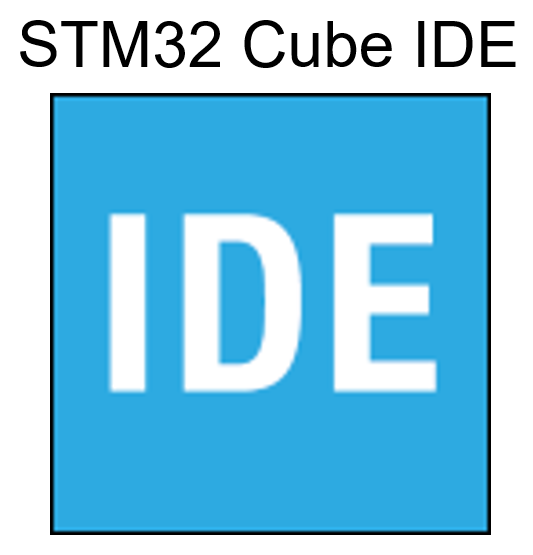
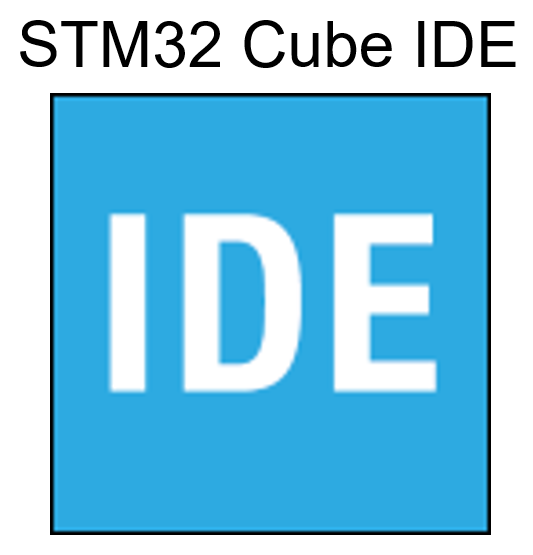
***STM32-Cube IDE initial Start-Up***

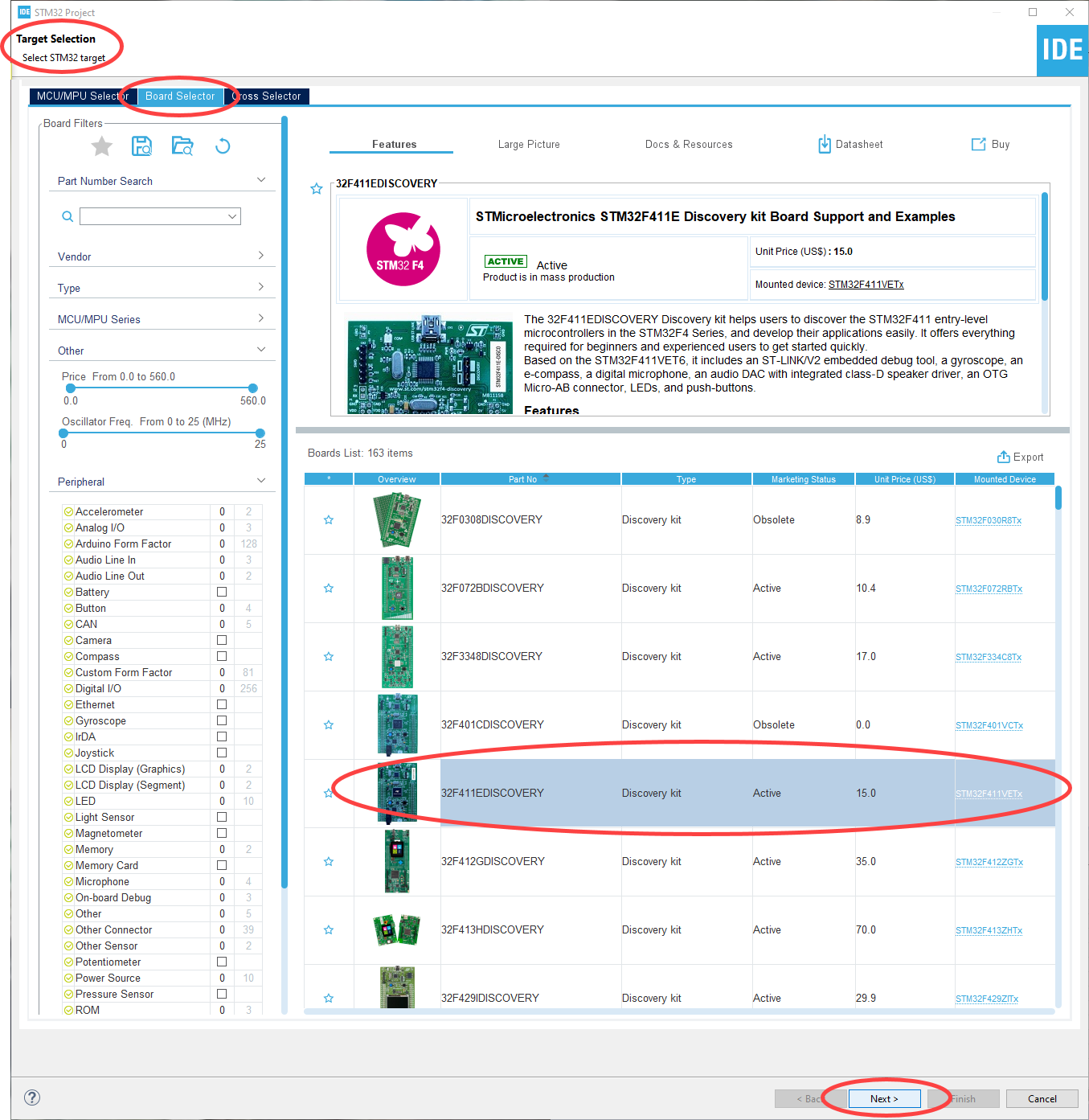
1. Open: 

