Setting up STM8L-DISCOVERY board and STVD (IDE)

***Note: STM8L-DISCOVERY's host processor model is the: STM8L152C6TC**

***Recommendations: Get a good text editor for opening .c and .h files such as Sublime Text 3**

***Product Page of STM8L152C6 for Datasheet**

https://www.st.com/en/microcontrollers-microprocessors/stm8l152c6.html

***Datasheet for STM8L_DISCOVERY**

https://www.st.com/content/ccc/resource/technical/document/user_manual/ca/6a/61/13/08/c4/48/f9/CD00278045.pdf/files/CD00278045.pdf/jcr:content/translations/en.CD00278045.pdf

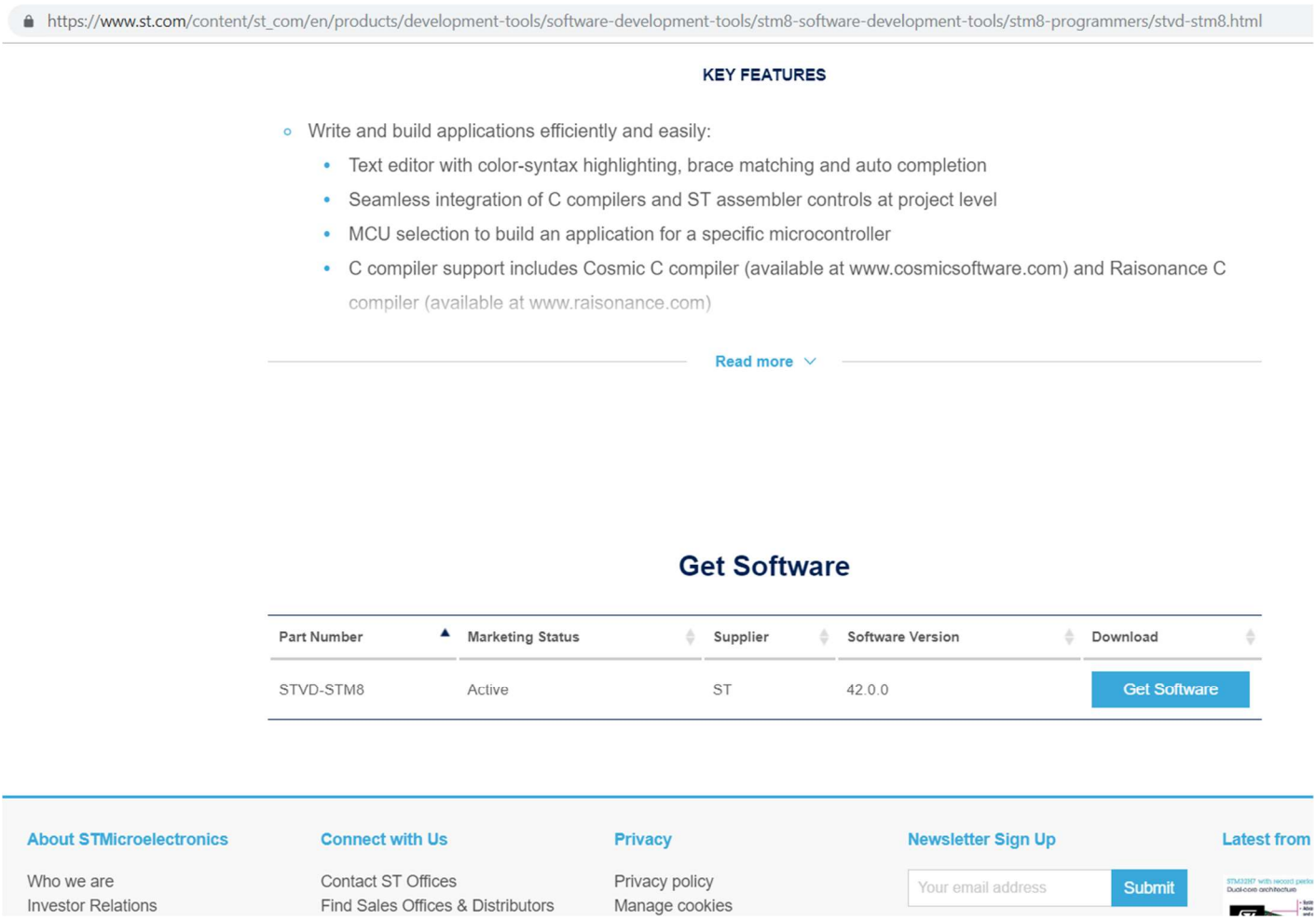
1.) Download the STVD IDE from the website

https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm8-software-development-tools/stm8-programmers/stvd-stm8.html

File Name: STVD-STM8

Latest Version: 6/7/2019 - v42.0.0

Press “Get Software” at the top of the page or scroll down and press “Get Software” again near the file name to download. May need an account for download.



2.) Accept the licensing terms of agreement and login/register an account. (account makes it easier for later downloads). Download should begin after logging in

3.) Extract the downloaded file into a folder. Then run application or '.exe' file.

Allow program to make changes to computer -> 'Yes'

Press next->Accept terms of agreement->next->next->(Deselect release notes)next->'Yes, I want to restart computer now'->Finish

'standard tools' for type

4a.) Once downloaded, there is a tool file called ‘tools.cnf’ that always causes STVD to crash when you try to run the programmer or flash the board. This is a common occurrence, and the solution to fix this is to replace this to an older version of ‘tool.cnf’

You will have to go this file location to locate ‘tools.cnf’ (Assuming you didn’t change the path location when installed STVD.

C:\Program Files (x86)\STMicroelectronics\st_toolset\stvp

4b.) Once you’re inside the ‘stvp’ folder, you’d want to scroll all the down and find ‘tools.cnf’

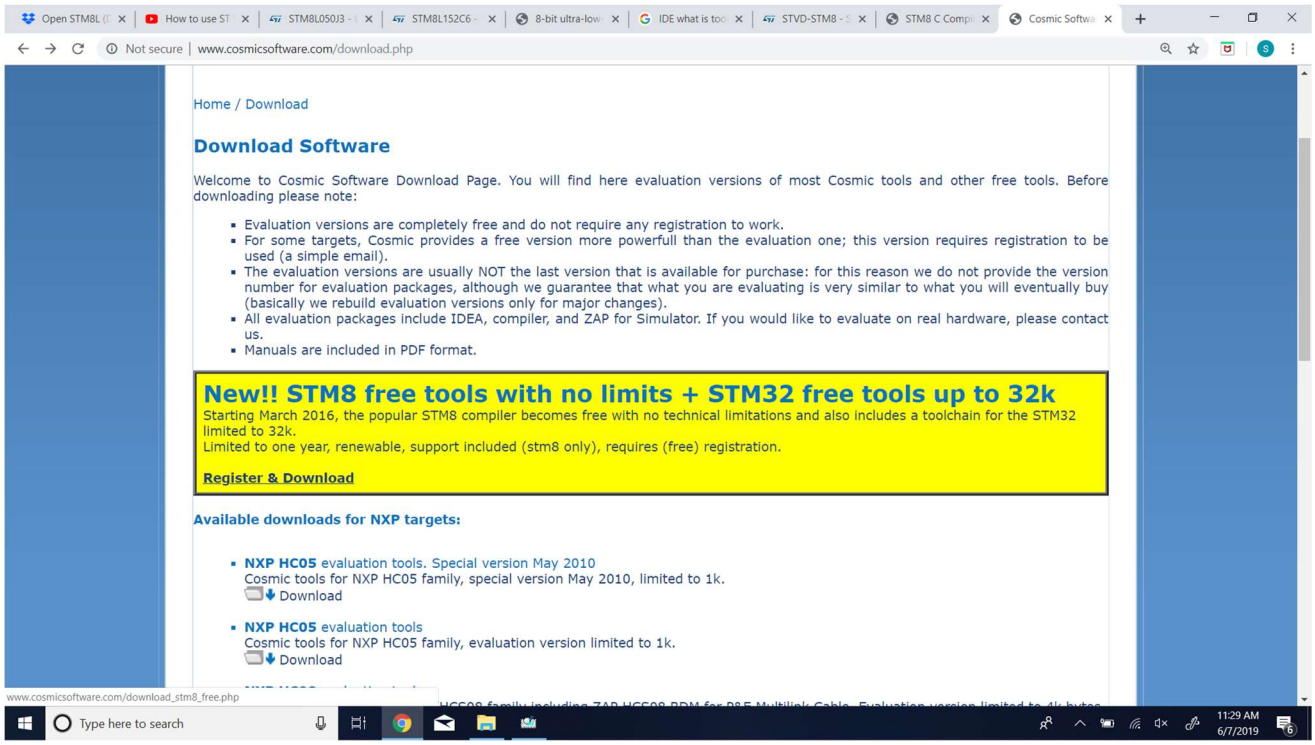
4c.) You can delete this file but I recommend moving ‘tools.cnf’ into a folder for the sake of having it just in case.

4d.) I’ve attached an older (working as of 6/10/2019) version of ‘tools.cnf’ in the same folder as this Document within the team’s DropBox.

Now move the working ‘tools.cnf’ file into the ‘stvp’ folder

5.) Next, download the C compiler supported for STVD. I chose Cosmic C since it was free (doesn’t natively support boolean (true/false) datatypes so you’ll need the **stdbool.h** library or #define true 1 and #define false 0. Many ways to go around it)

http://www.cosmicsoftware.com/download.php



Click ‘Register & Download’ and register an account to receive the free license. You should be able to download after pressing ‘Submit’

6.) Run the downloaded ‘.exe’ file. Window will appear

‘Next’ -> ‘Accept terms of agreement’ -> ‘Next’ -> (Enter User name and company name) ‘Next’

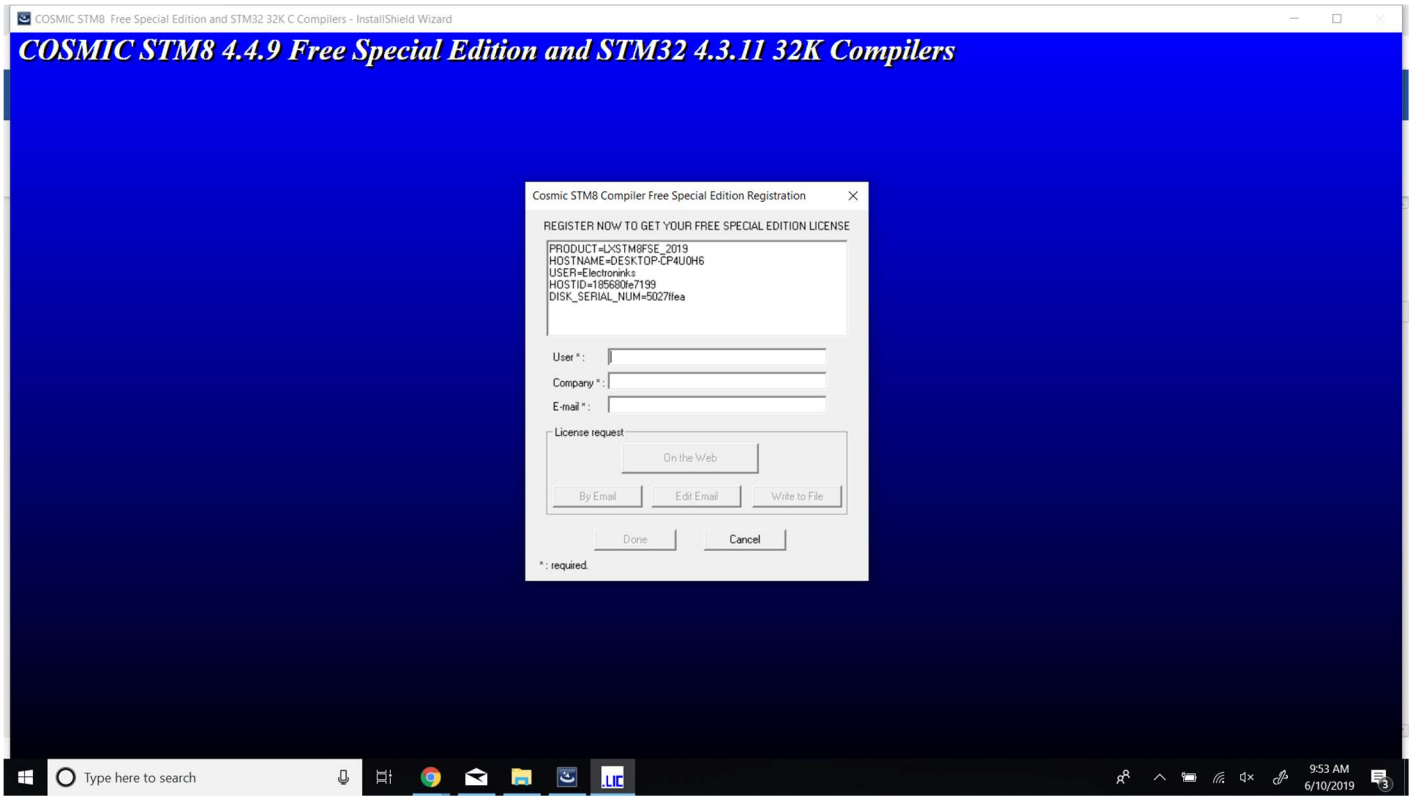
-> ‘Next’ -> ‘Next’ -> ‘Next’ (Not a lot of things to change).

You will then have to select a registry for Cosmic. Choose the ‘Current User’

Register in HKEY_CURRENT_USER

Then press ‘Next’ -> ‘Yes’ -> ‘Yes’ -> ‘Next’

7.) A notepad is going to open describing about the licensing for Cosmic C. Just close the notepad and a registration program should pop up like this



Enter the user name and company from before and an email.

Click the “On the Web” option and verify you’re not a robot. Wait to receive an email. (Check junk/spam folders)

8.) There should be a license.lic file attached to the email. Save the file into compiler location

i.e.

…\Program Files (x86)\COSMIC\FSE_Compilers\CXSTM8\License

9.) Next, download the STM8L Standard Peripheral Library (SPL) for your specific STM8 processor. In this case, we will need the SPL for the STM8L15x (Derived from **STM8L15**2C6T6)

https://www.st.com/content/st_com/en/products/embedded-software/mcu-mpu-embedded-software/stm8-embedded-software/stsw-stm8016.html#overview

File Name: STSW-STM8016

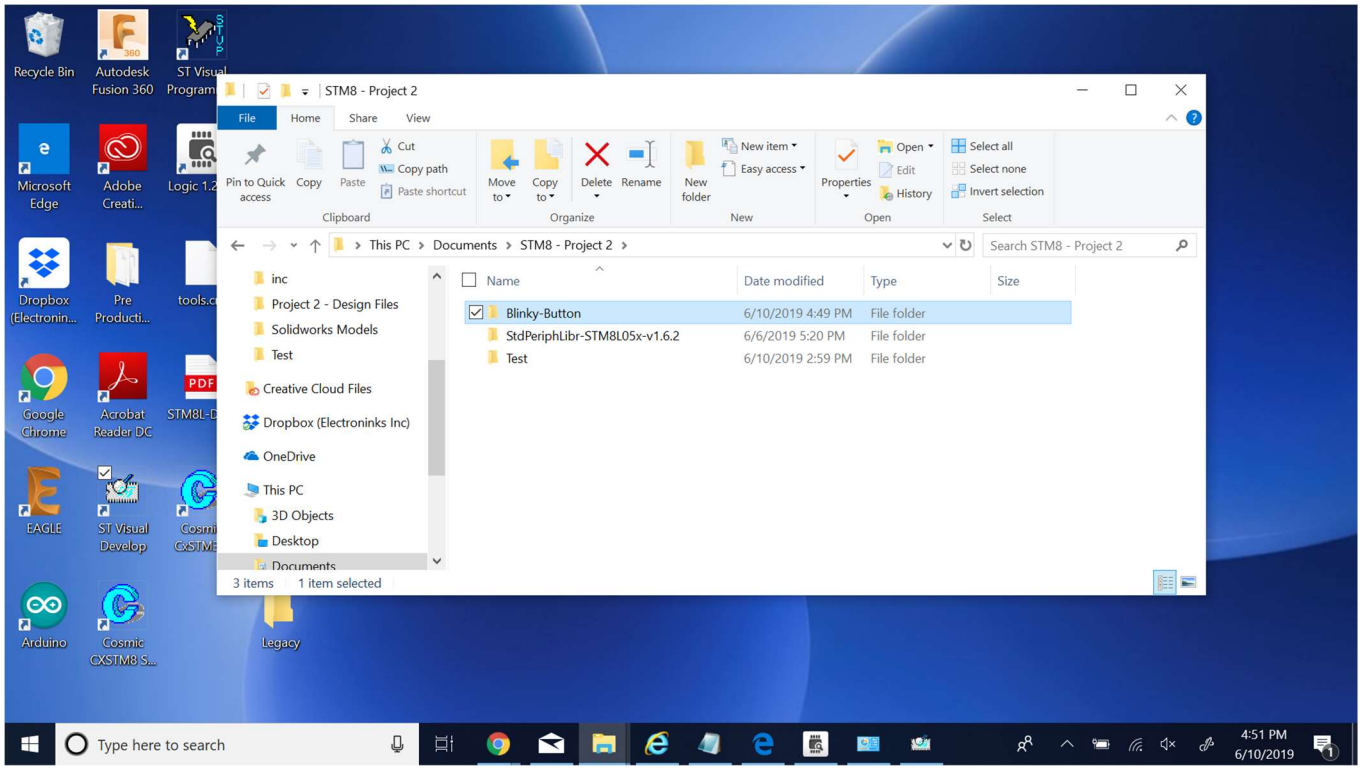
Latest Version: 6/7/2019 - v1.6.2

Repeat the download process.

i.e. Press “Get Software” at the top left of the page and then press “Get Software” again near the file name to download.

10.) Create a directory to store all your projects i.e. name it ‘STM8 Projects’ Extract the contents of the downloaded SPL .zip file into 'STM8 Projects’.

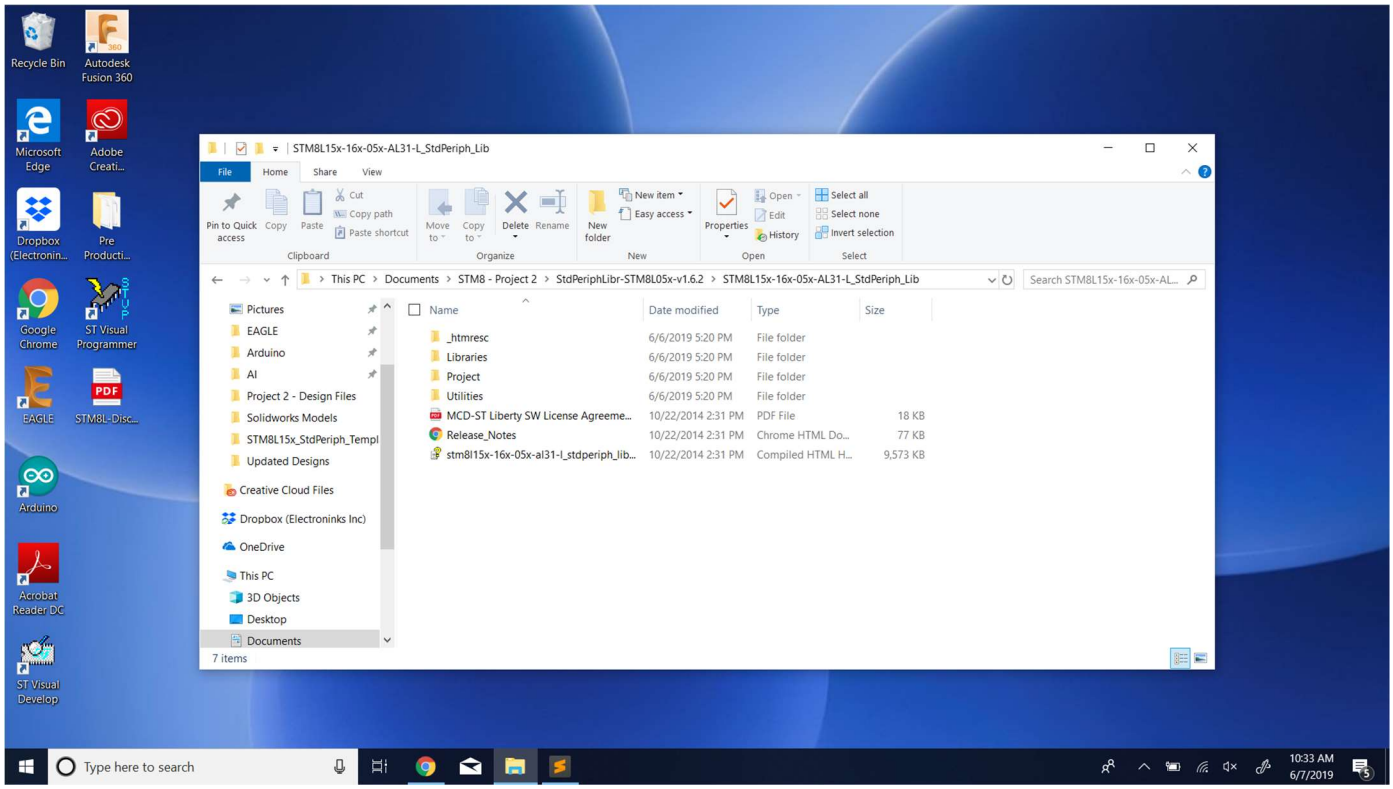
11.) Create a new directory with your project name. I’ll be using ‘Blinky-Button’. Don’t use any spaces for your project names. Use either ‘-’ or ‘_’ for the delimiters as this can sometimes cause accessing a file difficult.



