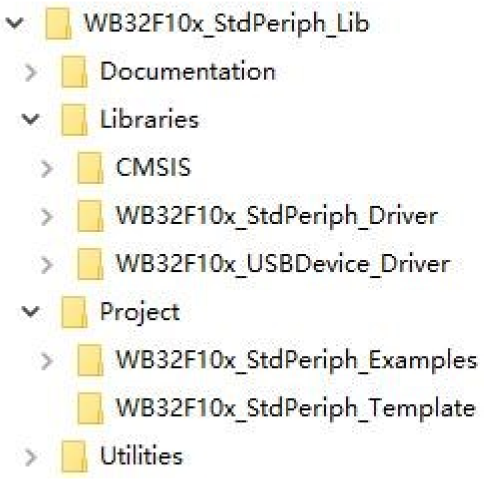
WB32F10x 开发入门

常州韦斯佰瑞电子科技有限公司

目录

1 固件库介绍

下图所示即为 WB32F10x 标准固件库的目录结构。



- WB32F10x_StdPeriph_Lib
  - Documentation
  - Libraries
    - CMSIS
    - WB32F10x_StdPeriph_Driver
    - WB32F10x_USBDevice_Driver
  - Project
    - WB32F10x_StdPeriph_Examples
    - WB32F10x_StdPeriph_Template
  - Utilities

Documentation 目录中存放了 WB32F10x 固件库的说明文档。

Libraries 目录中包含 **CMSIS**，**WB32F10x_StdPeriph_Driver**，**B32F10x_USBDevice_Driver** 三个子目录。其中 CMSIS 目录中存放了启动文件，头文件等

WB32F10x_StdPeriph_Driver 目录中存放了 WB32F10x 固件库源码文件；
WB32F10x_USBDevice_Driver 目录中存放了 WB32F10x USB 设备协议栈代码。

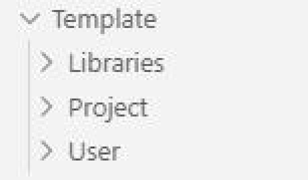
Project 目录中包含 **WB32F10x_StdPeriph_Examples** 和 **WB32F10x_StdPeriph_Template** 两个子目录。其中 **WB32F10x_StdPeriph_Examples** 目录中存放了 WestBerry 官方提供的固件示例源码，以便客户参考；

WB32F10x_StdPeriph_Template 目录中存放了创建工程所需要的文件模板。

Utilities 目录中存放了一些公用的源码。

2 使用 Keil MDK 建立工程

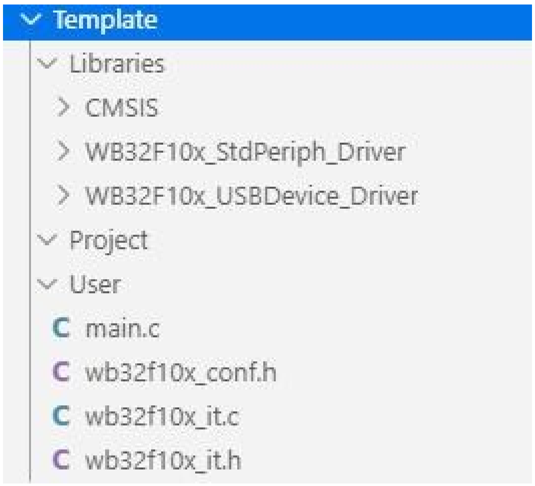
Step 01. 新建一个文件夹命名为 Template 用以存放整个工程。



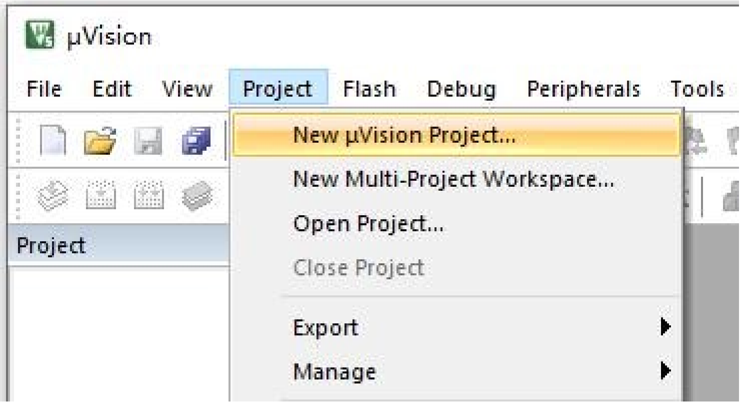
Step 02. 在 Template 文件夹中新建 Libraries，Project 和 User 三个子文件夹（当然用户可定义自己工程目录结构）

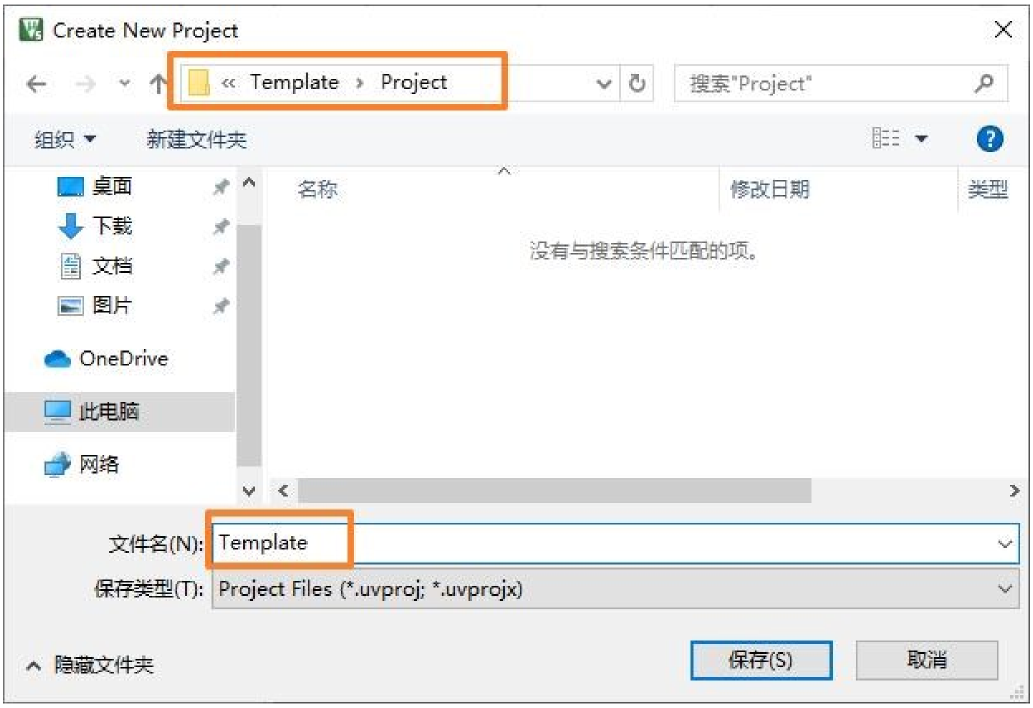
Step 03. 将 WB32F10x 标准固件库中 Libraries 目录中的内容复制到 Template\Libraries 目录中。

Step 04. 将 WB32F10x 标准固件库中 Project\WB32F10x_StdPeriph_Template 目录中的内容复制到 Template\User 目录中



Step 05. 打开 Keil MDK，在 Template\Project 目录中新建名为 Template 的工程

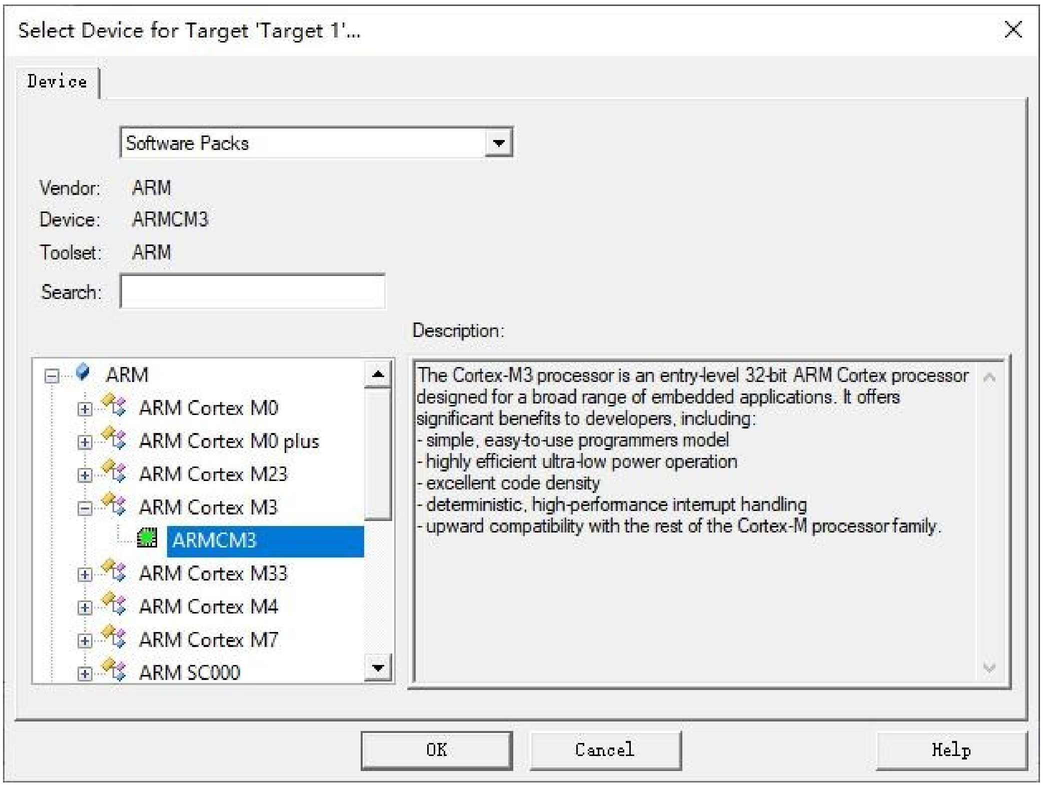




文件名(N): Template

保存类型(T): Project Files (*.uvproj; *.uvprojx)

