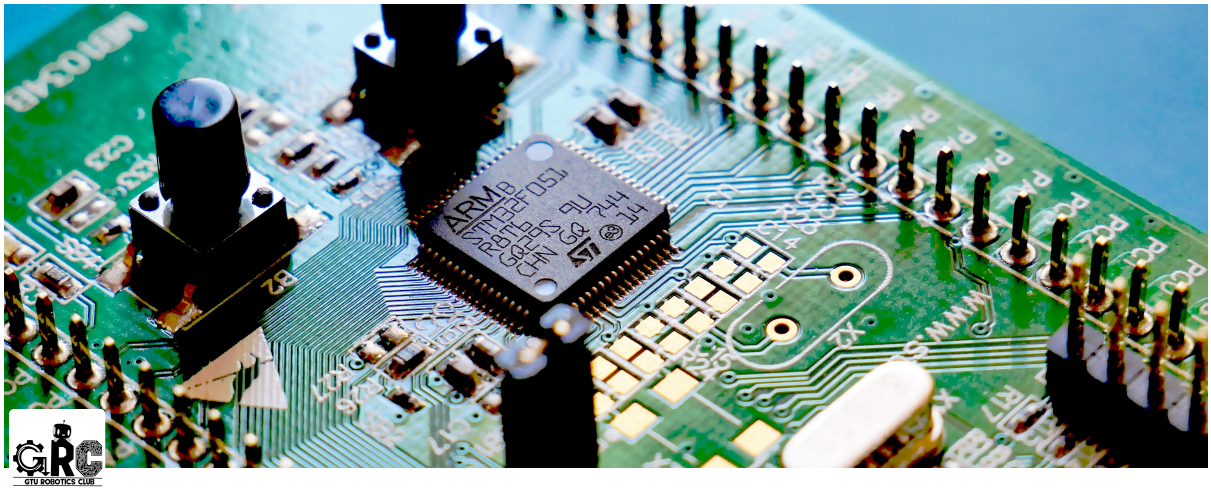


STM32 Debugging Techniques

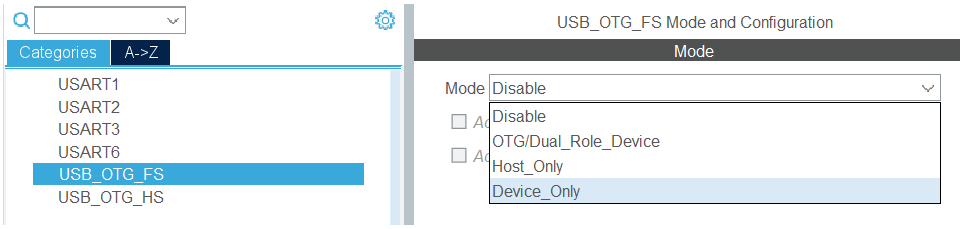
➡️ Before starting debugging, make sure your project does not contains any errors.

Serial Printing

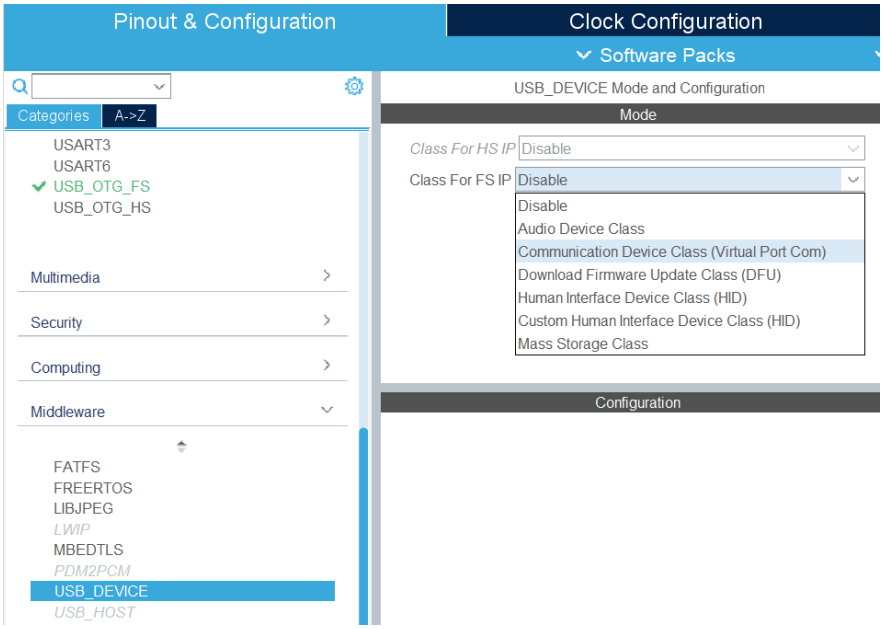
- If you are using a **Nucleo Board**, serial printing can be done with **USART** method and also using the **Virtual COM Port (VCP)** method.
- In the case of a **Discovery Board**, serial printing can be done only with **Virtual COM Port (VCP)** because the USART pins of the STM isn't connected with the USART of the onboard ST-LINK.
- The advantage of using USART to debug is that you can upload the code and serial print through the same cable. However, in the case of Virtual COM Port, you need to use both the USB ports of your board, one to upload and the other to debug.

Virtual COM Port Method (For Nucleo/Discovery/Custom Boards)

1. In the ioc file of your project, go to the **Connectivity** section, click on **USB_OTG_FS** and select Mode as **Device Only**.



2. Scroll down to the **Middleware** section and set Class for FS IP as **Communication Device Class (Virtual Port Com)**



