

STM32Cube高效开发教程（基础篇）

第4章 STM32CubeIDE的使用

王维波

中国石油大学（华东）控制科学与工程学院

STM32Cube高效开发教程（基础篇）

作者：王维波，鄢志丹，王钊

人民邮电出版社

2021年9月出版

如果有读者需要本书课件的PPT版本用于备课，可以给作者发邮件免费获取，并可加入专门的教学和技术交流QQ群

邮箱：wangwb@upc.edu.cn



第4章 STM32CubeIDE的使用

4.1 安装STM32CubeIDE

4.2 基本概念和MCU固件库设置

4.3 编码场景的界面功能和操作

4.4 CubeMX导出项目文件组成

4.5 项目管理、编译和下载调试

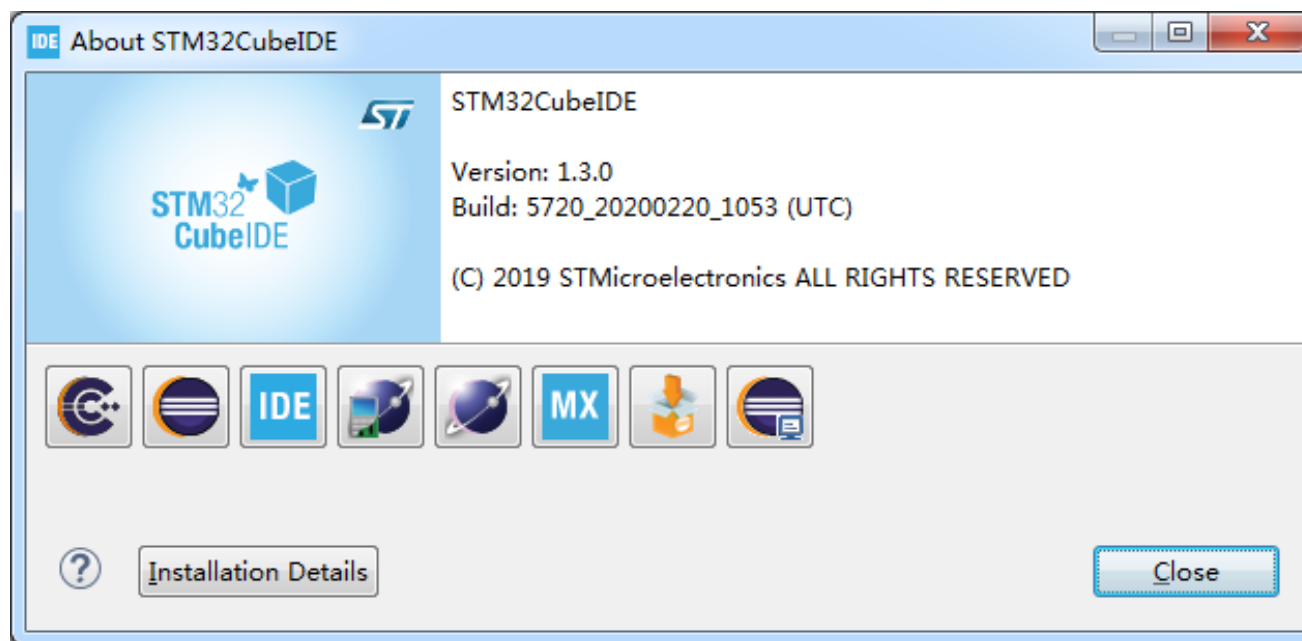
4.6 使用内置的CubeMX

4.7 CubeIDE使用偏好设置

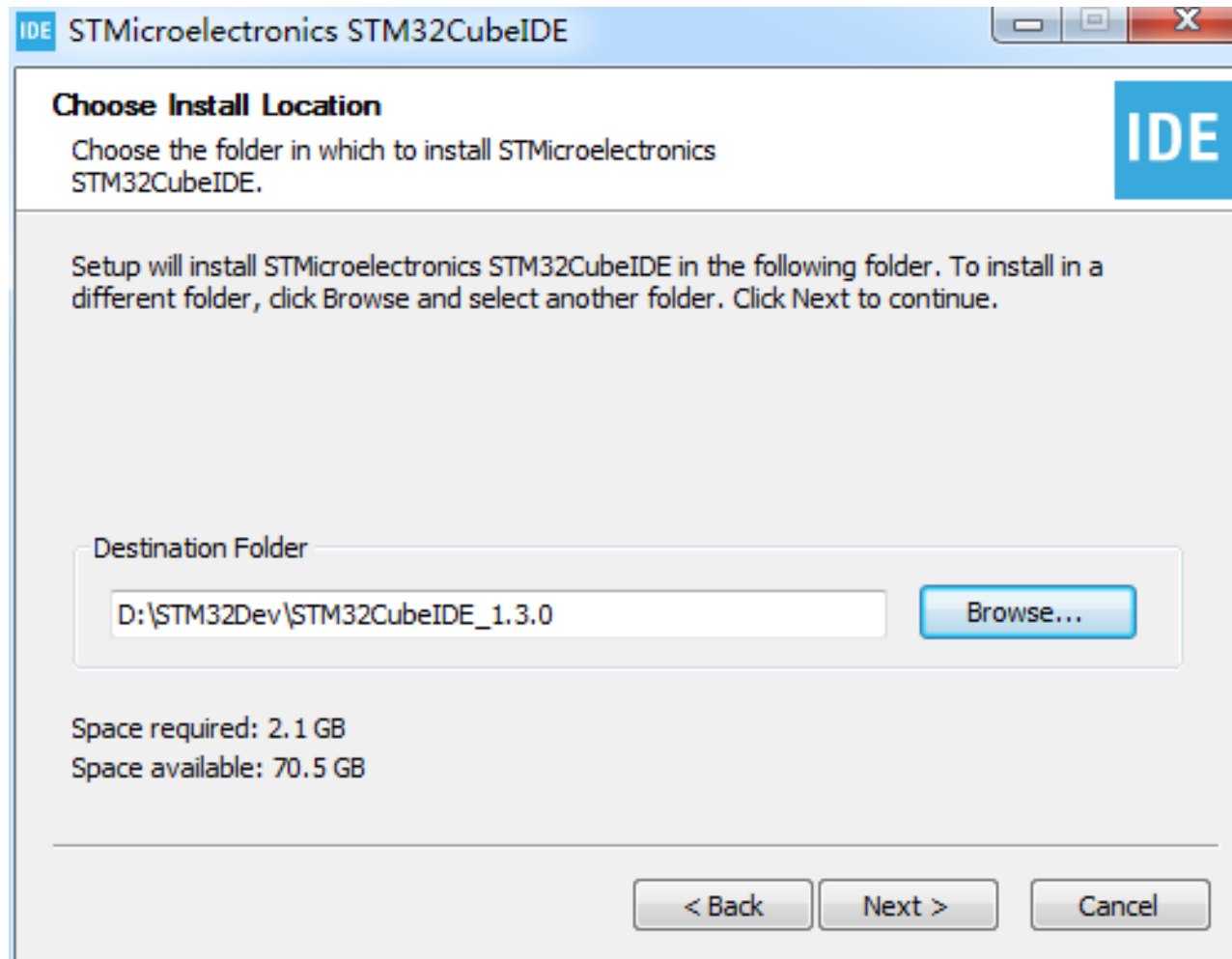
4.8 HAL库使用的一些问题

4.1 安装STM32CubeIDE

- 2019年4月才发布STM32CubeIDE 1.0.0
- CubeIDE是基于TrueSTUDIO的，使用GCC编译器
- CubeMX5.2以后版本才支持CubeIDE
- CubeIDE中集成了CubeMX

