EXPERIMENT 2: GENERAL-PURPOSE INPUT/OUTPUT (GPIO)

Objectives

The objectives of Experiment 1 are

- to learn tools/environment for STM32F4 microcontroller programme and architecture
- to use Driving GPIO functions (HAL_GPIO_WritePin, HAL_GPIO_ReadPin, HAL_GPIO_TogglePin) and GPIO Output Data Register (ODR), GPIO Input Data Register (IDR)

Apparatus Required:

- STM32CubeMx
- Keil µVision (MDK ARM)
- STM32 ST-Link Utility
- STM32F4 Microcontroller
- STM32F4 Reference Manual
- STM32F4 User Manual

Preliminary Work:

1. Install required programmes (STM32CubeMx, Keil µVision (MDK ARM), STM32 ST-Link Utility)
   STM32CubeMx-----> https://www.st.com/en/development-tools/stm32cubemx.html
   ST-Link Utility-----> https://www.st.com/en/development-tools/stsw-link004.html
   Keil µVision-------->https://www.keil.com/download/product/

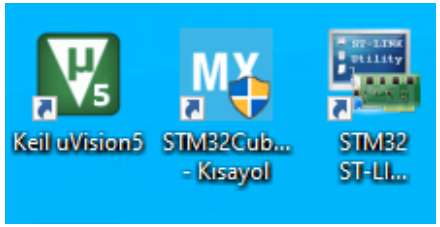


Figure 1

2. Study the GPIO (lecture 4) notes
3. Write the codes of the experimental work in Keil µVision at home.

Experimental Work:

