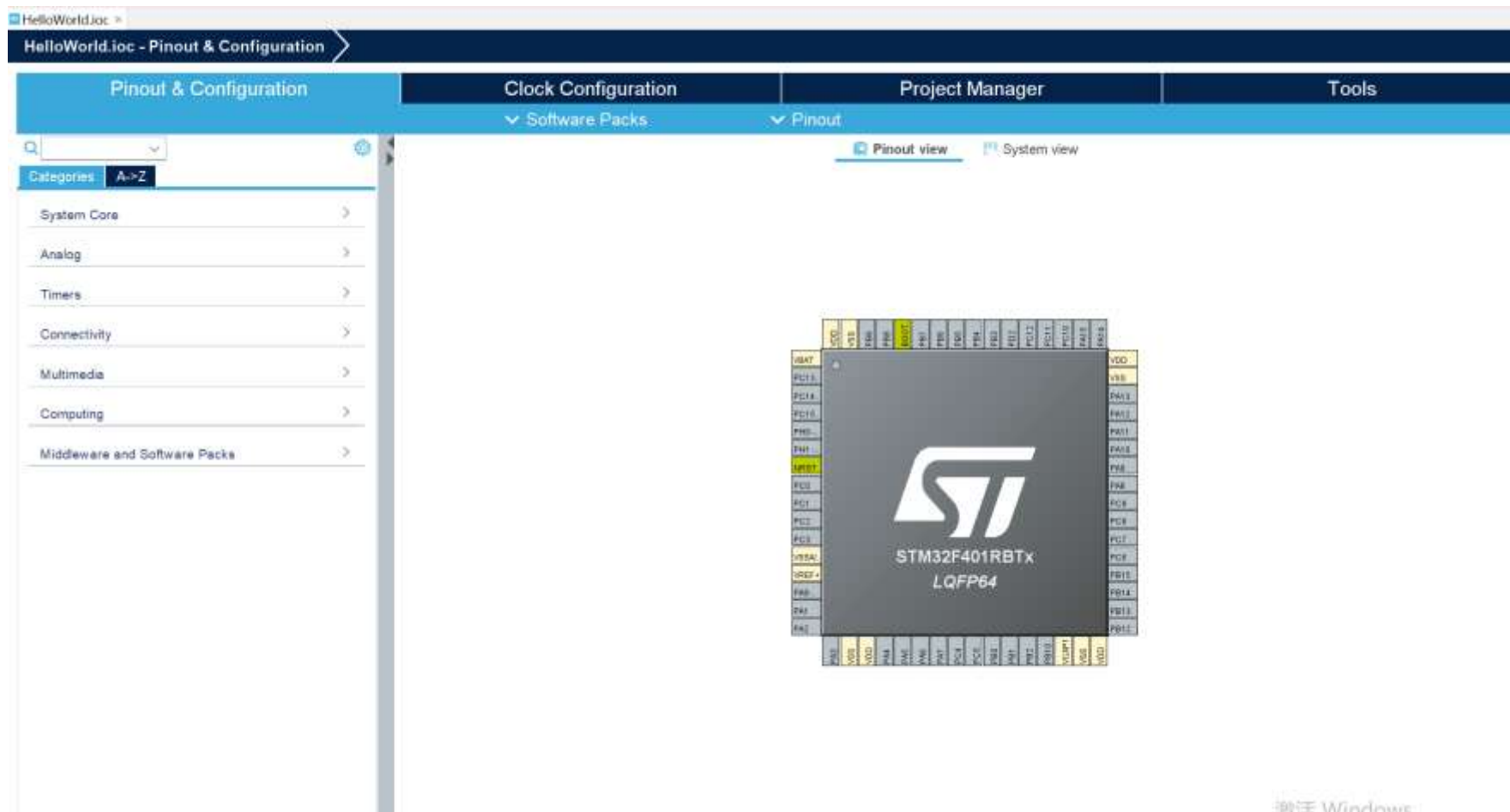




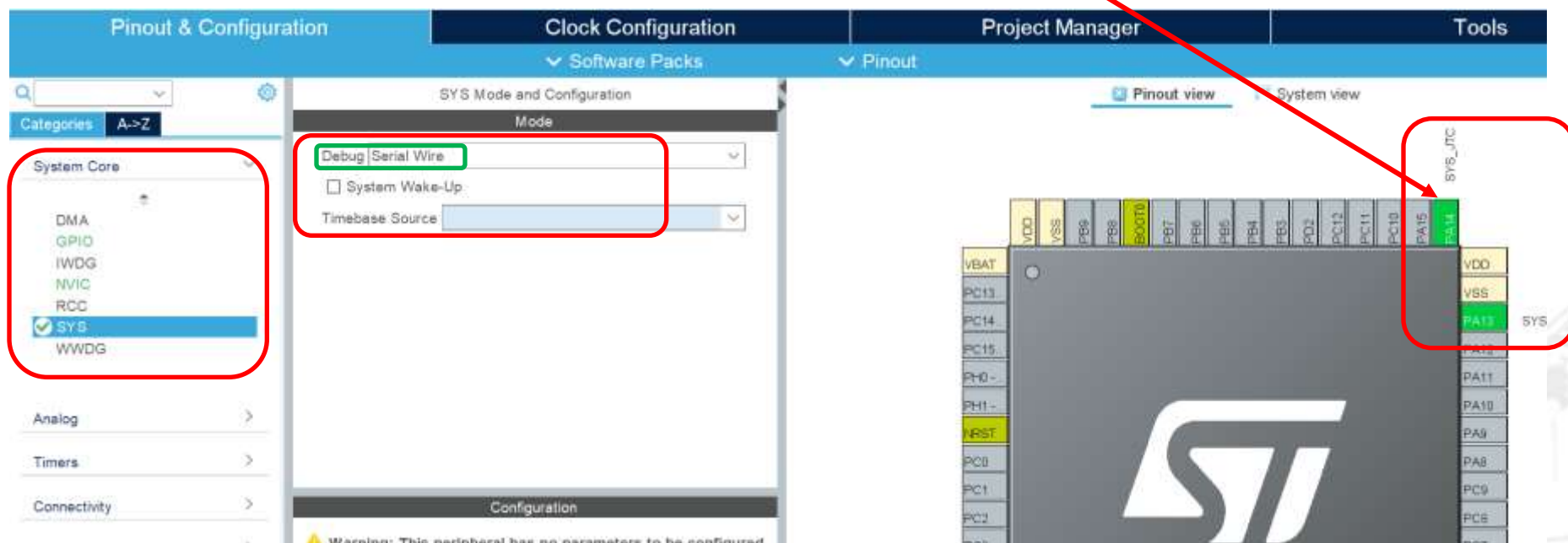
如果弹出Open Associated Perspective对话框中点OK



然后进入配置编辑模式窗口：

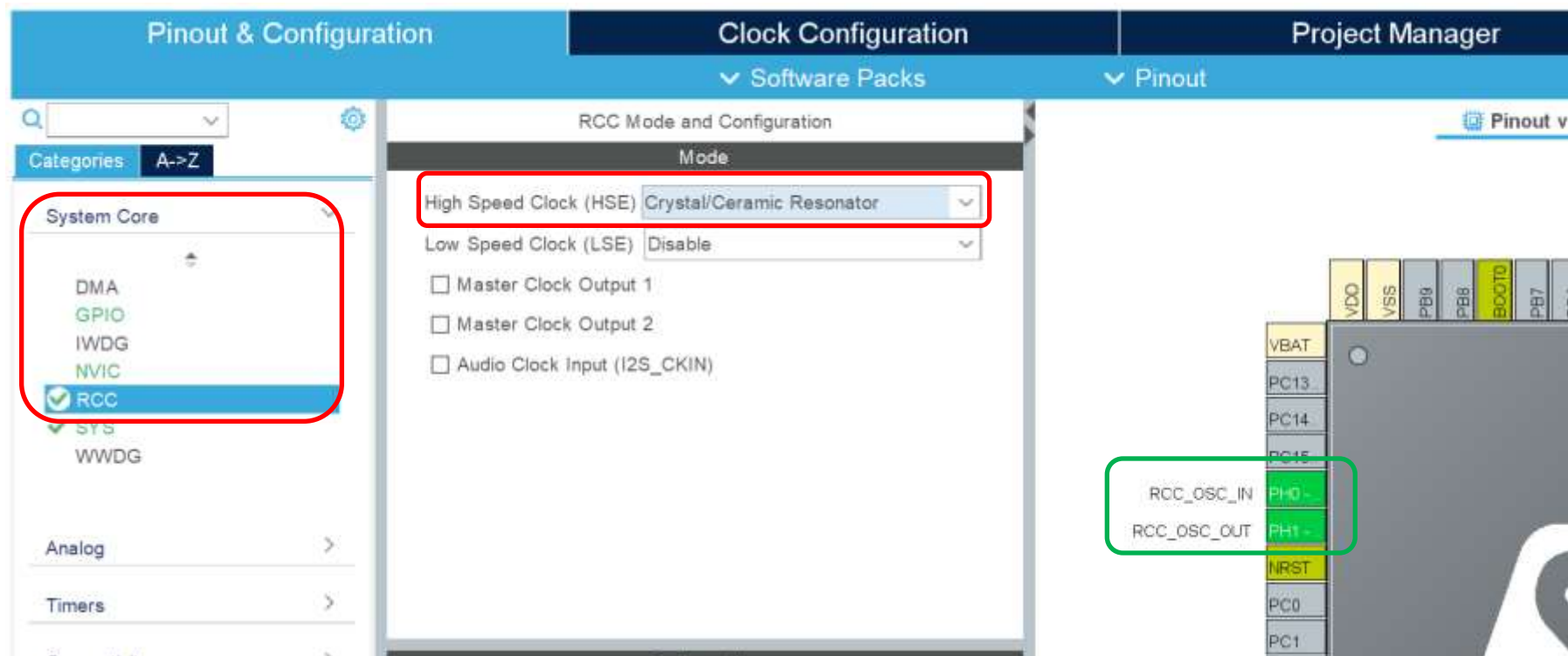


点击左上角的System Core->SYS，在弹出的对话框中，Debug选项选择Serial Wire。右侧的引脚编辑窗口（pinview）会自动显示调试功能占用额引脚。

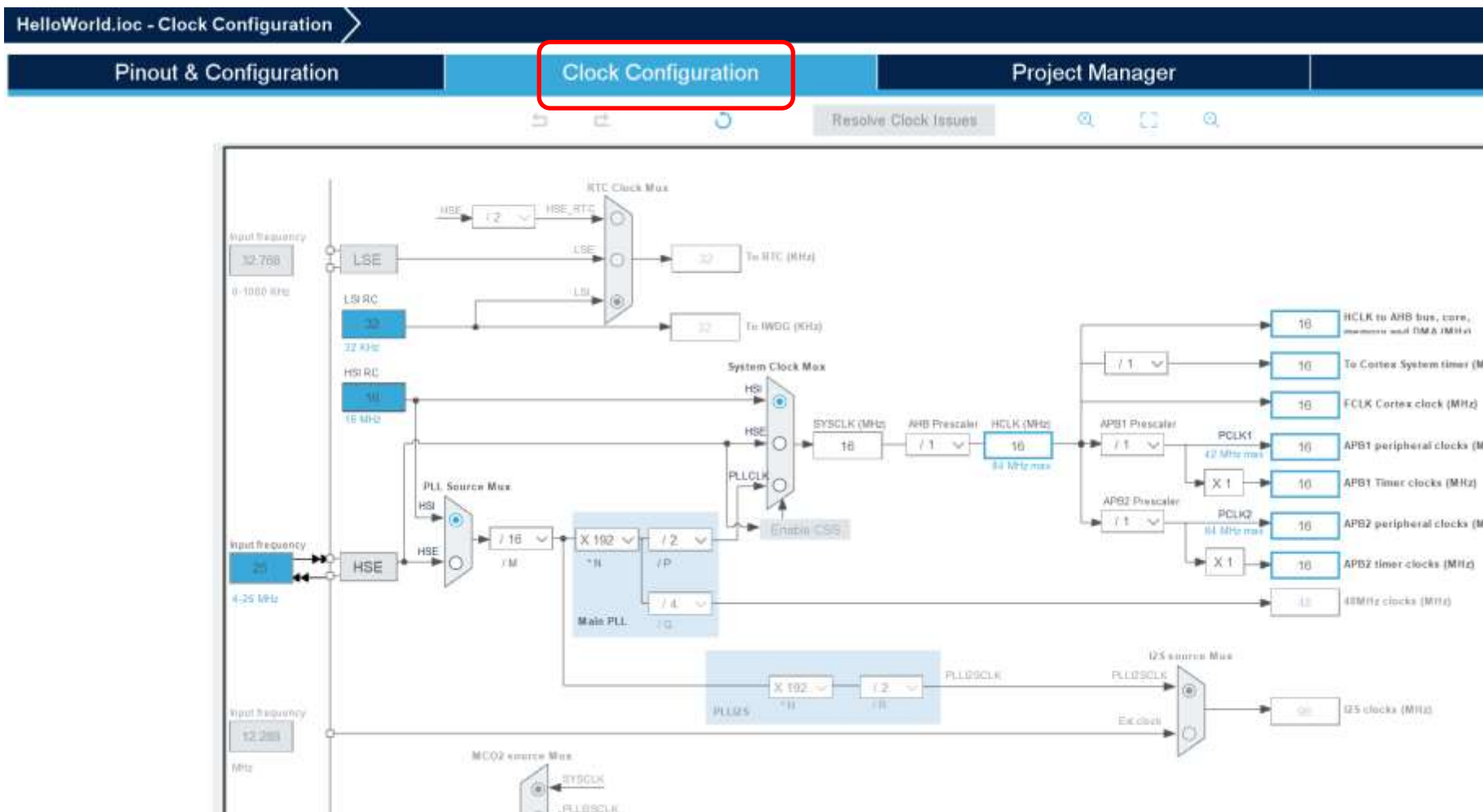


配置时钟来源:

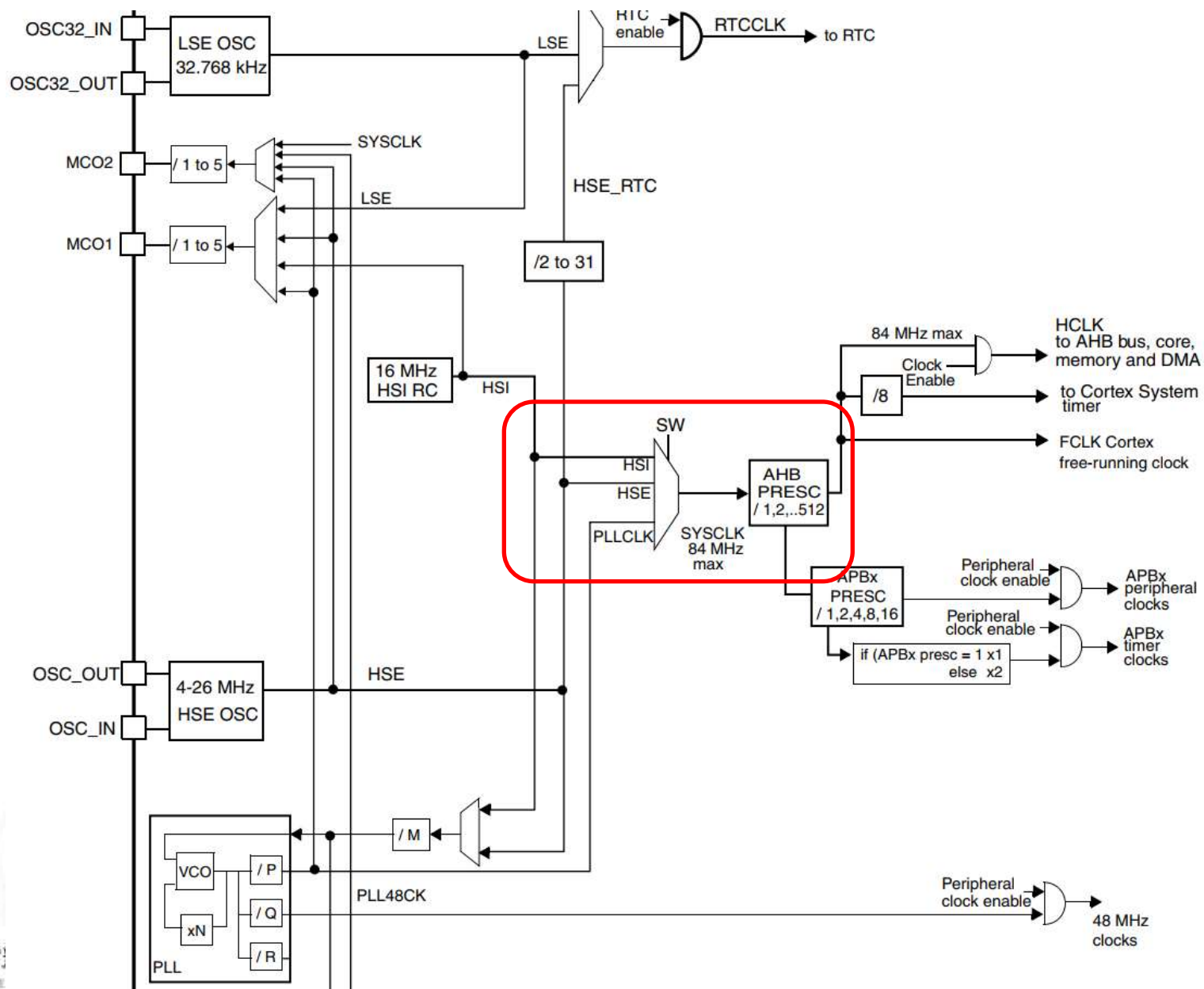
- 点击System Core->RCC。
- 由于开发板采用了外部8M的晶体振荡器作为时钟来源，因此在弹出的界面中，High Speed Clock (HSE)那一条选Crystal/Ceramic Resonator。
- 右侧引脚编辑窗口(pinout view)会自动显示所用的引脚。



点击Clock Configuration选项卡配置时钟



时钟树:

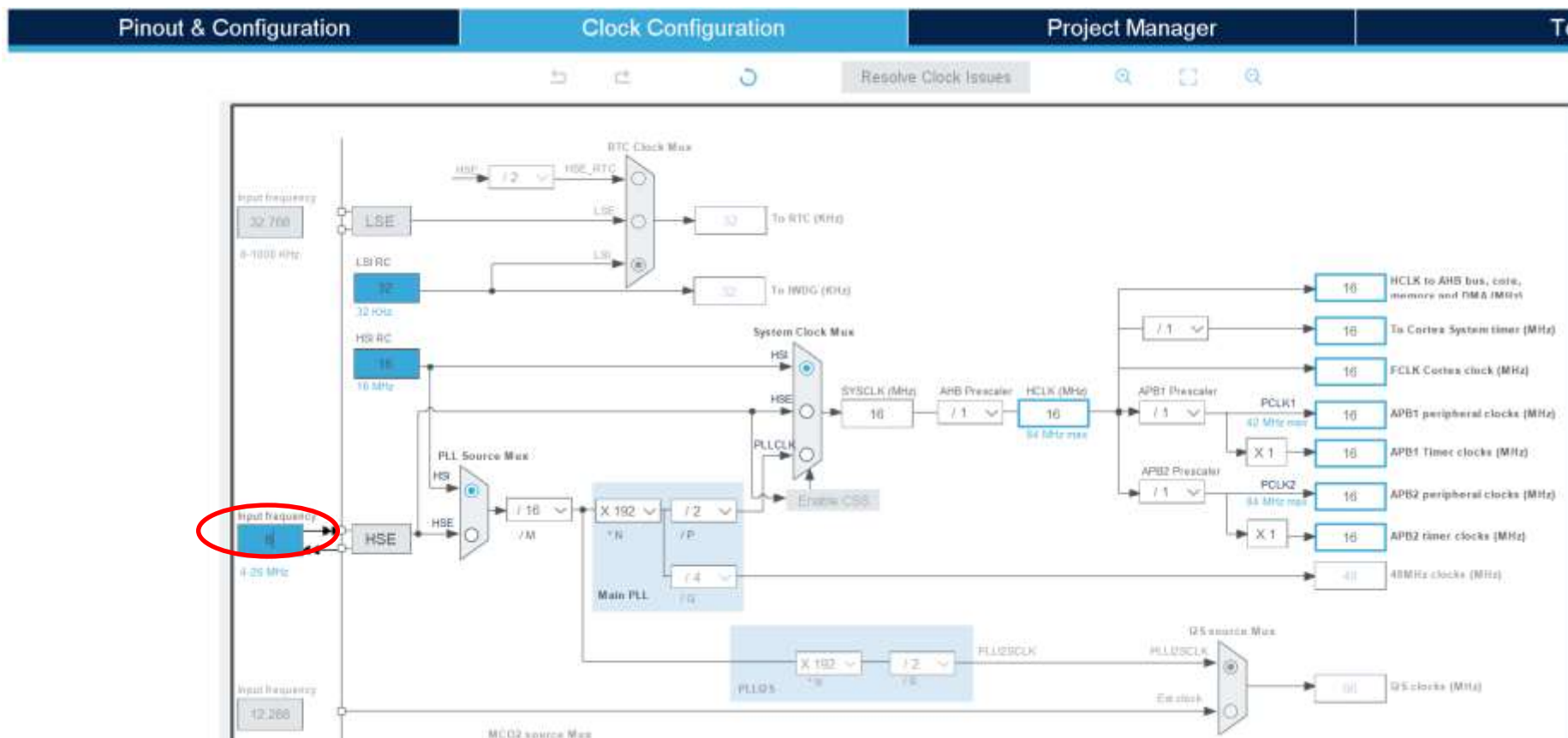


➤ 时钟树:

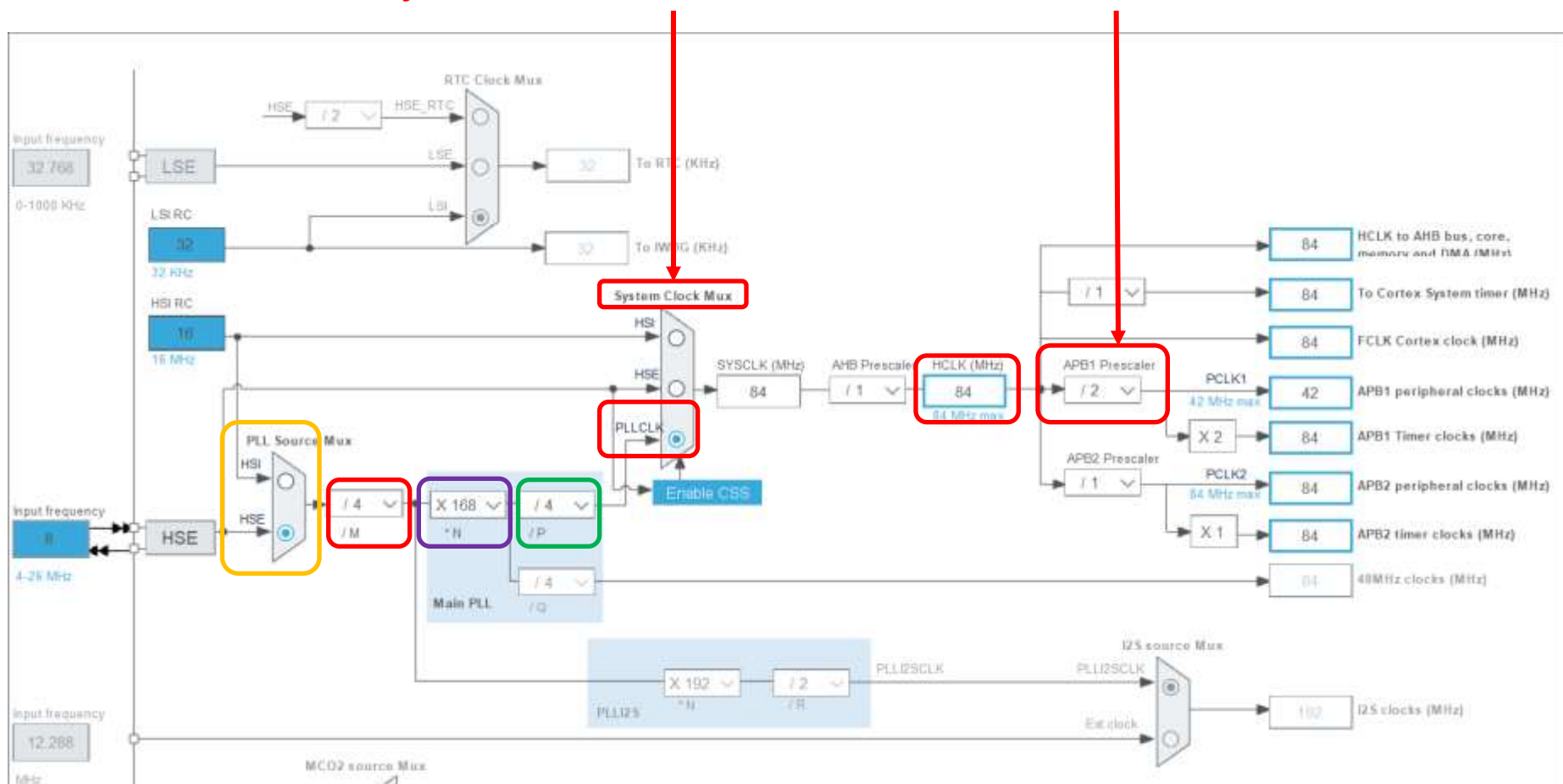
- 要想是MCU运行起来, 首先要正确设置系统时钟 (SYSCLK), STM32F401RB的SYSCLK的最高频率是84MHz。
- SYSCLK有三个来源, 可以由一个复选器通过软件设置选择。这三个来源分别是:
 - ① HSI (高速内部时钟), 由芯片内部的RC振荡器产生。频率固定为16MHz。系统上电后, 默认将HSI作为SYSCLK
 - ② HSE (高速外部时钟), 由外部晶体振荡器产生, 频率范围为4MHz-26MHz。一般情况下, 外部晶体振荡器的频率精度比内部RC振荡器更高。内部RC振荡器的频率容易受温度影响。
 - ③ PLLCLK (锁相环时钟输出): 由锁相环倍频器 (PLL) 产生的时钟, 可以达到很高的频率。
- PLL的输入可以是HSI也可以是HSE。由一个复选器选择。然后通过一个/M分频器降频后, 提供给PLL。PLL经过一个*N的倍频系数和一个/P的分频系数后, 输出PLLCLK。根据PLL的输入, 调节/M、*N和/P的值, 可以得到84MHz的PLLCLK, 用作SYSCLK



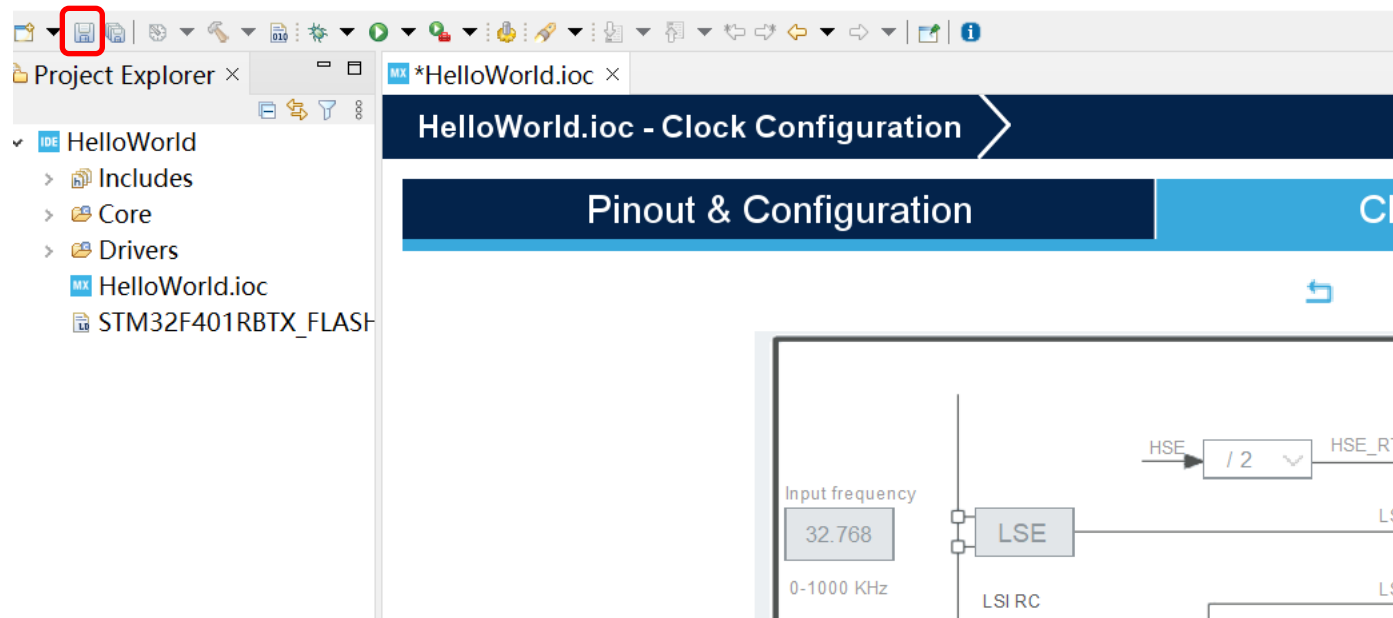
点击HSE左侧的Input frequency，将数值改为8，代表外部输入8MHz的时钟信号。



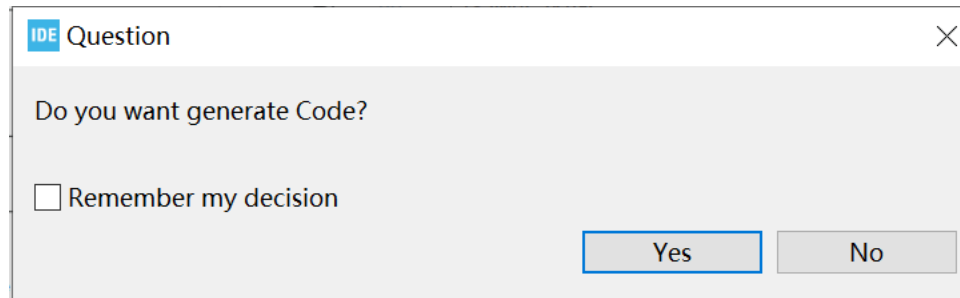
- 将PLL Source Mux选到HSE，代表使用外部8MHz的时钟信号作为PLL的信号源。
- 修改输入PLL的分频系数/M为4。锁相环的倍频系数改为 $\times 168$ ，分频系数/P改为4。
- 这样配置后，PLLCLK信号为84Mhz
- 系统时钟选择System Clock Mux改为PLLCLK，APB1 Prescaler 改为/2



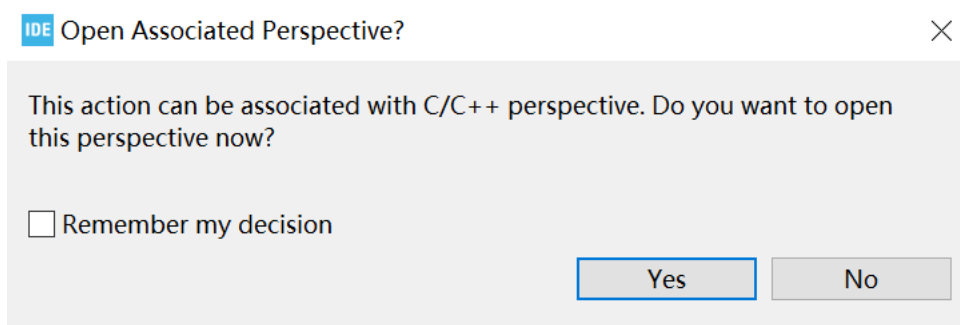
完成配置后，点击保存。



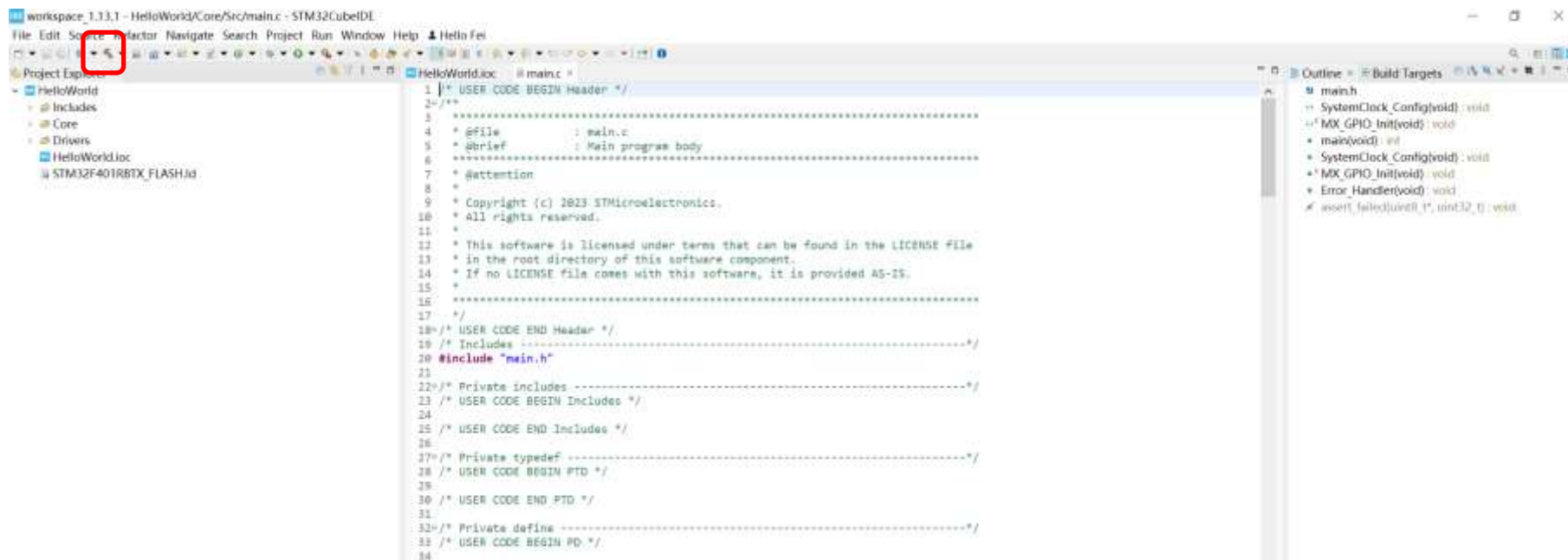
在弹出的对话框中，点击Yes，生成配置代码。



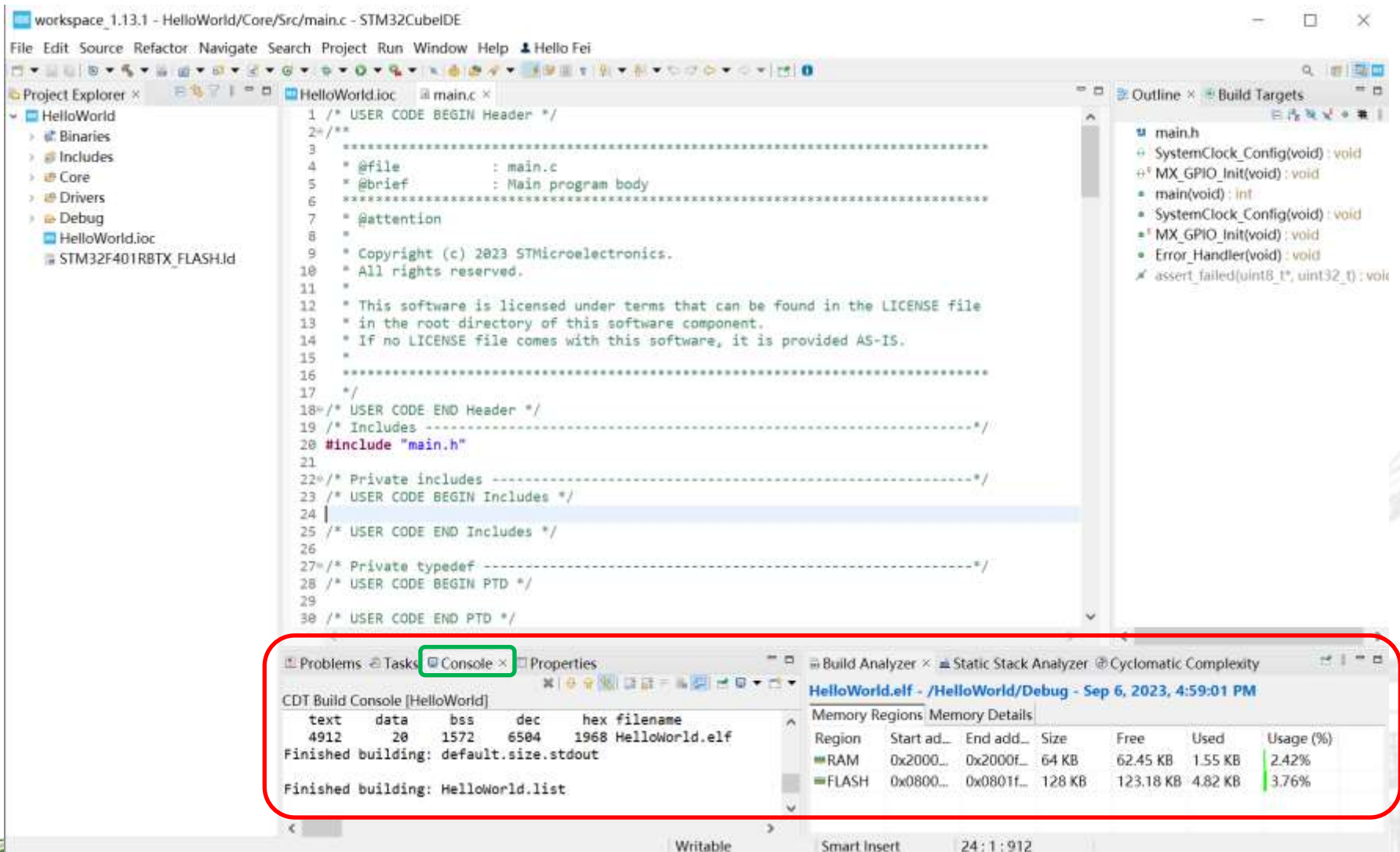
在弹出的Open Associated Perspective对话框中，点击Yes，进入编辑视图。