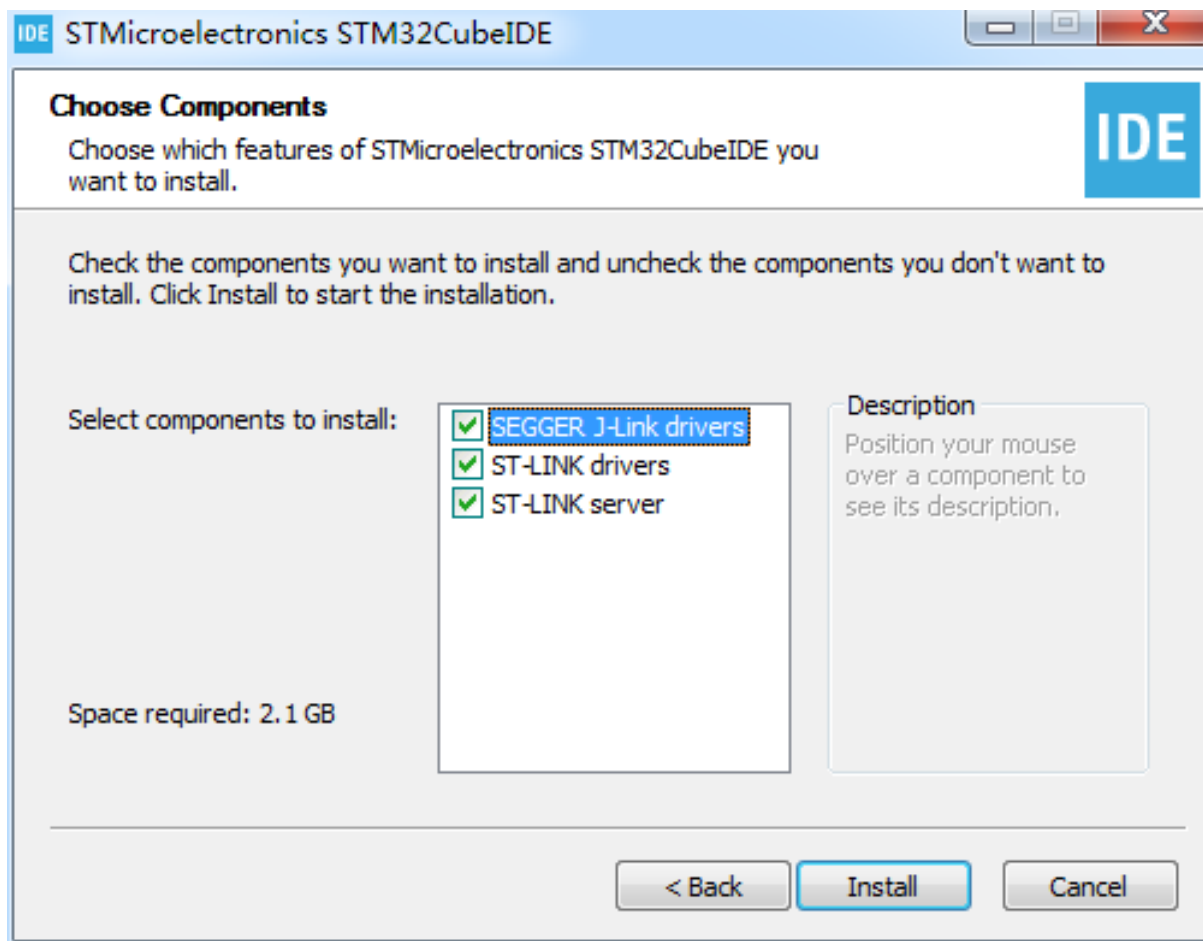


与CubeMX安装到同一个根目录下，如 D:\STM32Dev\



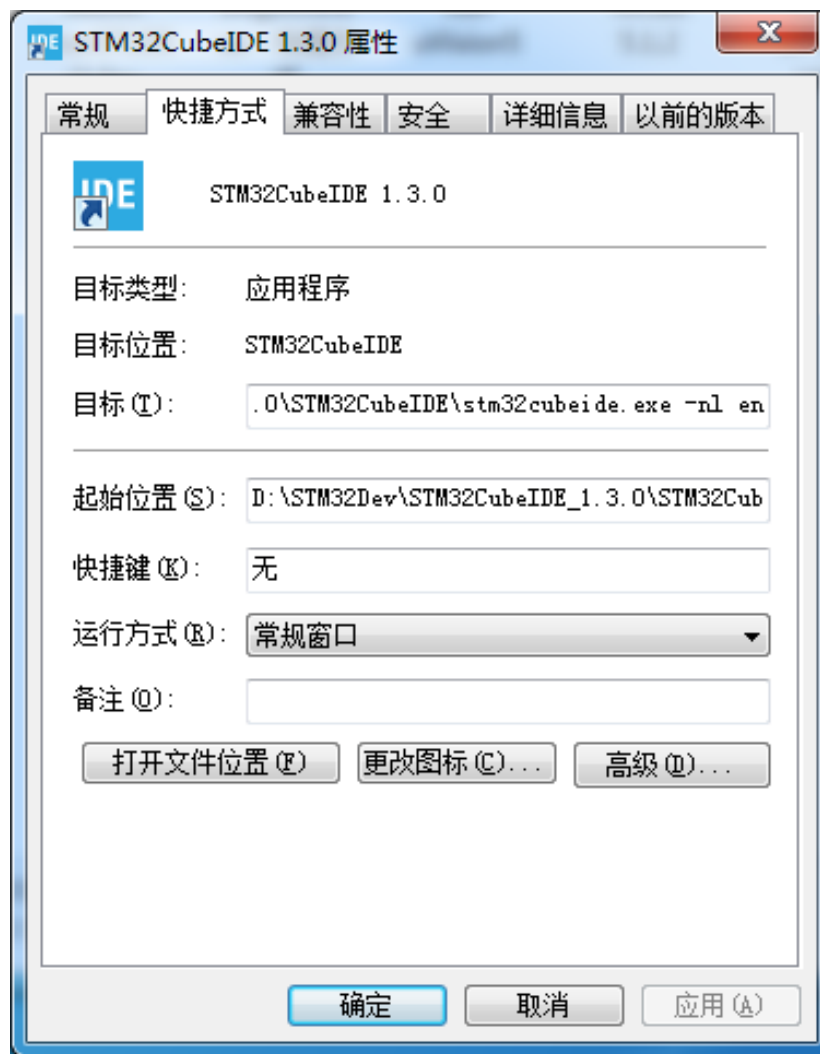
安装时选择安装仿真器驱动程序

CubeIDE目前只支持ST-LINK和J-LINK仿真器



为CubeIDE主程序设置命令行设置参数，设置为英文界面

增加: -nl en



4.2 基本概念和MCU固件库设置

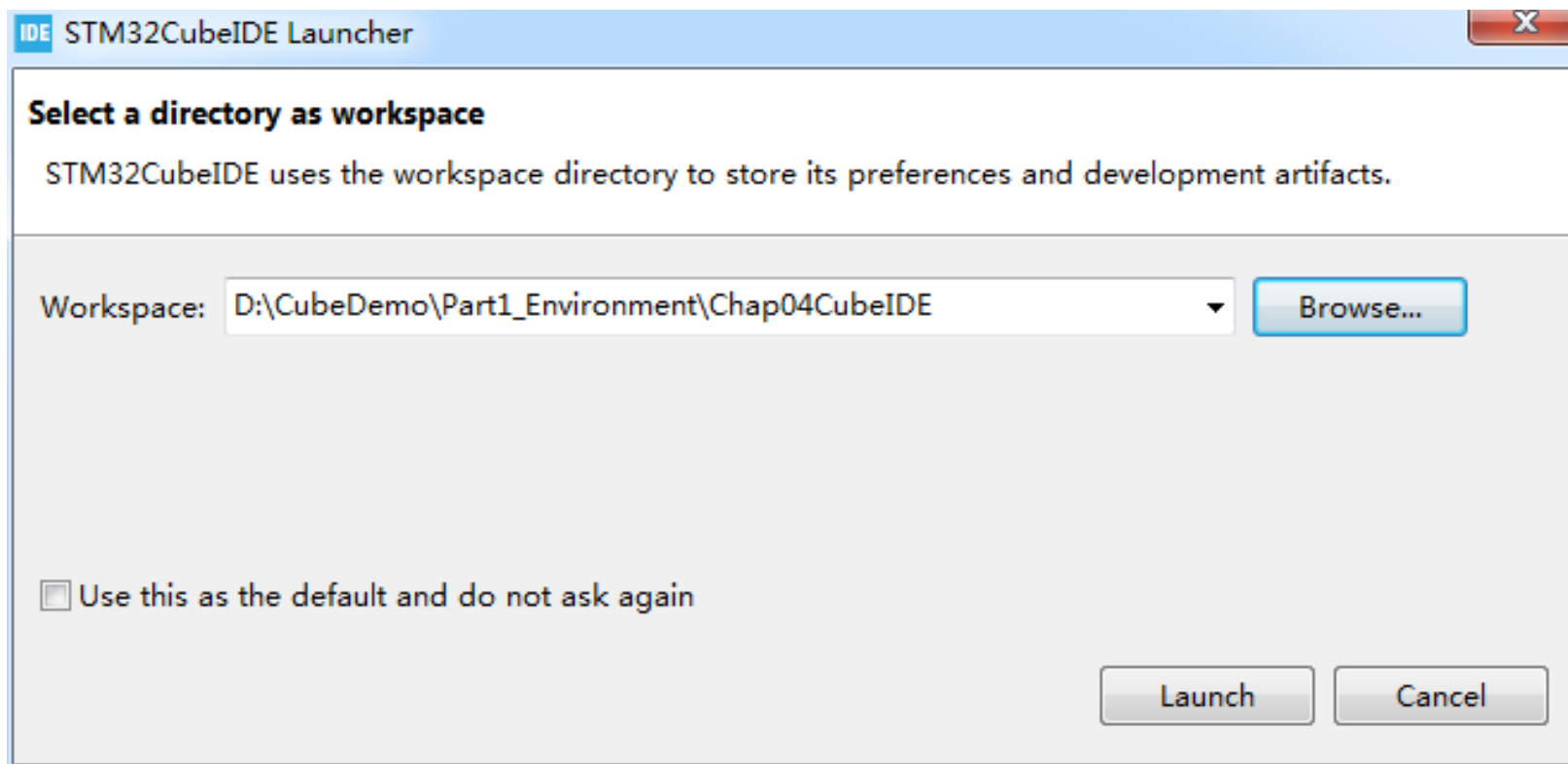
4.2.1 启动软件

4.2.2 打开项目

4.2.3 CubeIDE的一些基本概念

4.2.4 STM32固件库设置

4.2.1 启动软件



启动CubeIDE时选择工作目录，作为当前workspace

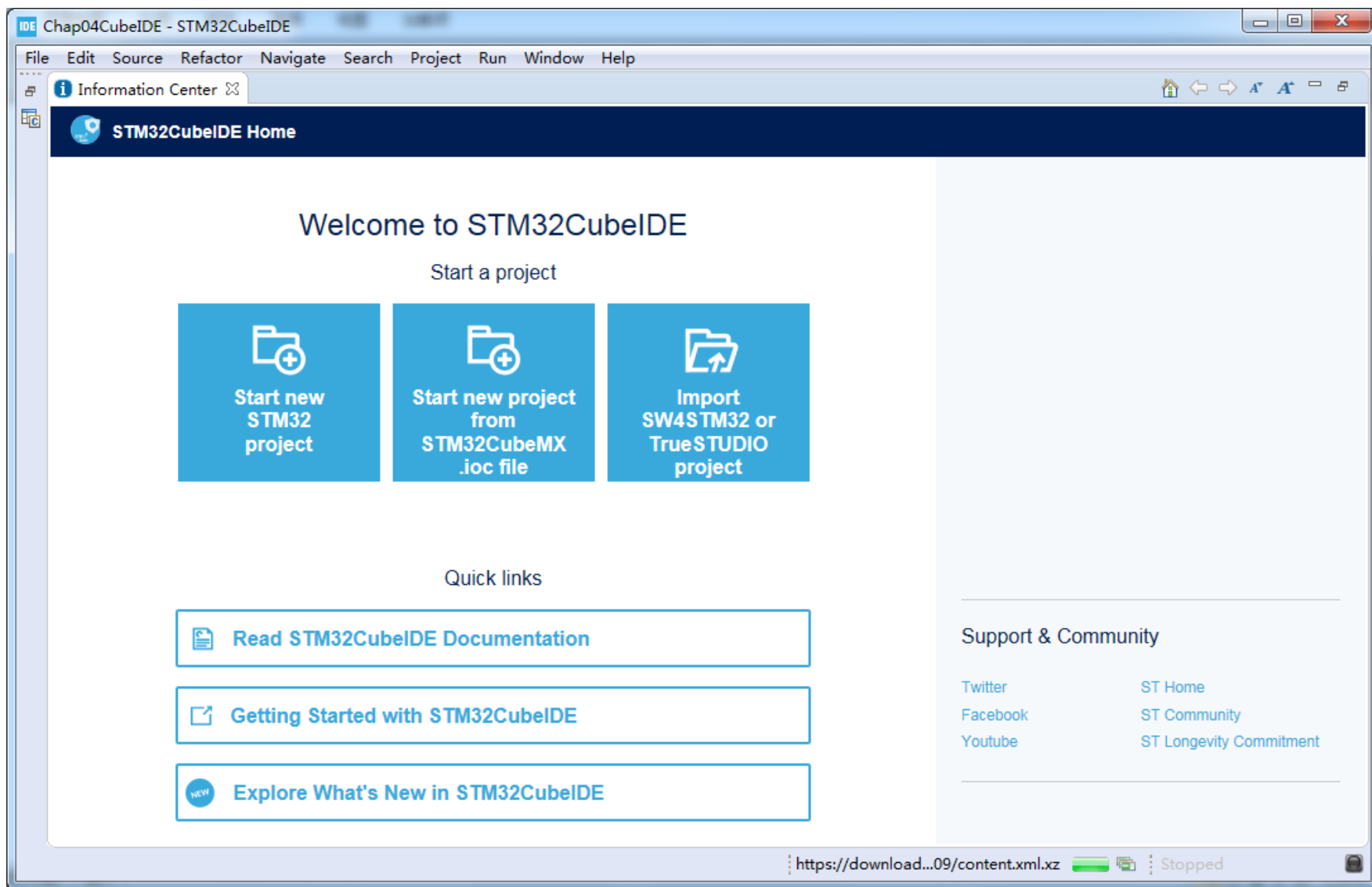
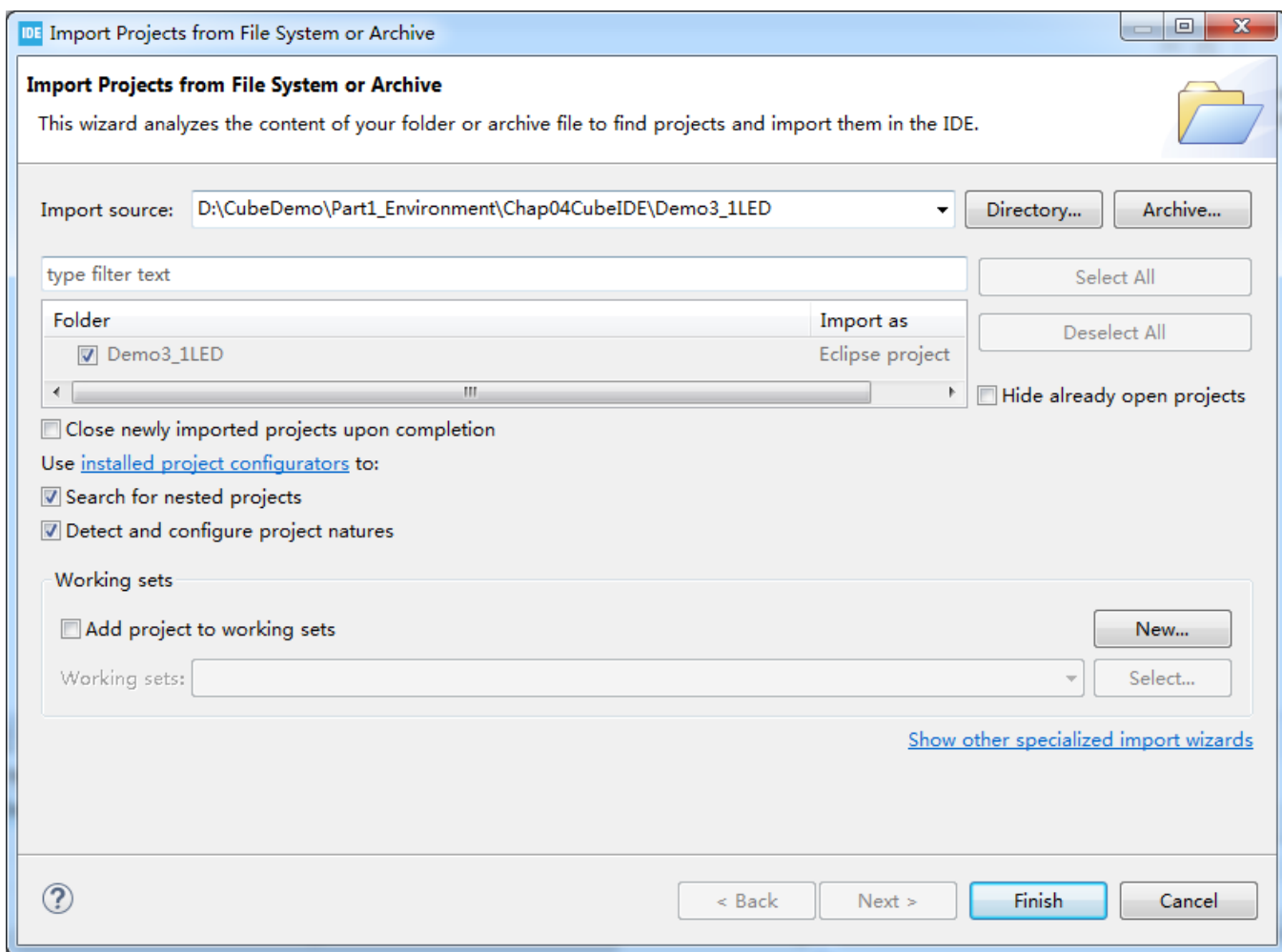