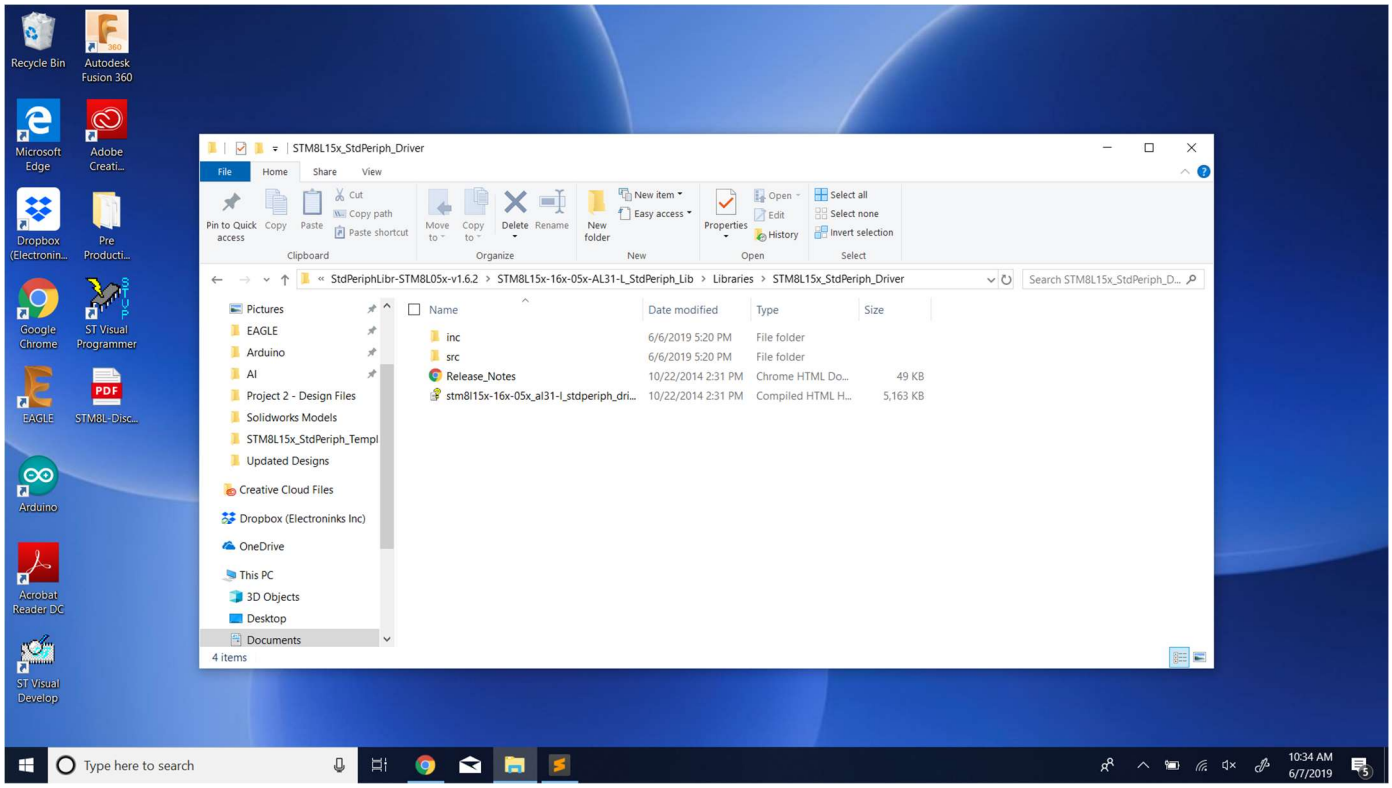
Should look something like this now ^^

Don’t panic if StdPeriphLibr-_____-v_____ doesn’t match. The library is compatible with multiple devices, including the STM8L15x

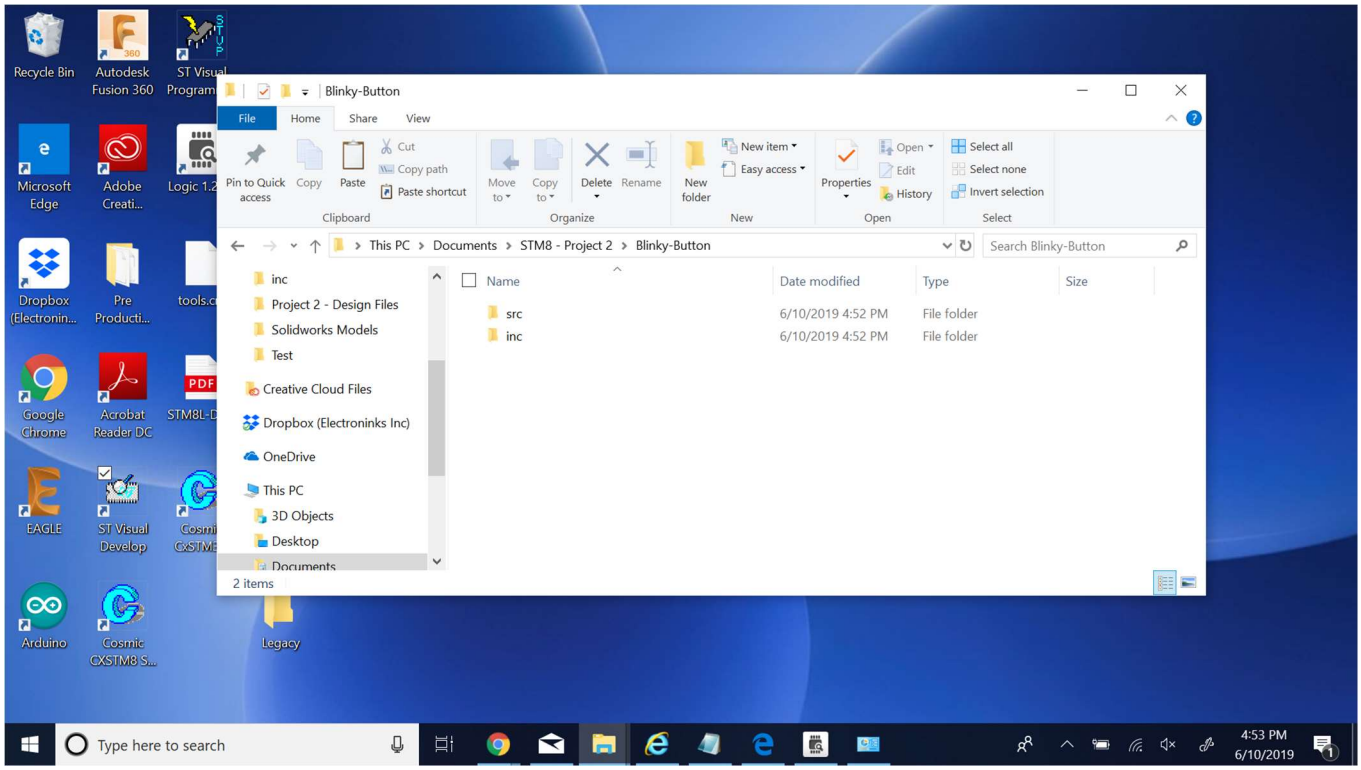
12.) Go into the StdPeriphLibr-_____-v____ folder until you reach something like this



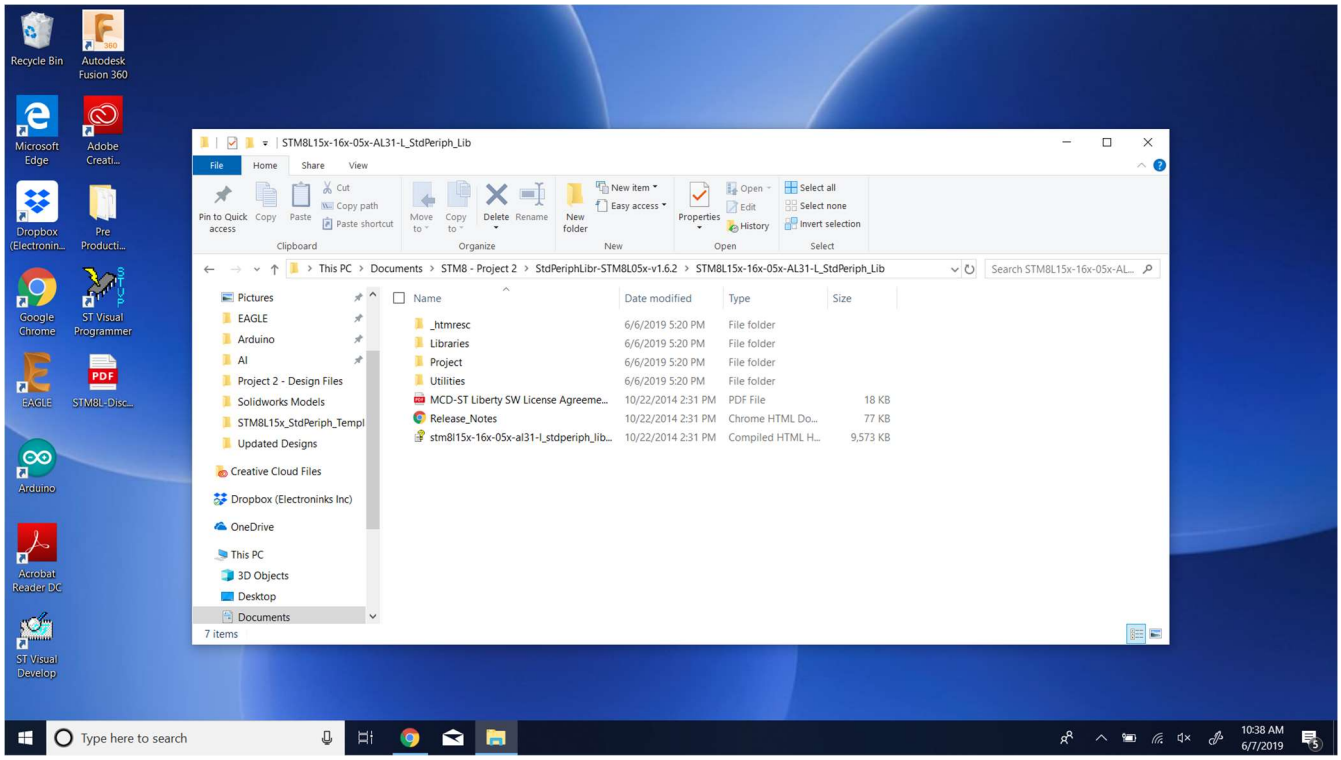
13.) Go into the ‘Libraries’ folder until you come across the ‘inc’ and ‘src’ folders.



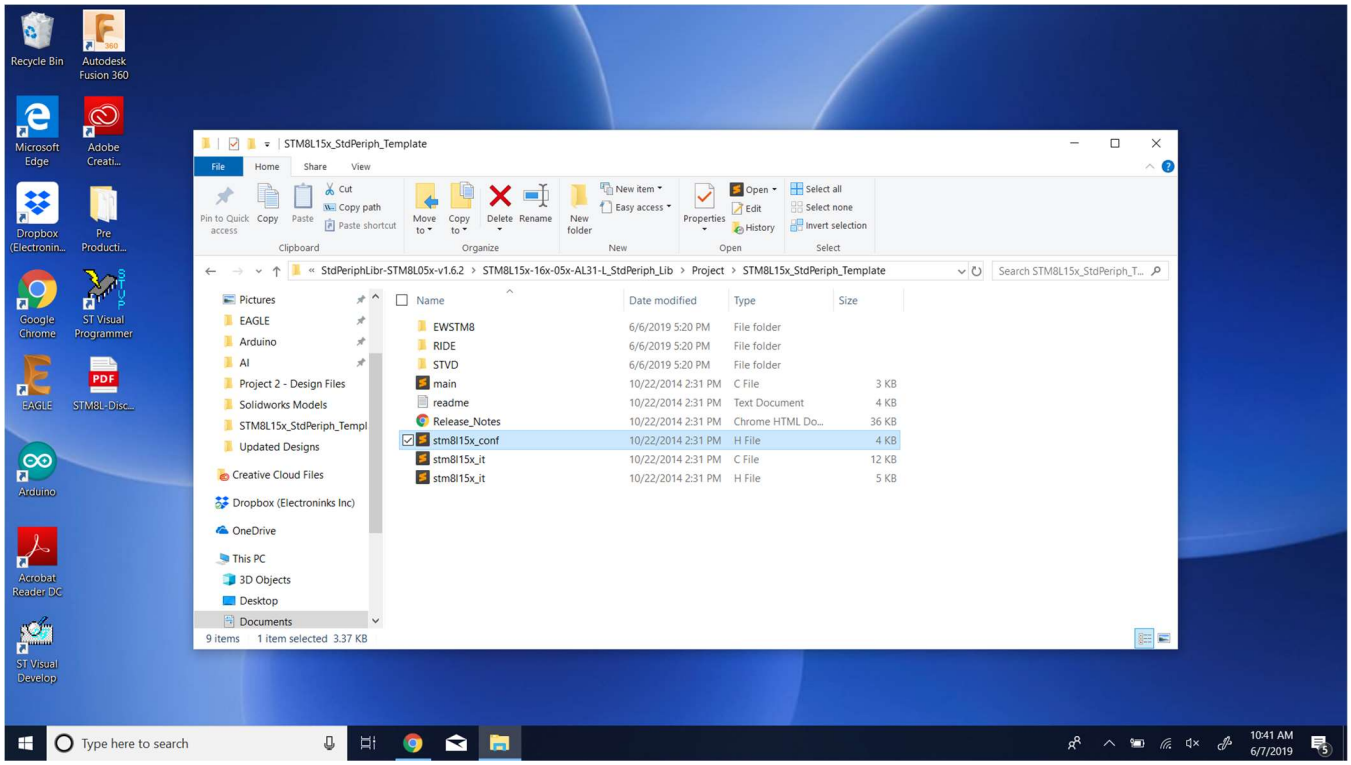
14.) Copy and Paste the ‘inc’ and ‘src’ folder into your specified project folder. I placed it into my project folder named ‘Blinky-Button’ that I made earlier



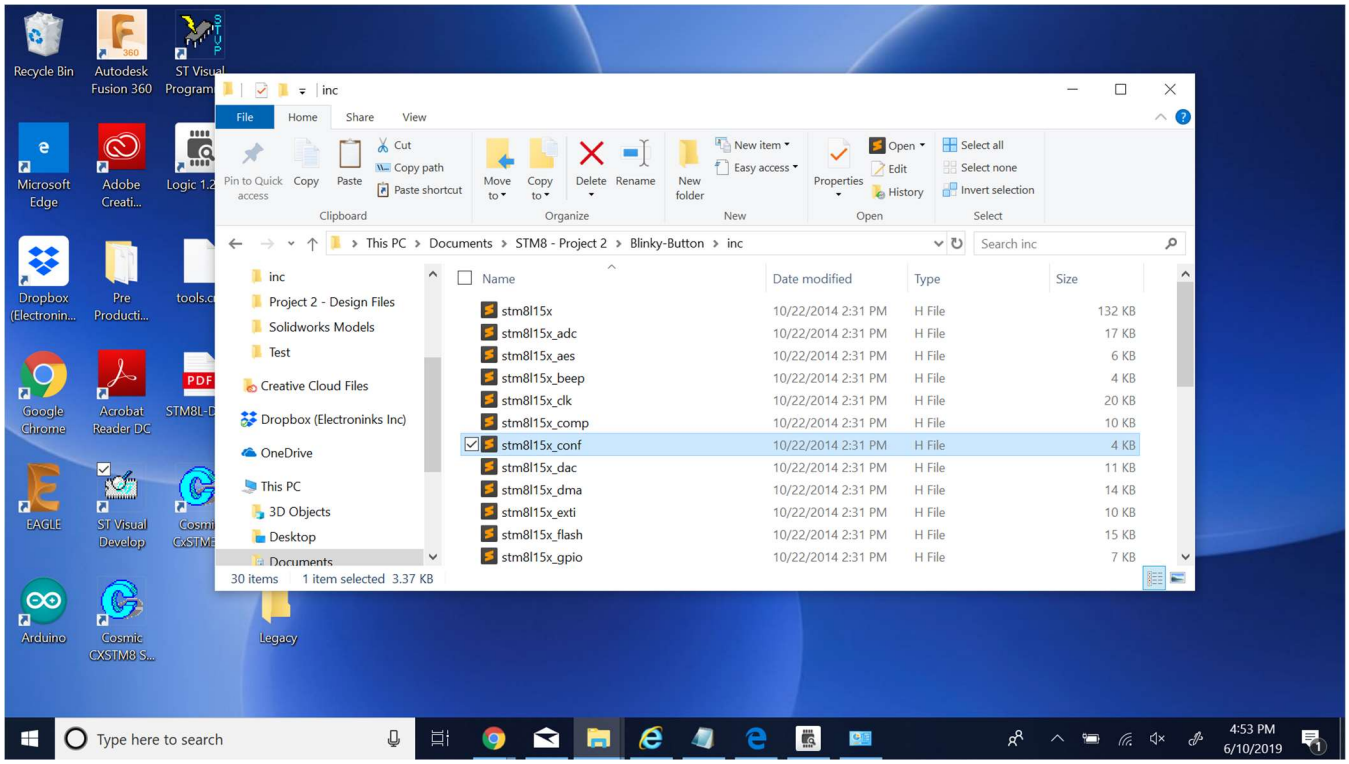
15.) Now go back into the StdPeriphLibr-_____-v____ folder and until the Libraries and Projects folder



16.) Now go into 'Project'->'STM8L15_StdPeriph_**Template**' and copy the **stm8l15x_conf.h** file



17.) Paste it into your project folder i.e. ‘Blinky-Button’->’inc’