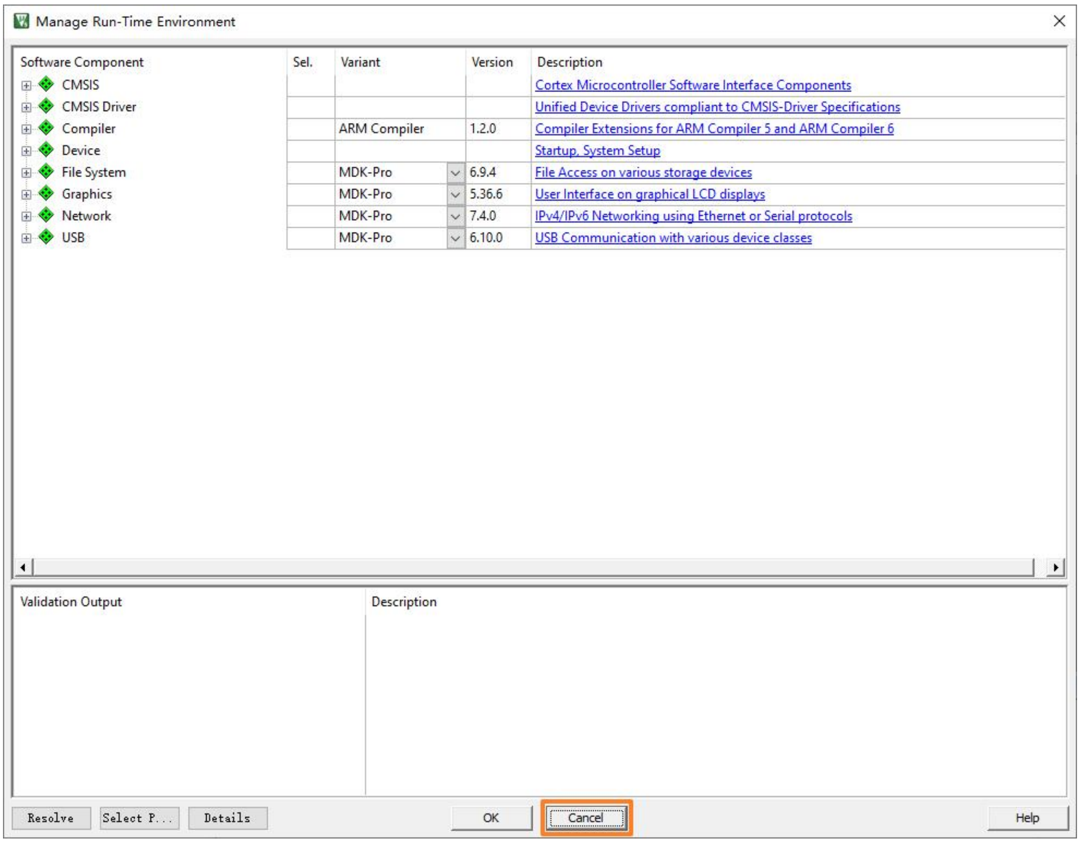
Step 06. 选择项目使用的设备为 ARMCM3，并点击 OK



Vendor: ARM
Device: ARMCM3
Toolset: ARM
Search:

Description:
The Cortex-M3 processor is an entry-level 32-bit ARM Cortex processor designed for a broad range of embedded applications. It offers significant benefits to developers, including:
- simple, easy-to-use programmers model
- highly efficient ultra-low power operation
- excellent code density
- deterministic, high-performance interrupt handling
- upward compatibility with the rest of the Cortex-M processor family.

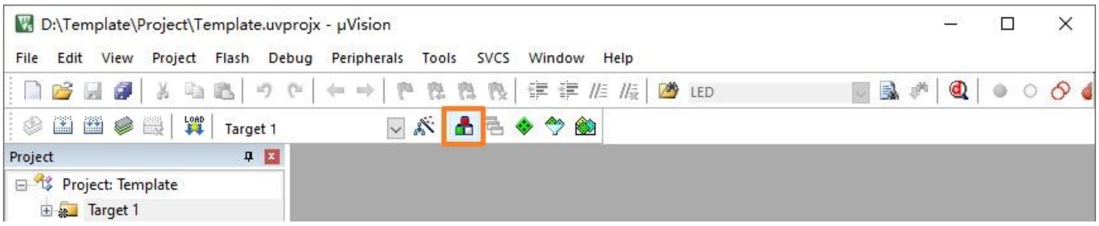
此时弹出 Manage Run-Time Environment 对话框，在该对话框上点击 Cancel

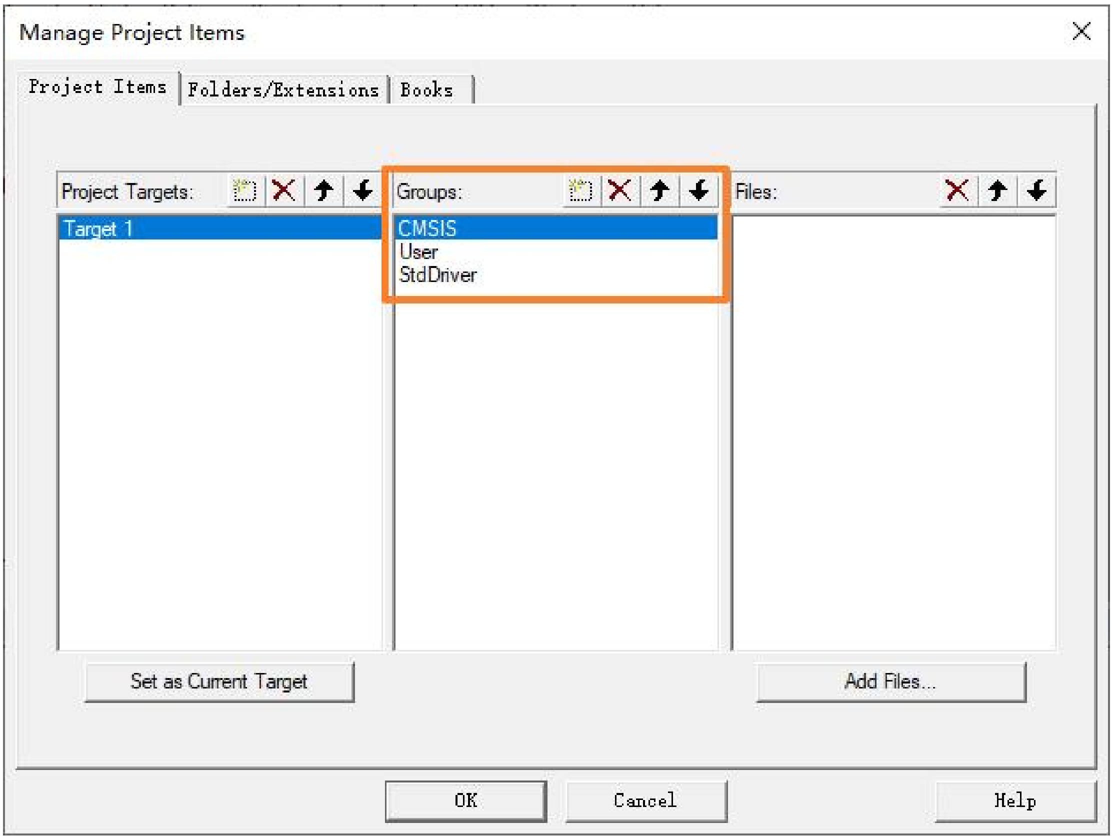


Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
CMSIS				Cortex Microcontroller Software Interface Components
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
Compiler		ARM Compiler	1.2.0	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Device				Startup, System Setup
File System		MDK-Pro	6.9.4	File Access on various storage devices
Graphics		MDK-Pro	5.36.6	User Interface on graphical LCD displays
Network		MDK-Pro	7.4.0	IPv4/IPv6 Networking using Ethernet or Serial protocols
USB		MDK-Pro	6.10.0	USB Communication with various device classes

Step 07. 在加入固件库文件之前，我们先建立三个 Groups：**CMSIS，User，StdDriver**


D:\Template\Project\Template.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
LED
Target 1
Project
Project: Template
Target 1



Step 08. 向 Group 里面添加固件库文件。

向 CMSIS Group 中添加：

Template\Libraries\CMSIS\Device\WB\WB32F10x\startup\arm\startup_wb32f10x.s

Template\Libraries\CMSIS\Device\WB\WB32F10x\system_wb32f10x.c

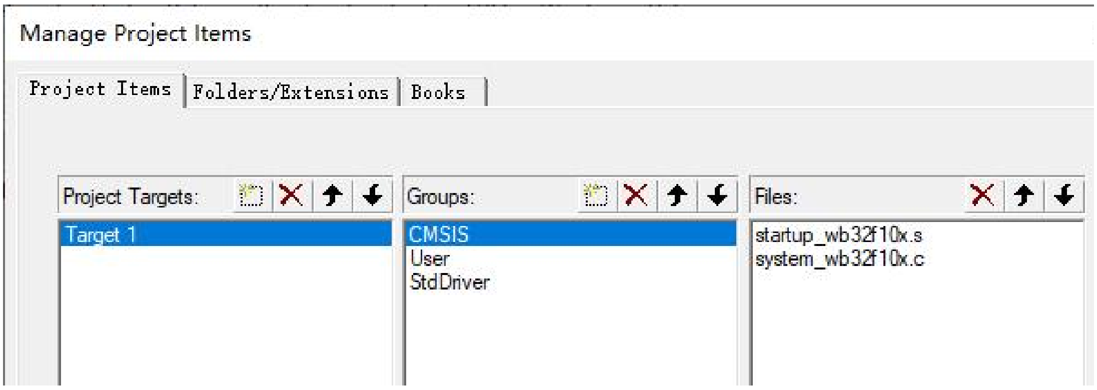
向 User Group 中添加：

Template\User\main.c

Template\User\wb32f10x_it.c

向 StdDriver Group 中添加 Template\Libraries\WB32F10x_StdPeriph_Driver\src 目录中所有的.c 文件

