

# Lab 3: DMA & PWR

## Activity 1: DMA Configuration

### Aim:

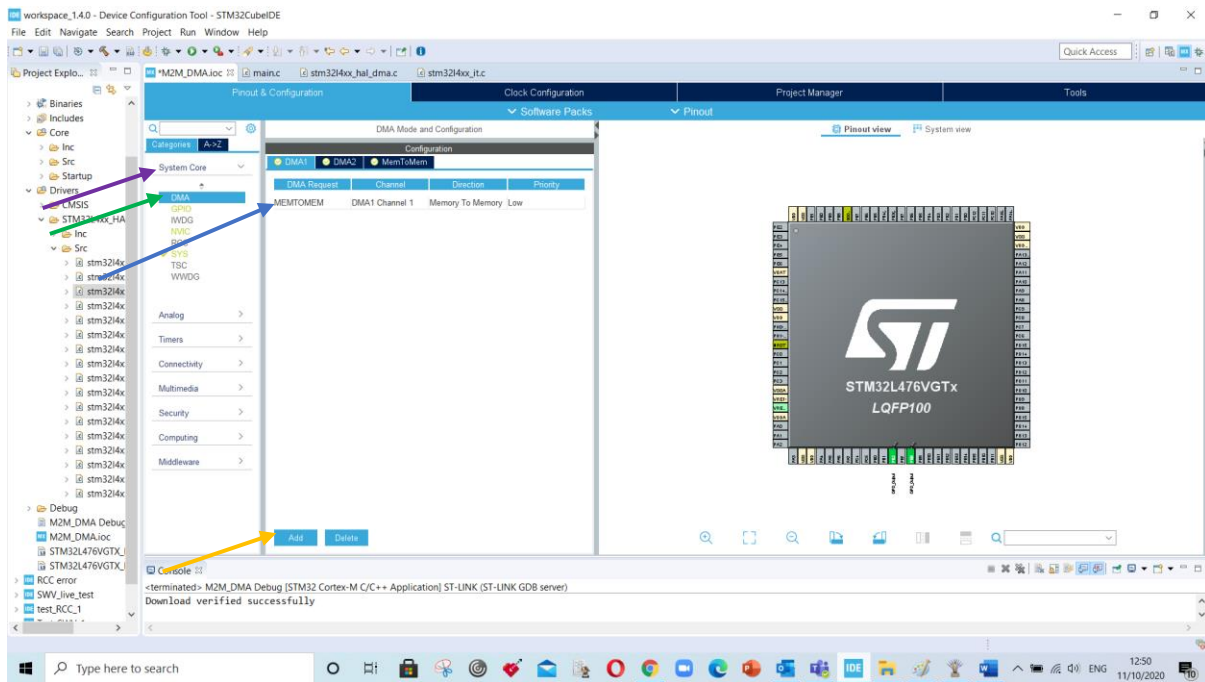
To configure a DMA channel for a memory-to-memory transfer.

### Objectives:

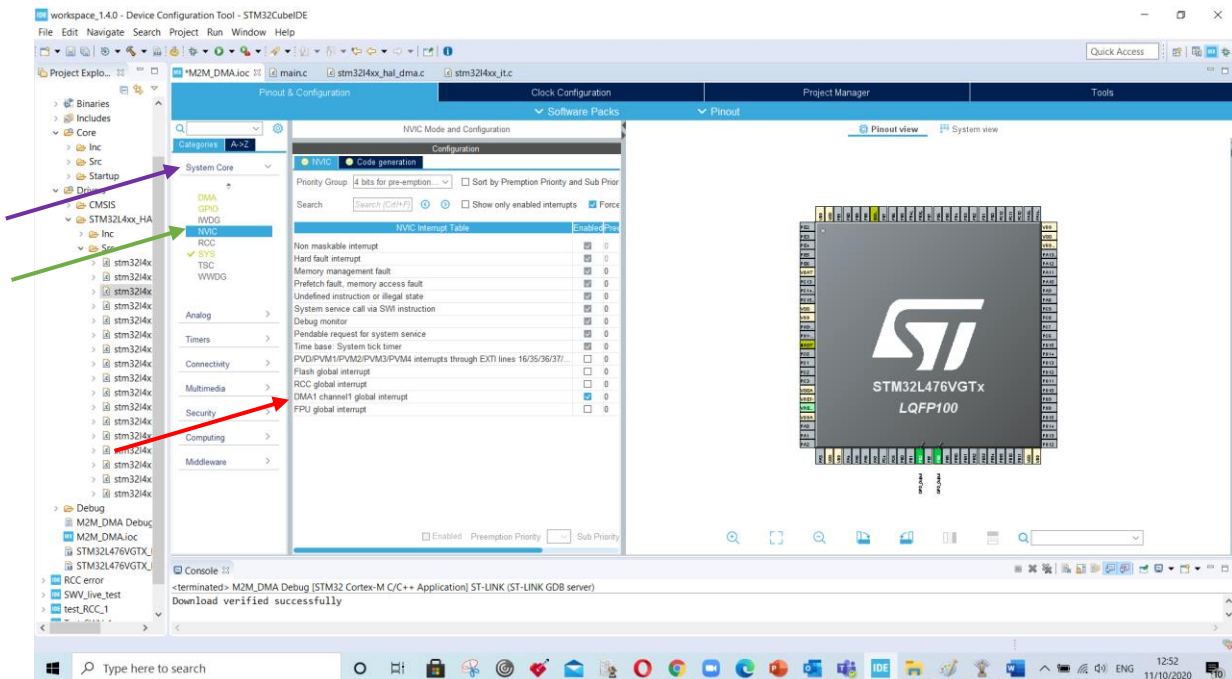
- 1- Learn how to configure DMA channel for a memory-to-memory transfer using NVIC interrupt.
- 2- Using logic analyzer to measure the CPU unloading and DMA transfer efficiency.
- 3- Developing understanding for various errors.