3. Save and generate the code.
4. Include the following files in your code

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "usb_device.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "usbd_cdc_if.h"
#include "string.h"
/* USER CODE END Includes */
```

5. Declare a `char buffer[]` which will contain the string to be printed.

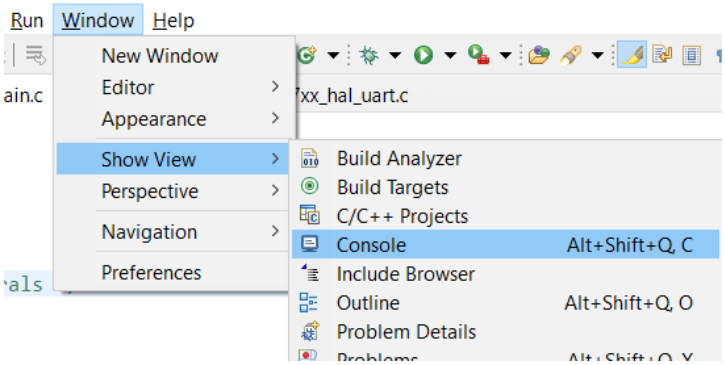
```
/* USER CODE BEGIN PV */
char buf[50]; //Increase the size of array depending on how long your string is, which is to be printed
/* USER CODE END PV */
```

6. Inside `while(1)` , convert the value of your variable from `int` to a `string` using `sprintf` .
   Use `CDC_Transmit_FS` to transmit the string to be printed on your Serial Monitor.

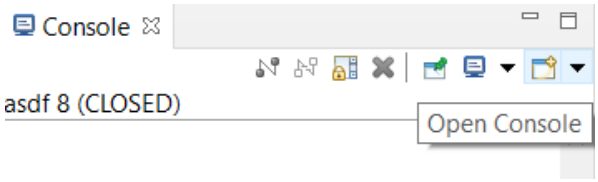
```
while (1) {
    /* USER CODE BEGIN 3 */
    sprintf(buf, "Hello from CDC! Our Counter: %d\n\r", counter);
    CDC_Transmit_FS(buf, strlen(buf));
    HAL_Delay(200);
    counter++;
}
```

`\n\r` signify "end of line" and this helps in proper printing on Serial Monitor

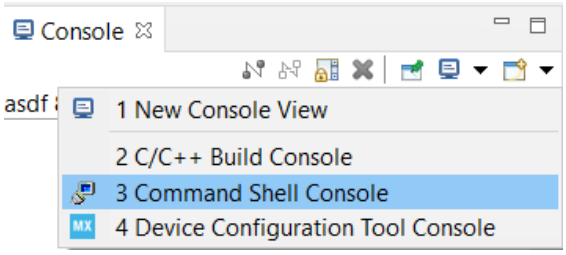
7. Now build and upload the code on your board.
8. Open the Console in STM32CubeIDE



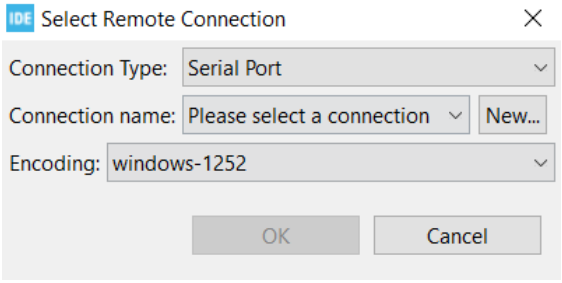
9. Click on the following icon



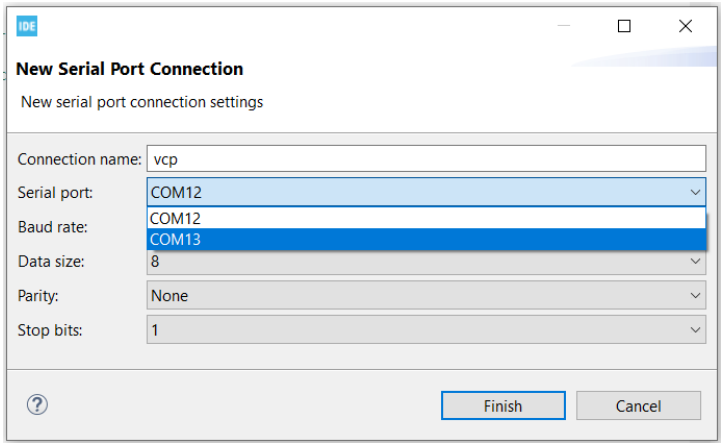
10. Open command shell console



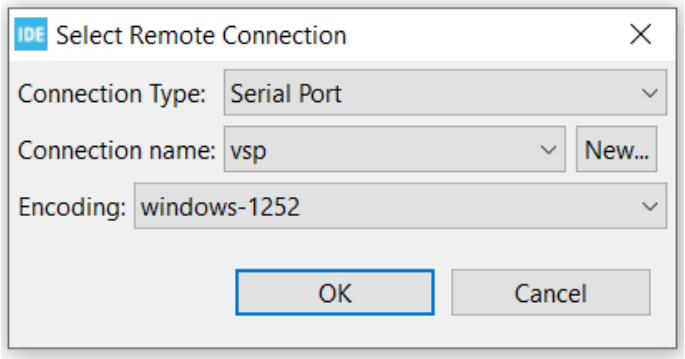
11. The following window should pop up.
    - Set connection type as **Serial Port**
    - Click on **New...** in Connection Name



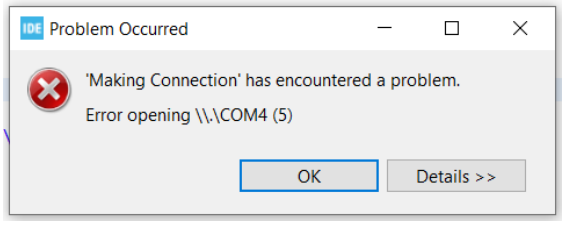
12. Enter the name for serial port, and you'll see that **there are two COM ports**. One for uploading, and the other for debugging, a **Virtual COM Port**. **Make sure you choose the debugging one**. You can find it in the device manager.



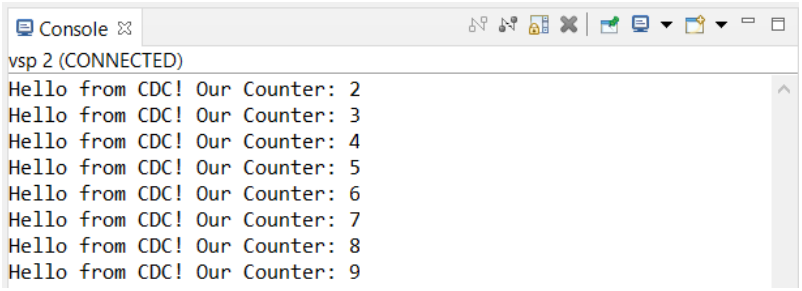
13. Click OK after configuring serial port settings.



14. If you are getting the following error, just disconnect and reconnect your Nucleo Board.



15. Now you should see data getting printed on Console



USART Method (For Nucleo Boards)

1. In the User Manual of your Nucleo Board, find the USART/UART pins which are connected to the onboard ST-Link of the nucleo. We are using STM32F746ZG and as mentioned in the table, we can use **PD8 and PD9** for Serial Printing.

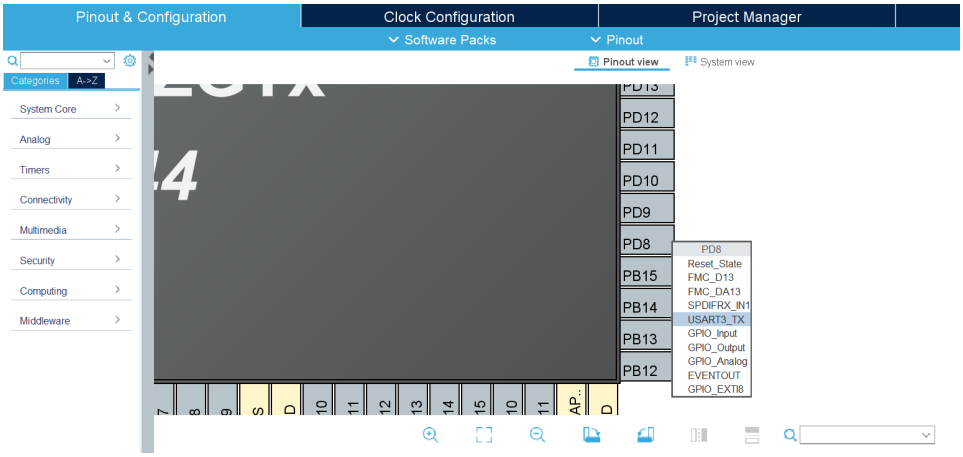
6.9 USART communication

The USART3 interface available on PD8 and PD9 of the STM32 can be connected either to ST-LINK or to ST morpho connector. The choice is changed by setting the related solder bridges. By default the USART3 communication between the target STM32 and the ST-LINK is enabled, to support the Virtual COM port for the mbed (SB5 and SB6 ON).

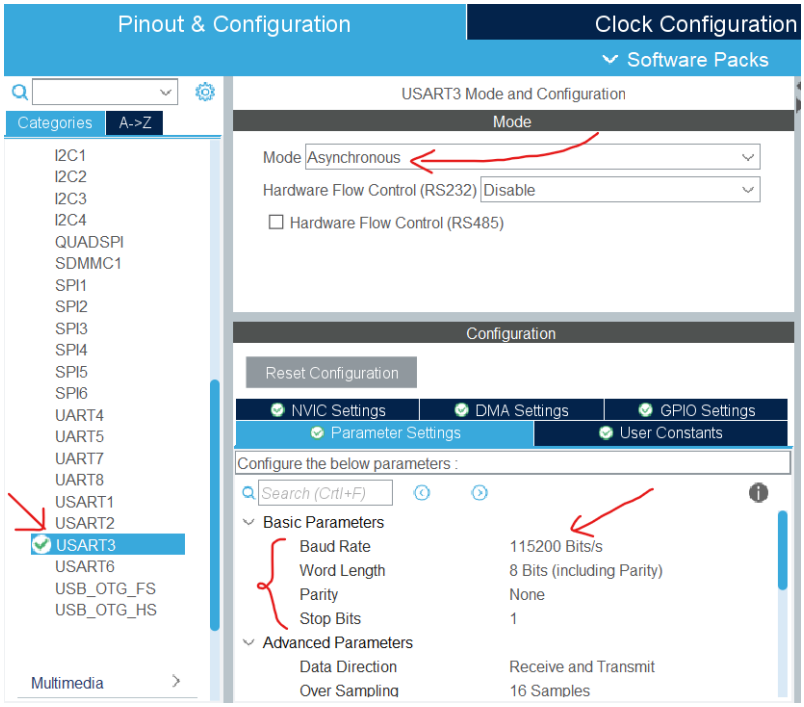
Table 9. USART3 pins

Pin name	Function	Virtual COM port (default configuration)	ST morpho connection
PD8	USART3 TX	SB5 ON and SB7 OFF	SB5 OFF and SB7 ON
PD9	USART3 RX	SB6 ON and SB4 OFF	SB6 OFF and SB4 ON

2. In the pinouts section of .ioc file, configure the USART TX/RX pins as found in the previous step.



3. Go to the specific USART settings and set mode as **Asynchronous.**
   Also set the parameters like **Baud Rate** as per your requirement in Parameter Settings.



4. Include the following files in your *main.c* file

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include <string.h>
/* USER CODE END Includes */
```

