STM32CubeIDE basics

USART – sending data to terminal

life.augmented

STM32 CubeIDE

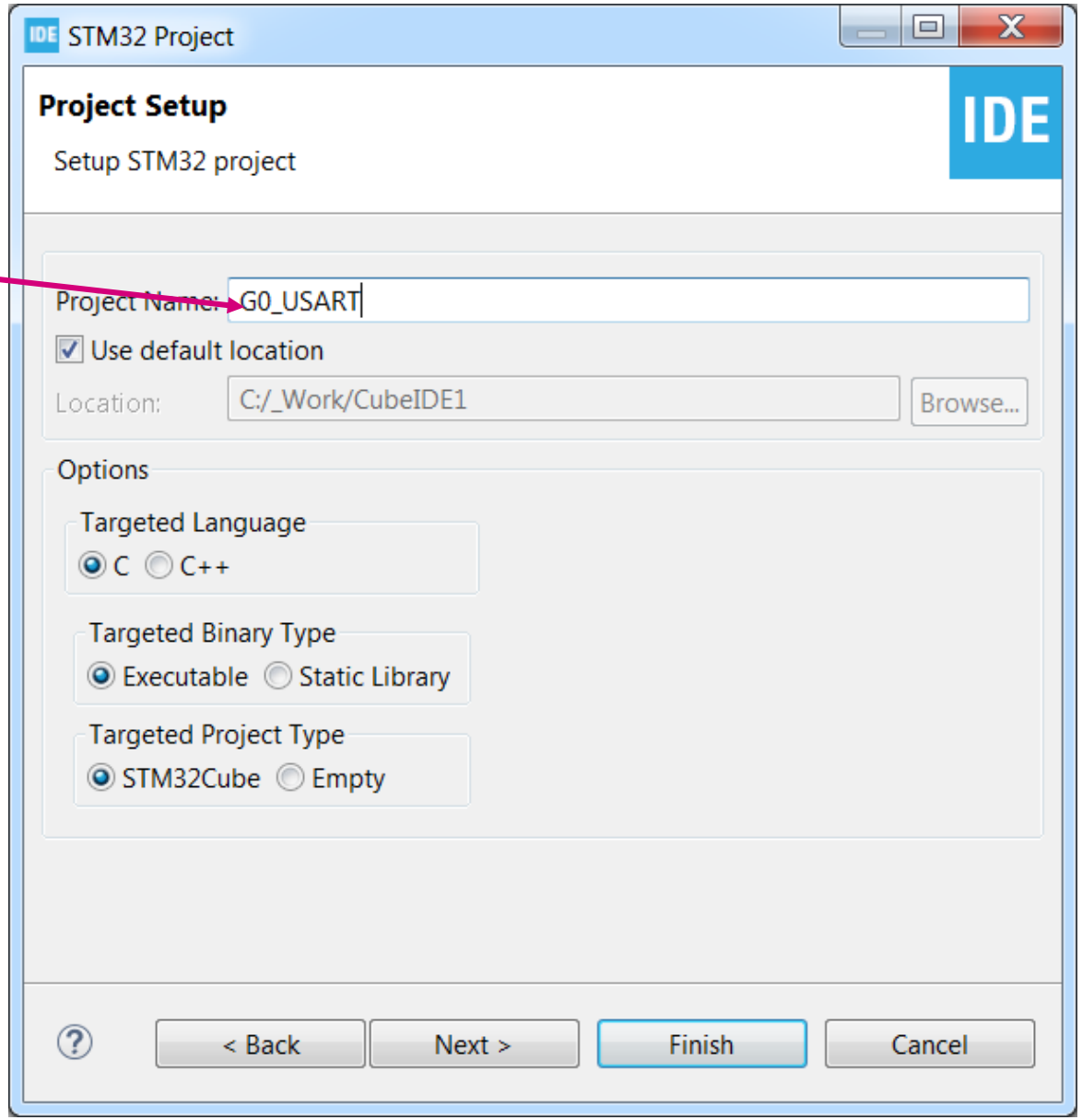
STM32CubeIDE – projects management

Objective:

- Demonstrate how we can configure USART in asynchronous mode to send data
- Demonstrate usage of connection of USART2 (STM32G0 MCU) to STLink on Nucleo board
- Demonstrate how to use build-in terminal application available in debug mode of STM32CubeIDE