1. First, open the CubeMx program. Select “Embedded software packages” from the Help menu (Figure 2).

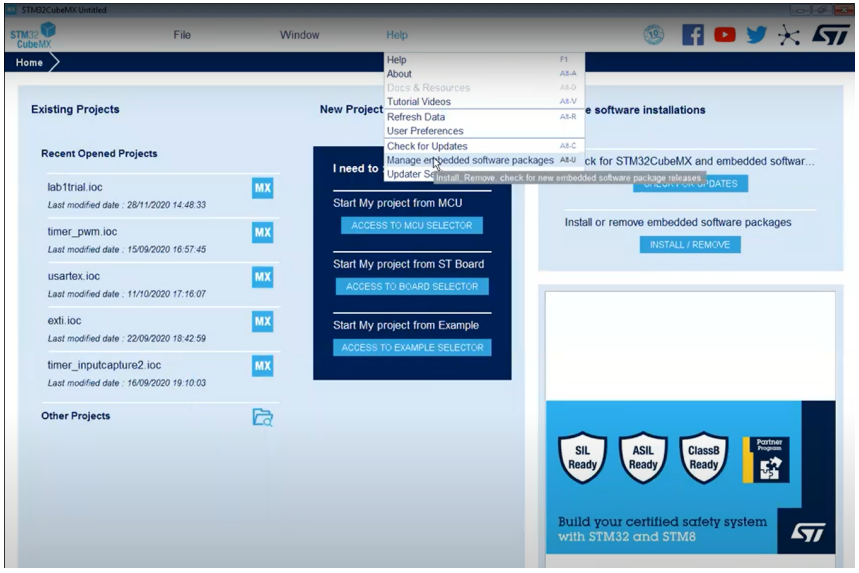


Figure 2

2. According to the microcontroller we have, the relevant package is selected and installed (Figure 3).

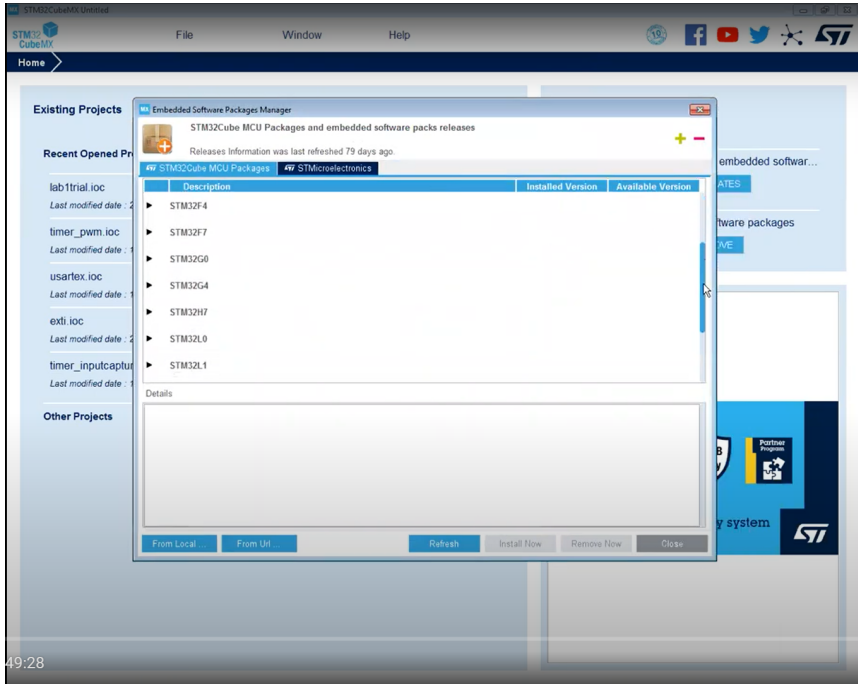


Figure 3

3. To prevent it from updating all the time, settings are made as seen in Figure 4 and Figure 5.

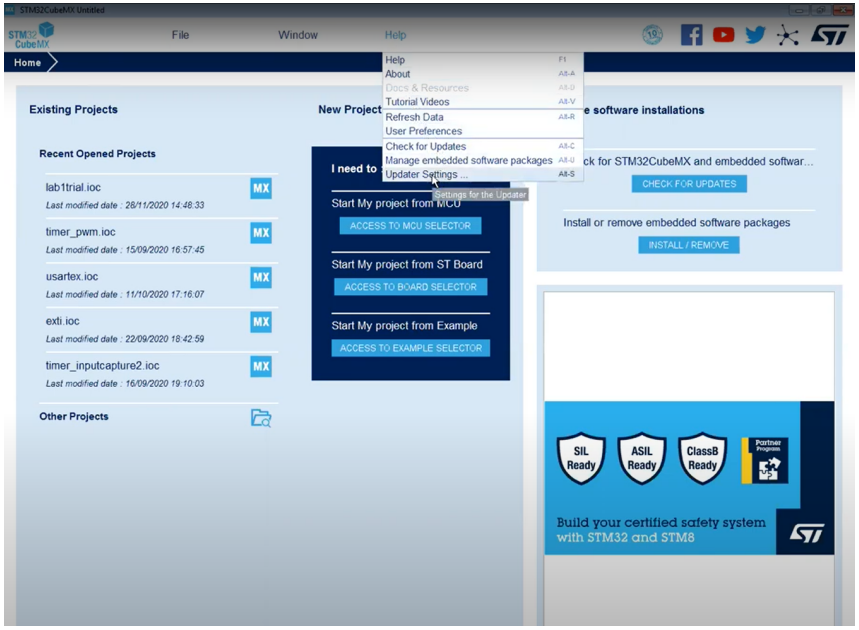

STM32CubeMX Untitled
File
Window
Help
Home
Help
About
Docs & Resources
Tutorial Videos
Refresh Data
User Preferences
Check for Updates
Manage embedded software packages
Updater Settings ...
Settings for the Updater
Existing Projects
Recent Opened Projects
lab1trial.ioc
Last modified date : 28/11/2020 14:48:33
timer_pwm.ioc
Last modified date : 15/09/2020 16:57:45
usartex.ioc
Last modified date : 11/10/2020 17:16:07
exti.ioc
Last modified date : 22/09/2020 18:42:59
timer_inputcapture2.ioc
Last modified date : 16/09/2020 19:10:03
Other Projects
New Project
I need to
Start My project from MCU
ACCESS TO MCU SELECTOR
Start My project from ST Board
ACCESS TO BOARD SELECTOR
Start My project from Example
ACCESS TO EXAMPLE SELECTOR
software installations
ck for STM32CubeMX and embedded softwar...
CHECK FOR UPDATES
Install or remove embedded software packages
INSTALL / REMOVE
SIL Ready
ASIL Ready
ClassB Ready
Build your certified safety system
with STM32 and STM8