**Step 1:** Create a project in STM32CubeIDE and configure GPIO 'PE8' and 'PB2' as outputs.

**Step 2:** Go to 'System Core', select 'DMA', click on 'Add' and select 'MEMTOMEM' and use default DMA settings.



**Step 3:** From 'System Core' menu, select 'NVIC' and check the 'DMAv1 Channel 1 global interrupt'.



**Step 4:** Generate the code and for no DMA transfer write the codes on location shown by comments, as below:

```
/* USER CODE BEGIN PTD */
#define M2M_DMA_size 8000
```

---

```
/* USER CODE BEGIN 1 */
uint32_t source[M2M_DMA_size];
uint32_t destination[M2M_DMA_size];
uint32_t i;
```

---


```
/* USER CODE BEGIN 2 */
for (i=0;i<=M2M_DMA_size; i++)
{
    source[i]=i;
}
```

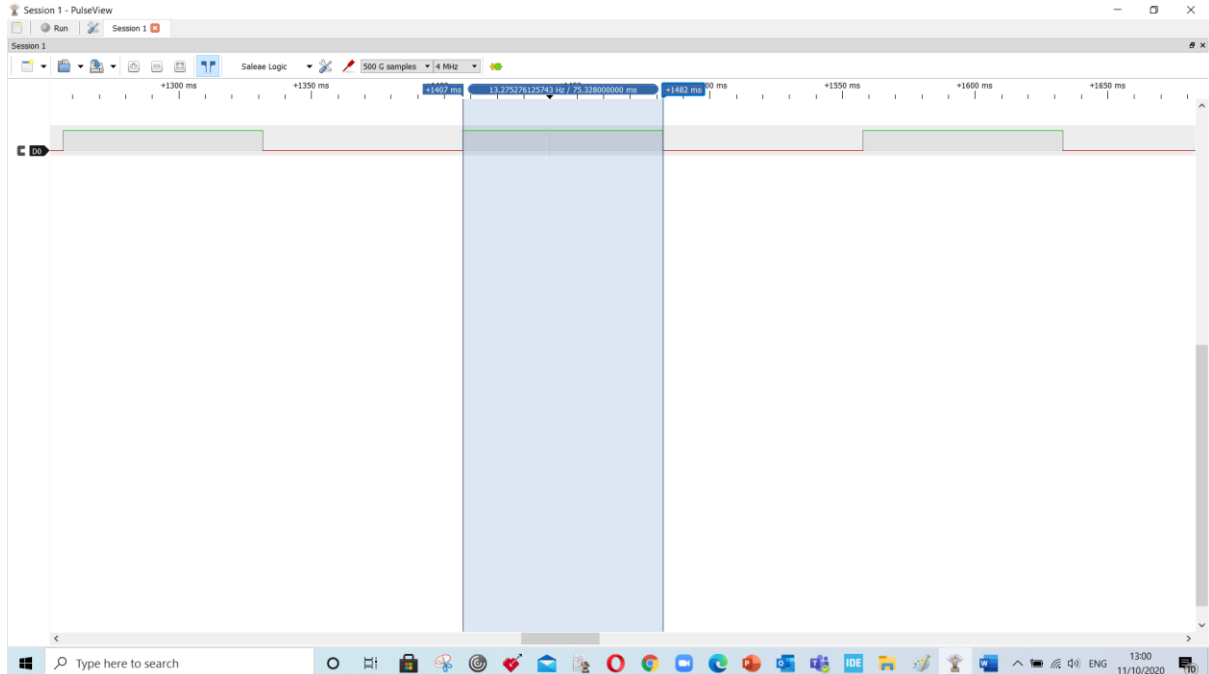
---

```
/* USER CODE BEGIN 3 */
HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_8);

for (i=0;i<=M2M_DMA_size; i++)
{
    destination[i] = source[i];
}
```



**Step 5:** Run ‘’, and build the code. You will not find any error. Now test your code execution by capturing the trace of ‘PE8’ in logic analyzer and measure the time period for data transfer (13Hz approx). Note the Sampling time is 4MHz.



**Step 6 (DMA Transfer):** In generated code files, select folders ‘Core’, then select folder ‘Src’, then select ‘STM32L4xx\_it.c’ and in function ‘DMA1\_channel1\_IRQHandler’ add toggle led code as below:

```
/* USER CODE BEGIN DMA1_Channel1_IRQn 1 */  
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_2);
```

Start the DMA Transfer by enabling the DMA global interrupt. Comment the for-loop transfer code as below:

```
/* USER CODE BEGIN 3 */  
  
HAL_DMA_Start_IT(&hdma_memtomem_dma1_channel1, source, destination, M2M_DMA_size);  
  
/*      HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_8);  
  
    for (i=0;i<=M2M_DMA_size; i++)  
    {  
        destination[i] = source[i];  
    }*/
```

```


191 }
192
193 /**
194  * STM32L4xx Peripheral Interrupt Handlers
195  * Add here the Interrupt Handlers for the used peripherals.
196  * For the available peripheral interrupt handler names,
197  * please refer to the startup file (startup_stm32l4xx.s).
198  */
199
200 /**
201  * @brief This function handles DMA1 channel1 global interrupt.
202  */
203
204 void DMA1_Channel1_IRQHandler(void)
205 {
206     /* USER CODE BEGIN DMA1_Channel1_IRQn 0 */
207
208     HAL_DMA_IRQHandler(&dma_memtomem_dma1_channel1);
209     /* USER CODE BEGIN DMA1_Channel1_IRQn 1 */
210
211     HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_2);
212     /* USER CODE END DMA1_Channel1_IRQn 1 */
213 }
214
215
216
217 /* USER CODE BEGIN 1 */
218
219 /* USER CODE END 1 */
220
221 /**
222  * ***** (C) COPYRIGHT STMicroelectronics *****END OF FILE*****
223  */

```

Build Analyzer: M2M\_DMA.elf - /M2M\_DMA/Debug - Oct 11, 2020 12:59:12 PM

Region	Start address	End address	Size	Free	Used	Usage (%)
RAM	0x20000000	0x20018000	96 KB	94.38 KB	1.62 KB	1.68%
RAM2	0x10000000	0x10008000	32 KB	32 KB	0 B	0.00%
FLASH	0x08000000	0x08100000	1024 KB	1016.88 KB	7.12 KB	0.70%



**Step 7:** Run '  ' and build the code. You will not find any error. Now test your code execution speed by Capturing the trace of 'PB2' in logic analyzer and measure the time period of waveform (40 Hz approx.)

Session 1 - PulseView

Scale: 10 ns

Time: 40 ns

**To do task:**

Try to introduce error and note their responses:

- 1- Define the higher sized of typedef variable 65k

```
/* USER CODE BEGIN PTD */  
#define M2M_DMA_size 65000
```

- 2- Declare 'i' of 8-bit

```
uint8_t i;
```

- 3- Use both for loop and DMA transfer together and observe the waveform
- 4- Optional: Add another DMA channel for memory-to-memory transfer and execute both at same time and observe the execution speed.