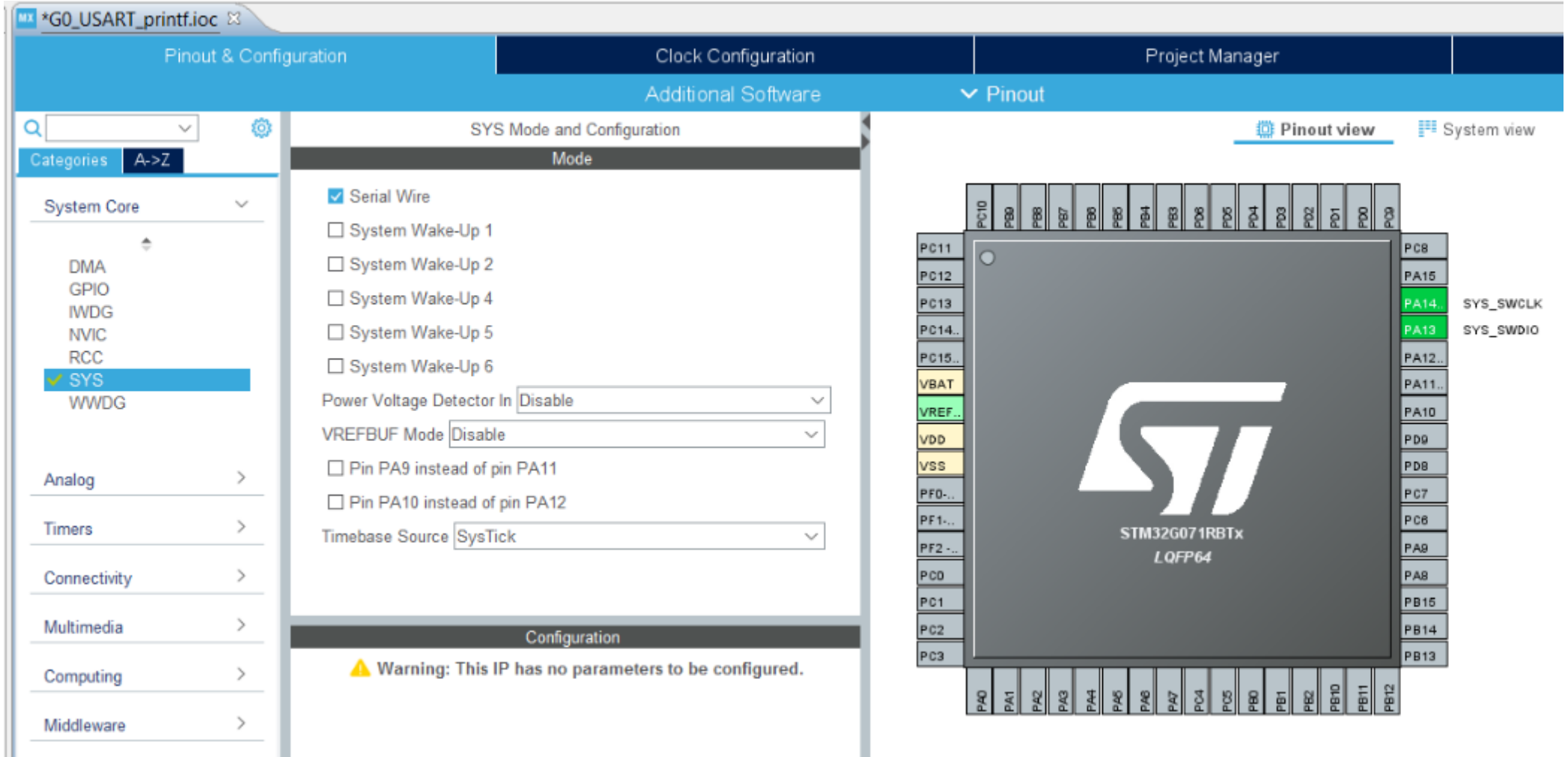
Enter project name

- Specify project name, optionally its location (if different from workspace one)
- Additionally we can specify target language (C or C++), binary type (executable or static library) and project type (generated by STM32CubeMX or an empty one)