If this is the first time you’ve opened ***STM32-Cube IDE***, you should be prompted to select a directory to store all your files. This is called your workspace. On RIT computers, this workspace should be created in your student account or on a flash drive. Do not use the local computer, your work will be lost when the computer reboots. When you install ***STM32-Cube IDE*** on your personal computer, you may create this workspace anywhere you like.

If you checked “Use this as the default and do not ask again”, the ***STM32-Cube IDE*** launcher window will not appear in the future.

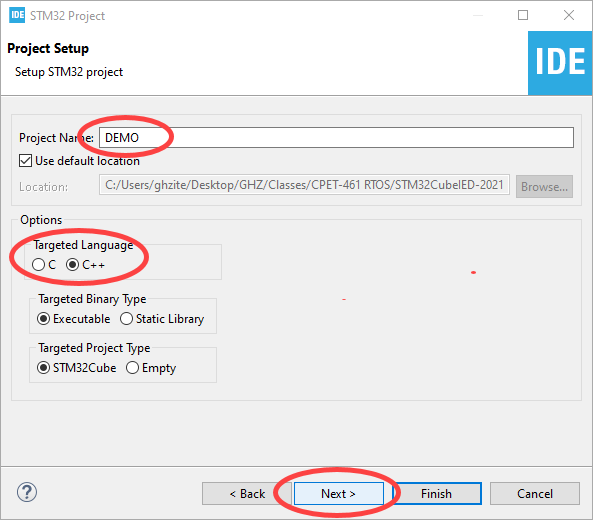
***Creating a new Project in STM-Cube IDE***

1. Open: 
2. Select: File 🡪 New 🡪 STM32 Project, this will invoke the ***STM32-Cube MX*** utility
3. In the **Target Selection** window, select *Board Selector*, the *32F411EDISCOVERY* board (fifth one down), and then *Next*.