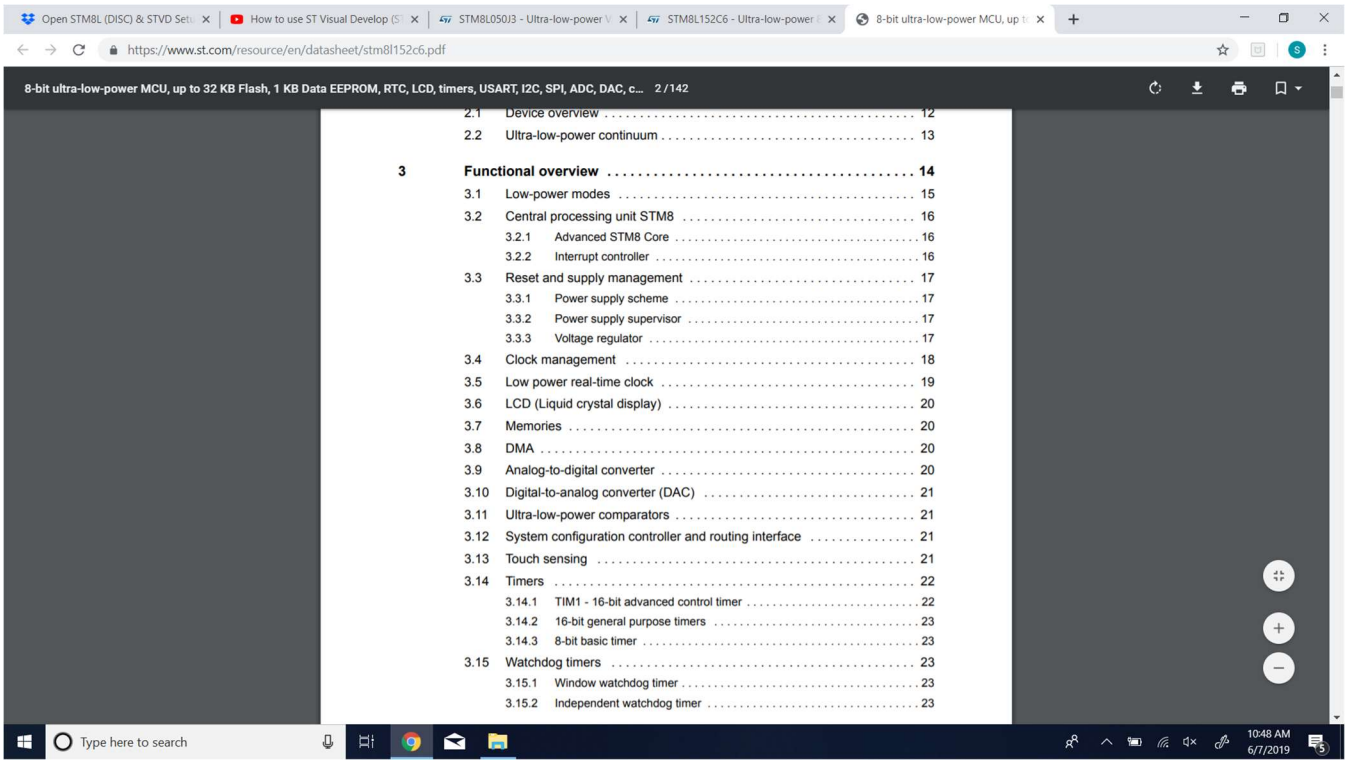


Extra Steps you don’t really need to do if you don’t want to. Delete all unnecessary peripheral library files that your processor doesn’t feature (like more than 1 UART line)

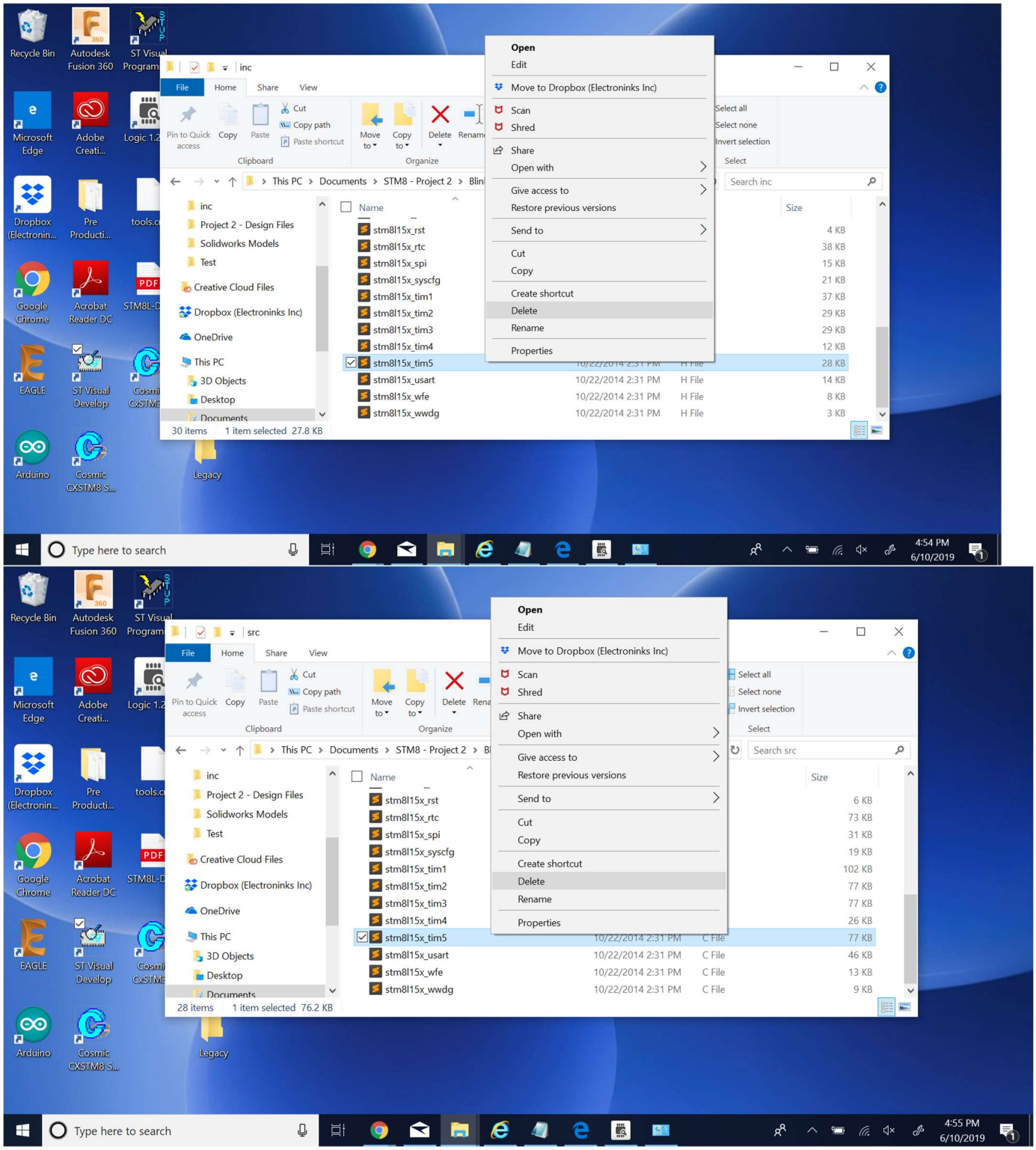
18.1) Now go to the datasheet of your device-processor. DISCOVERY uses STM8L152C6

https://www.st.com/en/microcontrollers-microprocessors/stm8l152c6.html

Scroll down into the Table of Contents and find the ‘Functional Overview’ or the ‘Product Overview’ section

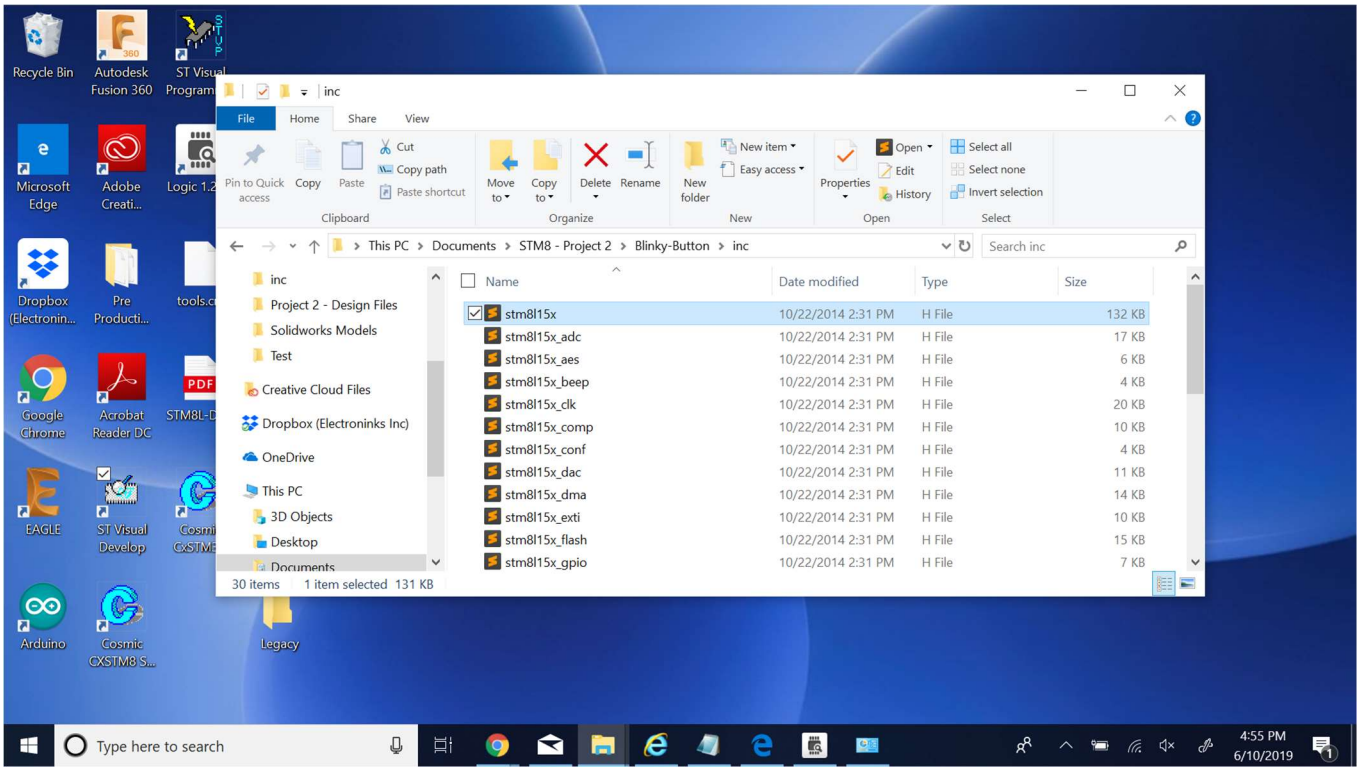


18.2) You can delete all the library files from your ‘inc’ and ‘src’ folder that your processor doesn’t have. For example, the STM8L152C6 only has 4 timers (TIM1, TIM2, TIM3, TIM4). So you can delete the **stm8l15x_tim5.h** and the **stm8l15x_tim5.c** files from ‘inc’ and ‘src’

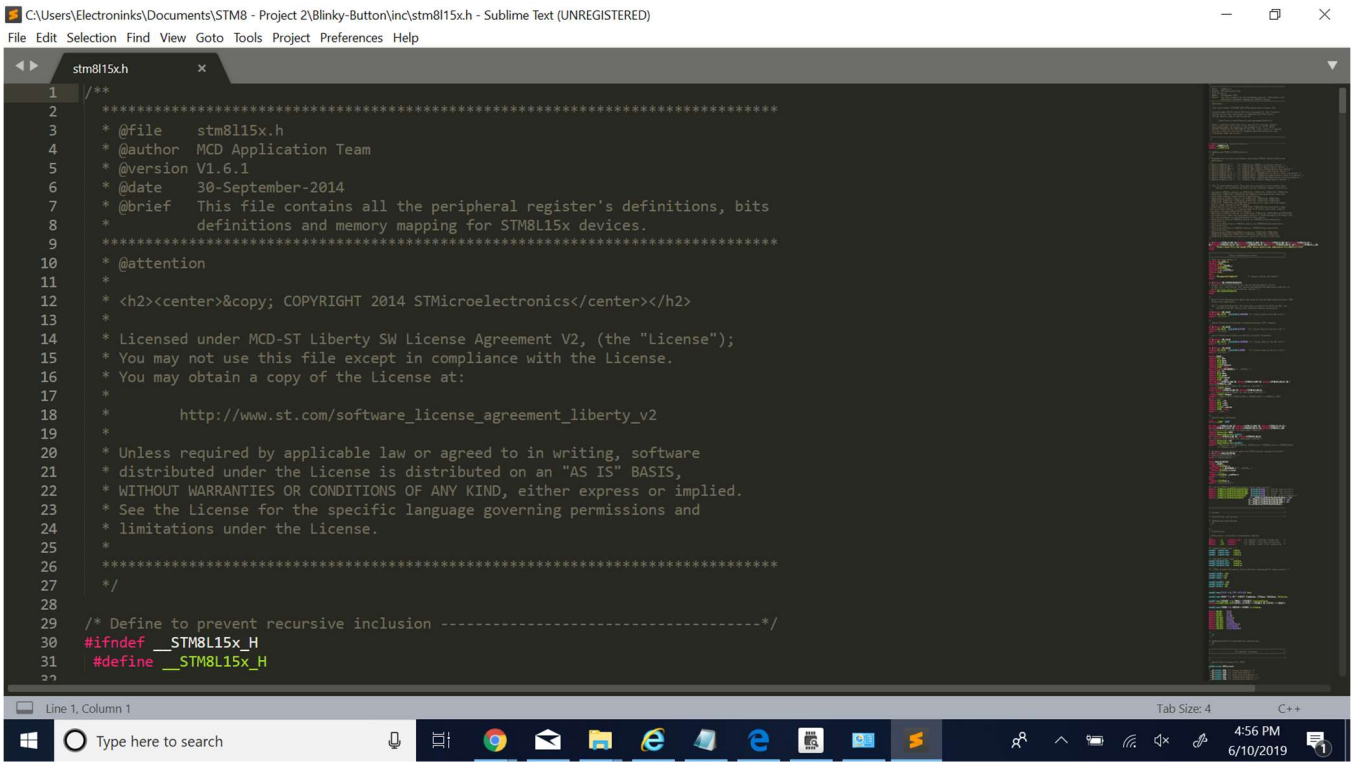


19.) Now, we need to define your device so the library knows what you're using.

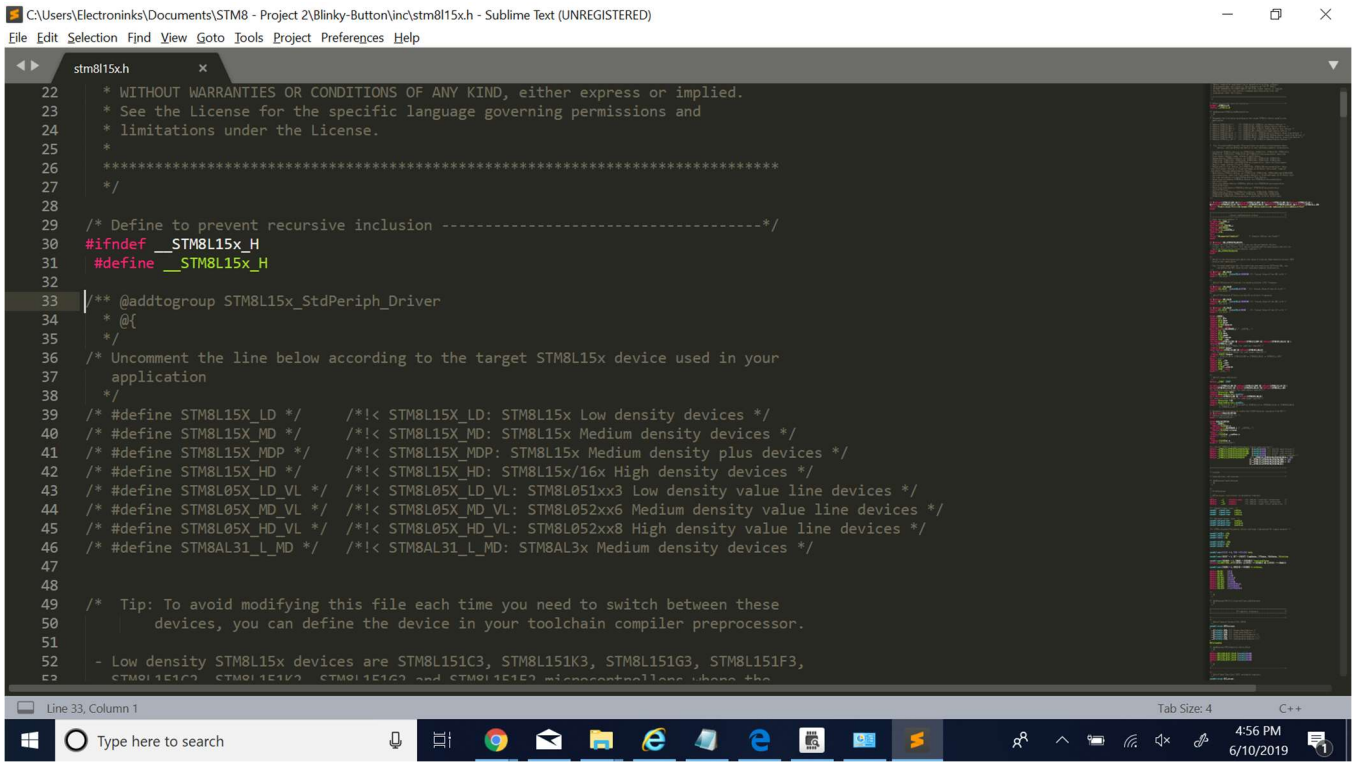
Go into your project folder ‘Blinky-Button’->’inc’ and open the library header file with no peripheral name attached, **stm8l15x.h**



Once opened, it should look something like this



20.) Scroll down a bit where there's a list of target devices. For my file, it's lines 33-47



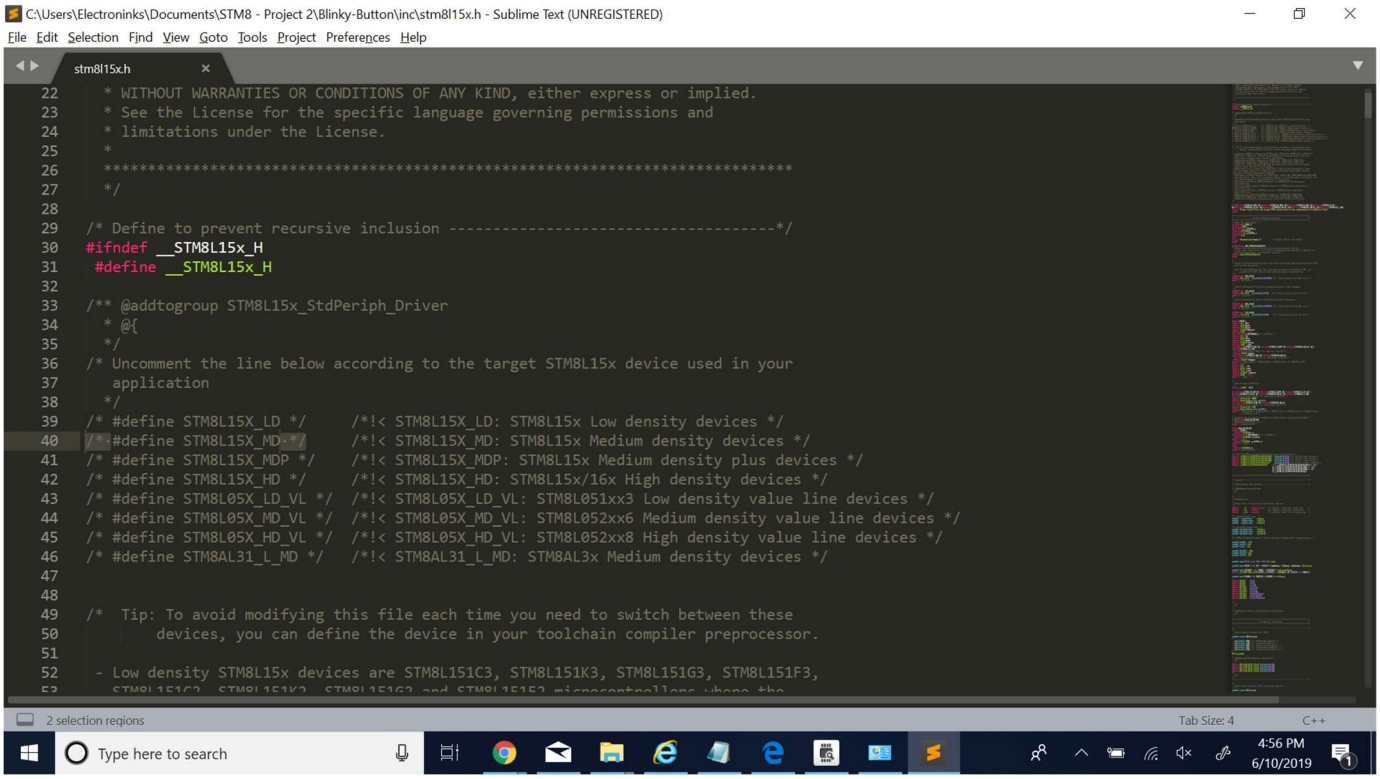
21.) Uncomment your target device that you will be using. For the STM8L152C6, it is a medium density device, so you would want to uncomment