- 等待生成代码，会自动打开main.c文件
- 点击编译按钮后，启动编译程序，生成二进制可执行文件。

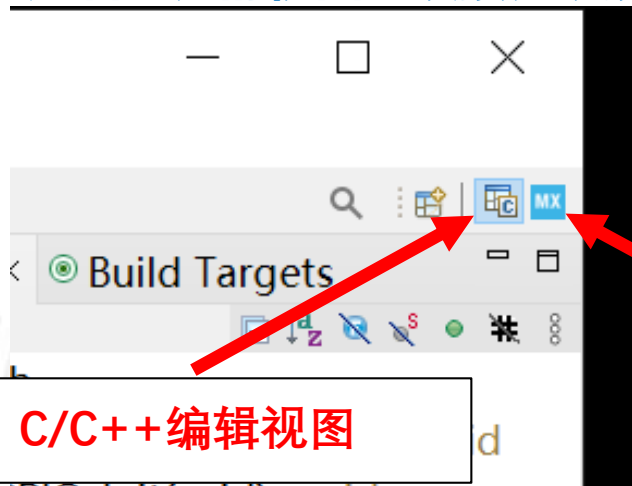


在Console窗口中，会有编译过程的输出和提示
在Build Analyzer窗口中，可以查看RAM和FLASH的占用情况。



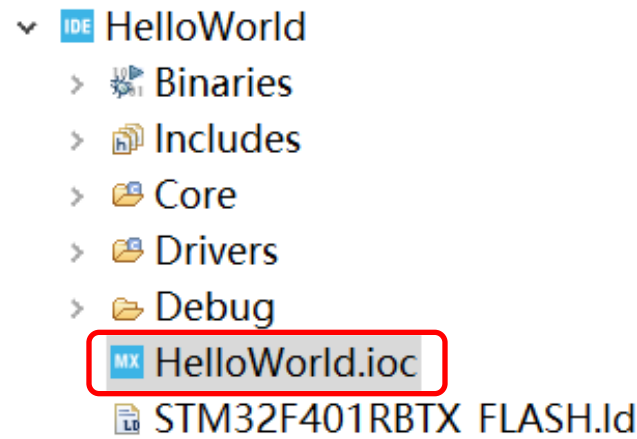
➤ 视图：

- STM32CubeIDE软件具有**代码编写、图形化MCU配置、代码调试**三项主要功能。
- 在进行代码编写时，为了方便编辑，一般需要在窗口左边打开Project Explorer窗口随时翻阅文件，在右边显示**Outline**窗口，显示当前**文件中的函数、变量、宏定义**等信息。在下方显示**Console、Problems**等窗口查看编译信息并显示**错误提示**。中间则是文件编辑器窗口，用于编辑文件。在对MCU进行图形化配置时，为了最大化的显示图形配置界面，一般不需要Outline、Console、Problems等串窗口。
- 这种完成某项功能时，各个辅助窗口的显示方式，被称为**视图(Perspective)**。
- STM32CubeIDE提供了**三种视图**，即**C/C++编辑视图、器件配置工具视图和调试视图**。通过右上角的视图选项按钮切换



➤ 视图：

- 新建工程后，会自动生成*.ioc*文件，此文件为**器件配置文件**。
- 在其他视图中，双击打开此文件会自动进入**器件配置工具视图**。

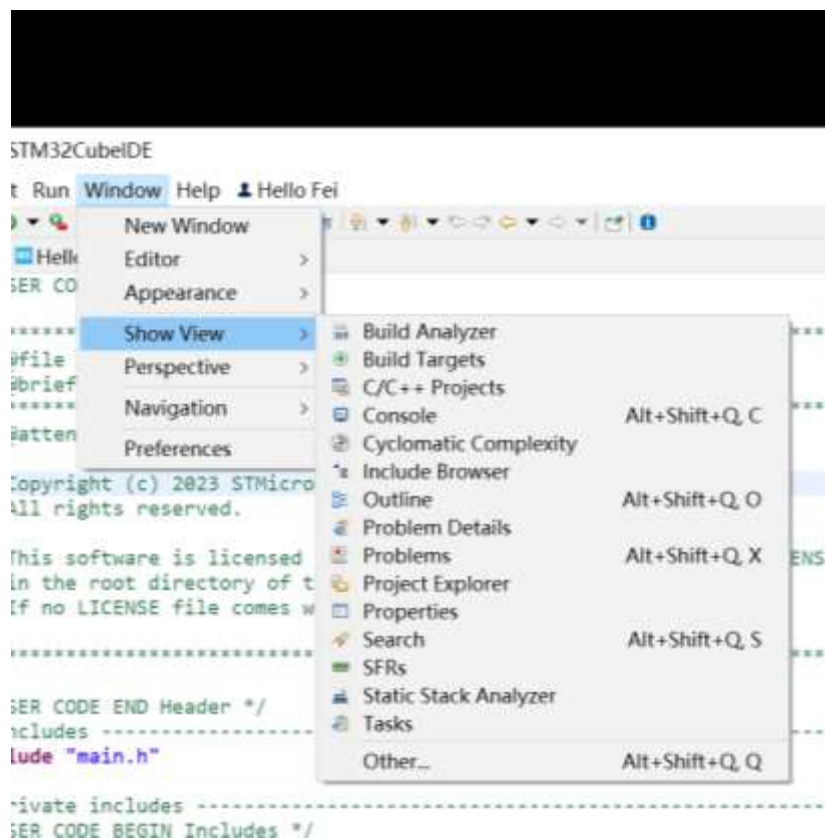


- 在器件配置工具视图打开其他*.c*或*.h*文件，会自动进入**C/C++ 编辑视图**。



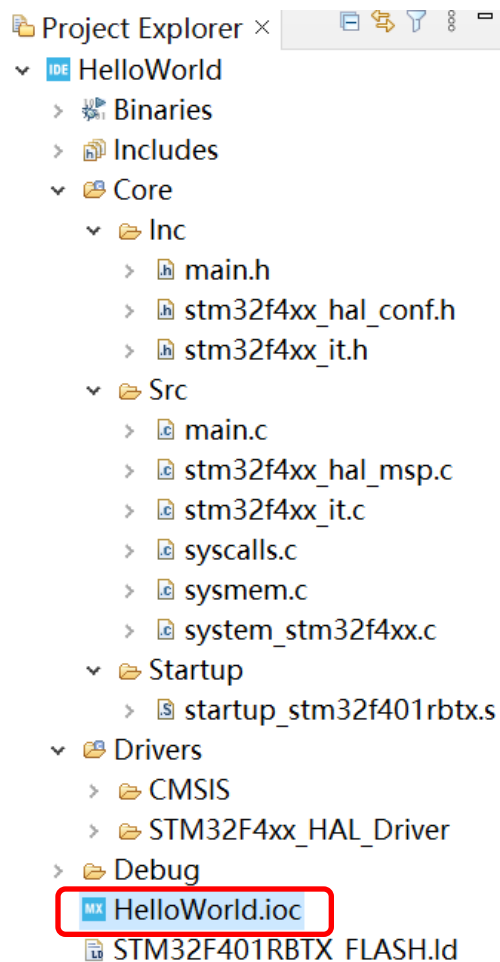
➤ 视图:

- 在每种视图中，都可以通过选择**Window->Show View**选择要打开的辅助窗口，而不影响其他视图的布局。



➤ 文件:

- 配置完成并自动生成代码后，会在**项目文件文件夹**下生成一系列文件和文件夹：。

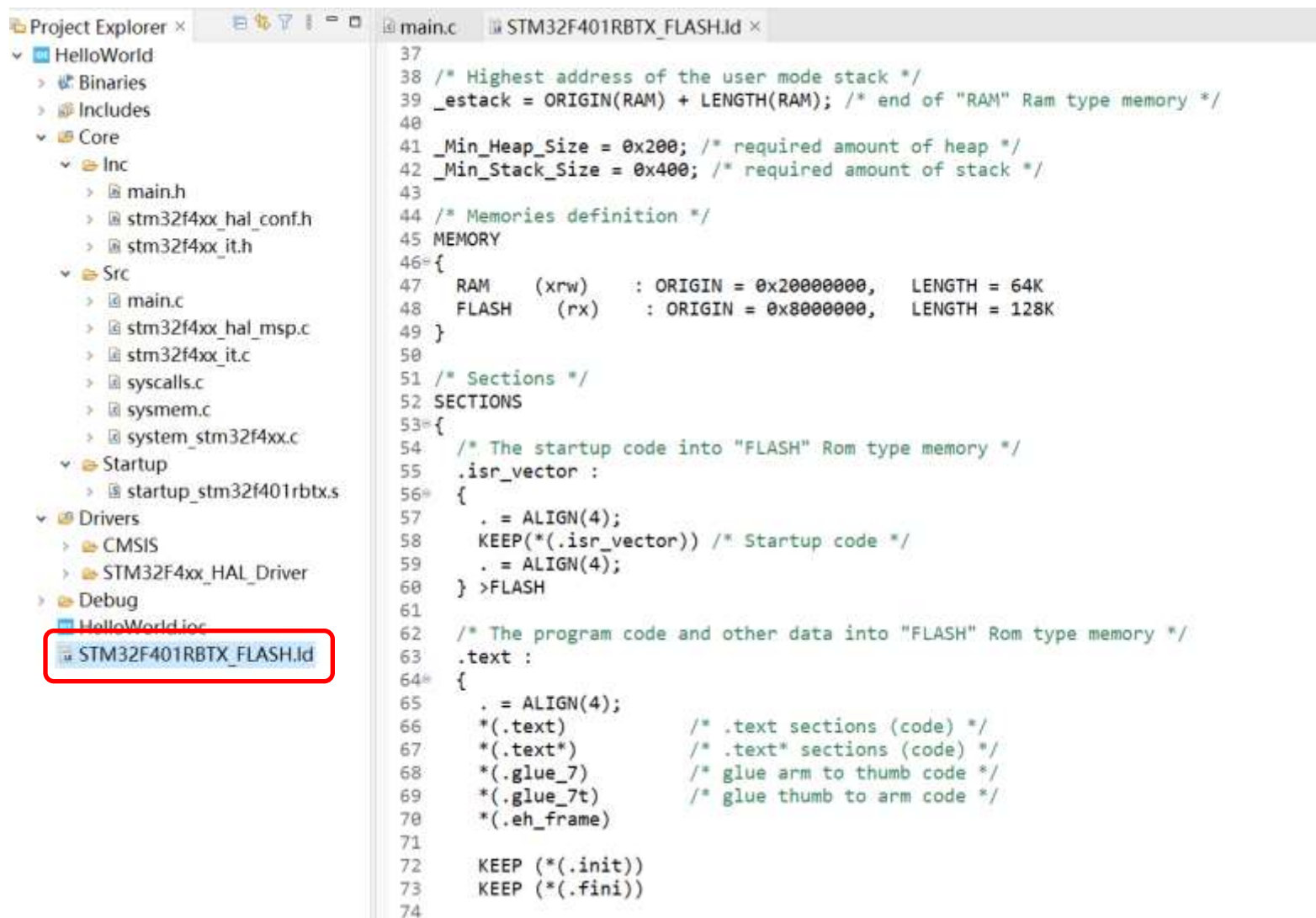


其中，刚刚打开过的HelloWorld.ioc文件为
器件配置文件



➤ 文件:

– STM32F401RCT6_FLASH.ld为存储配置文件:



The screenshot shows an IDE interface. On the left, the 'Project Explorer' pane displays a project named 'HelloWorld'. Under the 'Core' folder, the 'Inc' sub-folder contains 'main.h', 'stm32f4xx_hal_conf.h', and 'stm32f4xx_it.h'. The 'Src' sub-folder contains 'main.c', 'stm32f4xx_hal_msp.c', 'stm32f4xx_it.c', 'syscalls.c', 'systemem.c', and 'system_stm32f4xx.c'. The 'Startup' sub-folder contains 'startup_stm32f401rbtx.s'. The 'Drivers' sub-folder contains 'CMSIS' and 'STM32F4xx_HAL_Driver'. The 'Debug' sub-folder contains 'HelloWorld.ioc'. The 'STM32F401RBTX_FLASH.ld' file is highlighted with a red rectangle. On the right, the 'main.c' file is open, showing the following code:

```
37
38 /* Highest address of the user mode stack */
39 _estack = ORIGIN(RAM) + LENGTH(RAM); /* end of "RAM" Ram type memory */
40
41 _Min_Heap_Size = 0x200; /* required amount of heap */
42 _Min_Stack_Size = 0x400; /* required amount of stack */
43
44 /* Memories definition */
45 MEMORY
46 {
47     RAM        (xrw)  : ORIGIN = 0x20000000, LENGTH = 64K
48     FLASH      (rx)   : ORIGIN = 0x80000000, LENGTH = 128K
49 }
50
51 /* Sections */
52 SECTIONS
53 {
54     /* The startup code into "FLASH" Rom type memory */
55     .isr_vector :
56     {
57         . = ALIGN(4);
58         KEEP(*(.isr_vector)) /* Startup code */
59         . = ALIGN(4);
60     } >FLASH
61
62     /* The program code and other data into "FLASH" Rom type memory */
63     .text :
64     {
65         . = ALIGN(4);
66         *(.text)           /* .text sections (code) */
67         *(.text*)          /* .text* sections (code) */
68         *(.glue_7)         /* glue arm to thumb code */
69         *(.glue_7t)        /* glue thumb to arm code */
70         *(.eh_frame)
71
72         KEEP (*(.init))
73         KEEP (*(.fini))
74
```



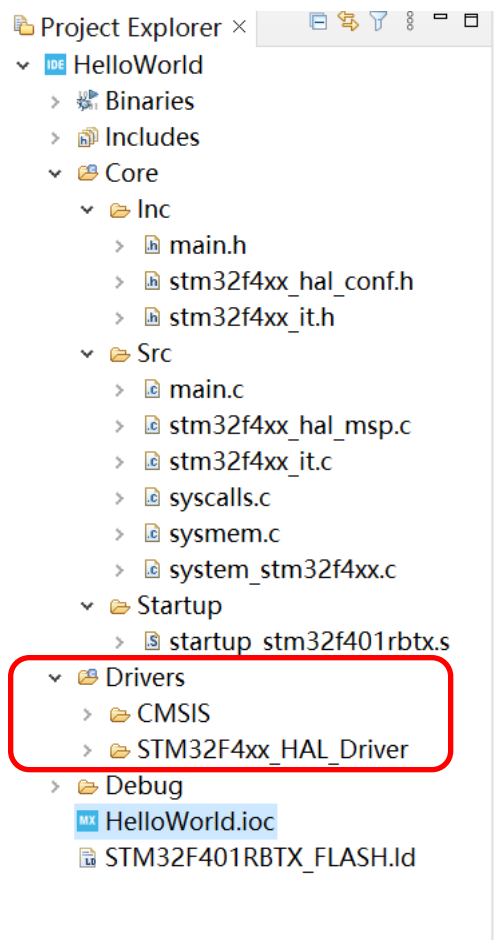
➤ 文件:

- **STM32F401RCT6_FLASH.ld**为存储配置文件。
- 描述了器件 (MCU) 各个**存储器在系统总线上的地址、大小**
- 同时描述了**指令、数据以及各个数据和代码段的存储位置**。
 - **MEMORY {}** 段描述了存储器在系统总线上的**地址和大小**。
 - STM32F401RB有两个存储器, 分别是静态随机存储器**SRAM**和**程序存储器FLASH**。
 - **SECTIONS {}** 段描述了不同类型的数据和代码的**存储位置**:
 - **.isr_vector** 表示**中断向量表**, 存储在FLASH中。
 - **.text**: 为**代码段**, 运行过程中只读不写, 存储在FLASH中。
 - **.rodata**: 为**常量数据**, 运行过程中只读不写, 存储在FLASH中。
 - **.init_array**: **进程初始化的**所运行的函数指针数组, 运行过程中只读不写, 存储在FLASH中。
 - **.fini_array**: **进程退出时的**所运行的函数指针数组, 运行过程中只读不写, 存储在FLASH中。
 - **.init_array**: 为**全局变量和静态变量的初始值**, 运行过程中只读不写, 存储在FLASH中。
 - **.data、.bss**: 为全局变量和静态局部变量保留的空间,



➤ 文件:

– Drivers 文件夹为ARM CMSIS库与HAL库文件，不建议修改。

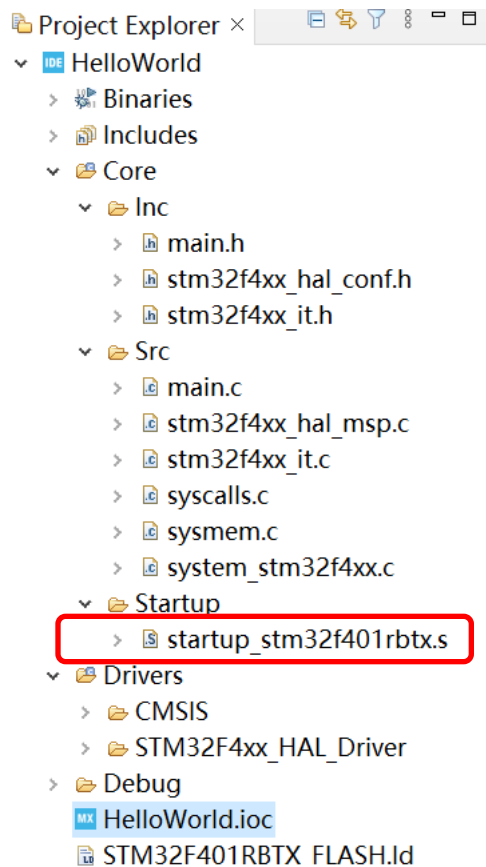


其中，刚刚打开过的HelloWorld.ioc文件为器件配置文件



➤ 文件:

- Core->Startup 文件夹下, 有startup_stm32f401rct6.s文件
- 启动文件, 由汇编语言编写。
- MCU上电后, 先运行该文件里的内容, 从Reset_Handler开始执行:



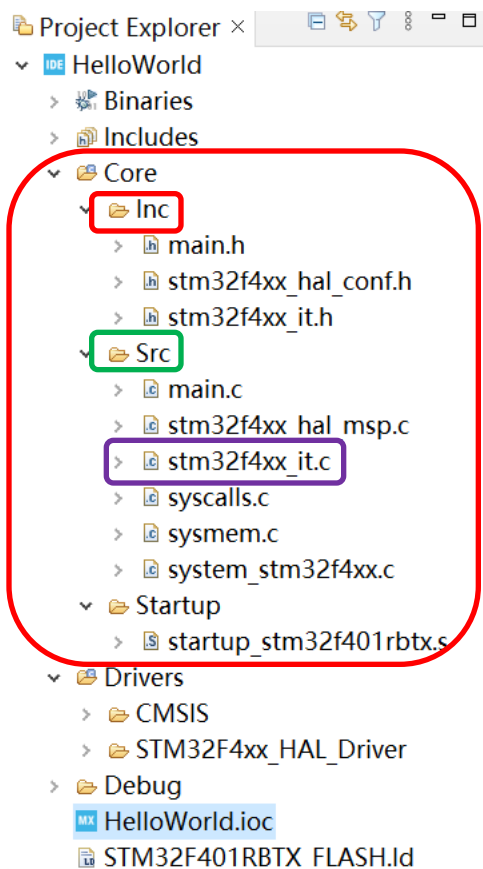
```
60 Reset_Handler:
61     ldr    sp, =_estack    /* set stack pointer */
62
63 /* Copy the data segment initializers from flash to SRAM */
64     ldr r0, =_sdata
65     ldr r1, =_edata
66     ldr r2, =_sidata
67     movs r3, #0
68     b LoopCopyDataInit
69
```