图4-4 CubeIDE的信息中心页面

4.2.2 打开项目

导入前一章用CubeMX生成的项目Demo3_1



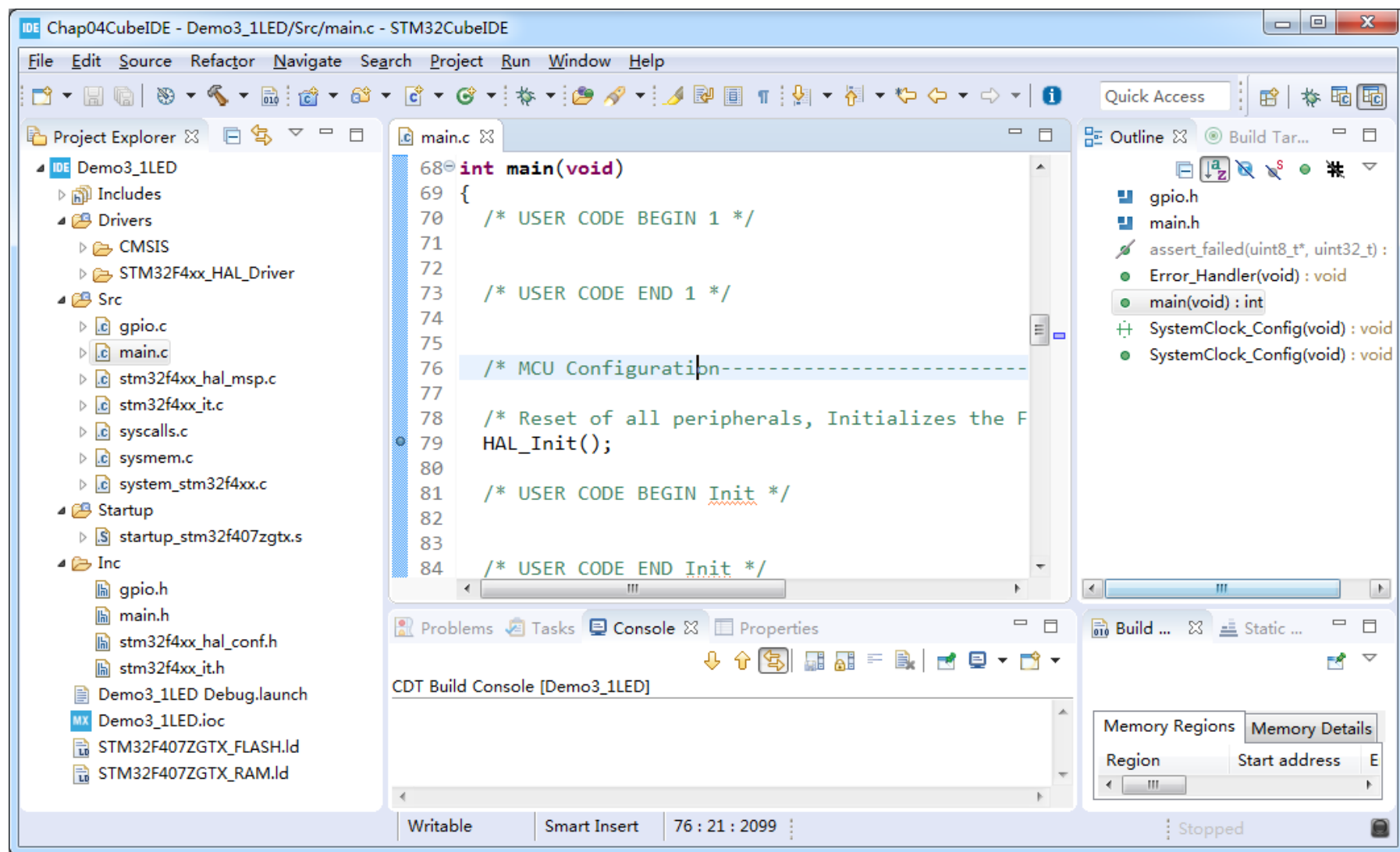







图4-7 打开了项目Demo3_1LED后的CubeIDE界面

4.2.3 CubeIDE的一些基本概念

CubeIDE是基于Eclipse的IDE软件。

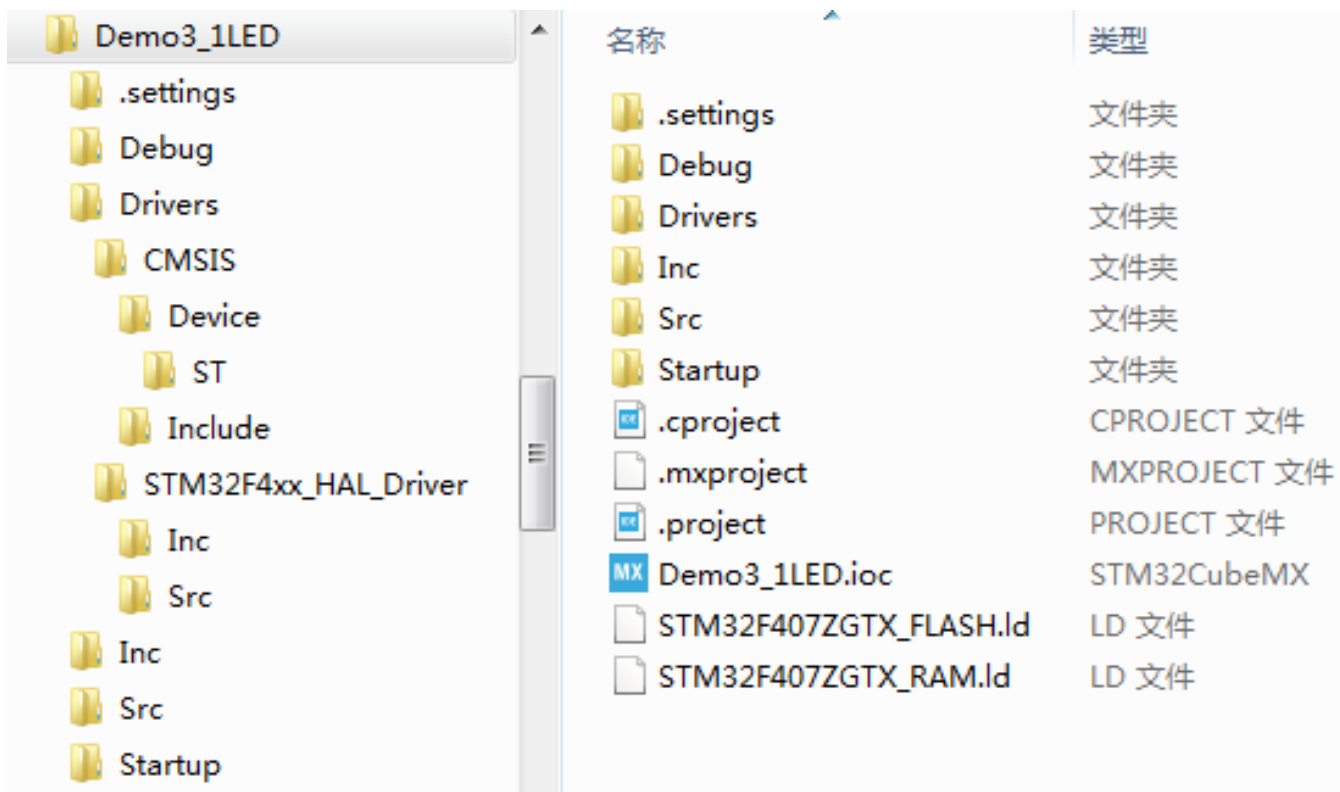
1. 工作空间（Workspace）

工作空间是多个项目（Project）的集合，一个工作空间就是一个实际的目录，包含多个项目的子目录和工作空间自建的文件夹。

名称	类型
 Chap04CubeIDE	
 .metadata	文件夹
 Demo3_1LED	文件夹
 Demo4_2EmbedMX	文件夹
 RemoteSystemsTempFiles	文件夹

2. 项目（Project）

一个项目（Project）就是一个文件夹，项目的名称就是文件夹的名称。一个项目包含很多文件和子目录，例如项目 Demo3_1LED根目录下的文件和子目录构成如图所示。



3. 视图（View）

IDE窗口里的子界面称为视图（View），一个视图就是实现一些功能的界面。例如显示项目文件组成的Project Explorer视图，显示文件内容的Outline视图。

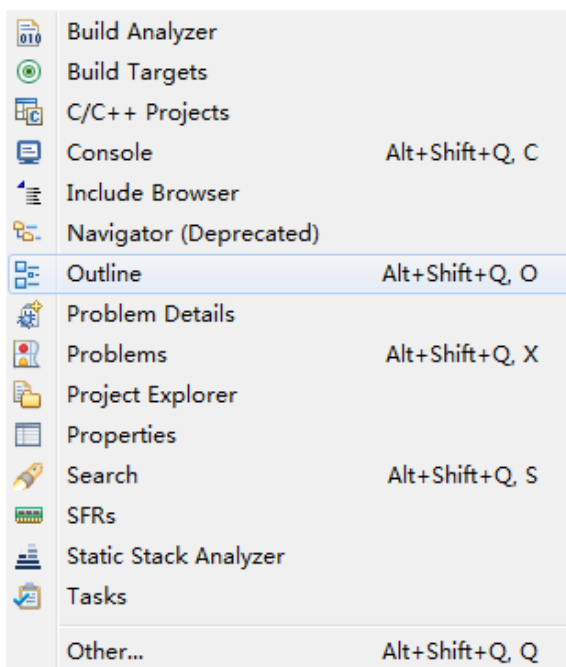


图4-10 常用视图菜单

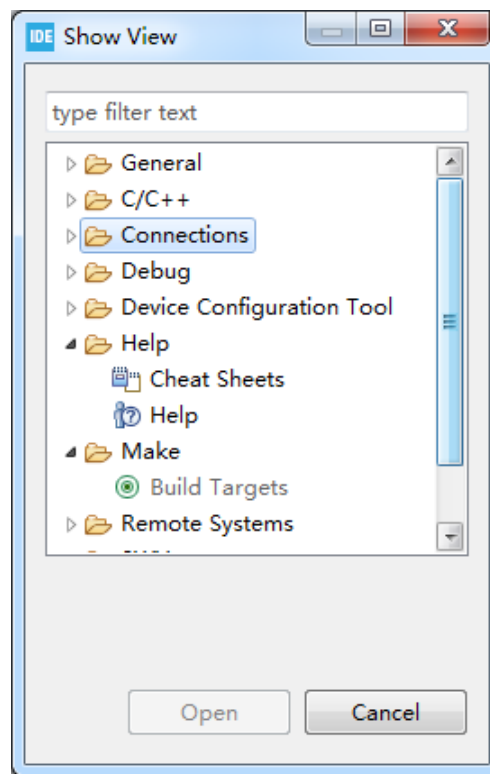


图4-11 Show View对话框

4. 场景（ Perspective ）

场景就是多个视图组成的一种工作界面，一个场景一般对应于一种工作需求，例如：

- C/C++ Code Editing, C/C++编码场景，最常用的场景
- Debug, 调试场景，用于程序调试时的工作场景

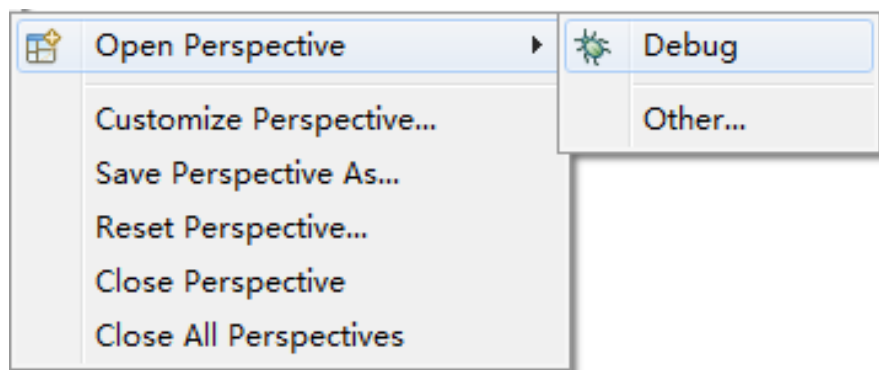


图4-12 场景管理子菜单

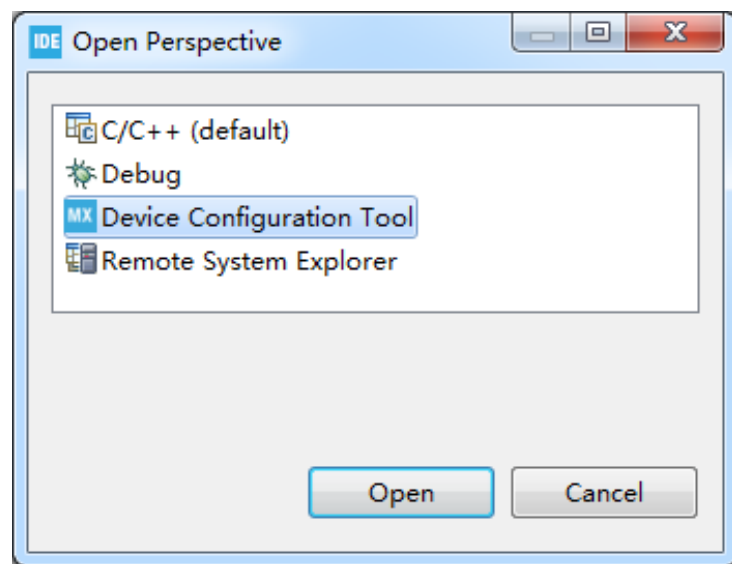
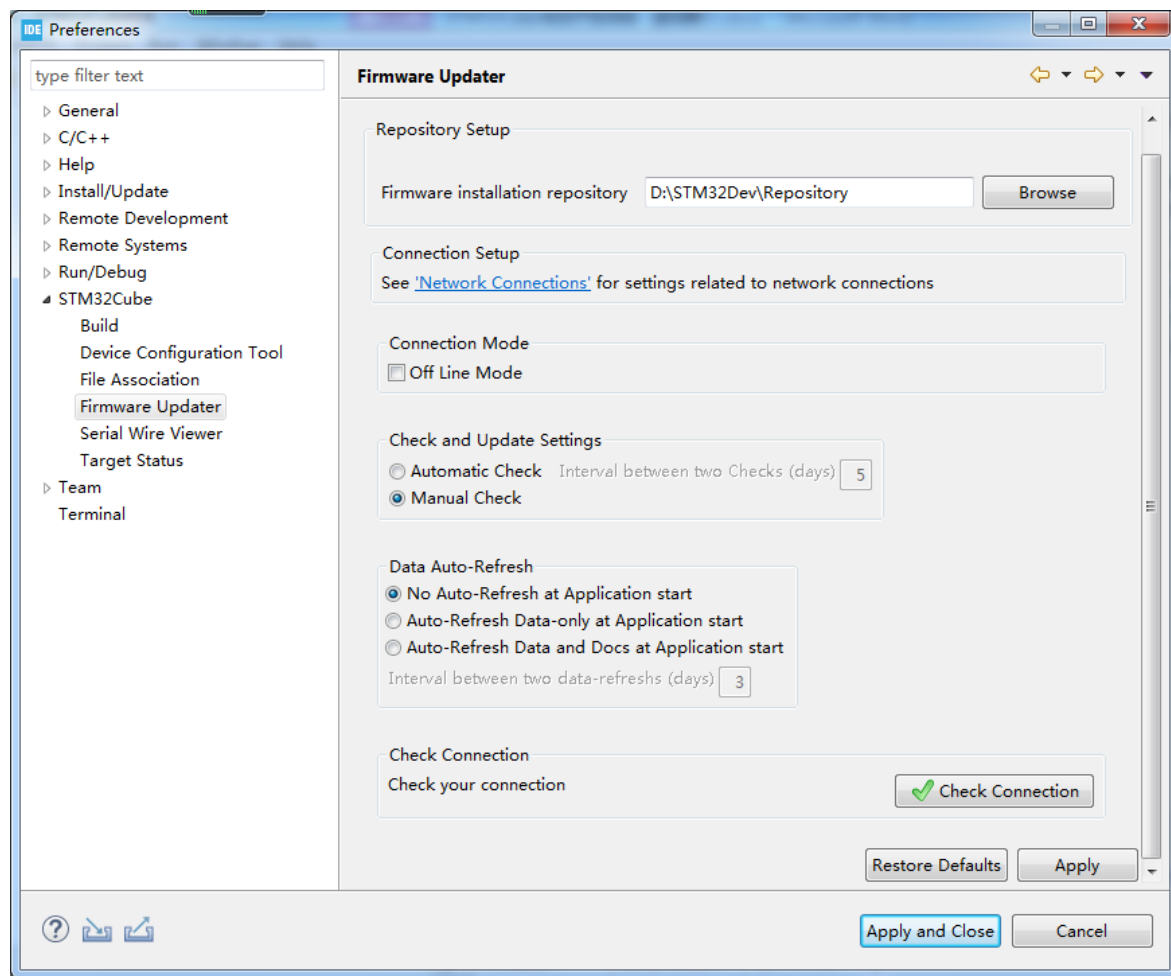