Enabling Serial Wire debug interface

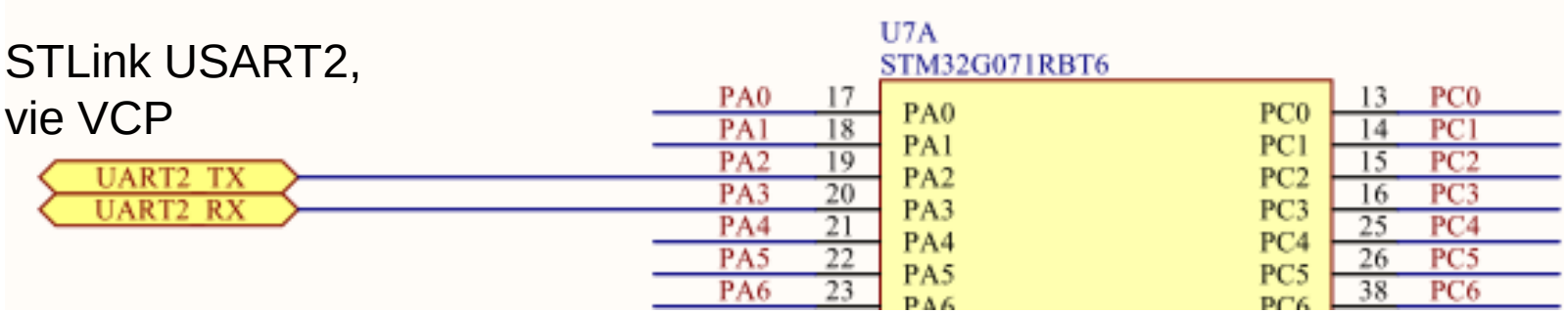
- Select “Serial Wire” from System Core -> SYS peripheral group
- As a result PA13 and PA14 will be assigned to SWD interface



USART2 connection

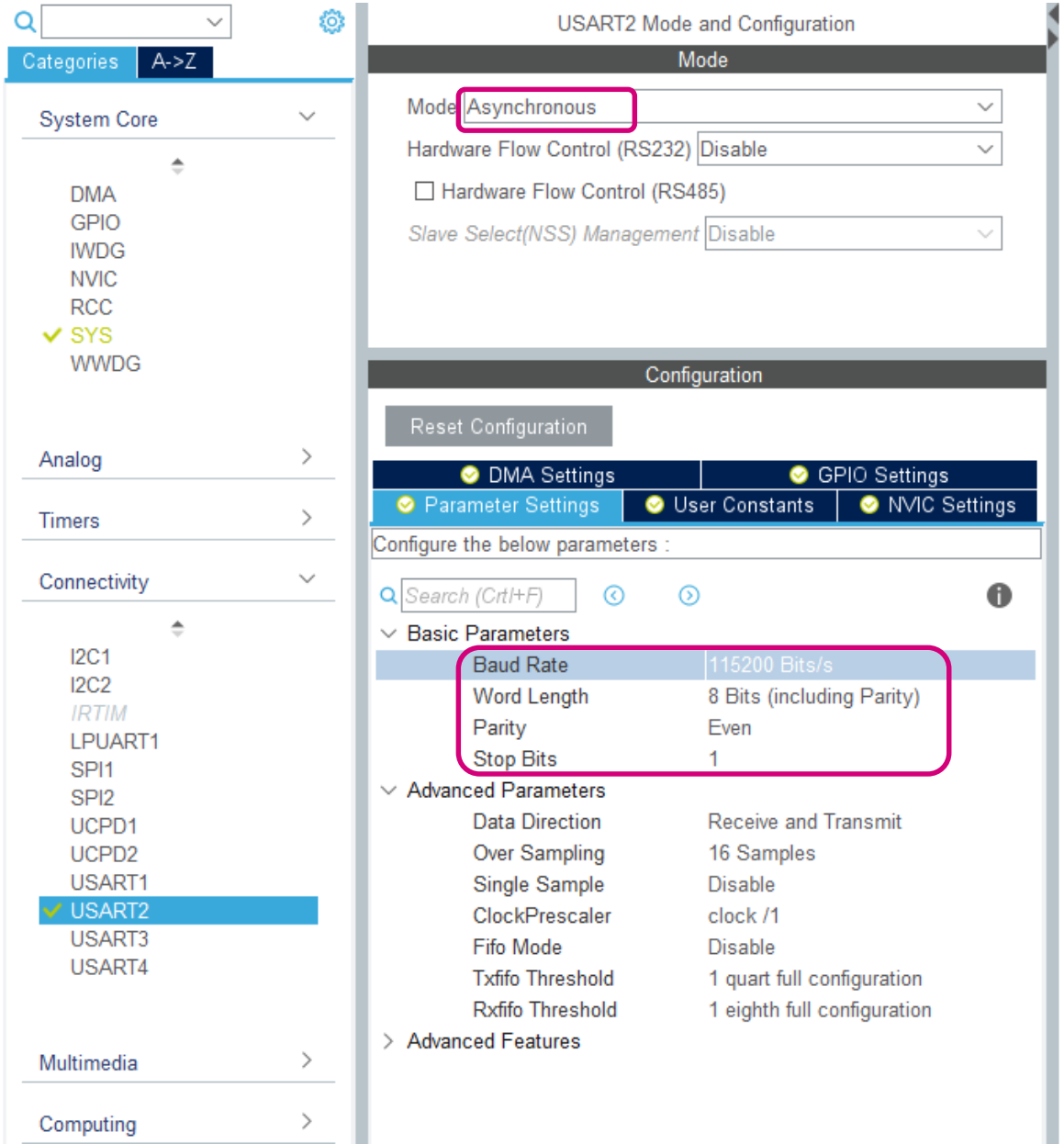
- USART2 (on PA2 and PA3 pins) is connected to USART2 @STLink and thanks to STLink application software it can be available via Virtual COM Port once we connect Nucleo board to PC.
- USART2 settings: Asynchronous mode – 115200 N/8/1, no HW Flow Control, no advanced features

