

LAB 09

THE CONSOLE



UNIVERSITY
OF TRENTO

Prof. Davide Brunelli

Dept. of Industrial Engineering – DII

University of Trento, Italy

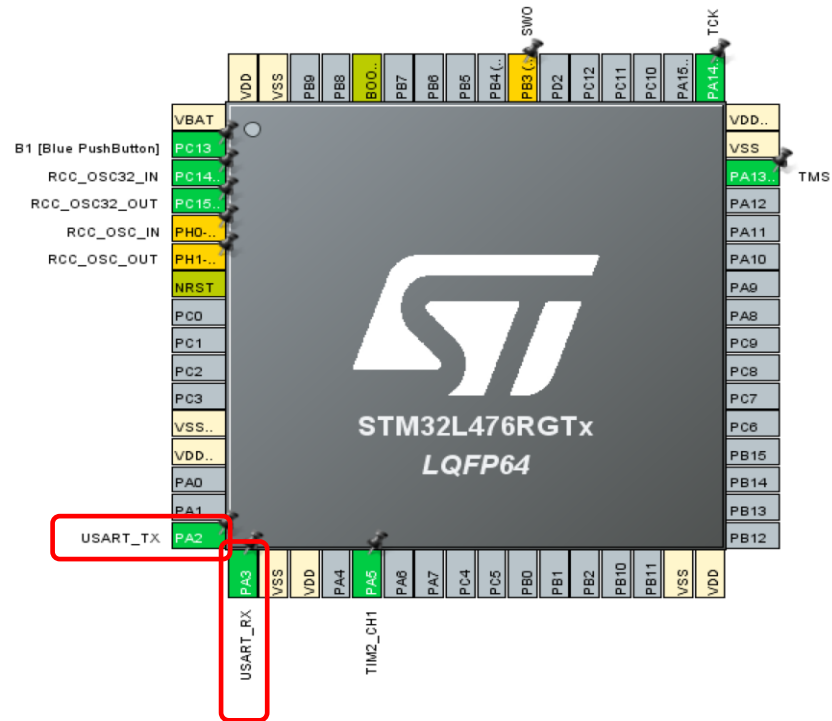
davide.brunelli@unitn.it

STM32 USART

Nucleo boards directly connect
USART RX/TX pins to the
miniUSB port integrated in the
STLink

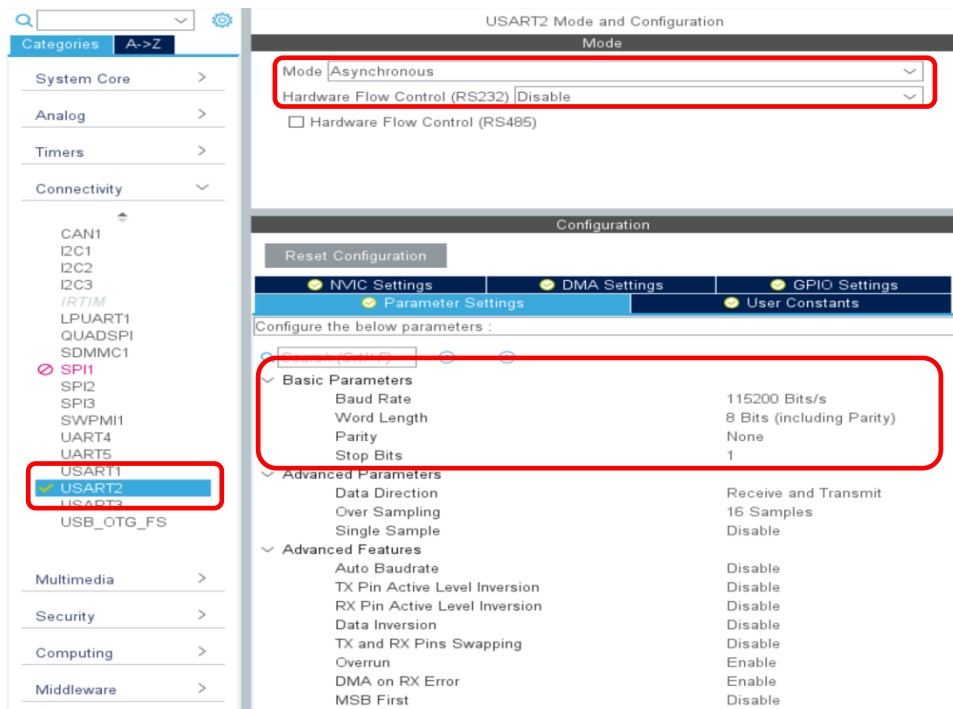
PA2 = TX

PA3 = RX



STM32 USART

Check that the USART2 is enabled.