如下面三张图片所示：



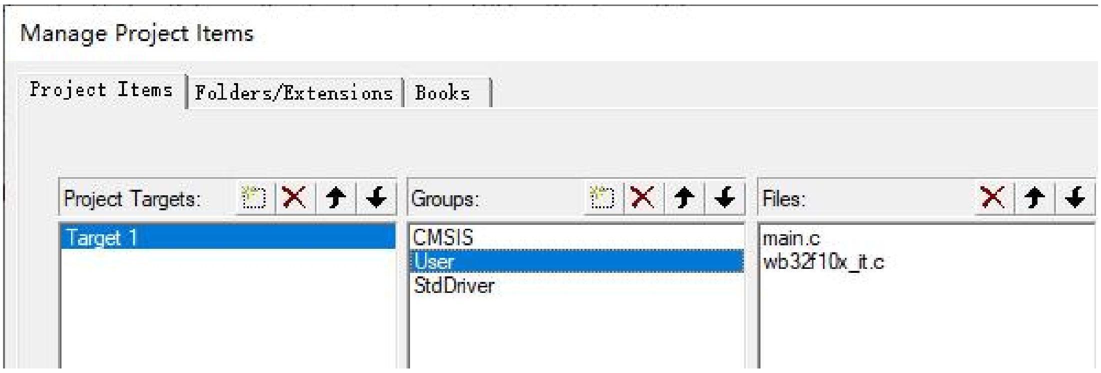
Manage Project Items

Project Items | Folders/Extensions | Books

Project Targets:
Target 1

Groups:
CMSIS
User
StdDriver

Files:
startup_wb32f10x.s
system_wb32f10x.c



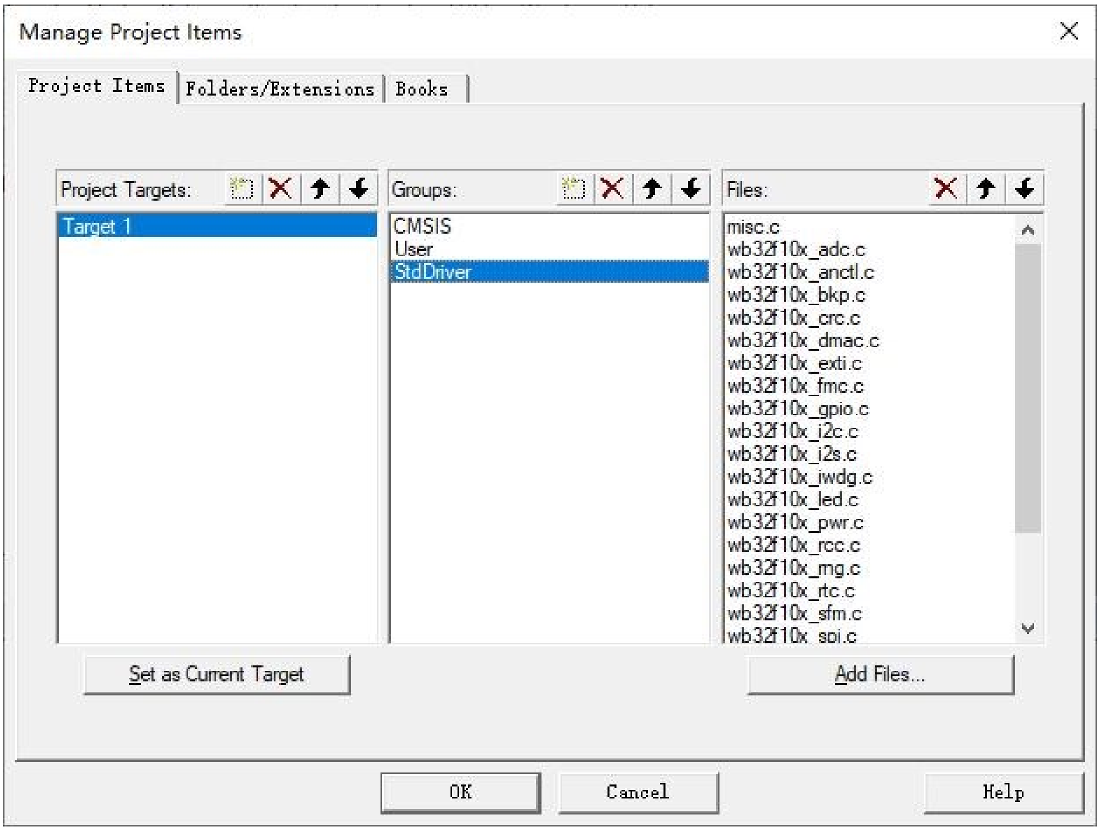
Manage Project Items

Project Items | Folders/Extensions | Books

Project Targets:
Target 1

Groups:
CMSIS
User
StdDriver

Files:
main.c
wb32f10x_it.c



Manage Project Items

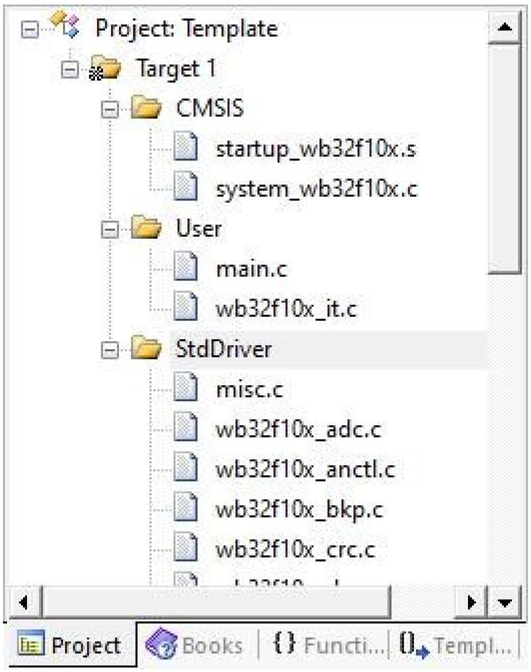
Project Items | Folders/Extensions | Books

Project Targets:
Target 1

Groups:
CMSIS
User
StdDriver

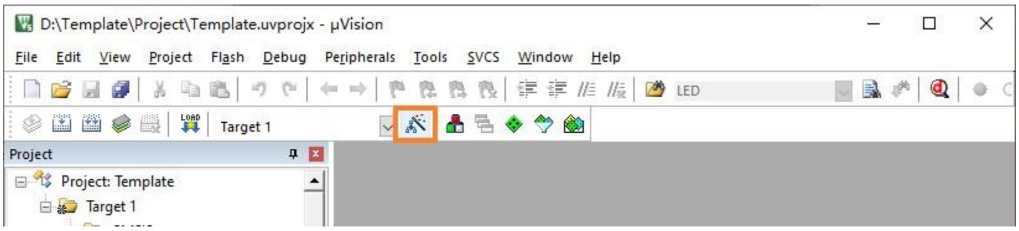
Files:
misc.c
wb32f10x_adc.c
wb32f10x_anctl.c
wb32f10x_bkp.c
wb32f10x_crc.c
wb32f10x_dmac.c
wb32f10x_exti.c
wb32f10x_fmc.c
wb32f10x_gpio.c
wb32f10x_i2c.c
wb32f10x_i2s.c
wb32f10x_iwdg.c
wb32f10x_led.c
wb32f10x_pwr.c
wb32f10x_rcc.c
wb32f10x_rng.c
wb32f10x_rtc.c
wb32f10x_sfm.c
wb32f10x_spi.c

最终目录结构：



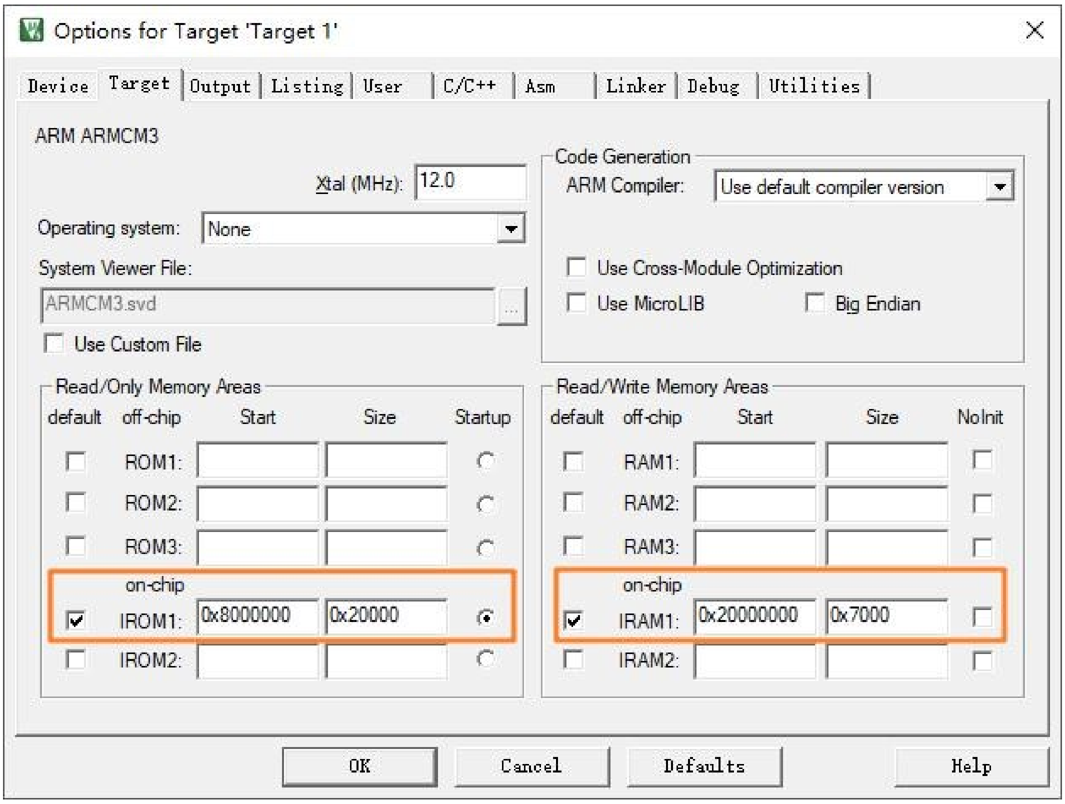
Project: Template
Target 1
CMSIS
startup_wb32f10x.s
system_wb32f10x.c
User
main.c
wb32f10x_it.c
StdDriver
misc.c
wb32f10x_adc.c
wb32f10x_anctl.c
wb32f10x_bkp.c
wb32f10x_crc.c

Step 09. 打开 Options for Target 对话框。


D:\Template\Project\Template.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
LED
Target 1
Project
Project: Template
Target 1