## **Activity 2: Low Power mode configuration**

### **Aims:**

Learn how to setup Low power mode on STM32L476VG MCU.

### **Objectives:**

1. Configure GPIO & Generate the Code.
2. Activate the low power mode.
3. Debug the code and investigate Special function register (SFR) for low power setting bits.
4. Configure RCC and activate MCO.

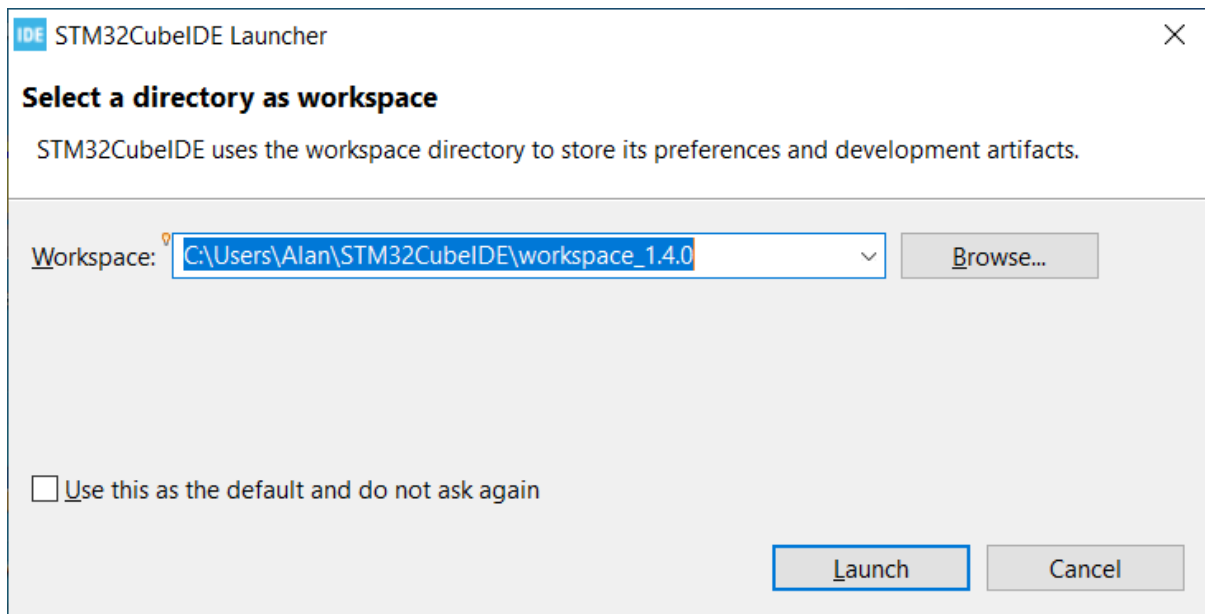
### **Step 1:**



Double click the 'STM32CubeIDE' icon on the desktop.

