










MG32F10x Developer Guide

1 Standard peripheral library

As the figure below shows MG32F10x standard peripheral library folder structure.

- ▼  MG32F10x_StdPeriph_Lib_V0.1.5
 - ▼  Libraries
 - >  CMSIS
 - >  MG32F10x_StdPeriph_Driver
 - >  MG32F10x_USBDevice_Driver
 - ▼  Project
 - >  MG32F10x_StdPeriph_Examples
 - >  MG32F10x_StdPeriph_Template
 - >  Utilities

Documentation folder contains MG32F10x standard peripheral library's documentation.

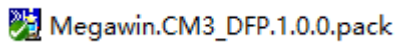
Libraries folder contains **CMSIS**, **MG32F10x_StdPeriph_Driver**, and **MG32F10x_USBDevice_Driver** subfolder. **CMSIS** folder contains startup file, chip head file. **MG32F10x_StdPeriph_Driver** folder contains MG32F10x standard peripheral library code file. **MG32F10x_USBDevice_Driver** folder contains MG32F10x USB device protocol stack code.

Project folder contains **MG32F10x_StdPeriph_Examples** and **MG32F10x_StdPeriph_Template** subfolder. **MG32F10x_StdPeriph_Examples** folder contains sample code provided by *megawin* Official. **MG32F10x_StdPeriph_Template** folder contains a template project.

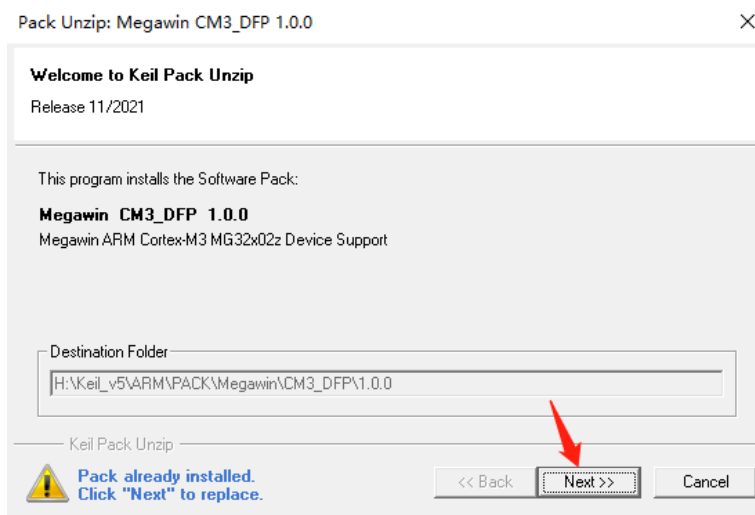
Utilities folder contains some public code.

2 Use Keil MDK to build a project

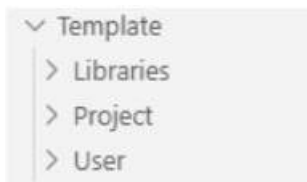
- 1) Set up MG32F10x data pack.
Open Megawin.CM3.DFP.1.0.0.pack.



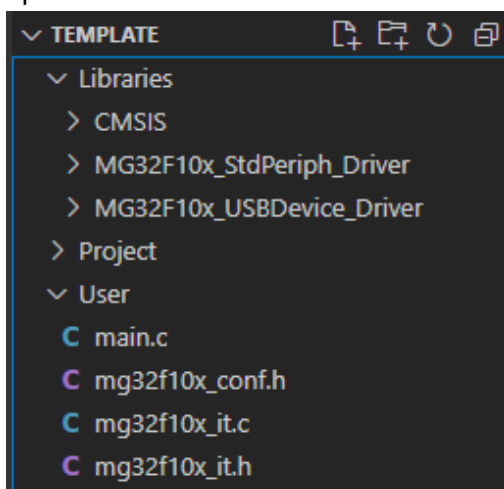
Click Next, and wait for finish.



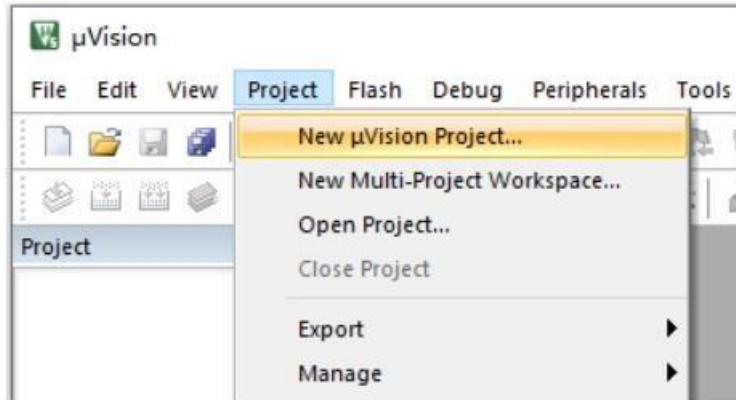
- 2) New a folder named "Template" to contain project.
- 3) New Libraries, Project and User subfolder in Template folder. (User can also customize their own folder structure)



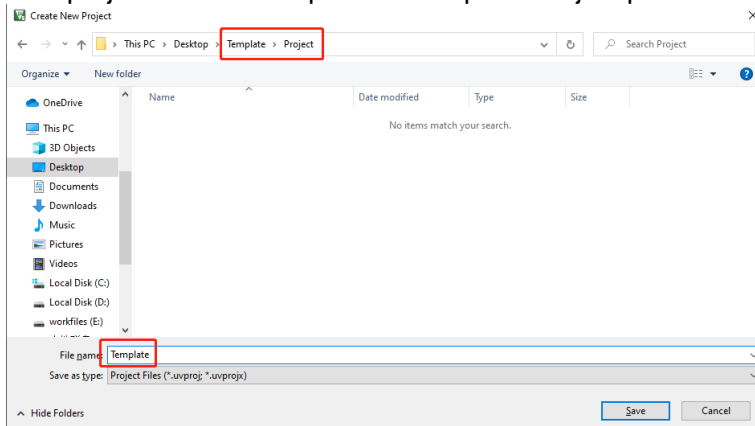
- 4) Copy MG32F10x standard peripheral library's Libraries folder's content to Template\Libraries.
- 5) Copy MG32F10x standard peripheral library's Project\MG32F10x_StdPeriph_Template folder's content to Template\User



