

## Lab 2: EXTI, NVIC and RCC

### Activity 1: EXTI Configuration

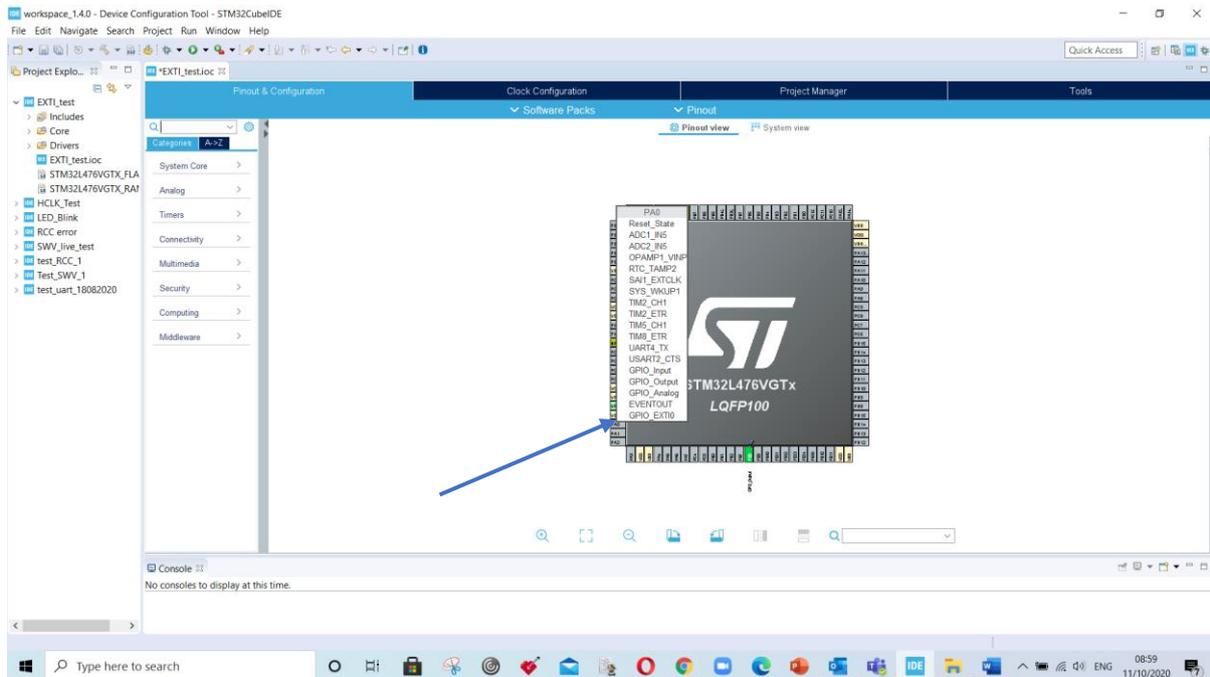
**Aim:** Configuring GPIO as an external interrupt on STM32L476VG MCU.

#### **Objectives:**

- 1- Learn how to configure the external interrupt controller for a GPIO
- 2- Declaring the callback function and its prototype
- 3- Testing the functionality by toggling LED using external interrupt

**Step 1:** Create a project in STM32CubeIDE and select 'PE8' as output mode, refer to previous lab if you are not sure how to do it.

**Step 2:** Left click on 'PA0' and select 'GPIO\_EXTI0' to configure it as external interrupt. This PIN is attached to center button of joystick.



**Step 3:** Go to 'System Core', select 'GPIO', choose pin 'PA0' and in 'GPIO mode' select 'External Interrupt Mode with Rising edge trigger detection'

The screenshot shows the STM32CubeIDE configuration tool. The left sidebar is titled 'Categories' and shows a tree view under 'System Core'. The 'GPIO' category is selected and highlighted in blue. A purple arrow points to 'System Core', a green arrow points to 'GPIO', and a yellow arrow points to 'PA0' in the table below. The main window is titled 'GPIO Mode and Configuration' and shows a configuration table for pins PA0 and PE8. Below the table, the 'PA0 Configuration' section is visible, with the 'GPIO mode' dropdown menu set to 'External Interrupt Mode with Rising edge trigger detection'. A red arrow points to this dropdown menu.

Categories: A->Z

- System Core
  - DMA
  - GPIO**
  - IWDG
  - NVIC
  - ⚠ RCC
  - ⚠ SYS
  - TSC
  - WWDG
- Analog >
- Timers >
- Connectivity >
- Multimedia >
- Security >
- Computing >
- Middleware >

GPIO Mode and Configuration

Configuration

Group By Peripherals

GPIO NVIC

Search Signals

Search (Ctrl+F)  Show only Modified Pins

Pin ...	Signal ...	GPIO o...	GPIO ...	GPIO ...	Maxim...	Fast M...	User L...	Modified
PA0	n/a	n/a	Extern...	No pull...	n/a	n/a		<input checked="" type="checkbox"/>
PE8	n/a	Low	Output ...	No pull...	Low	n/a		<input type="checkbox"/>

PA0 Configuration :

GPIO mode: External Interrupt Mode with Rising edge trigger detection

GPIO Pull-up/Pull-down: No pull-up and no pull-down

User Label:

**Step 4:** From 'System Core' menu, select 'NVIC' and check the 'EXTI line0 interrupt'

The screenshot shows the 'Pinout & Configuration' tool interface. The 'System Core' menu is expanded, and 'NVIC' is selected. The 'NVIC Mode and Configuration' window is open, showing the 'Configuration' tab. The 'EXTI line0 interrupt' is checked in the 'NVIC Interrupt Table'.