Figure 4

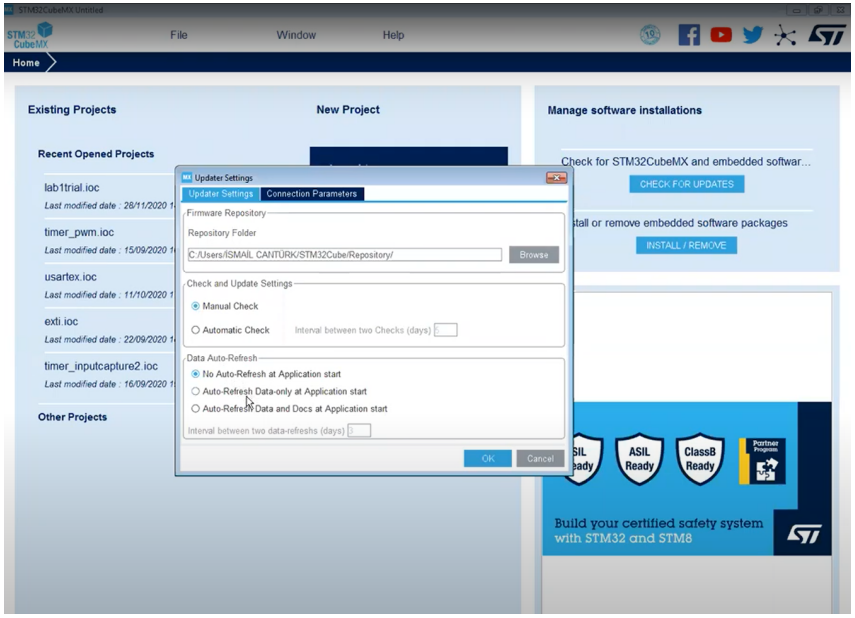


Figure 5

4. Now we can create a new project file to start programming. Select “New Project” from the File menu (Figure 6).

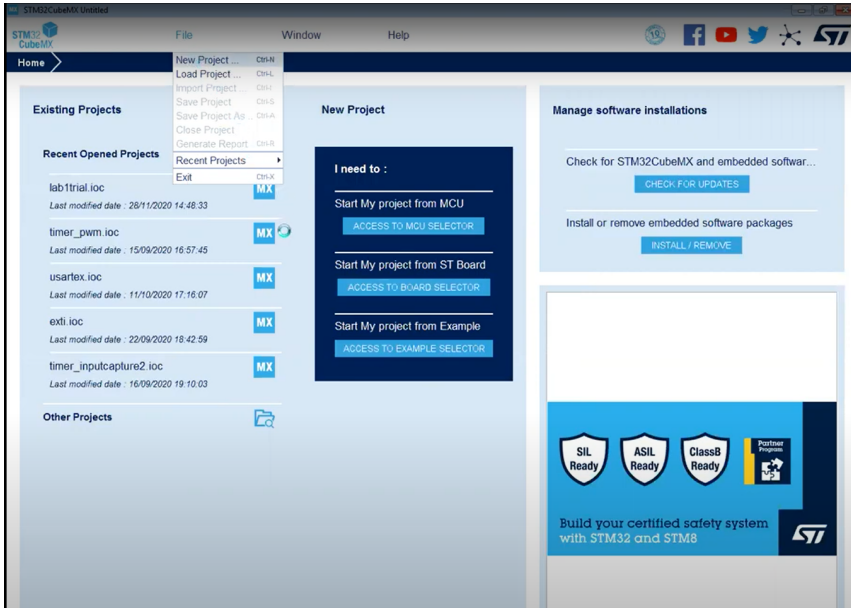


Figure 6

5. The relevant microcontroller is found and selected from the Part Number as in Figure 7.

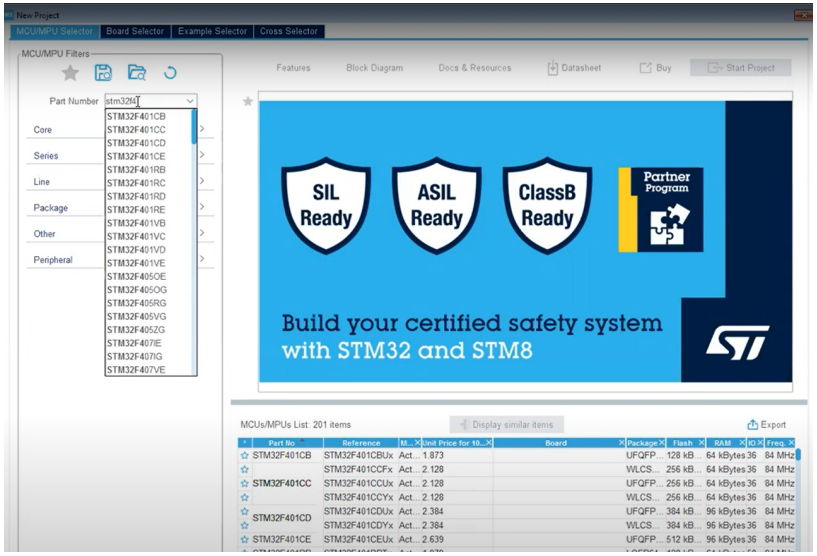


Figure 7

6. After choosing our STM32F4 discovery card, choose “Start Project” from the top menu.

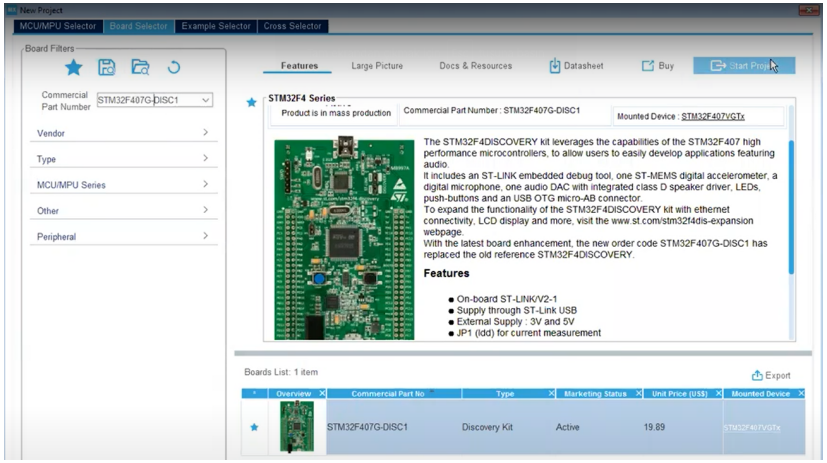


Figure 8

7. You will see a screen like Figure 9. Here we see the Microcontroller we will use. We can use it this way. However, in terms of power consumption and more stable operation, it will be better if we turn off the pins that we will not use in the lab. The green ones show the open pins and the gray ones show the reset state ones. In this lab, we will basically use push pull buttons and leds. So we can turn off the others. Which of these pins are can be understood by looking at the user manual. We should not close the pins that provide the connection between the microcontroller and the programmer. You can check which of these pins are in the user manual (Figure 10). We can do the closing and defining tasks of a pin by left-clicking on the pin. You completed the Pinout Configuration.