Connection to STLink USART2, then available vie VCP



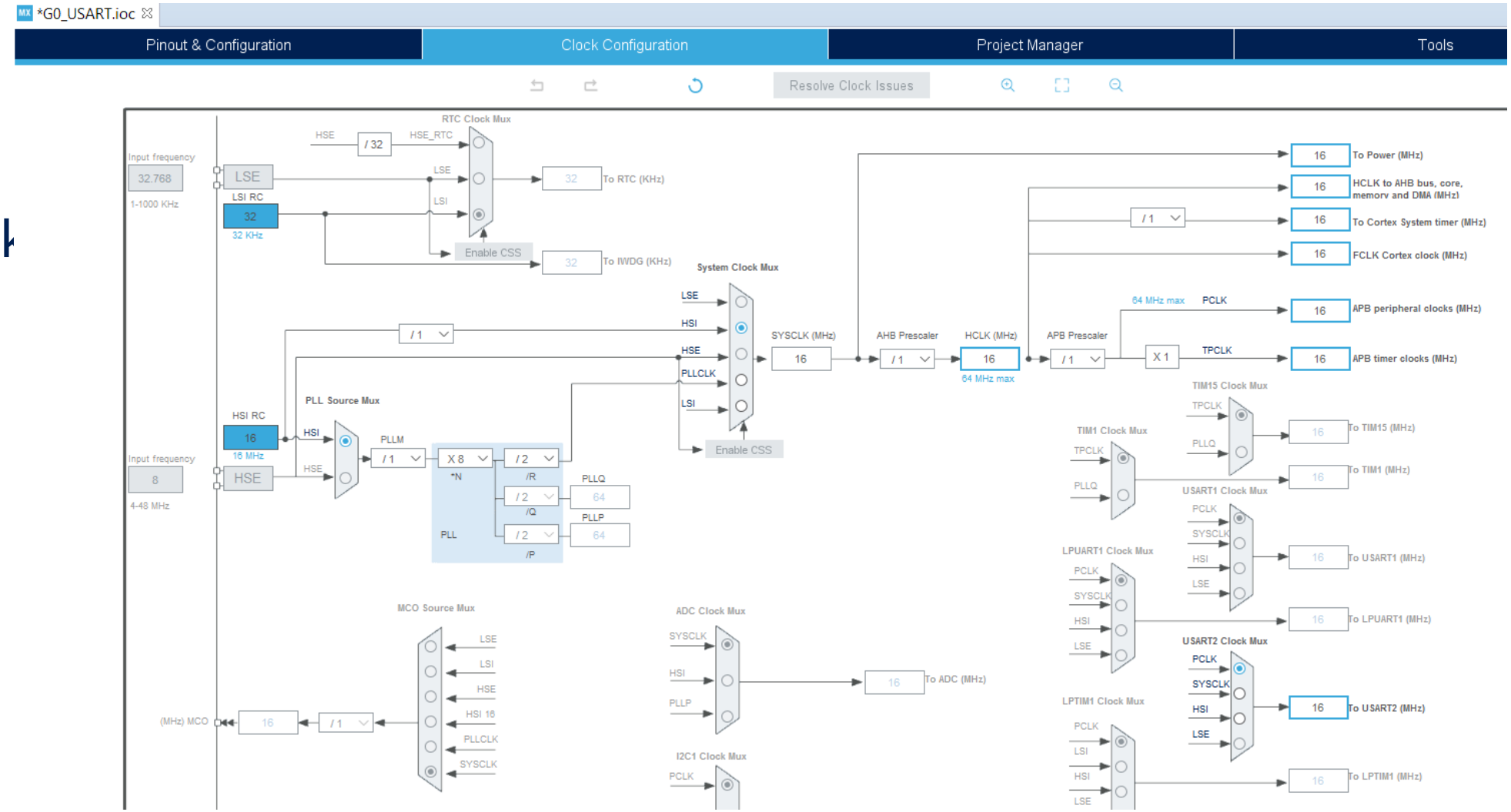
Enabling USART2 in asynchronous mode

- Select USART2-> Connectivity peripheral group
- Configure:
  - Mode: **Asynchronous**
  - Baud Rate: **115200bps**,
  - Word Length: **8 Bits (inc parity)**
  - Parity: **Even**
  - Stop Bits: **1**
- Keep all other settings in default values



Clock Configuration – default settings

- We keep default clock configuration (HSI 16MHz clk source, USART2 clocked by PCLK – 16MHz)



System Core

DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG

Analog

Timers

Connectivity

I2C1
I2C2
IRTIM
LPUART1
SPI1
SPI2
UCPD1
UCPD2
USART1
USART2
USART3
USART4

Multimedia

Computing

Middleware

Mode Asynchronous

Hardware Flow Control (RS232) Disable

Hardware Flow Control (RS485)