STM32 CONSOLE

Enable the USART interrupt.

The image displays three configuration windows from STM32CubeIDE:

- USART2 Mode and Configuration:** Shows Mode set to Asynchronous and Hardware Flow Control (RS232) set to Disable. In the Configuration section, the NVIC Interrupt Table is set to Enabled, and the USART2 global interrupt is checked.
- System view:** A summary view showing various system components. The USART2 component is highlighted with a red box.
- NVIC Settings:** A table showing the configuration of various interrupts. The USART2 global interrupt is highlighted with a red box.

USART2 Mode and Configuration - Configuration Section:

Reset Configuration	NVIC Settings	DMA Settings	GPIO Settings	User Constants
	Parameter Settings			
	NVIC Interrupt Table	Enabled	Reemption Priority	Sub Priority
	USART2 global interrupt	✓	0	

NVIC Settings - NVIC Interrupt Table:

Enabled interrupt table	Select for init sequence	Generate IRQ	Call HAL ha
Non maskable interrupt	✓	✓	✓
Hard fault interrupt	✓	✓	✓
Memory management fault	✓	✓	✓
Prefetch fault, memory acces...	✓	✓	✓
Undefined instruction or illegal...	✓	✓	✓
System service call via SWI i...	✓	✓	✓
Debug monitor	✓	✓	✓
Time base: System tick timer	✓	✓	✓
USART2 global interrupt	✓	✓	✓
EXTI line[15:10] interrupts	✓	✓	✓
TIM6 global interrupt, DAC ch...	✓	✓	✓

System view - Connectivity Section:

System Core	Analog	Timers	Connectivity
DMA	ADC1	TIM2	USART2
GPIO		TIM5	
NVIC			
RCC			

System Core Section:

System Core
DMA
GPIO
NVIC
RCC

Remember to double check under “code generation” if IRQ generation is enabled

STM32 CONSOLE

Once everything is configured and the code generated, we have to add the callback called when the USART receive a character.

In the **callback**, we take care of echoing back user input and storing it inside a buffer (defined as global variables).

```
#define MAXBUFSIZE 100
uint8_t RxBuf [MAXBUFSIZE];
uint8_t TxBuf [MAXBUFSIZE];

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
    if(RxBuf[0] == '\r'){ // Check new line
        HAL_UART_Transmit(&huart2, (uint8_t *)"\r\n", sizeof("\r\n"),10);
        //cmd_check(); //TO BE IMPLEMENTED!!
    }
    else if(RxBuf[0] == '\b'){ //Check back space
        HAL_UART_Transmit(&huart2, (uint8_t *)"\b\b", sizeof("\b\b"),10);
        TxBuf[--char_counter] = NULL;
    }else{ //Otherwise store inside our buffer
        TxBuf[char_counter++] = RxBuf[0];
        HAL_UART_Transmit(&huart2, (uint8_t *) RxBuf, sizeof(RxBuf),10); //Echoing
    }
    HAL_UART_Receive_IT(&huart2, RxBuf, 1); // Restart UART in Interrupt mode
}
```

STM32 CONSOLE

Finally start the UART in interrupt mode.



```
/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART2_UART_Init();
MX_TIM6_Init();
MX_ADC1_Init();
MX_TIM2_Init();
/* USER CODE BEGIN 2 */
HAL_UART_Transmit(&huart2, welcomeBuff, sizeof(welcomeBuff), 5000);
HAL_UART_Receive_IT(&huart2 , RxBuf, 1);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

}
/* USER CODE END 3 */
```

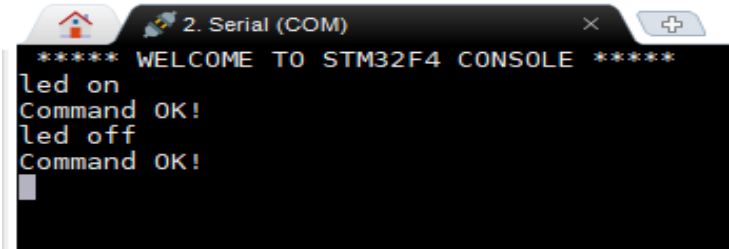


CONSOLE – INTERACT WITH THE EMBEDDED SYSTEM

6D ORIENTATION ESTIMATOR

Create a console program, that recognize a set of commands and execute them on board.

- Switch LED ON and OFF
- Toggle LED
- Start LED blinking
- Get internal temperature



```
***** WELCOME TO STM32F4 CONSOLE *****
led on
Command OK!
led off
Command OK!
```

Hint: you can use `strcmp((char *)TxBuf, command)` to compare two strings