**How to create a Keil workspace from SPL?**

---STM32 F103 VC for example

Workspace is a space contains necessary files for a particular project, usually a folder. The document aims to provide a step by step method to create a workspace folder that can be a template for most STM32 F1XX projects.

**0. Pre-request:**

a. First, you need a Standard Peripheral Library (SPL) for your device. Go download from ST website!

b. Of course you should complete Keil installation.

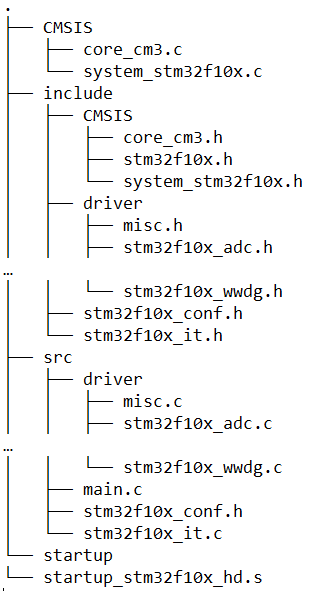
c. There are some references you should have a look at…

i. some notes

ii. official docs

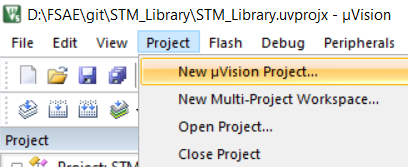
**1. Initialize your workspace:**

a. build a workspace folder with following structures: (All the files can be found in SPL, just use Ctrl+F)

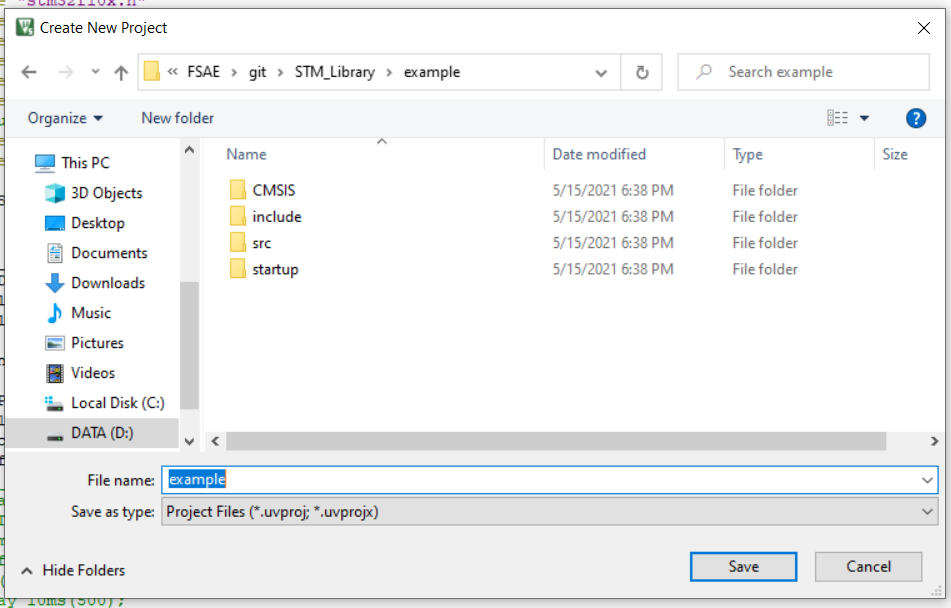


File structure. Hint: … means all the files in the same directory

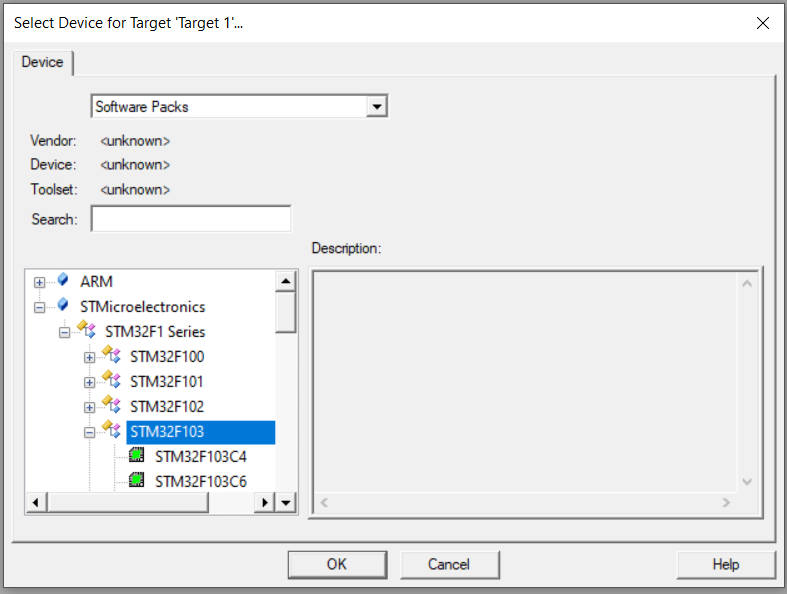
b. create a keil project located in the workspace folder.