- 建立C语言运行环境的启动代码。
- 建立堆栈
- 初始化全局变量和静态变量
- 进入main函数



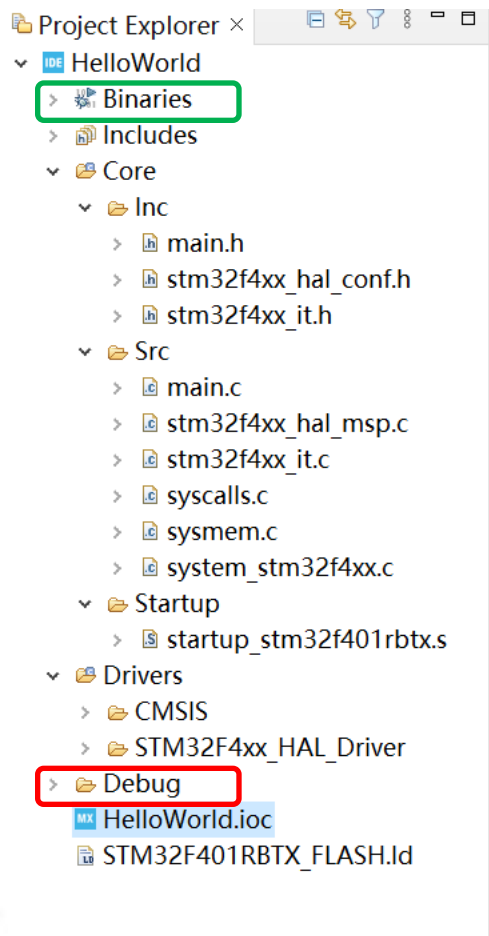
➤ 文件:

- Corel/Inc为头文件存放区域
- Corel/Src为C文件存放区域, main函数位于Corel/Src/main.c文件中。stm32f4xx_it.c文件中定义了中断服务函数。



➤ 文件:

- **Debug**文件夹存放编译生成的临时文件和二进制可执行文件。
- **Binaries**列出了项目中的二进制可执行文件



➤ 代码编辑:

- 打开main.c文件, 发现里面已经有很多内容, 其中大部分是注释。
- STM32CubeIDE软件规定了各部分代码的书写位置, 用户的代码只能写在/* USER CODE BEGIN XXX*/与/* USER CODE END XXX*/之间,
- 写在其他地方的代码会在下次修改器件配置文件并保存、生成代码后被删除。

```
main.c x
19 /* Includes ----- */
20 #include "main.h"
21
22 /* Private includes ----- */
23 /* USER CODE BEGIN Includes */
24
25 /* USER CODE END Includes */
26
27 /* Private typedef ----- */
28 /* USER CODE BEGIN PTD */
29
30 /* USER CODE END PTD */
31
32 /* Private define ----- */
33 /* USER CODE BEGIN PD */
34
35 /* USER CODE END PD */
36
37 /* Private macro ----- */
38 /* USER CODE BEGIN PM */
39
40 /* USER CODE END PM */
41
42 /* Private variables ----- */
43
44 /* USER CODE BEGIN PV */
45
46 /* USER CODE END PV */
47
48 /* Private function prototypes ----- */
49 void SystemClock_Config(void);
50 static void MX_GPIO_Init(void);
51 /* USER CODE BEGIN PFP */
52
53 /* USER CODE END PFP */
54
55 /* Private user code ----- */
56 /* USER CODE BEGIN 0 */
```



➤ 代码编辑:

```
/* USER CODE BEGIN Includes */
```

要包含的头文件

```
/* USER CODE END Includes */
```

```
/* USER CODE BEGIN PTD */
```

本文件需要用到的新的数据类型

```
/* USER CODE END PTD */
```

```
/* USER CODE BEGIN PD */
```

本文件需要用到的新的宏定义

```
/* USER CODE END PD */
```

```
/* USER CODE BEGIN PM */
```

本文件需要用到的新的宏

```
/* USER CODE END PM */
```



➤ 代码编辑:

```
/* USER CODE BEGIN PV */
```

本文件需要用到的全局变量

```
/* USER CODE END PV */
```

```
/* USER CODE BEGIN PFP */
```

声明本文件需要用到的函数

```
/* USER CODE END PFP */
```

```
/* USER CODE BEGIN 0 */
```

定义本文件需要用到的函数

```
/* USER CODE END 0 */
```



➤ 代码编辑:

- 进入main函数后, 自动生成了HAL_Init()代码, 用于初始化外设、FLASH和Systick。
- 在这之前需要运行的代码, 写在

```
/* USER CODE BEGIN 1 */
```

```
.....
```

```
/* USER CODE END 1 */
```

之间

```
main.c x
64 int main(void)
65 {
66     /* USER CODE BEGIN 1 */
67
68     /* USER CODE END 1 */
69
70     /* MCU Configuration-----*/
71
72     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
73     HAL_Init();
74
75     /* USER CODE BEGIN Init */
76
77     /* USER CODE END Init */
78
79     /* Configure the system clock */
80     SystemClock_Config();
81
82     /* USER CODE BEGIN SysInit */
83
84     /* USER CODE END SysInit */
85
86     /* Initialize all configured peripherals */
87     MX_GPIO_Init();
88     /* USER CODE BEGIN 2 */
89
90     /* USER CODE END 2 */
91
92     /* Infinite loop */
93     /* USER CODE BEGIN WHILE */
94     while (1)
95     {
96         /* USER CODE END WHILE */
97
98         /* USER CODE BEGIN 3 */
99
100     }
101 }
```



➤ 代码编辑:

- 系统生成了**SystemClock_Config()**代码，用于**初始化时钟**。时钟初始化之前，系统主频为**HIS RC 内部RC高速振荡器**提供的**16MHz**，时钟初始化之后，系统主频变为**PLL**提供的**84MHz**。

- 在这之前需要运行的代码，写在

```
/* USER CODE BEGIN Init */
```

.....

```
/* USER CODE END Init */
```

之间

```
main.c x
64 int main(void)
65 {
66     /* USER CODE BEGIN 1 */
67
68     /* USER CODE END 1 */
69
70     /* MCU Configuration-----*/
71
72     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
73     HAL_Init();
74
75     /* USER CODE BEGIN Init */
76
77     /* USER CODE END Init */
78
79     /* Configure the system clock */
80     SystemClock_Config();
81
82     /* USER CODE BEGIN SysInit */
83
84     /* USER CODE END SysInit */
85
86     /* Initialize all configured peripherals */
87     MX_GPIO_Init();
88     /* USER CODE BEGIN 2 */
89
90     /* USER CODE END 2 */
91
92     /* Infinite loop */
93     /* USER CODE BEGIN WHILE */
94     while (1)
95     {
96         /* USER CODE END WHILE */
97
98         /* USER CODE BEGIN 3 */
99     }
100     /* USER CODE END 3 */
101 }
```



➤ 代码编辑:

- 需要在**时钟初始化之后**、**外设初始化之前**运行的**初始化代码**写在

```
/* USER CODE BEGIN sysInit */
```

```
.....
```

```
/* USER CODE END sysInit */
```

之间

```
main.c x
64 int main(void)
65 {
66     /* USER CODE BEGIN 1 */
67
68     /* USER CODE END 1 */
69
70     /* MCU Configuration-----*/
71
72     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
73     HAL_Init();
74
75     /* USER CODE BEGIN Init */
76
77     /* USER CODE END Init */
78
79     /* Configure the system clock */
80     SystemClock_Config();
81
82     /* USER CODE BEGIN SysInit */
83
84     /* USER CODE END SysInit */
85
86     /* Initialize all configured peripherals */
87     MX_GPIO_Init();
88     /* USER CODE BEGIN 2 */
89
90     /* USER CODE END 2 */
91
92     /* Infinite loop */
93     /* USER CODE BEGIN WHILE */
94     while (1)
95     {
96         /* USER CODE END WHILE */
97
98         /* USER CODE BEGIN 3 */
99     }
100     /* USER CODE END 3 */
101 }
```



➤ 代码编辑:

- 系统生成了GPIO的初始化语句**MX_GPIO_Init()**;用于完成器件配置文件中的**GPIO的配置**。
- 初始化之后, 会进入主循环, 在进入主循环之前, 需要运行的语句, 写在

`/* USER CODE BEGIN 2 */`

`.....`

`/* USER CODE END 2 */`

之间

```
main.c x
64 int main(void)
65 {
66     /* USER CODE BEGIN 1 */
67
68     /* USER CODE END 1 */
69
70     /* MCU Configuration-----*/
71
72     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
73     HAL_Init();
74
75     /* USER CODE BEGIN Init */
76
77     /* USER CODE END Init */
78
79     /* Configure the system clock */
80     SystemClock_Config();
81
82     /* USER CODE BEGIN SysInit */
83
84     /* USER CODE END SysInit */
85
86     /* Initialize all configured peripherals */
87     MX_GPIO_Init();
88     /* USER CODE BEGIN 2 */
89
90     /* USER CODE END 2 */
91
92     /* Infinite loop */
93     /* USER CODE BEGIN WHILE */
94     while (1)
95     {
96         /* USER CODE END WHILE */
97
98         /* USER CODE BEGIN 3 */
99     }
100     /* USER CODE END 3 */
101 }
```



➤ 代码编辑:

– 进入主循环前后, 需要运行的程序语句, 写在

```
/* USER CODE BEGIN WHILE */
```

```
while (1)
```

```
{
```

```
/* USER CODE END WHILE */
```

之间

```
main.c ×
64 int main(void)
65 {
66     /* USER CODE BEGIN 1 */
67
68     /* USER CODE END 1 */
69
70     /* MCU Configuration-----*/
71
72     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
73     HAL_Init();
74
75     /* USER CODE BEGIN Init */
76
77     /* USER CODE END Init */
78
79     /* Configure the system clock */
80     SystemClock_Config();
81
82     /* USER CODE BEGIN SysInit */
83
84     /* USER CODE END SysInit */
85
86     /* Initialize all configured peripherals */
87     MX_GPIO_Init();
88     /* USER CODE BEGIN 2 */
89
90     /* USER CODE END 2 */
91
92     /* Infinite loop */
93     /* USER CODE BEGIN WHILE */
94     while (1)
95     {
96         /* USER CODE END WHILE */
97
98         /* USER CODE BEGIN 3 */
99
100     }
101     /* USER CODE END 3 */
102 }
```



➤ 代码编辑:

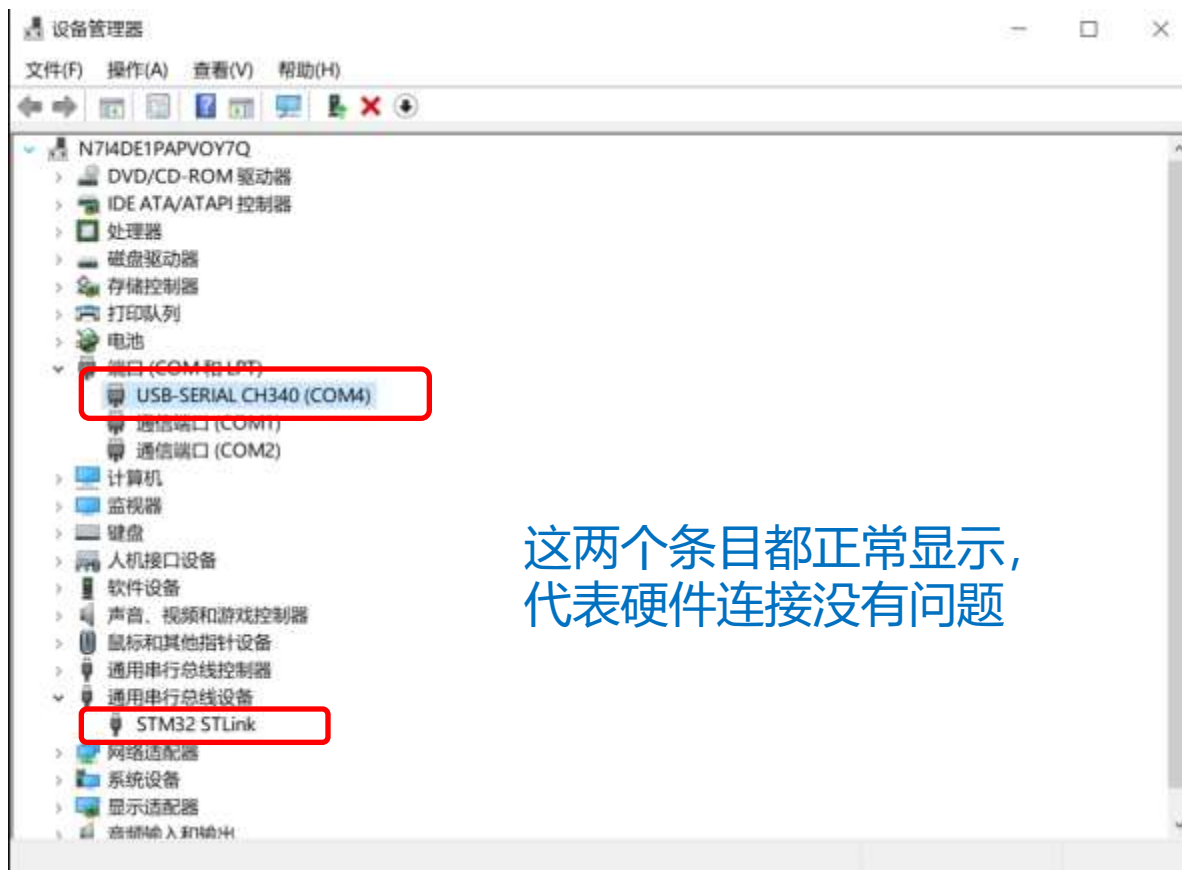
- Main函数之后的函数定义,

***如SystemClock_Config、MX_GPIO_Init等为系统自动生成的函数, 不可修改。**



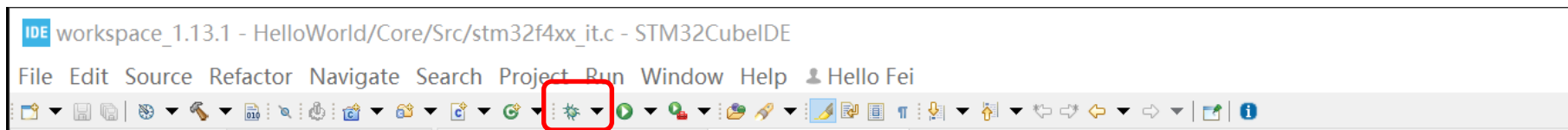
➤ 代码调试:

- 将开发板的USB线插到电脑的USB口上。开发板的**电源指示灯会亮起**。
- 如果提示**无法识别驱动**，则**安装CH341SER**，并安装驱动。
- 安装完成后，打开设备管理器确认。



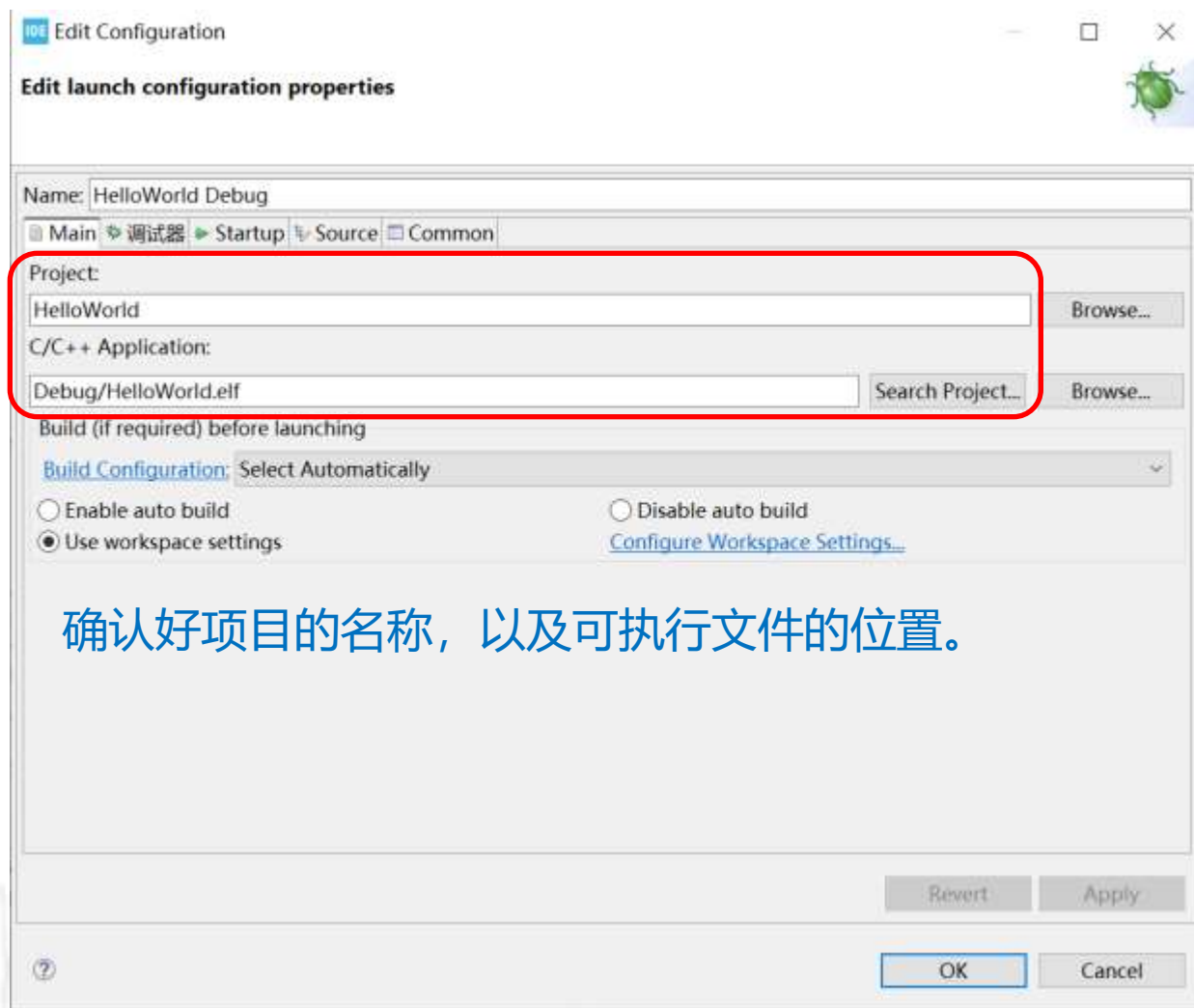
➤ 代码调试：

– 点击调试按钮



➤ 代码调试:

– 第一次调试会弹出调试配置对话框:

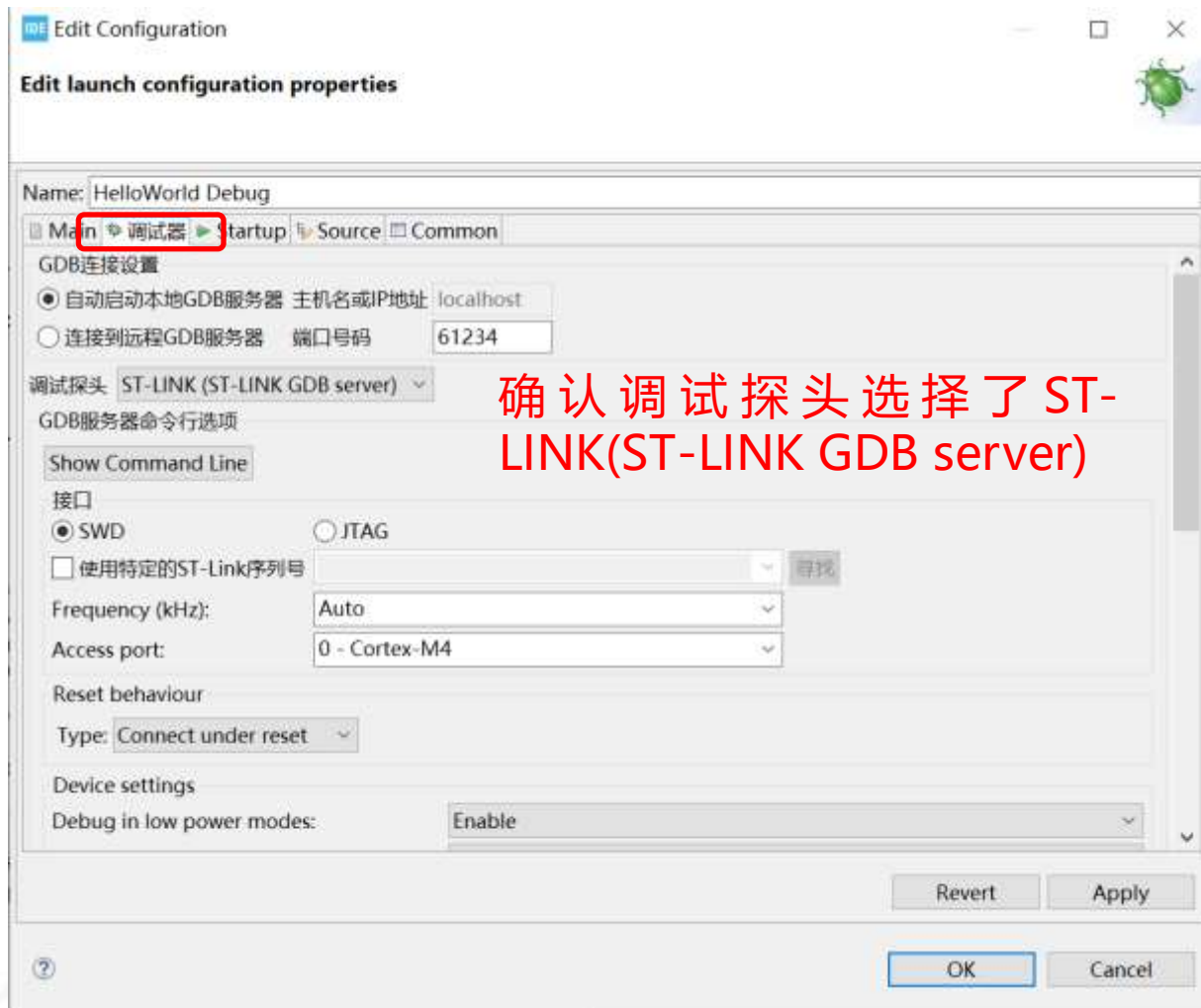


确认好项目的名称，以及可执行文件的位置。



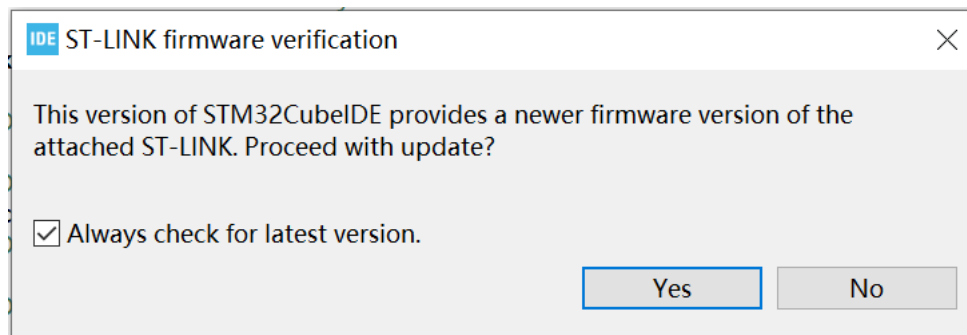
➤ 代码调试:

— 点击调试器选项:

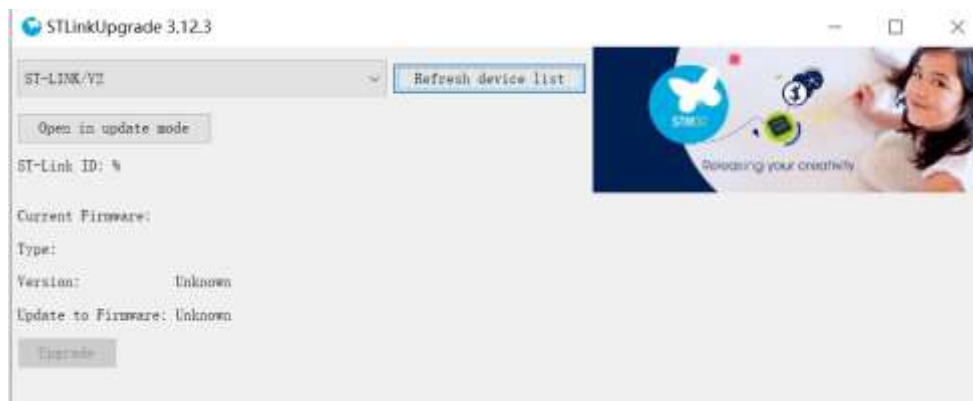


➤ 代码调试:

– 第一次使用调试器，会弹出是否需要更新的对话框

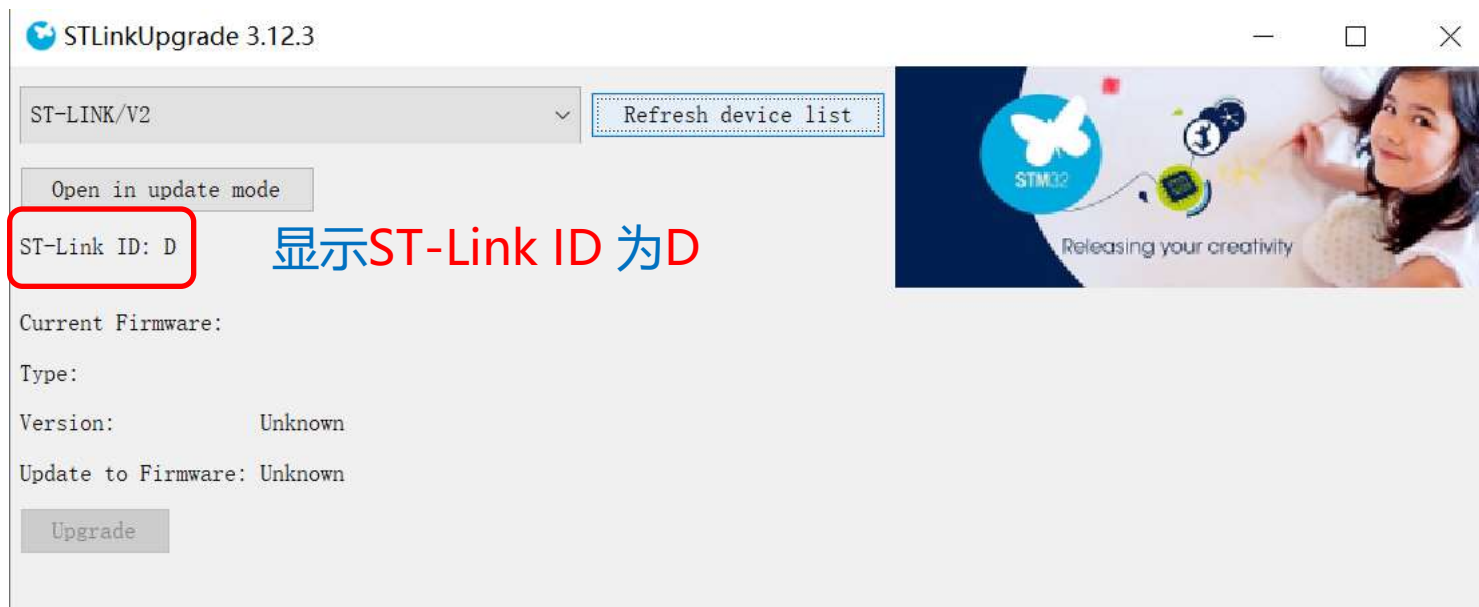


单击Yes，弹出升级对话框



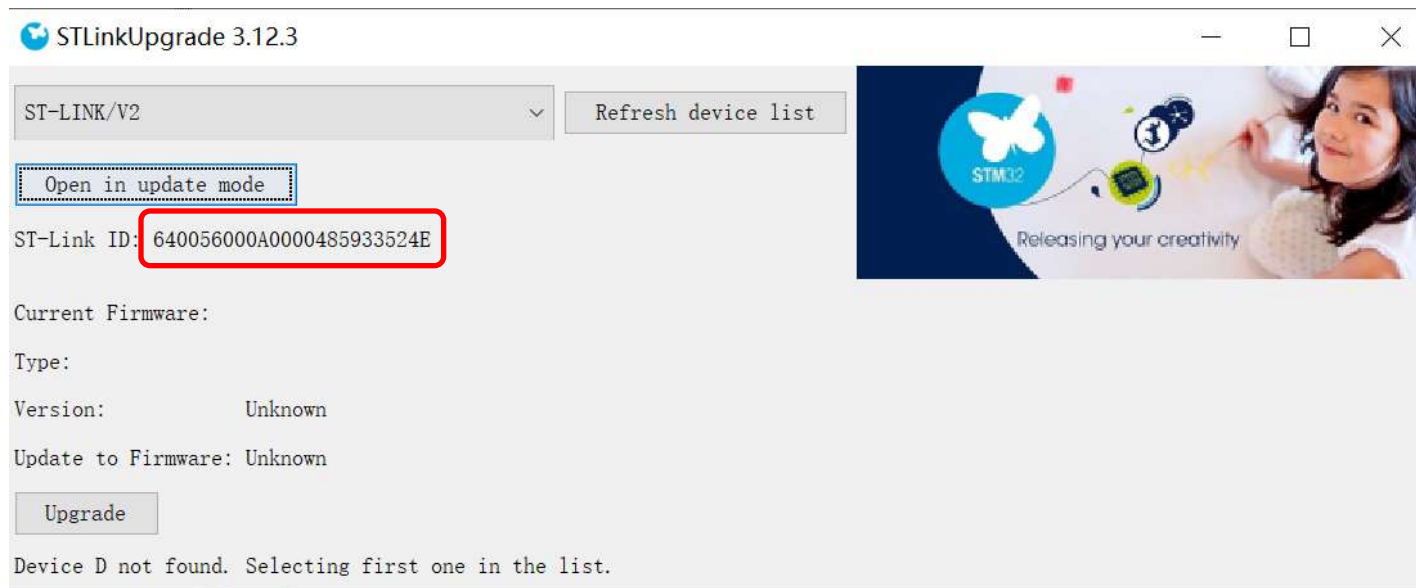
➤ 代码调试:

– 将USB口从电脑上拔下来，然后再插上。



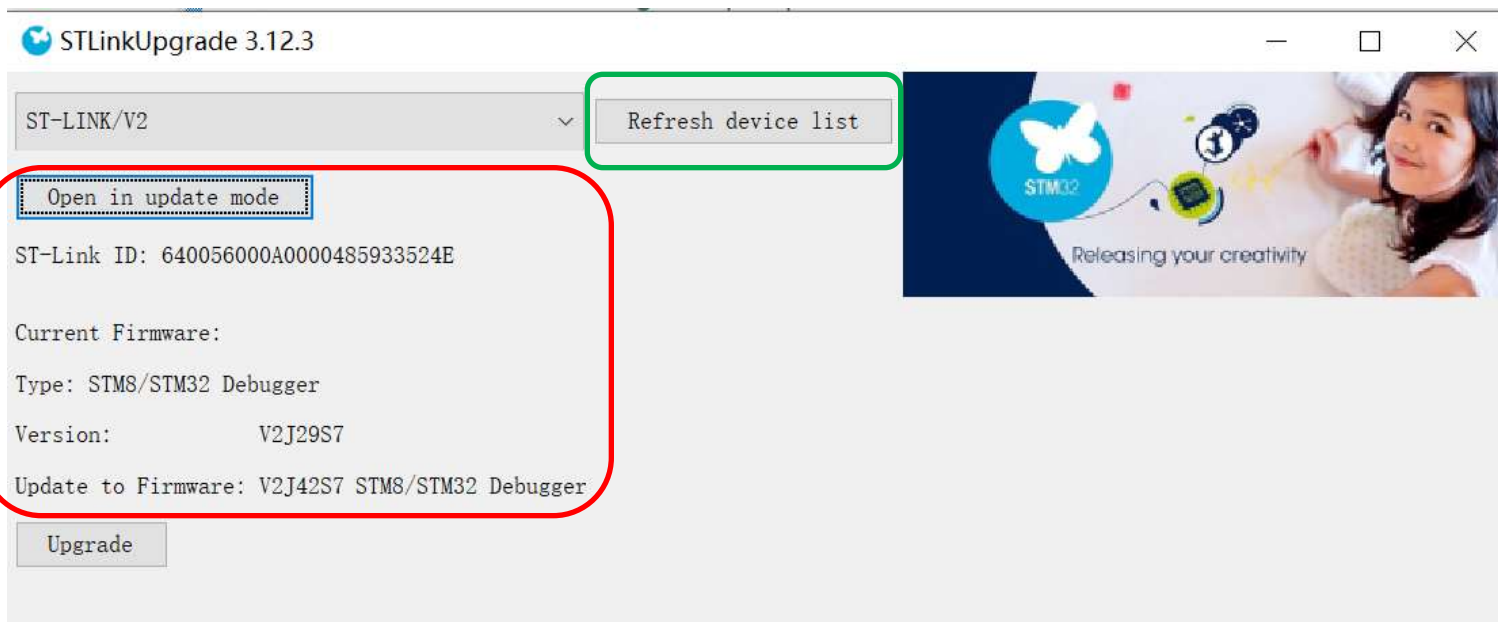
➤ 代码调试:

– 点 **Open in update mode**, 会 **读出ST-Link ID**



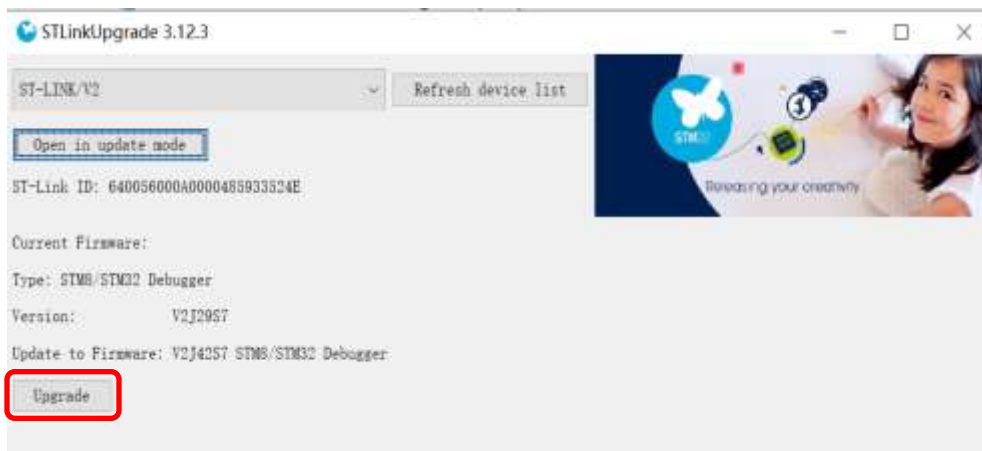
➤ 代码调试:

- 然后再点 **Refresh device list**, 再点 **Open in update mode**, 会读出当前ST-Link的信息:



➤ 代码调试:

- 点update开始升级:

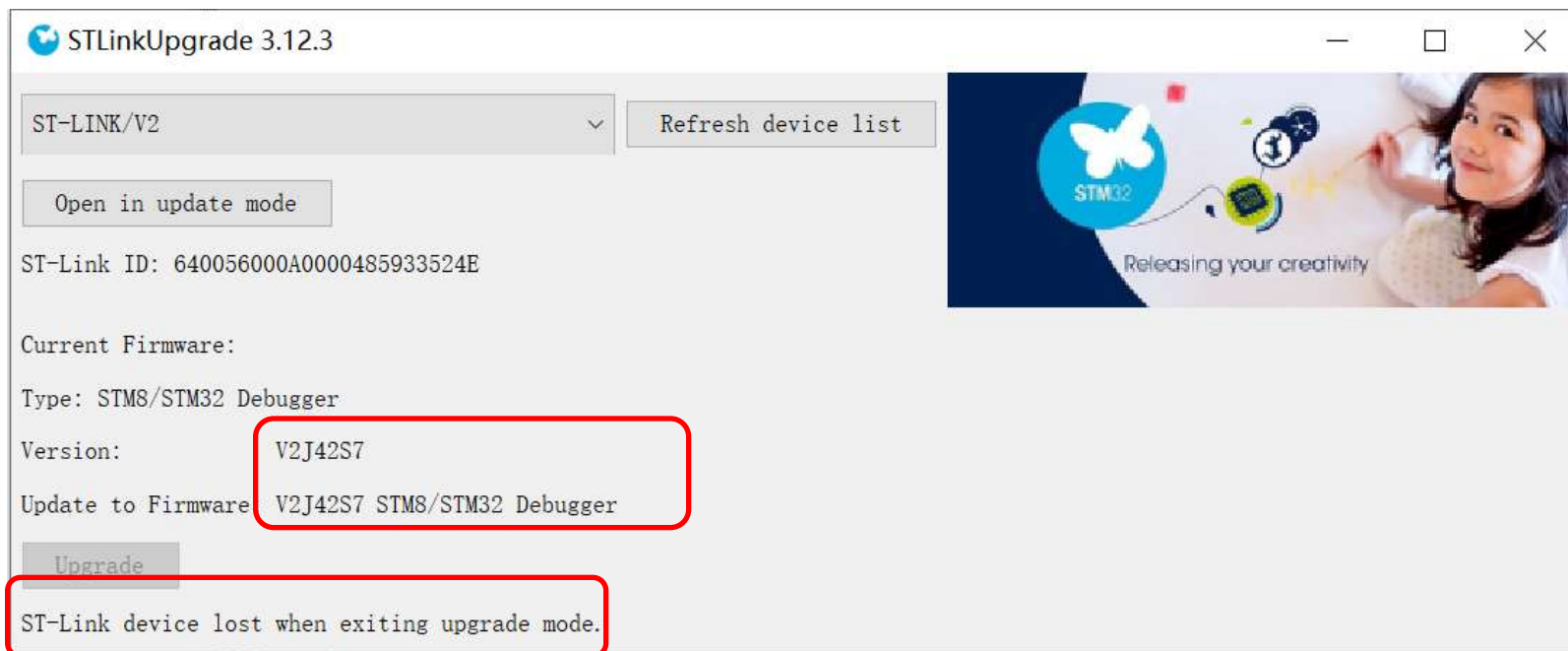


– 升级中



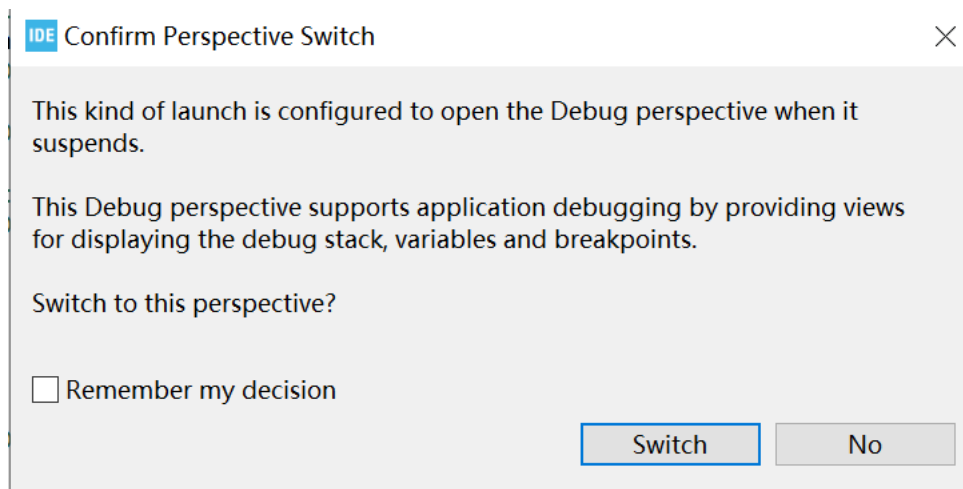
➤ 代码调试:

- 升级完成后, Version会与Update to Firmware中的版本号一致
- 左下角会显示 **ST-Link device lost when existing update mode.**
- 关闭升级对话框, 拔掉USB连接线, 再插上USB连接线。



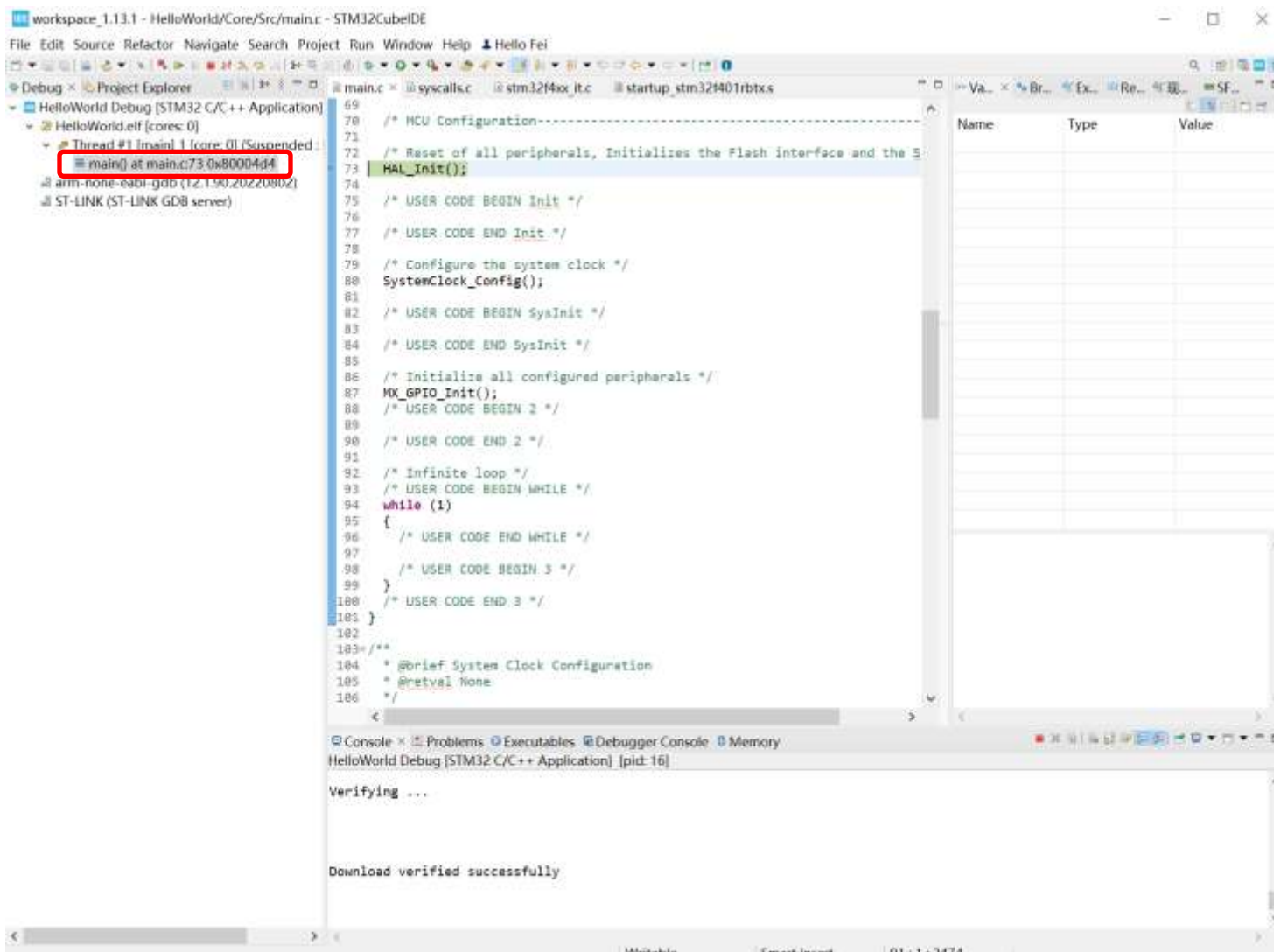
➤ 代码调试:

- 点击调试按钮，可进行正常调试。
- 软件会自动编译项目，生成二进制可执行程序，连接MCU，并将程序下载到MCU中。
- 调试时，会弹出对话框，询问是否进入调试视图，单击Switch进入调试视图。

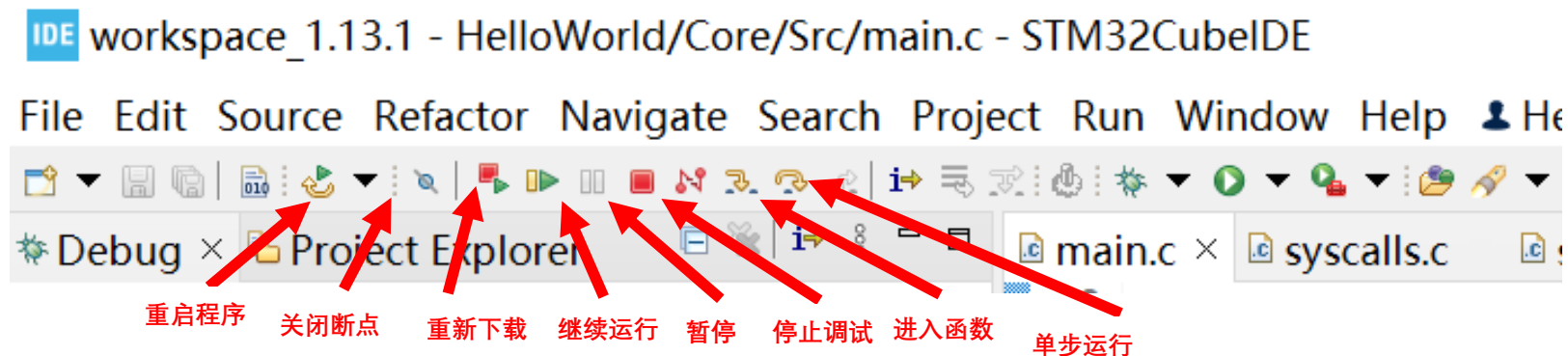


➤ 代码调试:

– 程序暂停在进入main函数后的第一条语句。



➤ 调试视图:

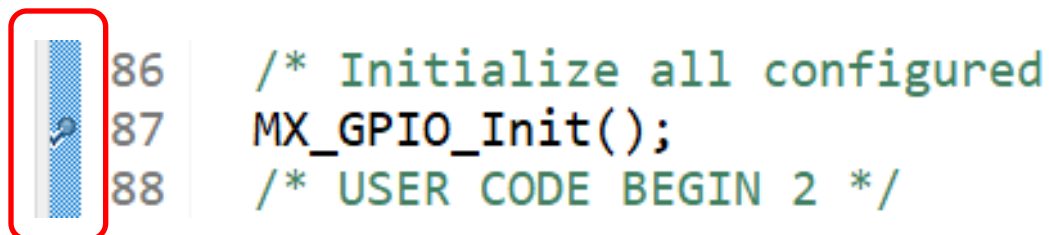


点击**停止按钮**，会**停止调试**
自动断开与MCU的连接
由调试视图自动切换到C/C++编辑视图。



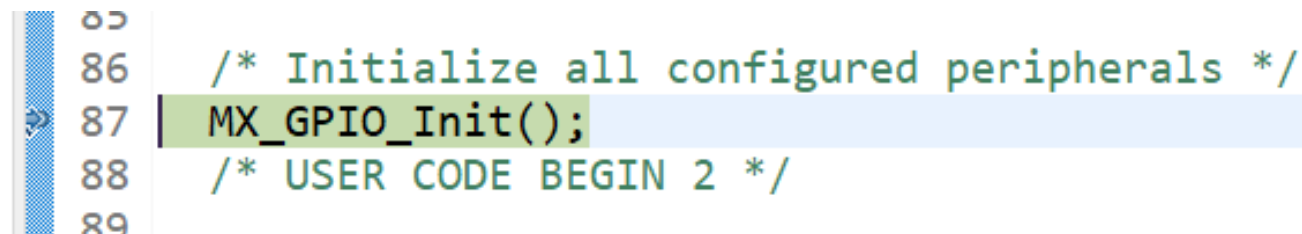
➤ 设置断点:

- 在行号左侧的蓝色区域**双击**，可以**设置断点**。



```
86  /* Initialize all configured
87  MX_GPIO_Init();
88  /* USER CODE BEGIN 2 */
```

- 程序运行到此处后，自动暂停。



```
85
86  /* Initialize all configured peripherals */
87  MX_GPIO_Init();
88  /* USER CODE BEGIN 2 */
89
```



➤ 代码调试实例:

– 定义一个全局变量g

```
/* USER CODE BEGIN PV */  
int32_t g=0;  
/* USER CODE END PV */
```

– 定义一个局部变量

```
/* USER CODE BEGIN 1 */  
    int32_t l=0;  
/* USER CODE END 1 */
```

– 写一点简单的逻辑代码

```
/* USER CODE BEGIN WHILE */  
    while (1)  
    {  
        g++;  
        l++;  
        HAL_Delay(500);  
/* USER CODE END WHILE */
```



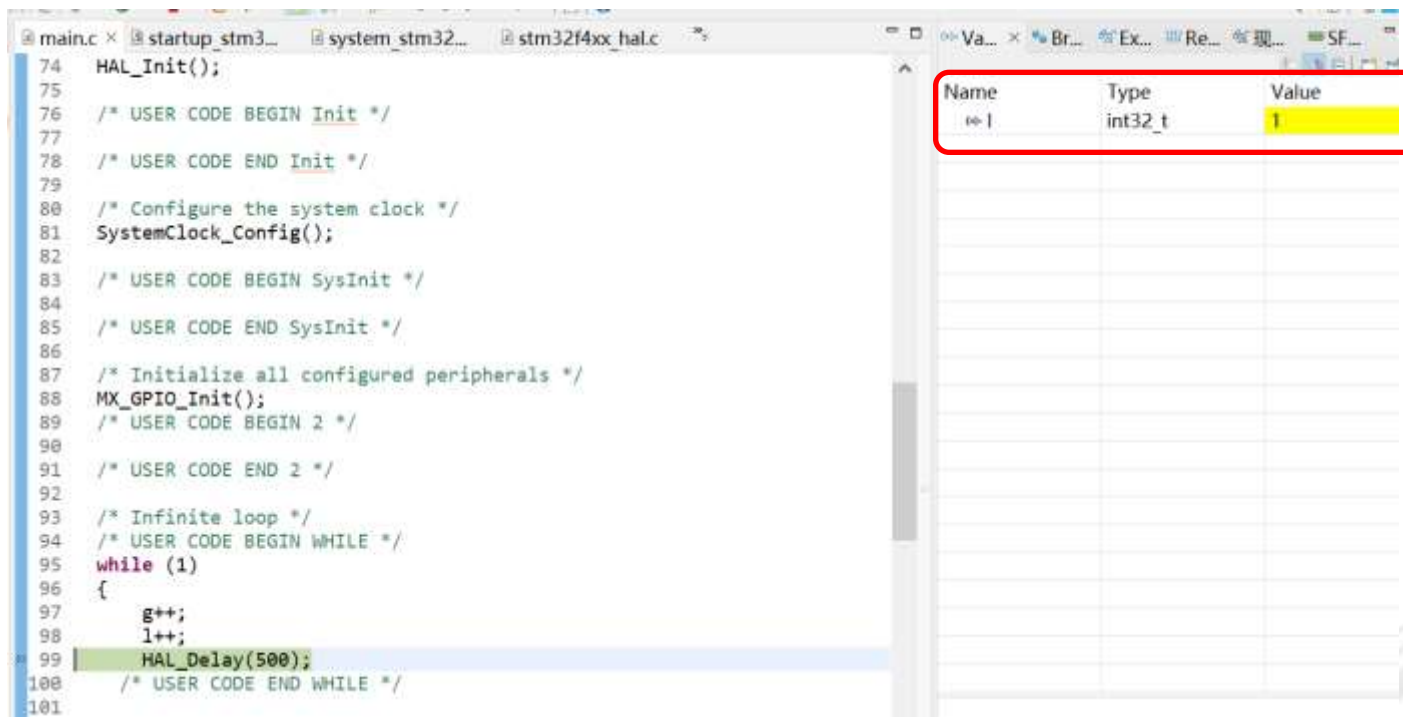
➤ 代码调试实例： — 设置一个断点

```
94  /* USER CODE BEGIN WHILE */
95  while (1)
96  {
97      g++;
98      l++;
99  HAL_Delay(500);|
100  /* USER CODE END WHILE */
```



➤ 代码调试实例：

- 点击调试按钮，进入调试视图
- 点击继续运行。程序会停在断点处。

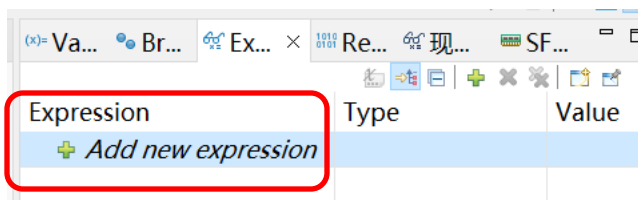


- 右侧的**Variables**窗口，会显示当前运行的函数（`main`函数）内的局部变量**`l`**的值。

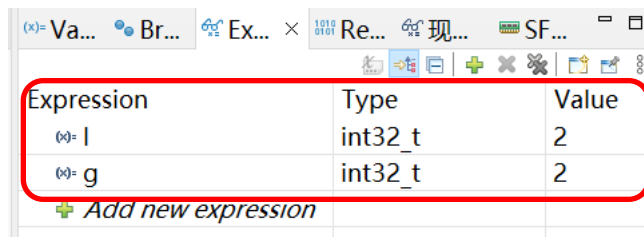


➤ 代码调试实例：

- 右侧的**Expressions**窗口中，点击**Add new expression**。



- 将**l**和**g**添加为**观察变量**。

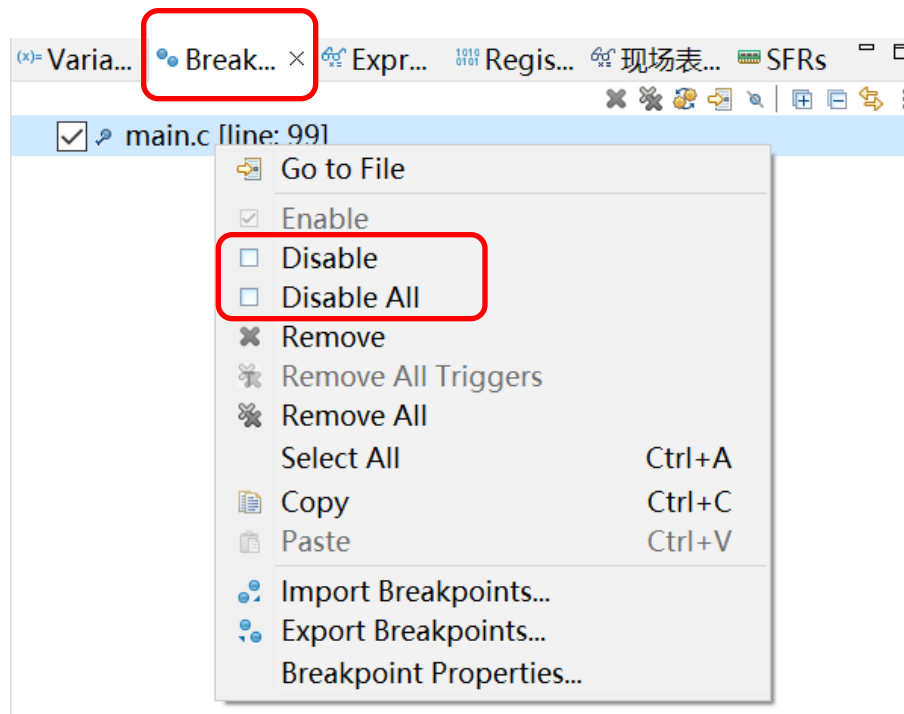


- **Expressions**窗口，不仅可以查看当前函数的**局部变量**，还可以查看全局变量的值。



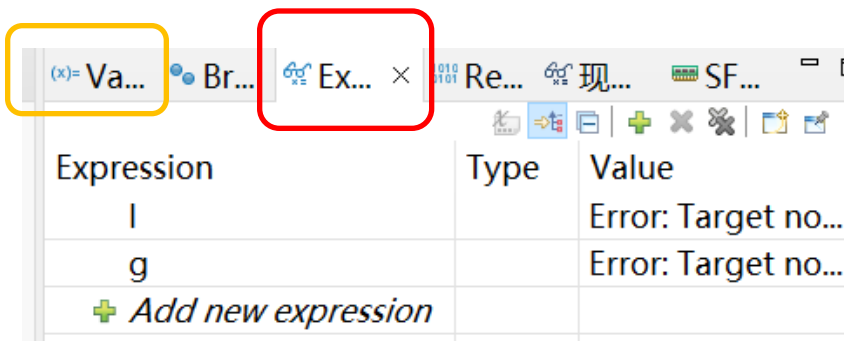
➤ 代码调试实例:

- 双击在HAL_Delay(500);那一行设置的断点，可以取消断点。
- 也可以在Breakpoints窗口中，Disable断点或者移除 (Remove) 断点。



➤ 代码调试实例：

- 点击继续运行按钮，使程序继续运行。
- 这时无论在Variables窗口还是在Expressions窗口，都无法查看变量。

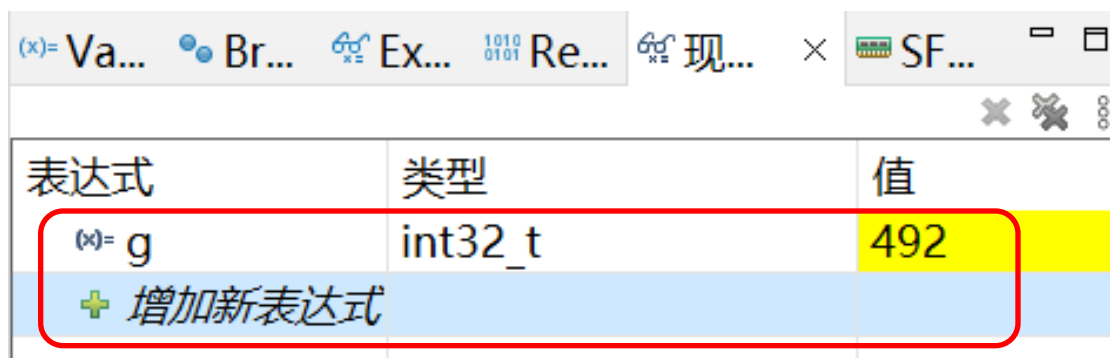


➤ 代码调试实例：

– 切换到**现场表达式窗口**。



– 点击“**增加新表达式**”，将**全局变量g**添加进窗口：



– 可以查看全局变量**g**的实时值。

– 但是局部变量**l**无法实时查看。

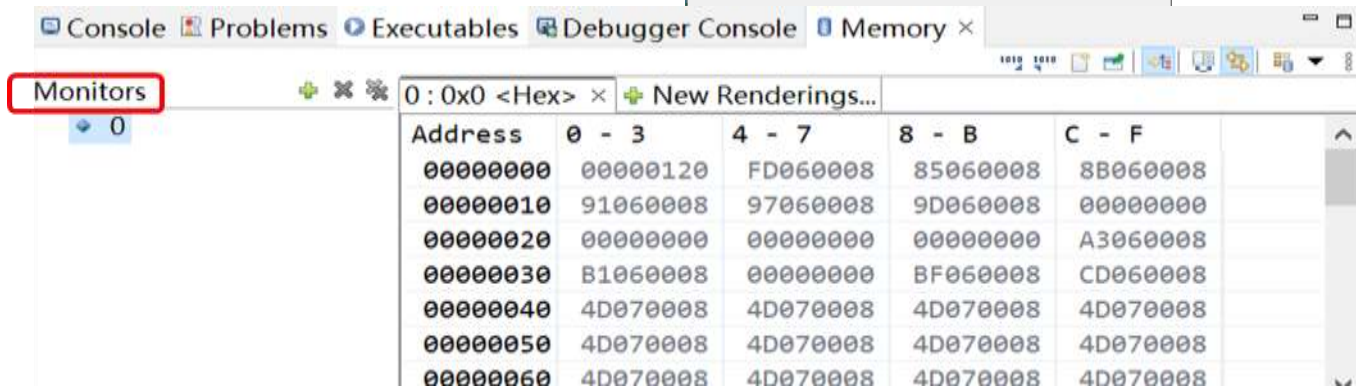
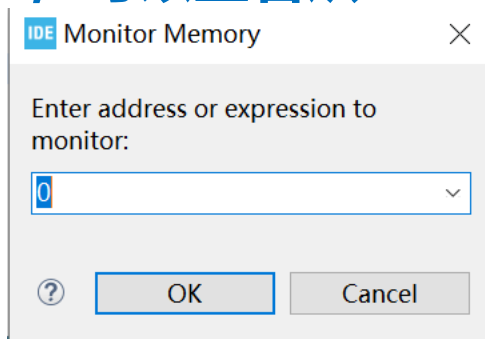


➤ 代码调试实例：

– 暂停程序后，在下方的**Memory**窗口中，可以查看存储器的值。



– 在**Monitors**选项的后面，点击添加按钮，添加要观察的存储器地址。输入0，点击OK，可以查看从0x00开始的存储器的值。

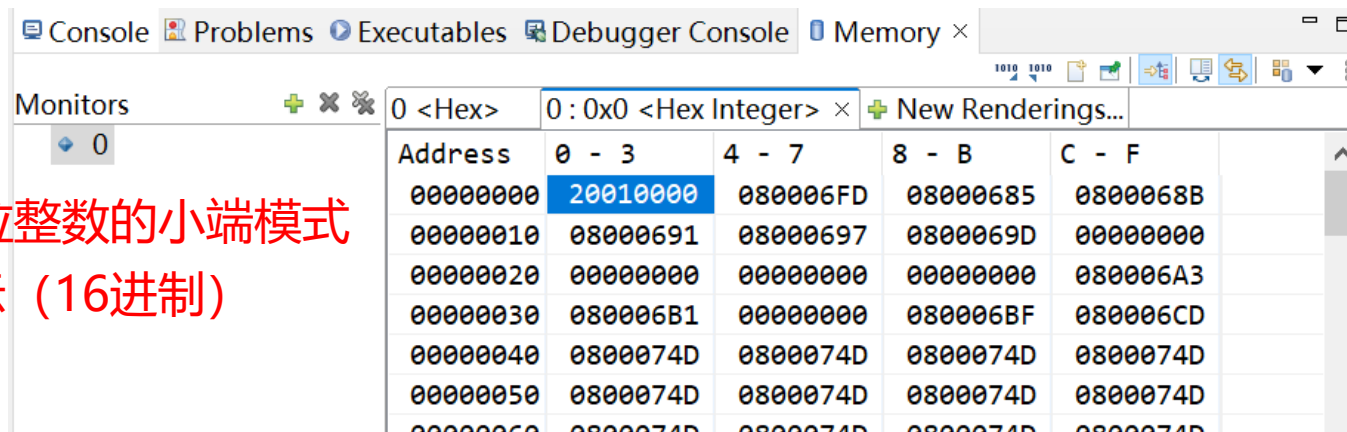
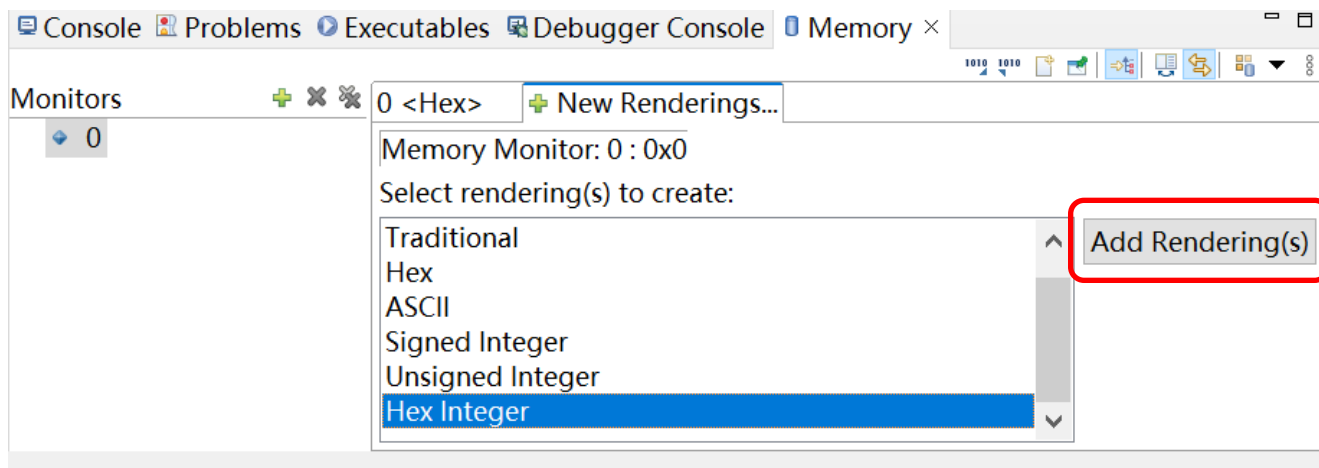


Memory窗口默认是Hex的形式显示存储器的值，也就是按照字节的存储顺序显示，而不是小端模式。



➤ 代码调试实例：

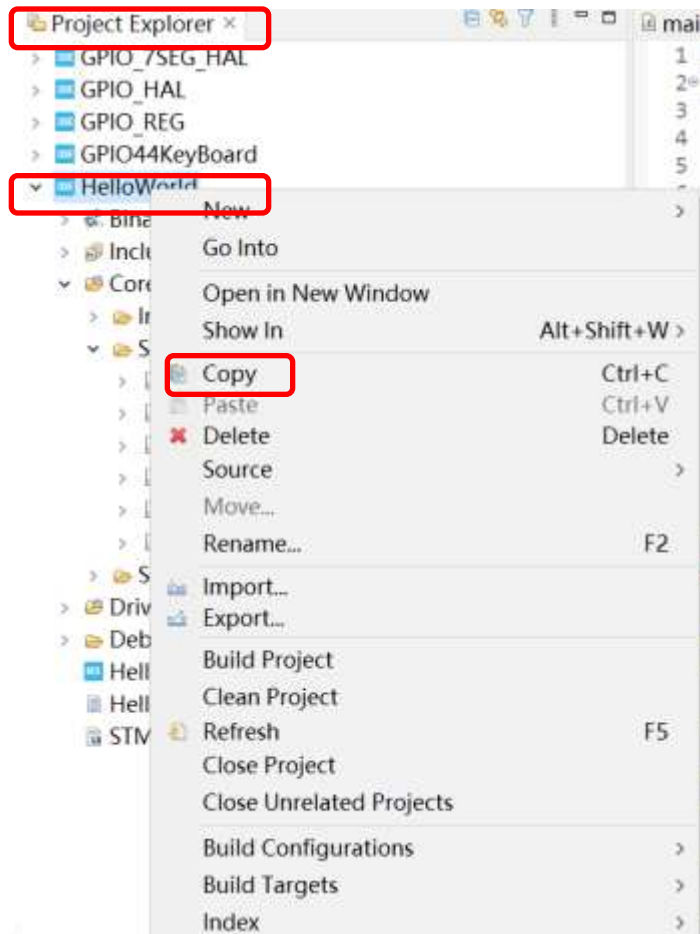
- 如果要按照小端模式显示32位整数，可以点击右上角的**New Renderings**
- 然后选择**Hex Integer**，然后点击右侧的**Add Rendering(s)**。



32位整数的小端模式 显示 (16进制)

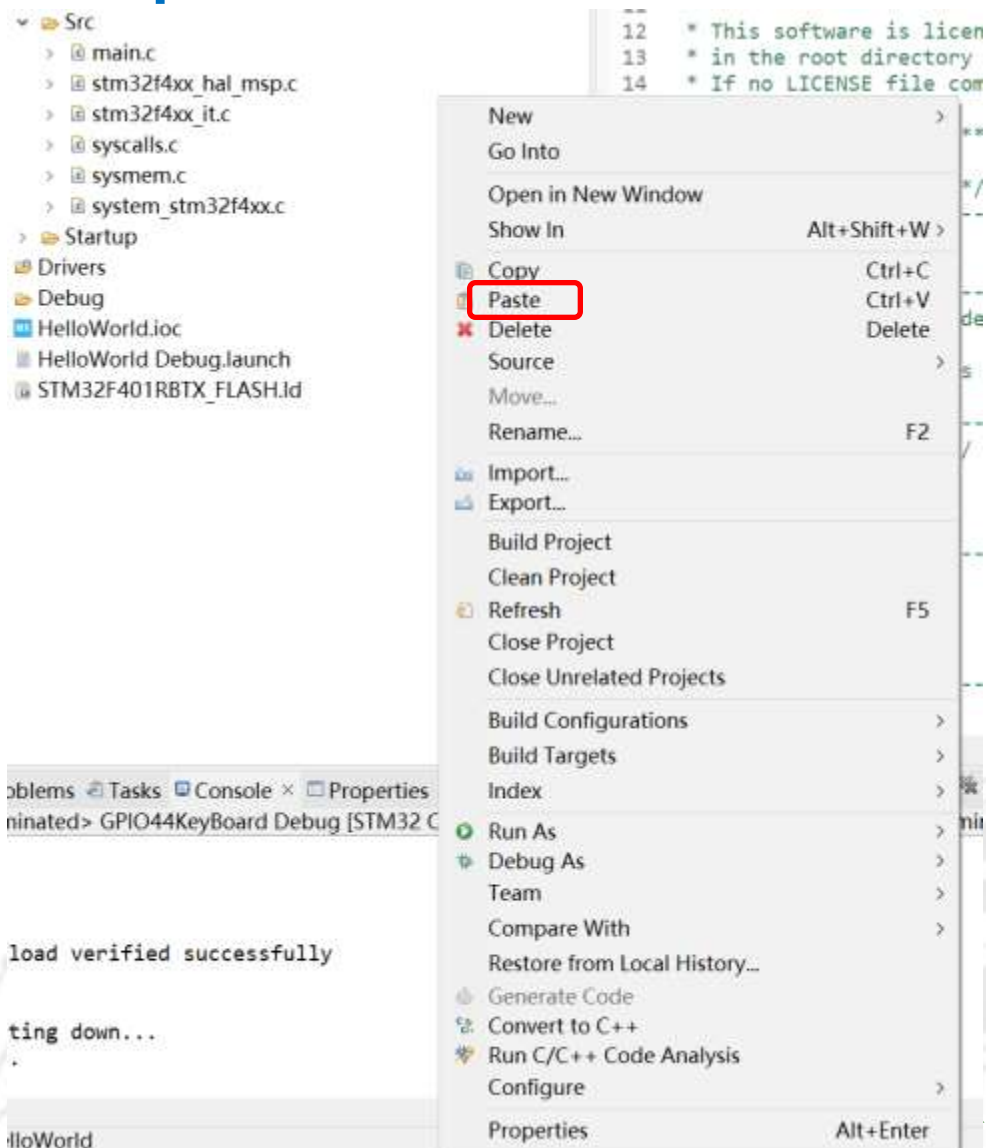
➤ 复制项目：

- 新建好HelloWorld项目后，如果要尝试不同的功能，可以以HelloWorld项目为模板，复制新的项目。
- 在左侧的Project Explorer窗口中，右键点击HelloWorld项目，在弹出的选项中，选择Copy。



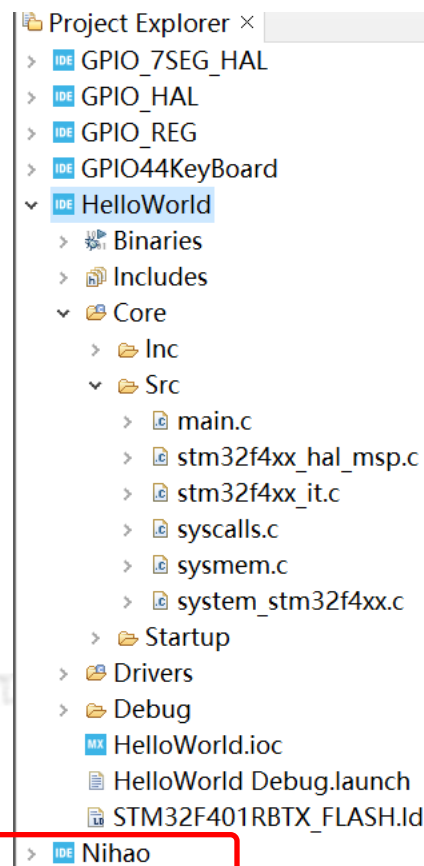
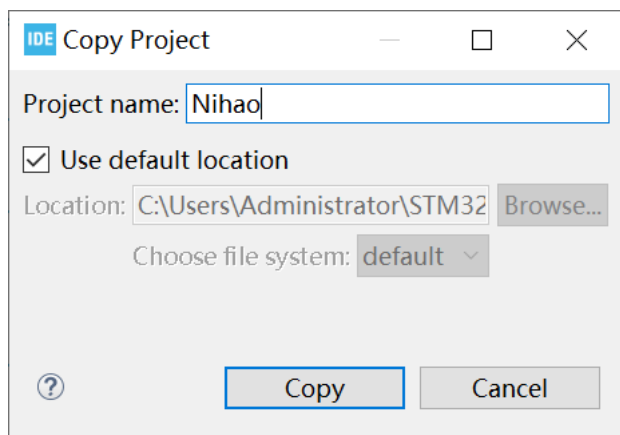
➤ 复制项目：

- 然后在Project Explorer窗口的空白处右键点击，在弹出的窗口中选择**Paste**。



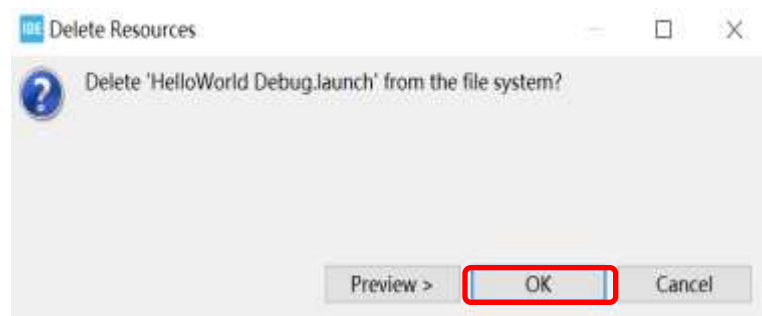
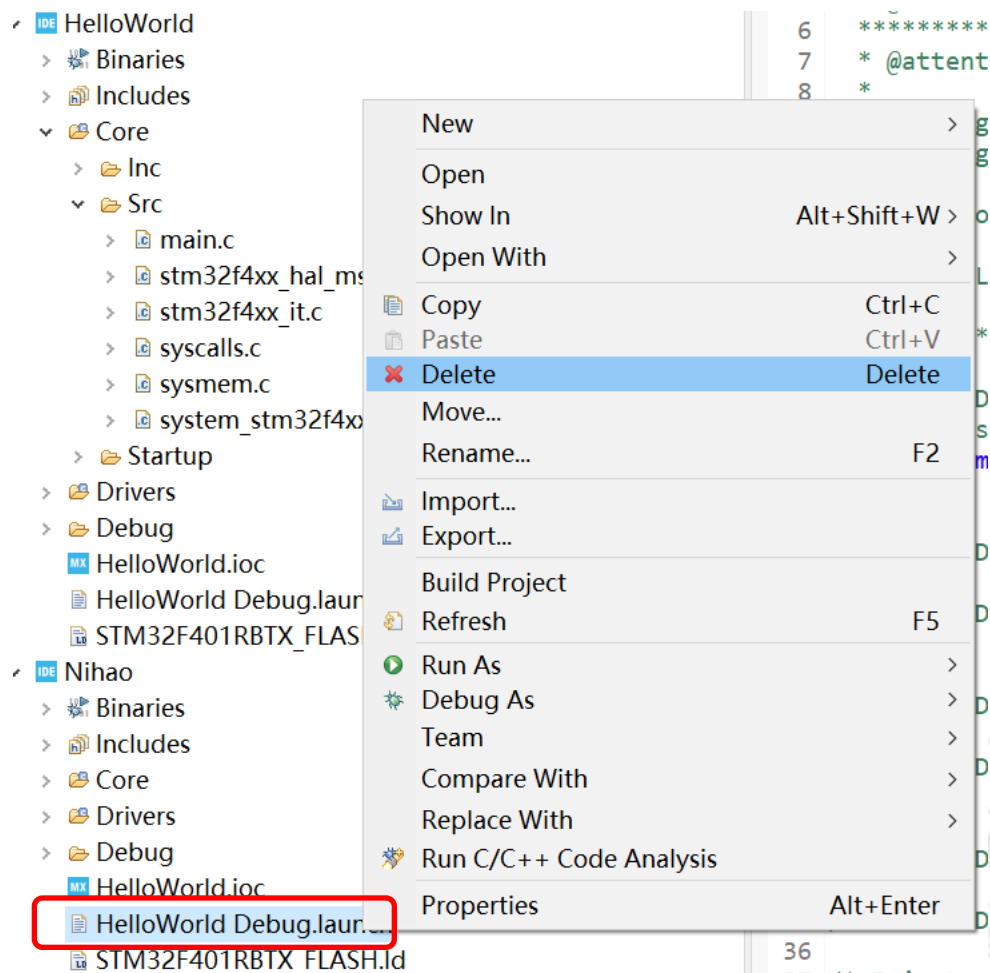
➤ 复制项目：

