STM32 Cube IDE Tutorial

Version 2225

STM32-Cube IDE initial Start-Up

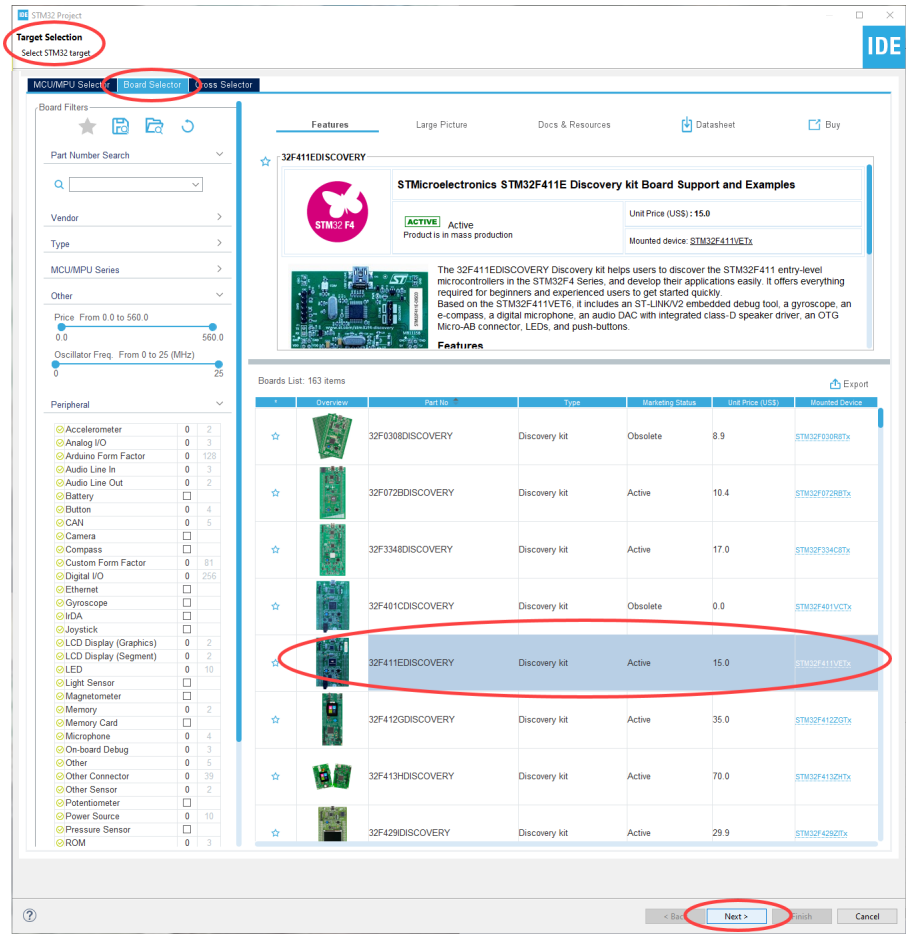
1) Open: 

   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.

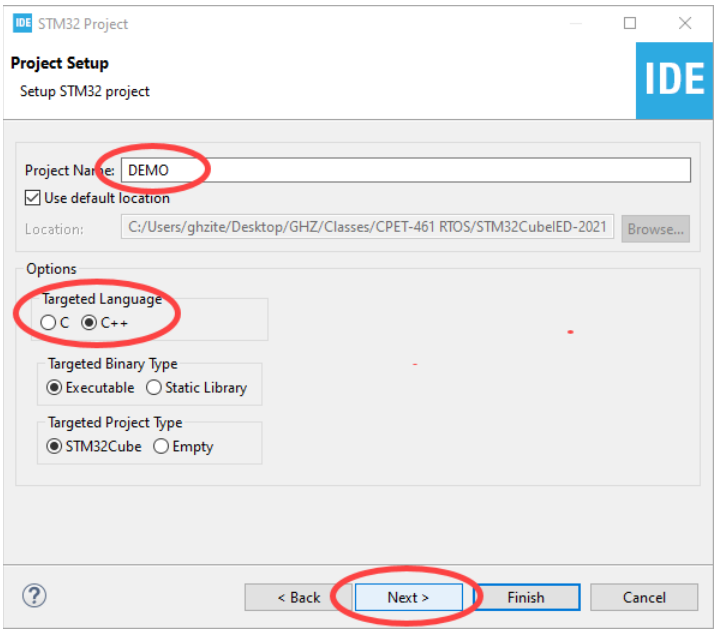
Creating a new Project in STM-Cube IDE

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*.