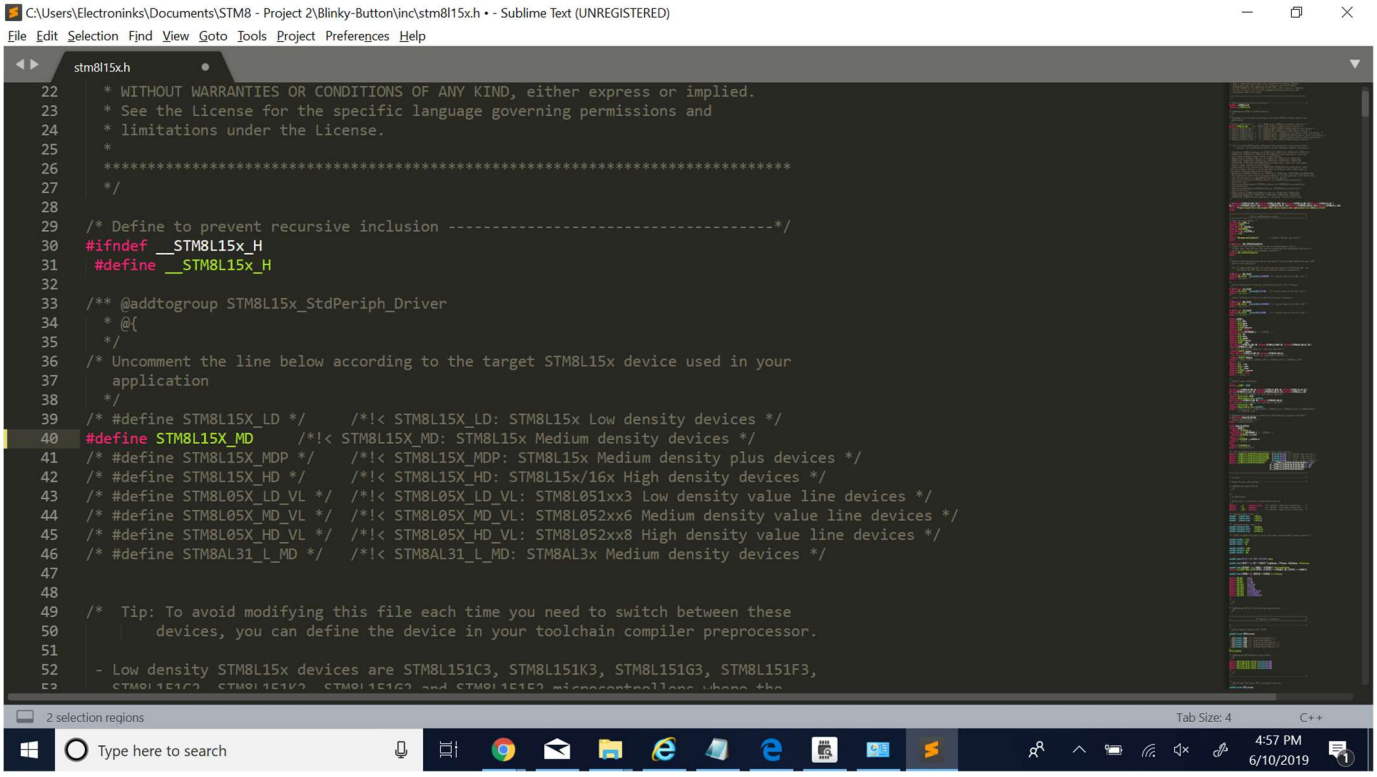
#define STM8L15X_MD

**NOTE: For the STM8L050J3, we will use the STM8L051X_LD_VL

#define STM8L051X_LD_VL

The STM8L051x3 can be used if there is no STM8L050J3 since they are both essentially the same processor. The only difference is the STM8L050J3 has 8 pins and the STM8L051x3 has 20 pins

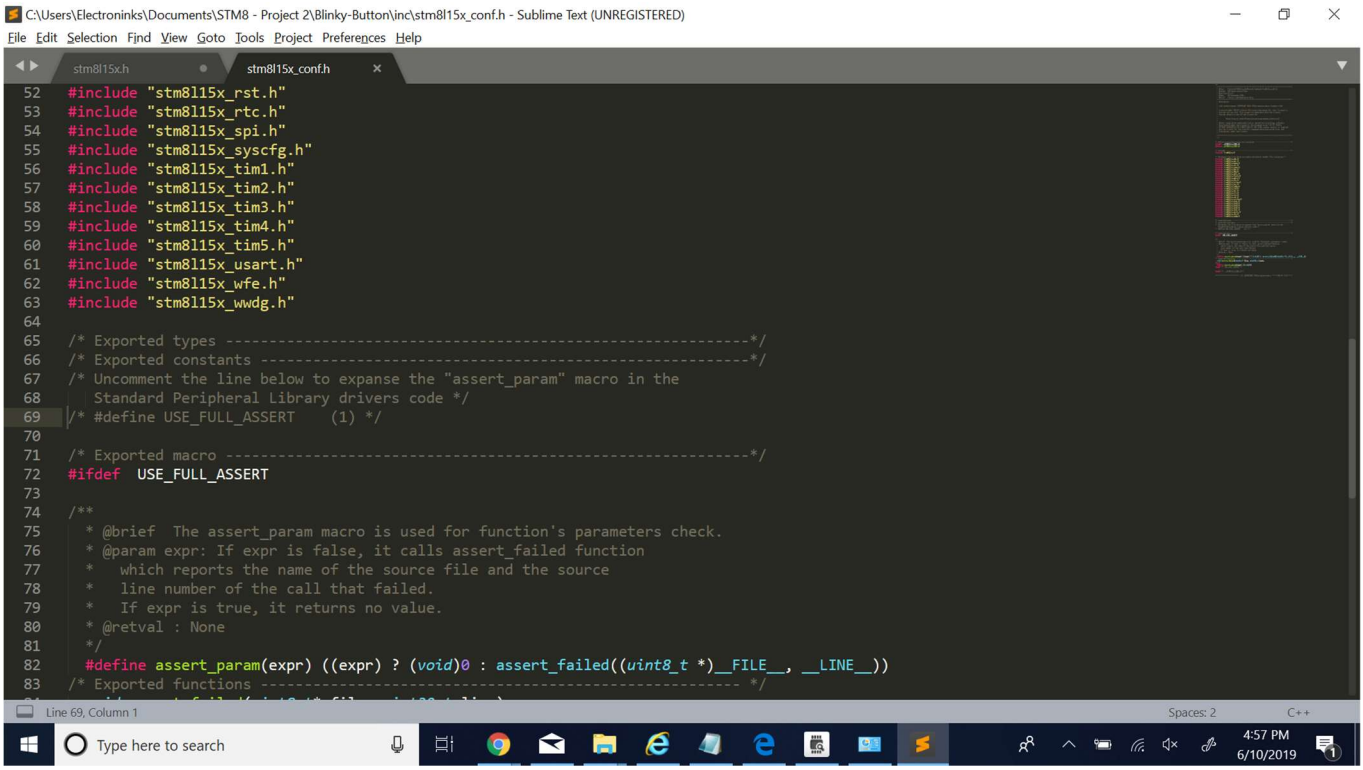




Save the file when done.

22.) Now open the **stm8l15x_conf.h** that you placed in the 'inc' folder from earlier

Make sure /* #define USE_FULL_ASSERT (1) */ is commented out at line 69



23.) Right click the ST Visual Development (STVD) IDE icon (i.e. in your 'Desktop') and 'Run as Administrator'.

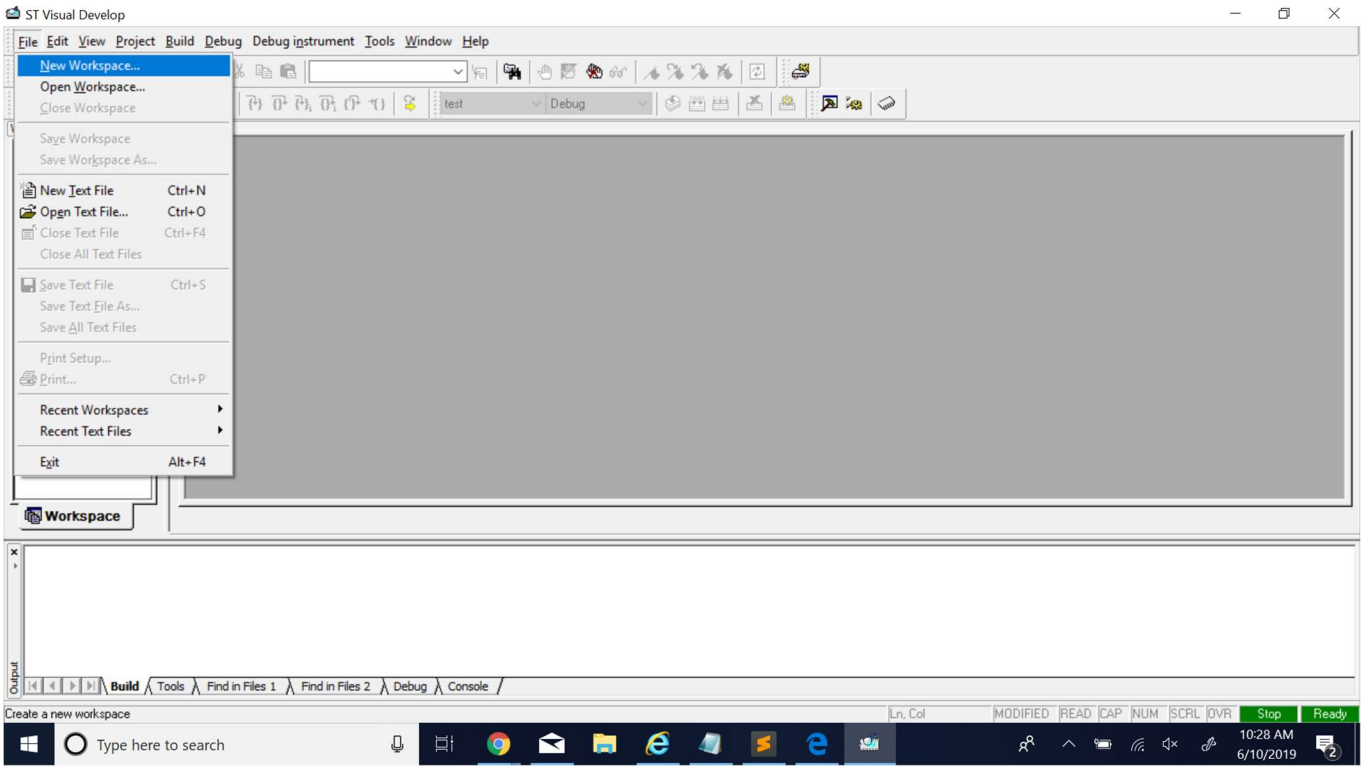
**** Important

Make sure you always open/run as administrator or you won't be able to flash your board. An important file access gets denied, preventing you from flashing the board sometimes, so it's best to always open/run as administrator.

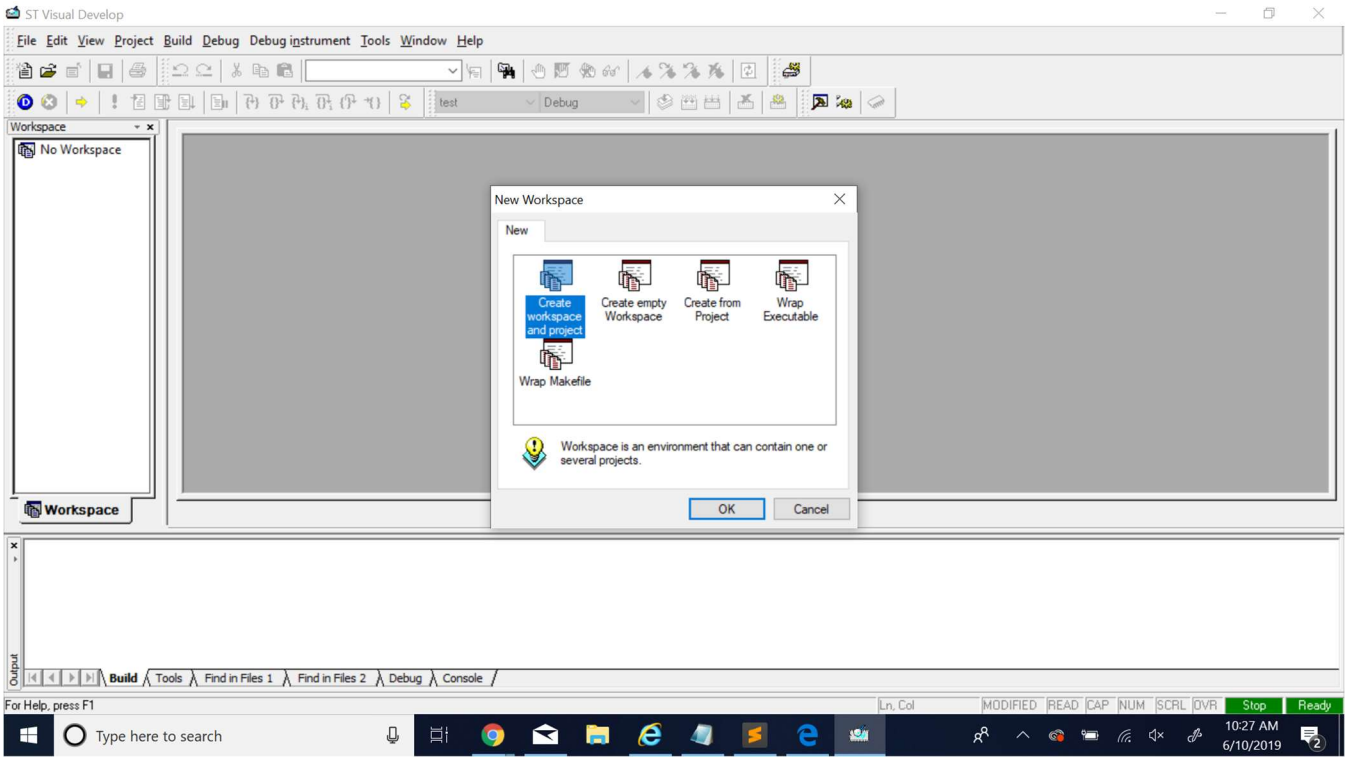
****

Then create new workspace

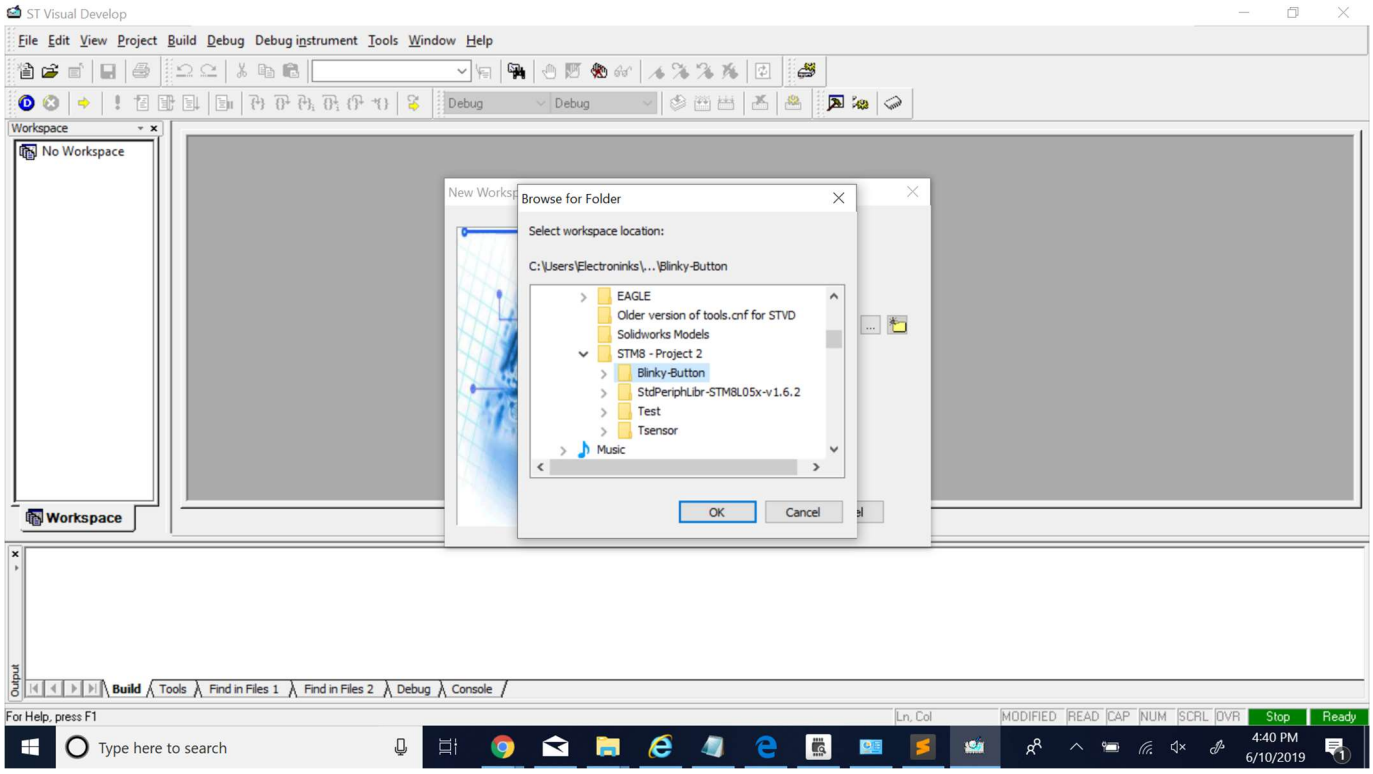
'File' -> 'New Workspace'



-> ‘Create workspace and project’ -> ‘OK’



24.) Name the ‘Workspace filename’, mine will be ‘blinky-button’. Click the ‘...’ under ‘Workspace location’ and select your project folder location (I.e. …\STM8 - Projects\Blinky-Button). Then press ‘OK’



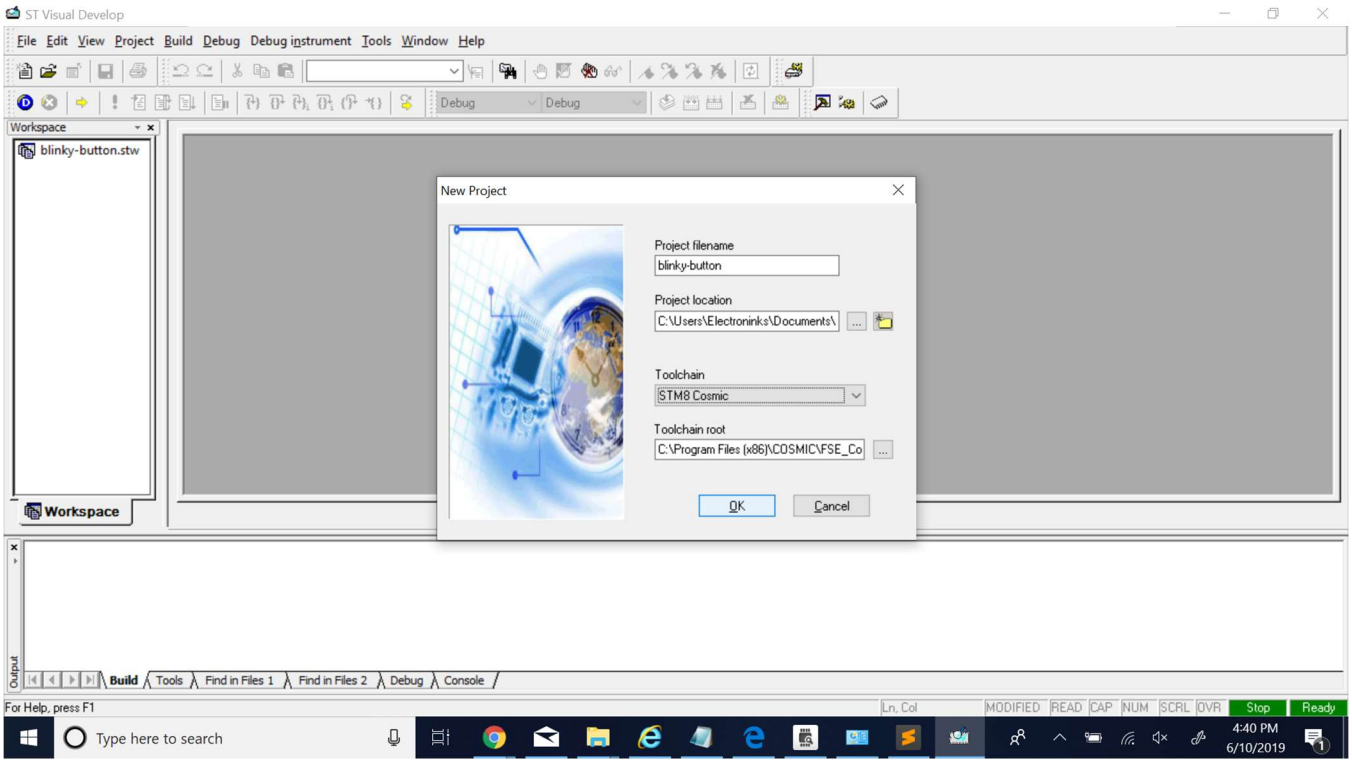
25.) You’ll see a ‘New Project’ popup window. Name the ‘Project filename’, mine will be ‘Blinky-Button’ to make it consistent with the ‘Workspace filename’

Your Project Location should be set for you to your Project Folder Location (I.e. …\STM8 - Projects\Blinky-Button) when you found it in ‘Workspace Location’.