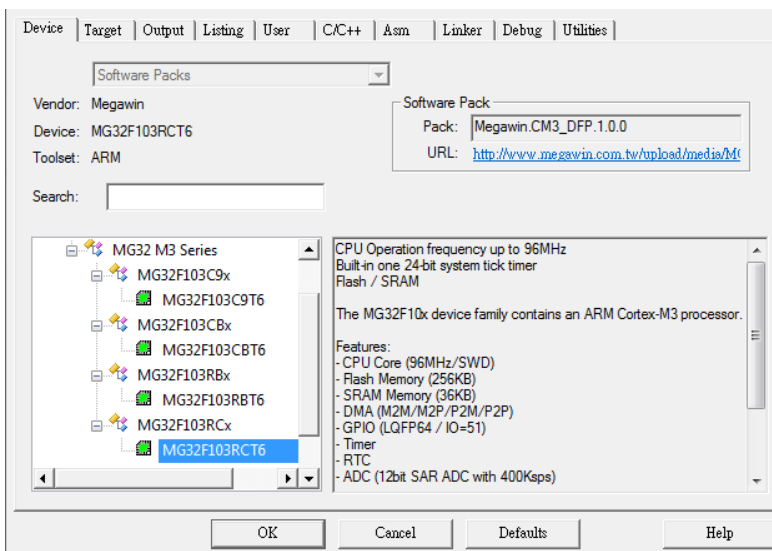
6) Open Keil MDK, New uVision project.



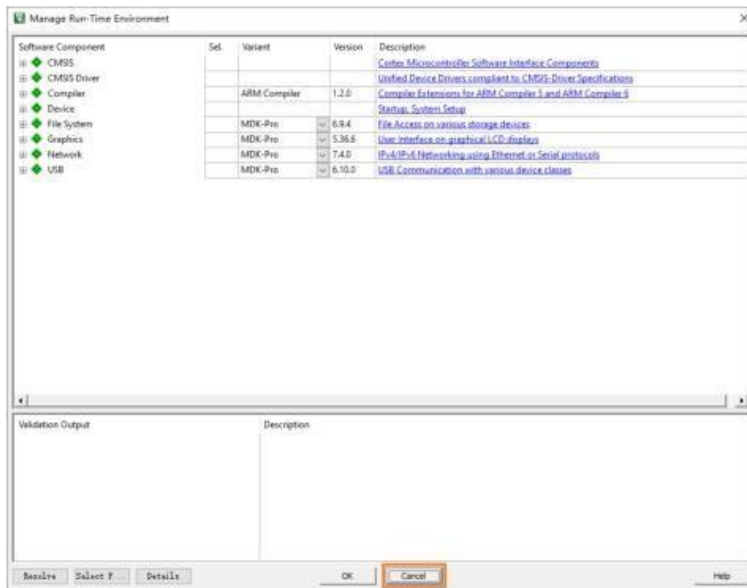
7) New a project named Template at Template\Project path.



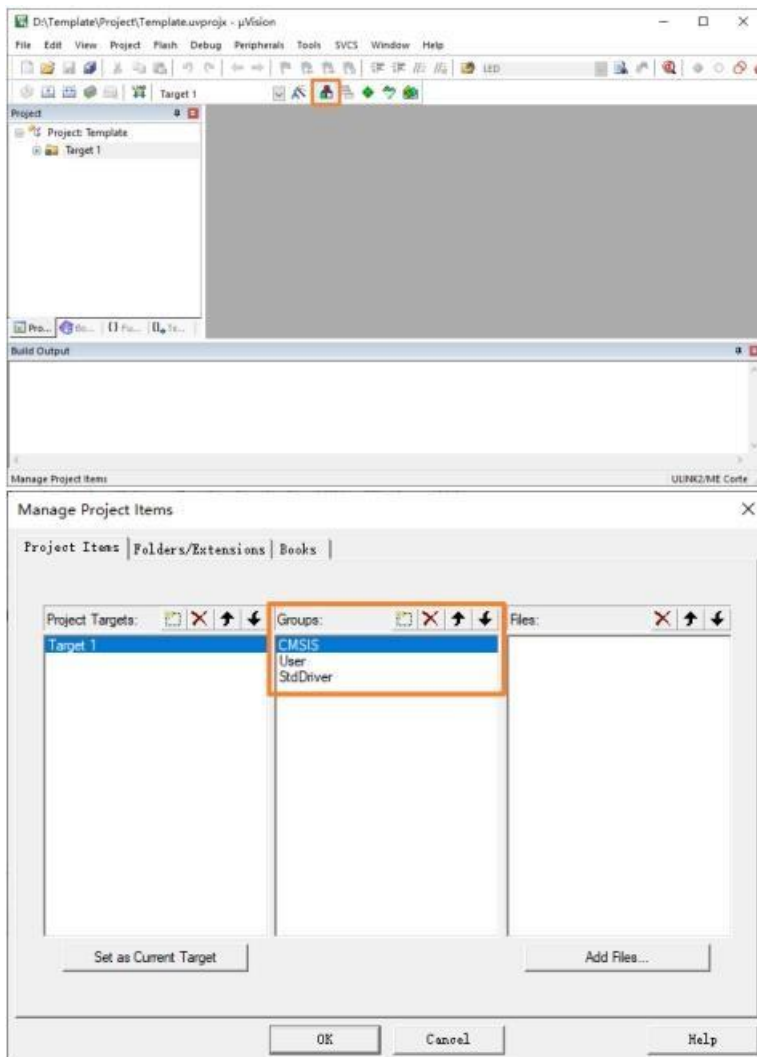
8) Select device which you need in this project, then click OK.



9) Then it will show Manage Run-Time Environment window. Click Cancel.



10) New 3 groups: CMSIS, User and StdDriver.