**System Core**

- DMA
- GPIO
- IWDG
- NVIC**
- RCC
- SYS
- TSC
- WWDG

**Categories** A->Z

**NVIC Mode and Configuration**

**Configuration**

- NVIC
- Code generation

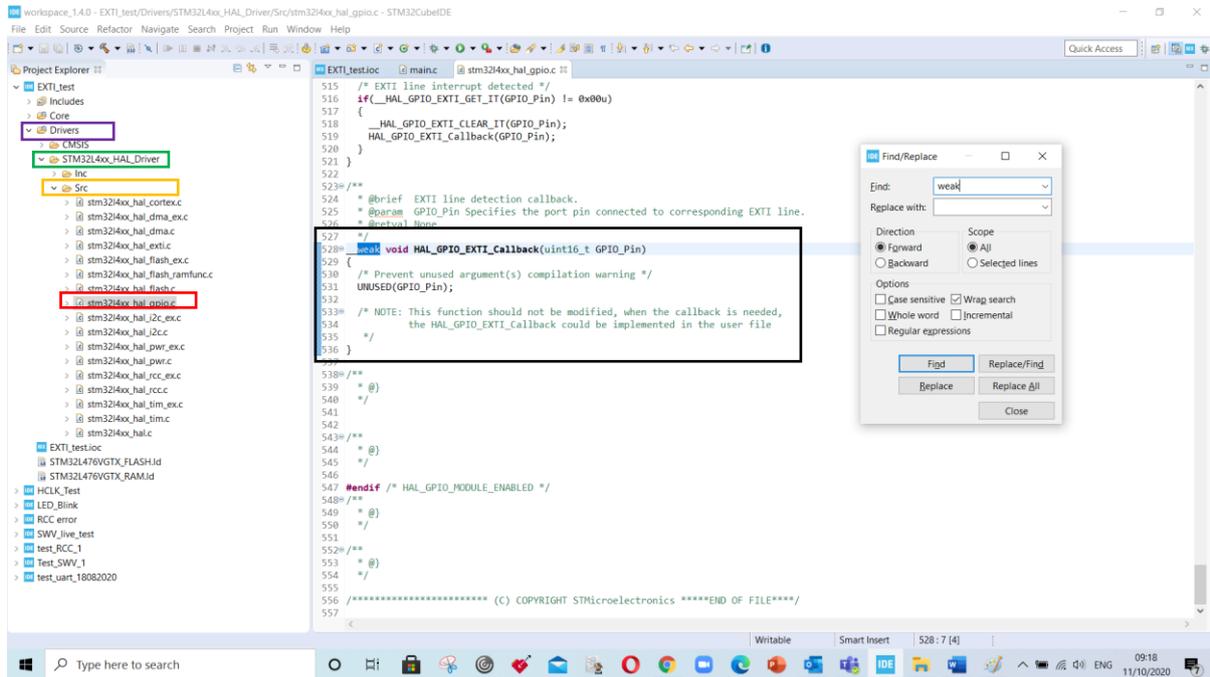
Priority Group: 4 bits for pre-emption...  Sort by Preemption Priority and Sub Priority

Search: Search (Ctrl+F)  Show only enabled interrupts  Force

NVIC Interrupt Table	Enabled	Preemption Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0
Memory management fault	<input checked="" type="checkbox"/>	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0
Debug monitor	<input checked="" type="checkbox"/>	0
Pendable request for system service	<input checked="" type="checkbox"/>	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0
PVD/PVM1/PVM2/PVM3/PVM4 interrupts through EXTI lines 16/35/36/37/...	<input type="checkbox"/>	0
Flash global interrupt	<input type="checkbox"/>	0
RCC global interrupt	<input type="checkbox"/>	0
<b>EXTI line0 interrupt</b>	<input checked="" type="checkbox"/>	0
FPU global interrupt	<input type="checkbox"/>	0

Enabled Preemption Priority: 0 Sub Priority

**Step 5: Generate the code** and in files select folders 'Driver', then select folder 'STM32L4xx\_HAL\_Driver', then select 'Src' and finally select 'Stm32l4xx\_hal\_gpio.c'. Now find the WEAK of instance function 'HAL\_GPIO\_EXTI\_callback'.



Copy and paste the code in the 'main.c' file in area of `/* USER CODE BEGIN 4 */`, as below:

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
}
```

Create/Declare the function prototype in main.c file, as shown below. (See Appedix A)

```
/* USER CODE BEGIN PFP */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);
```

In external interrupt function callback add the GPIO PE8 toggle code: (See Appedix B)

```
if (GPIO_Pin == GPIO_PIN_0)
{
    HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_8);
}
```

**Note:** We have use if statement to verify that this call back was generated by our desired pin GPIO PA0.



Run 'Run' and build the code. You will not find any error or warning messages. Now test your code by toggling the LED using joystick middle button.

**To do:** Blink the Red LED on STM32L476\_DISCO board with delay of 500ms and toggle the Green LED using External interrupt through joystick.

## Appendix A: Prototype function declaration

The screenshot shows the IDE workspace for a project named 'workspace\_1.40 - EXTL\_test/Core/Src/main.c - STM32CubeIDE'. The main editor displays the file 'stm324xx\_hal\_gpio.c'. The code includes various headers and defines macros. The function prototypes section is highlighted with a yellow box, showing the following declarations:

```
static void MX_GPIO_Init(void);  
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);
```

The Build Console shows the build process for 'test\_uart\_18082020.list' completed successfully at 09:48:08. The Build Analyzer shows memory usage details for RAM, RAM2, and FLASH.

Region	Start address	End address	Size	Free	Used	Usage (%)
RAM	0x20000000	0x20018000	96 KB	94.45 KB	1.55 KB	1.61%
RAM2	0x10000000	0x10008000	32 KB	32 KB	0 B	0.00%
FLASH	0x08000000	0x08100000	1024 KB	1017.84 KB	6.16 KB	0.60%

## Appendix B: External interrupt call back function

The screenshot shows the IDE workspace for the same project. The main editor displays the file 'stm324xx\_hal\_gpio.c'. The implementation of the 'void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin)' function is highlighted with a yellow box. The code includes the following implementation:

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)  
{  
    if (GPIO_Pin == GPIO_PIN_0)  
    {  
        HAL_GPIO_TogglePin(GPIO10E, GPIO_PIN_8);  
    }  
}
```

The Build Console shows the build process for 'test\_uart\_18082020.list' completed successfully at 09:48:08. The Build Analyzer shows memory usage details for RAM, RAM2, and FLASH.