Slave Select(NSS) Management Disable

Configuration

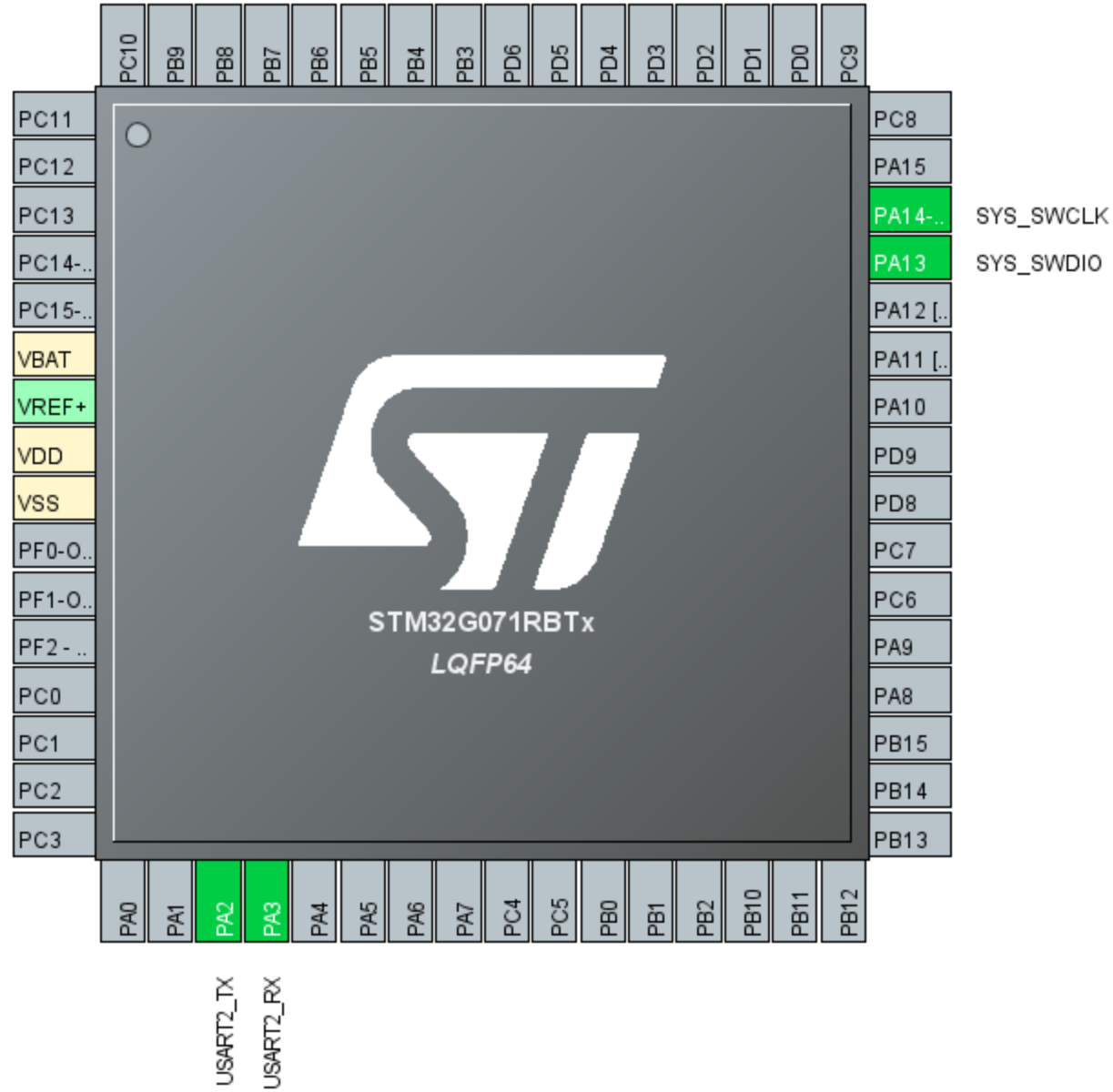
Reset Configuration

DMA Settings | GPIO Settings | Parameter Settings | User Constants | NVIC Settings

Configure the below parameters :

Search (Crtl+F)

Basic Parameters	
Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	Even
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples
Single Sample	Disable
ClockPrescaler	clock /1
Fifo Mode	Disable
Txfifo Threshold	1 quart full configuration
Rxfifo Threshold	1 eighth full configuration
Advanced Features	



Lab USART usage

- Generated main.c file

G0_USART.ioc | main.c

```
66   * @retval int
67   */
68  int main(void)
69  {
70    /* USER CODE BEGIN 1 */
71
72    /* USER CODE END 1 */
73
74
75    /* MCU Configuration--------------------------
76
77    /* Reset of all peripherals, Initializes the F
78    HAL_Init();
79
80    /* USER CODE BEGIN Init */
81
82    /* USER CODE END Init */
83
84    /* Configure the system clock */
85    SystemClock_Config();
86
87    /* USER CODE BEGIN SysInit */
88
89    /* USER CODE END SysInit */
90
91    /* Initialize all configured peripherals */