图4-13 Open Perspective对话框

4.2.4 STM32固件库设置

主菜单项 Window → Preference 会打开软件设置对话框



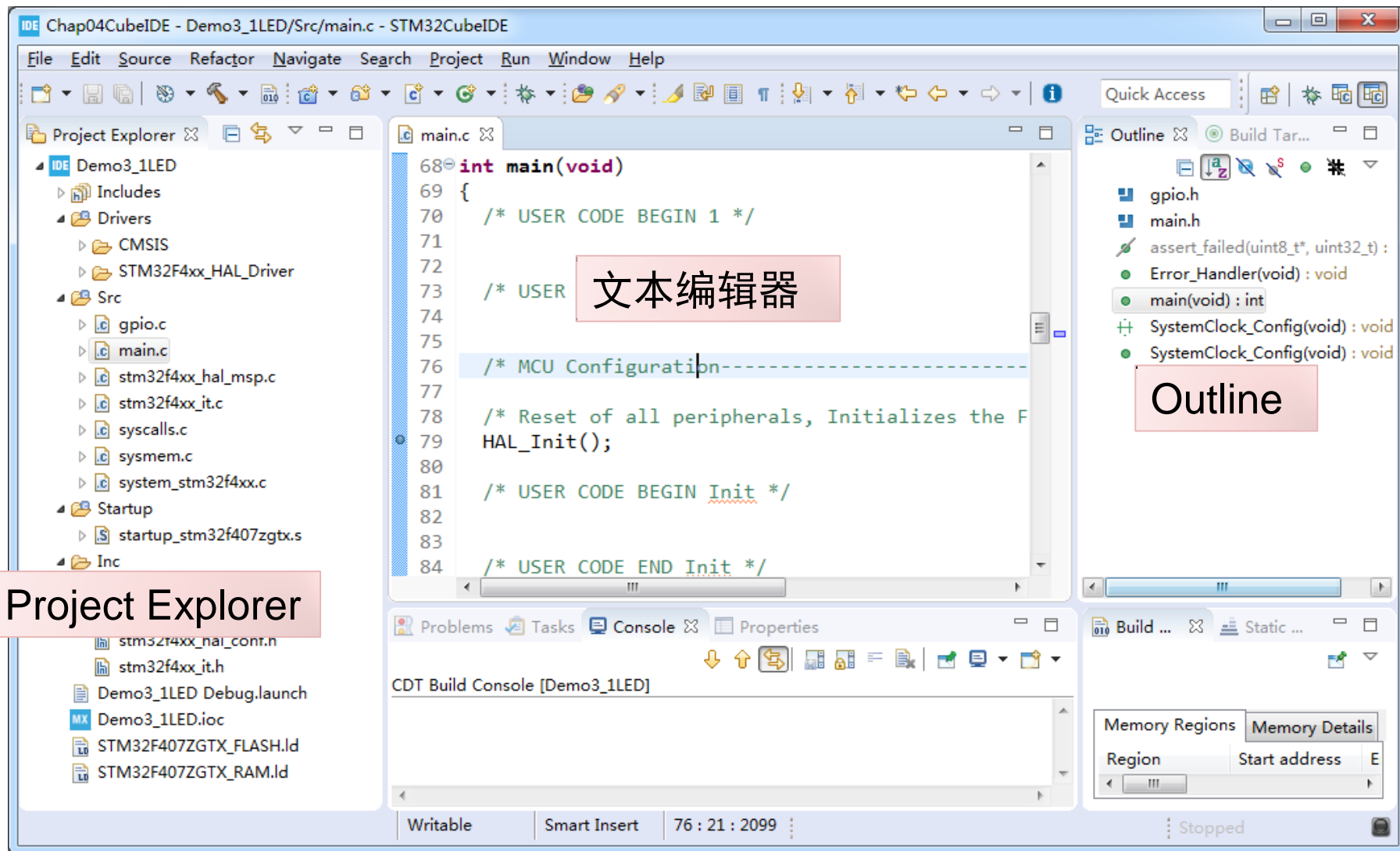
4.3 编程场景的界面功能和操作

4.3.1 主要的视图

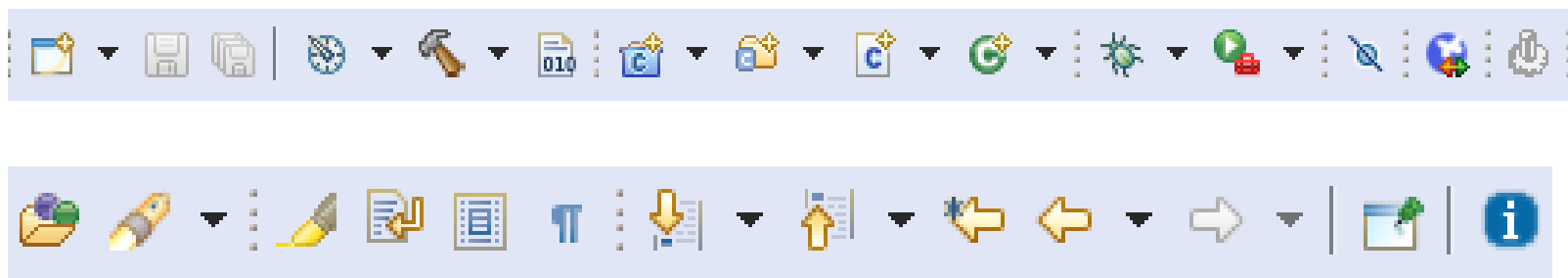
4.3.2 工具栏按钮功能

4.3.3 文本编辑器功能操作

4.3.1 主要的视图



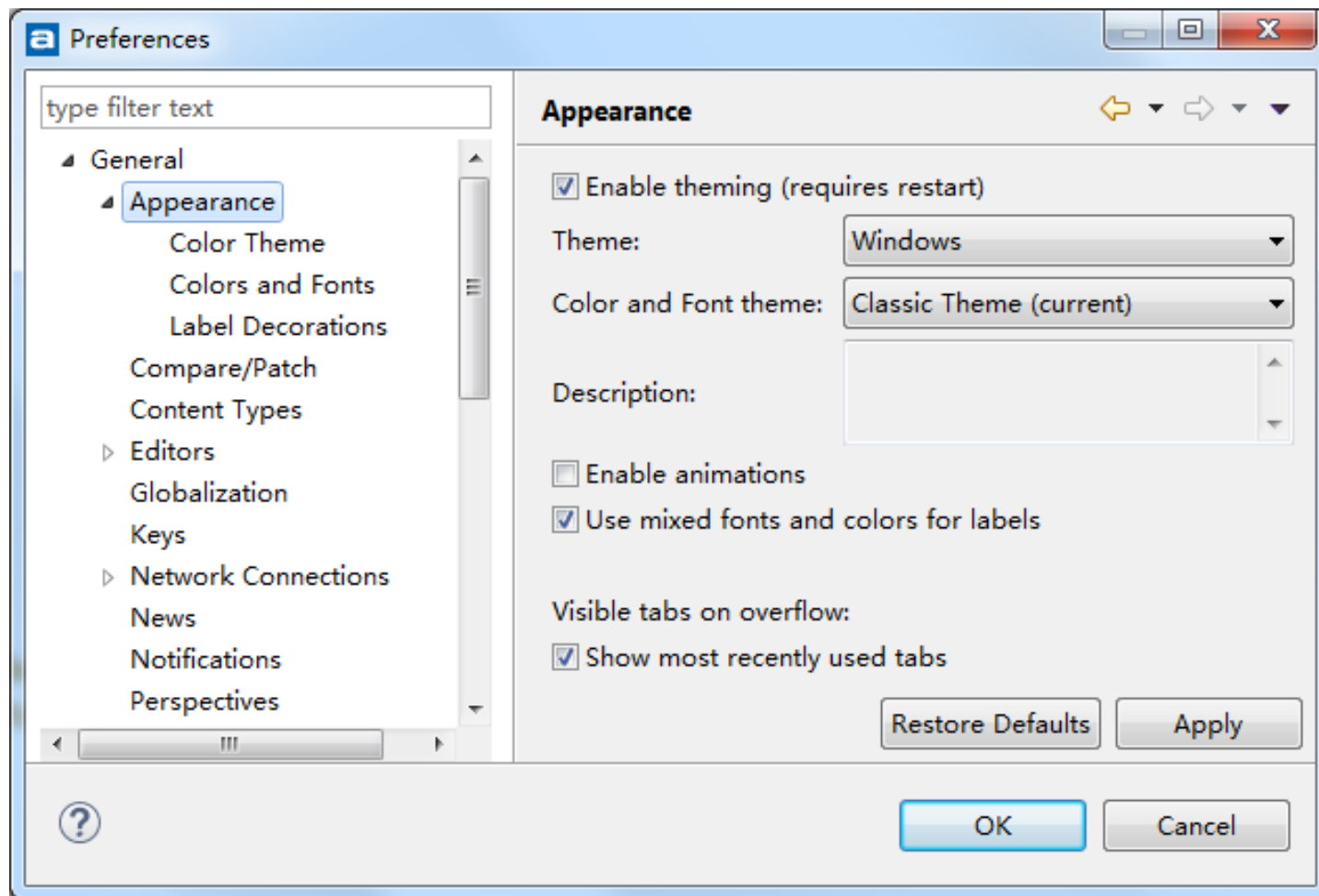
4.3.2 工具栏按钮功能



【打开软件实际操作演示，每个图标的意义见讲义】

5.2.3 文本编辑器功能操作

1. 界面主题和编辑器颜色主题



2. 代码追踪和导航

```
74  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
75  HAL_Init();
76
77  /* USER CODE BEGIN Init */
78
79  /* USER CODE END Init */
80
81  /* Configure the system clock */
82  SystemClock_Config();
83
84  /**
85   * @brief System Clock Configuration
86   * @retval None
87   */
88  void SystemClock_Config(void)
89  {
90      RCC_OscInitTypeDef RCC_OscInitStruct = {0};
91      RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
92
93      /** Configure the main internal regulator output voltage
94       */
95      __HAL_RCC_PWR_CLK_ENABLE();
96
97      while (1)
98      {
```

- F2: 代码就地显示
- F3: 代码追踪（最常用操作）
- Ctrl+Tab: 在头文件和源程序文件之间切换
- Outline视图

3. 编辑功能快捷操作

标题	快捷键	功能说明
Toggle Comment	Ctrl+7 或Ctrl+/ 	在选中的文本行前面加 “//” 使其变为注释，或解除注释
Add Block Comment	Ctrl+Shift+/ 	将选中的代码用 /* */ 进行块注释
Remove Block Comment	Ctrl+Shift+\ 	移除块注释的注释符号
Shift Left	Shift+TAB	选中的代码行向左移动，减少缩进
Correct Indentation	Ctrl+I	将选中的行向右缩进
Format	Ctrl+Shift+F	自动调整所选行的缩进
Toggle Source/Header	Ctrl+TAB	在头文件和程序文件之间切换
Toogle Breakpoint	Ctrl+Shift+B	在当前行设置或取消断点

4.4 项目文件组成

4.4.1 CMSIS驱动程序文件

4.4.2 HAL驱动程序文件

4.4.3 用户程序文件

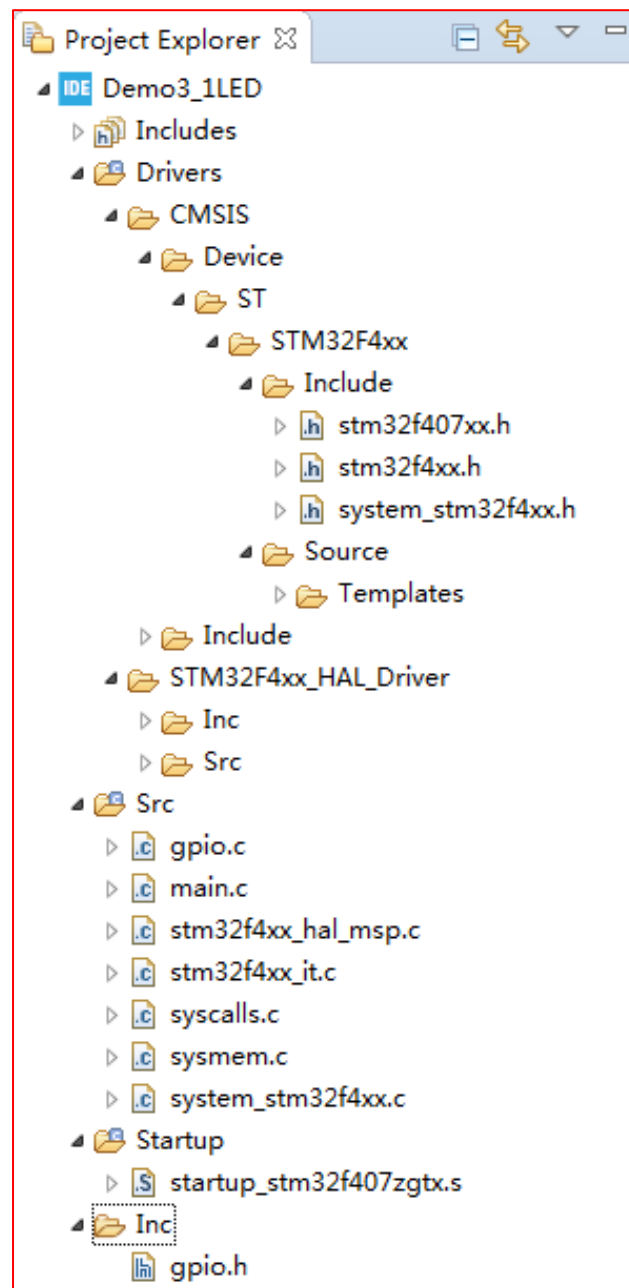
4.4.4 启动文件和其他文件

4.4.5 根目录下文件

4.4.6 Include搜索路径

项目Demo3_1LED的文件组成，主要包括以下几项：

- ◆ CMSIS驱动
- ◆ MCU外设的HAL驱动
- ◆ 用户程序文件，Inc和Src目录
- ◆ 项目根目录下的一些文件



4.4.1 CMSIS驱动程序文件

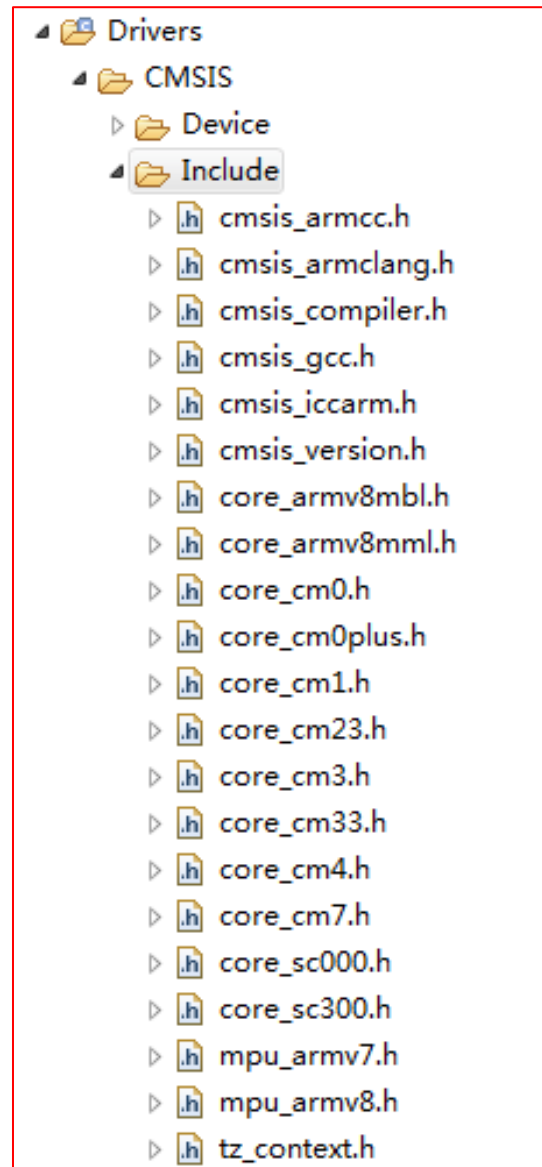
目录 \Drivers\CMSIS下是CMSIS（Cortex Microcontroller Software Interface Standard）标准的驱动程序，包括Cortex内核的驱动程序和具体MCU器件的基础定义头文件。