Region	Start address	End address	Size	Free	Used	Usage (%)
RAM	0x20000000	0x20018000	96 KB	94.45 KB	1.55 KB	1.61%
RAM2	0x10000000	0x10008000	32 KB	32 KB	0 B	0.00%
FLASH	0x08000000	0x08100000	1024 KB	1017.84 KB	6.16 KB	0.60%

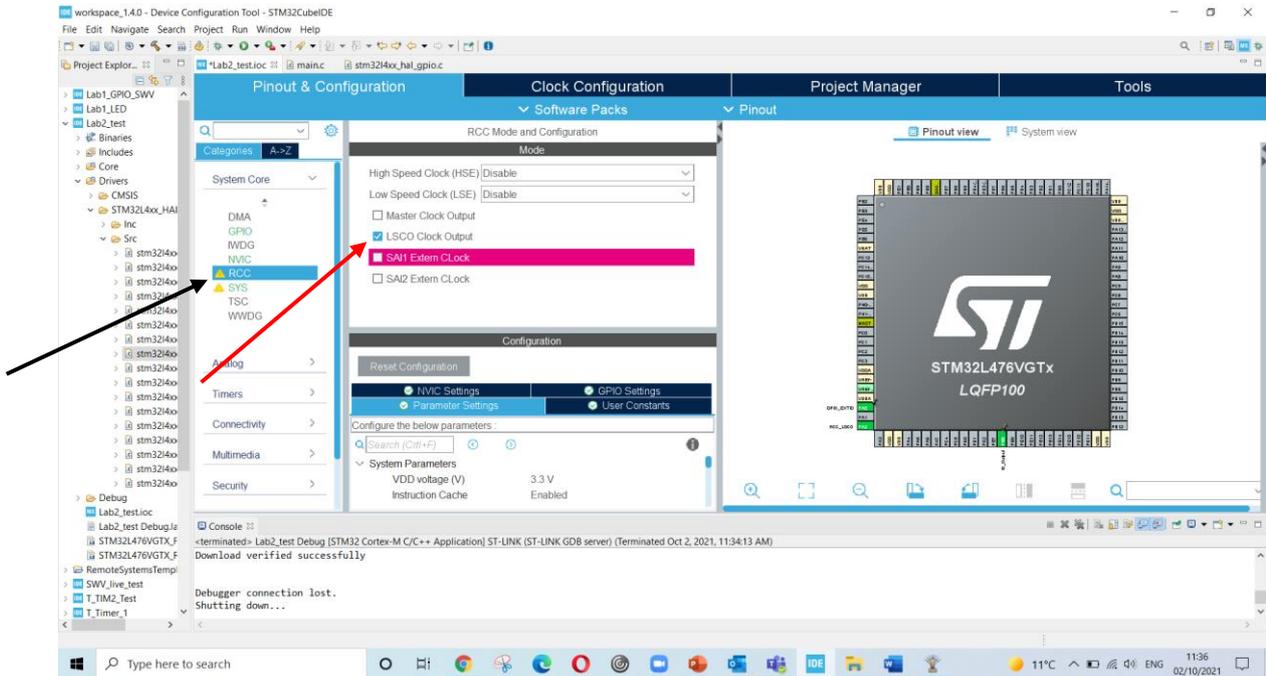
## Activity 2: RCC configuration

**Aim:** To configure Reset and clock controller to generate a Low speed clock output (LSCOCO) on STM32L476VG MCU.

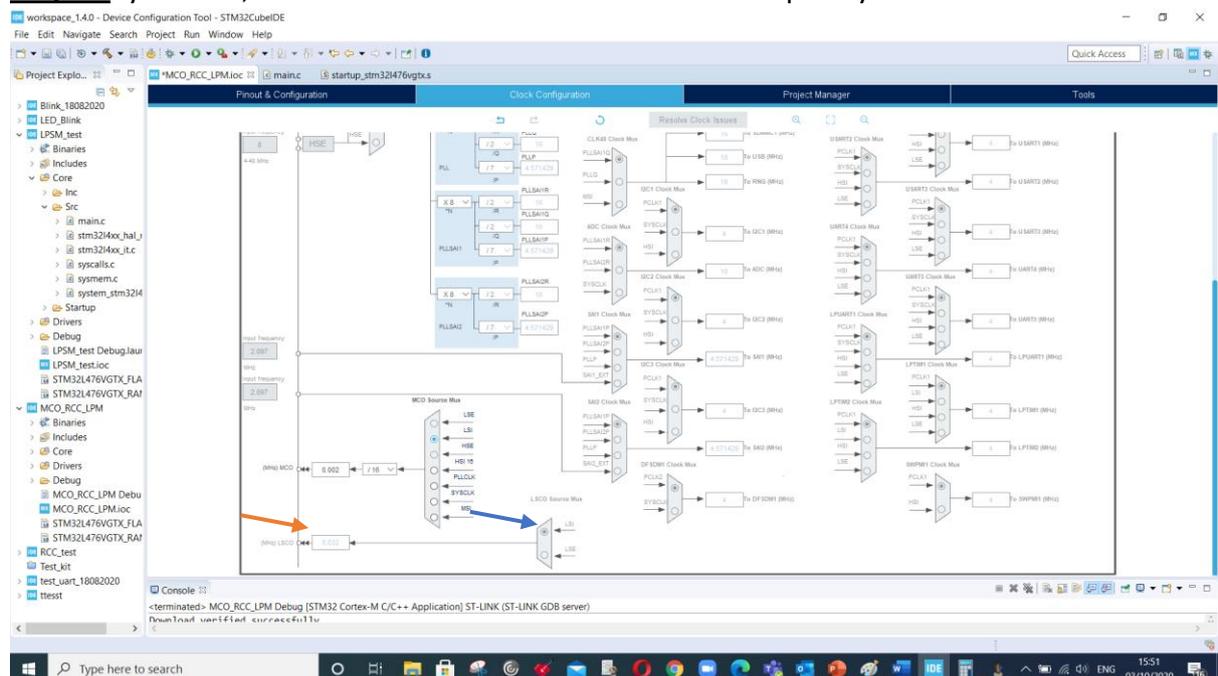
### Objectives:

- 1- Generate a clock output on Pin PA2 for external circuitry
- 2- Observer the Waveform on function Generator

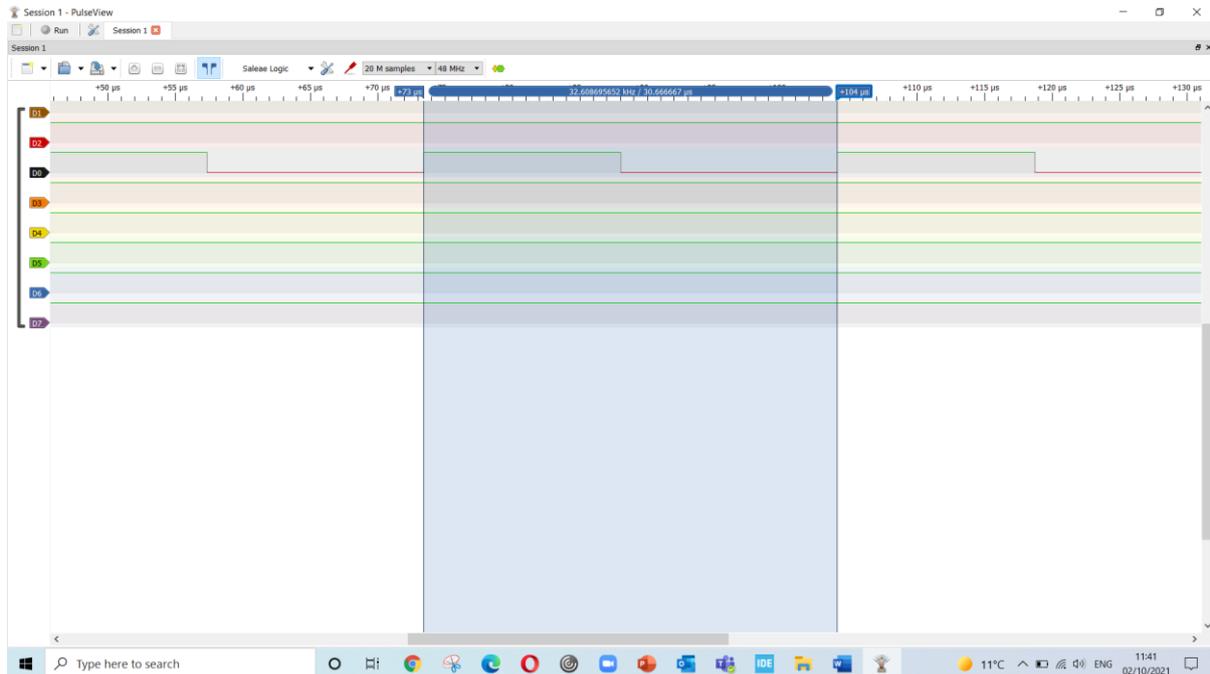
**Step 1:** In the 'System Core' select the 'RCC' and check the 'LSCO Clock Output'



**Step 2:** By default, "LSI" will be clock source for LSCO at frequency of 32kHz.



**To do:** Capture the waveform in logic analyzer on pin 'PA2' and measure the waveform frequency.



### **Activity 3: SWV trace analysis**

**Aim:** To debug the signal on STM32L476VG MCU inside STM32CubeIDE.

**Objectives:**

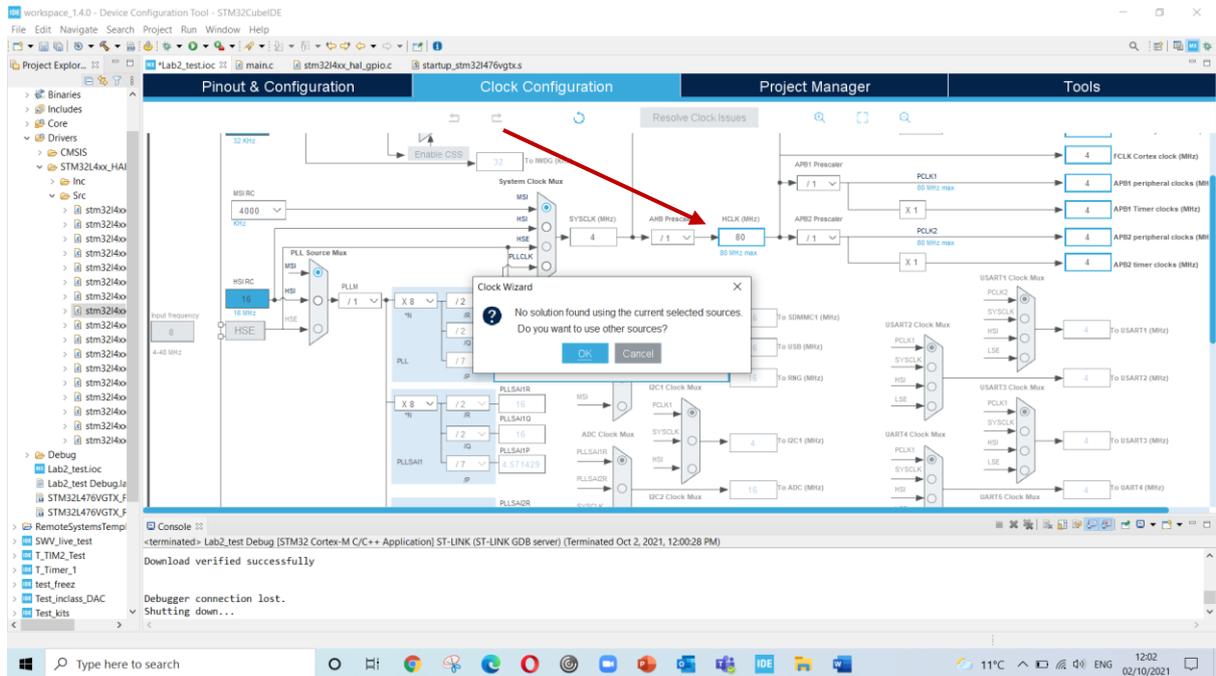
- 1- Generate a clock output on Pin PE8.
- 2- Configure SWV trace analysis
- 3- Observer the Waveform inside STM32CubeIDE