- 会弹出一个Copy Project对话框，要求输入新项目的名称。
- 此处将新项目命名为Nihao。
- 点击Copy，会开始创建新的工程。复制完成后，左侧的Project Explorer窗口中会多出一个Nihao项目：



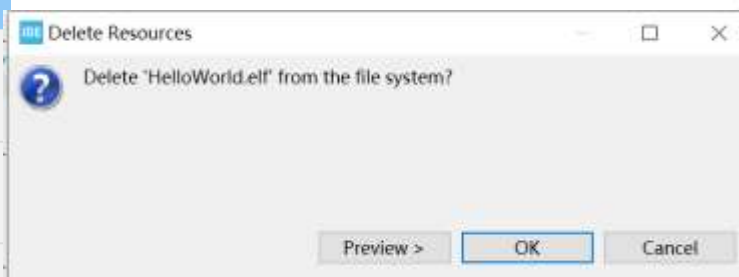
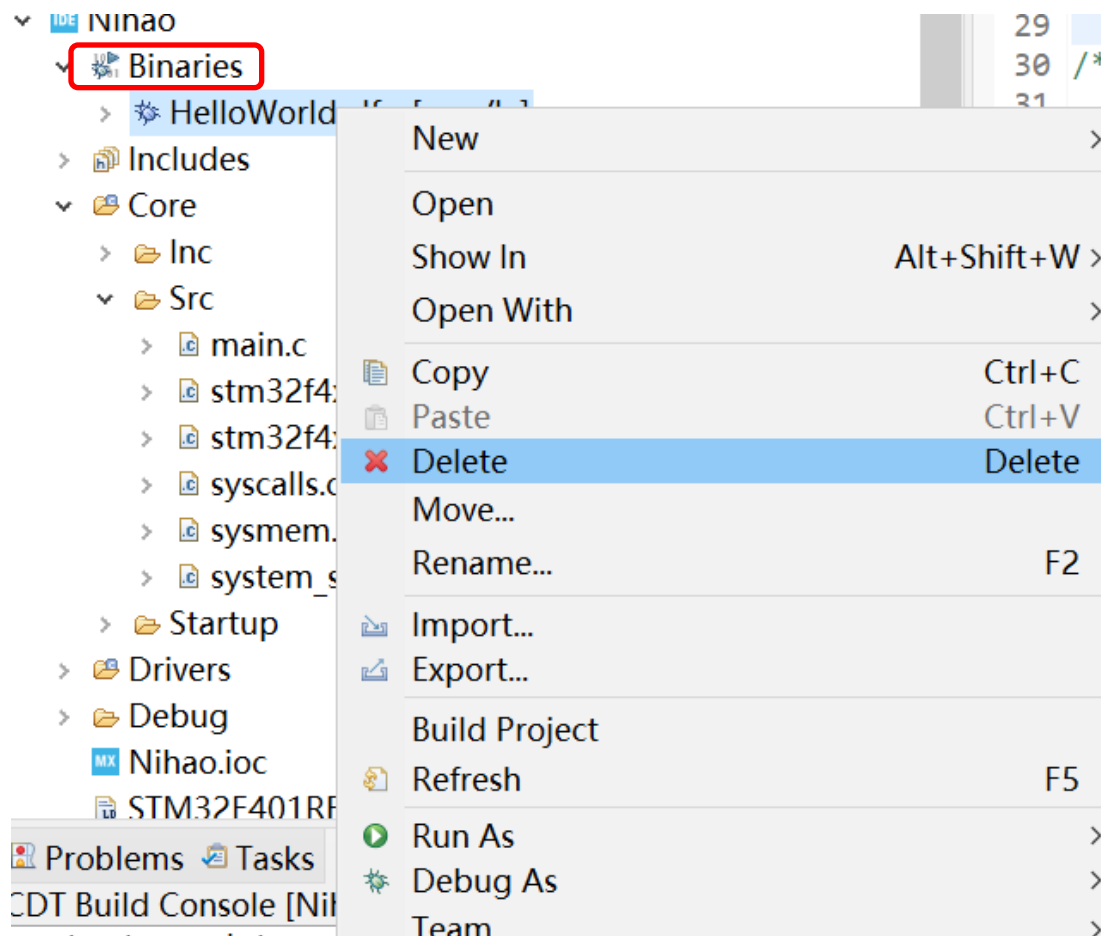
➤ 复制项目：

- 点击**Nihao**项目左侧的>符号，展开项目文件列表。
- 然后右键点击**HelloWorld Debug.launch**并选择**删除（Delete）**。



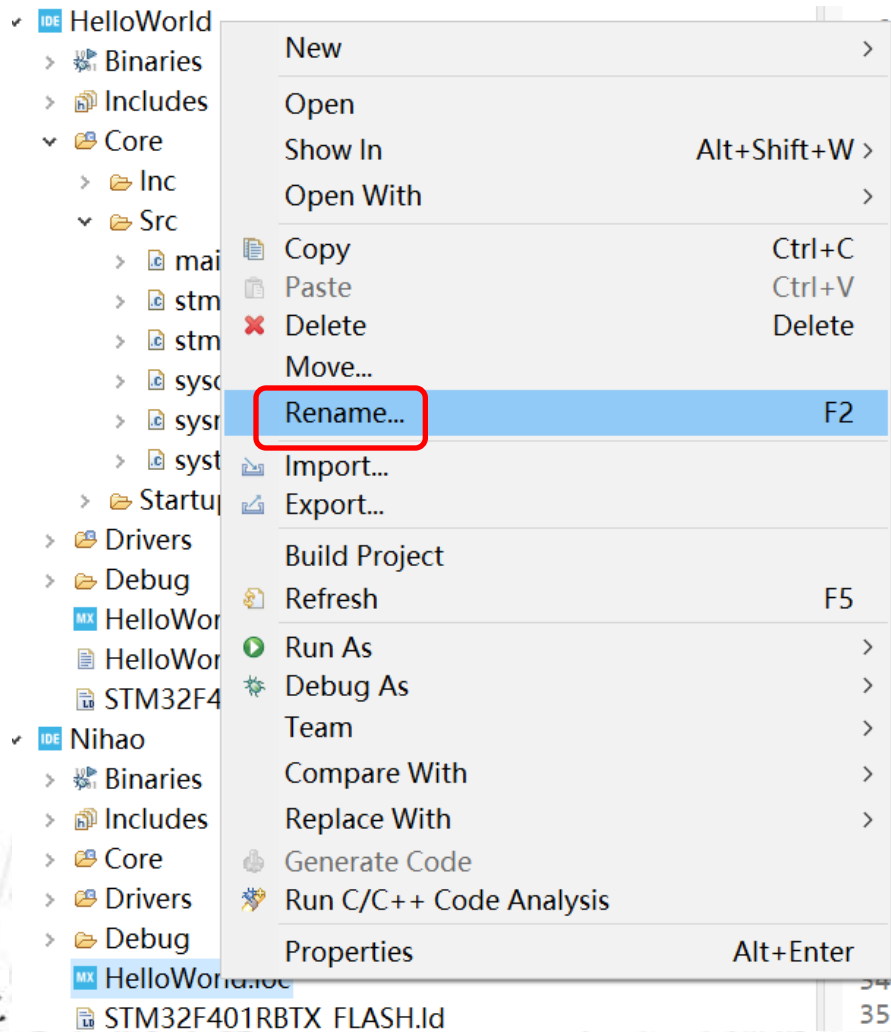
➤ 复制项目：

- 用同样的方法删除Nihao项目Binaries文件夹下面的HelloWorld.elf文件。



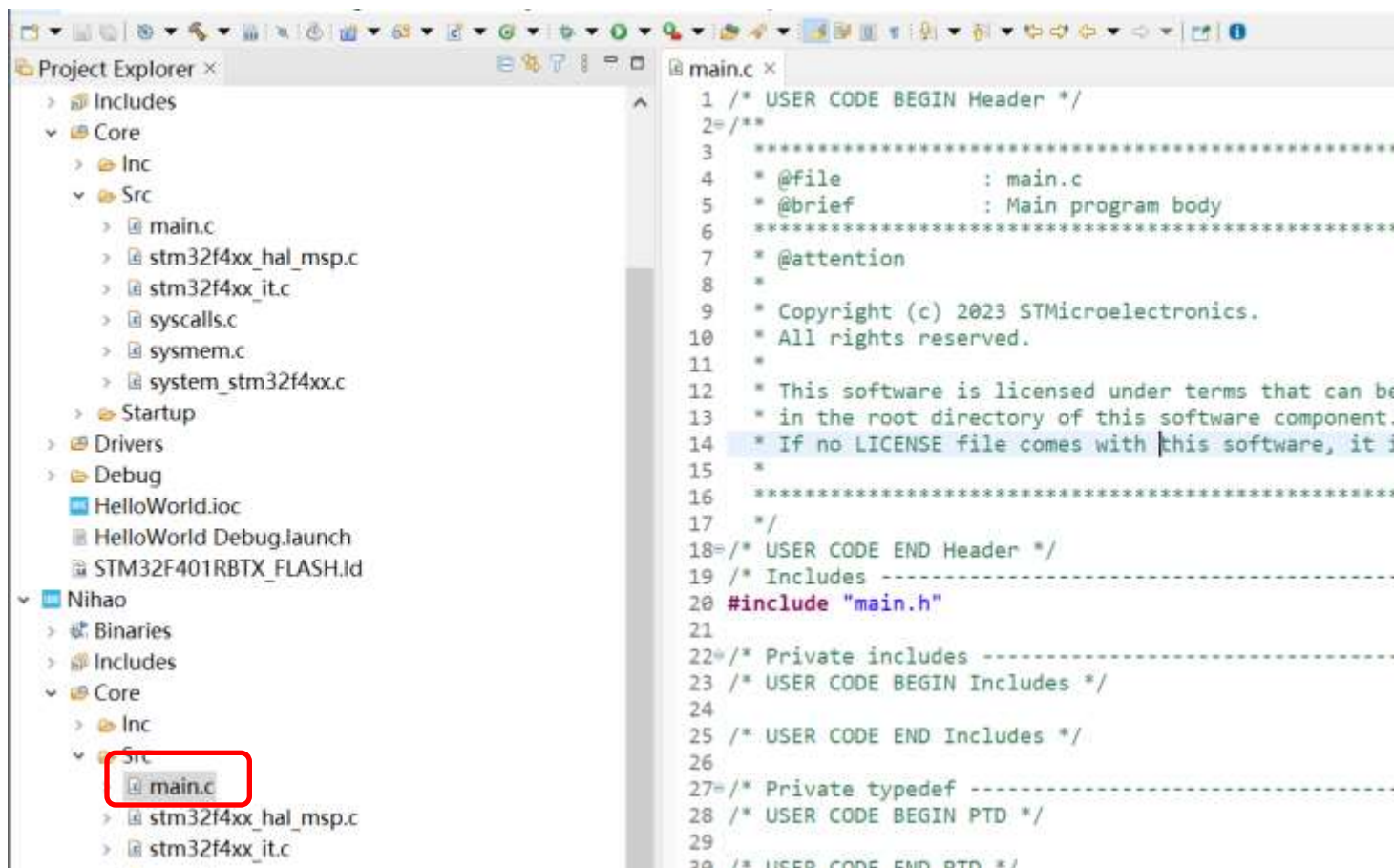
➤ 复制项目：

- 右键点击Nihao项目列表中的HelloWorld.ioc文件，选择重命名(Rename...)
- 在弹出的对话框中，将HelloWorld.ioc改名为Nihao.ioc。



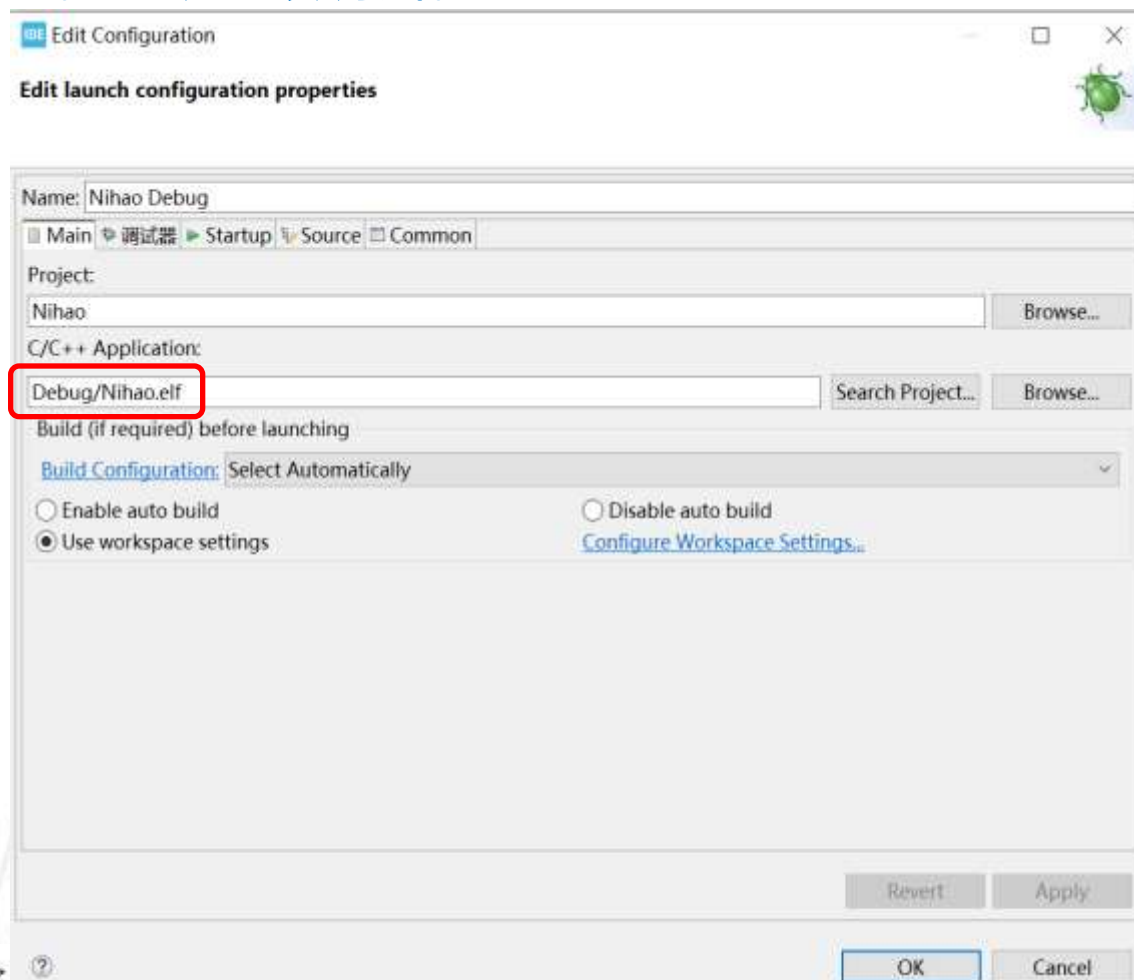
➤ 复制项目：

- 需要对Nihao工程进行调试时，双击左侧**Project Explorer**窗口中的Nihao项目下的Core->Src->**main.c**文件，打开该文件。
- **点击编译按钮**，正确编译并生成**二进制可执行文件nihao.elf**。



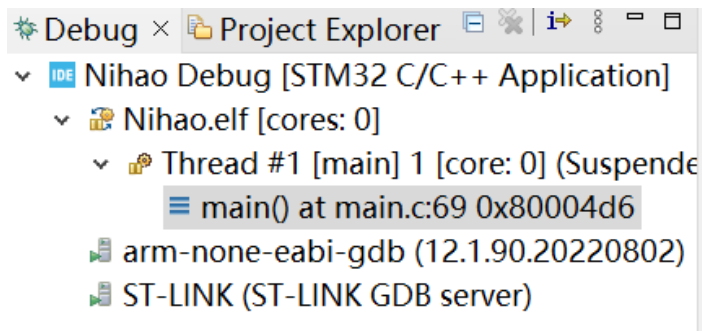
➤ 复制项目：

- 点击调试图标，会弹出调试器配置对话框。
- 确认Project为Nihao，C/C++ Applications:为Debug/Nihao.elf
- 然后点击OK。可以继续调试。

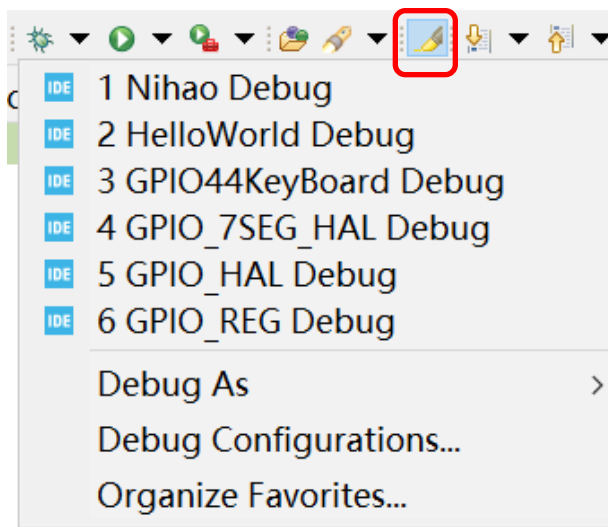


➤ 复制项目：

- 调试时，注意左侧**Debug对话框**显示的当前的调试文件，不要选错。



- 如果工作空间中有多个工程，**调试按钮图标**的右侧会有下拉菜单，用于选择调试哪个项目。



课程小结:

STM32F401RB开发板硬件资源介绍

开发板的硬件资源, 电源、时钟、重置按钮、LED灯与案件等。

STM32CubeIDE软件的安装与使用。

软件的安装

时钟树

.c文件的结构, 代码编辑方式

程序的调试方法。

- 有了开发板和编程环境, 你觉得你能做些什么?
- 查看0x08000000处的存储器数据, 看看是否与0x00000000处的数据一致。



谢谢!