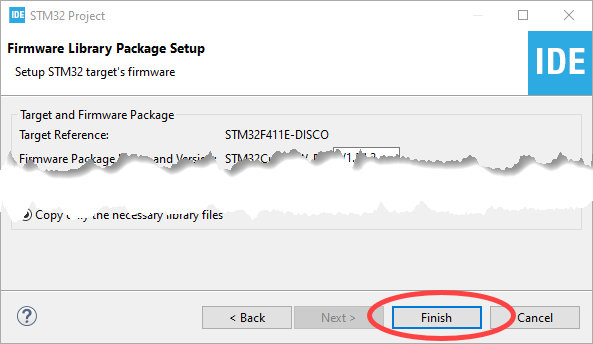


1. In the **Project Setup** window, enter a *Project Name*, select *C++*, and then *Next*.

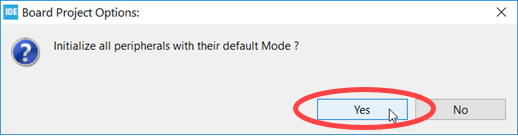
Make sure the location is the workspace you entered previously. If it isn’t, correct the location.



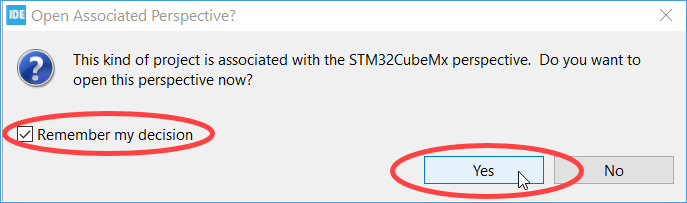
1. In the **Firmware Library Package Setup** window, use all the default settings and then select *Finish*.



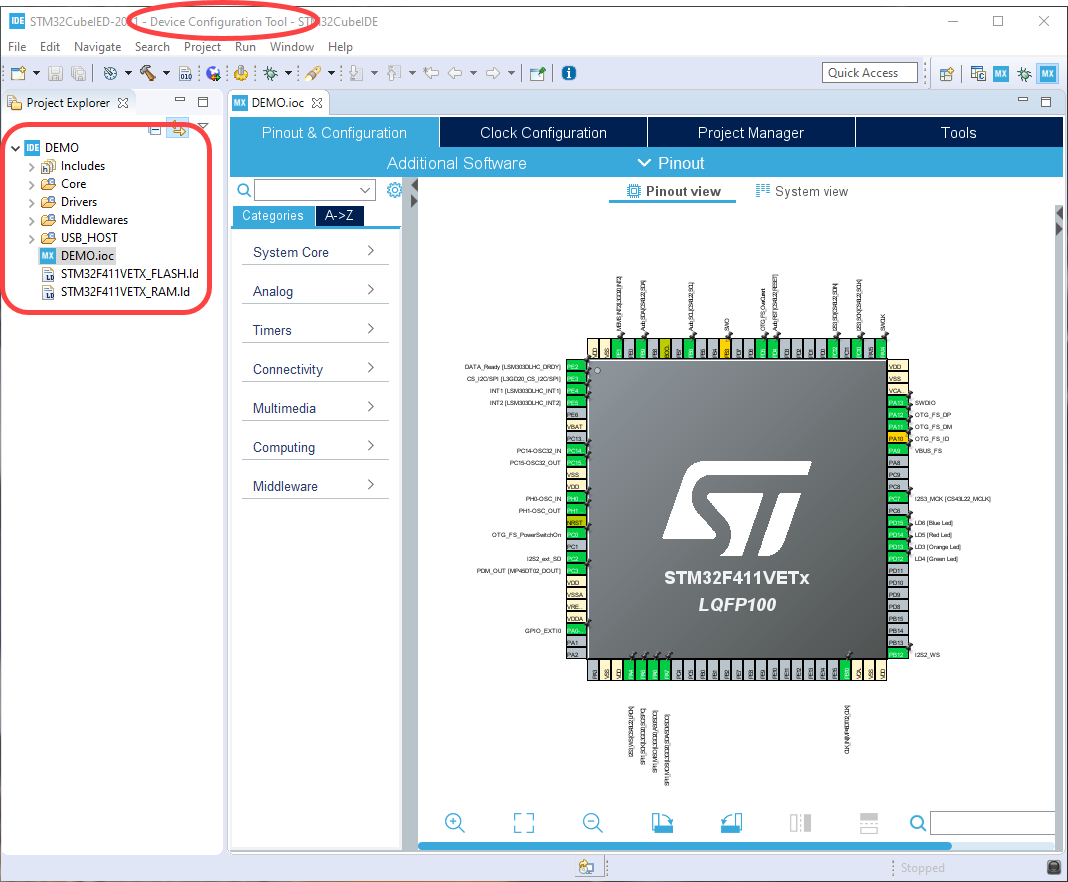
1. Select *Yes*, to initialize all peripherals with their default Mode.