92    MX_GPIO_Init();
93    MX_USART2_UART_Init();
94    /* USER CODE BEGIN 2 */
95
96    /* USER CODE END 2 */
97
98    /* Infinite loop */
99    /* USER CODE BEGIN WHILE */
100   while (1)
101   {
102     /* USER CODE END WHILE */
103
104     /* USER CODE BEGIN 3 */
105   }
106   /* USER CODE END 3 */
107 }
```

Clock configuration based on “clock configuration” tab settings within Device Configuration perspective

IO lines configuration based on settings from Device Configuration perspective

Configuration of USART2 based on settings from Device Configuration perspective

STM32 CubeIDE

Coding time – test

main.c file

- Define macro to calculate text buffer length

```
/* USER CODE BEGIN PM */
#define COUNTOF(__BUFFER__)   (sizeof(__BUFFER__) / sizeof(*(__BUFFER__)))
```

- Define text buffer

```
/* USER CODE BEGIN PV */
char data[]="Test\n";
```

- Once per second display test message, which should be sent over USART2 to STLink and then via VCP to the terminal

```
/* USER CODE BEGIN WHILE */
  while (1)
  {
       HAL_UART_Transmit(&huart2, (uint8_t *)data, (COUNTOF(data) - 1), 50);
       HAL_Delay(1000);
```