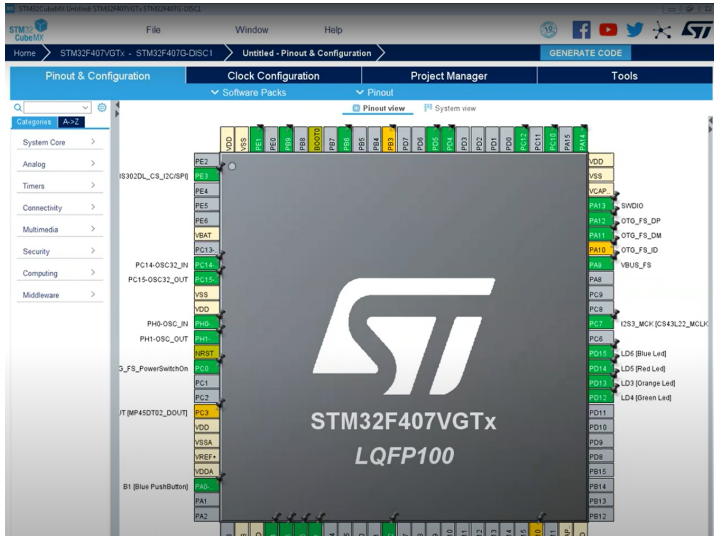
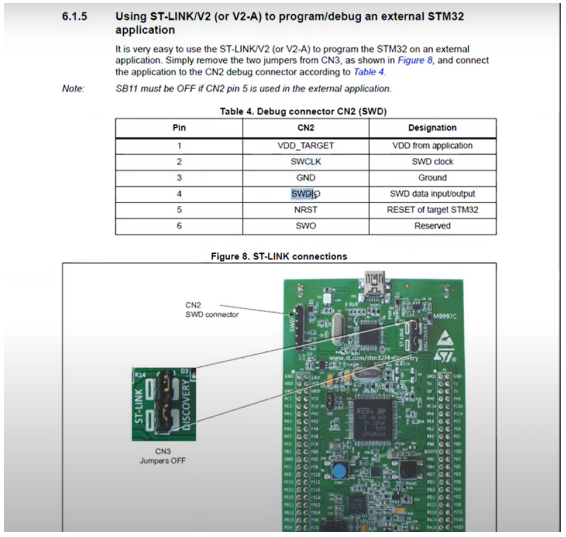


Figure 9



6.1.5 **Using ST-LINK/V2 (or V2-A) to program/debug an external STM32 application**

It is very easy to use the ST-LINK/V2 (or V2-A) to program the STM32 on an external application. Simply remove the two jumpers from CN3, as shown in *Figure 8*, and connect the application to the CN2 debug connector according to *Table 4*.

Note: SB11 must be OFF if CN2 pin 5 is used in the external application.

Table 4. Debug connector CN2 (SWD)

Pin	CN2	Designation
1	VDD_TARGET	VDD from application
2	SWCLK	SWD clock
3	GND	Ground
4	SWDIO	SWD data input/output
5	NRST	RESET of target STM32
6	SWO	Reserved

Figure 8. ST-LINK connections

Figure 10

8. There is nothing we can change in the Clock Configuration section for this experiment. We come to the Project Manager menu and set the fields there as in Figure 11. Then you will select the Generate Code and now our template file is out. Select “Open Project” from the opened screen (Figure 12). Keil µVision will open.