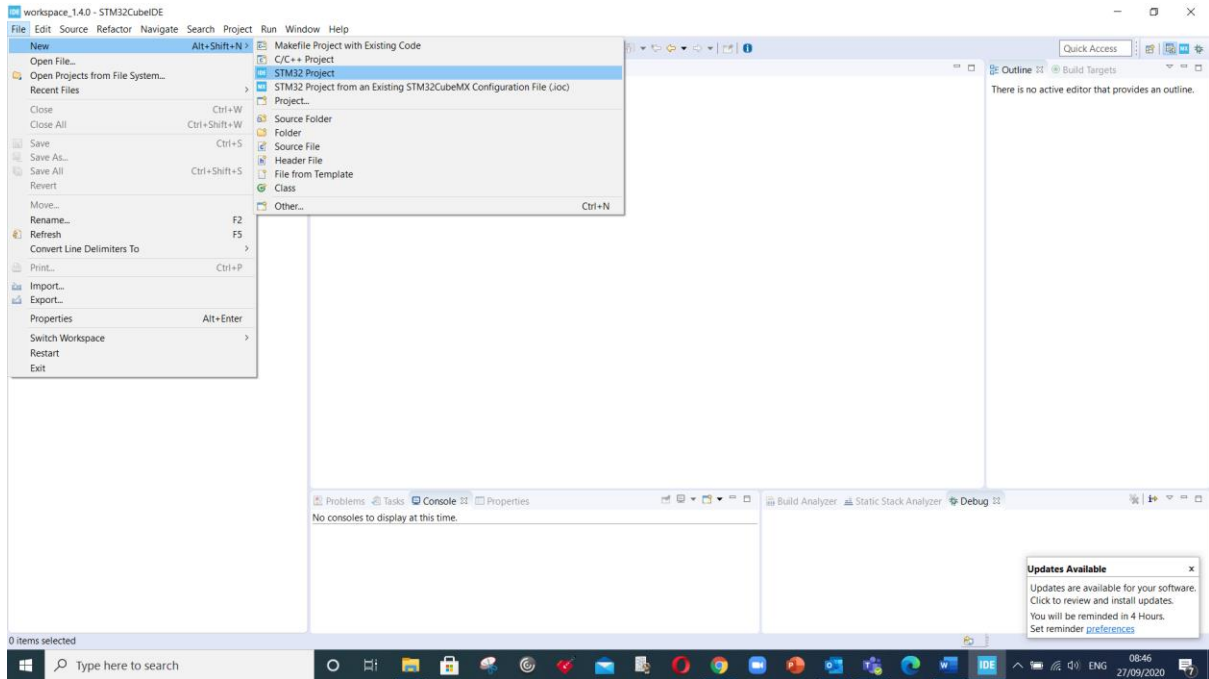
### **Step 2:**

Select the workspace location, Default location is preferred



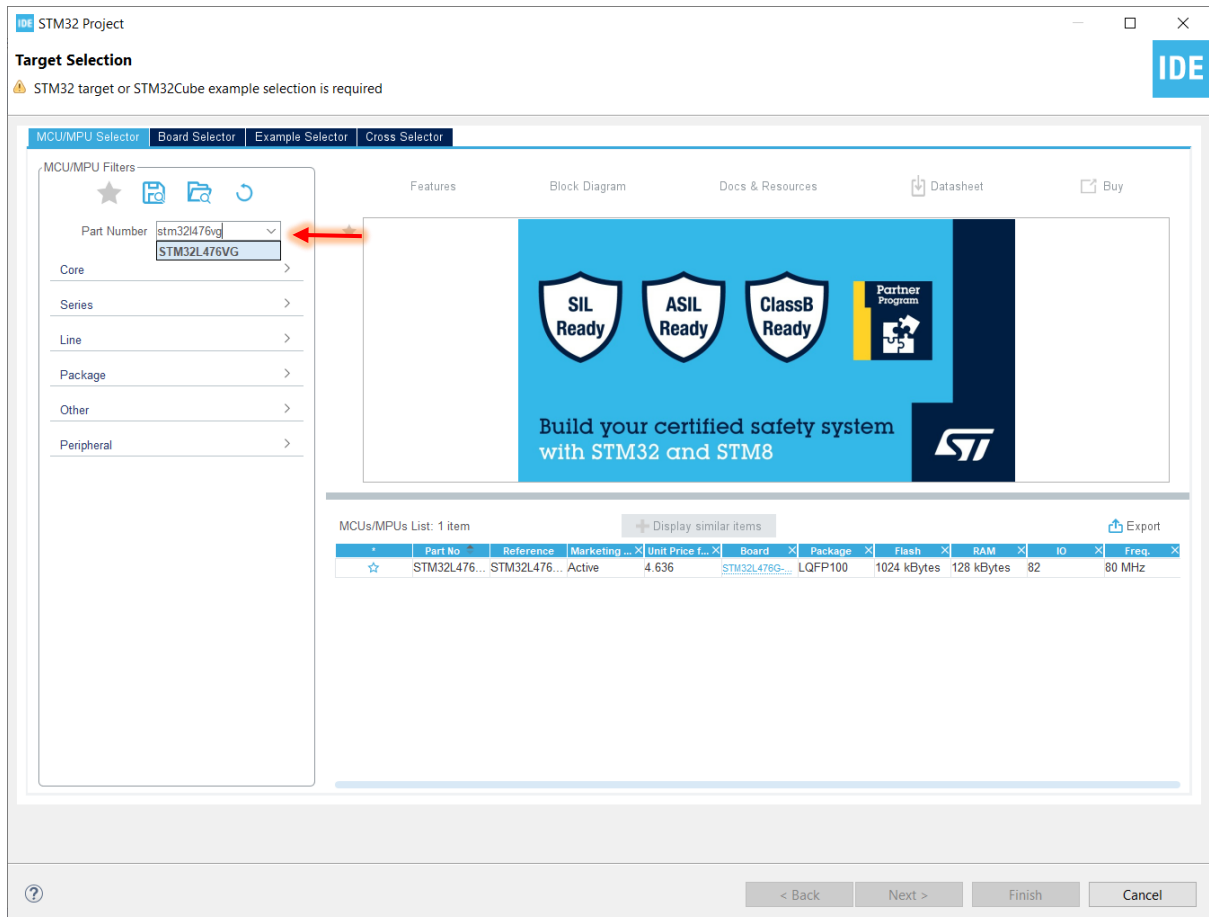
### Step 3:

Open new project: File-> New-> STM32 Project



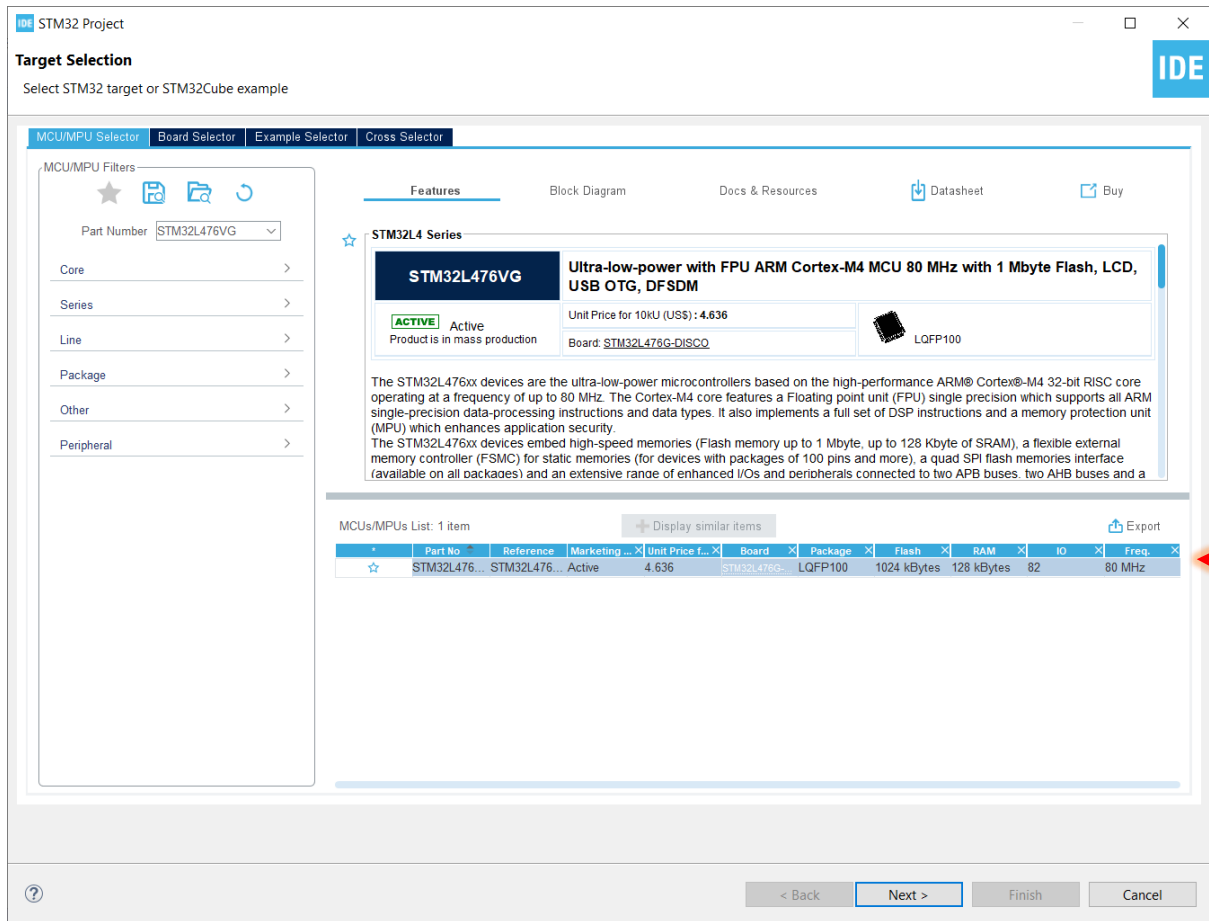
### Step 4:

Type in 'STM32L476VG' in part number area.