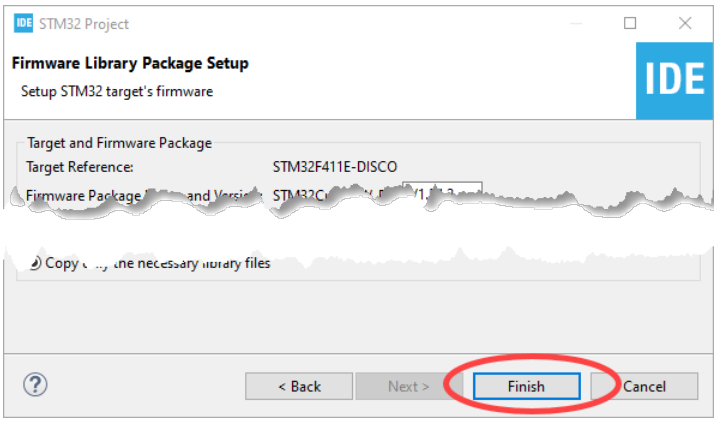


4) In the **Project Setup** window, enter a *Project Name*, select *C++*, and then *Next*.

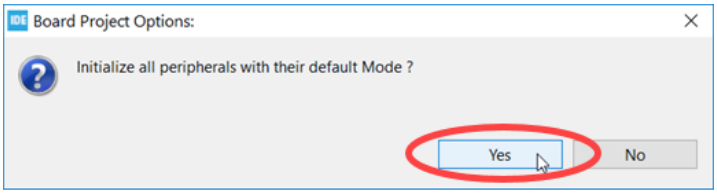
   Make sure the location is the workspace you entered previously. If it isn't, correct the location.



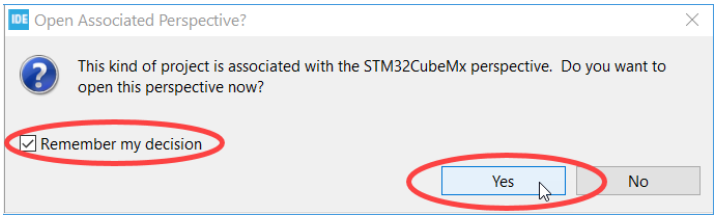
5) In the **Firmware Library Package Setup** window, use all the default settings and then select *Finish*.



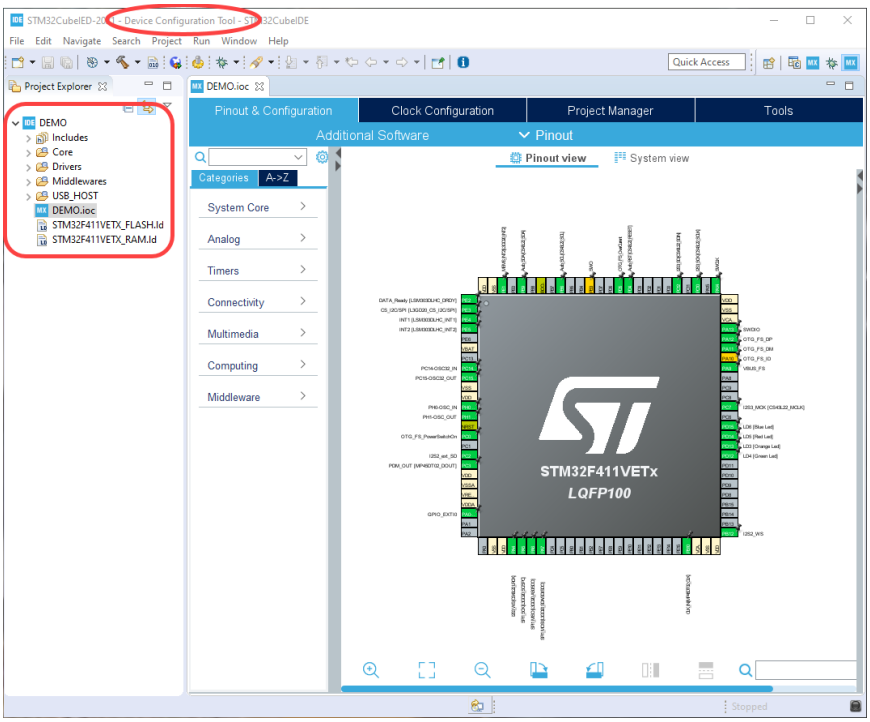
6) Select *Yes*, to initialize all peripherals with their default Mode.