5. Create an array of data type `uint8_t` and give it a size, for example, 100. Remember that the size of your array should always be greater than the number of characters you are going to print at a given time. Otherwise the buffer will overflow and you will not see any output on the Serial Monitor.

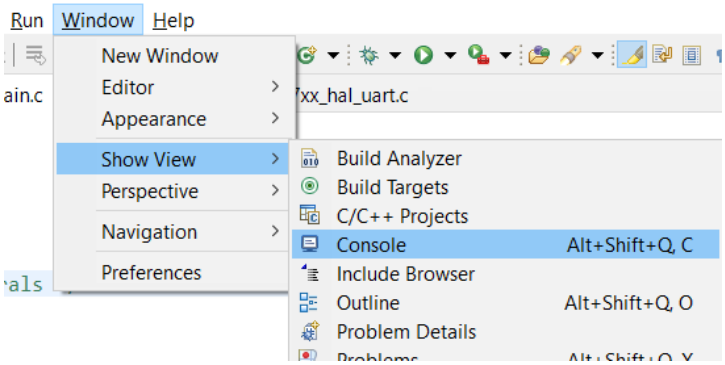
```
/* USER CODE BEGIN 1 */
  uint8_t data[100] = {};
/* USER CODE END 1 */
```

6. Add the following two lines inside `while(1)` to print your code using Serial print: `sprintf()` and `HAL_UART_Transmit()`

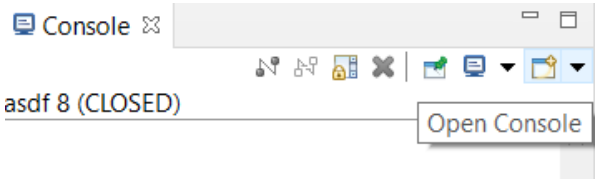
   For example, we are printing the text `Hello! We are tracing the value of x :` at every 500 milliseconds.

```
while (1) {
    sprintf((char *)data, "Hello! We are tracing the value of x : %d\n\r", x);
    HAL_UART_Transmit(&huart3, data, sizeof(data), 100);
    HAL_Delay(500);
    x++;
}
```

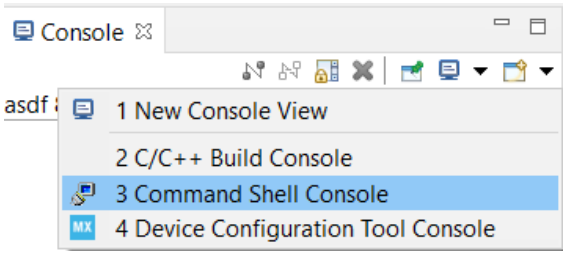
7. Now build and upload the code on your board.
8. Open the Console in STM32CubeIDE