（1）目录 \Drivers\CMSIS\Device\ST\SM32F4xx\Include，这个目录下是具体型号的MCU的相关定义文件

- ◆ `stm32f407xx.h`，是STM32F407器件的外设访问头文件
- ◆ `stm32f4xx.h`，是STM32F4xx系列器件的配置文件
- ◆ `system_stm32f4xx.h`，这个文件里定义系统初始化函数 `SystemInit()`，这个函数是在系统复位之后，执行 `main()` 函数之前调用的

(2) 目录 \Drivers\CMSIS\Include

这个目录下都是与Cortex-M4内核相关的一些定义（如图4-27），是ARM公司提供的定义文件，与具体的MCU型号无关。



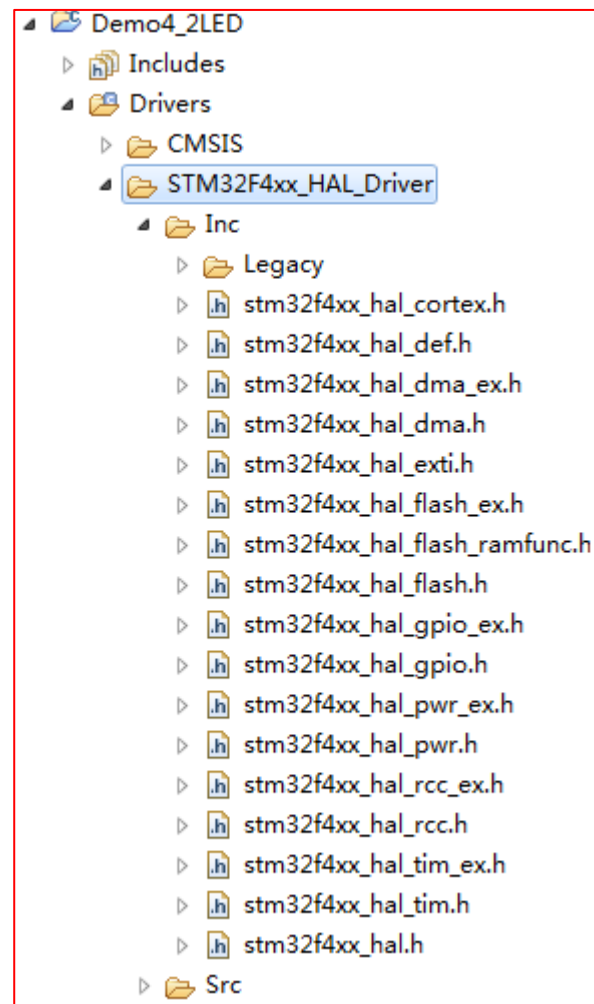
ARM内核驱动文件

4.4.2 HAL驱动程序文件

1. 外设驱动程序

每一种外设有一个基本驱动文件，有的还有一个扩展驱动文件。

- ◆ `stm32f4xx_hal_ppp.h`是外设ppp的基本驱动程序头文件，如`stm32f4xx_hal_gpio.h`、`stm32f4xx_hal_tim.h`等。
- ◆ `stm32f4xx_hal_ppp_ex.h`是外设ppp的扩展驱动程序头文件，包括某个型号或系列MCU的特定API函数，或重新定义的用于替换基本驱动程序的一些API函数。如`stm32f4xx_hal_gpio_ex.h`、`stm32f4xx_hal_tim_ex.h`等。



HAL外设驱动文件

2. HAL驱动程序头文件stm32f4xx_hal.h

stm32f4xx_hal.h文件里有HAL驱动的一些宏定义和函数定义。最主要的一个函数是HAL_Init()，它用于HAL库的初始化，其功能是复位所有外设、初始化Flash接口、配置系统定时器Systick周期为1ms。main()函数里首先就会调用这个函数。

stm32f4xx_hal.h中还有一个常用的延时函数HAL_Delay(uint32_t Delay)，它基于系统定时器SysTick实现精确的毫秒级延时。

3. HAL通用定义文件stm32f4xx_hal_def.h

这个文件里有HAL的一些通用定义、枚举类型、宏定义、结构体等

4. Cortex HAL模块驱动文件stm32f4xx_hal_cortex.h

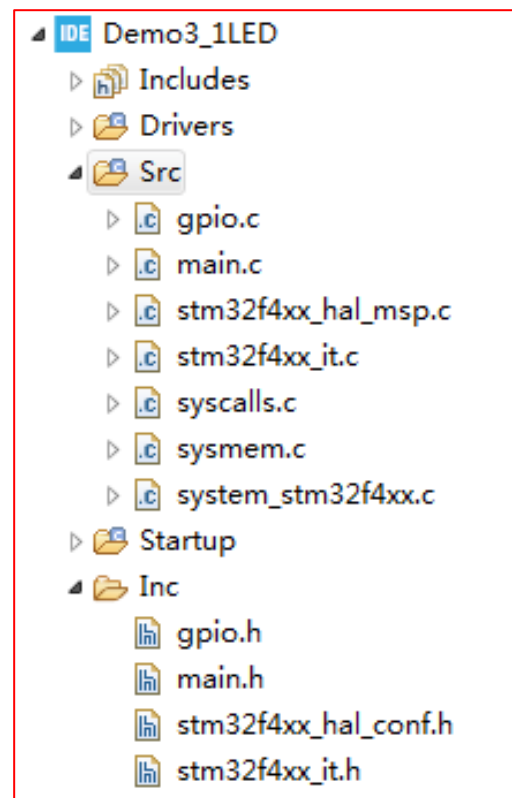
这个文件是Cortex HAL模块驱动文件，它提供HAL驱动中用到的Cortex内核的一些常量、结构体和函数定义，如中断优先级定义、中断优先级设置函数等

4.4.3 用户程序文件

用户的程序文件分布在项目根目录下的\Inc和\Src两个子目录下，因为我们在CubeMX里设置导出代码时选择生成.h/.c文件对。

1. 外设的初始化程序文件

在CubeMX中启用的外设都会生成一个外设初始化程序文件。如本实例中用到PF9和PF10，所以生成文件gpio.h和gpio.c



用户程序文件

2. HAL配置文件stm32f4xx_hal_conf.h

可以修改这个文件的内容，对HAL驱动程序进行一些配置。

文件stm32f4xx_hal_conf.h的内容是根据CubeMX中的配置自动生成的，一般不要直接修改此文件，而是在CubeMX里修改配置后重新生成代码。

3. 中断处理程序文件

文件stm32f4xx_it.h中是中断处理函数的原型定义，文件stm32f4xx_it.c中是中断处理函数的实现文件。

4. HAL的MSP初始化程序文件

stm32f4xx_hal_msp.c是MSP程序文件。MSP是MCU Specific Package，即MCU特定程序包。这个文件里定义MSP的初始化函数和反初始化函数。

5. 处理器系统初始化文件system_stm32f4xx.c

这个文件里主要的函数是系统初始化函数SystemInit()，这个函数在系统复位之后，执行main()函数之前被执行。其功能复位RCC时钟配置、对SRAM的向量表重定位，配置FSMC/FMC外设以使用外部的SRAM或SDRAM存储器。

6. 最小系统调用文件

文件syscalls.c和sysmem.c是CubeIDE最小系统需要用到的文件，syscalls.c定义了一些底层函数，sysmem.c定义了内存管理函数。这两个文件里的函数是被CubeIDE调用的，用户程序不会直接用到。

7. 主程序文件

文件main.h，在STM32CubeMX中定义的引脚标签会在这个文件里生成宏定义，如定义的引脚标签LED1、LED2

文件main.c，打开主程序代码分析

4.4.4 启动文件

目录\startup下的文件startup_stm32f407zgtx.s是处理器的启动文件，这是个汇编语言程序文件，是MCU复位后首先执行的程序。所有中断的ISR函数名称在这个文件里定义。

它初始化堆栈指针SP、代码指针PC、设置中断程序向量表、执行system_stm32f4xx.c文件中的函数SystemInit()对RCC、SRAM等进行初始化，然后执行主函数main()。

4.4.5 根目录下的文件

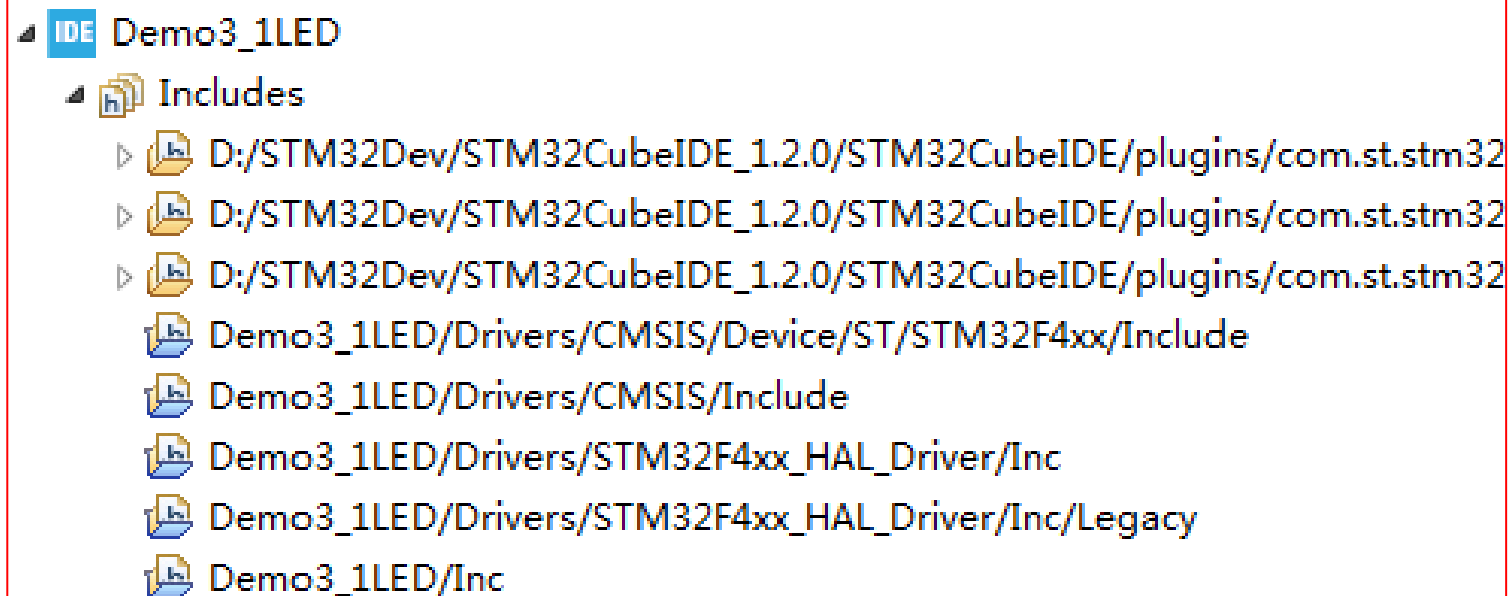
项目根目录下还有两个后缀为.ld的文件，这两个文件是存储器的编译连接脚本文件。

- ◆ 文件STM32F407ZGTX_FLASH.ld，用于设置堆的大小、栈的大小及位置、默认的各种程序段和数据段的地址和长度等。如果使用了外部存储器，还设置存储器位置和大小。
- ◆ 文件STM32F407ZGTX_RAM.ld，与前一个文件的功能基本相同，但只是在RAM中调试时才用到。

如果项目成功用仿真器下载到开发板上调试过，还会在根目录下生成一个后缀为.launch的文件，如Demo3_1LED Debug.launch，这个文件保存了仿真调试器的启动配置参数。

4.4.6 Include搜索路径

项目虚拟目录\Includes。前3条路径是CubeIDE的编译工具链标准库的路径，后面5条是本项目的Include搜索路径。



项目下的虚拟目录\Includes

4.5 项目管理、编译和下载调试

4.5.1 项目管理

4.5.2 项目编译

4.5.3 下载和调试

4.5.1 项目管理

- ◆ 一个Workspace可以管理多个项目，这些项目的存储文件夹最好就是Workspace的子目录。
- ◆ Workspace里的项目有打开和关闭两种状态。
- ◆ 当Workspace里有多个项目处于打开状态时，只有一个项目是当前项目，最好只打开一个项目，其他项目关闭。