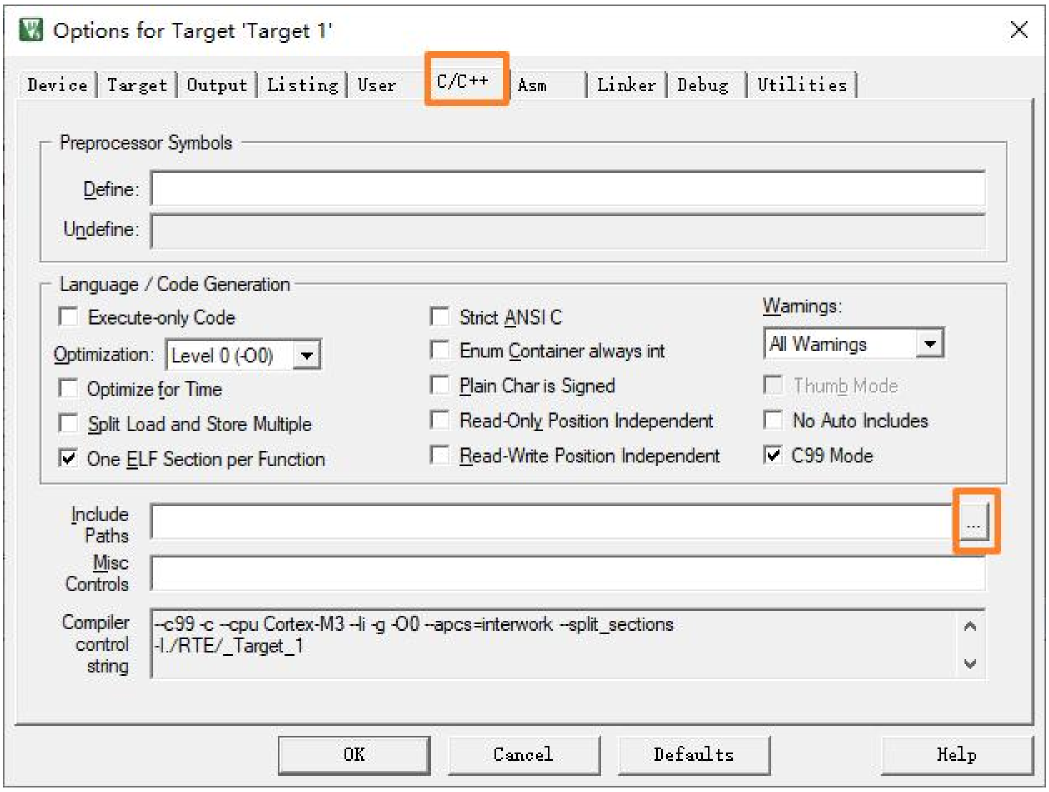
Step 10. 配置 Read/Only Memeory Areas(ROM) 和 Read/Write Memory Areas(RAM)。

***注意：**需根据您使用的型号，配置 Flash 和 SRAM 大小，图中以 128KB Flash 和 28KB SRAM 举例（其他型号容量配置可参照下表）。*


Options for Target 'Target 1'
Device Target Output Listing User C/C++ Asm Linker Debug Utilities
ARM ARMCM3
Code Generation
ARM Compiler:
Use default compiler version
Xtal (MHz): 12.0
Operating system: None
System Viewer File:
ARMCM3.svd
□ Use Custom File
□ Use Cross-Module Optimization
□ Use MicroLIB
□ Big Endian
Read/Only Memory Areas
default off-chip Start Size Startup
ROM1:
ROM2:
ROM3:
on-chip
☑ IROM1: 0x8000000 0x20000 ●
IROM2:
Read/Write Memory Areas
default off-chip Start Size NoInit
RAM1:
RAM2:
RAM3:
on-chip
☑ IRAM1: 0x20000000 0x7000 □
IRAM2:
OK
Cancel
Defaults
Help

产品型号	Flash 大小	SRAM 大小
WB32F10xx6	0x8000（32KB）	0x3000（12KB）
WB32F10xx8	0x10000（64KB）	0x5000（20KB）
WB32F10xx9	0x18000（96KB）	0x7000（28KB）
WB32F10xxB	0x20000（128KB）	0x7000（28KB）
WB32F10xxC	0x40000（256KB）	0x9000（36KB）

Step 11. 在 C/C++选项卡中配置项目的头文件包含路径。



Preprocessor Symbols

Define:

Undefine:

Language / Code Generation

- □ Execute-only Code
- Optimization: Level 0 (-O0)
- □ Optimize for Time
- □ Split Load and Store Multiple
- ☑ One ELF Section per Function
- □ Strict ANSI C
- □ Enum Container always int
- □ Plain Char is Signed
- □ Read-Only Position Independent
- □ Read-Write Position Independent
- Warnings: All Warnings
- □ Thumb Mode
- □ No Auto Includes
- ☑ C99 Mode

Include Paths: ...

Misc Controls:

Compiler control string:

```
--c99 -c --cpu Cortex-M3 -li -g -O0 --apcs=interwork --split_sections
-I./RTE/_Target_1
```

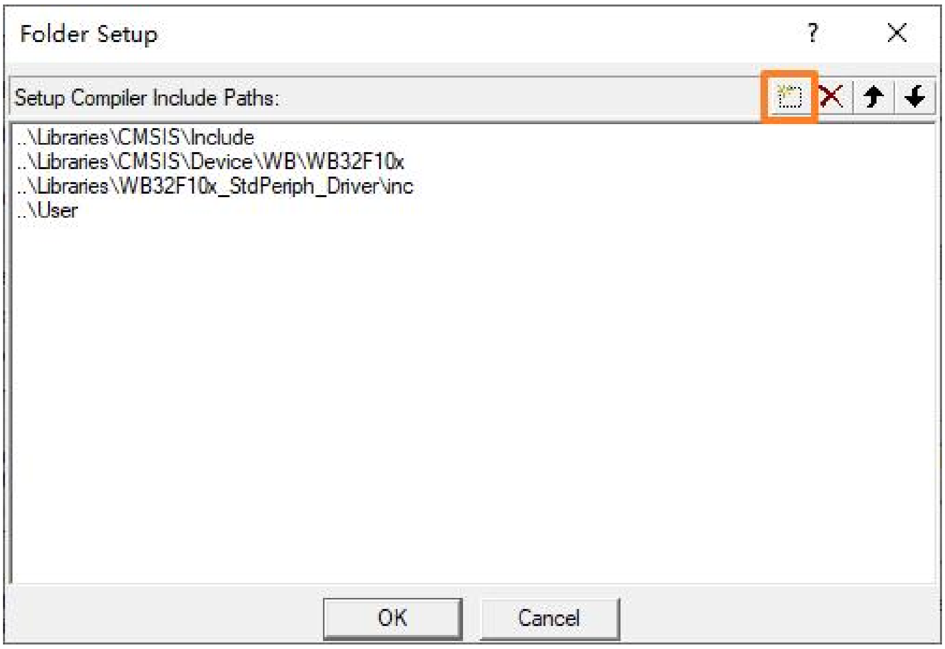
加入以下四个路径：

..\Libraries\CMSIS\Include

..\Libraries\CMSIS\Device\WB\WB32F10x

..\Libraries\WB32F10x_StdPeriph_Driver\inc

.\User



Setup Compiler Include Paths:

```
..\Libraries\CMSIS\Include
..\Libraries\CMSIS\Device\WB\WB32F10x
..\Libraries\WB32F10x_StdPeriph_Driver\inc
..\User
```

Step 12. 在 **Preprocessor Symbols** 中加入几个全局的宏定义 （这里以使用外部 12MHz 晶振，主频配置为 72MHz 举例）：

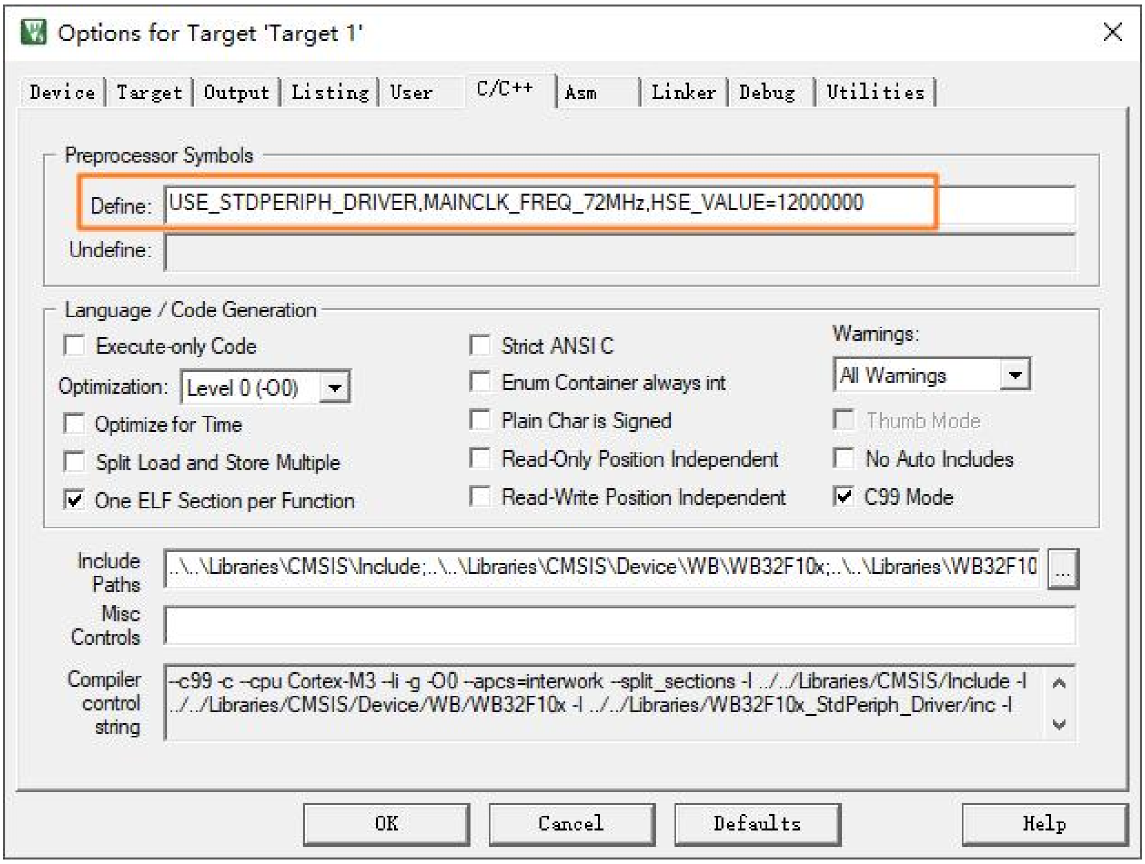
USE_STDPERIPH_DRIVER,MAINCLK_FREQ_72MHz,HSE_VALUE=12000000

USE_STDPERIPH_DRIVER 宏定义指明使用标准固件库。

MAINCLK_FREQ_72MHz 宏定义指明使用 **system_wb32f10x.c** 中预定义的 72MHz 主时钟配置函数配置主时钟。 （注意：主频不能超过您所使用型号最高支持的主频!!!）

HSE_VALUE=12000000 宏定义指明使用的外部晶振频率是 12MHz。

关于这几个定义的详解见后文。



Preprocessor Symbols

Define: USE_STDPERIPH_DRIVER,MAINCLK_FREQ_72MHz,HSE_VALUE=12000000

Undefine:

Language / Code Generation

□ Execute-only Code

Optimization: Level 0 (-O0)

□ Optimize for Time

□ Split Load and Store Multiple

☑ One ELF Section per Function

□ Strict ANSI C

□ Enum Container always int

□ Plain Char is Signed

□ Read-Only Position Independent

□ Read-Write Position Independent

Warnings: All Warnings

□ Thumb Mode

□ No Auto Includes

☑ C99 Mode

Include Paths: ..\..\Libraries\CMSIS\Include;..\..\Libraries\CMSIS\Device\WB\WB32F10x;..\..\Libraries\WB32F10

Misc Controls:

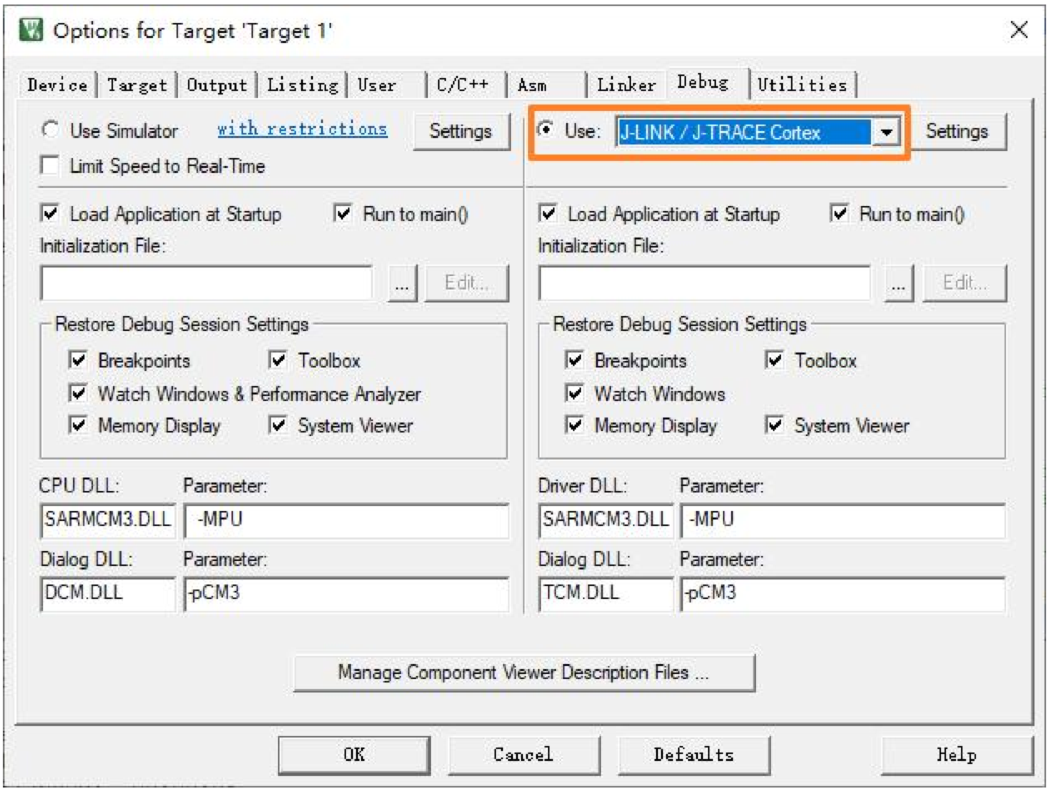
Compiler control string: --c99 -c --cpu Cortex-M3 -li -g -O0 --apcs=interwork --split_sections -I ../../Libraries/CMSIS/Include -I ../../Libraries/CMSIS/Device/WB/WB32F10x -I ../../Libraries/WB32F10x_StdPeriph_Driver/inc -I

Step 13. 点击 OK。至此，项目建立配置完成。接下来进行调试配置。

Step 14. WB32F10x 是 ARM Cortex-M3 的芯片，所以可以使用支持 Cortex-M3 的调试器调试程序（比如：JLink，ULink，CMSIS-DAP 等）。下面以 Jlink 为例演示 WB32F10x 的调试配置。

Step 15. 将 JLink 连接到电脑，使用 Jlink 的 SWD 接口与 WB32F10x 芯片连接，并给芯片上电。

Step 16. 打开 Options for Target 对话框，切换到 Debug 选项卡，选择使用 JLink 调试器。

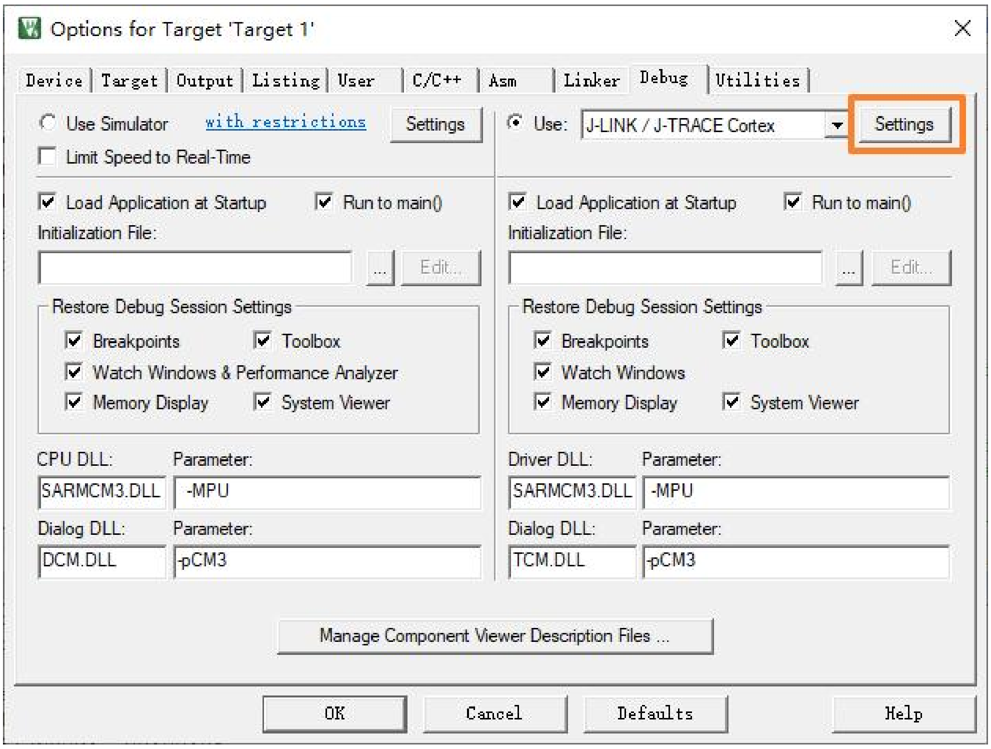


Options for Target 'Target 1'

○ Use Simulator with restrictions
□ Limit Speed to Real-Time
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows & Performance Analyzer
☑ Memory Display ☑ System Viewer
CPU DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: DCM.DLL Parameter: -pCM3

● Use: J-LINK / J-TRACE Cortex
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows
☑ Memory Display ☑ System Viewer
Driver DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: TCM.DLL Parameter: -pCM3

Step 17. 配置调试器选项。

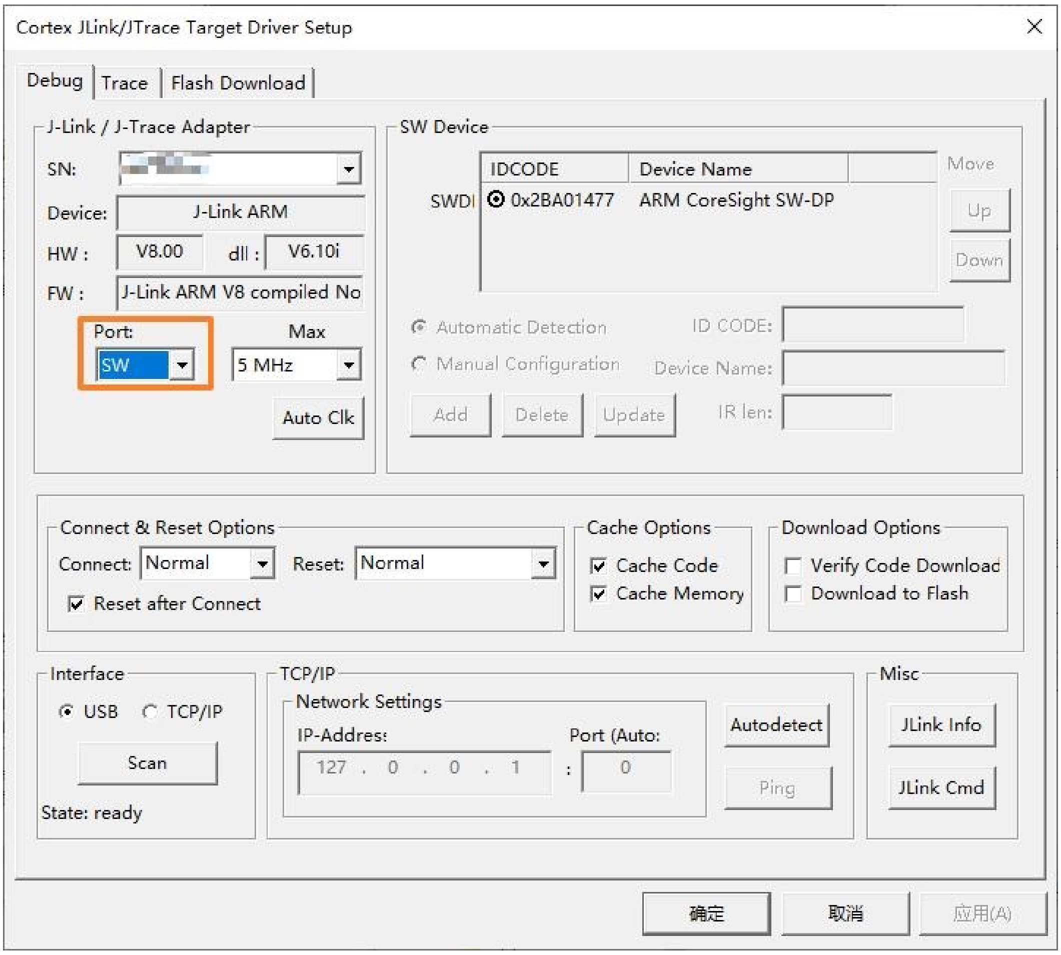


Options for Target 'Target 1'

○ Use Simulator with restrictions
□ Limit Speed to Real-Time
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows & Performance Analyzer
☑ Memory Display ☑ System Viewer
CPU DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: DCM.DLL Parameter: -pCM3