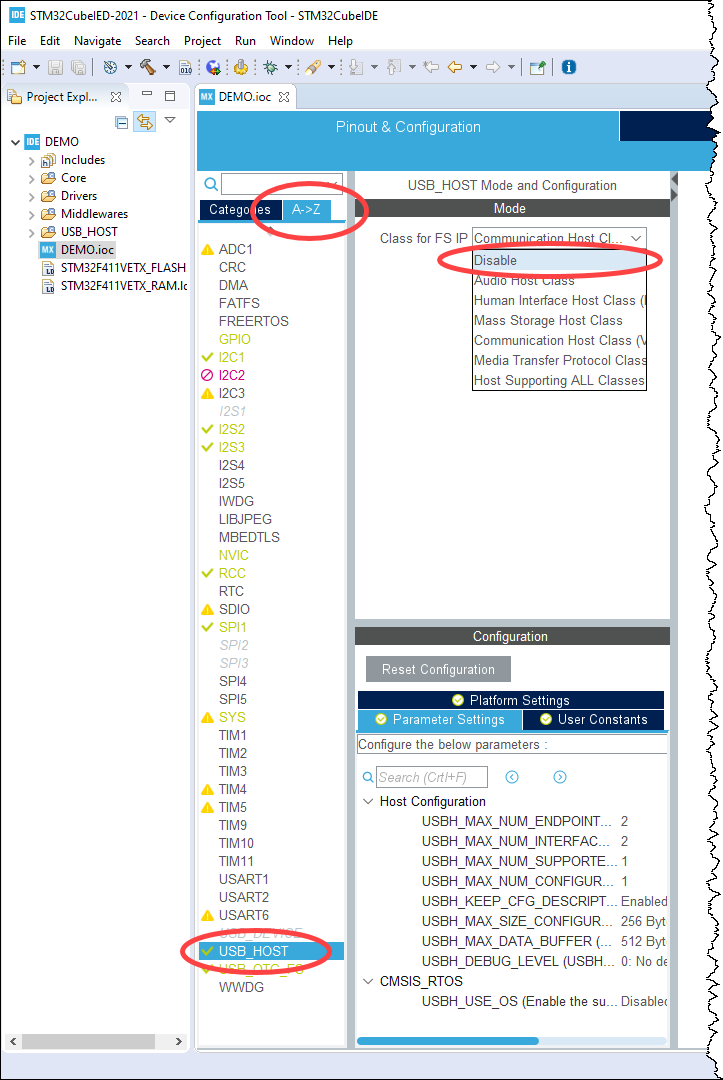
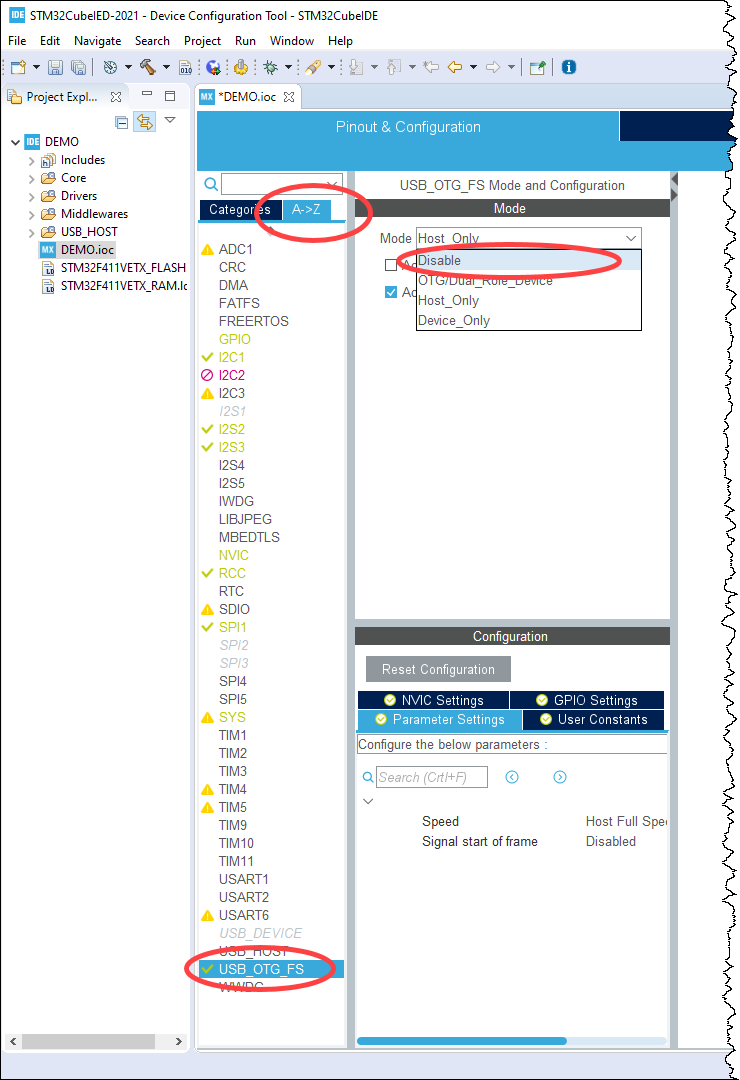
1. Select “Remember my decision” and then select *Yes*. (note: this window will only appear the first time you run the software).