**Step 1:** Configure GPIO PB2 as output and connect SWO and PB3 on STM32L4 DISCO board.

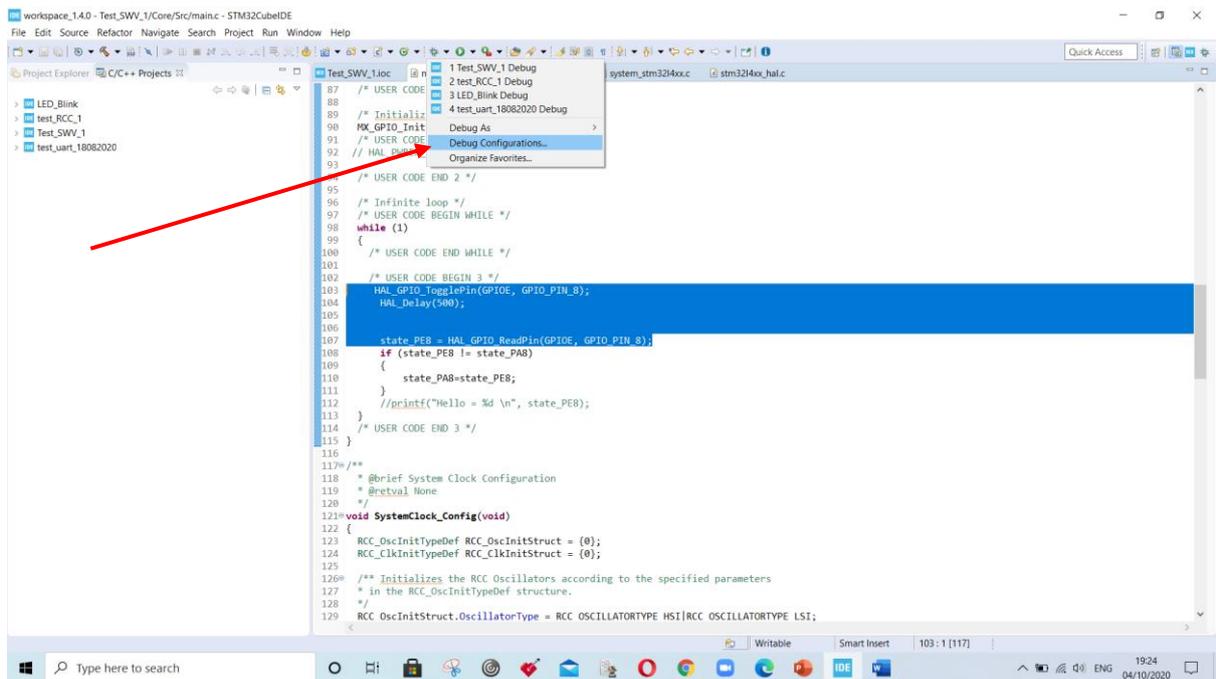
**Step 2:** use the following code to toggle led and once the code is generated.

```
/* USER CODE BEGIN 1 */
int state_PB2 = 0;
/* USER CODE BEGIN 3 */
HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_2);
HAL_Delay(500);
state_PB2 = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_2);
```