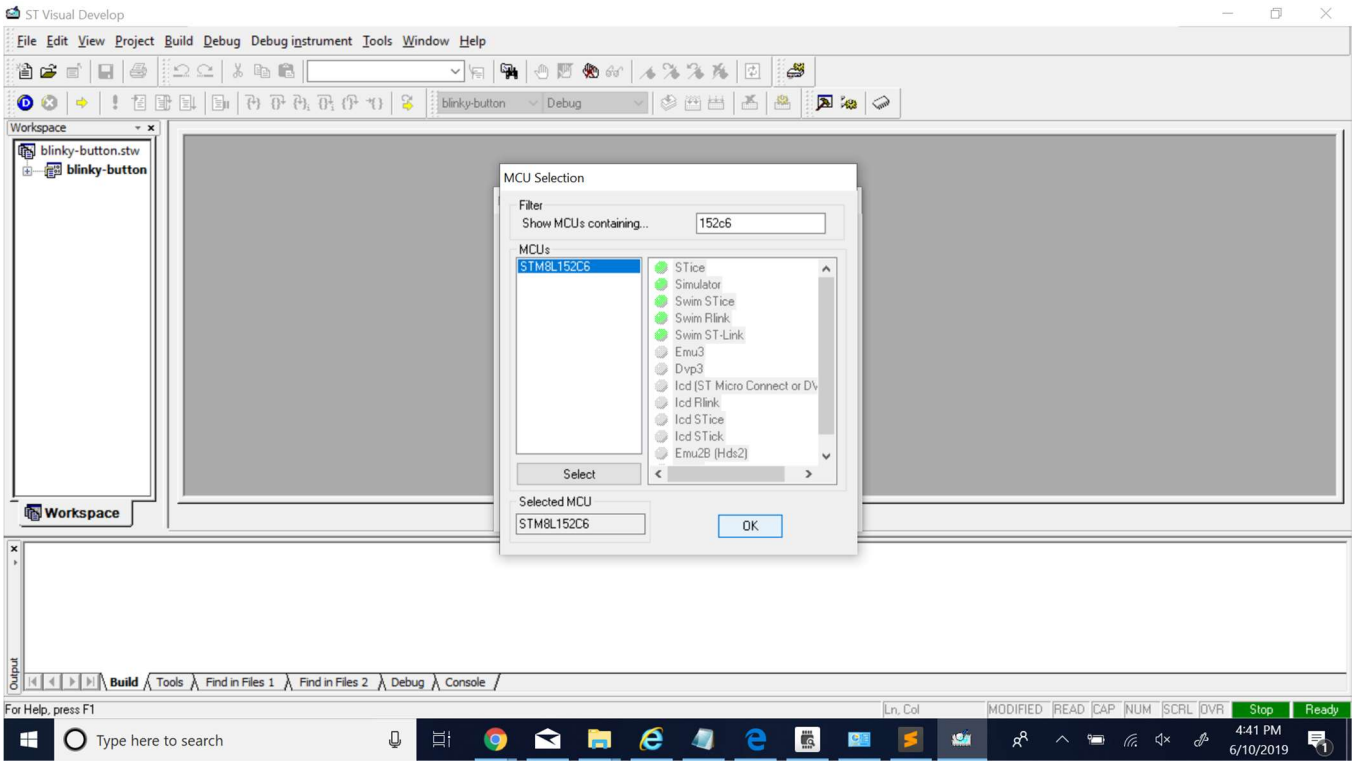
Select the ‘STM8 Cosmic’ for the toolchain

If the ‘Toolchain root’ is not automatically filled, Click the ‘…’ to find the Cosmic C Compiler folder location i.e.

C:\Program Files (x86)\COSMIC\FSE_Compilers\CXSTM8

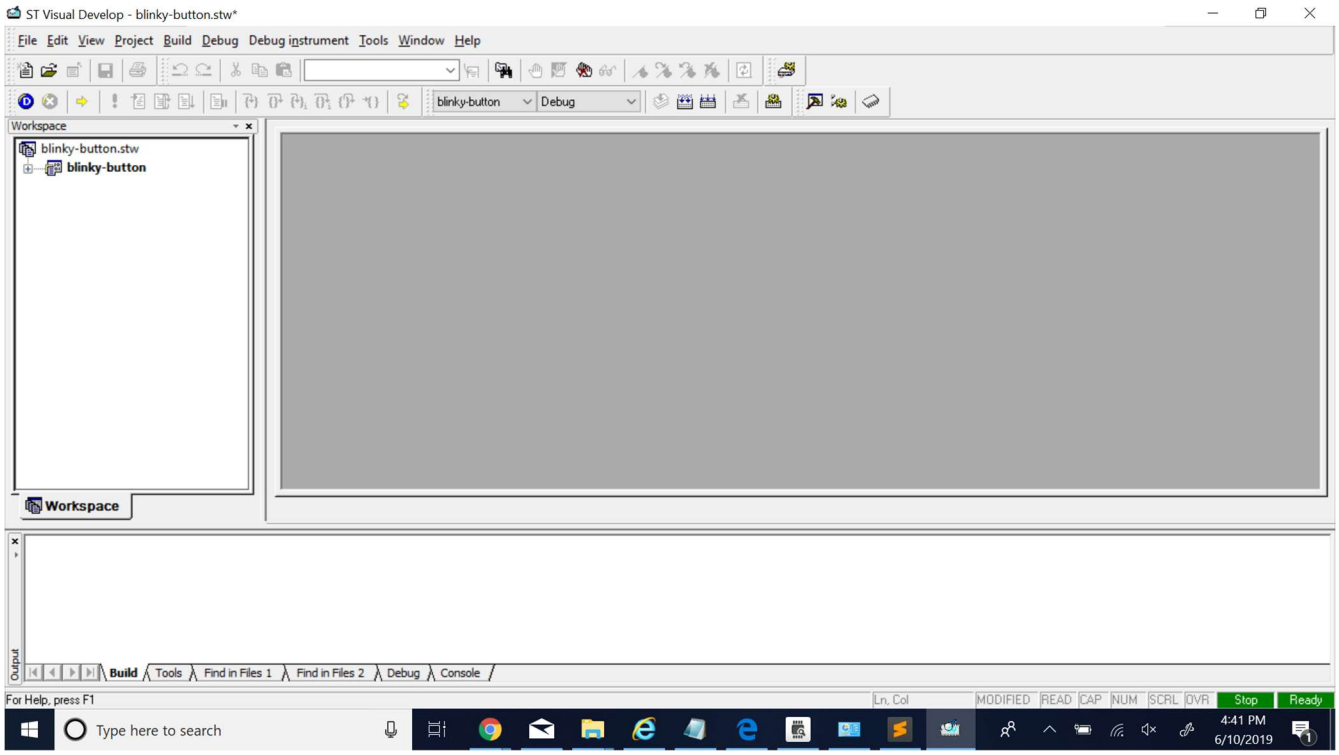


26.) Next select your MCU (device) (i.e. **STM8L152C6** for the STM8L-DISC)

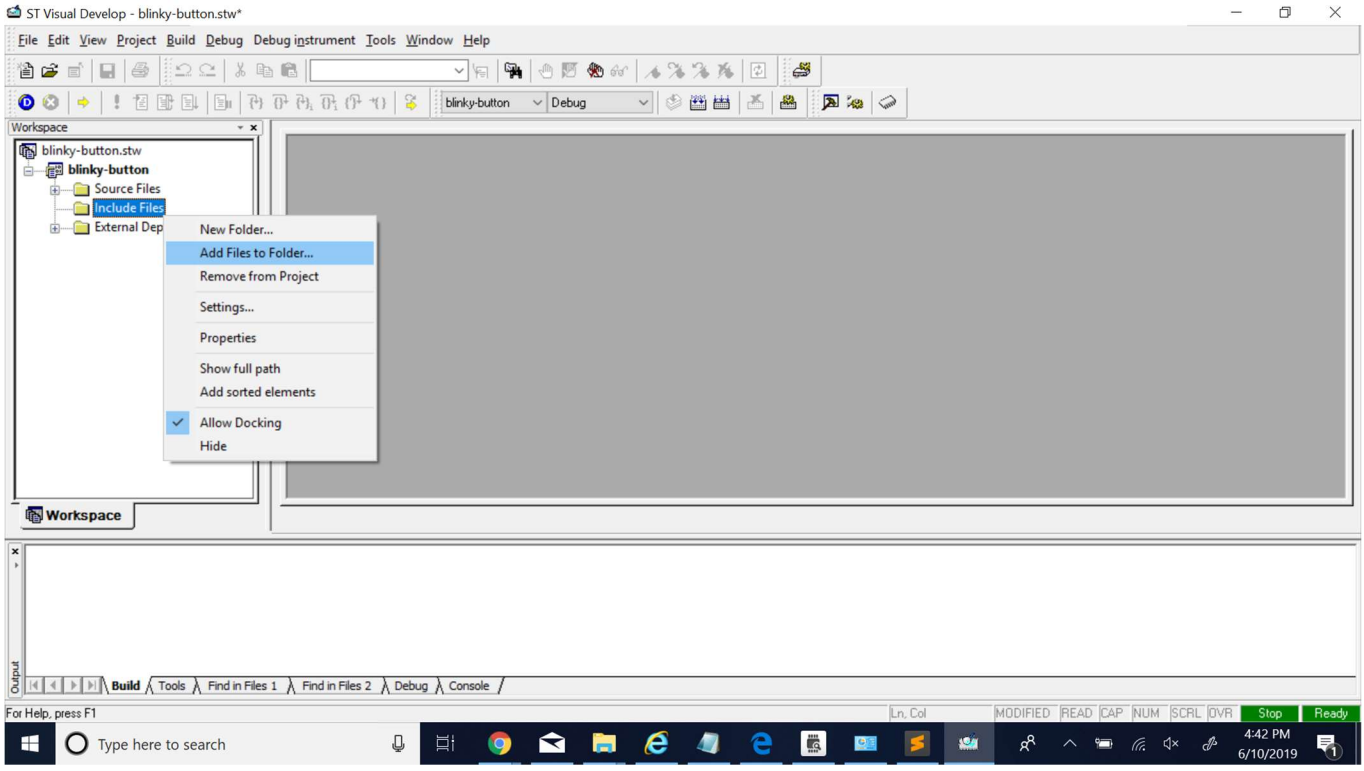


Then press Select -> OK

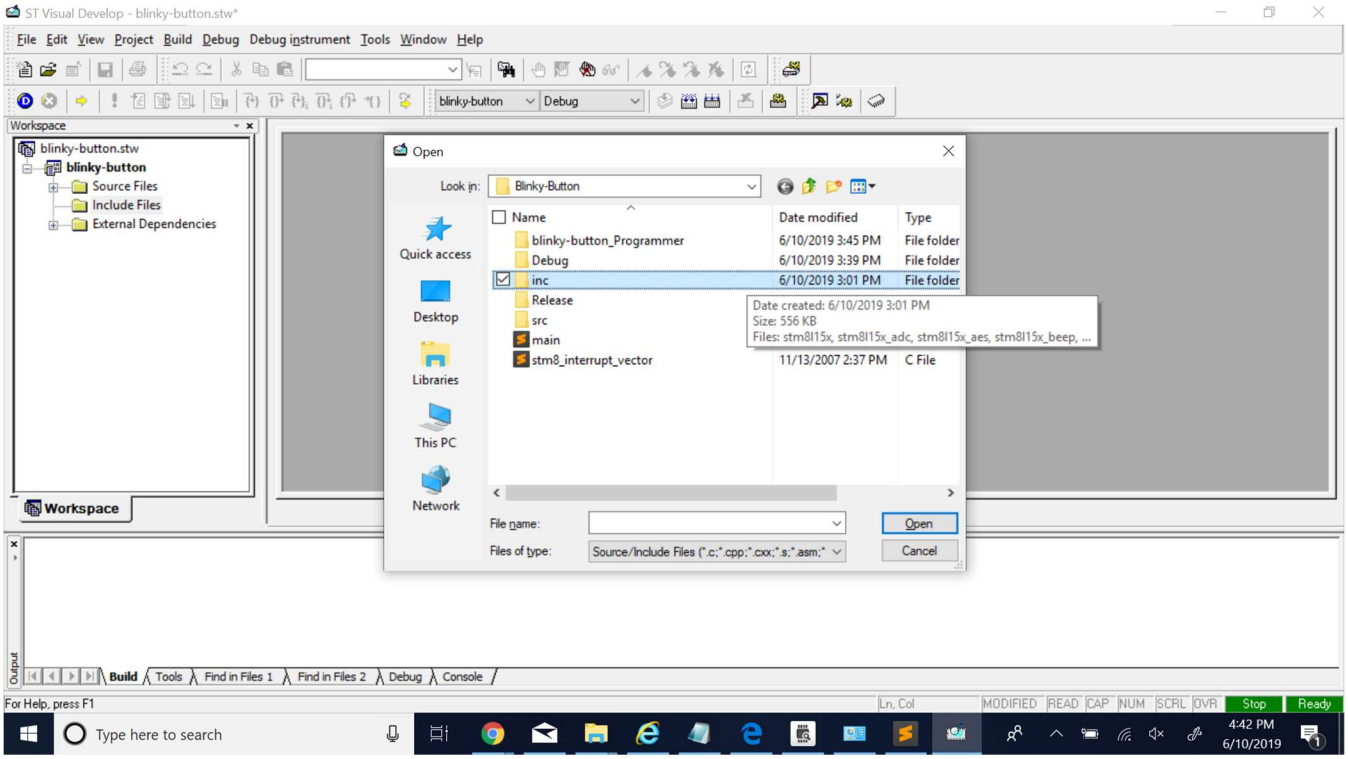
27.) At this point, it should look something like this.



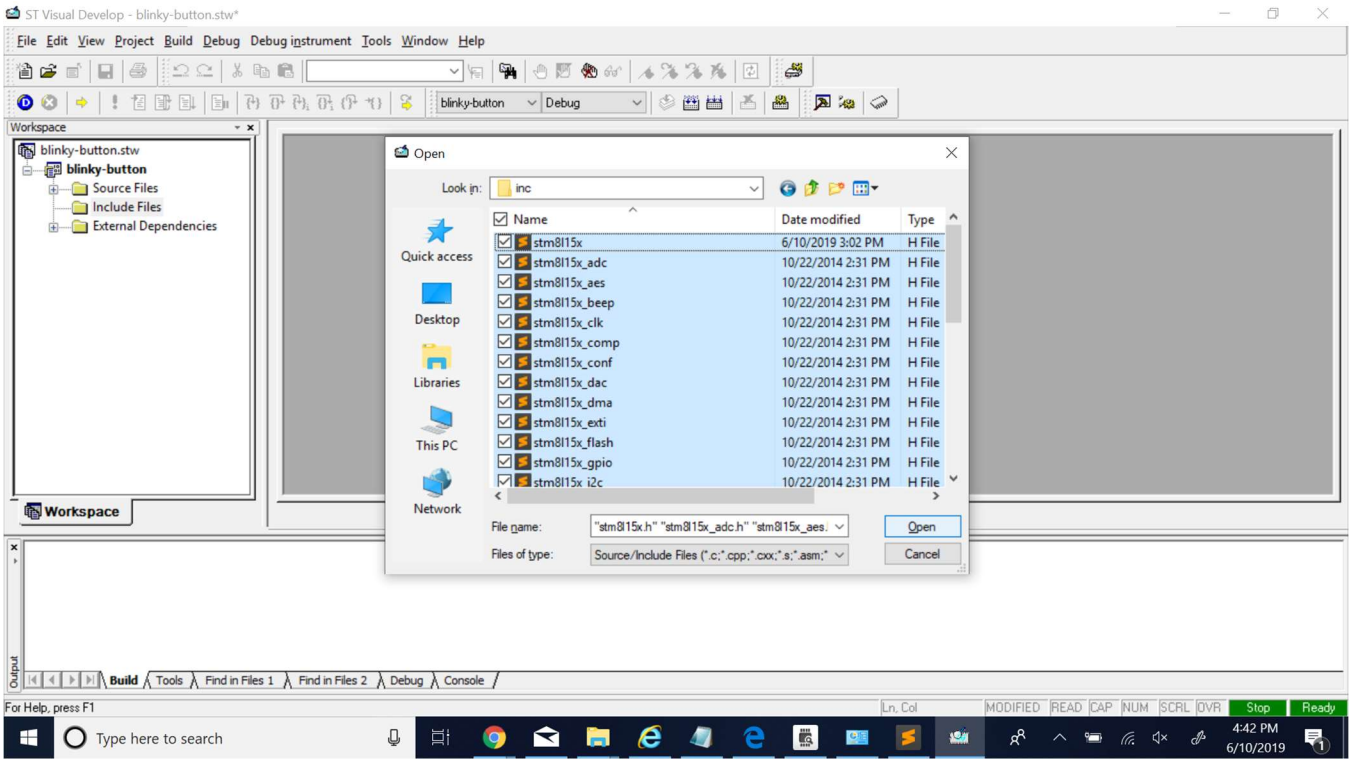
28.) Click the '+' beside your project name in the 'Workspace' section on the left. Then right click the "Include Files" folder and select 'Add Files to Folder...'