11) Add peripheral library file into Group

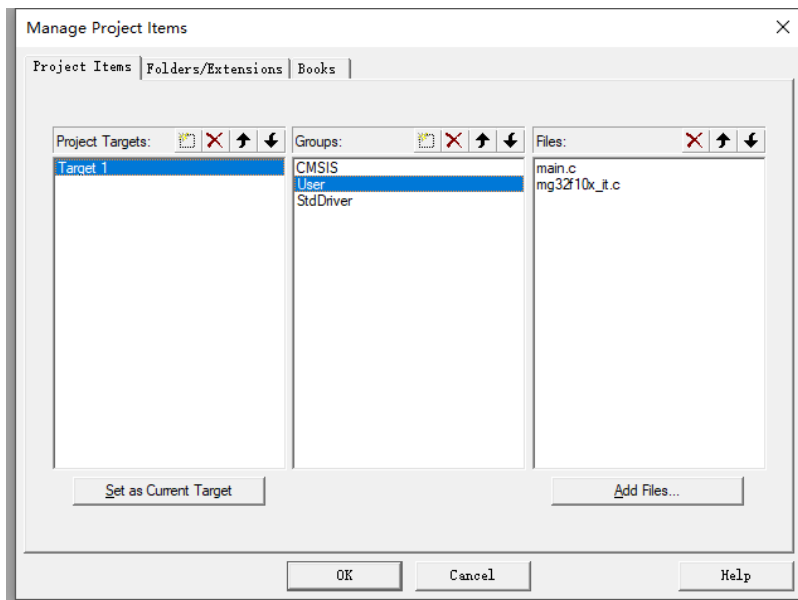
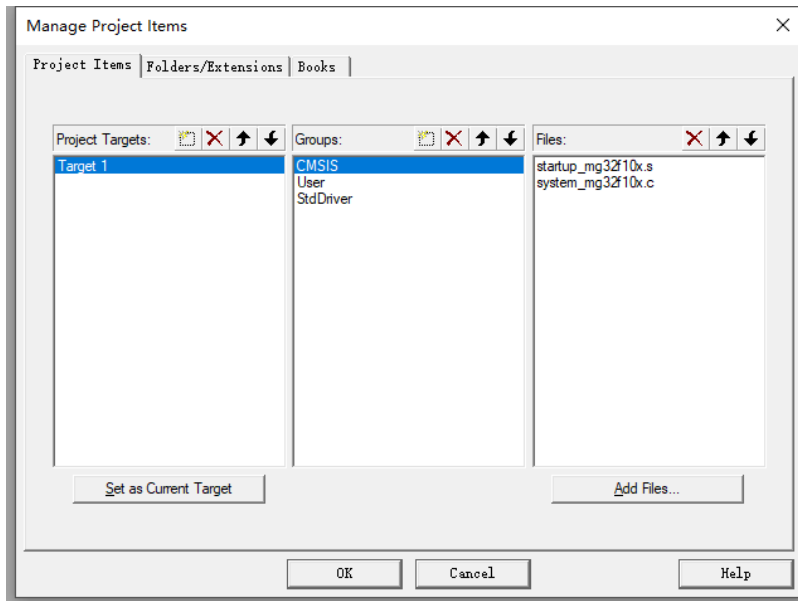
Add in CMSIS Group:

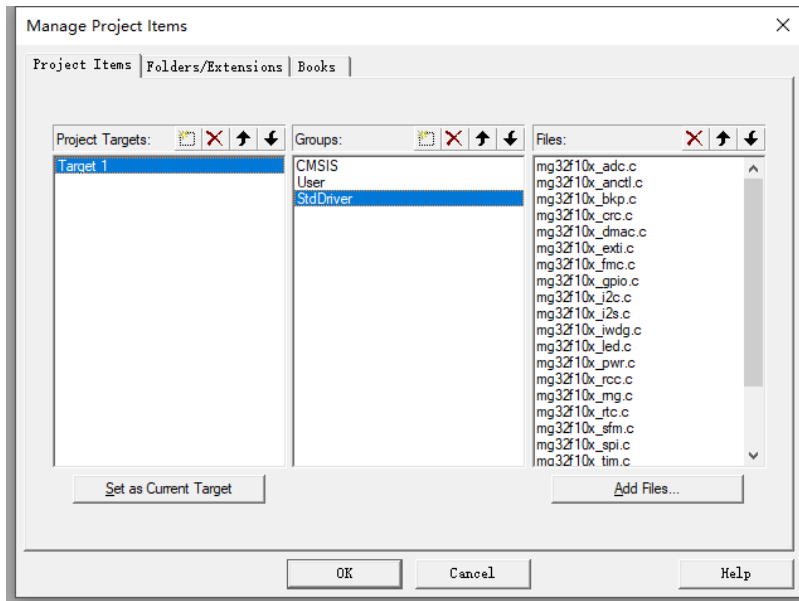
Template\Libraries\CMSIS\Device\MG\MG32F10x\startup\arm\startup_mg32f10x.s
Template\Libraries\CMSIS\Device\MG\MG32F10x\system_mg32f10x.c

Add in User Group:

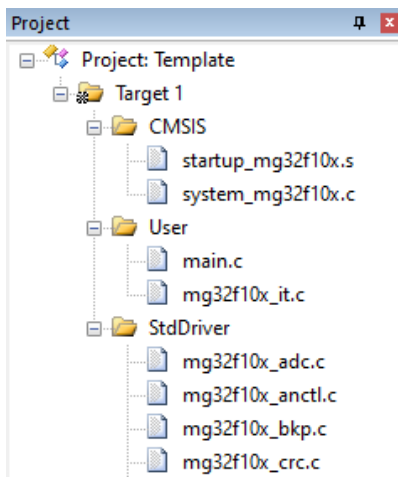
Template\User\main.c
Template\User\mg32f10x_it.c

Add in StdDriver Group Group:

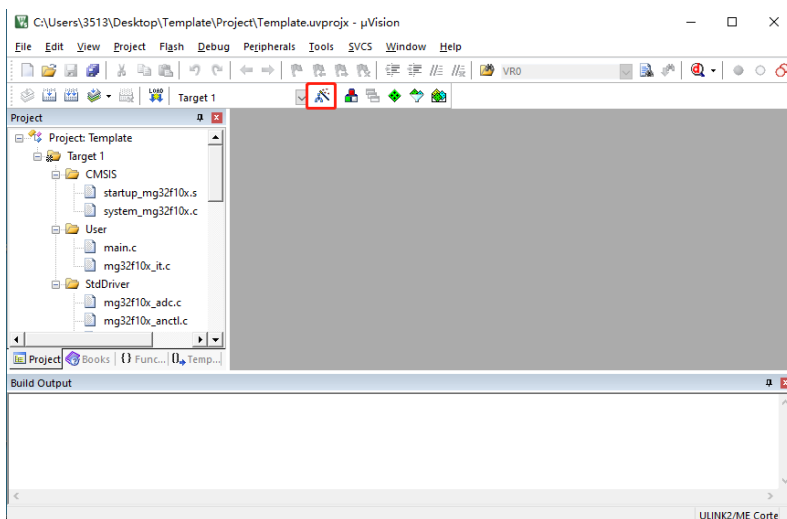
Template\Libraries\MG32F10x StdPeriph Driver\src folder's every .c file.