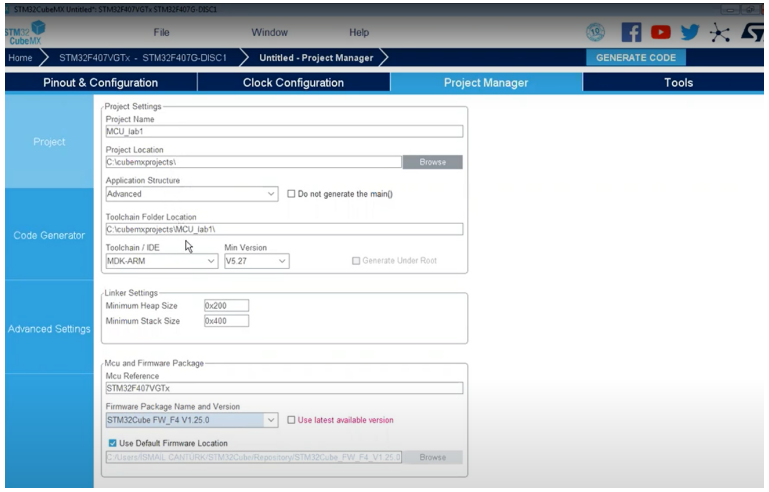


Figure 11

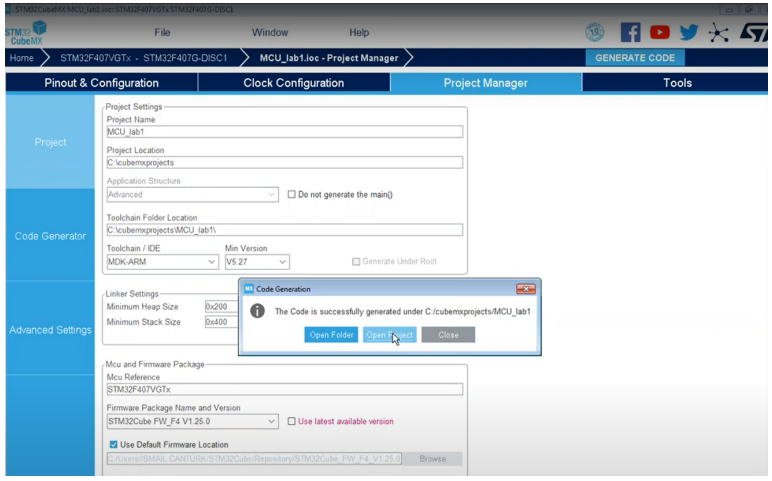


Figure 12

9. First, we need to come to the Pack Installer (Figure 13) and install the package of the relevant microcontroller, as we did in CubeMx. The relevant microcontroller is selected and Install is selected for the required package (Figure 14).

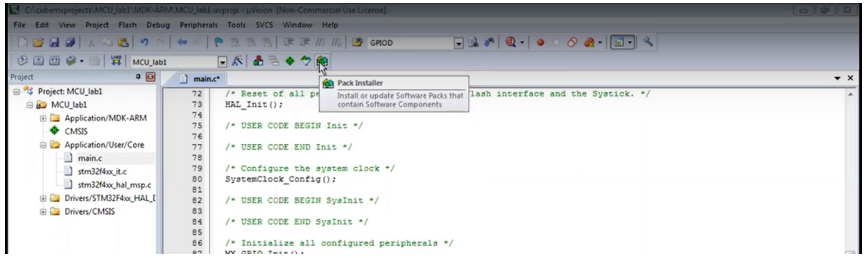


Figure 13

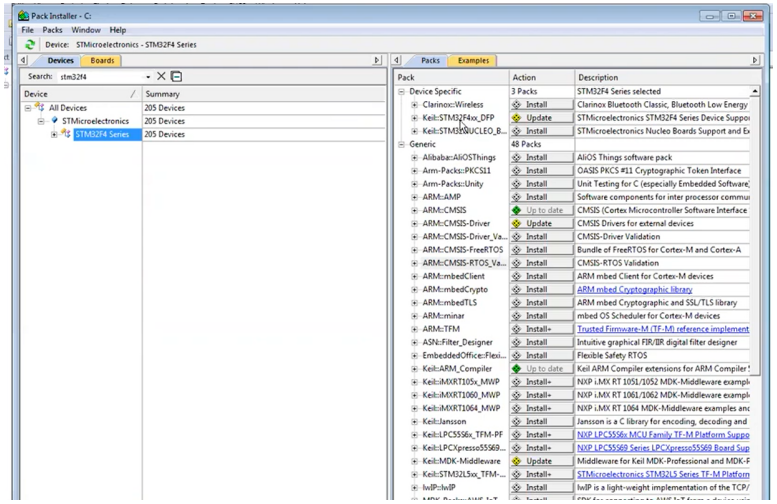


Figure 14

10. Select “Options for Target” menu (Figure 15). Come to the “Debug” menu and select “ST Link Debugger” and Settings respectively (Figure 16). Then we choose “Reset and Run” from “Flash Download" so that when we run our code, it will reset its previous

information. If we do not select it, we have to press the reset button for the code to run (Figure 17). When we go back to the previous menu, we choose the “C++” menu and select the “Level 0” for Optimization (Figure 18).

