USART with STM32CubeMX and HAL

This tutorial will demonstrate how to utilize STM32CubeMX tool to initialize peripherals, build and generate C code using HAL libraries.

After this tutorial you will be able to:

- Create and configure STM32CubeMX project and generate initialization code
- Program and use HAL functions to take input and send messages to the terminal

Hardware:

- Nucleo-L476RG board(64-pin),available at: www.st.com/en/evaluation-tools/nucleo-l476rg.html
- Standard-A -to- Mini USB cable

Literature:

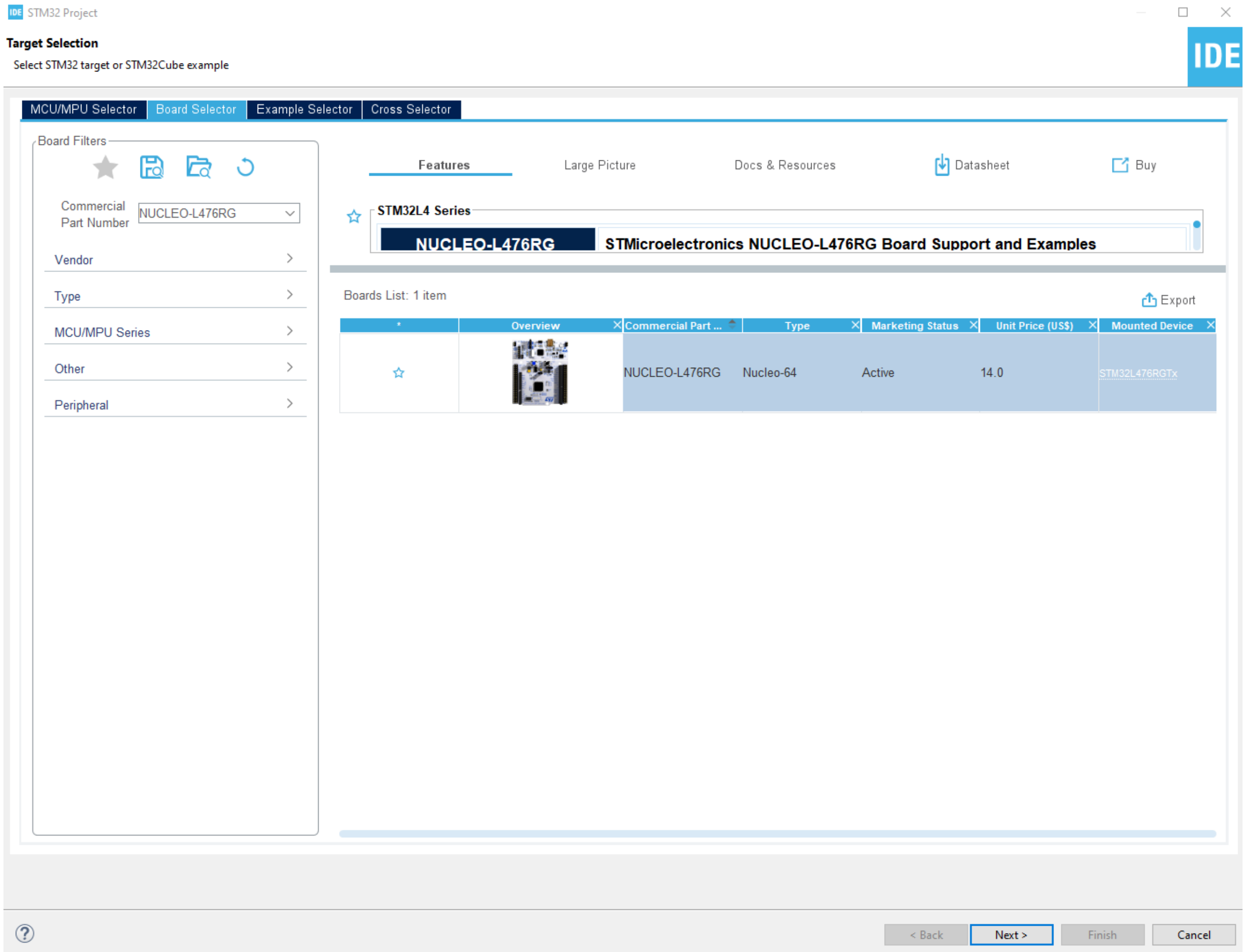
- STM32L476xx Datasheet
- UM1724 User manual STM32 Nucleo-64 boards
- UM1884 Description of STM32L4/L4+ HAL and low-layer drivers
- UM1718 User manual STM32CubeMX for STM32 configuration and initialization C code generation

Stages

- 1: Create New Project with STM32CubeMX
- 2: Pinout Configuration
- 3: Clock Configuration
- 4: Configure project and Generate Source Code
- 5: Edit main.c
- 6: Build Project
- 7: Debug the Project

1: CREATE NEW PROJECT USING STM32CUBEMX:

- Open STM32CubeIDE
- Click *File -> New -> STM32 Project*. A target selection window will open.
- From Board Selector type *Nucleo-L476RG.* Select the board and click next.
- Name your project “Nucleo_L476RG_USART” and click Finish.
- Answer “Yes” to “Initialize all peripherals with their default mode?” popup.