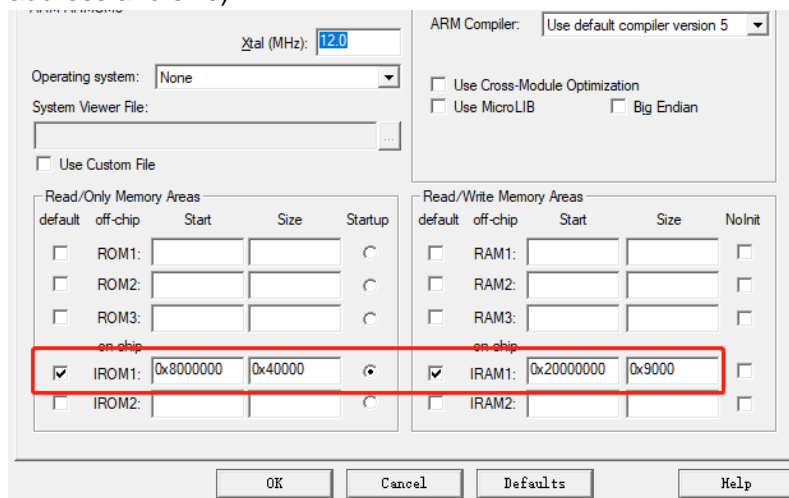
12) Final project structure as below:



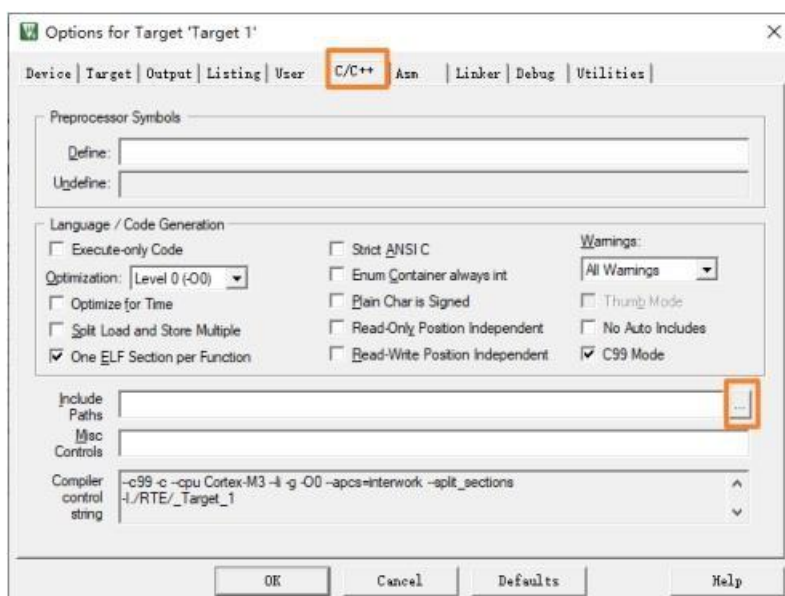
13) Open Options for Target window.



- 14) Configure Read/Only Memory Areas and Read/Write Memory Areas (Configure Flash and SRAM start address and size).



- 15) Configure project head file path in C/C++ tab.



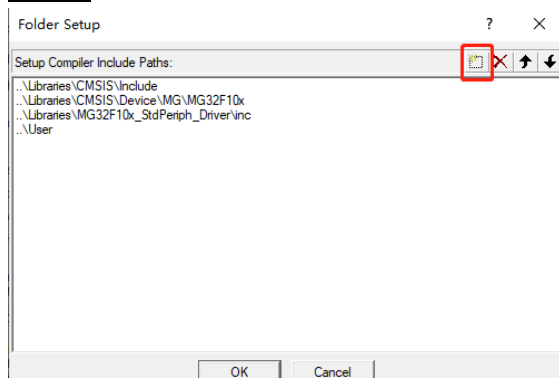
Add 4 path as below:

..\Libraries\CMSIS\Include

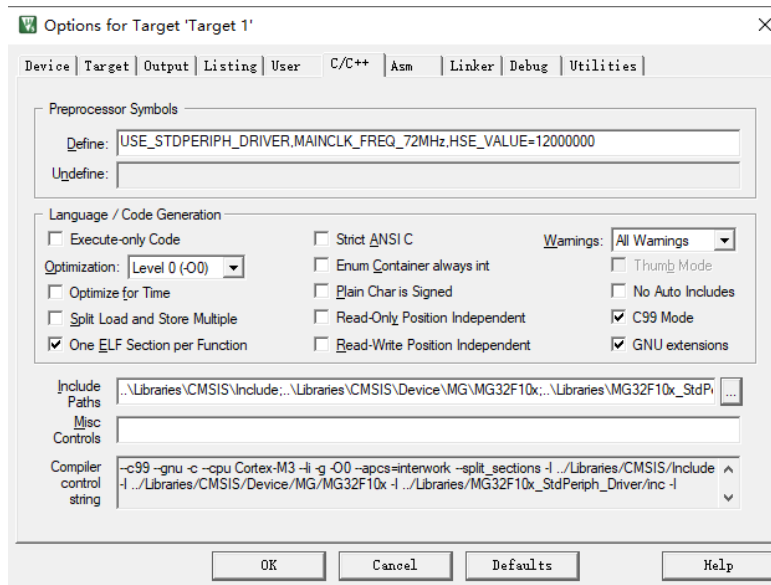
..\Libraries\CMSIS\Device\MG\MG32F10x

..\Libraries\MG32F10x_StdPeriph_Driver\inc

..\User



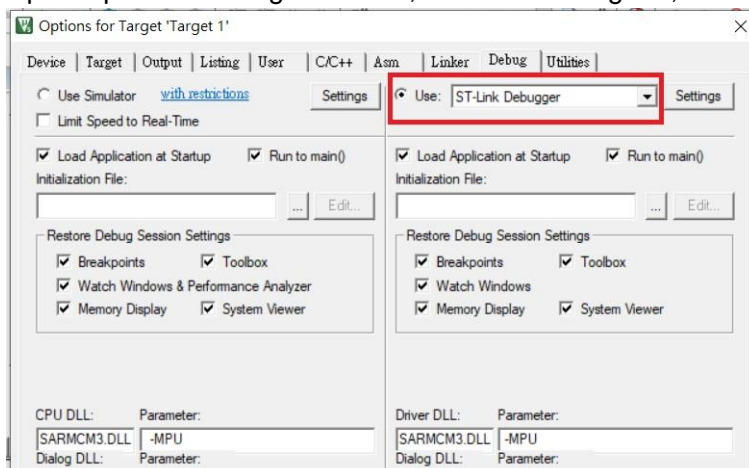
- 16) Add define in Preprocessor Symbols: **USE_STDPERIPH_DRIVER,MAINCLK_FREQ_72MHz,HSE_VALUE=12000000** (The details of these two definitions are described later)



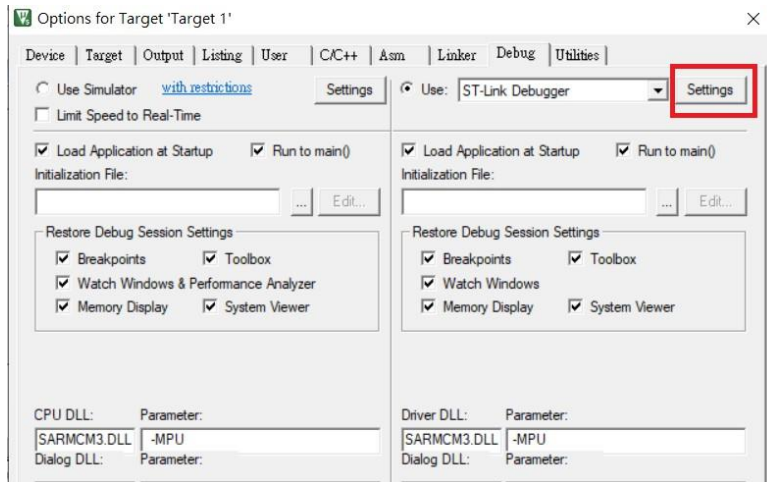
- 17) Click OK. So far, the project configuration is completed. Next, configure debug function.
 18) MG32F10x is embedded a CPU core of ARM Cortex-M3 processor. So MG32F10x supports kinds of debugger which support Cortex-M3 core MCU (such as: JLink, ULink, STLink and CMSIS-DAP). Take ST-link as example to demonstrate the MG32F10x debugging configuration. **Please refer the How_to Use_J_Link_Debug_Download_MG32F10x.pdf under Document folder in SDK if you need to use J-Link to debug.**

- 19) Connect ST-Link to PC, connect ST-Link and MG32F10x through SWD interface. Power on the MCU.

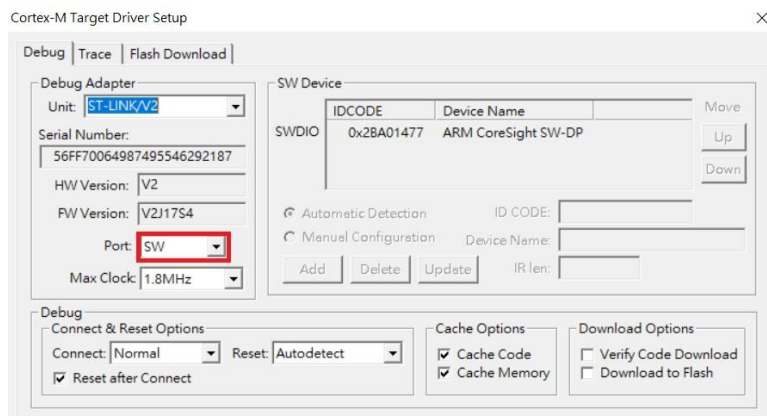
- 20) Open Options for Target window, switch to Debug tab, select ST-Link debger.



21) Setting debugger configuration.

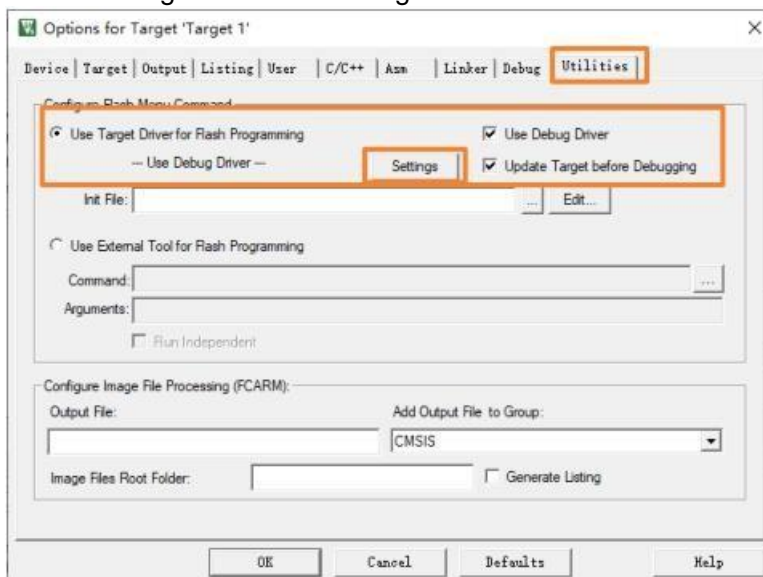


Select SW Port, then MG32F10x should be detected by ST-Link and be seen on SW Device.