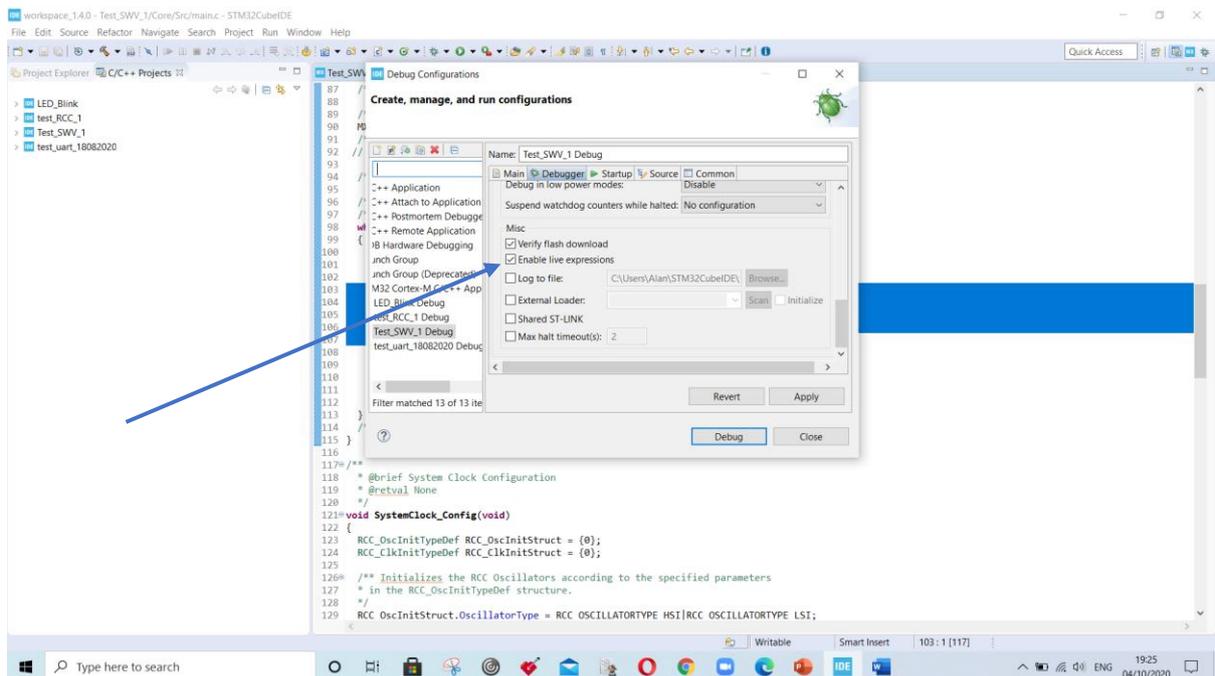
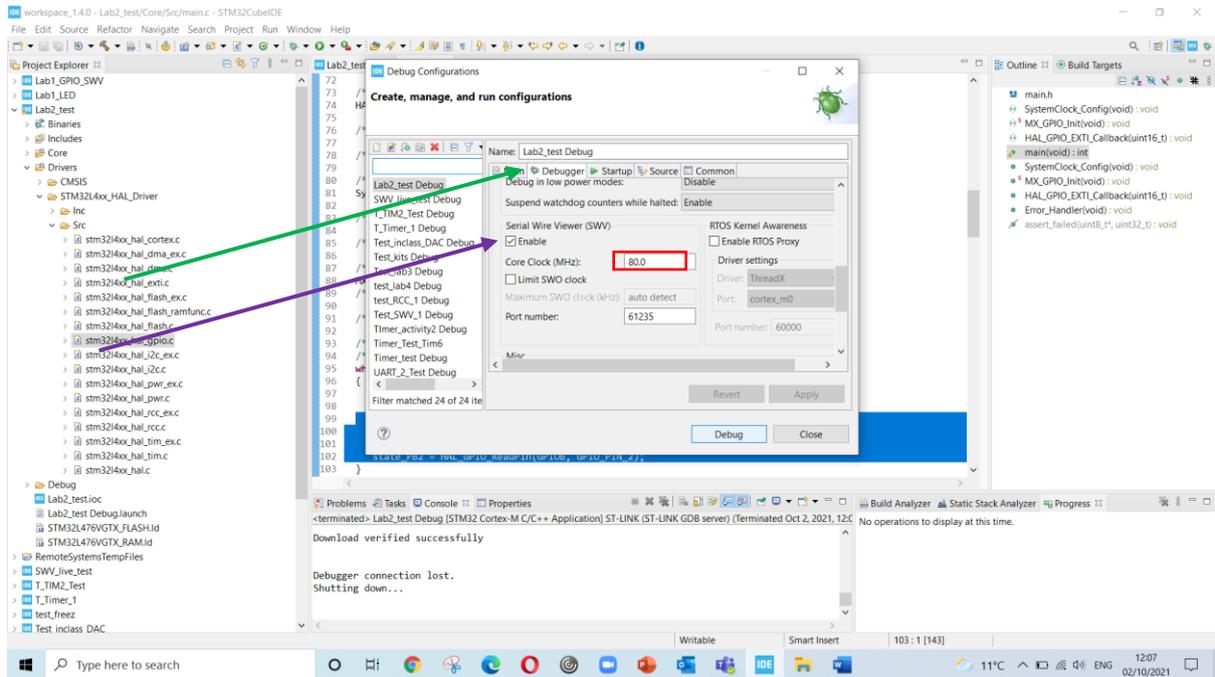
**Step 3:** Set **HCLK** to 80 MHz and press Enter Key then it will prompt the 'CLOCK WIZARD' window and press OK.