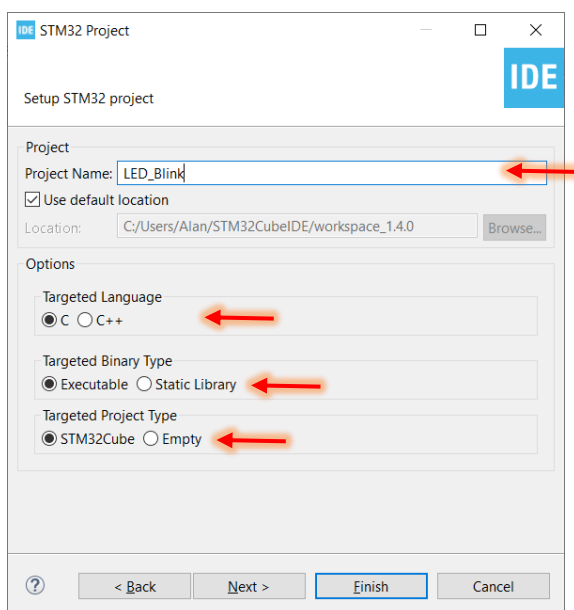
## Step 5:

Select the MCU and Press 'Next'



## Step 6:

Name you project and observer the default options.

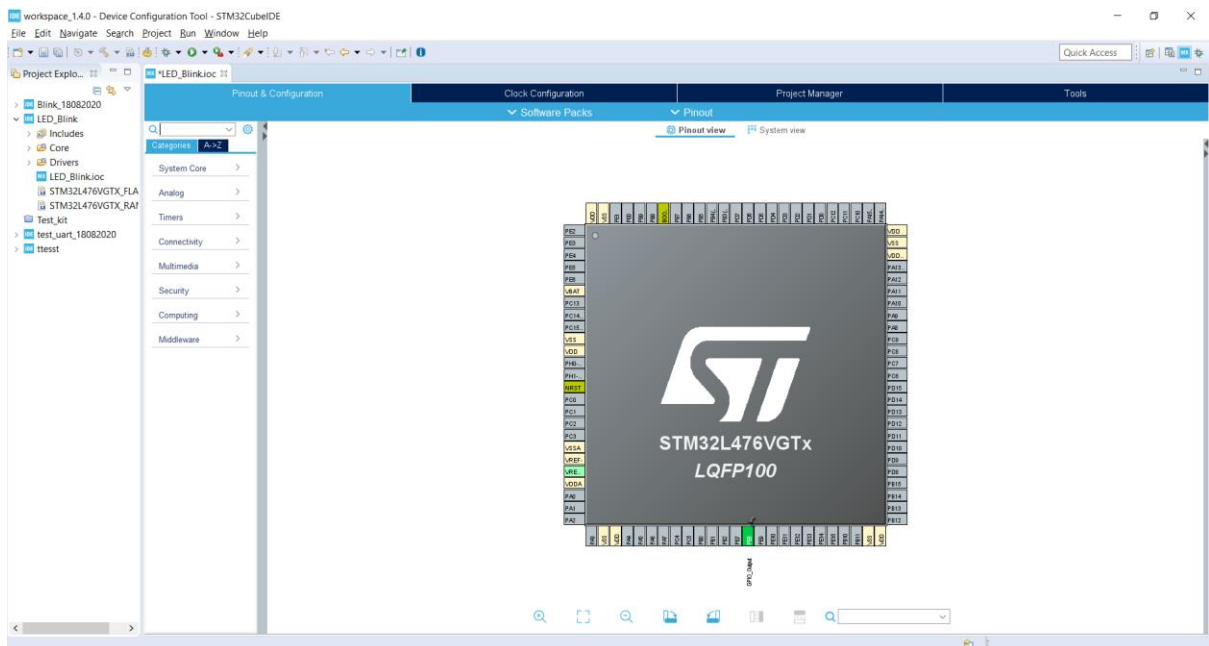
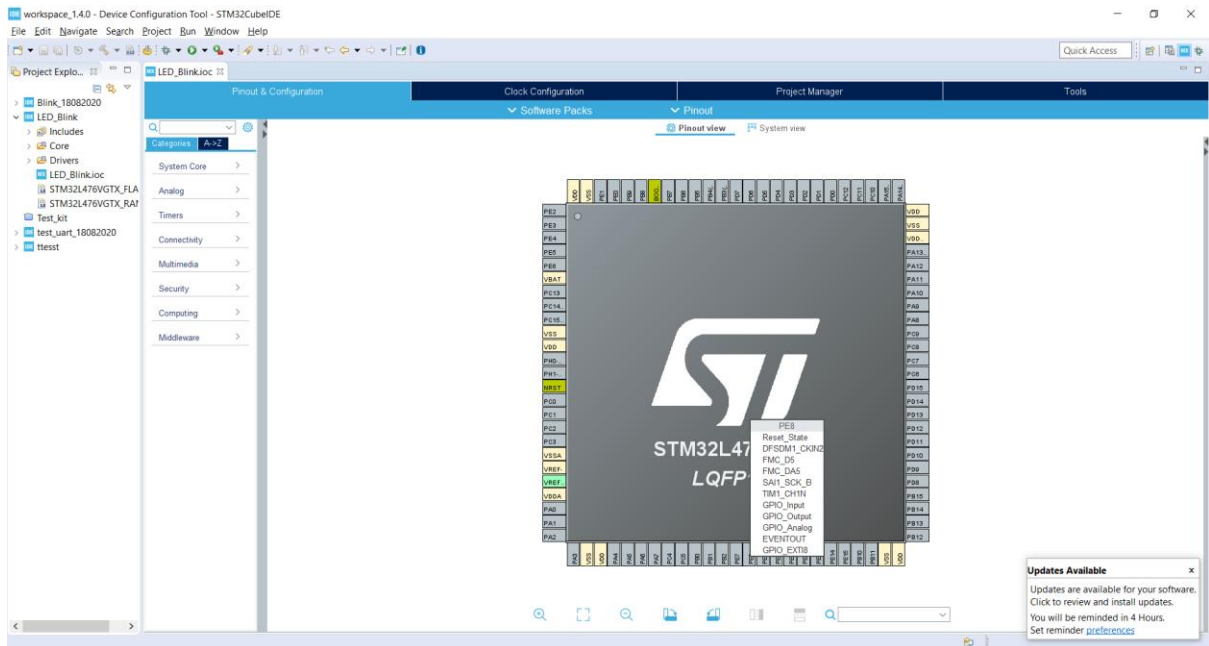