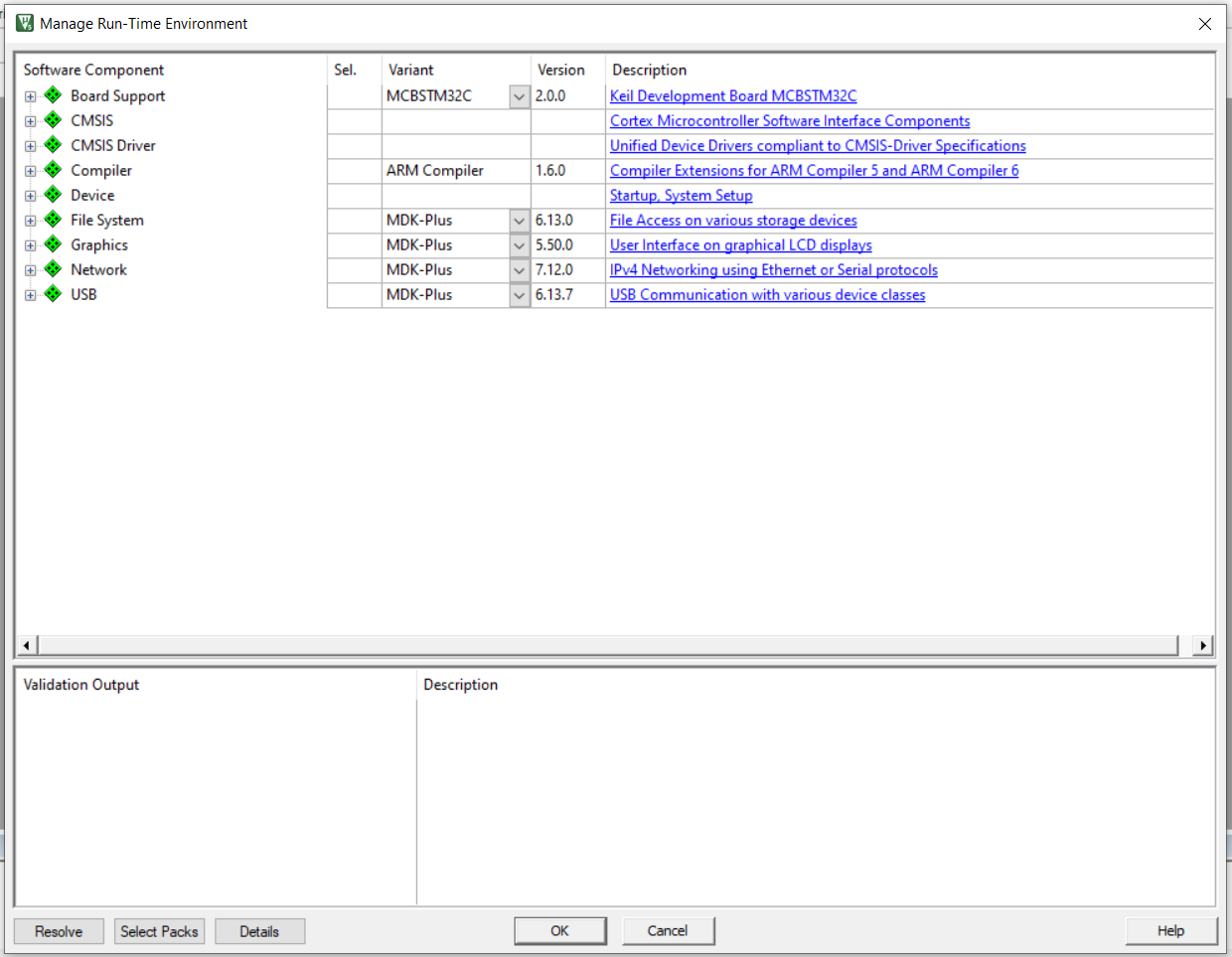
Step 1: Create a project



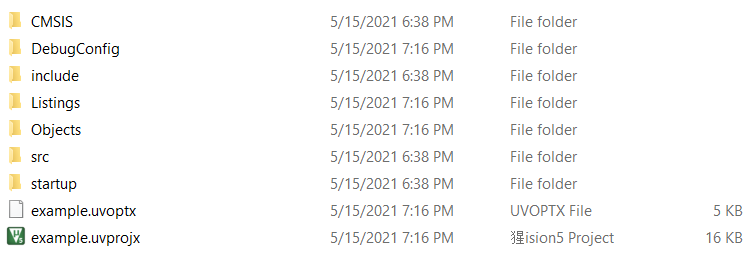
Step 2: Name and locate your project



Step 3: choose a STM type that fit your device



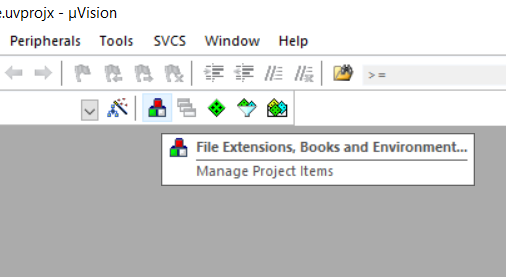
Step 4: Just press OK