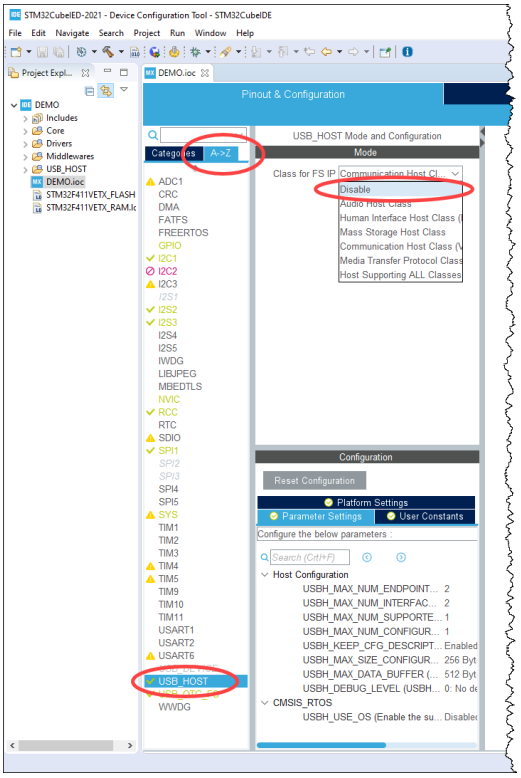
7) Select "Remember my decision" and then select *Yes*. (note: this window will only appear the first time you run the software).

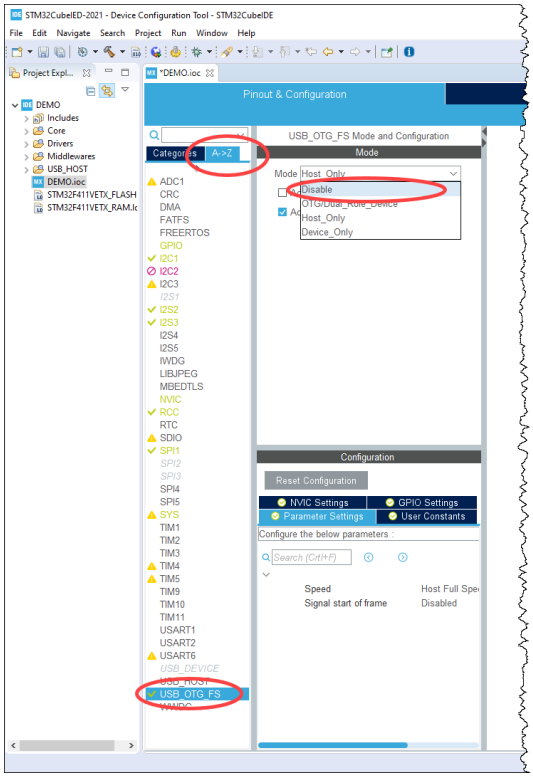


8) Once the project has been created, the **Device Configuration Tool** window will display the MCU's pinout. For your designs, the pinout will be based on the layout of the 32F411EDISCOVERY board. This new project, DEMO, is listed under the **Project Explorer** tab where the IDE icon next to the project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