## Step 7:

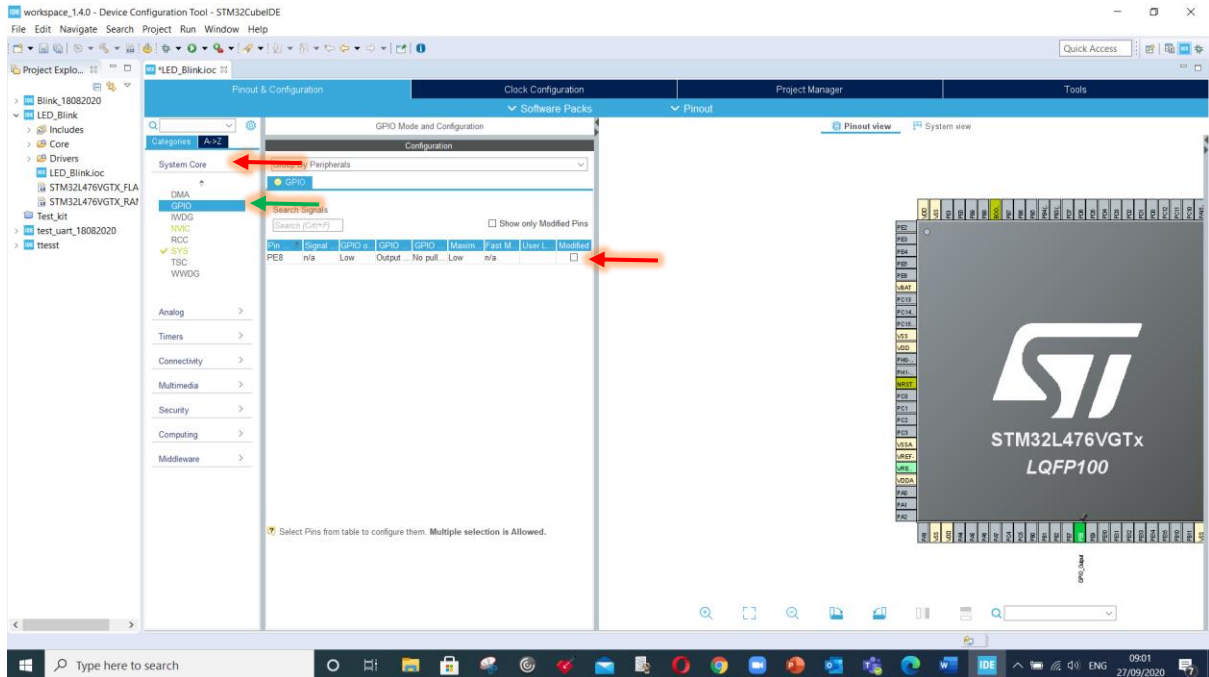
Left click on 'PE8' and select 'GPIO\_Output'.

Note: The PIN PE8 is connected to RED LED on the discovery Kit.

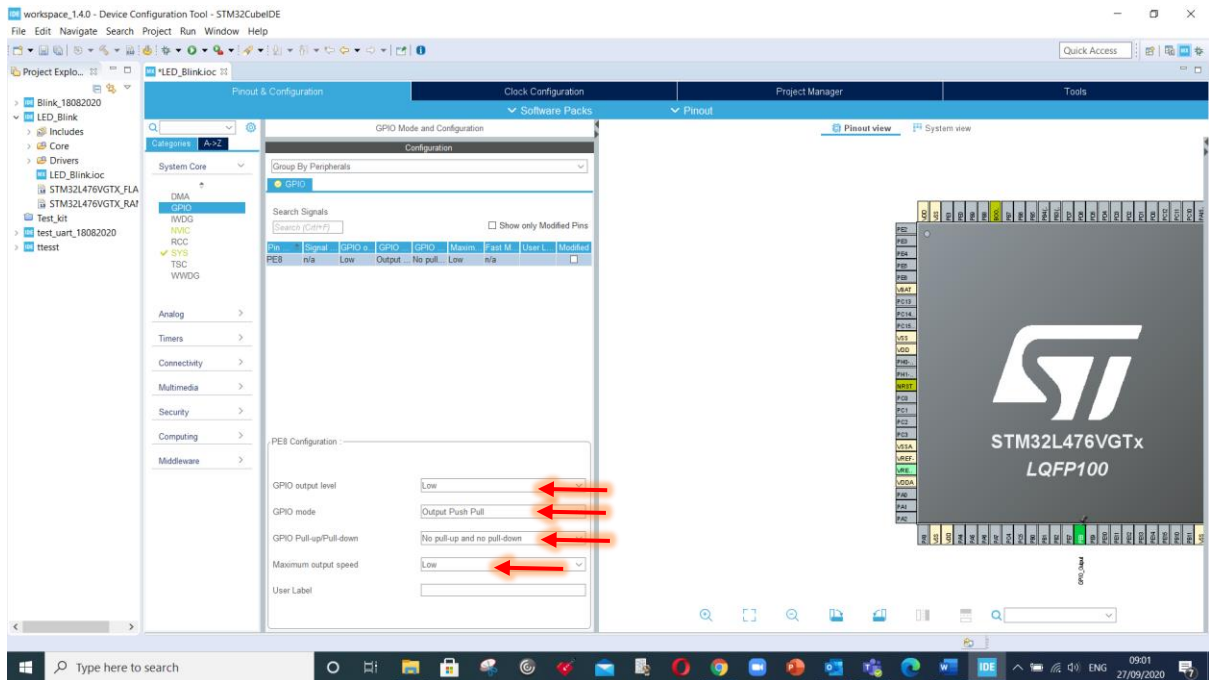


## Step 8:

For GPIO configuration go to 'System Core' and click drop down menu and select 'GPIO'.