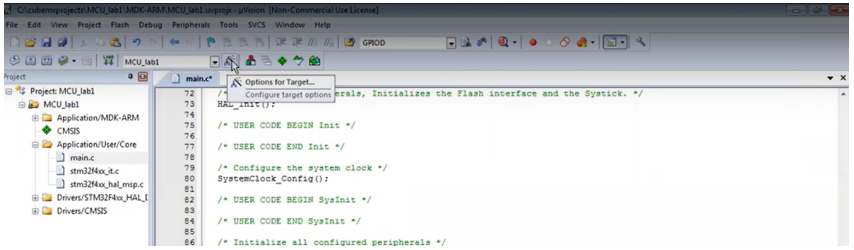


Figure 15

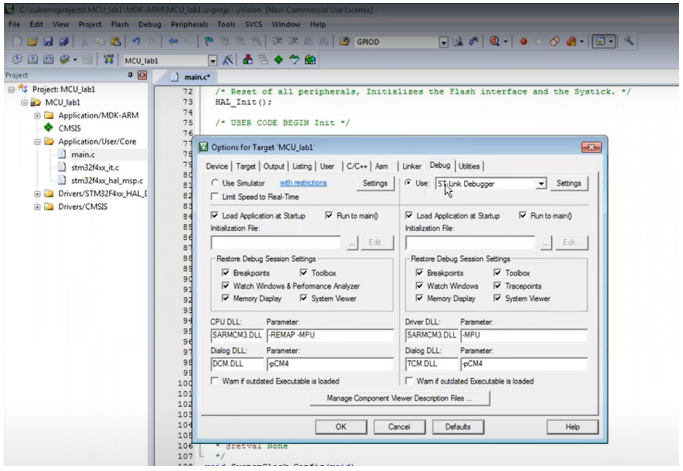


Figure 16

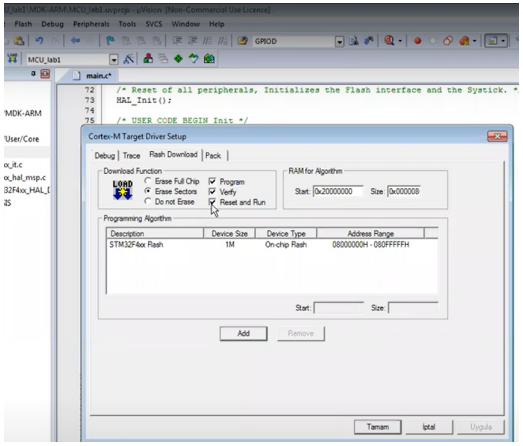


Figure 17

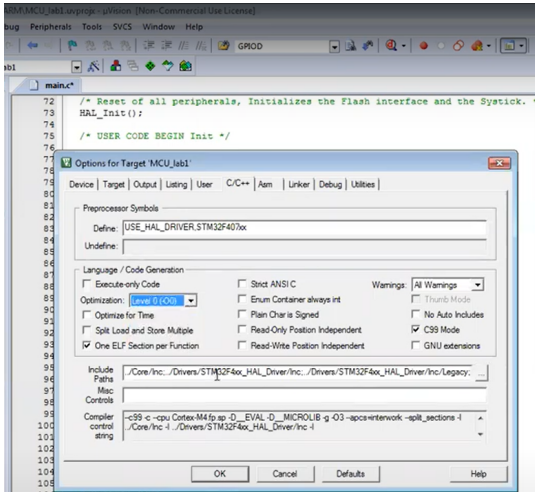


Figure 18

11. Now we can start to write our code. The “main.c” file on the left is the file created for us in Keil µVision. Here, we can write various codes using the C language.

    Follow the steps below for each code you want to run.

- Write the relevant codes in the part reserved for the user in the while loop.
- After writing the codes, click the build button to create the hex file and other files that will be uploaded to the microcontroller.
- Click the Load button to load the codes into the microcontroller.