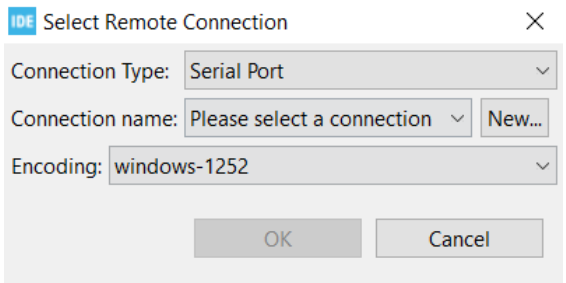
9. Click on the following icon



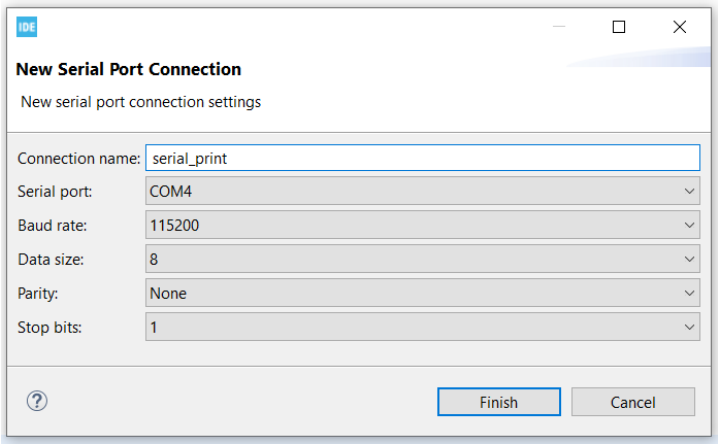
10. Open command shell console



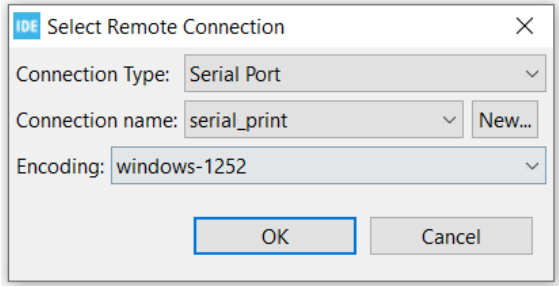
11. The following window should pop up.
    - Set connection type as **Serial Port**
    - Click on **New...** in Connection Name



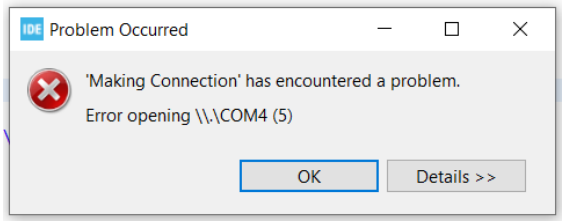
12. Enter the name for serial port and also **make sure that the settings match the USART settings as done in Step 3.** Click Finish.



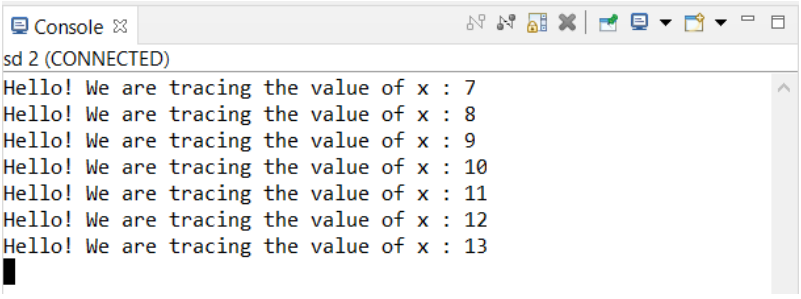
13. Click OK after configuring serial port settings.



14. If you are getting the following error, just disconnect and reconnect your Nucleo Board.



15. Now you should see data getting printed on Console

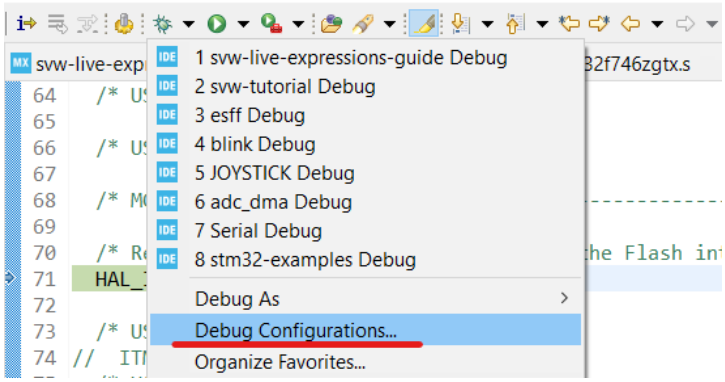


Live Expressions for Variable Values

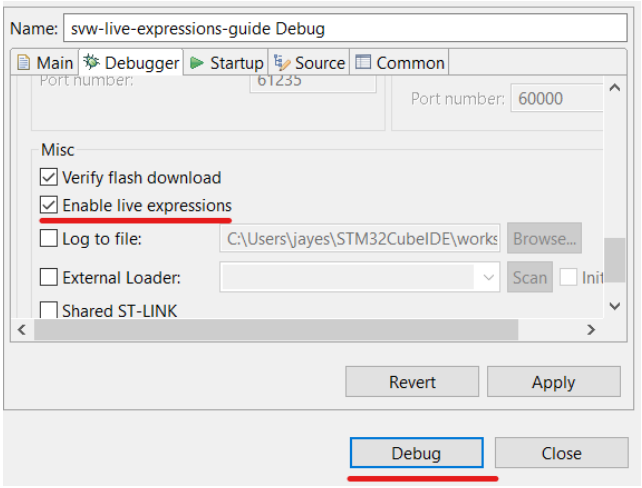
1. The variables you want to track through Live Expressions **MUST be global variables, i.e. declared before** `int main(void)` **or preferably between** `/* USER CODE BEGIN PV */` **and** `/* USER CODE END PV */`

```
/* USER CODE BEGIN PV */
uint16_t loop = 0;
/* USER CODE END PV */
```