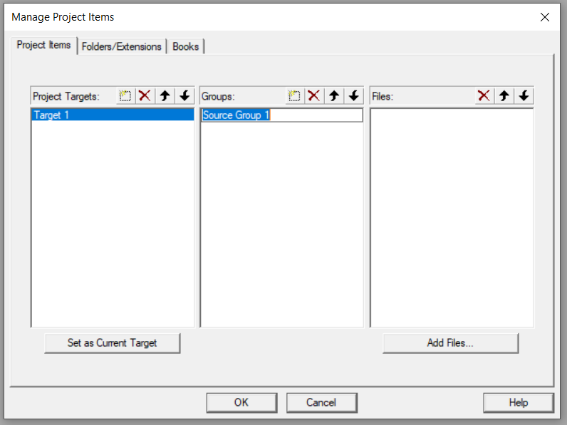
Step 5: Workspace after project creation. Check it out.

**2. Setup your environment:**

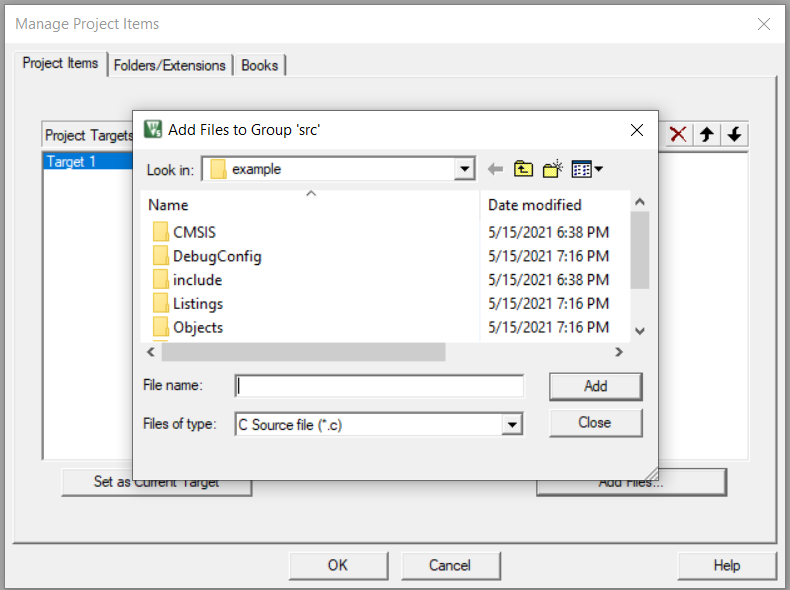
a. Configure source groups: Source groups are where Keil finds source files(.c/.s)