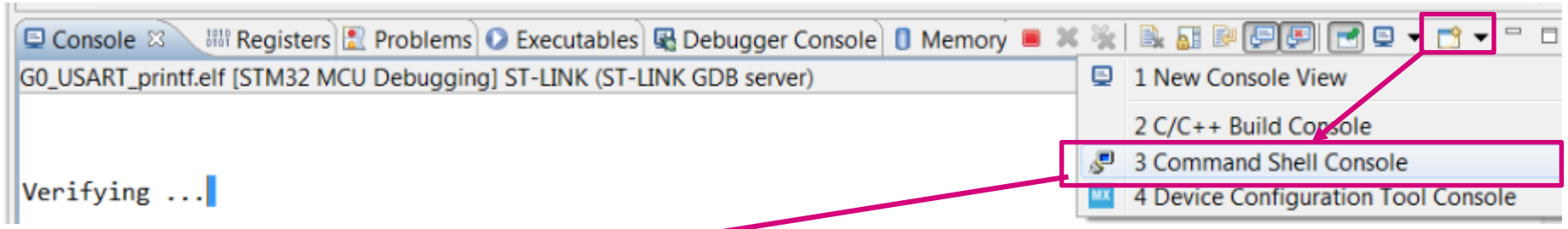
life.augmented

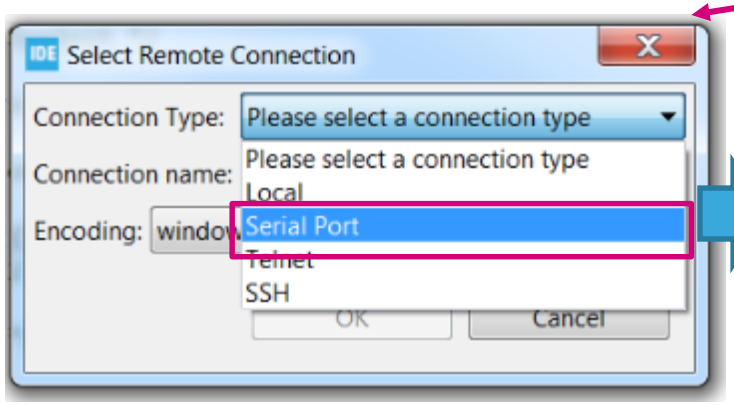
… Let’s check it

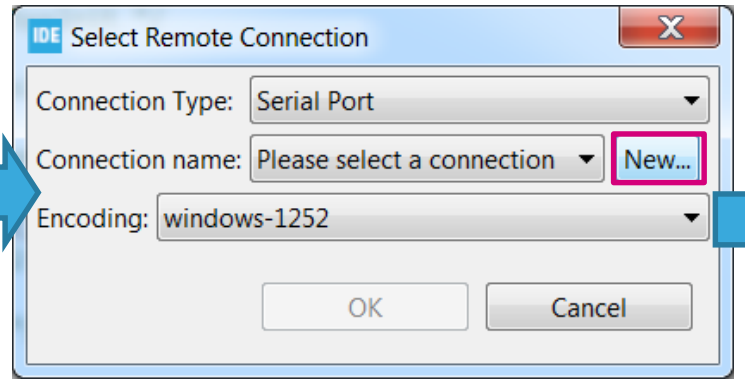
- After all code processing we can build the project, start debug session and run the application