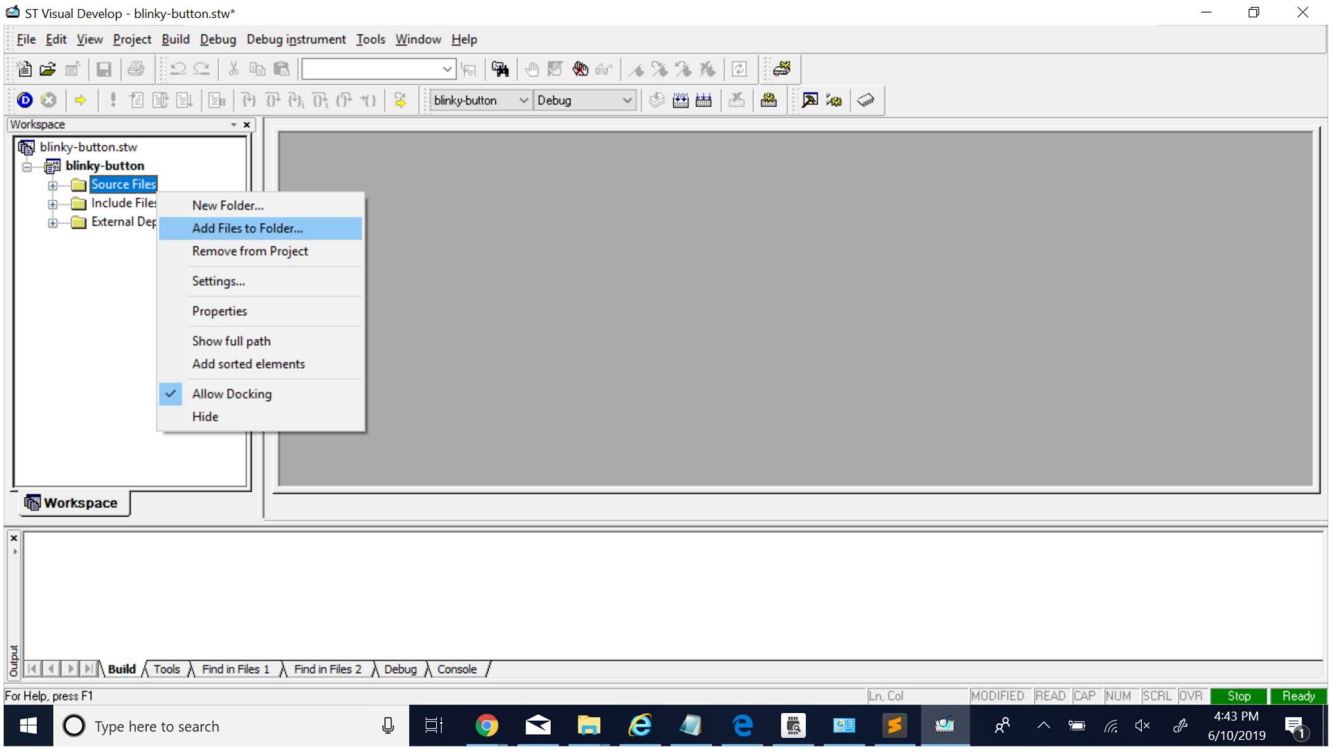
29.) This will pop up and go into the ‘inc’ folder from earlier.



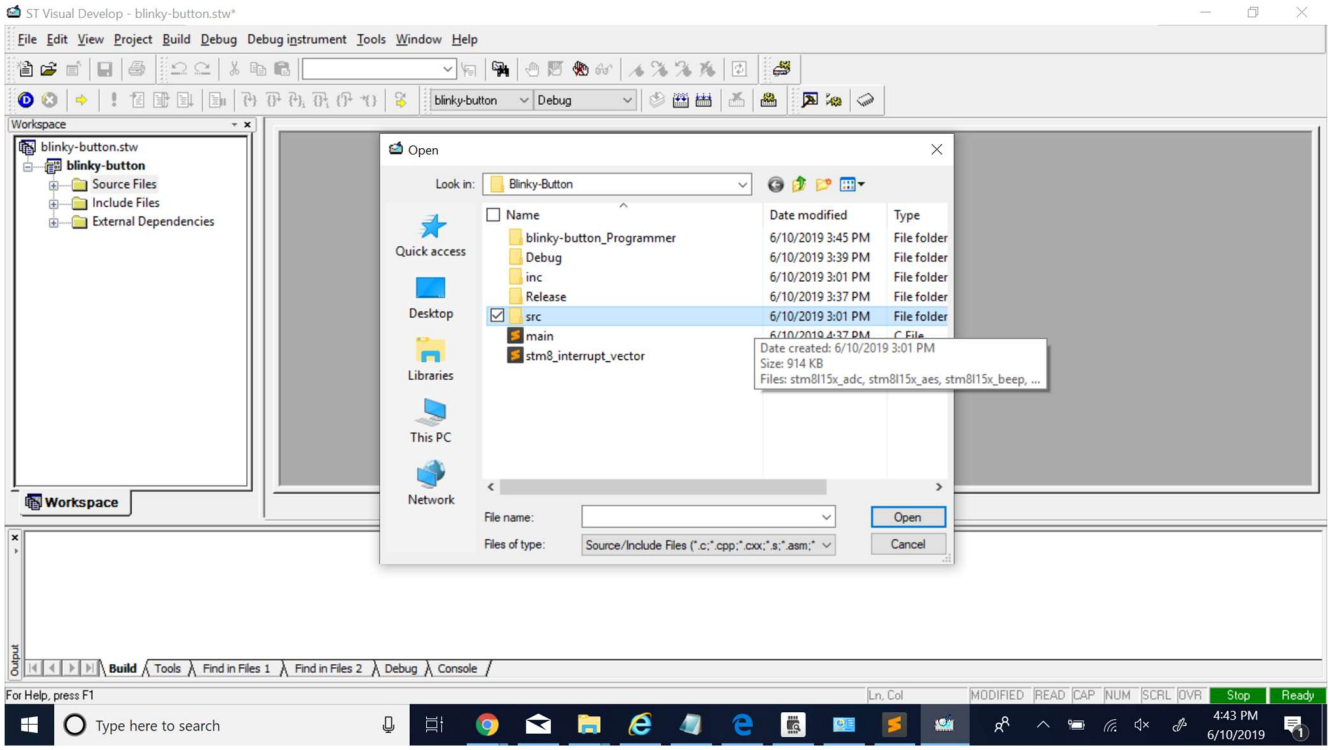
30.) Select every file (CTRL + A) in the ‘inc’ folder and click ‘Open’



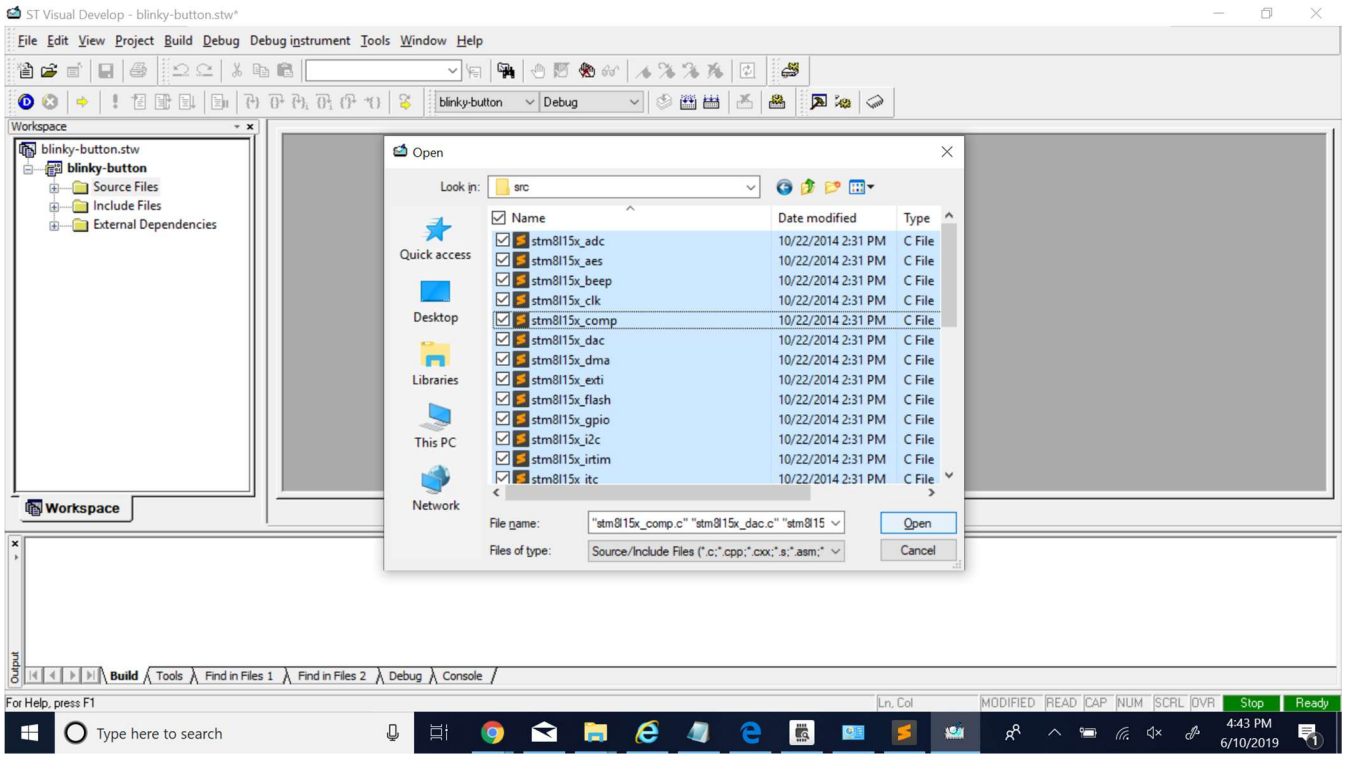
31.) Repeat for the ‘Source Files’ folder. Right click and ‘Add Files to Folder...’



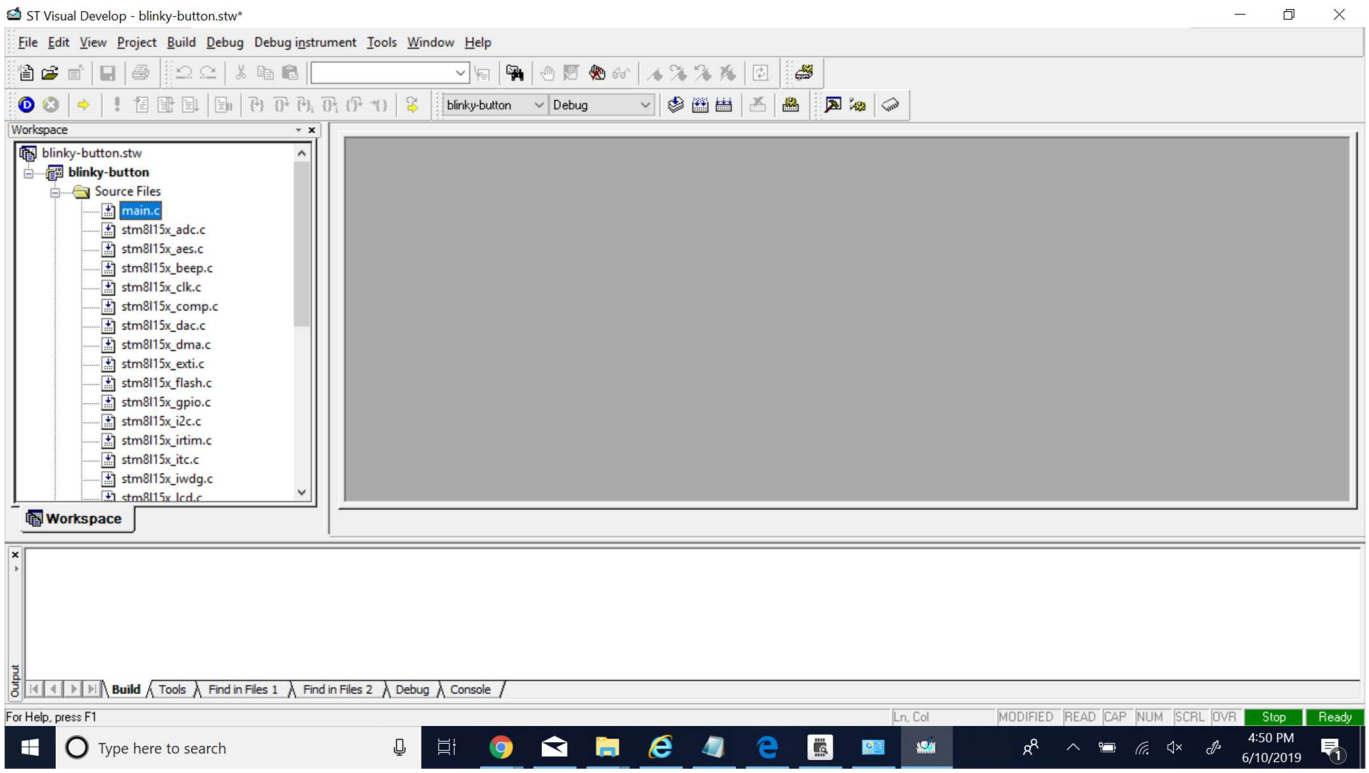
32.) Select the ‘src’ folder this time.

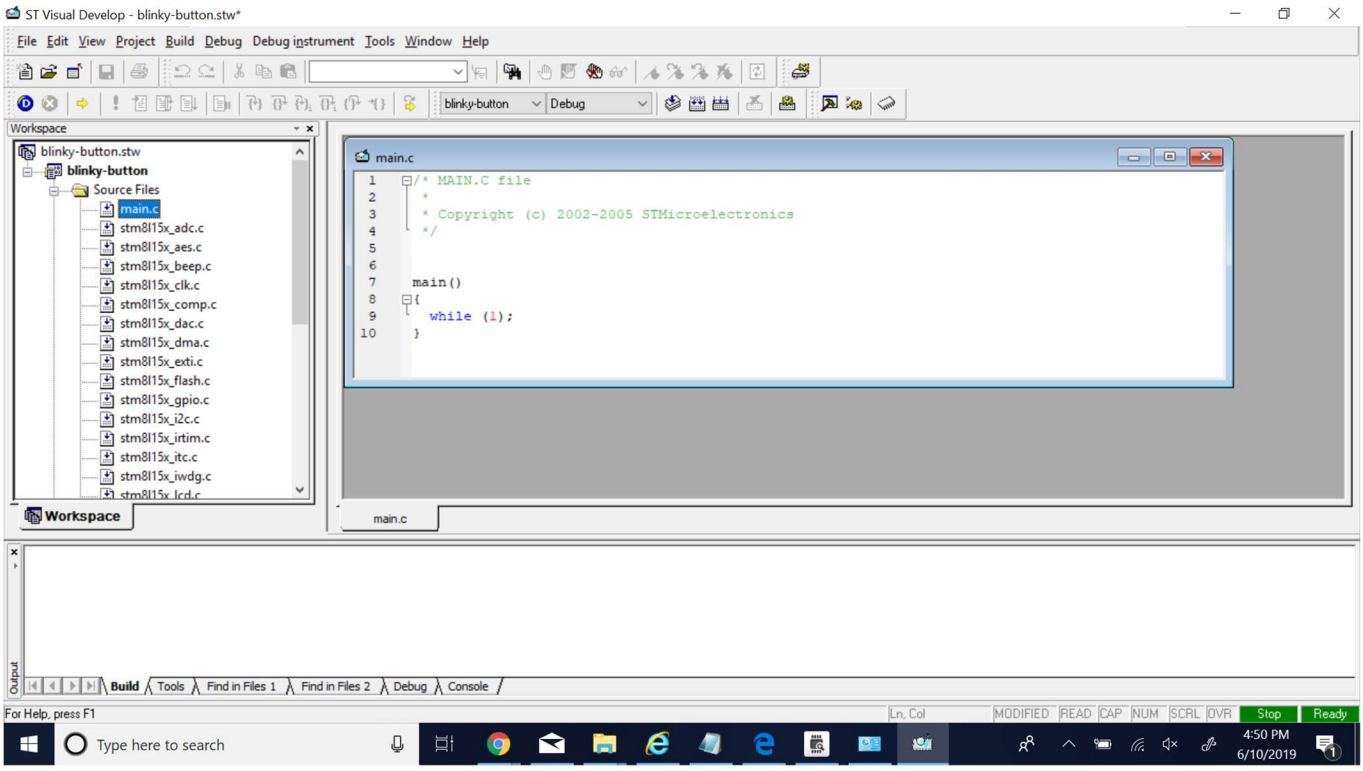


33.) Select all files (CTRL + A) and press ‘Open’



34.) Expand the ‘Source Files’ Folder and Open the ‘main.c’ file





****

You can repeat the above steps for any new project you want to create

****

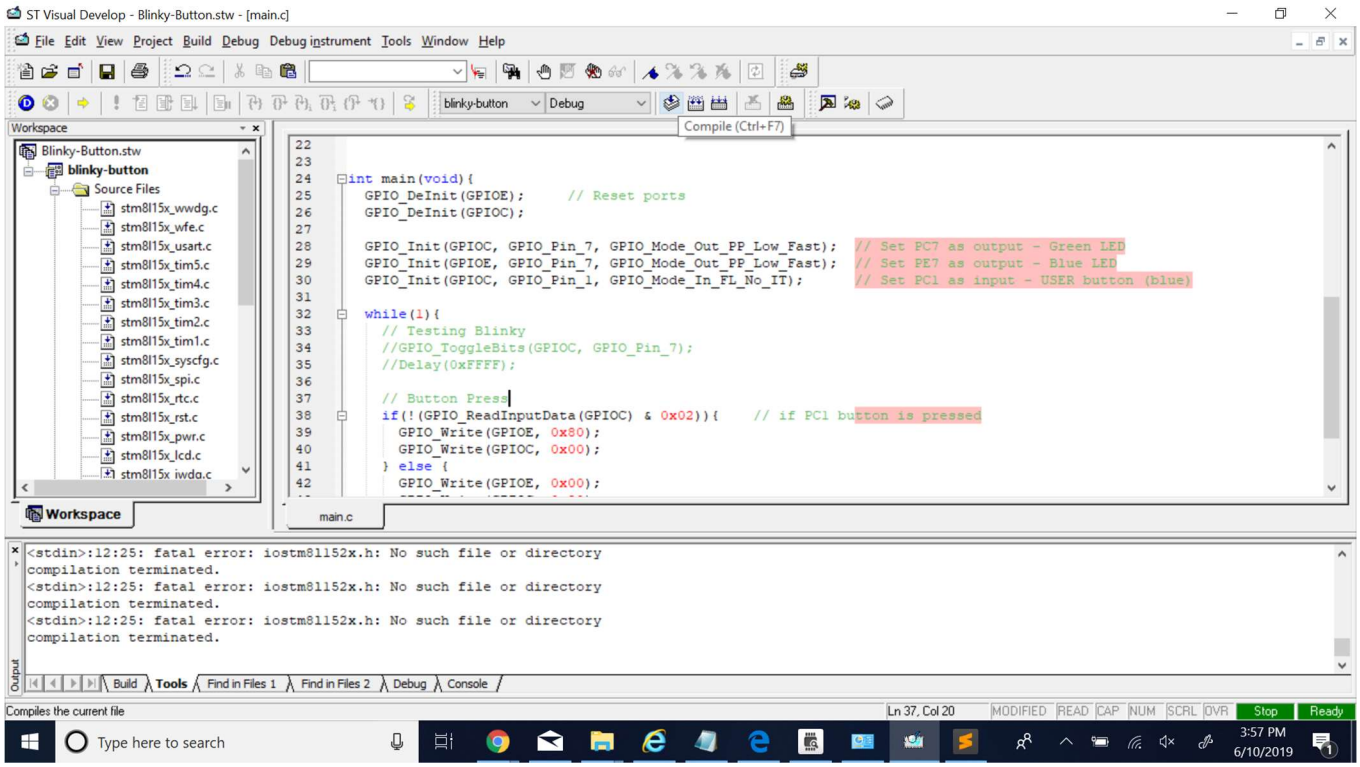
35.) When starting a new project, you'd want to delete the contents of 'main.c' and set it up yourself. Here, we'll have the two on-board LEDs blink with the push of the on-board USER button.

```
#include "stm8l15x.h"
#include "stm8l15x_gpio.h"

void Delay(__IO uint16_t nCount)
{
  /* Decrement nCount value */
```

```
  while (nCount != 0)
  {
    nCount--;
  }
}


int main(void){
   GPIO_DeInit(GPIOE);          // Reset ports
   GPIO_DeInit(GPIOC);

   GPIO_Init(GPIOC, GPIO_Pin_7, GPIO_Mode_Out_PP_Low_Fast);    // Set PC7 as output - Green LED
   GPIO_Init(GPIOE, GPIO_Pin_7, GPIO_Mode_Out_PP_Low_Fast);  // Set PE7 as output - Blue LED
   GPIO_Init(GPIOC, GPIO_Pin_1, GPIO_Mode_In_FL_No_IT);       // Set PC1 as input - USER button (blue)

   while(1){
      // Button Press
      if(!(GPIO_ReadInputData(GPIOC) & 0x02)){       // if PC1 button is pressed
        GPIO_Write(GPIOE, 0x80);
          GPIO_Write(GPIOC, 0x00);
      } else {
          GPIO_Write(GPIOE, 0x00);
          GPIO_Write(GPIOC, 0x80);
      }
   }
}
```