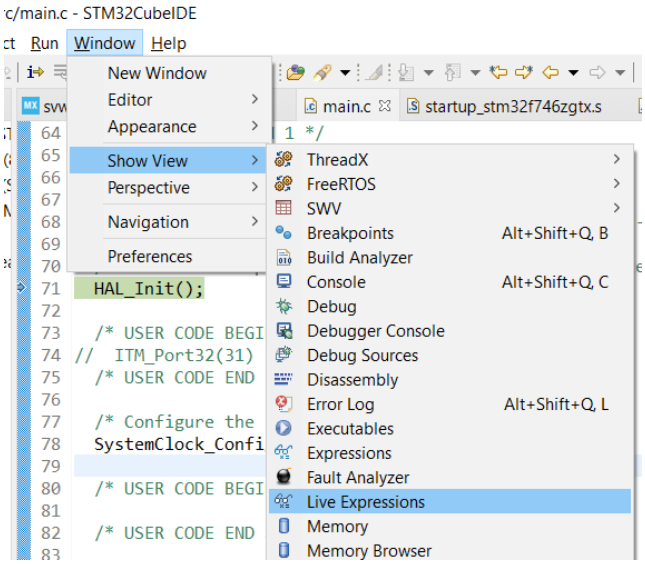
2. Go to **Debug Configurations**



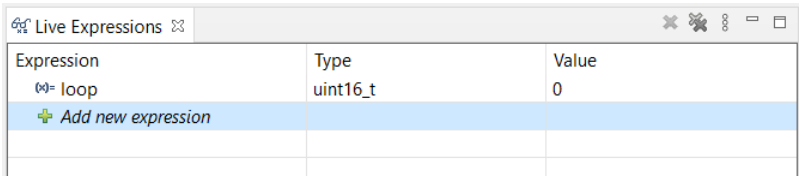
3. Go to Debug tab, check **Enable live expressions**, then click on **Debug**



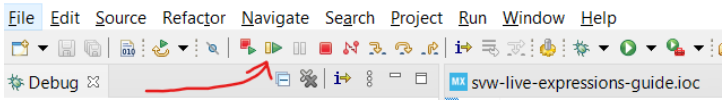
4. After the program is compiled, open the **Live Expressions** tab from here:



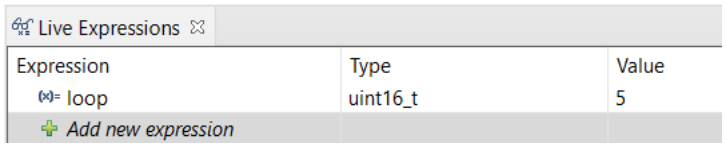
5. In the Live Expressions tab, click on `Add new expression` and enter the name of the variable name (which is case sensitive)



6. Resume the program from here:



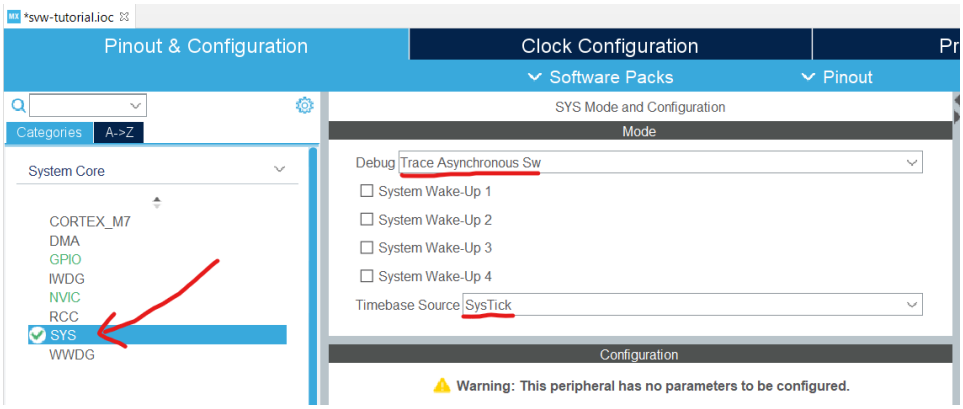
8. Now in the Live Expressions tab, you can see the updated values of variables as the program proceeds



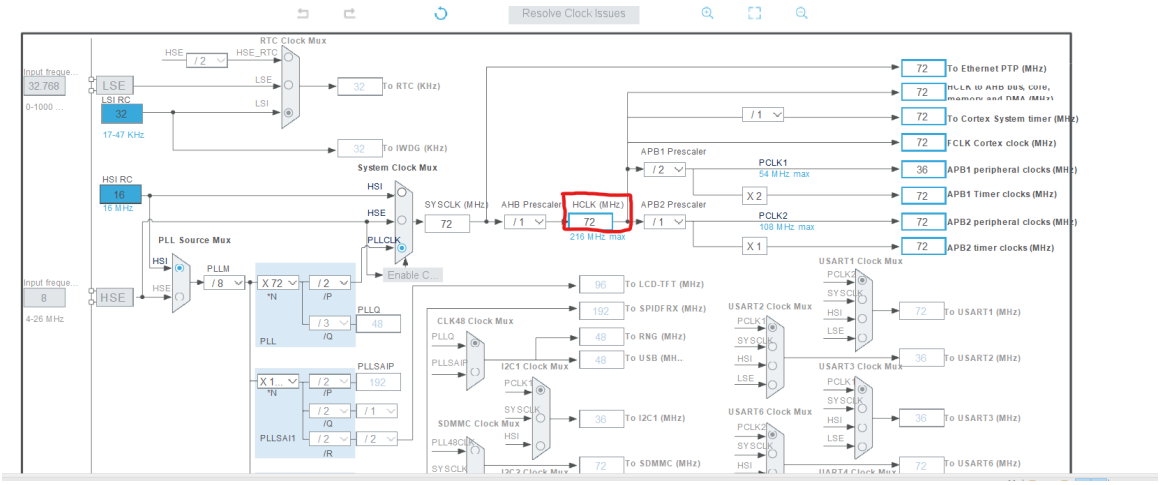
SVW ITM Data Trace

- You can use this method to track variable values.
- This is an alternative method to Live Expressions.
- The only disadvantage of this method is that you can only track 4 variables at a time, not more than that.

1. Go to **SYS** section and set,
   a. Debug as "**Trace Asynchronous Sw**"
   b. Timebase Source as "**SysTick**"

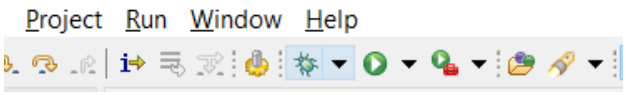


2. Go to Clock Configuration section in *your_project_name.ioc* file and look for HCLK value. Remember the value of HCLK because it will be used later in configurations.