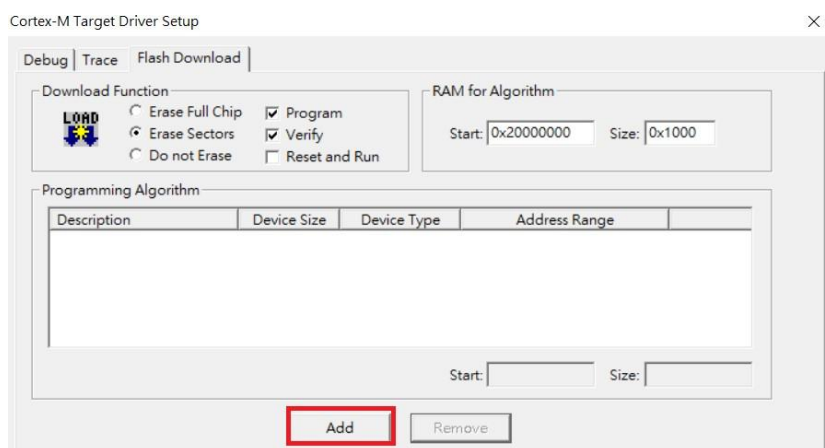
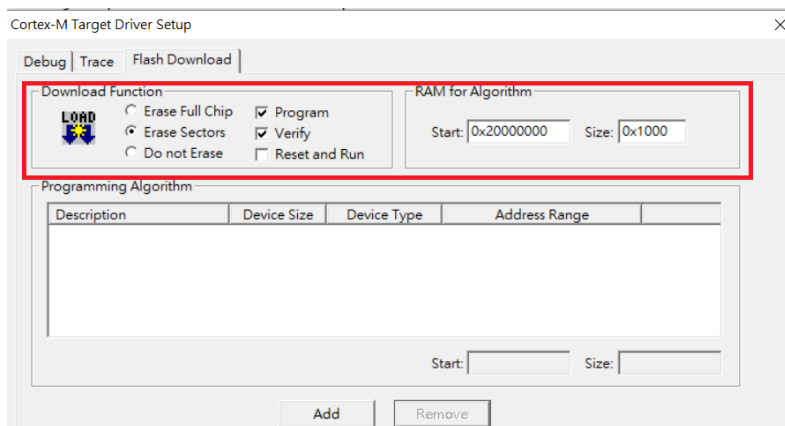


Then click OK.

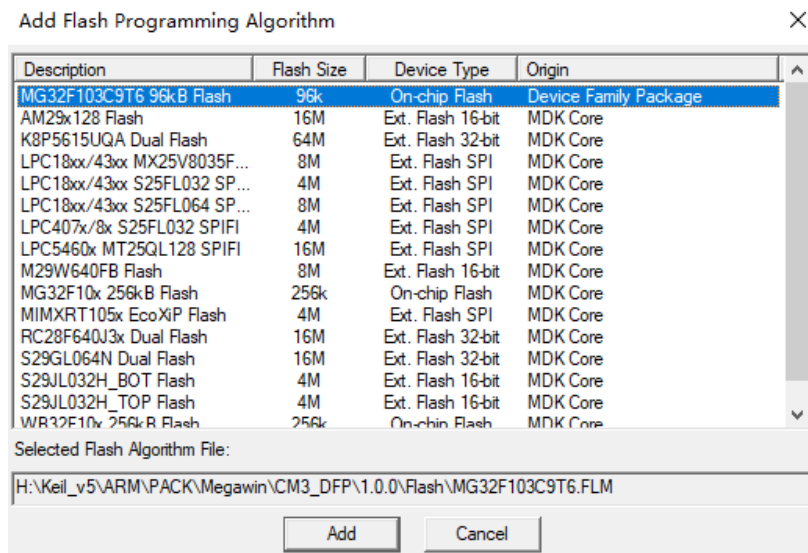
22) Refer to the figure below to configure Utilities tab.

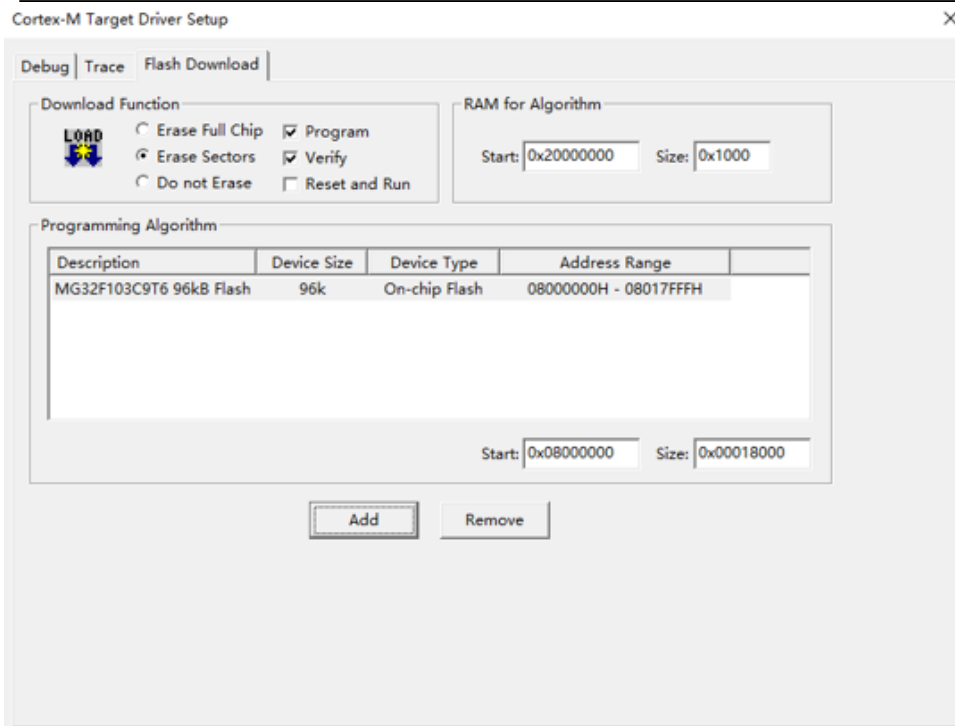


Then click Settings button, switch to Flash Download tab, and configure as below.