Target Selection

Select STM32 target or STM32Cube example

Commercial Part Number: NUCLEO-L476RG

STM32L4 Series

NUCLEO-L476RG **STMicroelectronics NUCLEO-L476RG Board Support and Examples**

Boards List: 1 item

*	Overview	Commercial Part ...	Type	Marketing Status	Unit Price (US$)	Mounted Device
☆		NUCLEO-L476RG	Nucleo-64	Active	14.0	STM32L476RGTx

2: Pinout Configuration

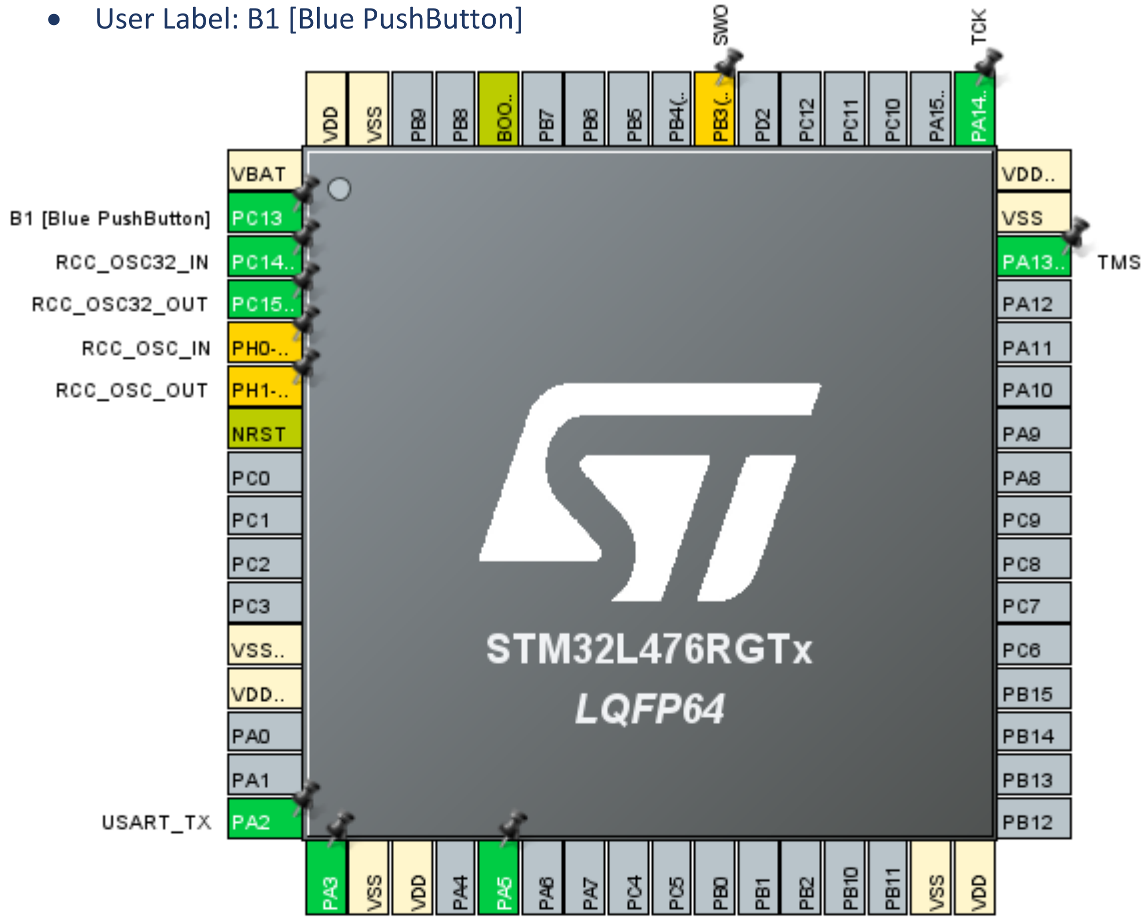
Right click PC13 and and select GPIO_Input. Right click PA5 and select GPIO_Output.