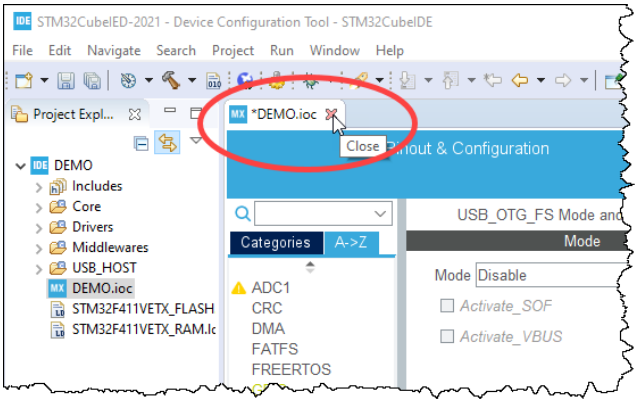


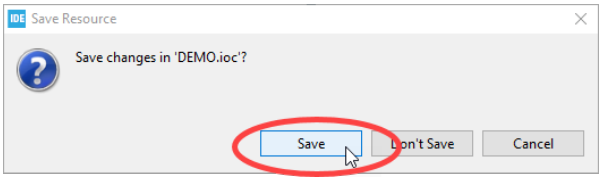
9) With two exceptions, this design will use the default configuration. The two changes required are to disable **USB_HOST** and **USB_OTG_FS**. To do so, in the **Pinout & Configuration Window**, select **A-Z**, **USB_HOST**, and **Disable** (from the dropdown window). Repeat this for **USB_OTG_FS**.



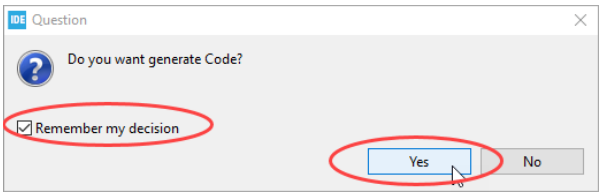


10) To save the hardware configuration and to create the source code files, select the X to *close* the **Device Configuration Tool**.

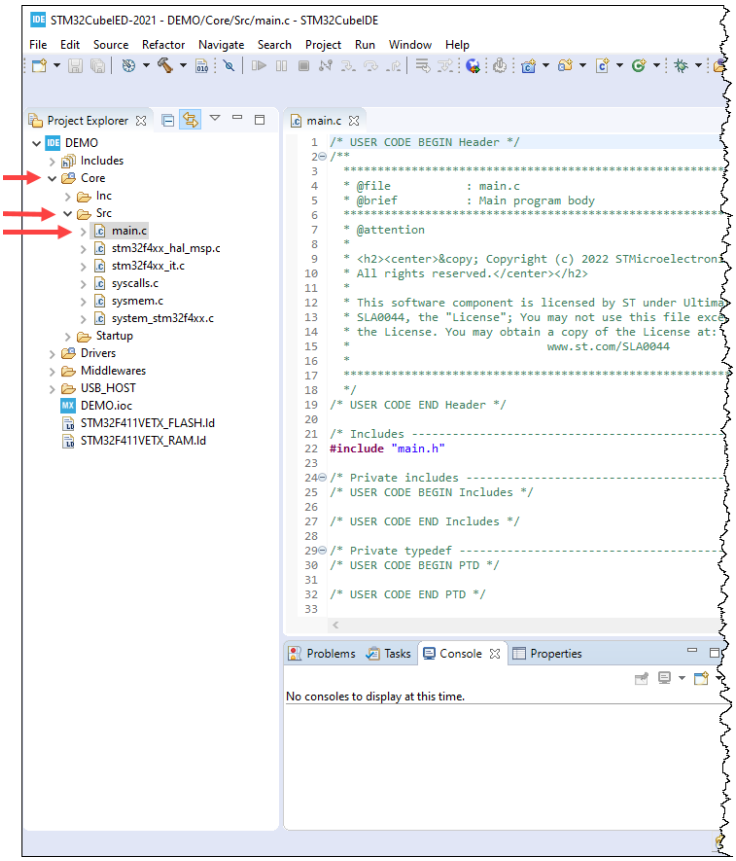




11) Select “Remember my decision” and then select *Yes*. (note: again, this window will only appear the first time you run the software).



12) 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**.



13) 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 <u>your code will be overwritten</u>.