● Use: J-LINK / J-TRACE Cortex
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows
☑ Memory Display ☑ System Viewer
Driver DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: TCM.DLL Parameter: -pCM3

选择 SW 接口，可以在右侧看到 JLink 检测到了 WB32F10x 芯片。



Cortex JLink/JTrace Target Driver Setup

Debug | Trace | Flash Download

J-Link / J-Trace Adapter

SN:

Device: J-Link ARM

HW : V8.00 dll : V6.10i

FW : J-Link ARM V8 compiled No

Port: SW

Max: 5 MHz

SW Device

	IDCODE	Device Name
SWDI	⊙ 0x2BA01477	ARM CoreSight SW-DP

○ Automatic Detection ID CODE:

○ Manual Configuration Device Name:

IR len:

Connect & Reset Options

Connect: Normal Reset: Normal

☑ Reset after Connect

Cache Options

☑ Cache Code

☑ Cache Memory

Download Options

□ Verify Code Download

□ Download to Flash

Interface

● USB ○ TCP/IP

State: ready

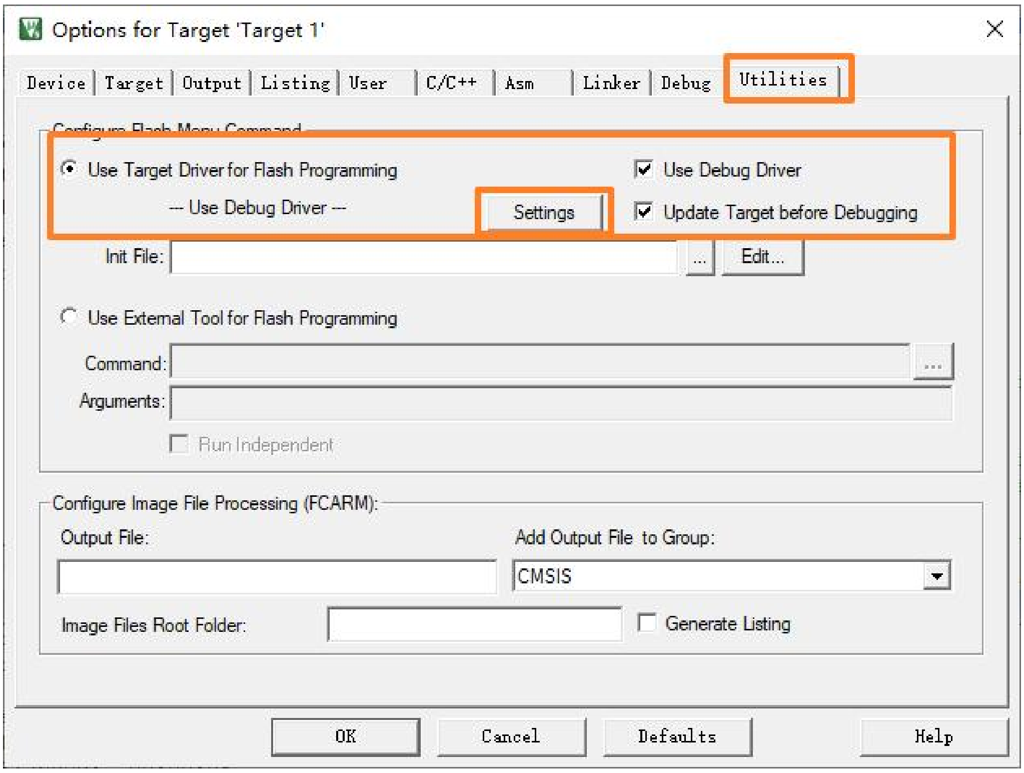
TCP/IP

Network Settings

IP-Addres: 127 . 0 . 0 . 1 Port (Auto: 0

然后点击确定。

Step 18. 在 Utilities 选项卡中，进行如图所示的设置。



Options for Target 'Target 1'

Configure Flash Menu Command

● Use Target Driver for Flash Programming

-- Use Debug Driver --

☑ Use Debug Driver

☑ Update Target before Debugging

Init File:

○ Use External Tool for Flash Programming

Command:

Arguments:

□ Run Independent

Configure Image File Processing (FCARM):

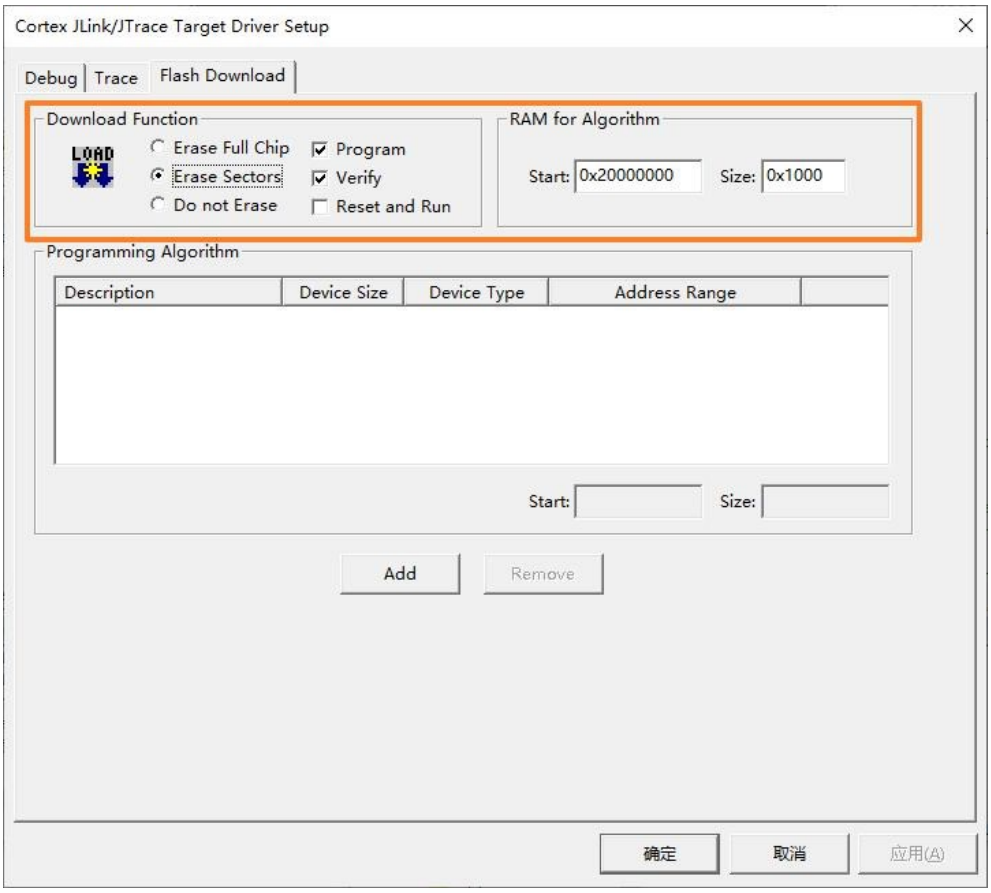
Output File:

Add Output File to Group: CMSIS

Image Files Root Folder:

□ Generate Listing

然后点击 Settings 按钮，打开烧录算法配置对话框，进行如图所示的配置。



Cortex JLink/JTrace Target Driver Setup

Download Function

○ Erase Full Chip
● Erase Sectors
○ Do not Erase

☑ Program
☑ Verify
□ Reset and Run

RAM for Algorithm

Start: 0x20000000 Size: 0x1000

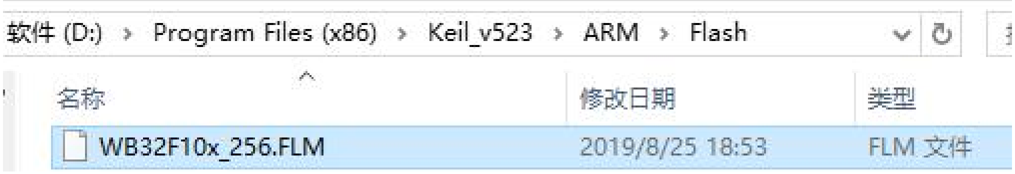
Programming Algorithm

Description	Device Size	Device Type	Address Range

Start: Size:

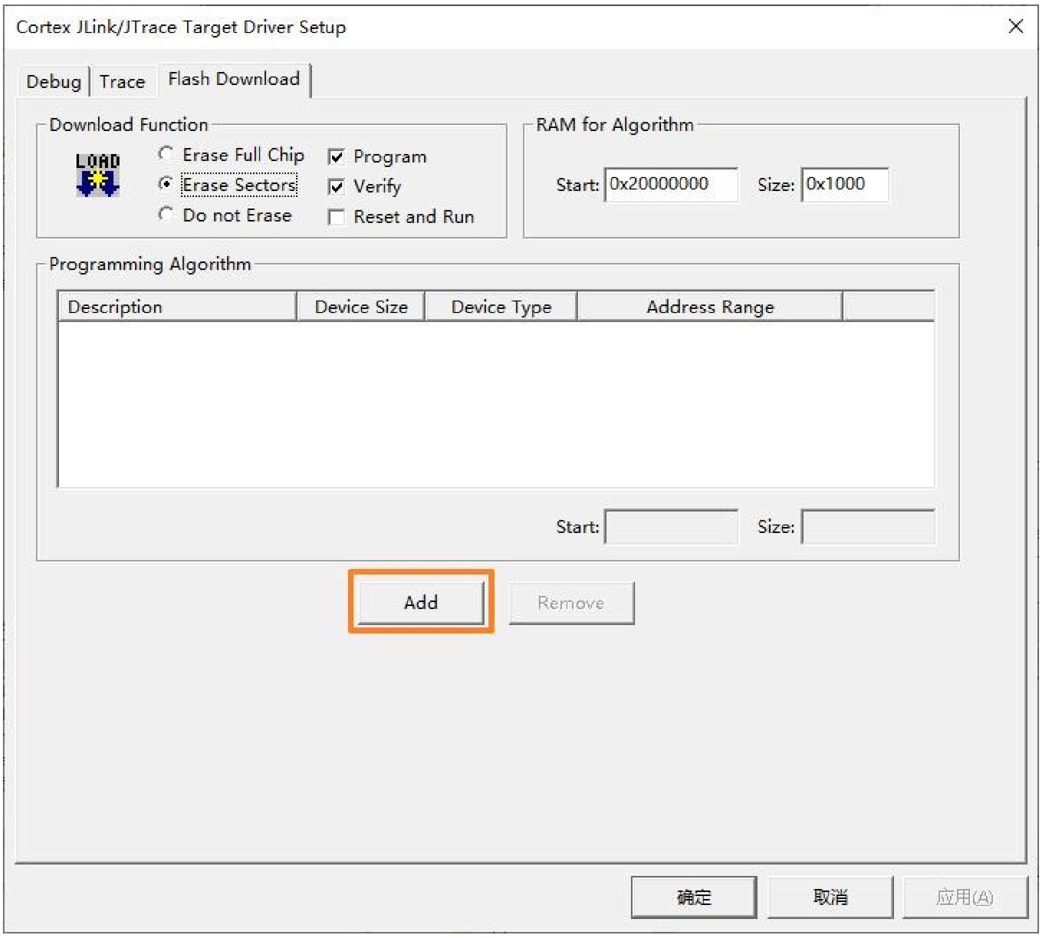
Step 19. 烧录算法配置。

将 WestBerryTech 提供的 **WB32F10x_256.FLM** 文件复制到 Keil MDK 的安装目录中对应的文件夹下（在我的电脑上的路径是 D:\Program Files (x86)\Keil_v523\ARM\Flash）