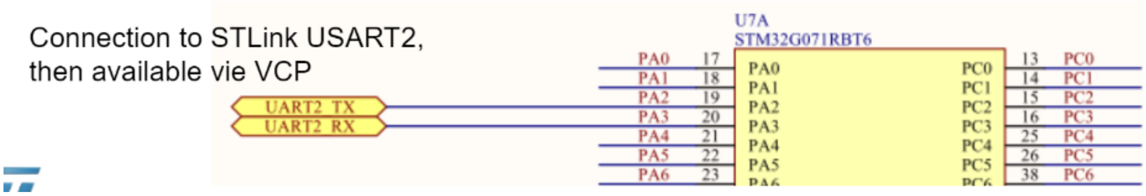
Under System Core->GPIO select PA5.

- GPIO output level : Low
- GPIO mode: Output Push Pull
- GPIO Pull-up/Pull down: No pull-up and no pull-down
- Max output speed: Low
- User label: LD2 [green Led]

Under System Core -> GPIO select PC13

- GPIO mode: External Interrupt Mode with Falling edge trigger detection
- GPIO Pull-up/Pull-down: No pull-up and no pull-down
- User Label: B1 [Blue PushButton]

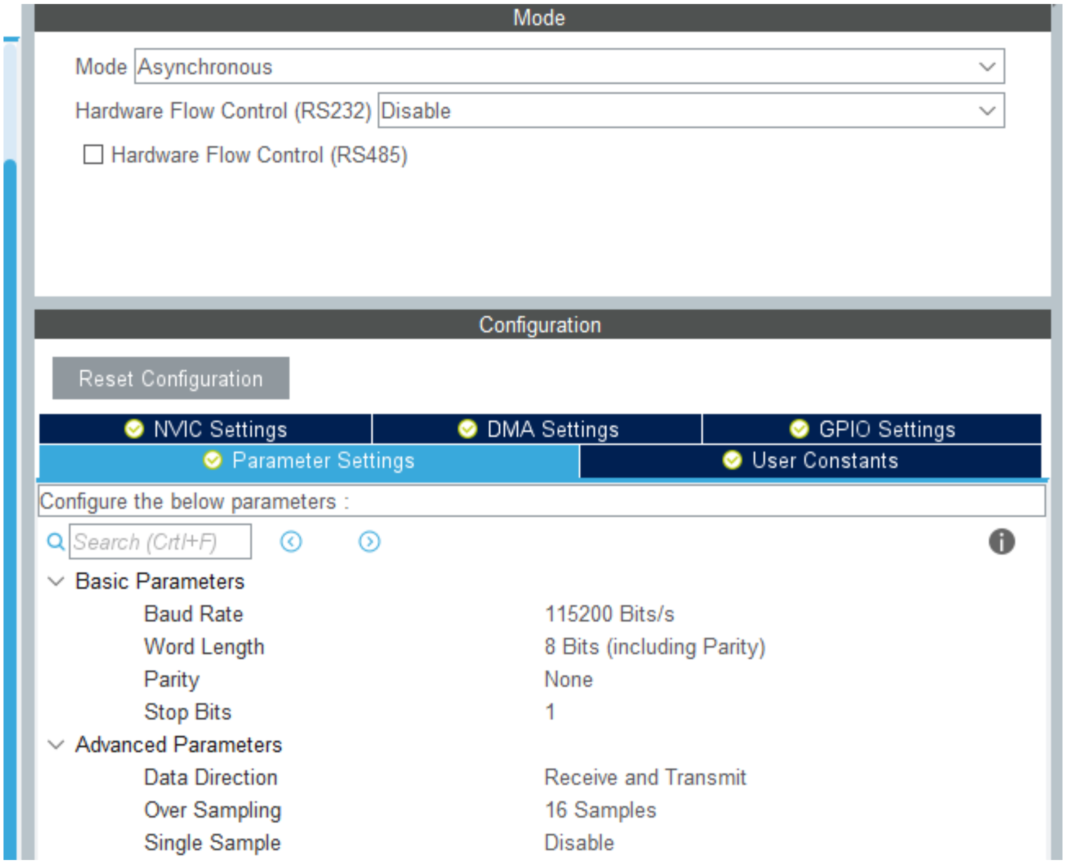



Connection to STLink USART2, then available vie VCP

Go to Connectivity->USART2

- Mode: Asynchronous
- Baud Rate: 115200
- Word Length 8 bits (including Parity)
- Parity: Even
- Stop Bits: 1

Leave all other settings as default.