1. Once the project has been created, the **Device Configuration Tool**window will display the MCU’s pinout. For your designs, the pinout will be based on the layout of the 32F411EDISCOVERY board. This new project, DEMO, is listed under the **Project Explorer** tab where the  icon next to the project name indicates it is the current (active) project.



1. With three exceptions, this design will use the default configuration. The first two changes required are to disable **USB\_HOST** and **USB\_OTG\_FS**. To do so, in the **Pinout & Configuration Window**, select **A-Z**, **USB\_HOST**, and **Disable** (from the dropdown window). Repeat this for **USB\_OTG\_FS**.

The third changes required is to change the High Speed Clock (HSE) from **BYPASS Clock Source** to **Crystal/Ceramic Resonator**. To do so, in the **Pinout & Configuration Window**, select **A-Z**, **RCC**, and change the High-Speed Clock (HSE) to **Crystal/Ceramic Resonator** (from the dropdown window). ***After this change, you must power cycle the board.***

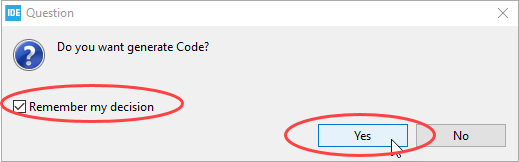
A screenshot of a computer

Description automatically generated

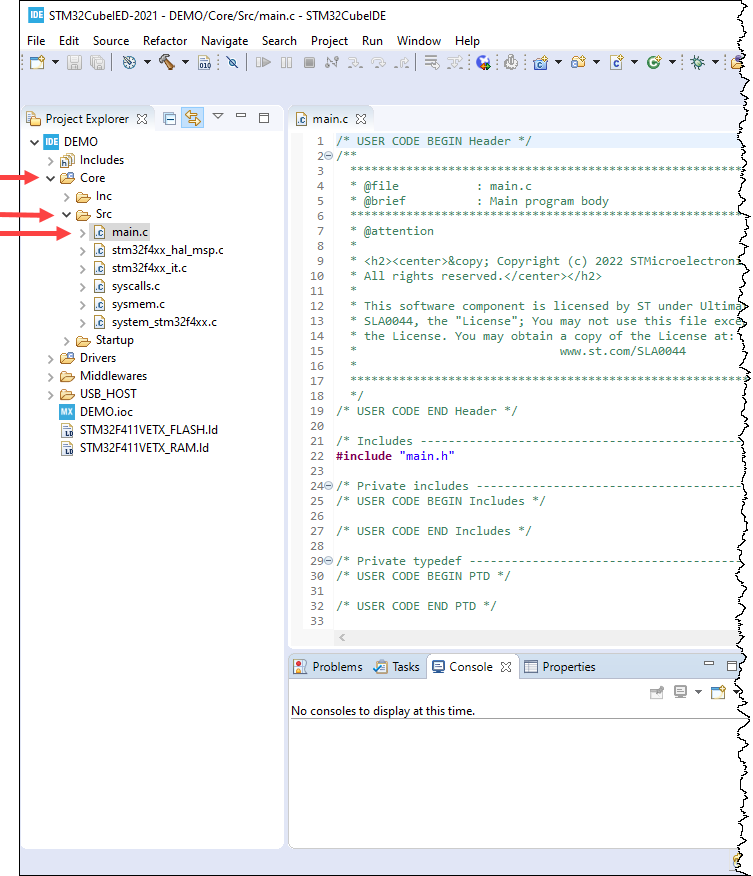
1. To save the hardware configuration and to create the source code files, select the X to *close* the **Device Configuration Tool**.