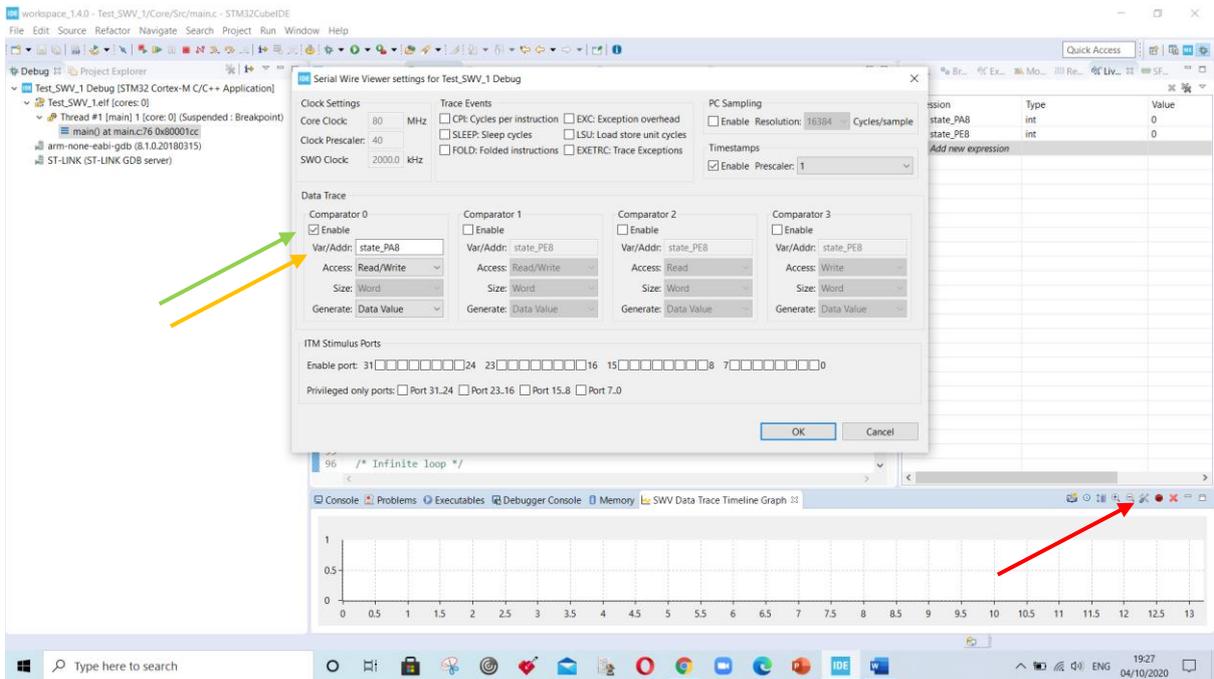
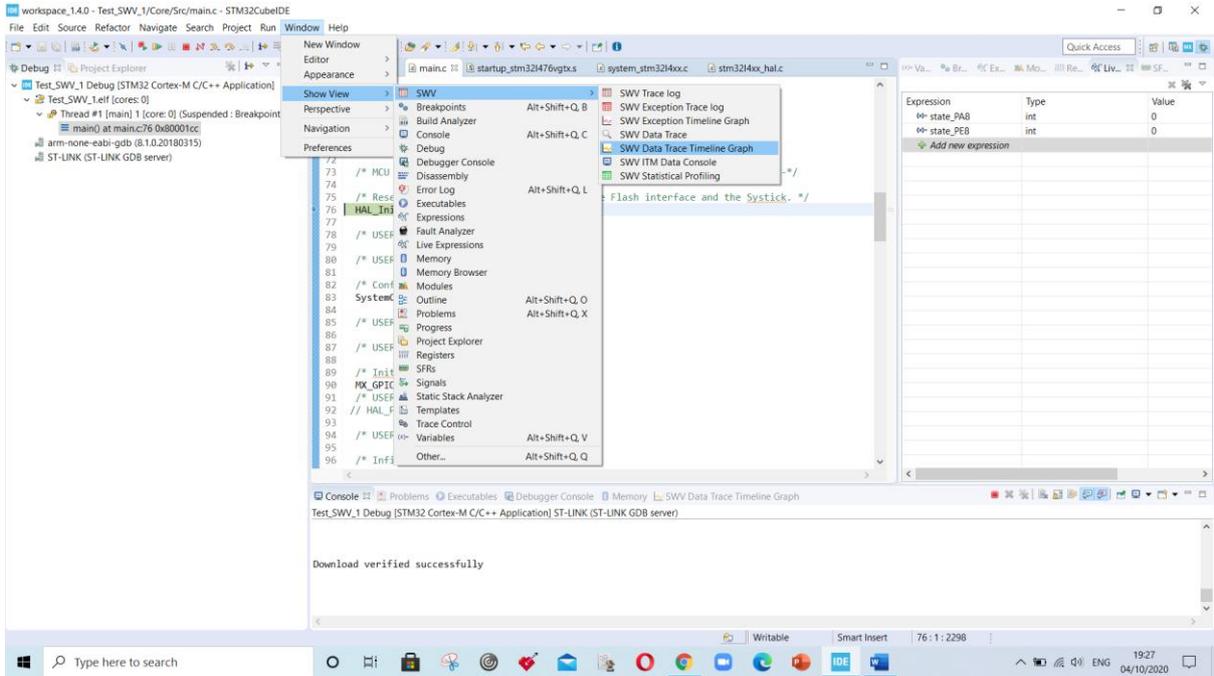
Build the project by using Run button then go to 'Debug Configuration'.



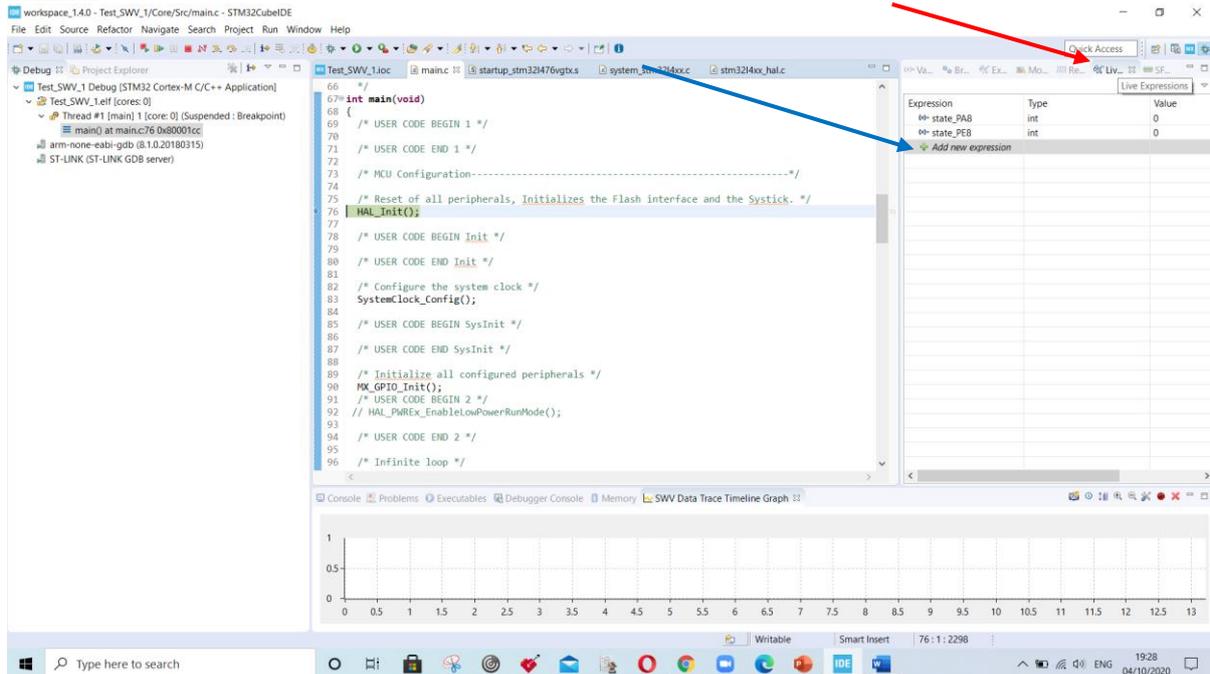
**Step 4:** Go to **Debugger** tab, scroll down to **Serial Wire Viewer** then check **Enable** and add the **80** in core clock. Finally check **Enable Live Expression** option scrolling down the window.