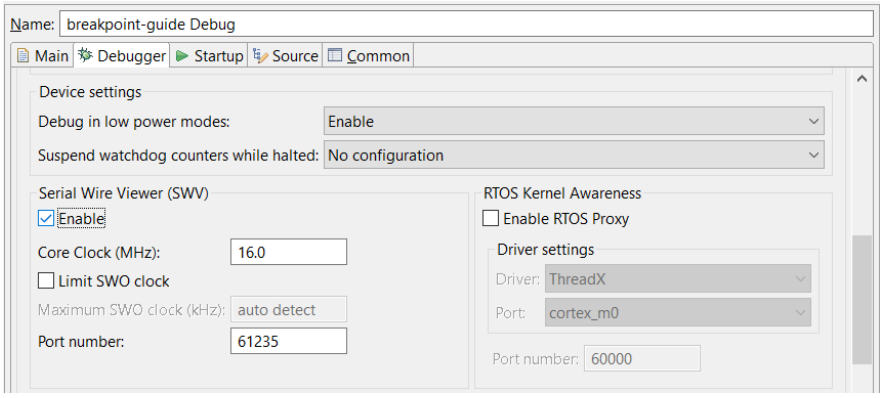


3. Save and generate your code.

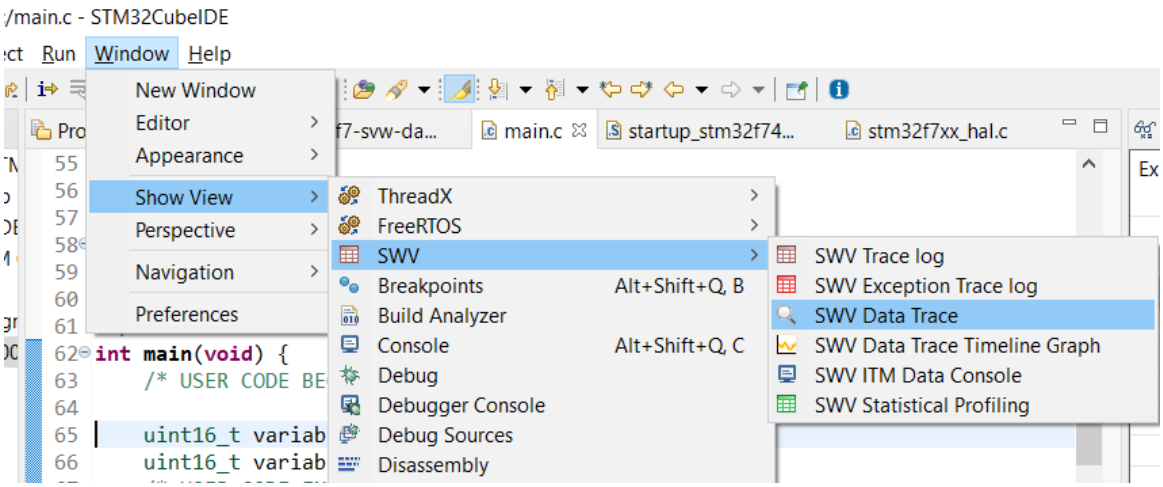
9. Debug the file



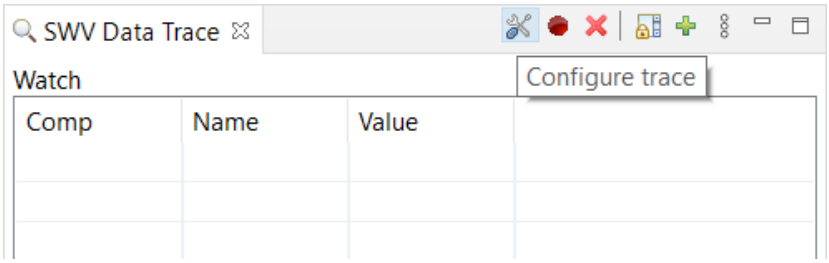
10. In Debug configurations, under Debugger tab, **set your Core Clock equal to the HCLK value found in the clock configurations section (the value which was asked to remember earlier) and click on Debug.**



11. Open SVW Data Trace

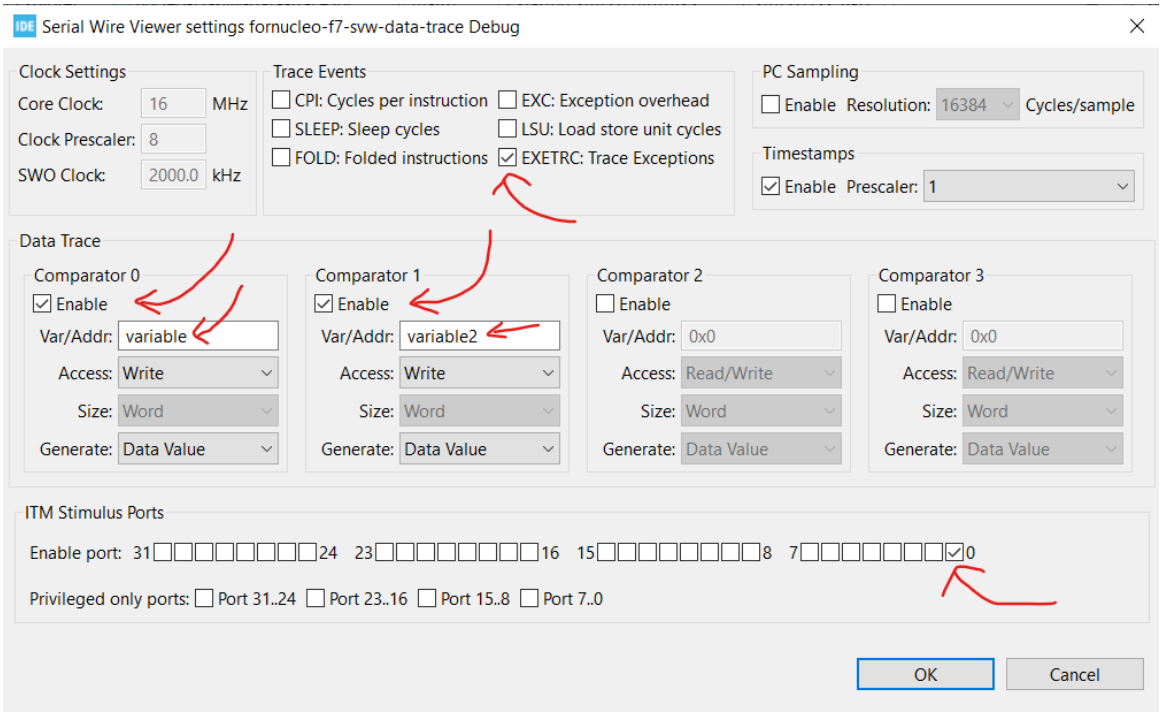


12. Click on Configure Trace option

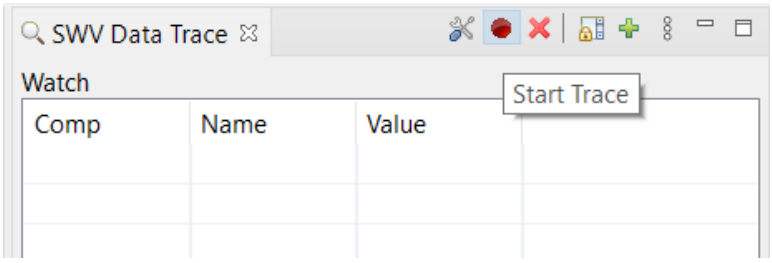


13. Do the following settings in Configuration trace.
    a. Enable any comparator and enter the name of the variable to track.
    b. The name of the variables are case sensitive.
    c. You can add up to 4 variables to track.