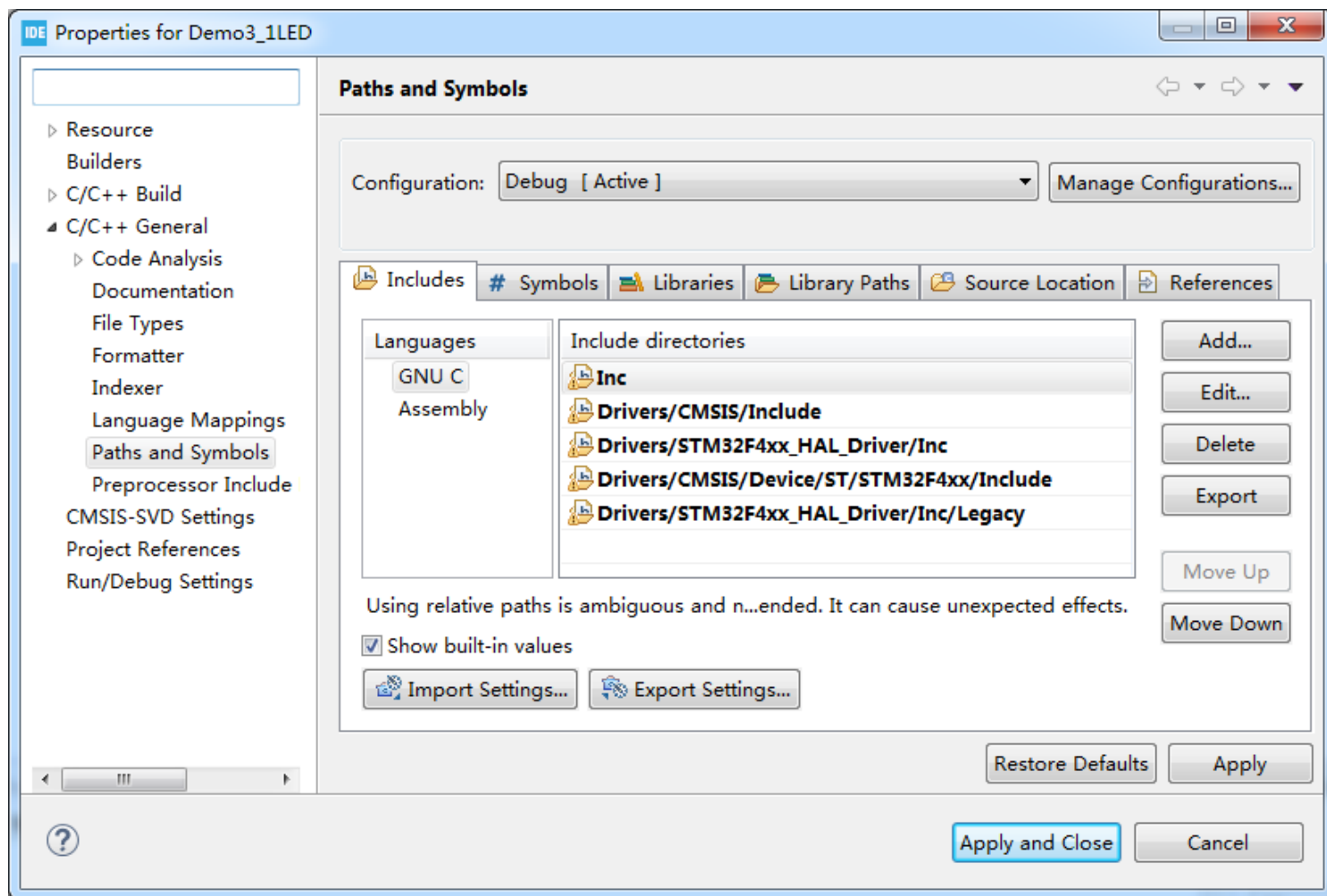
4.5.2 项目编译

打开项目属性设置对话框，进行项目编译相关设置：

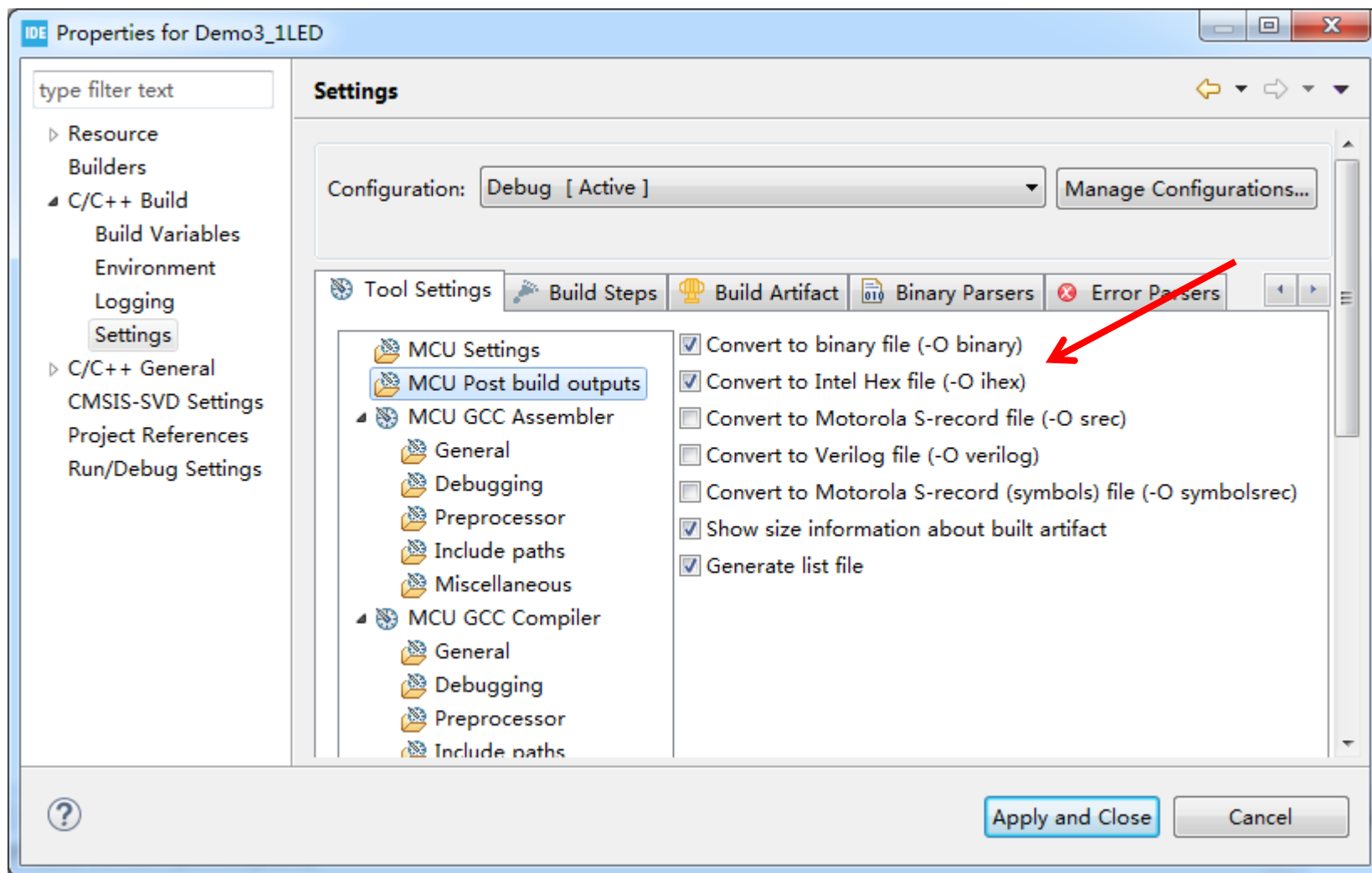
- ◆ 设置Include和Source Location搜索路径
- ◆ 设置输出二进制文件类型，如生成hex、bin文件
- ◆ 设置编译优化级别
- ◆ 设置并行编译（默认开启并行编译）