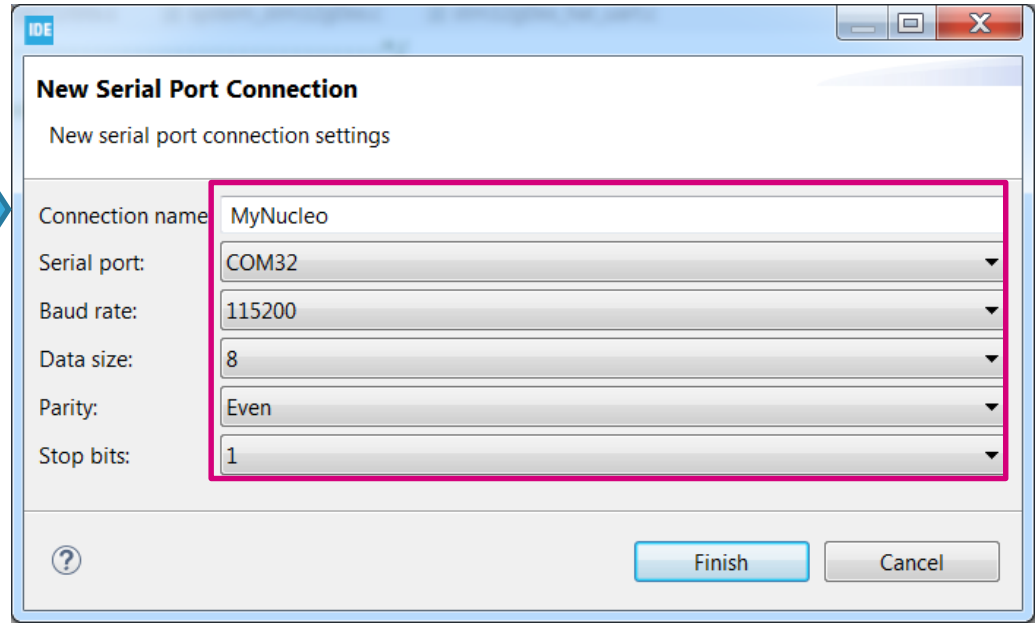
Terminal configuration and run

- Being in debug session we can configure and run a Terminal which could be connected to any of COM ports (including our VCP)





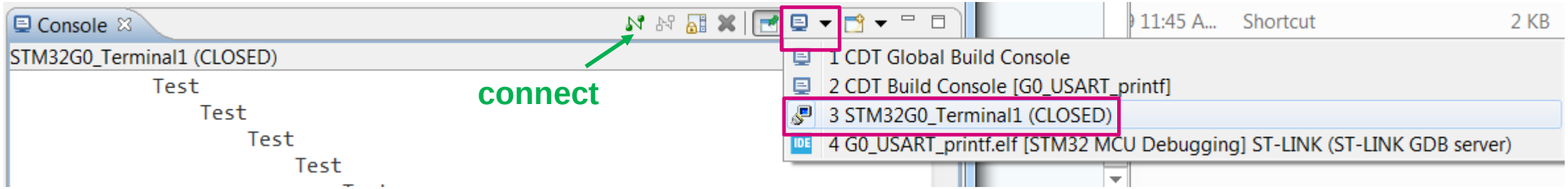




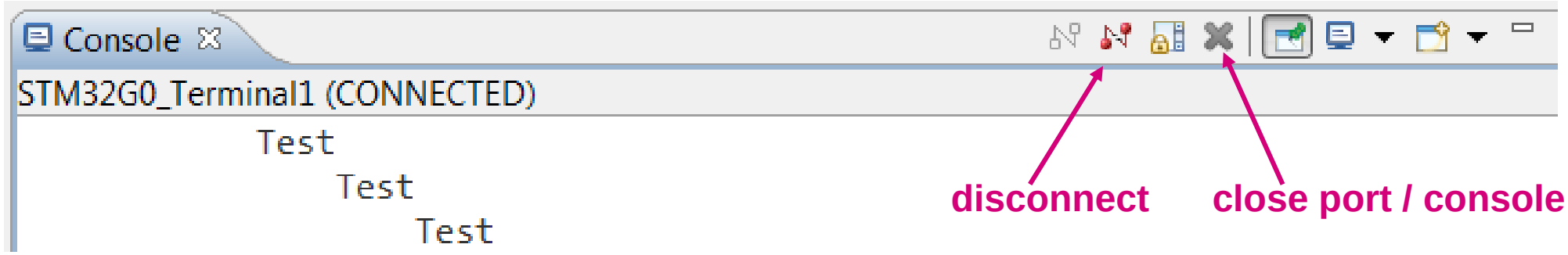
Specify your name, select Serial Port number which is assigned to STLink VCP, set 115200 / 8 / E / 1 transmission settings

Using the terminal within STM32CubeIDE

- Being in debug session select already created terminal connection and press connect button



- To PAUSE the terminal press Disconnect button, to remove press CLOSE button



STM32 CubeIDE

Thank you

/STM32

@ST_World

community.st.com

life.augmented

www.st.com/stm32g0, www.st.com/stm32cubeide