|  |  |
| --- | --- |
| C:\Users\ghzite\AppData\Local\Temp\SNAGHTML8695933.PNG | C:\Users\ghzite\AppData\Local\Temp\SNAGHTML86c4c25.PNG |

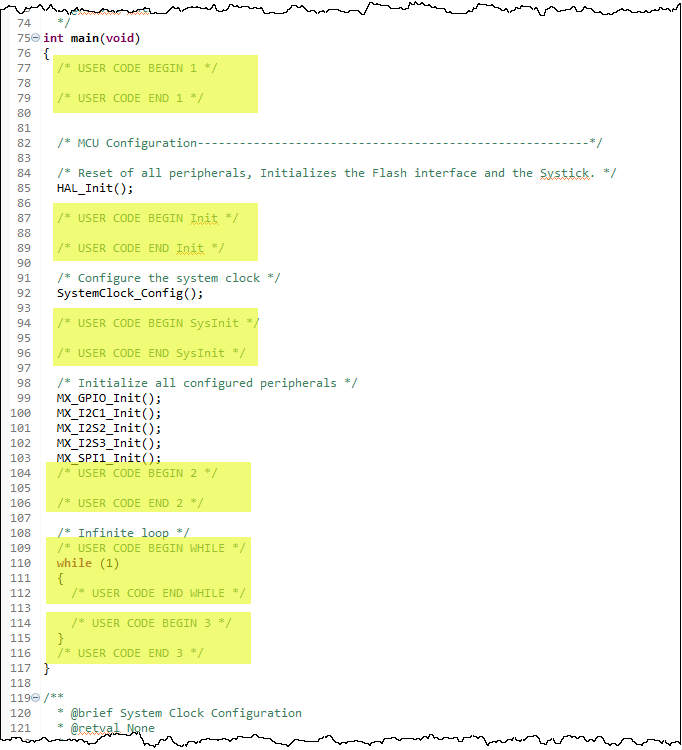
1. Select “Remember my decision” and then select *Yes*. (note: again, this window will only appear the first time you run the software).