Mode: Asynchronous

Hardware Flow Control (RS232): Disable

□ Hardware Flow Control (RS485)

Basic Parameters

Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1

Advanced Parameters

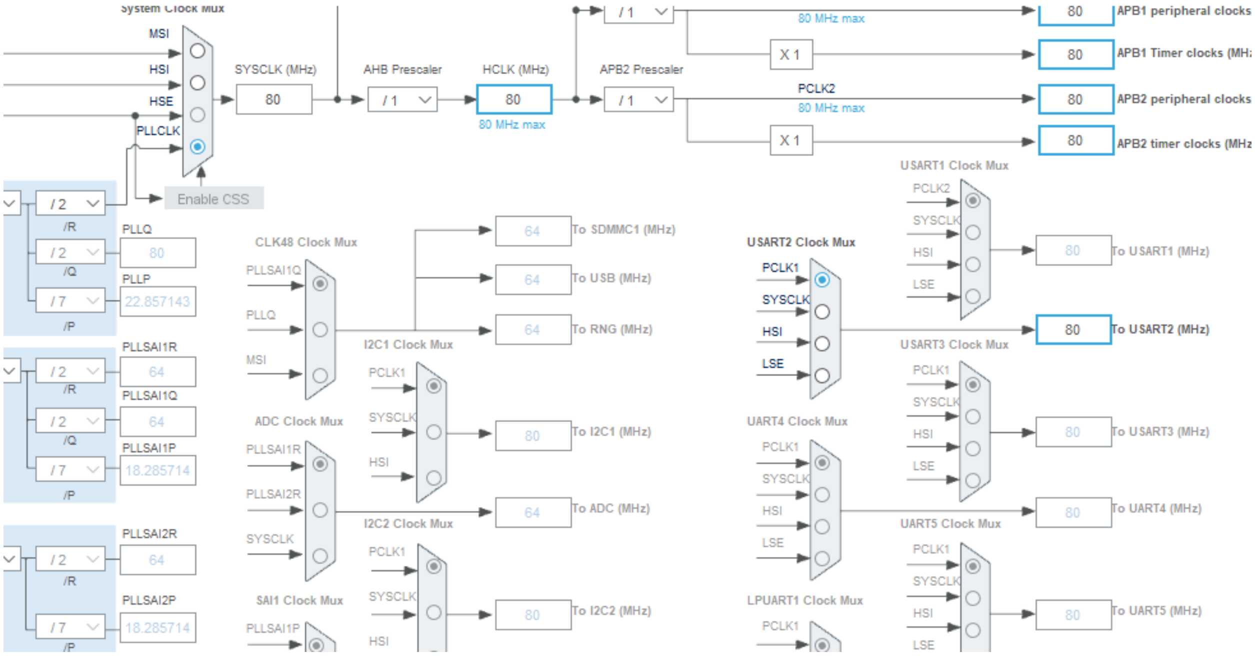
Data Direction	Receive and Transmit
Over Sampling	16 Samples
Single Sample	Disable

3. CLOCK CONFIGURATION

In the clock configuration tab you can see that STM32CubeMX automatically configures the internal oscillator in the clock system with PLL @80MHz. The HIS is selected as the PLL source and the PLLCLK is selected in the system clock mux.

HCLK is set to 80 MHz.

USART2 clocked by PLCK1.



SYSCLK (MHz) 80
AHB Prescaler /1
HCLK (MHz) 80
APB2 Prescaler /1
PCLK2
/1 → 80 APB1 peripheral clocks
X 1 → 80 APB1 Timer clocks (MH
/1 → 80 APB2 peripheral clocks
X 1 → 80 APB2 timer clocks (MHz

/2 /R
/2 /Q PLLQ 80
/7 /P PLLP 22.857143
/2 /R PLLSAI1R 64
/2 /Q PLLSAI1Q 64
/7 /P PLLSAI1P 18.285714
/2 /R PLLSAI2R 64
/7 /P PLLSAI2P 18.285714

64 To SDMMC1 (MHz)
64 To USB (MHz)
64 To RNG (MHz)
80 To I2C1 (MHz)
64 To ADC (MHz)
80 To I2C2 (MHz)
80 To USART1 (MHz)
80 To USART2 (MHz)
80 To USART3 (MHz)
80 To UART4 (MHz)
80 To UART5 (MHz)

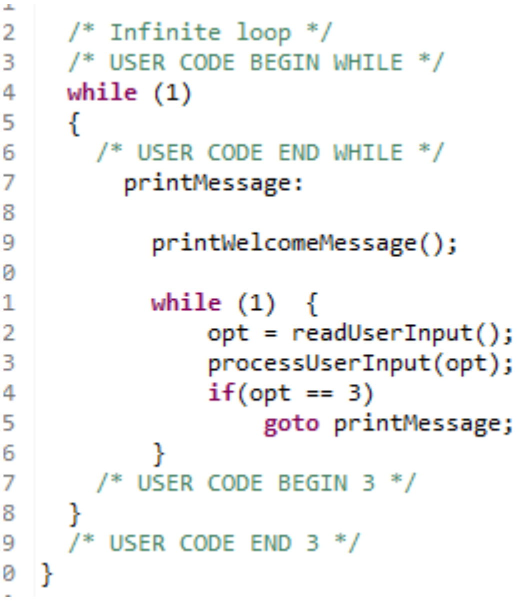
4: GENERATE CODE

We can now generate code. Click File->Save. You will be asked to generate code, press yes.