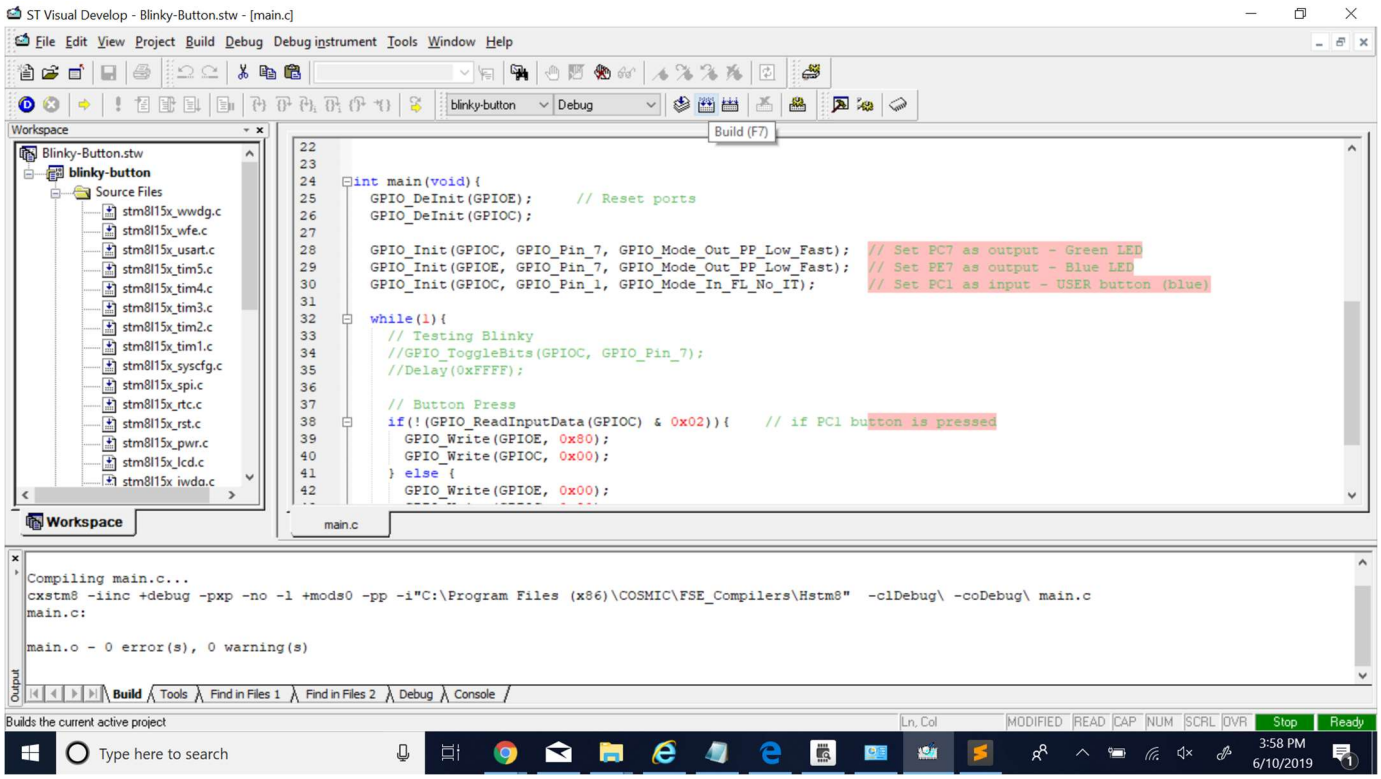
36.) ***** Compiling your Code

Click the icon on the top center tool window beside the ‘Debug’ pull down to compile

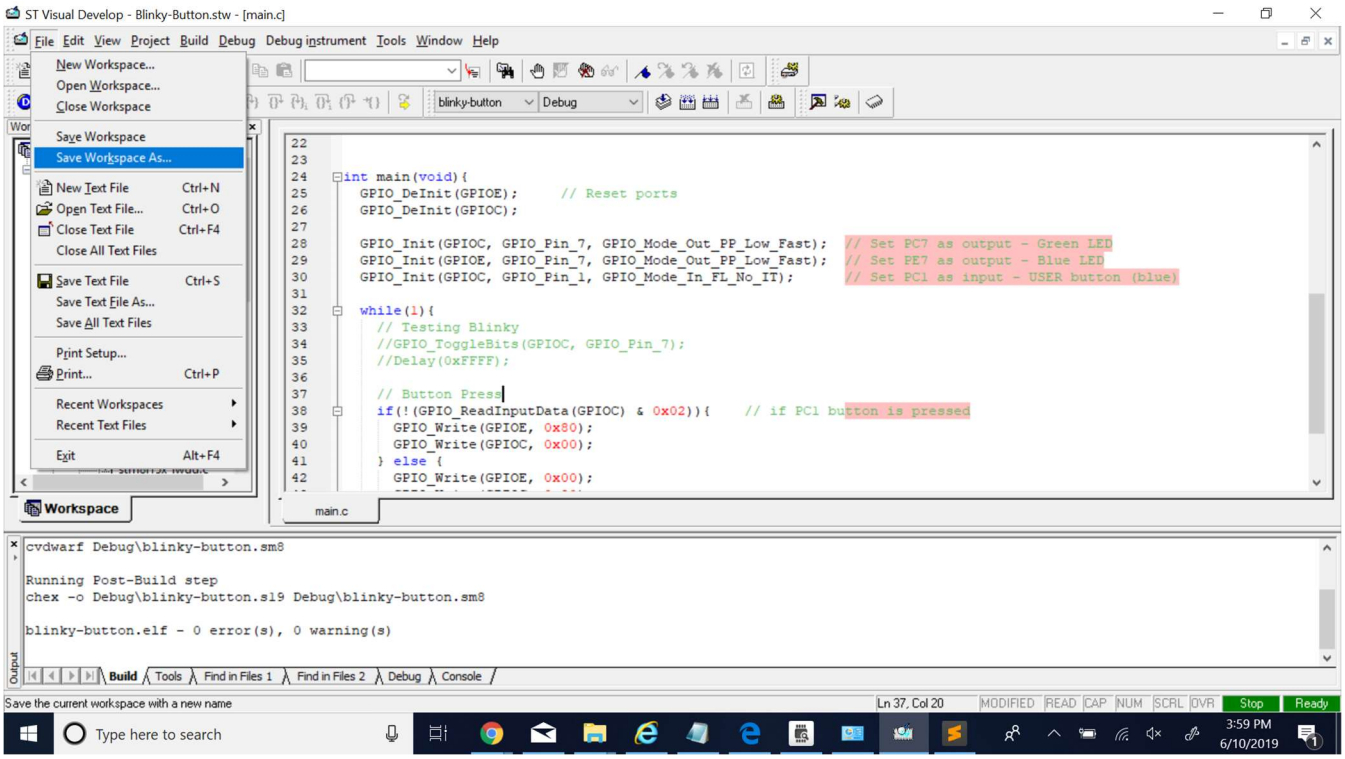


37.) ***** Building your project

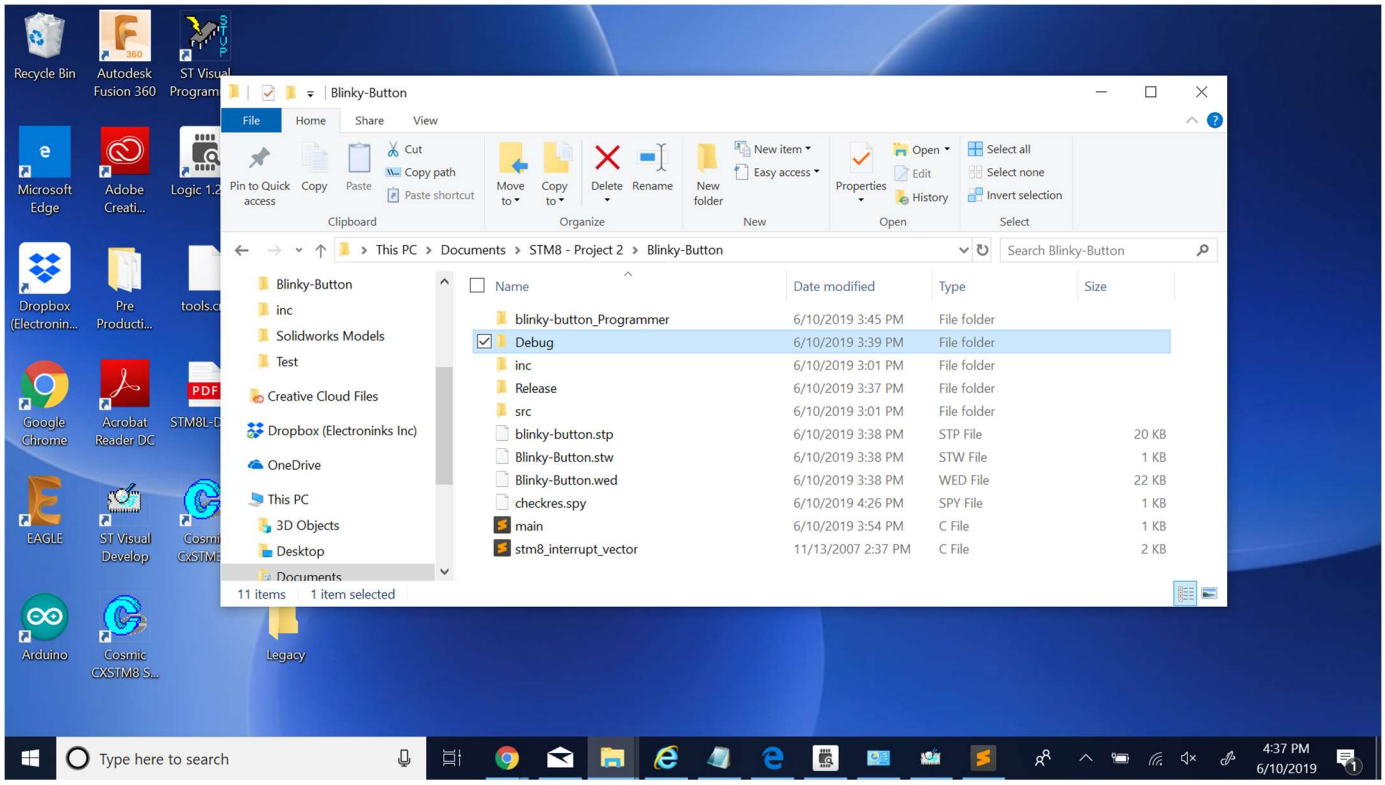
Once done compiling, now ‘Build’ the project by click the icon next to ‘Compile’



38.) Now before we flash the board, we want to ‘Save Workspace as…’ into the project folder to reopen STVD and our project. The Workspace project file is not saved when we first created the STVD project. Make sure it’s saved inside the same project folder from earlier.

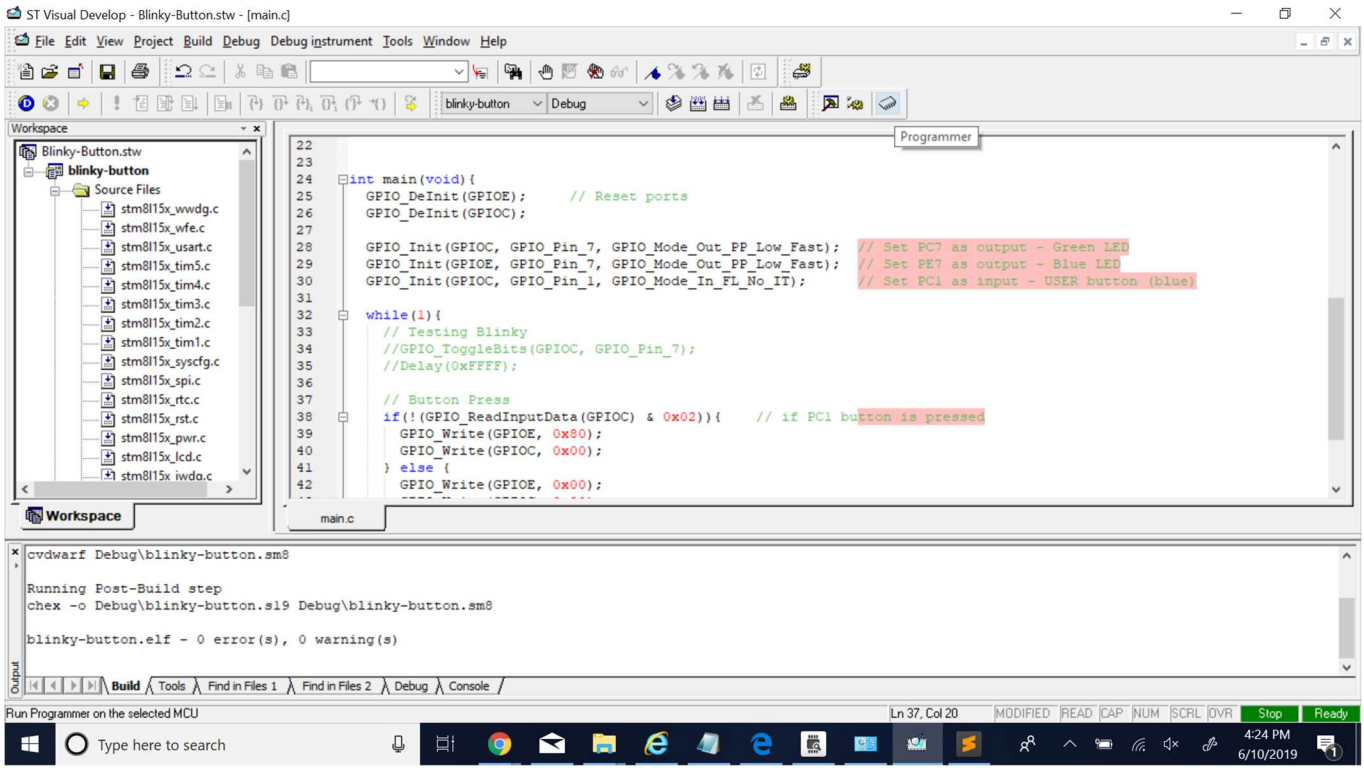


39.) When you built your program, assuming your ‘Workspace’ path location is the same as the folder, you should see a ‘Debug’ folder generated in your project folder (mt project is ‘Blinky-Button’). If you don’t see it, you need to recreate the Workspace and Project.

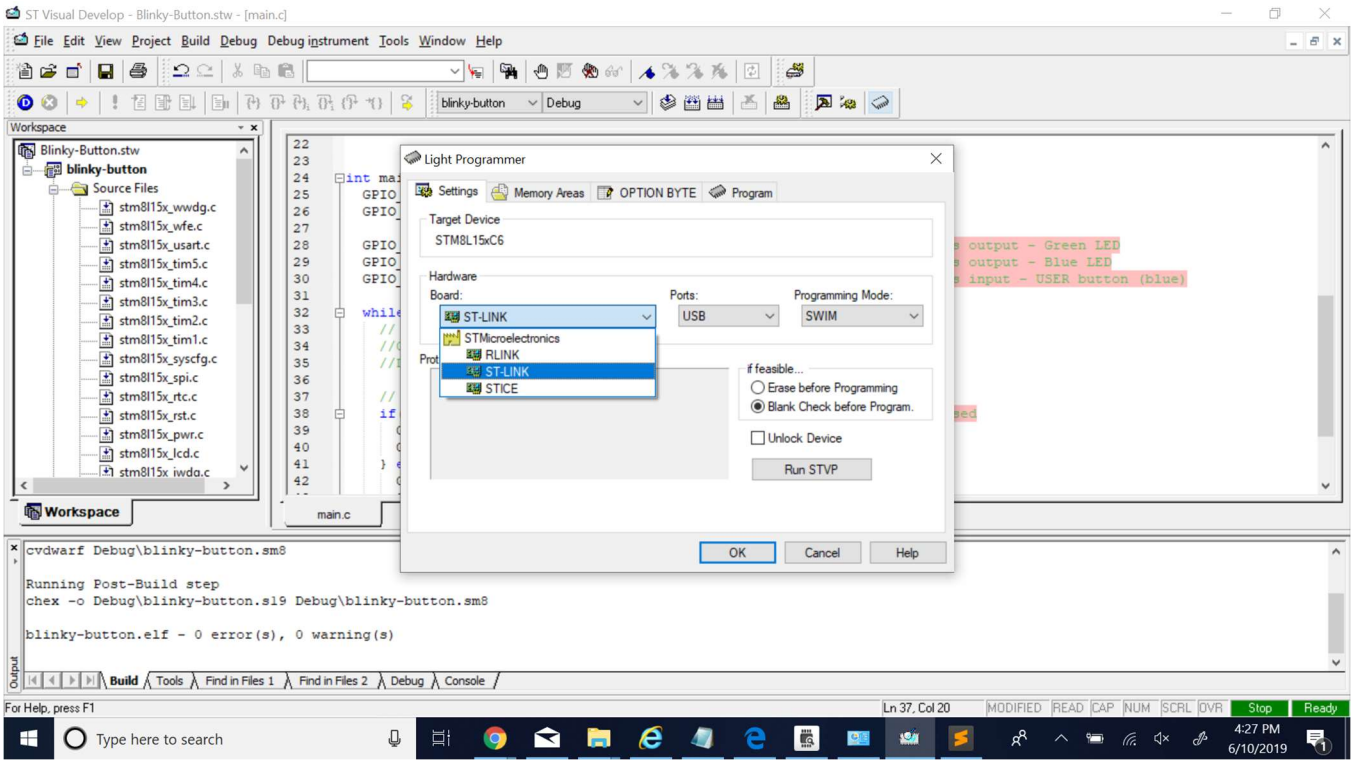


40.) ***** Flashing your board

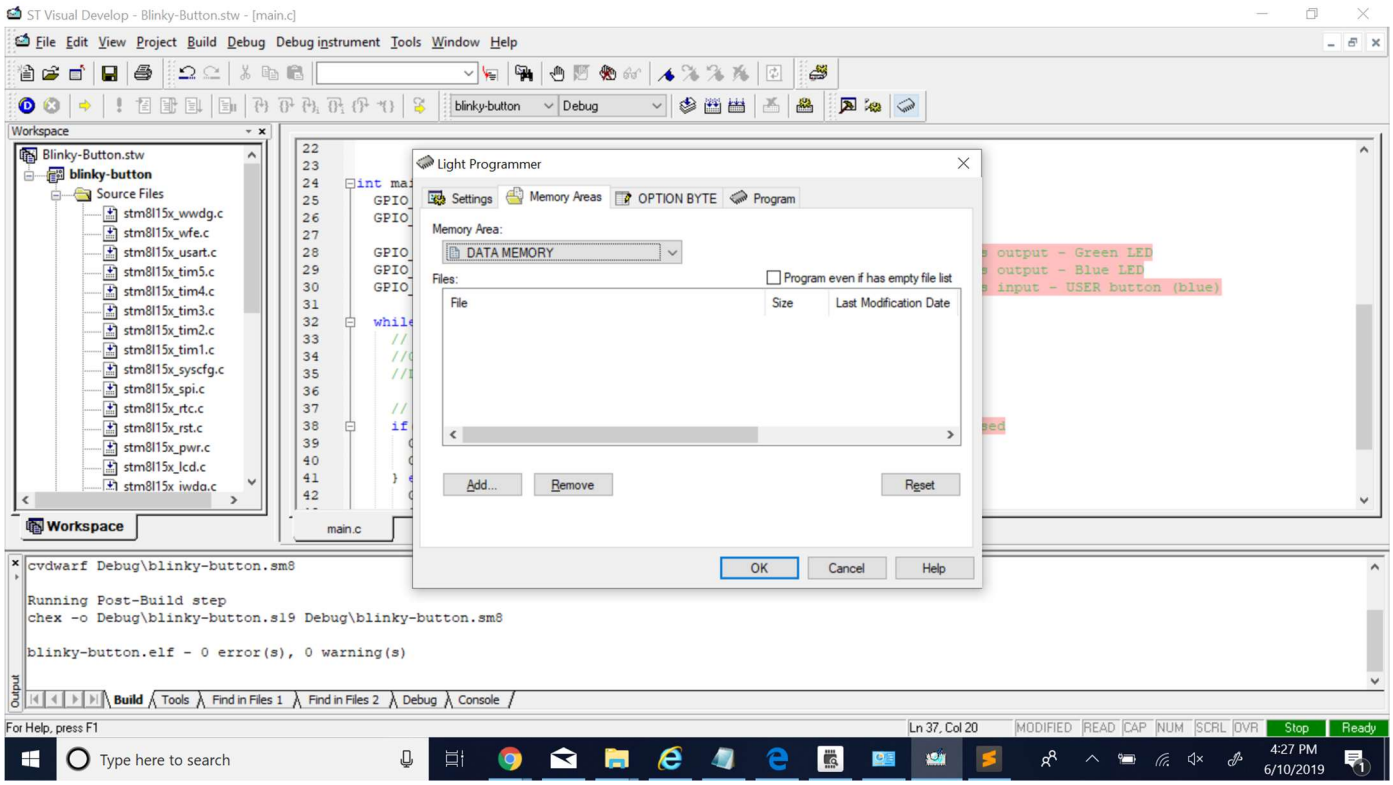
Click on the ‘Programmer’ (IC looking icon) to Flash your board.