软件 (D:) › Program Files (x86) › Keil_v523 › ARM › Flash

名称	修改日期	类型
WB32F10x_256.FLM	2019/8/25 18:53	FLM 文件

然后在烧录算法配置对话框中，点击 Add 按钮。


Cortex JLink/JTrace Target Driver Setup

Download Function: ○ Erase Full Chip ● Erase Sectors ○ Do not Erase ☑ Program ☑ Verify □ Reset and Run

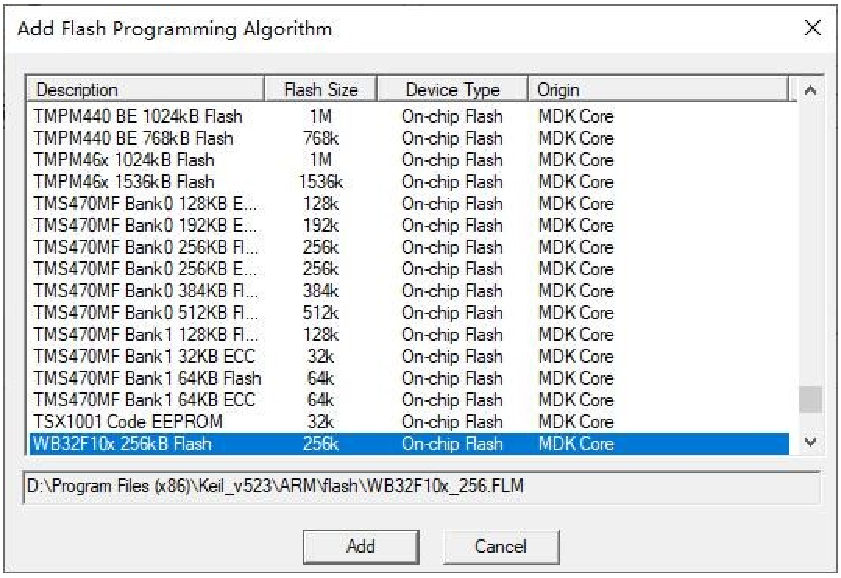
RAM for Algorithm: Start: 0x20000000 Size: 0x1000

Programming Algorithm

Description	Device Size	Device Type	Address Range

Start: Size:

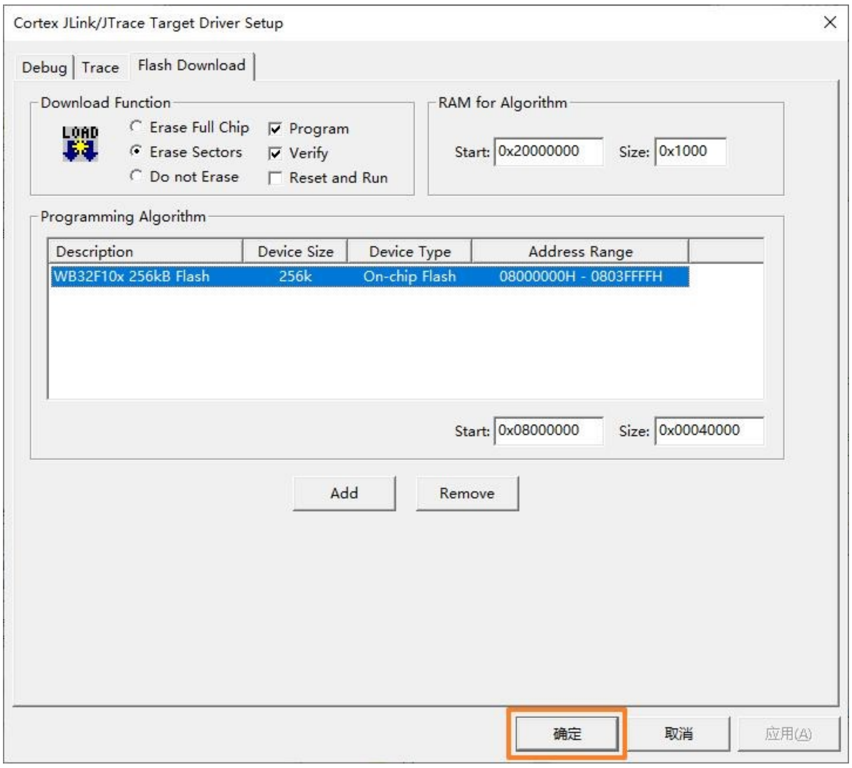
找到名为 WB32F10x 256kB Flash 的烧录算法，并点击 Add。



Add Flash Programming Algorithm

Description	Flash Size	Device Type	Origin
TMPM440 BE 1024kB Flash	1M	On-chip Flash	MDK Core
TMPM440 BE 768kB Flash	768k	On-chip Flash	MDK Core
TMPM46x 1024kB Flash	1M	On-chip Flash	MDK Core
TMPM46x 1536kB Flash	1536k	On-chip Flash	MDK Core
TMS470MF Bank0 128KB E...	128k	On-chip Flash	MDK Core
TMS470MF Bank0 192KB E...	192k	On-chip Flash	MDK Core
TMS470MF Bank0 256KB Fl...	256k	On-chip Flash	MDK Core
TMS470MF Bank0 256KB E...	256k	On-chip Flash	MDK Core
TMS470MF Bank0 384KB Fl...	384k	On-chip Flash	MDK Core
TMS470MF Bank0 512KB Fl...	512k	On-chip Flash	MDK Core
TMS470MF Bank1 128KB Fl...	128k	On-chip Flash	MDK Core
TMS470MF Bank1 32KB ECC	32k	On-chip Flash	MDK Core
TMS470MF Bank1 64KB Flash	64k	On-chip Flash	MDK Core
TMS470MF Bank1 64KB ECC	64k	On-chip Flash	MDK Core
TSX1001 Code EEPROM	32k	On-chip Flash	MDK Core
WB32F10x 256kB Flash	256k	On-chip Flash	MDK Core

D:\Program Files (x86)\Keil_v523\ARM\flash\WB32F10x_256.FLM



Cortex JLink/JTrace Target Driver Setup

Download Function

○ Erase Full Chip
● Erase Sectors
○ Do not Erase
☑ Program
☑ Verify
☐ Reset and Run

RAM for Algorithm

Start: 0x20000000 Size: 0x1000

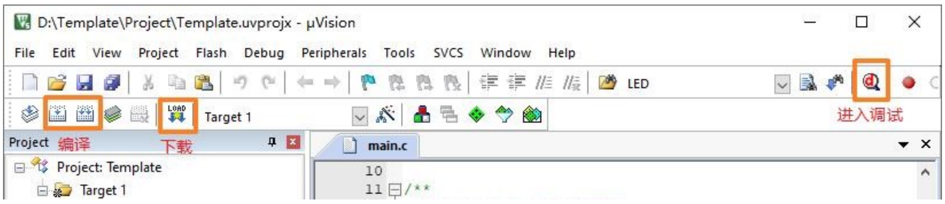
Programming Algorithm

Description	Device Size	Device Type	Address Range
WB32F10x 256kB Flash	256k	On-chip Flash	08000000H - 0803FFFFH