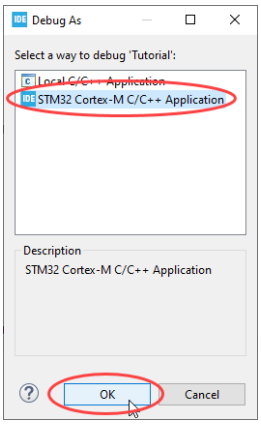
```
74    */
75  int main(void)
76  {
77    /* USER CODE BEGIN 1 */
78
79    /* USER CODE END 1 */
80
81
82    /* MCU Configuration--------------------------------------------------------*/
83
84    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
85    HAL_Init();
86
87    /* USER CODE BEGIN Init */
88
89    /* USER CODE END Init */
90
91    /* Configure the system clock */
92    SystemClock_Config();
93
94    /* USER CODE BEGIN SysInit */
95
96    /* USER CODE END SysInit */
97
98    /* Initialize all configured peripherals */
99    MX_GPIO_Init();
100   MX_I2C1_Init();
101   MX_I2S2_Init();
102   MX_I2S3_Init();
103   MX_SPI1_Init();
104   /* USER CODE BEGIN 2 */
105
106   /* USER CODE END 2 */
107
108   /* Infinite loop */
109   /* USER CODE BEGIN WHILE */
110   while (1)
111   {
112     /* USER CODE END WHILE */
113
114     /* USER CODE BEGIN 3 */
115   }
116   /* USER CODE END 3 */
117 }
118
119 /**
120   * @brief System Clock Configuration
121   * @retval None
```

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

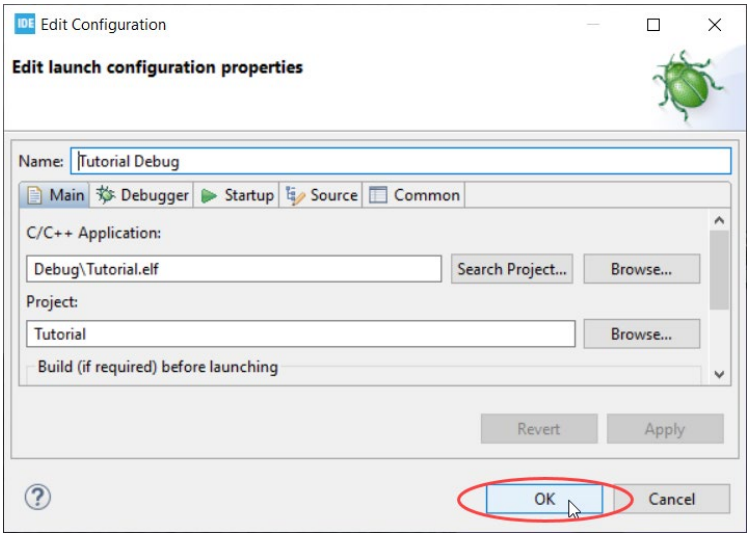
```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12); // Toggle the Green LED
  for (int i=0; i < 300000; i++);         // Software Delay

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

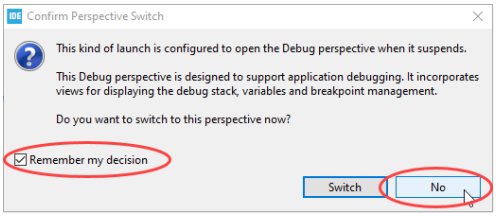
15) After saving the code, compile main.c (**Project → Build All**). If there are any build errors make the necessary modifications, save, and re-build.

16) 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*.



17) 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).





18) 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.

19) 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.

```
// Pushbutton is Port; A Pin-0
// Green LED is Port D; Pin-12
// Orange LED is Port D; Pin-13
// Red LED is Port D; Pin-14
// Blue LED is Port D; Pin-15

// Writes a logic (1) to Port D Pin_15: Turns Blue LED On
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET);

// Writes a logic (0) to Port D Pin_14 : Turns Red LED Off
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);

// Declares a variable Push_Button of type GPIO_PinState
GPIO_PinState Push_Button;

// Read the Port A Pin_0 A and assigns it to variable Push_Button
Push_Button = HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_0);

// Writes the variable Push_Button to Port D Pin_13
// Orange LED will be on if Push-Button is was pressed
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, Push_Button);
```