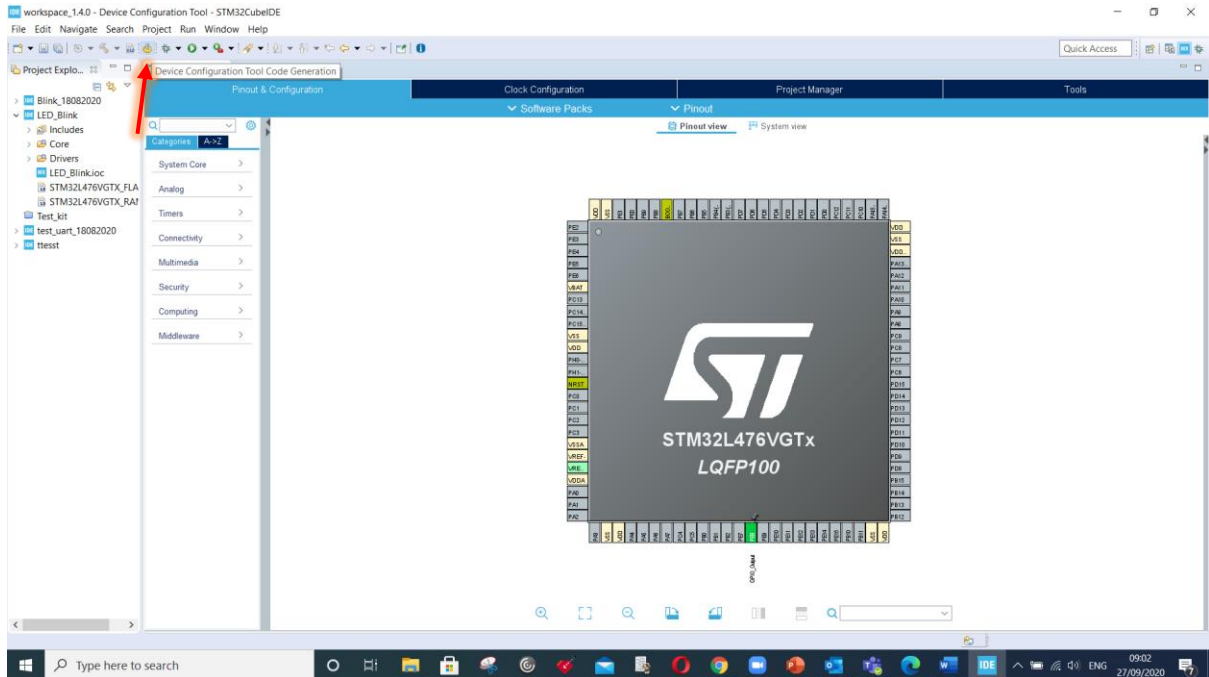


In GPIO menu select 'PE8' then it will the configuration make sure 'GPIO output level' is 'low', 'GPIO mode' is 'Output push pull', 'GPIO Pull-Up/Pull-down' is 'No pullup and pulldown' and 'maximum output speed' is 'Low'.



## Step 9:

Click on 'Project' menu and select 'Generate Code' or use the icon as shown here

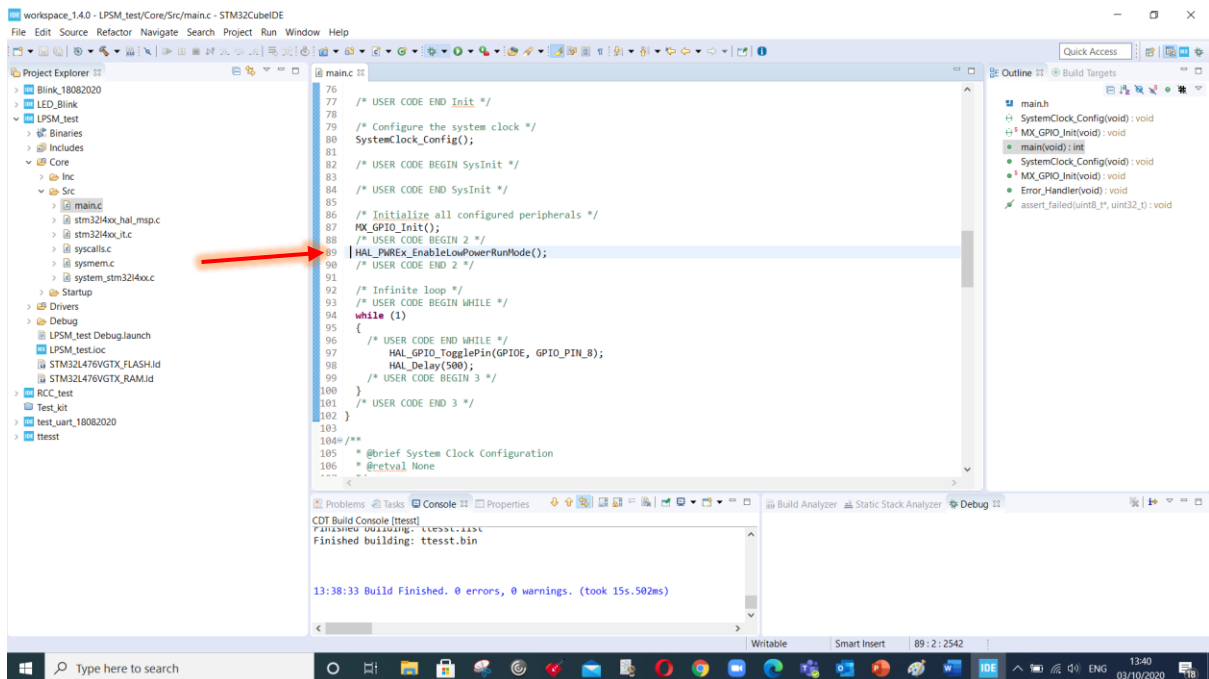


## Step 10:

Enable the low power mode by using code below:

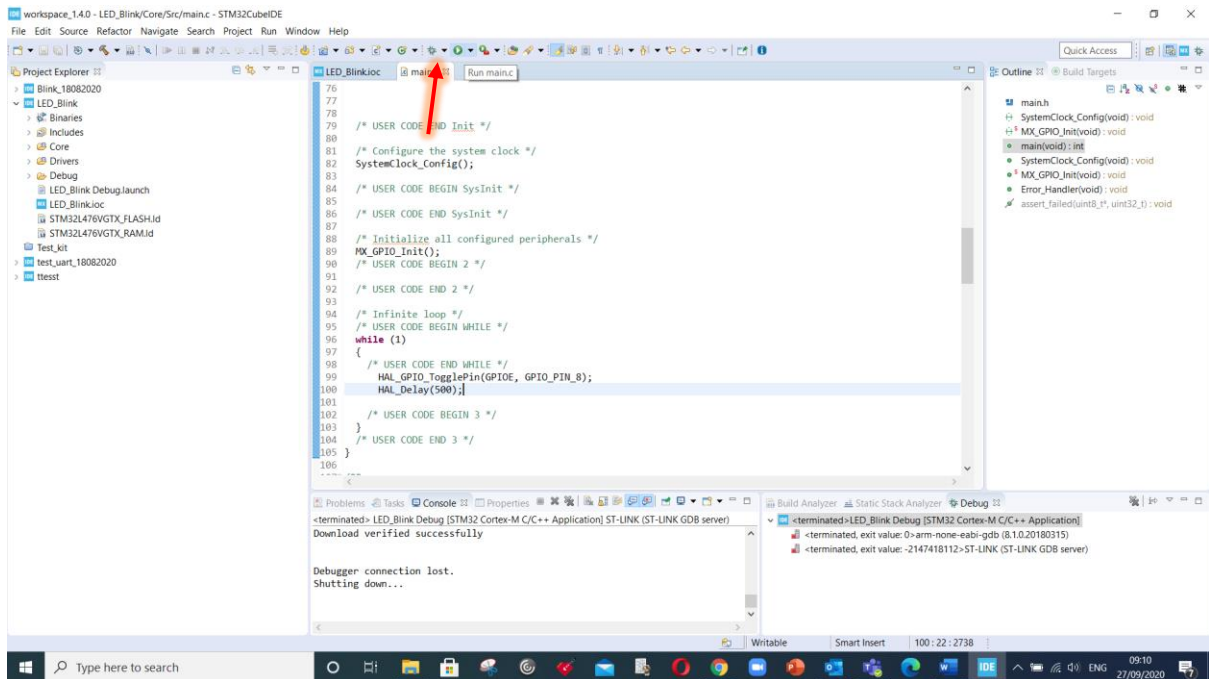
```
/* USER CODE BEGIN Init */  
HAL_PWREx_EnableLowPowerRunMode();
```

Add the toggle LED Code HAL\_GPIO\_Toggle, use CTRL + SPACE to autofill the optsins.



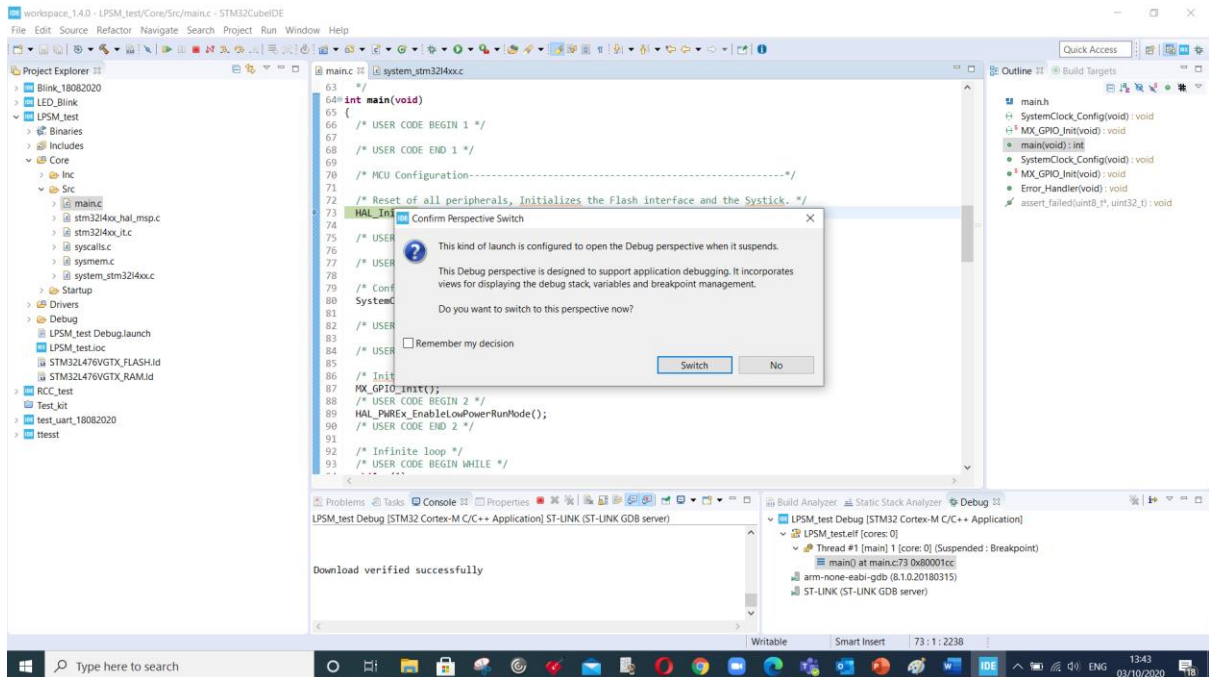
## Step 11:

Press the Run button to build the project and then click on 'Debug' icon



## Step 12:

Click 'Switch' on appearing menu.



### Step 13:

Add a break point on toggle pin statement and then Click on 'Resume' icon to run/stop the execution. Observe LPR register-> Click on 'SFRs' and select 'PWR' and expand the list to view setting of 'LPR' bit which should change from '0' to '1'.

The screenshot shows the STM32CubeIDE interface during a debug session. The main editor displays the source code for `stm324xx_it.c`. A breakpoint is set on line 110, `HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_8);`. The debugger console shows the message "Download verified successfully".

The SFRs (System Function Registers) window is open, showing a list of registers. The 'PWR' register is selected and expanded, showing the 'LPR' register. The 'LPR' register is highlighted in yellow, and its value is shown as `0x1`. The bit field information for the LPR register is also displayed:

Register	Address	Value
PWR	0x40007000	0x4200
LPR	[14:1]	0x1
VOS	[9:2]	0x1
DBP	[8:1]	0x0
LPMS	[0:3]	0x0
CR2	0x40007004	0x0
CR3	0x40007008	0x0000
CR4	0x4000700c	0x0
SR1	0x40007010	0x0
SR2	0x40007014	0x300
SCR	0x40007018	0x0
PUCRA	0x40007020	0x0
PCRCA	0x40007024	0x0
PUCRB	0x40007028	0x0
PCRCB	0x40007030	0x0

The bit field information for the LPR register is:

Bit field	Value
LPR	14
LSB	1
MSB	15
Size	1
Reset value	0x0
Access permission	RW
Read action	