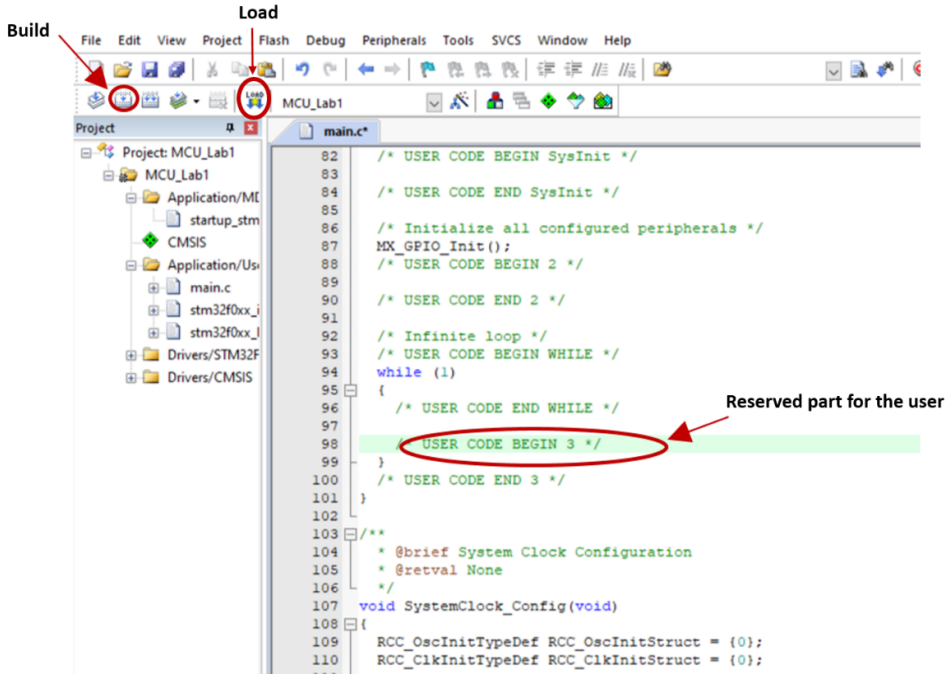


Figure 19

1. Use the HAL_GPIO_TogglePin function. Write the following code. Then write the same code without using the HAL_Delay function. Explain what you will see with reasons?

```
//Toggle the LED connected to the C8 pin at half-second intervals.
HAL_GPIO_TogglePin(GPIOD,GPIO_PIN_12); // to toggle led which is connected to the D12 pin.
HAL_Delay(500); //Wait 500 ms
```

2. Use the HAL_GPIO_WritePin function. Write the following code.

```
// Toggle the LED at 2 second intervals
HAL_GPIO_WritePin (GPIOD, GPIO_PIN_14, GPIO_PIN_SET); //Write logic 1 to the output data register of the pin
HAL_Delay(2000); //Wait 2 s
HAL_GPIO_WritePin (GPIOD, GPIO_PIN_14, GPIO_PIN_RESET); //Write logic 0 to the output data register of the pin
HAL_Delay(2000); //Wait 2 s
```

3. Use the Output Data Register (ODR) directly

```
// Use Output Data register directly
GPIOD->ODR|=0xF000; // Turn on the leds connected to the D12,D13,D14&D15 pins.
```

4. Toggle the LEDs using ODR

```
// Use Output Data register directly to do toggle leds
GPIOD->ODR|=0xF000; // Turn on the leds connected to the D12,D13,D14&D15 pins.
HAL_Delay(2000); //Wait 2 s
GPIOD->ODR&=0x0000; // Turn off the leds connected to the D12,D13,D14&D15 pins.
HAL_Delay(2000); //Wait 2 s
```

5. Floating light respectively using ODR

```
//bitwise shifting
GPIOD->ODR|=0xF000; //All of the leds on
HAL_Delay(500); //Wait 500 ms
//Shift the bits right
```

```
for (i=1;i<5;i++)
{
  GPIOD->ODR>>=1; //Shift the bits right
  HAL_Delay(500); //Wait 500 ms

}
```

6. Floating light respectively using ODR

```
//bitwise shifting
 GPIOD->ODR=0x0F00; //All of the leds off (assign bit values)
 HAL_Delay(500); //Wait 500 ms
 //Shift the bits left
 for (i=1;i<5;i++)
     {
       GPIOD->ODR<<=1; //Shift the bits left
       HAL_Delay(500); //Wait 500 ms
      }
```

7. The program that toggle the LEDs when we push the button using IDR.

```
if (GPIOA->IDR&0x0001) //Checking if the button is pushed
{
 HAL_GPIO_TogglePin(GPIOD,GPIO_PIN_12); // to toggle led which is connected to the D12 pin.
 HAL_Delay(200); //Wait 200 ms
}
```