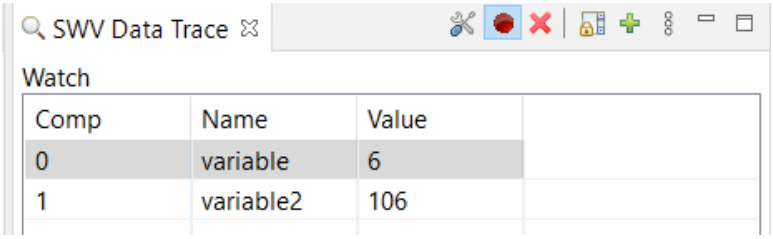
    Press OK when done.



14. After the configurations, Start Trace



15. Now press Resume or F8 and you can see variable values in the SVW Data Trace.

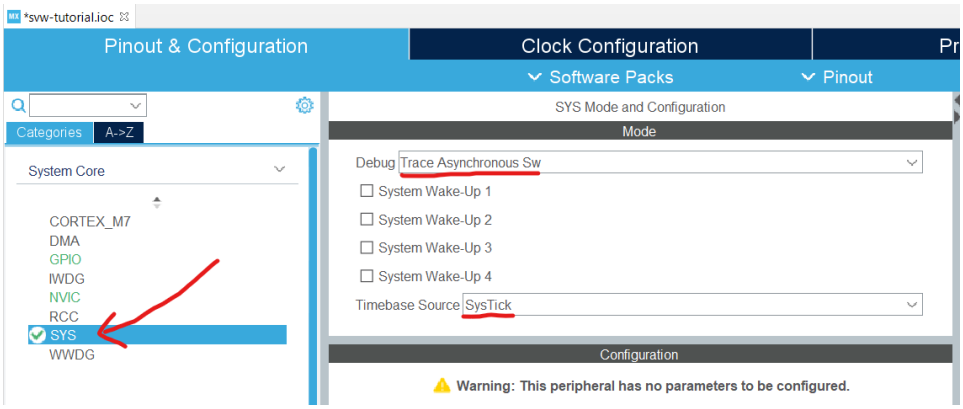


SVW ITM Data Console

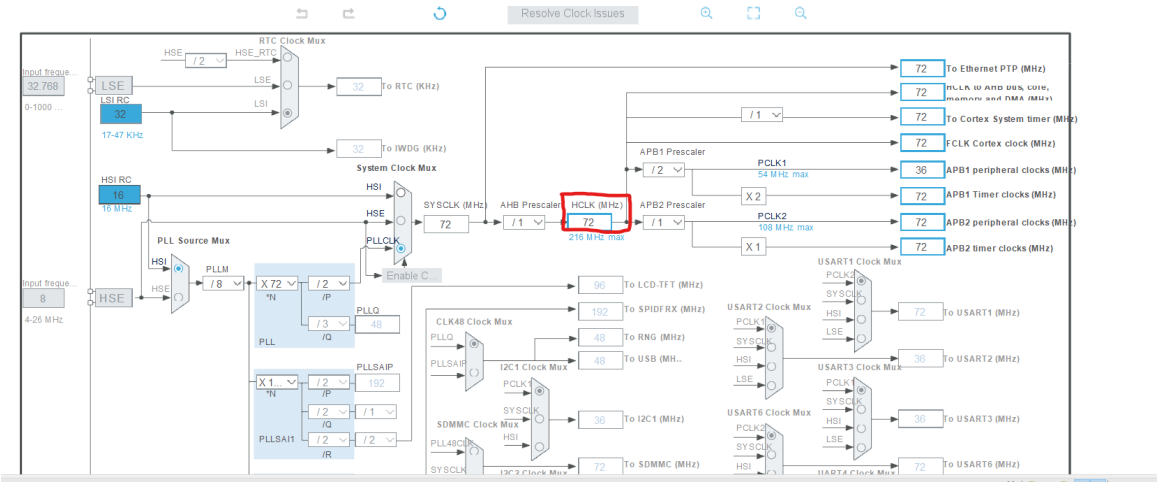
- You can use this method to print data using the `printf()` function of C instead of Serial printing.
- The disadvantage being, it is slower compared to Serial Print.

1. Go to **SYS** section and set,
    a. Debug as "**Trace Asynchronous Sw**"

b. Timebase Source as "**SysTick**"



2. Go to Clock Configuration section in *your_project_name.ioc* file and look for HCLK value. Remember the value of HCLK because it will be used later in configurations.



3. Save and generate your updated code.

4. Add the following lines of codes **before** `int main(void)`

```
/* USER CODE BEGIN Includes */
#include "stdio.h"
/* USER CODE END Includes */
```

```
/* USER CODE BEGIN PM */
#define ITM_Port32(n) (*((volatile unsigned long *) (0xE0000000+4*n)))
/* USER CODE END PM */
```

```
/* USER CODE BEGIN PV */
uint16_t variable = 0;
/* USER CODE END PV */
```

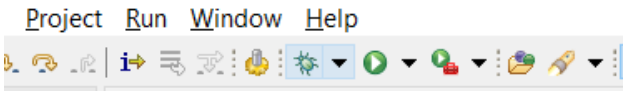
5. Add the following lines of codes **inside** `int main(void)`

```
/* USER CODE BEGIN Init */
  ITM_Port32(31) = 1;
  /* USER CODE END Init */
```

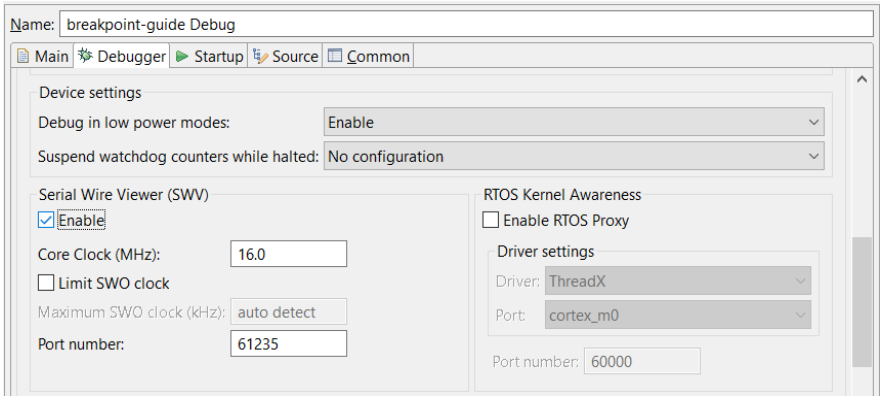
6. Scroll down below and find `/* USER CODE BEGIN 4 */` and add the following code after it to make sure `printf()` works.

```
/* USER CODE BEGIN 4 */
int _write(int file, char *ptr, int len) {
  int i = 0;
  for (i = 0; i < len; i++) {
    ITM_SendChar((*ptr++));
  }
  return len;
}
/* USER CODE END 4 */
```