Step 1: Let’s tell Keil where should it find sources to compile

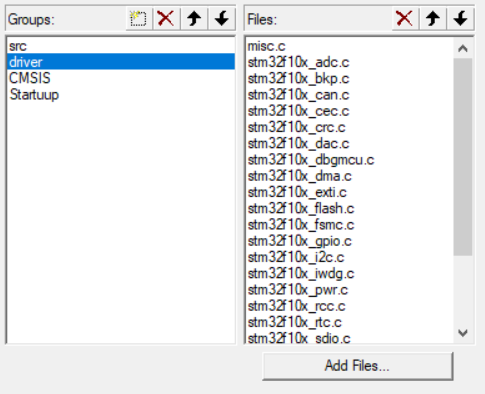
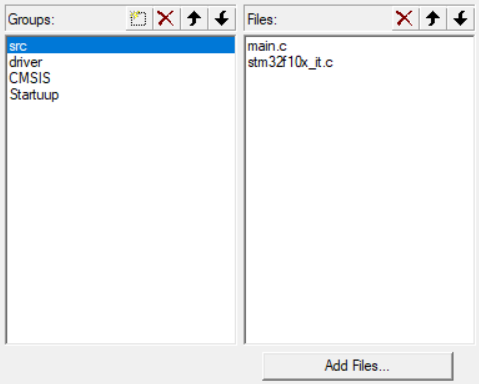


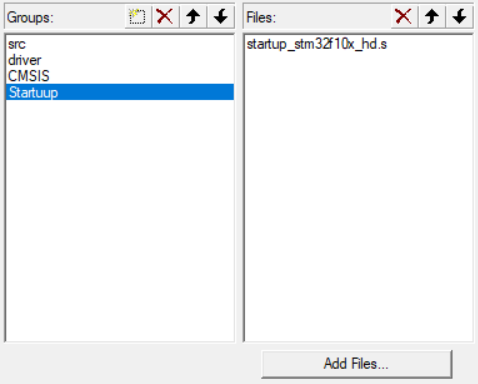
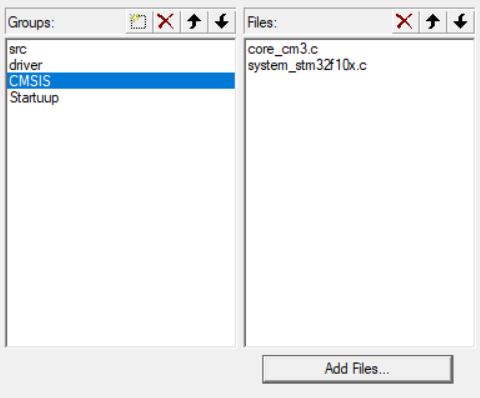
Step 2: Rename source group by double click



Step 3: Click “Add Files…”, then add source files (.c/.s) to corresponding groups.