Under the project explorer navigate *to Core->Src->main.c*.

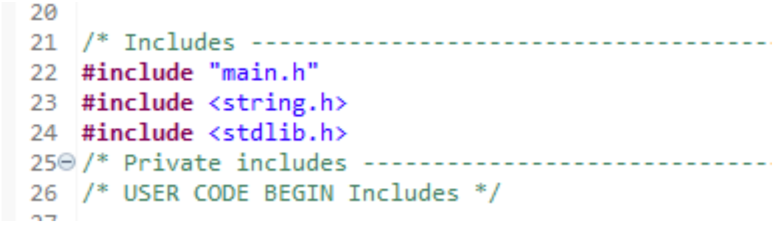
5: EDIT main.c

First let us take a look at our driver code:



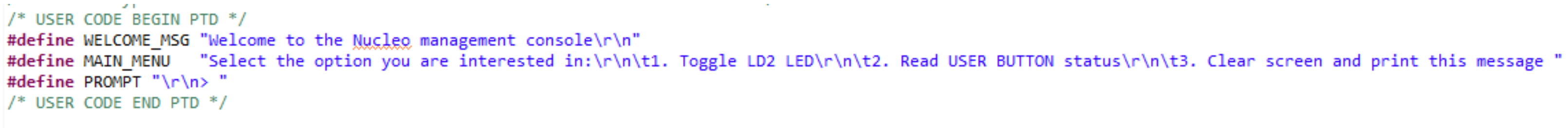
```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
      printMessage:

        printWelcomeMessage();

        while (1)  {
            opt = readUserInput();
            processUserInput(opt);
            if(opt == 3)
                goto printMessage;
        }
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

For this demonstration, we want to print a welcome message. Then we want to read user input and process it. If user inputs 3, we go back to the printMessage label. Make sure to have these lines at the top of the main.c file:



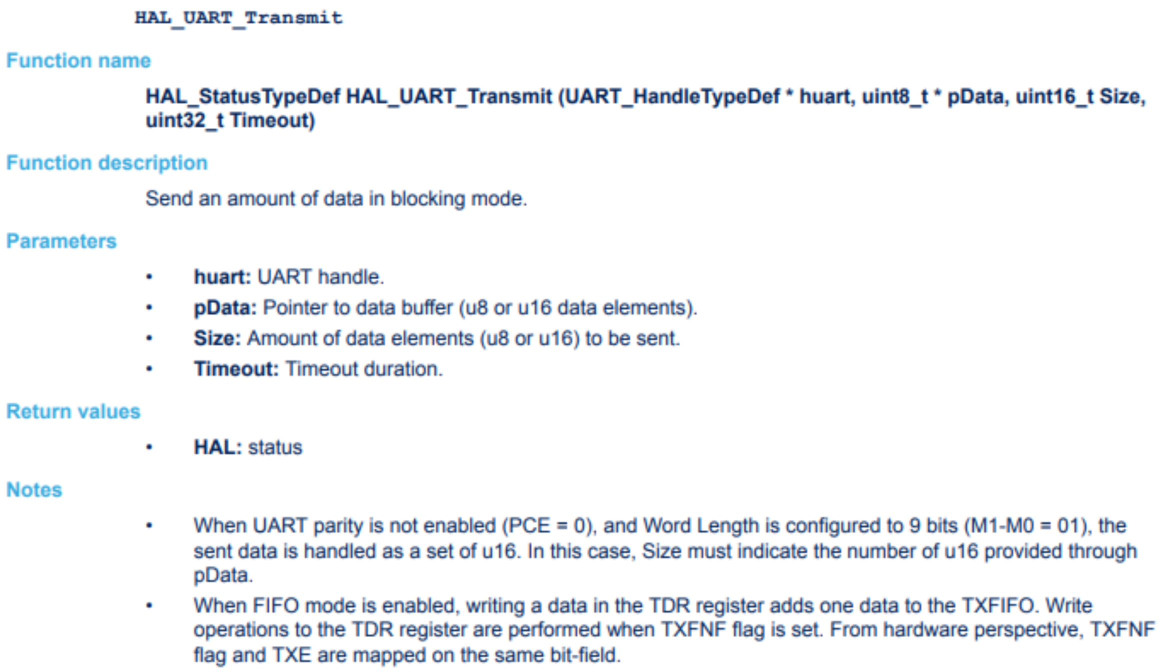
```
/* Includes ------------------------------------
#include "main.h"
#include <string.h>
#include <stdlib.h>
/* Private includes ----------------------------
/* USER CODE BEGIN Includes */
```

Define these strings for later use:



```
/* USER CODE BEGIN PTD */
#define WELCOME_MSG "Welcome to the Nucleo management console\r\n"
#define MAIN_MENU   "Select the option you are interested in:\r\n\t1. Toggle LD2 LED\r\n\t2. Read USER BUTTON status\r\n\t3. Clear screen and print this message "
#define PROMPT "\r\n> "
/* USER CODE END PTD */
```

To read input and output to the terminal we will be utilizing these functions found in the UM1884 manual.



`HAL_UART_Transmit`

Function name

HAL_StatusTypeDef HAL_UART_Transmit (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size, uint32_t Timeout)

Function description

Send an amount of data in blocking mode.

Parameters

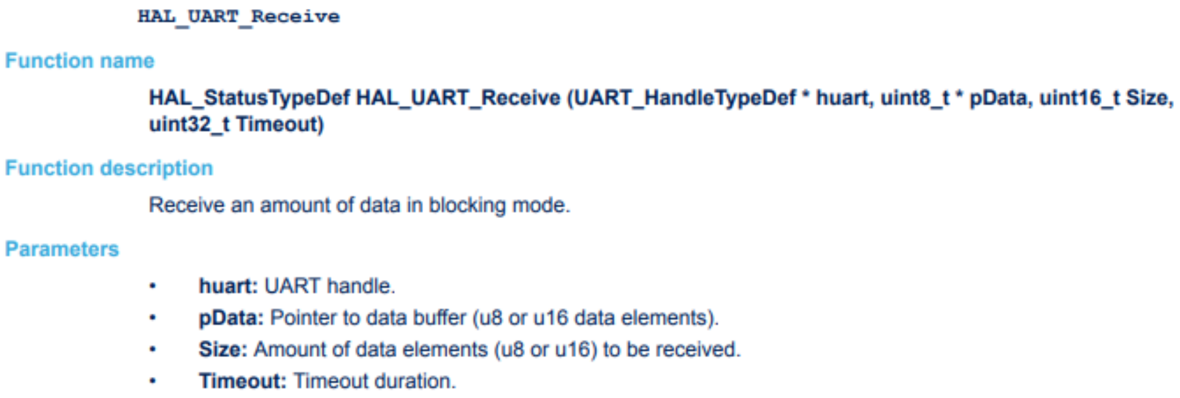
- **huart:** UART handle.
- **pData:** Pointer to data buffer (u8 or u16 data elements).
- **Size:** Amount of data elements (u8 or u16) to be sent.
- **Timeout:** Timeout duration.

Return values

- **HAL:** status

Notes

- When UART parity is not enabled (PCE = 0), and Word Length is configured to 9 bits (M1-M0 = 01), the sent data is handled as a set of u16. In this case, Size must indicate the number of u16 provided through pData.