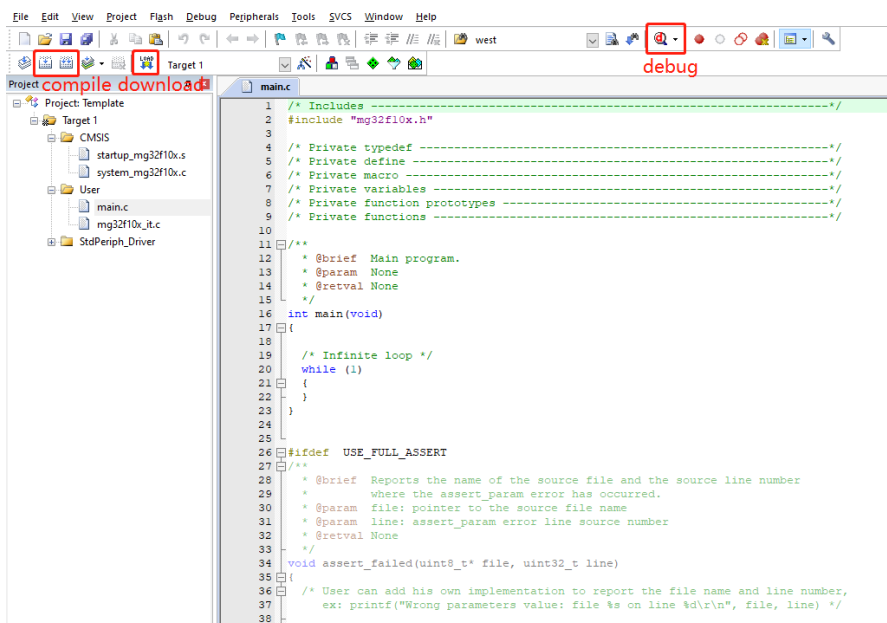
Find the device you need and click Add.





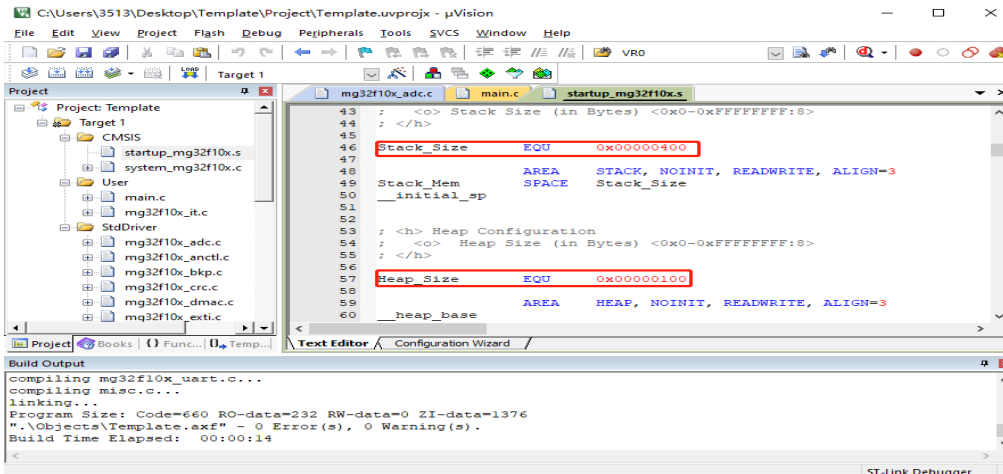
Click OK.

23) Now, user should be able to compile, download and debug their own code.



3 peripheral library configuration

1) startup_mg32f10x.s could be used for configure application stack and heap size as below.



```

43 : <0> Stack Size (in Bytes) <0x0-0xFFFFFFFF:8>
44 : </h>
45
46 Stack_Size EQU 0x00000400
47
48 AREA STACK, NOINIT, READWRITE, ALIGN=3
49 Stack_Mem SPACE Stack_Size
50 __initial_sp
51
52
53 : <h> Heap Configuration
54 : <0> Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
55 : </h>
56
57 Heap_Size EQU 0x00000100
58
59 AREA HEAP, NOINIT, READWRITE, ALIGN=3
60 __heap_base

```

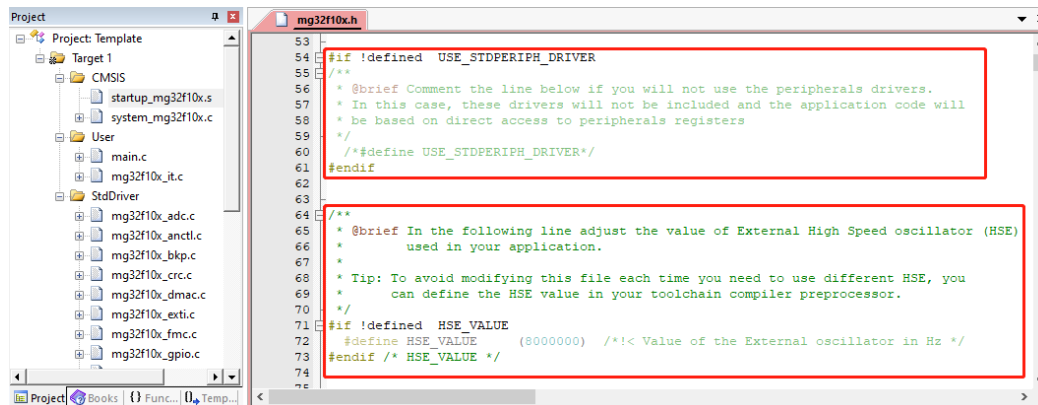
Build Output

```

Compiling mg32f10x_uart.c...
Compiling misc.c...
Linking...
Program Size: Code=660 RO-data=232 RW-data=0 ZI-data=1376
".\Objects\Template.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:14