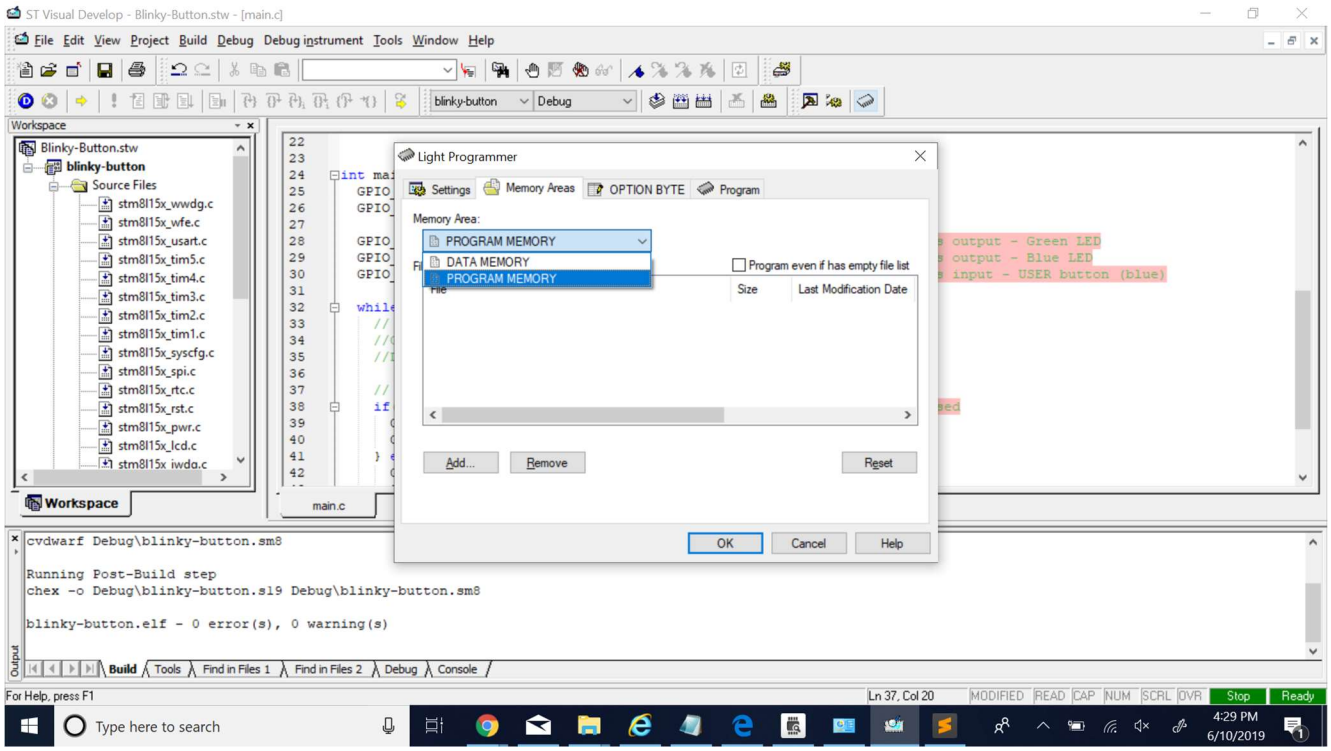
41.) Make sure you have the correct ‘Target Device’. Change the ‘Board’ in the ‘Hardware’ block to ‘ST-LINK’



42.) Go into the ‘Memory Areas’ tab at the top of the popup



43.) When flashing, you always want to change the 'Memory Area' to 'PROGRAM MEMORY'



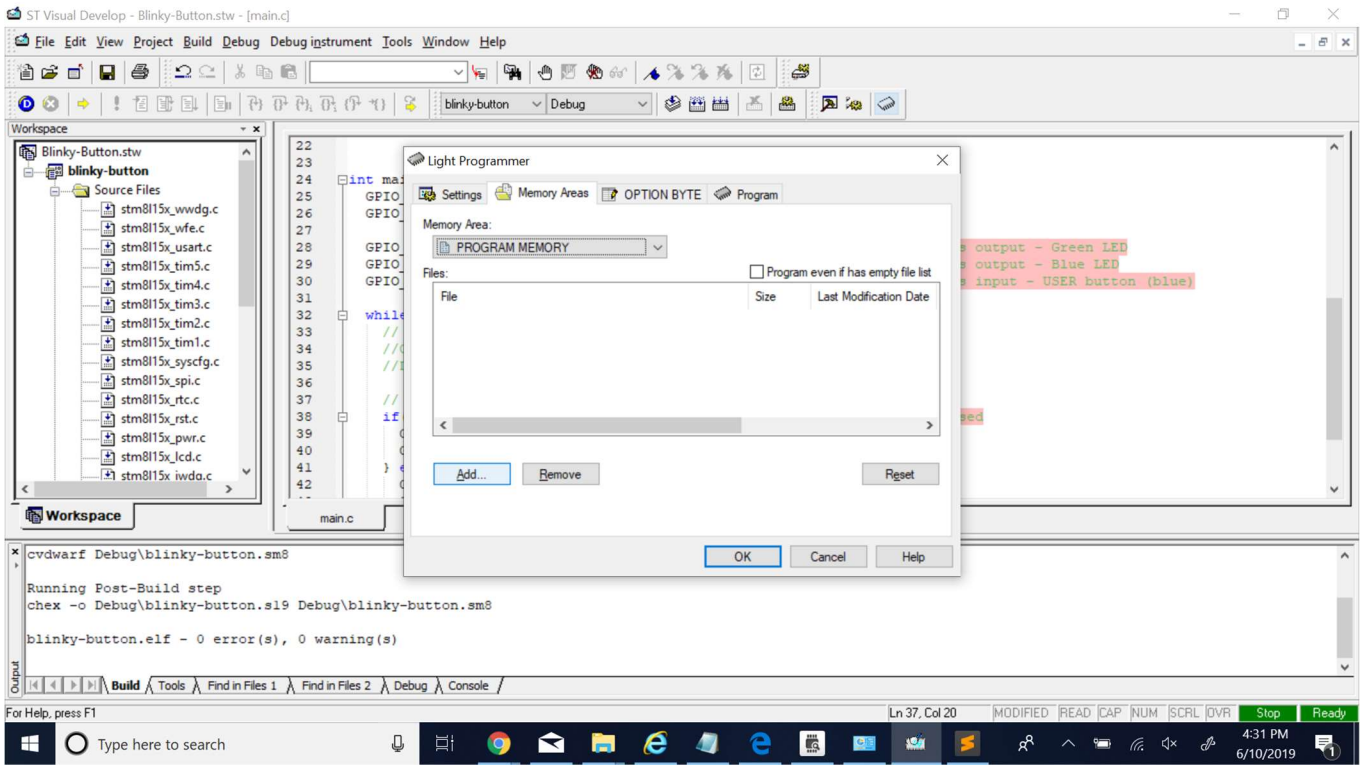
***** Important

You always have to select ‘PROGRAM MEMORY’ each time your try to flash your board. When you close that popup, the ‘Memory Area’ defaults to ‘DATA MEMORY’

*****

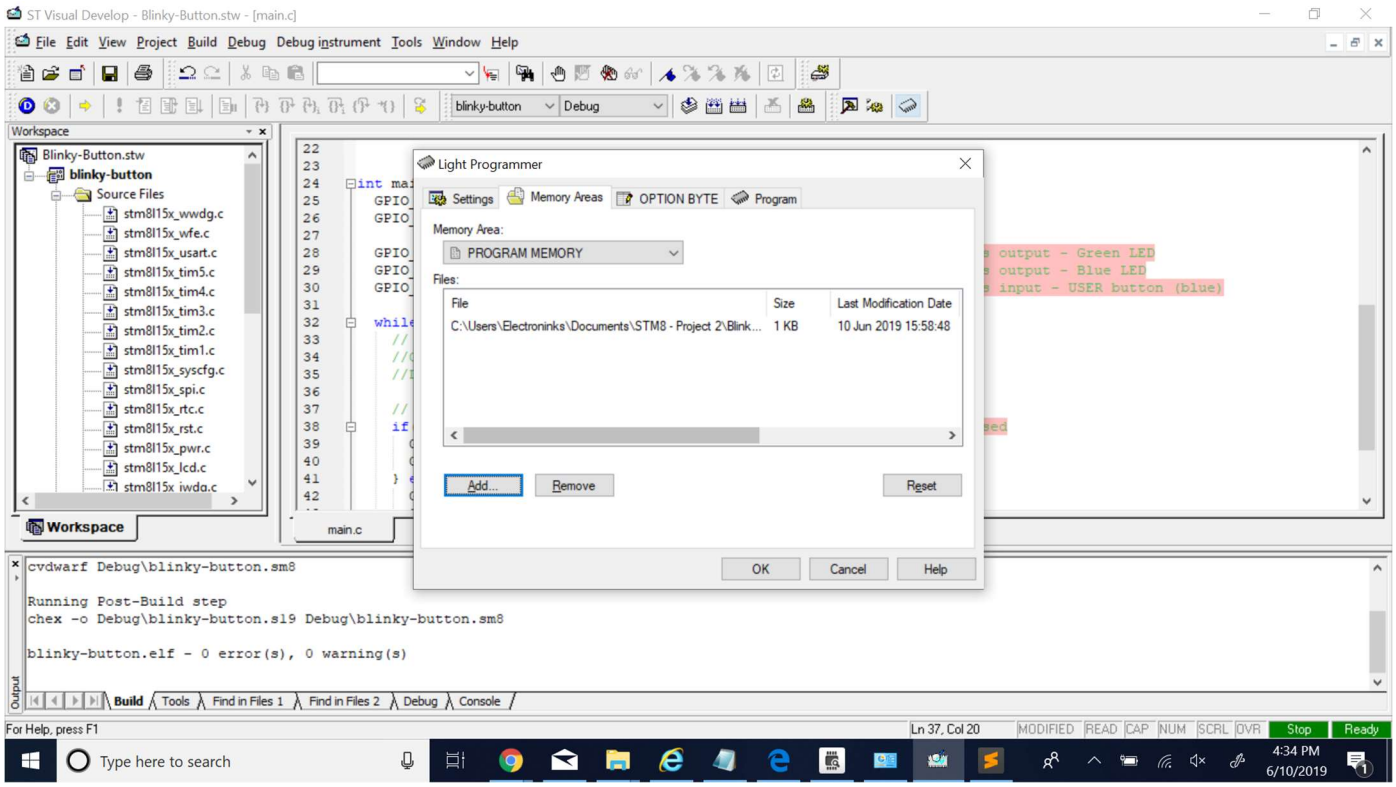
44.) Click ‘Add...’

We want to add the ‘your_project_name.s19’ file that was generated when you built your project into

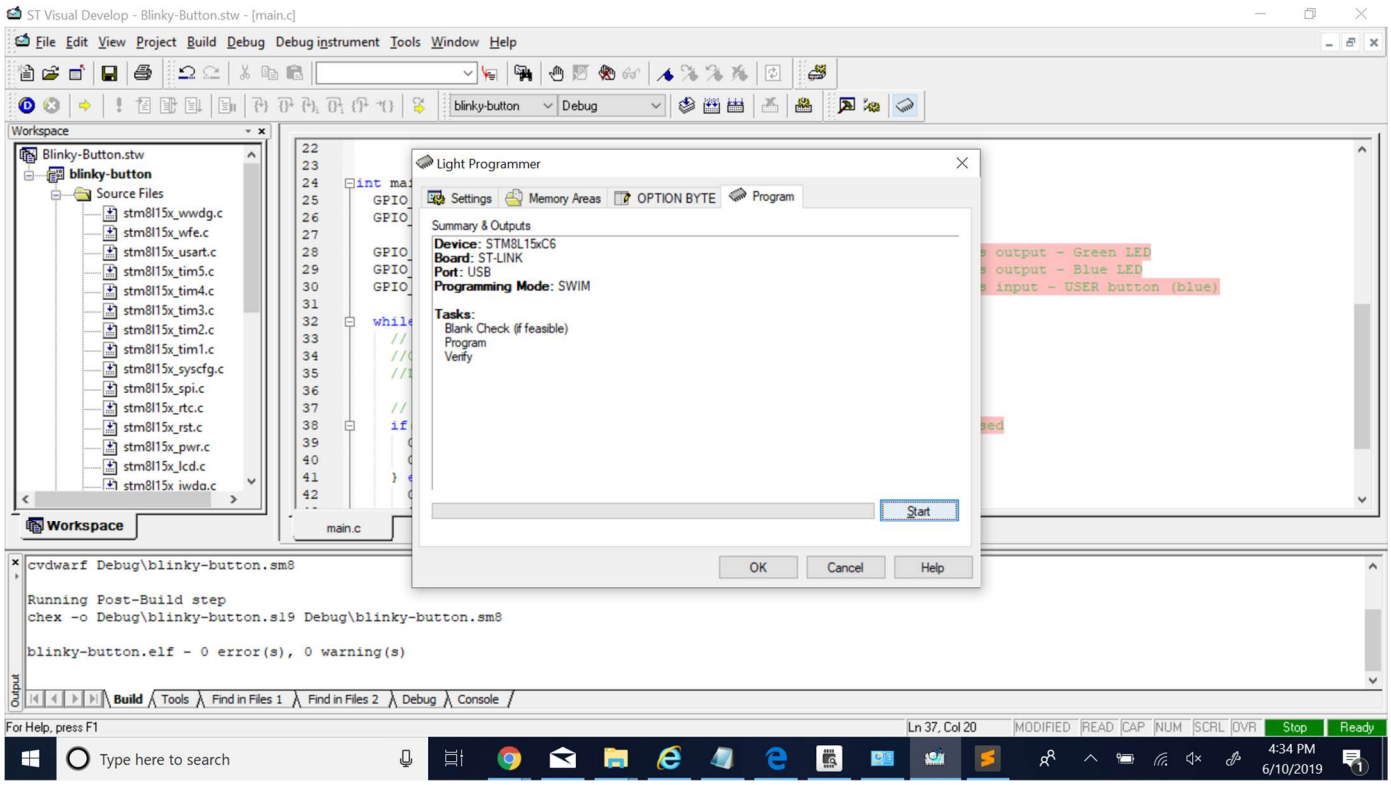


45.) Select your ‘your_project_name.s19’ file (mine is ‘blinky-button.s19’) and press ‘Open’

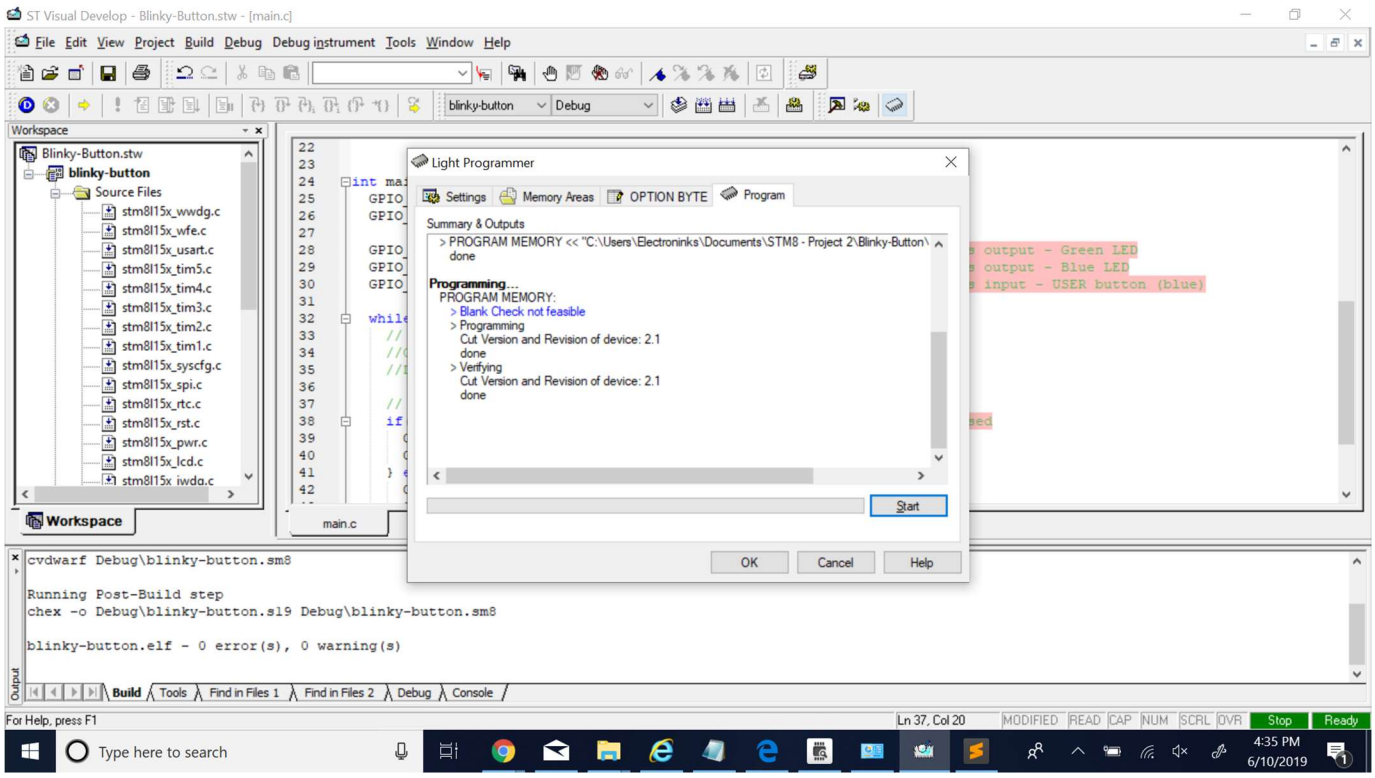
Your ‘Memory Areas’ tab should look like this now



46.) Now go to the ‘Program’ tab. Make sure your board is connected. Press ‘Start’ flash



47.) It should look like this if everything was compiled and built correctly.



You board is now programmed, Happy Coding!