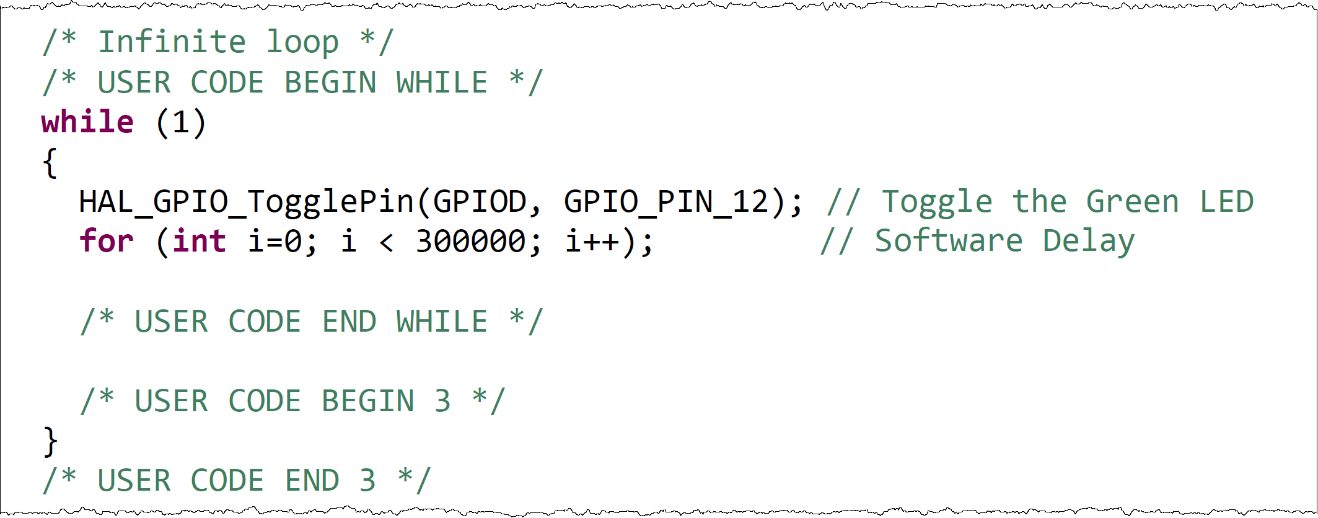
1. Within the **Project Explorer** tab, expand the *Core* folder then the *Src* folder. Open the *main.c* file to display the source code that was generated by the **Device Configuration Tool**. This file is over 400 lines long and contains the code required to initialize the hardware on the 32F411EDISCOVERY board based on the hardware option selected in the **Device Configuration Tool***.*



1. Shown below is the main() section of main.c. Note the highlighted comment blocks. To ensure you do not modify any of the hardware configuration code, ONLY place your user code within these regions. Also, if/when you re-run the **Device Configuration Tool** an updated version of main.c will be generated. If you limit your code placement to the highlighted regions, the updated main.c will retain your code, otherwise your code will be overwritten.

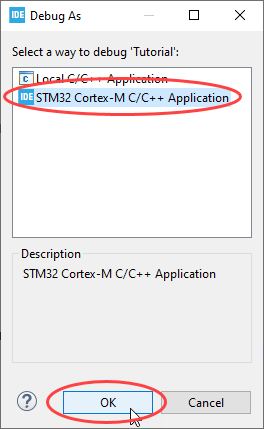


1. Modify the while loop within main.c (lines 108-119) to contain the following code:



* We will use three different delays in these labs.

1. The ***for loop*** above is a simple delay that makes the CPU spend its time counting to some big number. It is often considered the worst type of delay but the fact that it does not use any interrupts or require an RTOS makes it useful later in lab.
2. ***HAL\_Delay(delay)*** is part of the hardware abstraction layer (HAL). The argument delay is the number milliseconds you want to wait. HAL\_Delay uses interrupts and keeps the CPU busy, making it less desirable.
3. ***osDelay(delay)*** is part of the CMSIS-RTOS Library. The argument delay is the number SysTicks (often 1 SysTick = 1 millisecond) you want to wait. This RTOS function will change the status of the *task to blocked* and allow *context switching* to other tasks.
4. After saving the code, compile main.c (**Project → Build All**). If there are any build errors make the necessary modifications, save, and re-build.
5. To download the executable to the 32F411EDISCOVERY board, select **Run → Debug** (F11). You may see the following dialog box. If you do, select the STM32Cortex-M C/C++ Application and select *Yes*.



1. In the **Edit launch configuration properties** window, use all the default settings and then select *OK*. In the **Confirm Perspective Switch** window, select “Remember my decision” and then select *No*. (note: again, this window will only appear the first time you run the software).

|  |  |
| --- | --- |
|  | C:\Users\ghzite\AppData\Local\Temp\SNAGHTML72d5934.PNG |

1. After you received the “*Download verified successfully*” message in the console window, select **Run → Resume** (F8). Your code should now be running on the board.
2. Using the code segment shown below as a guide (note, you’re NOT writing this code), modify main.c to do the following:
   * Initialized the **Red** and **Blue** LEDs on and the **Green** and **Orange** LEDs off.
   * Within the while loop…
     + toggle both the **Red** and **Green** LEDs.
     + execute a software delay { **for** (**int** i=0; i < 300000; i++); }
     + poll the pushbutton switch.
       - output the value of the pushbutton to the **Orange** LED.
       - output the value of the !pushbutton to the **Blue** LED.