```

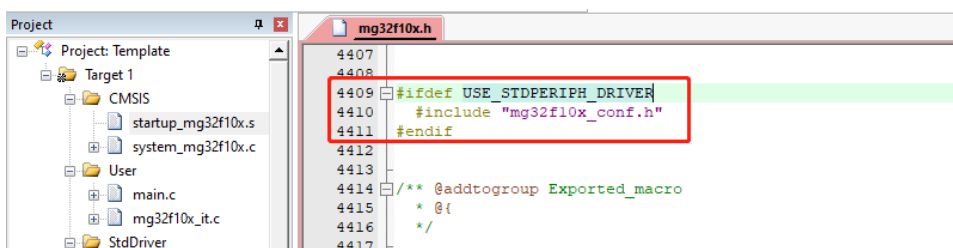
2) Two macro definition in mg32f10x.h should be noticed.



```

53
54 #if !defined USE_STDPERIPH_DRIVER
55 /**
56  * @brief Comment the line below if you will not use the peripherals drivers.
57  * In this case, these drivers will not be included and the application code will
58  * be based on direct access to peripherals registers
59  */
60 /*#define USE_STDPERIPH_DRIVER*/
61 #endif
62
63
64 /**
65  * @brief In the following line adjust the value of External High Speed oscillator (HSE)
66  * used in your application.
67  *
68  * Tip: To avoid modifying this file each time you need to use different HSE, you
69  * can define the HSE value in your toolchain compiler preprocessor.
70  */
71 #if !defined HSE_VALUE
72 #define HSE_VALUE (8000000) /*!< Value of the External oscillator in Hz */
73 #endif /* HSE_VALUE */
74
75

```



```

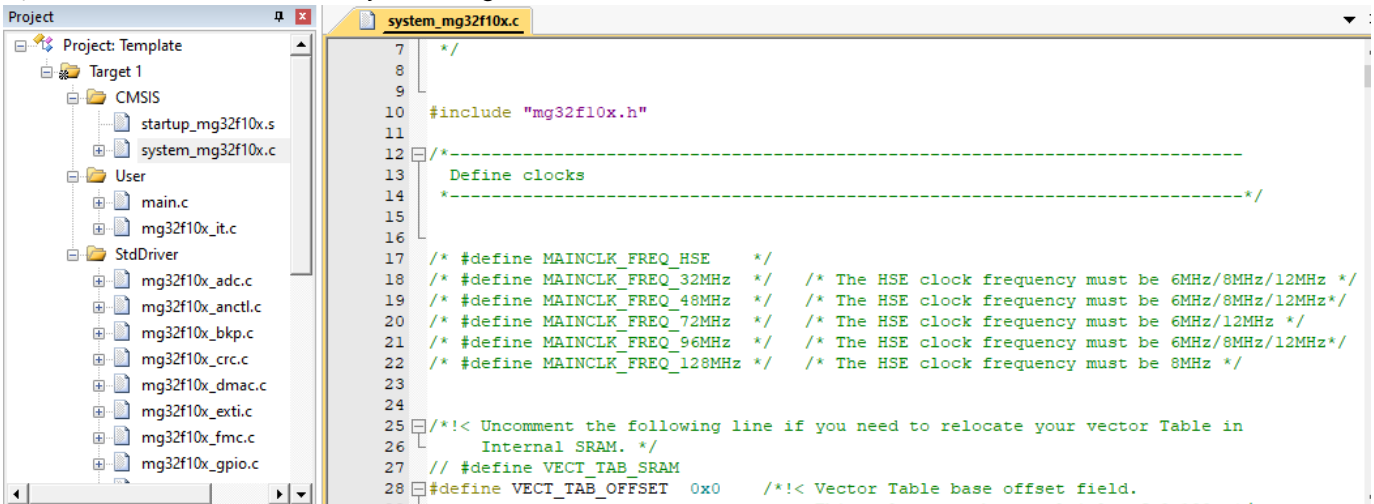
4407
4408
4409 #ifdef USE_STDPERIPH_DRIVER
4410 #include "mg32f10x_conf.h"
4411 #endif
4412
4413
4414 /** @addtogroup Exported_macro
4415  * @{
4416
4417

```

USE_STDPERIPH_DRIVER This macro definition represents that this application will use standard driver, and this project will include mg32f10x_conf.h head file.

HSE_VALUE This macro definition is used for define MG32F10x Xtal's frequency. Peripheral library will set external HSE Xtal frequency is **8MHz** by default. User should modify this definition here or at the preprocessor symbol if the Xtal which is using is not **8MHz**.

3) Some macro definition in system_mg32f10x.c should be noticed.



```

7  /*
8
9
10 #include "mg32f10x.h"
11
12 /*-----
13 Define clocks
14 -----*/
15
16
17 /* #define MAINCLK_FREQ_HSE */
18 /* #define MAINCLK_FREQ_32MHz */ /* The HSE clock frequency must be 6MHz/8MHz/12MHz */
19 /* #define MAINCLK_FREQ_48MHz */ /* The HSE clock frequency must be 6MHz/8MHz/12MHz */
20 /* #define MAINCLK_FREQ_72MHz */ /* The HSE clock frequency must be 6MHz/12MHz */
21 /* #define MAINCLK_FREQ_96MHz */ /* The HSE clock frequency must be 6MHz/8MHz/12MHz */
22 /* #define MAINCLK_FREQ_128MHz */ /* The HSE clock frequency must be 8MHz */
23
24
25 /*!< Uncomment the following line if you need to relocate your vector Table in
26 Internal SRAM. */
27 // #define VECT_TAB_SRAM
28 #define VECT_TAB_OFFSET 0x00 /*!< Vector Table base offset field.
  
```

MAINCLK_FREQ_* This macro definition is used for configure the main clock frequency of MCU. Only one of the definitions can be selected at the preprocessor symbol (If no definition is selected, the chip main clock would be MHSI). Should be noticed that all these frequency definitions have requirements for the Xtal of MCU. For example, the Xtal frequency must be 6/12MHz when user select **MAINCLK_FREQ_72MHz** (Notify: macro definition **HSE_VALUE** should also be modified).

VECT_TAB_SRAM This macro definition represents that interrupt vector will be mapping to SRAM. (Only while the project needs to be run in SRAM)

VECT_TAB_OFFSET This macro definition is used for configure interrupt vector table start address offset. (related to Flash or SRAM start address)