◆ 设置Include和Source Location搜索路径

需要使用其他目录下的源程序文件时，需要设置Include和Source Location路径

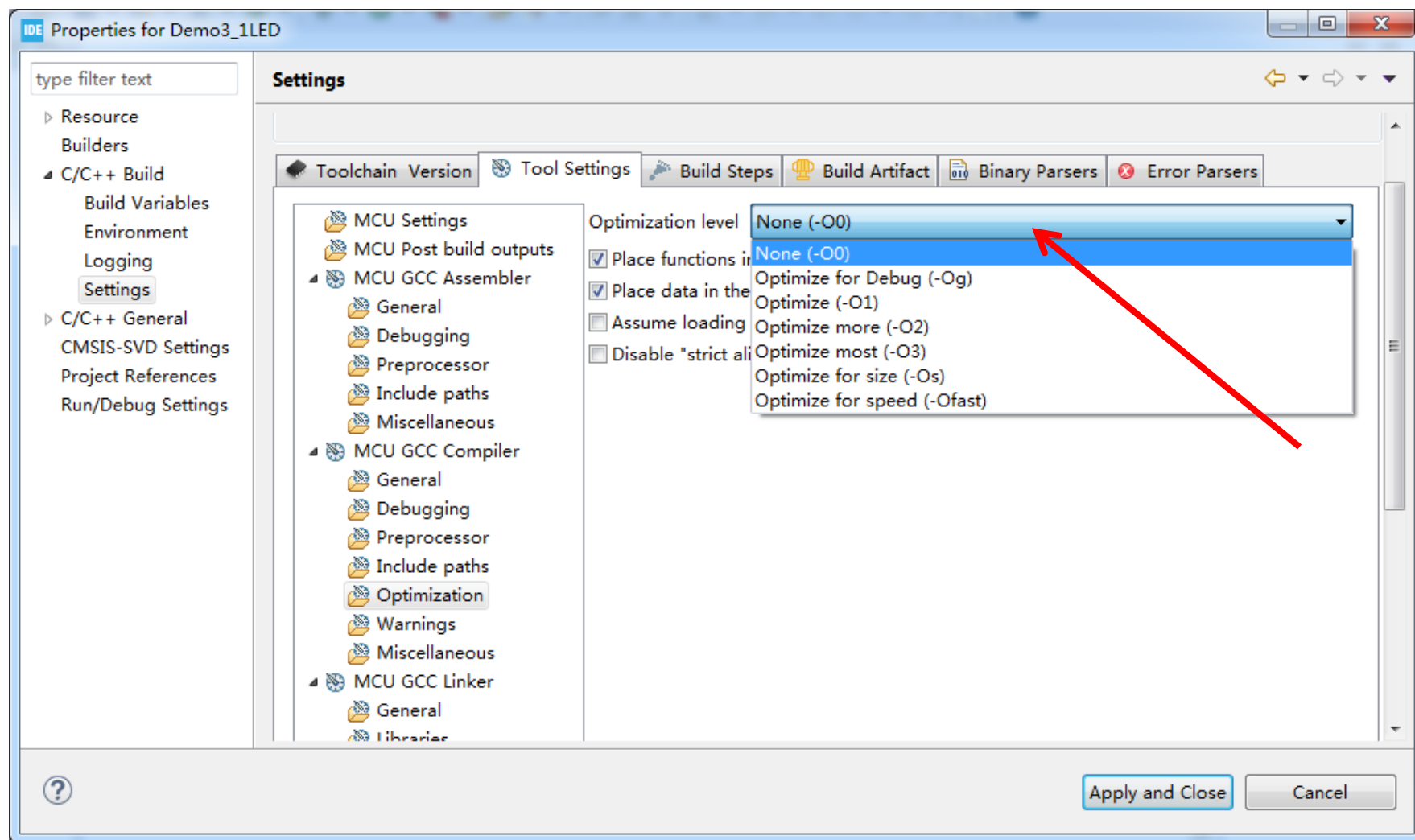


- ◆ 设置输出二进制文件类型，如生成hex、bin文件
- CubeIDE默认输出二进制文件是elf文件



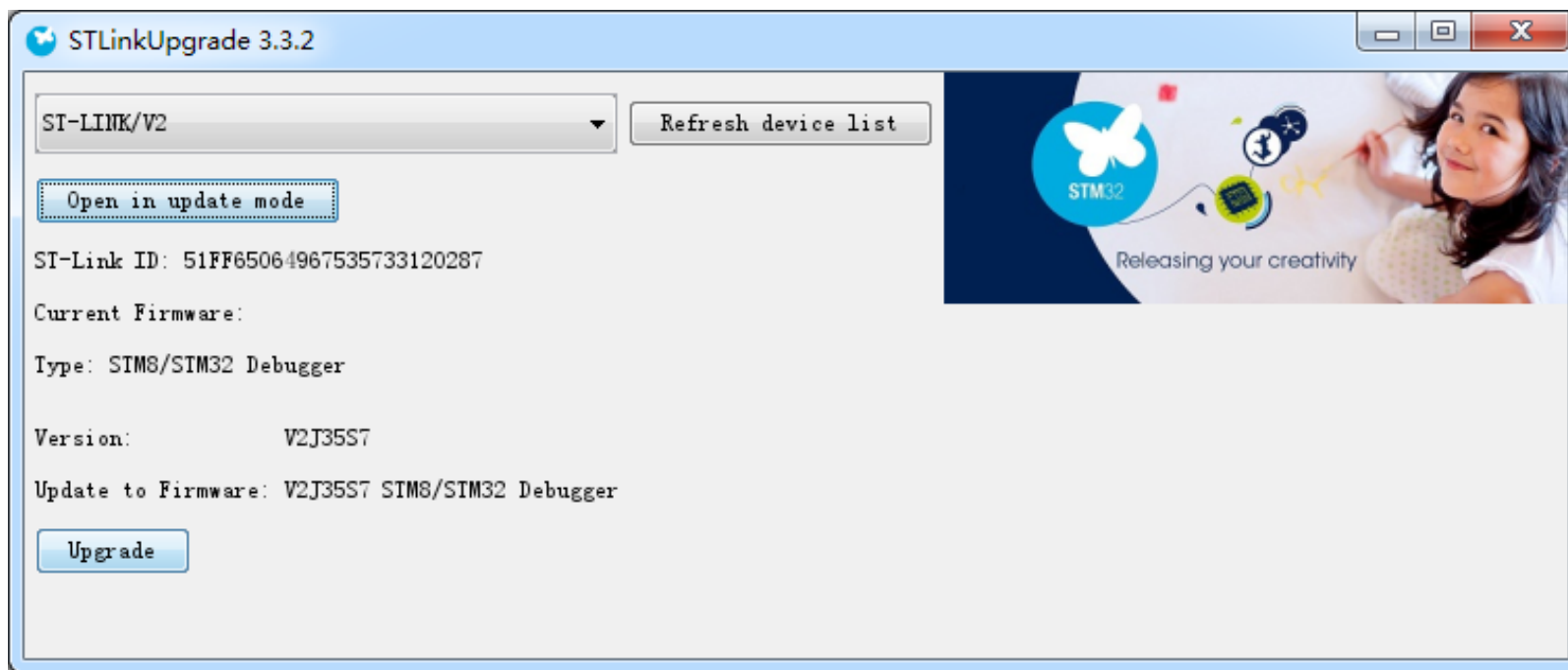
◆ 设置编译优化级别

要进行断点调试，编译优化级别必须是-O0或-Og，默认是-O0



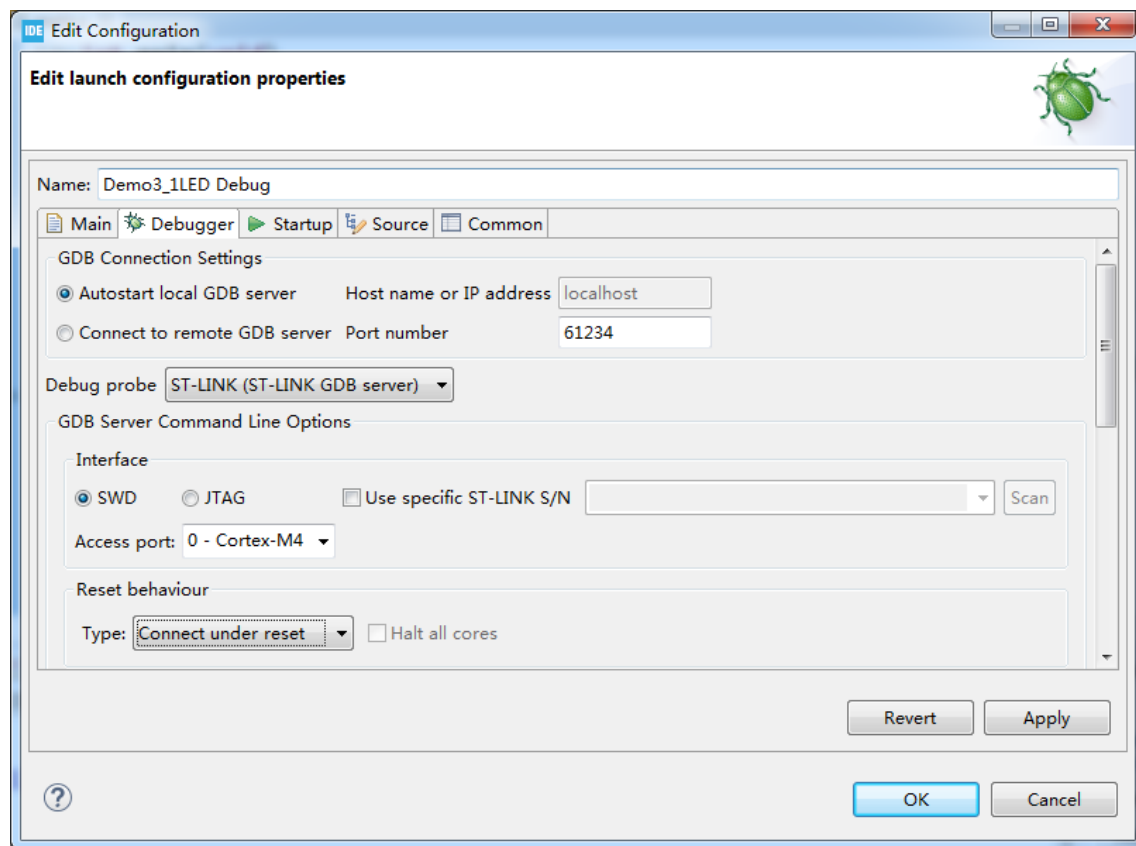
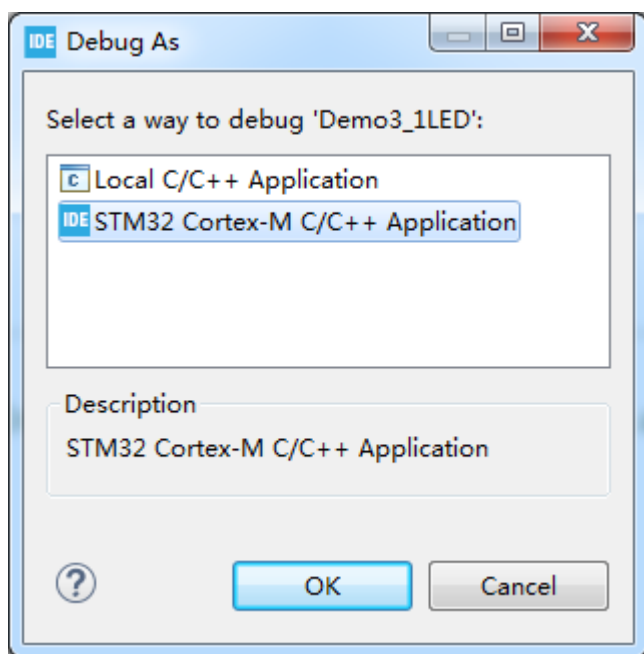
4.5.3 下载和调试

1. 仿真器检测和固件升级



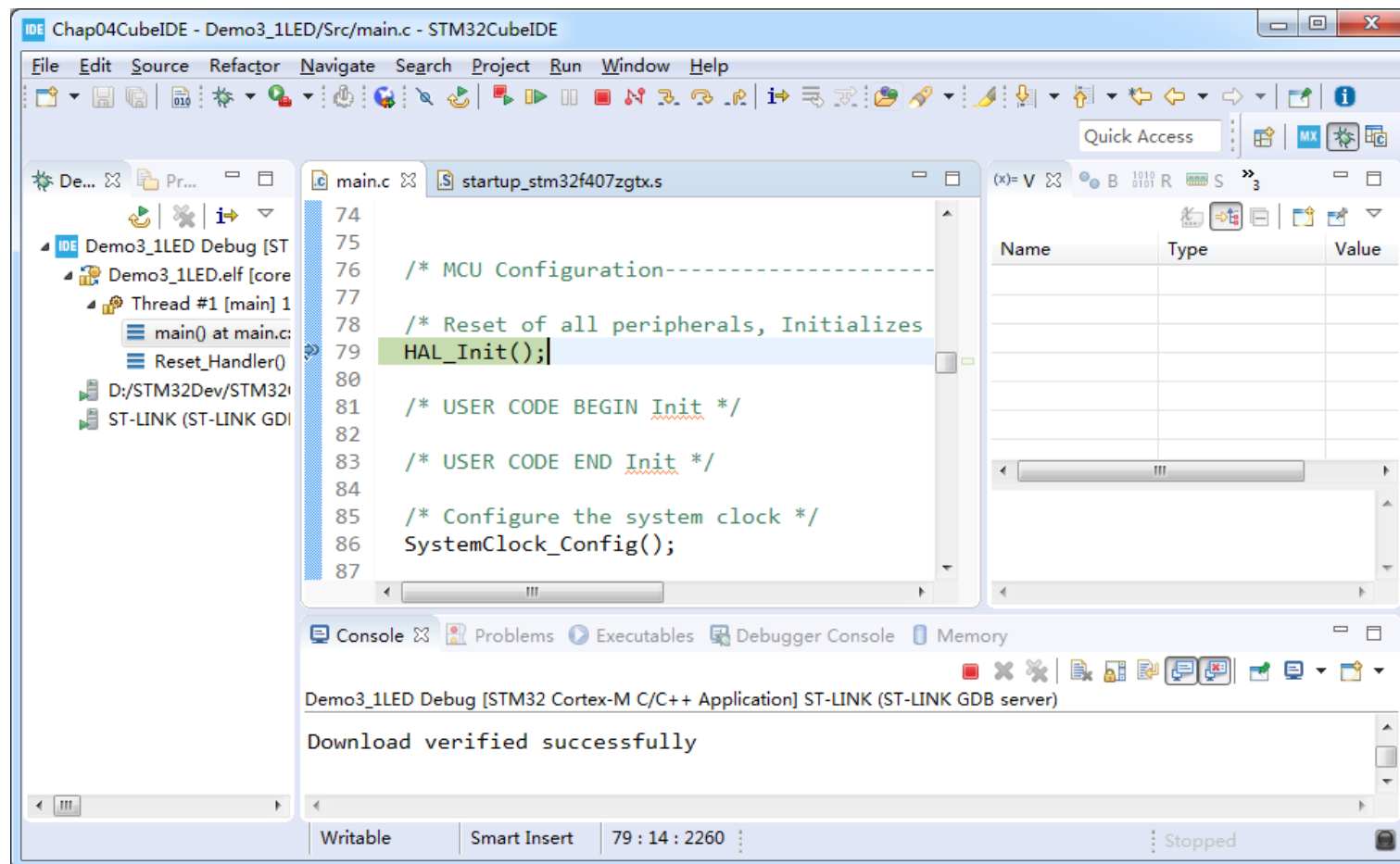
在CubeIDE里，如果固件版本低，将无法下载和调试

2. 程序下载



下载过程中出现的2个对话框













3. Debug场景界面和调试操作



要会使用断点，善于单步调试



表 4-3 用于程序调试的工具栏按钮的功能

图标	提示文字	快捷键	功能说明
	Skip All Breakpoints	Ctrl+Shift+B	一个复选按钮，按下时将忽略所有断点
	Terminate and relaunch		停止程序，重新下载程序后开始调试
	Resume	F8	从当前停止的行开始，继续连续运行
	Suspend		程序暂停
	Terminate	Ctrl+F2	终止程序执行，并退出 Debug 场景，返回 C/C++ 编程场景
	Disconnect		断开连接，并退出 Debug 场景，返回 C/C++ 编程场景
	Step Into	F5	若当前行是一个函数调用，将会跳转入此函数的代码内执行
	Step Over	F6	一步执行当前行程序，不管是否有函数调用。如果执行的函数里有断点定义，会执行到断点处暂停
	Step Return	F7	如果当前代码段是跳转进来的一个函数，按此键可以执行完此函数的代码，返回上一层
	Run to Line	Ctrl+R	运行到鼠标光标所在的行
	Instruction stepping mode		进入反汇编视图（Disassembly），单步调试汇编语言指令
	Reset the chip and restart debug session		芯片复位，重新开始调试过程

4. 修改LED引脚的输出

可以直接修改CubeIDE项目中两个LED引脚的初始化设置程序，即文件gpio.c中的函数MX_GPIO_Init()的代码，这个函数中有如下的代码：

```
/*Configure GPIO pin Output Level */  
HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_SET);  
HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
```

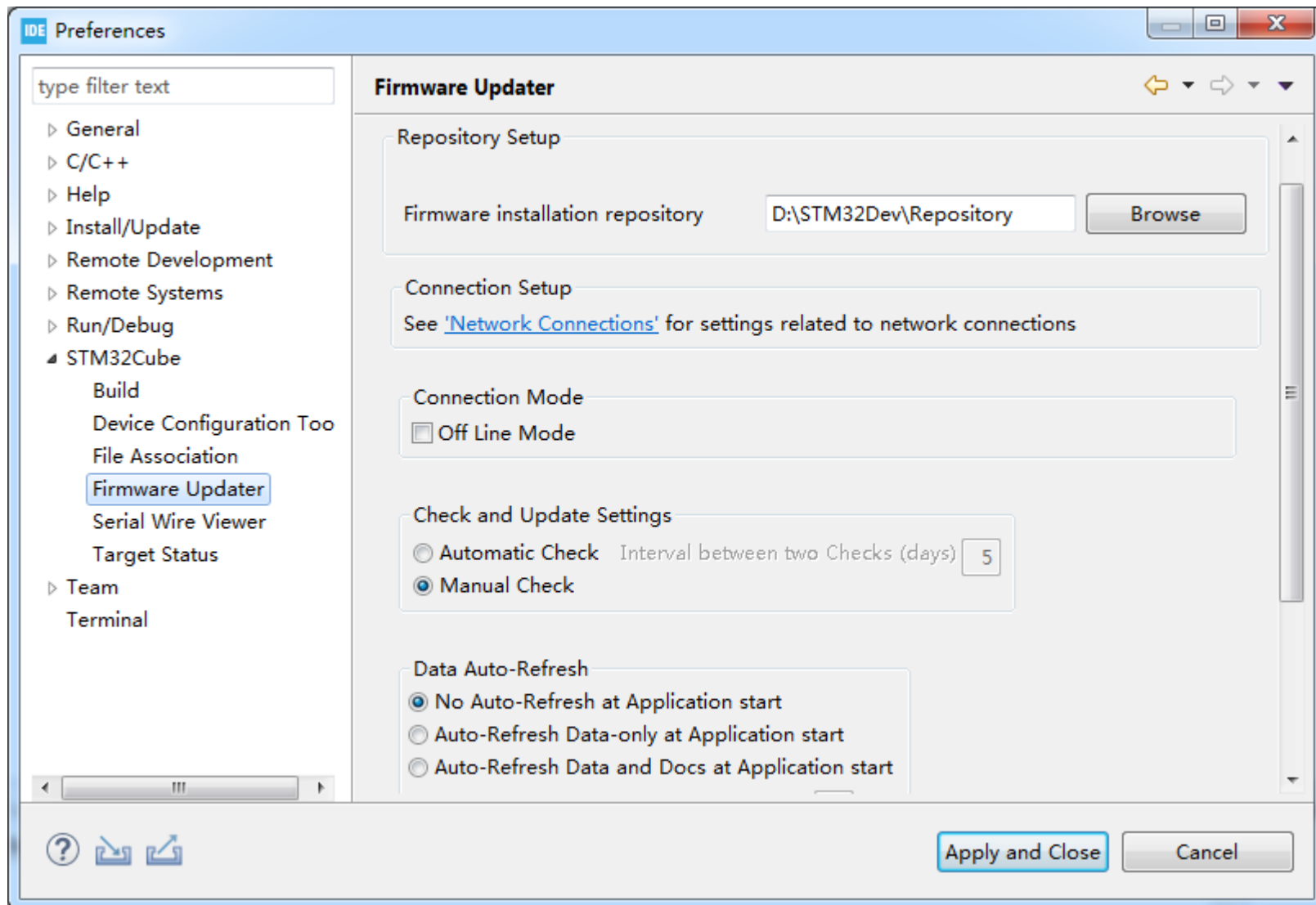
函数HAL_GPIO_WritePin()就是用于设置一个GPIO引脚输出电平的，最后的参数GPIO_PIN_SET表示输出高电平，GPIO_PIN_RESET表示输出低电平。

4.6 使用内置的CubeMX

4.6.1 创建项目

4.6.2 配置MCU和生成代码

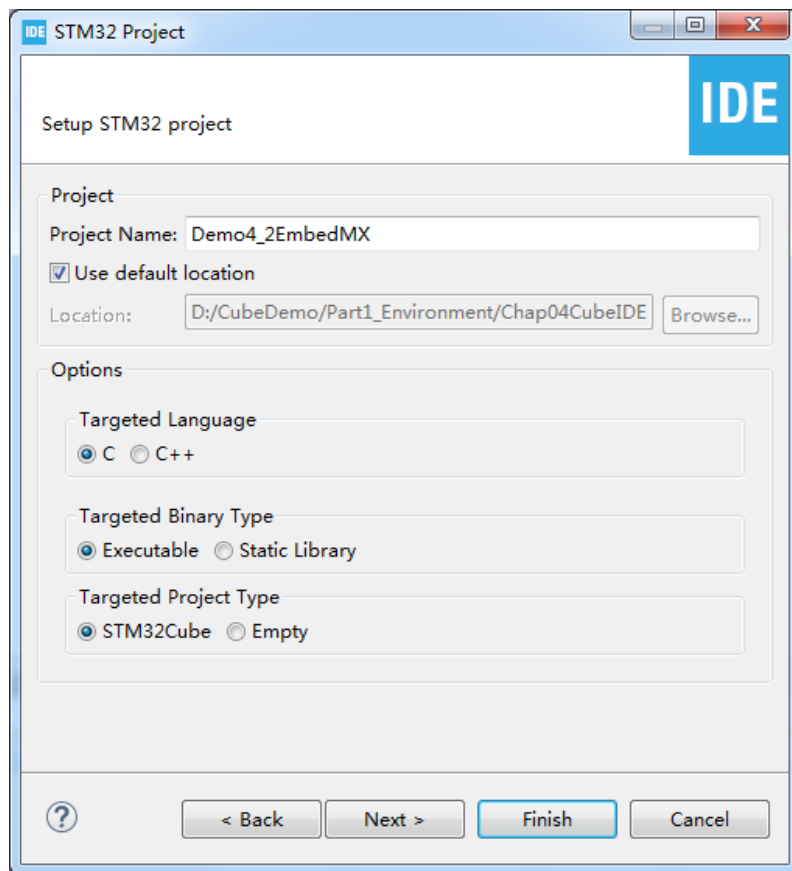
使用内置CubeMX之前，需要设置固件库路径。点击主菜单 Windows → Preferences打开对话框，设置固件库路径。可以与独立的CubeMX共用固件库。