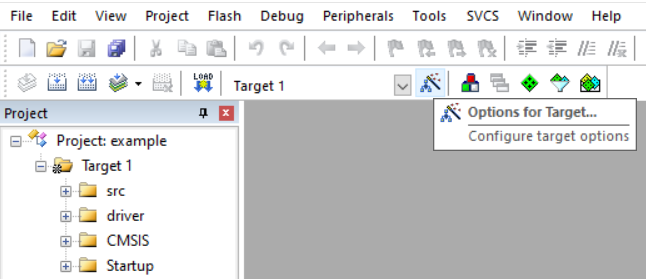
Hint: .s is assembly files



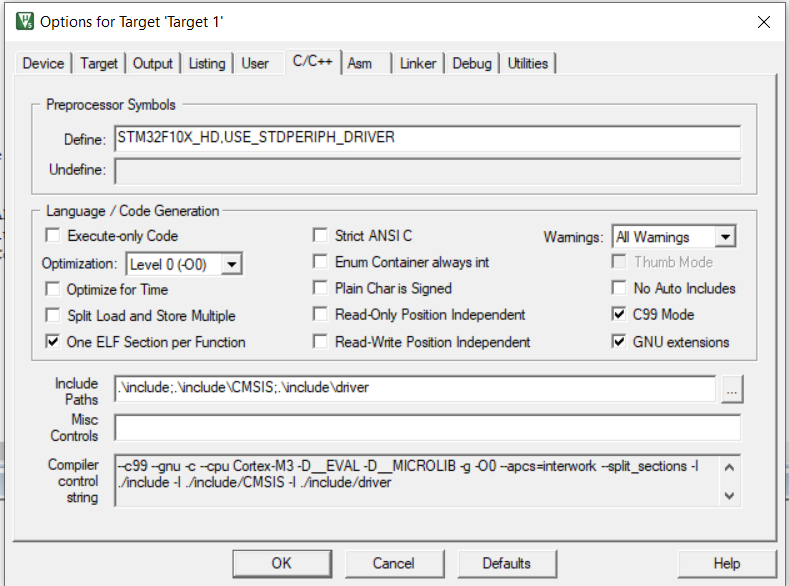


Step 4: Check the source groups are added as above

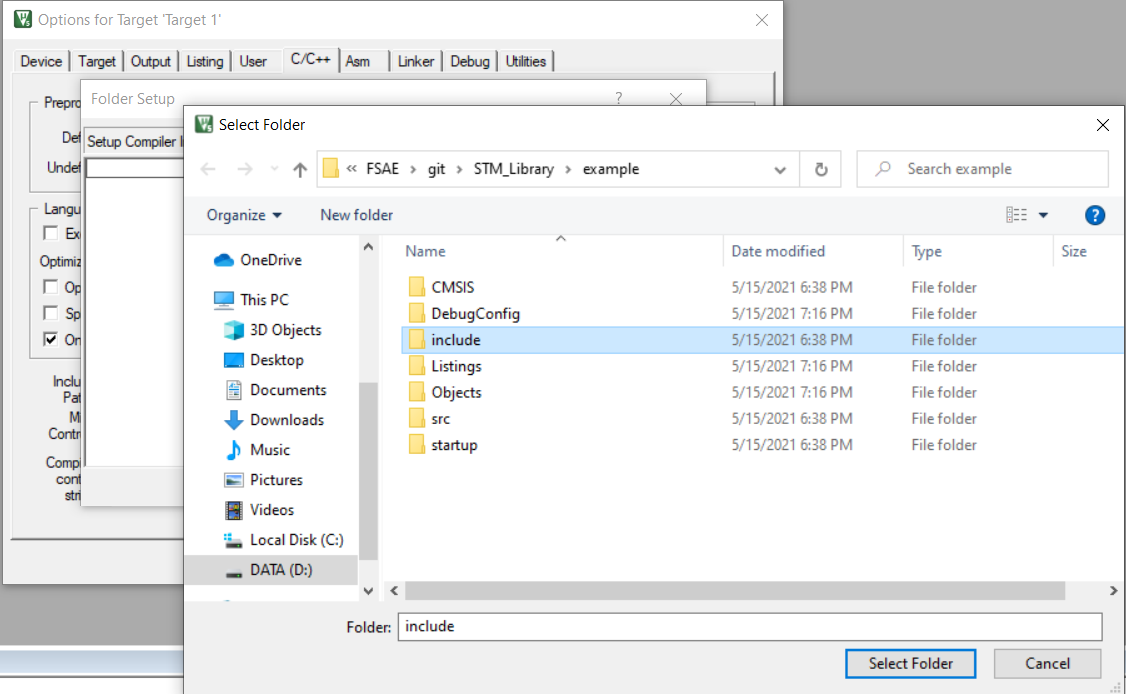
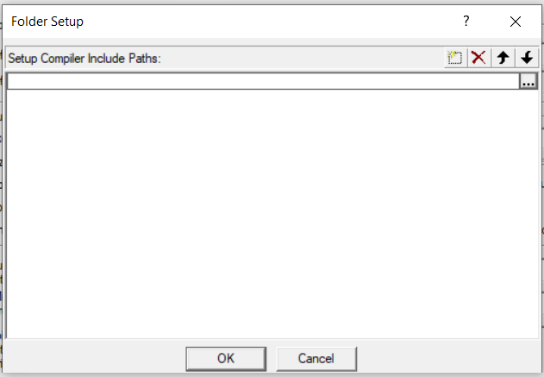
b. configure “options for targets”



Step 1: Click “Options for target”