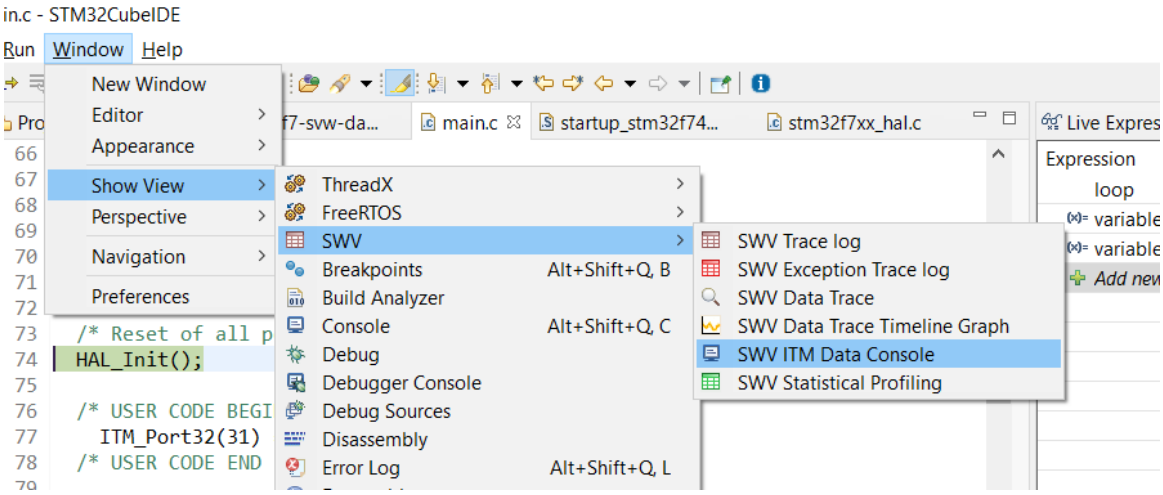
9. Debug the file



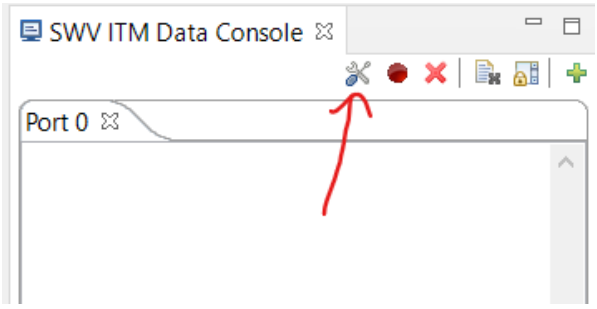
10. In Debug configurations, under Debugger tab, set your **Core Clock** equal to the **HCLK** value found in the clock configurations section (the value which was asked to remember earlier) and click on **Debug.**



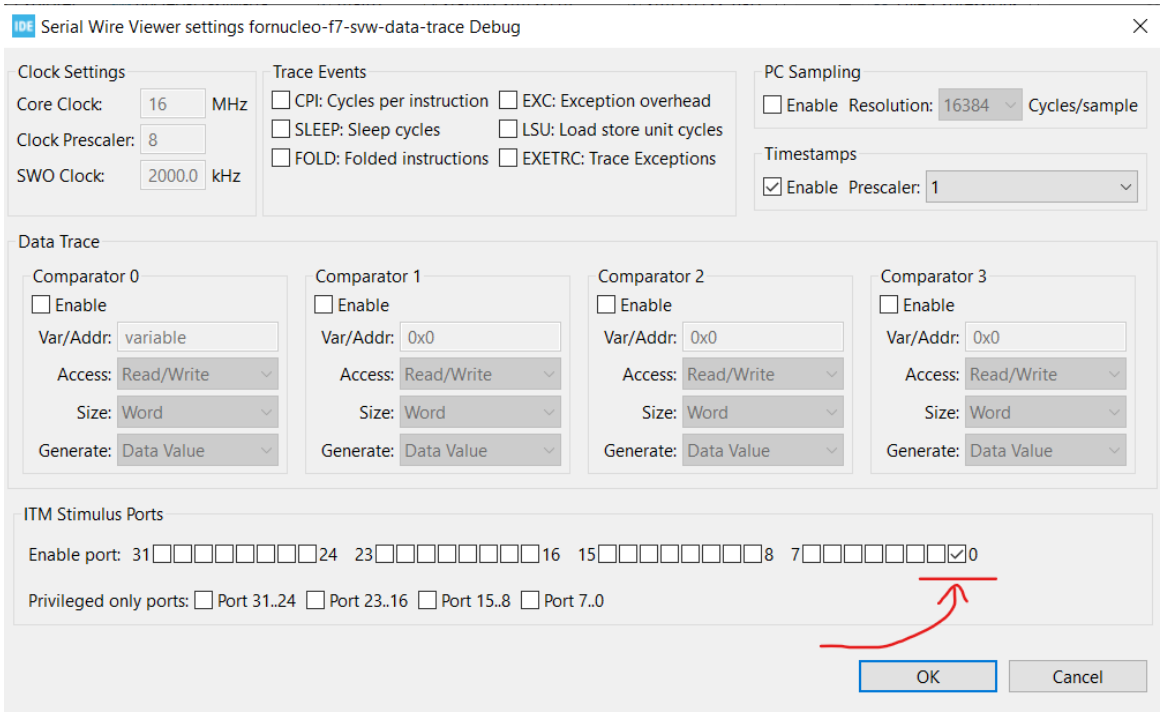
11. Open **SVW ITM Data Console**



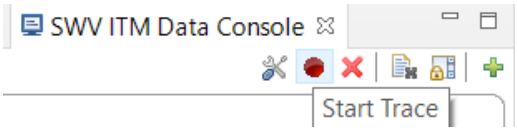
12. Now press on the following icon for configuring SWV ITM Data Console



11. Enable **Port 0** and click OK.



12. Press on the following button to Start trace



13. Now press Resume or F8 and you can see data getting printed in the ITM Data Console.