**Step 5:** Select **Window->Show View->SWV -> SWV Trace Timeline** graph. Then Click on **configuration** option and **Enable comparator 0** and add variable name **'state\_PB2'**.



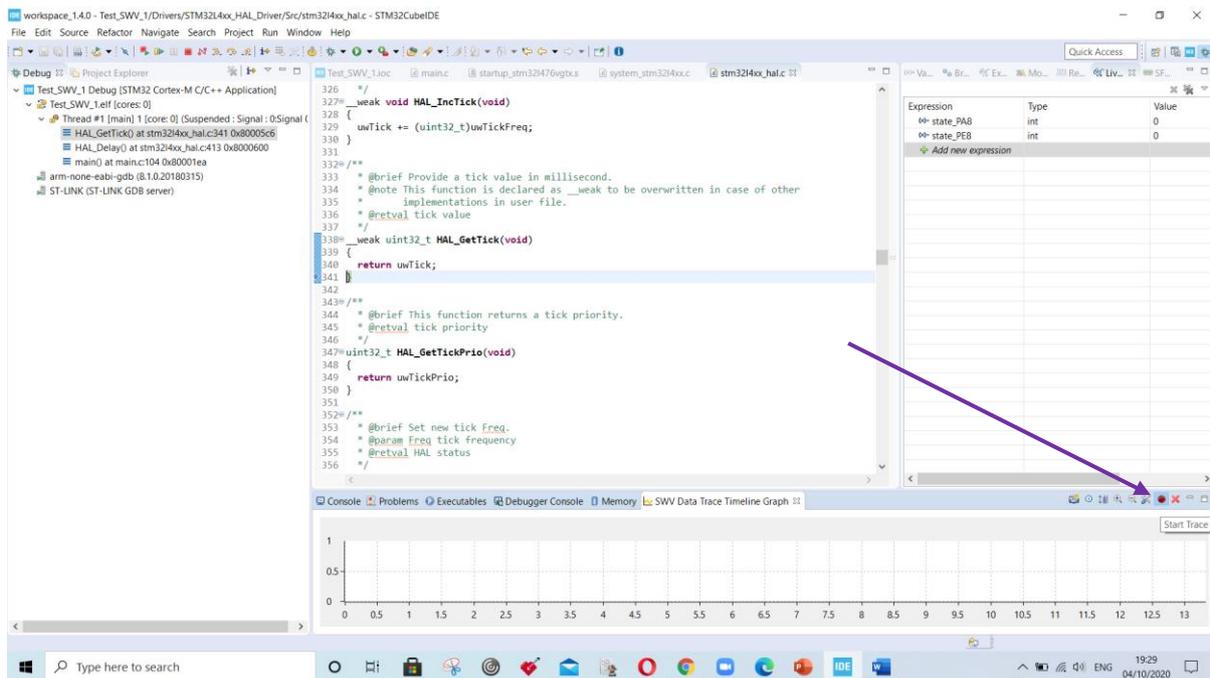
## Step 6: Select Live expression window and Add the variable to monitor 'state\_PB2'.



The screenshot shows the STM32CubeIDE interface. The main editor displays the `main` function in `stm324xx_hal.c`. The `HAL_Init()` function is highlighted. On the right, the 'Live Expressions' window is open, showing a table with columns for 'Expression', 'Type', and 'Value'. The table contains two entries: `state_PB2` (int, 0) and `state_PEB` (int, 0). A red arrow points to the 'Quick Access' button in the top right corner of the IDE. A blue arrow points to the 'Add new expression' button in the Live Expressions window.

Expression	Type	Value
state_PB2	int	0
state_PEB	int	0

Select start trace option the press the resume button to run the debug mode.



The screenshot shows the STM32CubeIDE interface. The main editor displays the `HAL_IncTick` function in `stm324xx_hal.c`. The `HAL_IncTick` function is highlighted. On the right, the 'Live Expressions' window is open, showing the same table as in the previous screenshot. A purple arrow points to the 'Start Trace' button in the bottom right corner of the Live Expressions window.

Expression	Type	Value
state_PB2	int	0
state_PEB	int	0

You can view the square wave as below:

The screenshot shows an IDE window with a C code file named `stm324xx_hal.c`. The code includes functions for handling ticks, such as `HAL_IncTick` and `HAL_GetTick`. A red arrow points to the 'Run' button in the IDE's toolbar. Below the code, the 'SWV Data Trace Timeline Graph' is visible, displaying a square wave signal for `state_PAB` over a time period from 0 to 13 seconds. The signal alternates between 0 and 1, creating a flashing pattern. The graph is highlighted with an orange border.

Expression	Type	Value
state_PAB	int	1
state_PEB	int	1

**To do:** Create the light house flashing pattern in SWV Data Trace Timeline Graph.