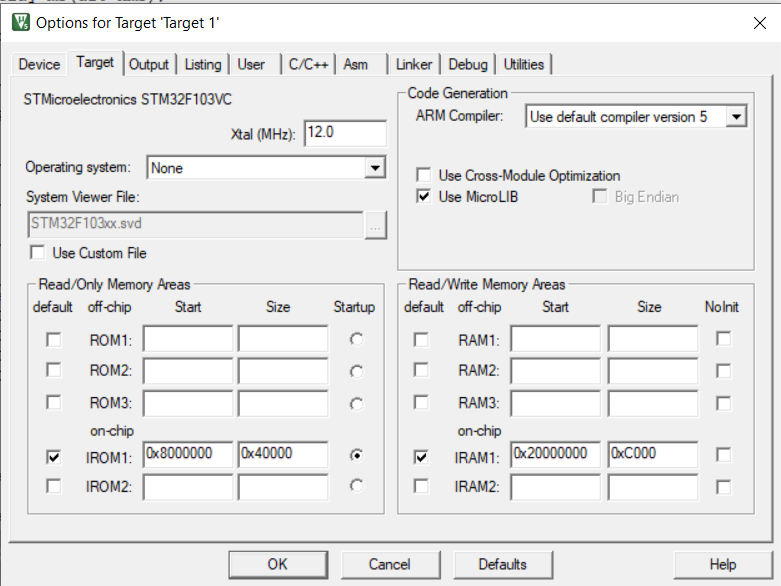
Step 2: Define preprocessor symbols and include paths



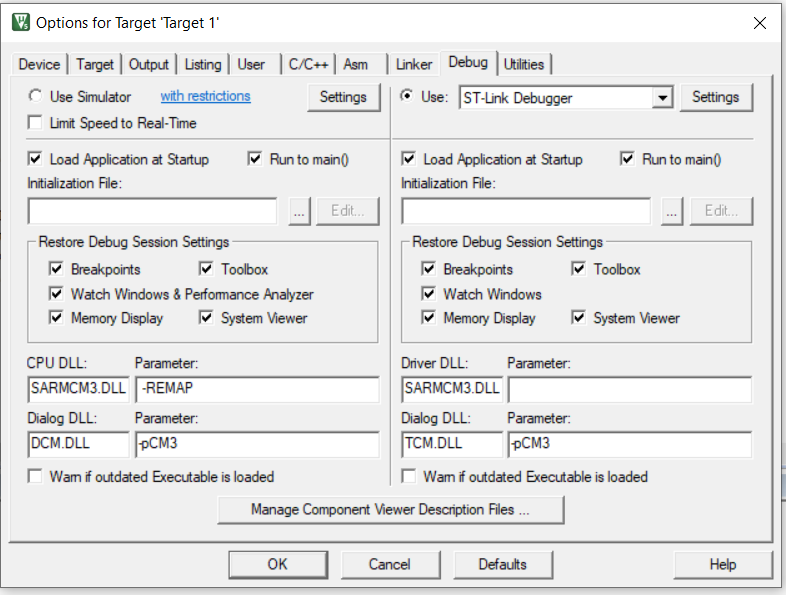
Step 2b: in this case, we should assign “include”, “include/driver” and “include/CMSIS”