4.6.1 创建项目

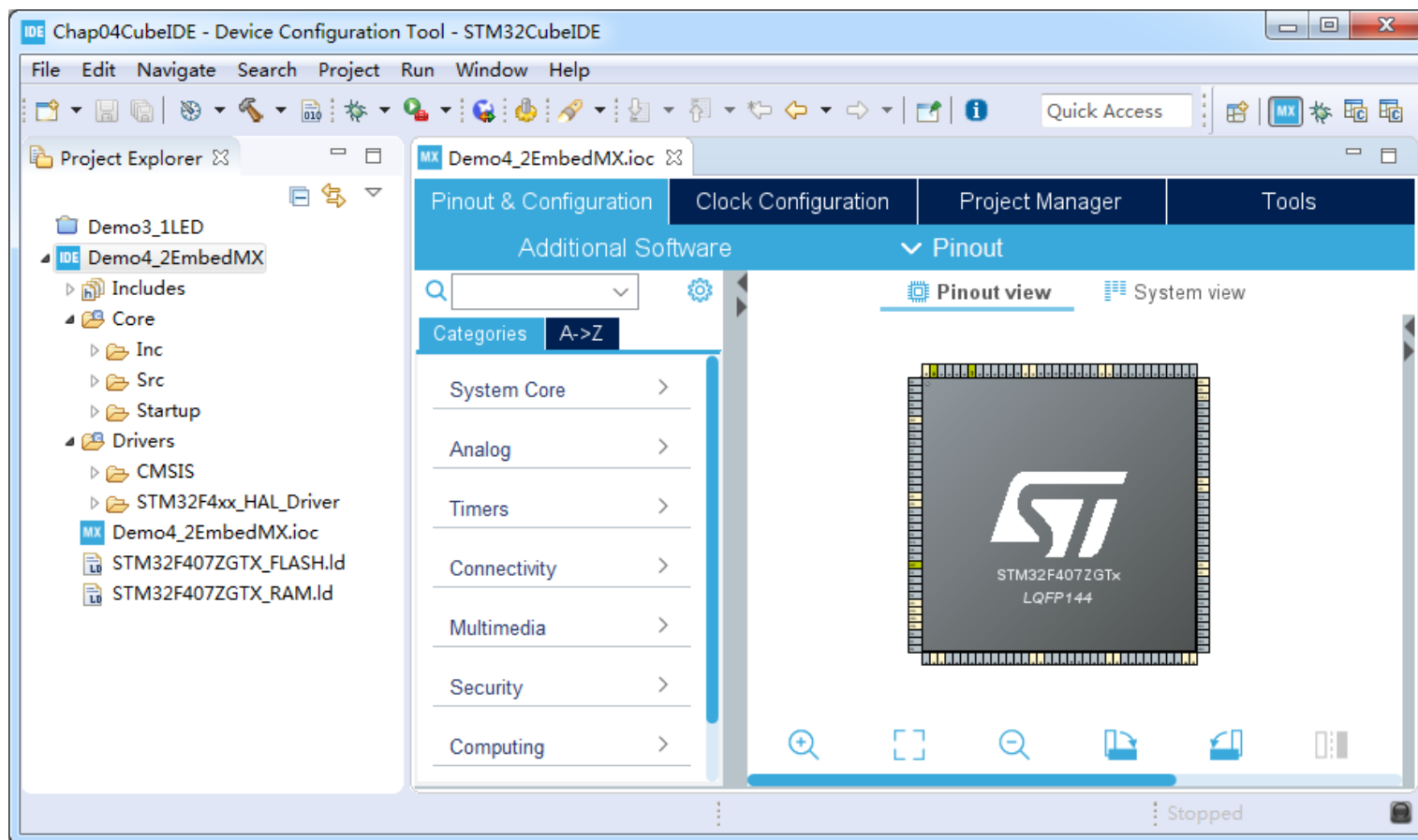
创建项目Demo4_2EmbedMX，与使用独立的CubeMX类似

【过程演示】



4.6.2 配置MCU和生成代码

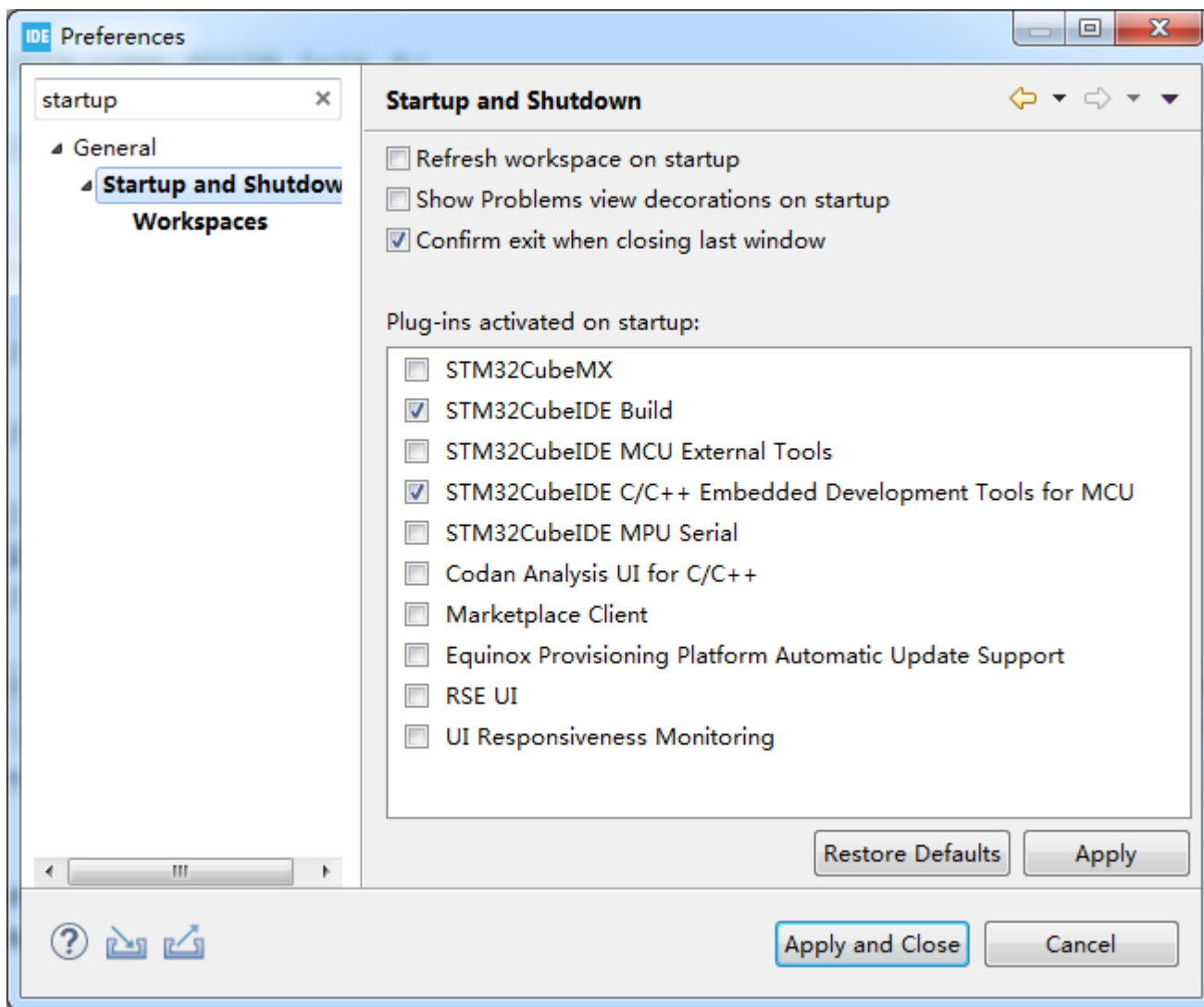
虽然可以使用内置的CubeMX在CubeIDE里创建STM32项目，但是使用独立的CubeMX软件更方便。



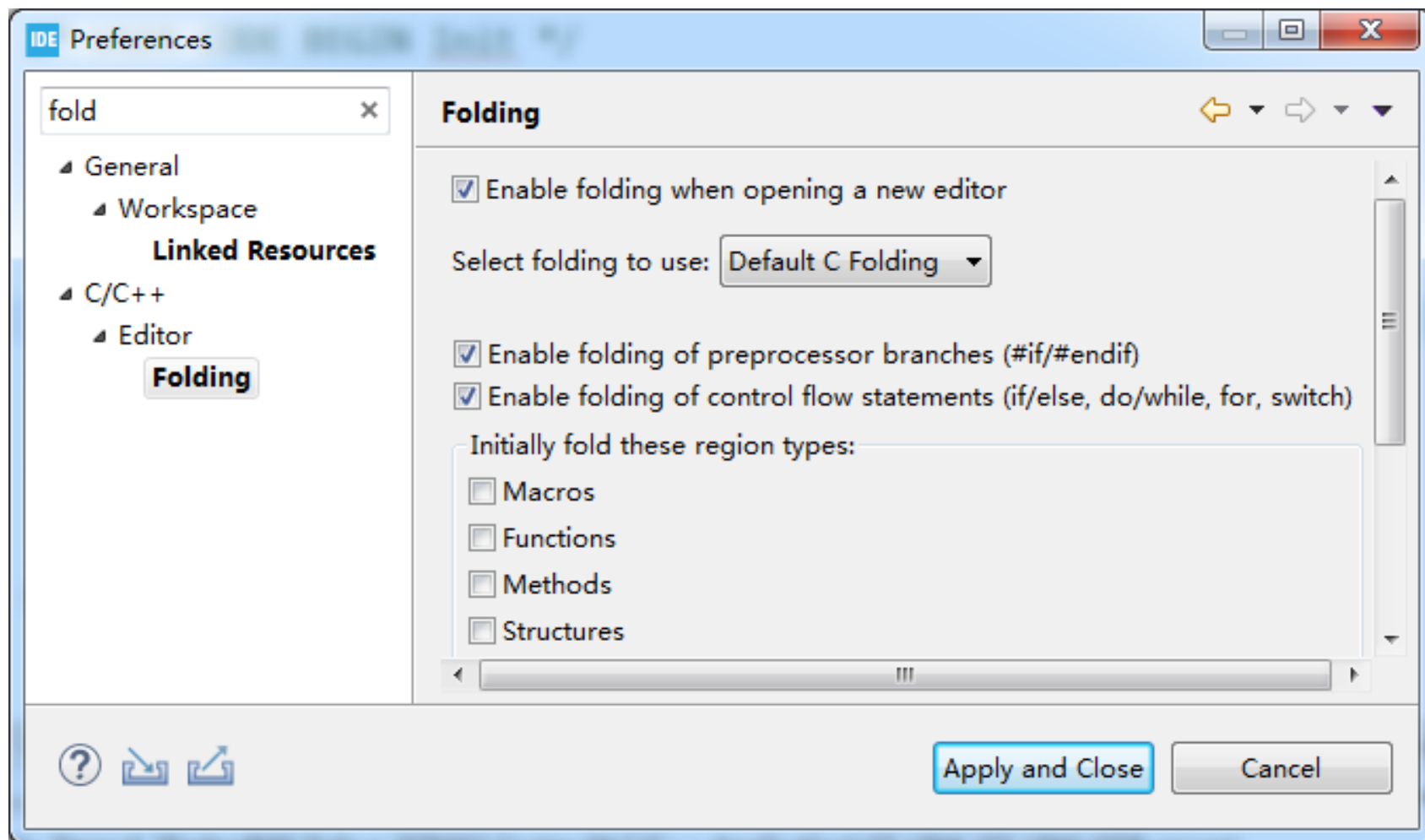
4.7 CubeIDE使用偏好设置

点击主菜单 Windows → Preferences，打开对话框，对 CubeIDE 进行各种设置。

不启动无用插件，加快软件启动速度，减少内存占用



代码折叠（Folding）设置，便于观察代码结构



4.8 HAL驱动库使用的一些问题

4.8.1 基本数据类型

4.8.2 一些通用定义

4.8.3 获取HAL库函数帮助信息

4.8.1 基本数据类型

STM32编程中的数据类型简化定义符号

数据类型	C语言等效定义	字节数
int8_t	signed char	1
uint8_t	unsigned char	1
int16_t	signed short	2
uint16_t	unsigned short	2
int32_t	signed int	4
uint32_t	unsigned int	4
int64_t	long long int	8
uint64_t	unsigned long long int	8

4.8.2 一些通用定义

- ◆ 文件stm32f4xx_hal_def.h中定义的代表函数返回值类型的枚举类型HAL_StatusTypeDef，定义如下。

```
typedef enum
{
    HAL_OK      = 0x00U,
    HAL_ERROR   = 0x01U,
    HAL_BUSY    = 0x02U,
    HAL_TIMEOUT = 0x03U
} HAL_StatusTypeDef;
```

◆ 文件stm32f4xx.h中定义的几个通用的枚举类型和常量

```
typedef enum
{
    RESET = 0U,
    SET = !RESET
} FlagStatus, ITStatus;           //一般用于判断标志位是否置位

typedef enum
{
    DISABLE = 0U,
    ENABLE = !DISABLE
} FunctionalState;

typedef enum
{
    SUCCESS = 0U,
    ERROR = !SUCCESS
} ErrorStatus;                   //一般用于函数返回值，表示成果或失败两种状态
```

4.8.3 获取HAL库函数帮助信息

最好的方法是查看HAL库的源程序，有详细的注释，而且使用快捷键 F3 进行代码跟踪很方便。

参考资料

- ST User Manual, UM2563, *STM32CubeIDE installation guide*
- ST User Manual, UM2553, *STM32CubeIDE quick start guide*

练习任务

- 练习本章的示例，下载到开发板上测试，将软硬件开发环境搭建好，跑得通。