- When FIFO mode is enabled, writing a data in the TDR register adds one data to the TXFIFO. Write operations to the TDR register are performed when TXFNF flag is set. From hardware perspective, TXFNF flag and TXE are mapped on the same bit-field.



`HAL_UART_Receive`

Function name

HAL_StatusTypeDef HAL_UART_Receive (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size, uint32_t Timeout)

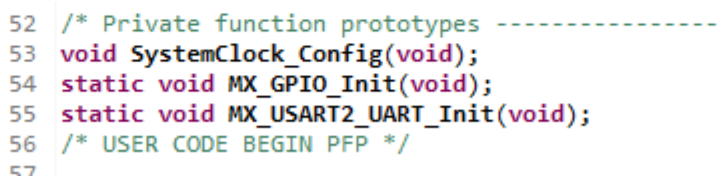
Function description

Receive an amount of data in blocking mode.

Parameters

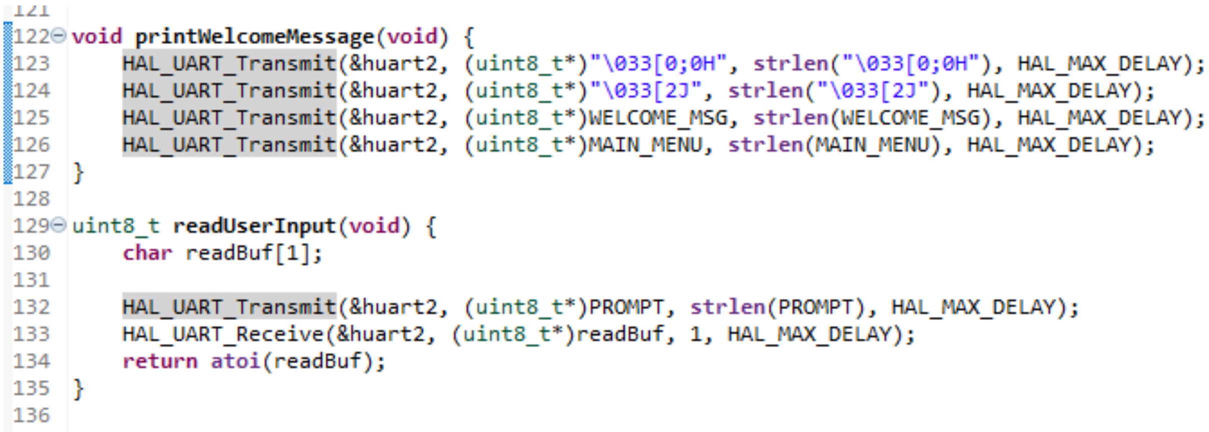
- **huart:** UART handle.
- **pData:** Pointer to data buffer (u8 or u16 data elements).
- **Size:** Amount of data elements (u8 or u16) to be received.
- **Timeout:** Timeout duration.

Next write our function prototypes:



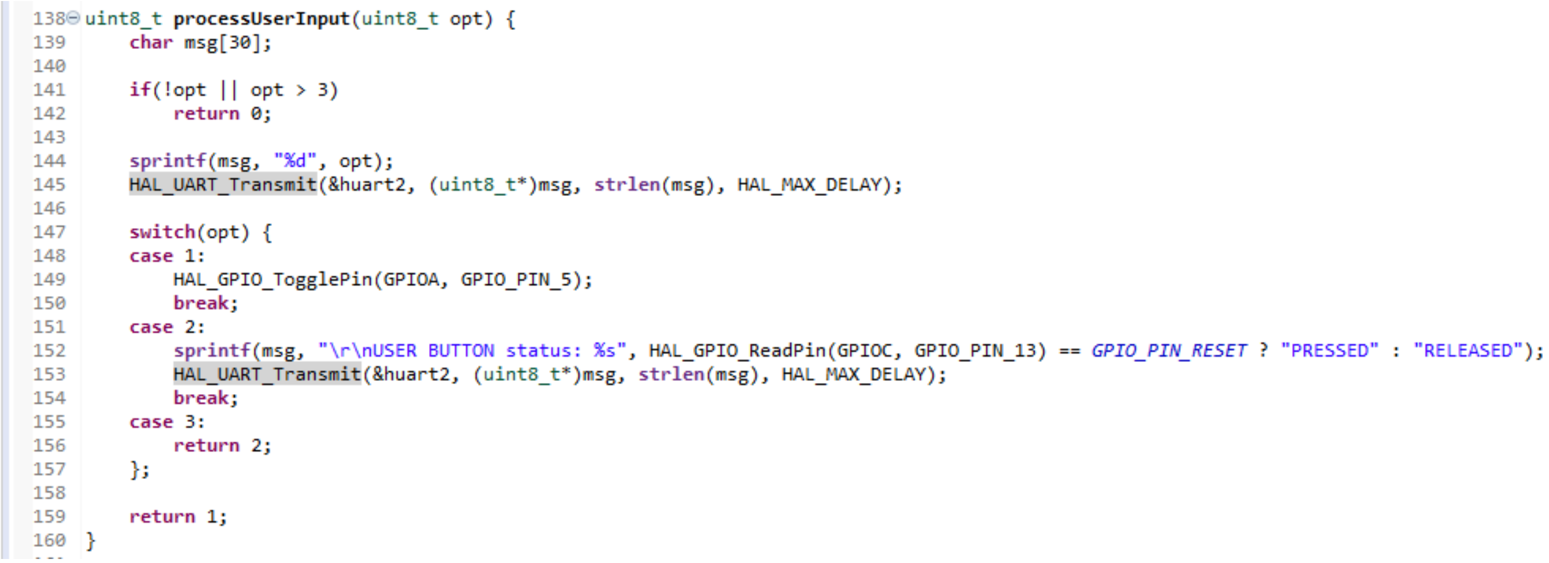
```
/* Private function prototypes -----------------
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
```

Welcome Message and Read Input Function:



```
void printWelcomeMessage(void) {
    HAL_UART_Transmit(&huart2, (uint8_t*)"\033[0;0H", strlen("\033[0;0H"), HAL_MAX_DELAY);
    HAL_UART_Transmit(&huart2, (uint8_t*)"\033[2J", strlen("\033[2J"), HAL_MAX_DELAY);
    HAL_UART_Transmit(&huart2, (uint8_t*)WELCOME_MSG, strlen(WELCOME_MSG), HAL_MAX_DELAY);
    HAL_UART_Transmit(&huart2, (uint8_t*)MAIN_MENU, strlen(MAIN_MENU), HAL_MAX_DELAY);
}

uint8_t readUserInput(void) {
    char readBuf[1];

    HAL_UART_Transmit(&huart2, (uint8_t*)PROMPT, strlen(PROMPT), HAL_MAX_DELAY);
    HAL_UART_Receive(&huart2, (uint8_t*)readBuf, 1, HAL_MAX_DELAY);
    return atoi(readBuf);
}
```

Process User Input Function:



```
uint8_t processUserInput(uint8_t opt) {
    char msg[30];

    if(!opt || opt > 3)
        return 0;

    sprintf(msg, "%d", opt);
    HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);

    switch(opt) {
    case 1:
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
        break;
    case 2:
        sprintf(msg, "\r\nUSER BUTTON status: %s", HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == GPIO_PIN_RESET ? "PRESSED" : "RELEASED");
        HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
        break;
    case 3:
        return 2;
    };

    return 1;
}
```

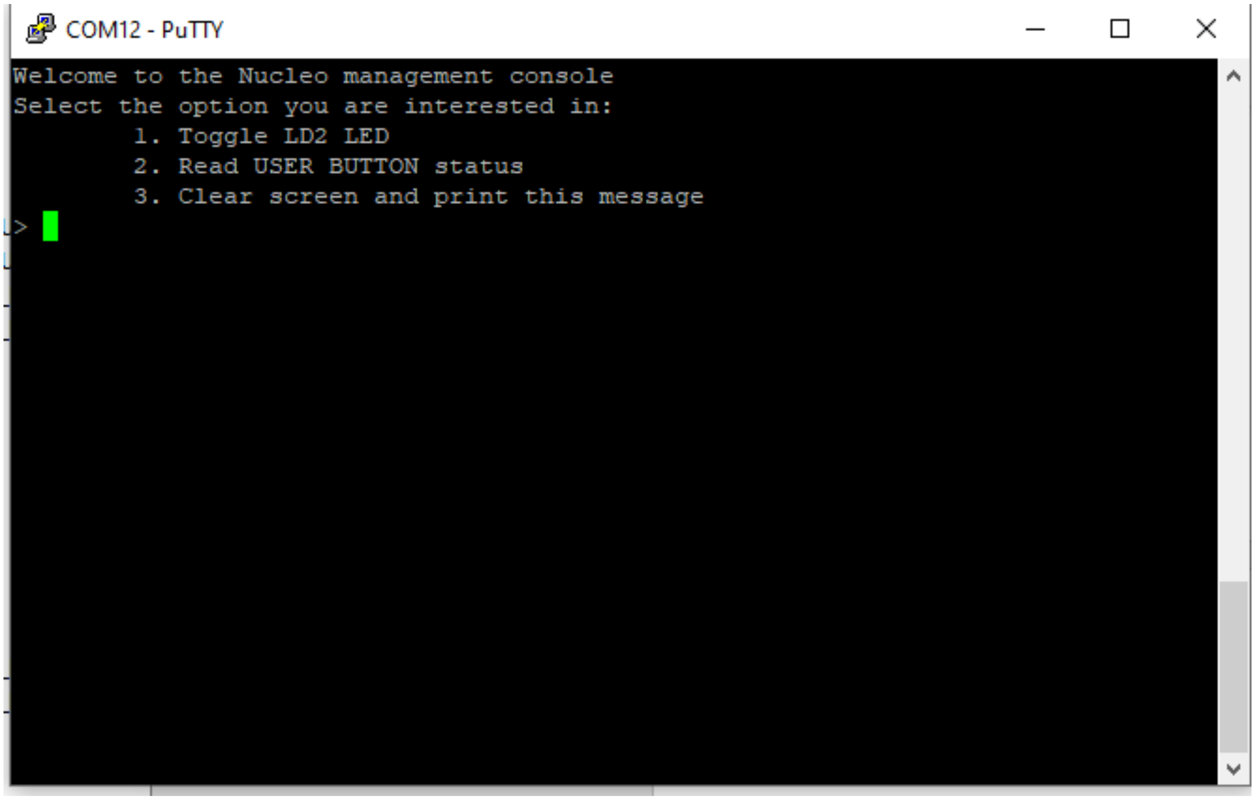
6: BUILD THE PROJECT

Connect your USB cable from the computer to your Nucleo Board. Right click the project from the project explorer and click “Build project” to compile the project.

7: DEBUG THE PROJECT

Click on the Debug toolbar icon to start the debug session. Another way to debug is to *Run->Debug* . 🐞

Click the Resume icon to continue the execution. Open a serial monitor such as Putty and select your com port with the appropriate baud rate (115200). We can now send input from our keyboard to communicate with the board.



```
Welcome to the Nucleo management console
Select the option you are interested in:
        1. Toggle LD2 LED
        2. Read USER BUTTON status
        3. Clear screen and print this message
> 
```