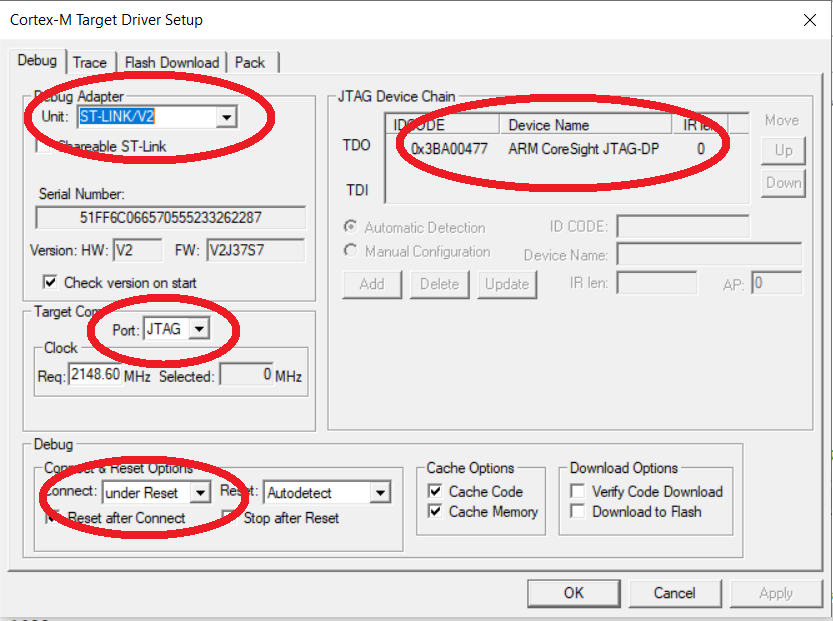
Step 2c: Should end up like that



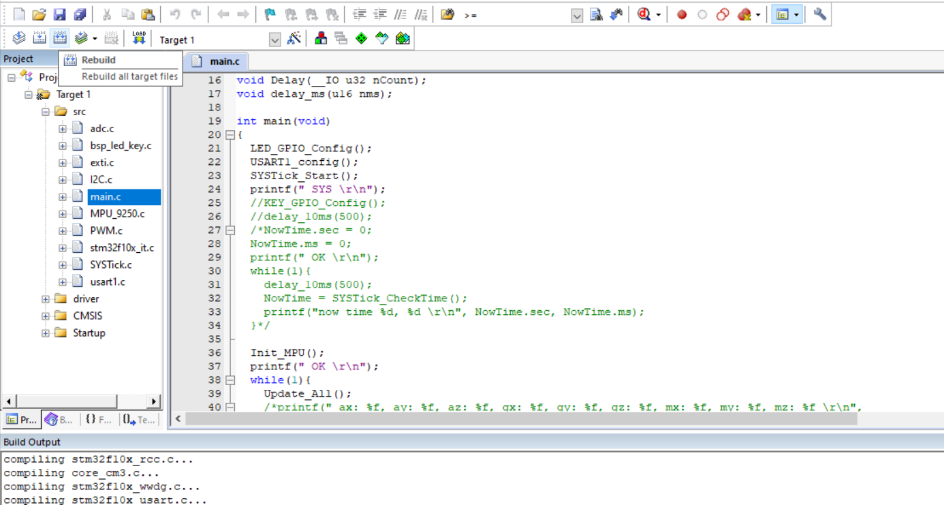
Step 3: choose MicroLib (so as to use printf)



Step 4: Choose the right debugger (ST-Link in this case)

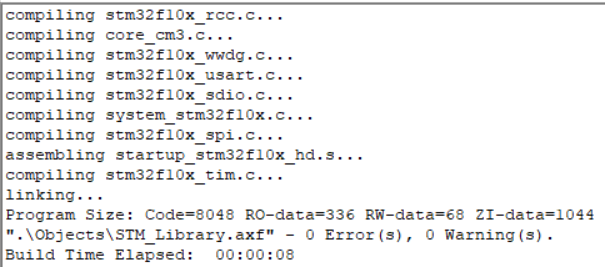


Step 4b: be aware of Port (Down left), Choose Jtag or SW,



Step 5: Try to compile it with some codes

(Example codes can be found below)



Step 6: Make it 0 Error and Warning

c. example: example can be arbitrary, but and is one

main.c:

#include "stm32f10x.h"

void delay\_ms(u16 nms);

int main(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd( RCC\_APB2Periph\_GPIOE, ENABLE);

GPIO\_InitStructure.GPIO\_Pin=GPIO\_Pin\_5;

GPIO\_InitStructure.GPIO\_Mode=GPIO\_Mode\_Out\_PP;

GPIO\_InitStructure.GPIO\_Speed=GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOE, &GPIO\_InitStructure);

while(1){

GPIO\_SetBits(GPIOE,GPIO\_Pin\_5);

delay\_ms(1000);

GPIO\_ResetBits(GPIOE,GPIO\_Pin\_5);

delay\_ms(1000);

}

}

void delay\_ms(u16 nms)

{

u32 temp;

SysTick->LOAD = 9000\*nms;

SysTick->VAL=0X00;//clear the tickle

SysTick->CTRL=0X01;//enable

do

{

temp=SysTick->CTRL;//read the tickle

}while((temp&0x01)&&(!(temp&(1<<16))));//wait for the end

SysTick->CTRL=0x00; //close it

SysTick->VAL =0X00; //clear it

}