Start: 0x08000000 Size: 0x00040000

最后点击确定。

至此，用户可以编译，下载和调试该程序了。固件库有关配置详见下节。

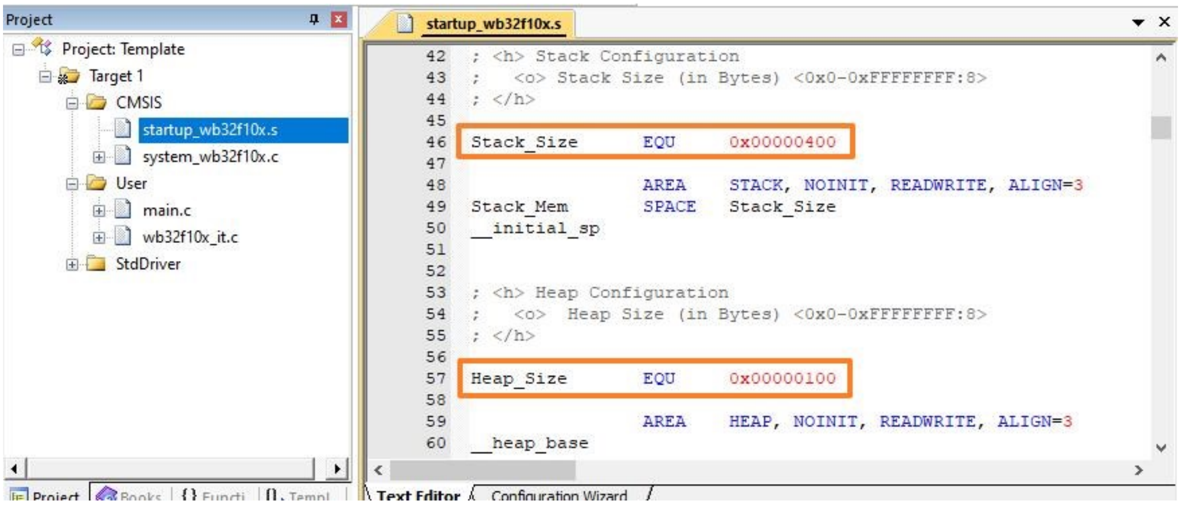


D:\Template\Project\Template.uvprojx - µVision

```
10
11 ⊟/**
```

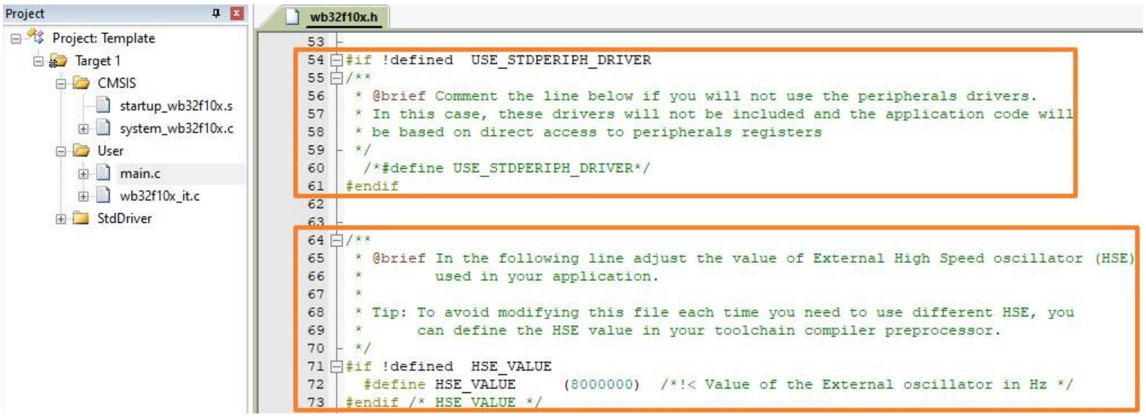
3 固件库配置详解

01. 在 startup_wb32f10x.s 可配置应用程序栈和堆的大小，如下图：

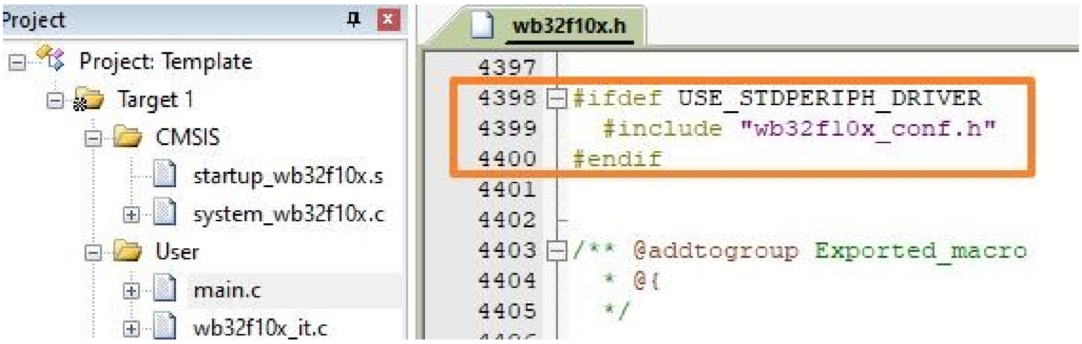


```
; <h> Stack Configuration
;   <o> Stack Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Stack_Size      EQU     0x00000400

                AREA    STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE   Stack_Size
__initial_sp


; <h> Heap Configuration
;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Heap_Size       EQU     0x00000100

                AREA    HEAP, NOINIT, READWRITE, ALIGN=3
__heap_base
```

02. 在 wb32f10x.h 中存在两个宏定义，需要用户关注。



```

#if !defined  USE_STDPERIPH_DRIVER
/**
 * @brief Comment the line below if you will not use the peripherals drivers.
 * In this case, these drivers will not be included and the application code will
 * be based on direct access to peripherals registers
 */
  /*#define USE_STDPERIPH_DRIVER*/
#endif


/**
 * @brief In the following line adjust the value of External High Speed oscillator (HSE)
 *        used in your application.
 *
 * Tip: To avoid modifying this file each time you need to use different HSE, you
 *      can define the HSE value in your toolchain compiler preprocessor.
 */
#if !defined  HSE_VALUE
  #define HSE_VALUE    (8000000)  /*!< Value of the External oscillator in Hz */
#endif /* HSE_VALUE */
```

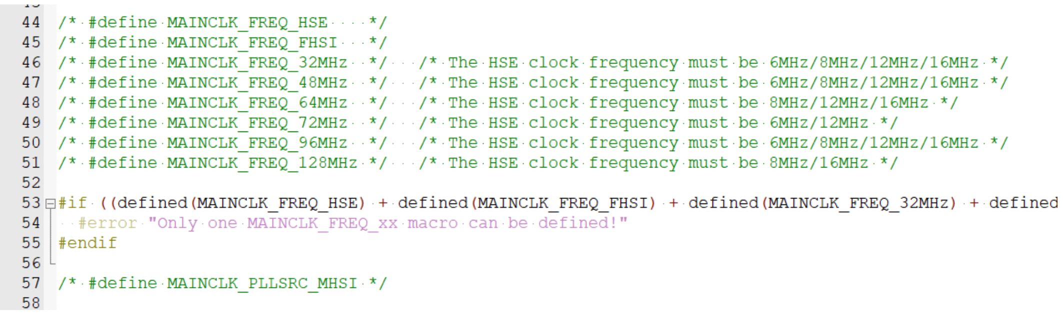


```

#ifdef USE_STDPERIPH_DRIVER
  #include "wb32f10x_conf.h"
#endif


/** @addtogroup Exported_macro
  * @{
  */
```

USE_STDPERIPH_DRIVER 定义这个宏表示应用程序需要使用固件库中的外设驱动，且会在项目中包含 **wb32f10x_conf.h** 头文件。

HSE_VALUE 该宏用于指定 WB32F10x 芯片外接晶振的频率。默认情况下，固件库假定外部 HSE 晶振的频率是 8MHz。 *如果用户外接晶振不是 8MHz，务必修改或在编译器全局预定义处覆盖该定义！！！*

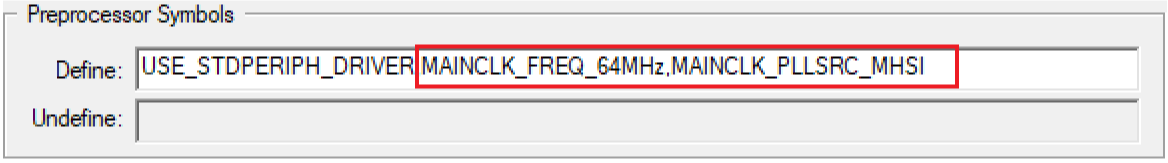
03. 在 **system_wb32f10x.c** 中有几处定义需要用户关注。



```
/* #define MAINCLK_FREQ_HSE    */
/* #define MAINCLK_FREQ_FHSI   */
/* #define MAINCLK_FREQ_32MHz  */    /* The HSE clock frequency must be 6MHz/8MHz/12MHz/16MHz */
/* #define MAINCLK_FREQ_48MHz  */    /* The HSE clock frequency must be 6MHz/8MHz/12MHz/16MHz */
/* #define MAINCLK_FREQ_64MHz  */    /* The HSE clock frequency must be 8MHz/12MHz/16MHz */
/* #define MAINCLK_FREQ_72MHz  */    /* The HSE clock frequency must be 6MHz/12MHz */
/* #define MAINCLK_FREQ_96MHz  */    /* The HSE clock frequency must be 6MHz/8MHz/12MHz/16MHz */
/* #define MAINCLK_FREQ_128MHz */    /* The HSE clock frequency must be 8MHz/16MHz */

#if ((defined(MAINCLK_FREQ_HSE) + defined(MAINCLK_FREQ_FHSI) + defined(MAINCLK_FREQ_32MHz) + defined
  #error "Only one MAINCLK_FREQ_xx macro can be defined!"
#endif

/* #define MAINCLK_PLLSRC_MHSI */

```

MAINCLK_FREQ_* 这些宏定义用以配置在 SystemInit 函数中如何配置芯片主时钟的频率。只能选择定义其中的一个（如果不定义任何一个，那么芯片主时钟是 MHSI）。可以在编译器全局预定义处给出定义。这些宏定义对芯片外部晶振是有要求，比如要定义 **MAINCLK_FREQ_72MHz**，那么芯片外部晶振频率必须是 **12MHz 或 6MHz**（切记：也要覆盖 **HSE_VALUE** 的定义）。

MAINCLK_PLLSRC_MHSI：这是在标准固件库 V0.1.9 以上的版本，加入的使用内部晶振通过 PLL 倍频的宏定义。下面以使用 MHSI 倍频到 64MHz 为例：



Preprocessor Symbols

Define: USE_STDPERIPH_DRIVER,MAINCLK_FREQ_64MHz,MAINCLK_PLLSRC_MHSI

Undefine:

VECT_TAB_SRAM 定义这个宏表示将中断向量表映射到 SRAM 中（在 SRAM 中运行的工程才需要定义这个宏）。

VECT_TAB_OFFSET 该宏用以设置中断向量表起始地址偏移（相对于 Flash 或 SRAM 的起始地址）。

版本历史

Revision	Date	Description
1.0	2020/09/05	Initial Release
1.2	2022/06/30	Added MAINCLK_PLLSRC_MHSI define

免责声明

本文档中的信息仅针对 WB 产品提供。本文档，包括本文档中描述的任何 WB 产品（“产品”），根据中华人民共和国和全球其他司法管辖区的知识产权法律和条约，归属 WB 所有。常州韦斯佰瑞电子科技有限公司及其子公司（“WB”）保留随时对本文档以及文档中所描述的产品与服务进行更改、更正、修改或改进的权利，恕不另行通知。WB 不承担任何因应用程序或使用本文档中描述的任何产品引起的任何责任。

购买者应对 WB 产品与服务的选择、选型和使用承担全部责任，并且 WB 不承担对 WB 产品与服务的选择、选型和使用的任何责任。

本文档未通过禁反言或其他方式对任何知识产权授予任何明示或暗示的许可。如果本文档的任何部分提及任何第三方产品或服务，则不应视为 WB 授予使用此类第三方产品或服务或其中包含的任何知识产权的许可，或视为涵盖在此类第三方产品或服务或其中包含的任何知识产权的任何方式。

除适用协议中明确规定的定制产品外，产品仅为普通商业、工业、个人或家庭应用而设计、开发或制造。产品并非设计、意图或授权用作设计或用于操作武器、武器系统、核装置、原子能控制仪器、燃烧控制仪器、飞机或宇宙飞船仪器、运输仪器、交通信号系统中仪器、生命支持设备或系统、其他医疗设备或系统（包括复苏设备和外科植入物)、污染控制或有害物管理、由于设备或仪器的故障可能导致人身伤害、死亡、财产损失或环境破坏的其他用途。

转授 WB 产品的条款与本文档中规定的声明和/或技术特征不同的，将立即使 WB 对此处描述的 WB 产品或服务的任何保证失效，并且不得以任何方式产生或扩展 WB 的任何责任。