ECEN 3320-002

Assembly Language Programming

Lab Assignment # 3.3

Cameron Biniamow

University of Nebraska-Lincoln

Department of Electrical and Computer Engineering

Peter Kiewit Institute

Due: 12/04/2020

Summary

Lab 3.3 , Introduction to STM32F103 USART, introduces the STM32CubeIDE that will be used for configuration and programming to access USART capabilities. Additionally, Lab 3.3 introduces how to write a program for the ARM Cortex-M3 using C language. Following the completion of the program, the STM32F103C8 will be programmed to send and receive "Hello World!" or a unique string using the USART channel.

Background

STM32CubeIDE is a C/C++ development platform with code generation, code compilation, and debug features for STM32 microcontrollers. The STM32CubeIDE software allows for selection of a desired STM32 microcontroller and will generate all necessary initialization code in respect to the chosen configuration settings. Furthermore, STM32CubeIDE provides access to the HAL (Hardware Abstraction Library), which allows for portability of code between different STM32 microcontrollers.

USART (Universal Synchronous Asynchronous Receiver Transmitter) is used in this lab through a loopback connection, which connects the transmit and receive pins on the same device for simple testing and debugging. The USART of the STM32F103C8 will be used in this lab to transfer predefined strings on the device. Successful operation of the lab will be determined through debugging the program within the STM32CubeIDE and observing the changes in the USART receive buffer.

Procedure

Completion of this lab requires the installation of the STM32CubeIDE software provided by ST Electronics. Upon final installation, the STM32CubeIDE is opened and a workspace location is selected. The provided default workspace location is selected and a prompt to create a new project will appear. A new project can also be created through the following steps:

- File >> New >> STM32 Project

Following the creation of a new project, the Target Selection window appears. Within the Target Selection window, "STM32F103C8" is searched for in the Part Number box. On the right half of the Target Selection window, the STM32F103C8 controller is selected and the Next button is clicked. The new project is titled, and the Finish button is selected.

Once the new project is created, controller configuration may begin through the Device Configuration window. The device is initially configured to allow for debugging by performing the following steps:

1. Pinout & Configuration >> System Core >> SYS
2. Debug >> Serial Wire
3. Timebase Source >> SysTick

Still under the Pinout & Configuration tab, the USART settings are configured through the following steps:

1. Connectivity >> USART2
2. Mode >> Mode >> Asynchronous
3. Parameter Settings
4. Baud Rate >> 9600
5. Word Length >> 8 Bit
6. Parity >> None
7. Stop Bits >> 1
8. NVIC Settings >> USART2 global interrupt >> Enabled

Following the USART configuration, the project code is generated through the following:

- Project >> Generate Code

Now that the majority of the code has been generated by the STM32CubeIDE, the desired USART code is entered in the main function between the lines "USER CODE BEGIN WHILE" and "USER CODE END 3". In order to send the string "Hello World!" through USART, the following code is entered:

```
/* USER CODE BEGIN WHILE */
   uint8_t data[] = "Hello World!";
   uint8_t rxBuf[12];

   while (1)
   {
         /* USER CODE END WHILE */

         /* USER CODE BEGIN 3 */
               HAL_UART_Receive_IT(&huart2, rxBuf, 12);
               HAL_UART_Transmit(&huart2, data,12,10000);
               HAL_Delay(1000);
   }
   /* USER CODE END 3 */
}
```

Once the code has been entered, the device is debugged for proper operation. Debugging configuration is completed by the following steps:

1. Run >> Debug Configurations >> Debugger
2. Debug probe >> ST – LINK

Now that debugging is configured, the USART loopback is verified and tested by connecting the following pins on the STM32F103C8:

- A2 ←→ A3

The debugging session is begun by selecting the following settings:

1. Run >> Debug
2. Debug Perspective >> Switch

The code is stepped through and the variable values are examined in the Variables window on the right side of the IDE. The Variables window is accessed through the following:

- Window >> Show View >> Variables

While stepping through the code, the "rxBuf" variable is viewed for the "Hello World!" string under "Details". Proper operation will be determined given that the string under "Details" changes from before transmission to after transmission.

For transmission of a unique string (first name), the majority of the code used for transmission of "Hello World!" is used. The only changes needed are the following:

- `uint8_t data[] = "Hello World!";` → `uint8_t data[] = "Cameron";`
- `uint8_t rxBuf[12];` → `uint8_t rxBuf[7];`
- `HAL_UART_Receive_IT(&huart2, rxBuf, 12);` → `HAL_UART_Receive_IT(&huart2, rxBuf, 7);`
- `HAL_UART_Transmit(&huart2, data, 12, 10000);` → `HAL_UART_Transmit(&huart2, data, 7, 10000);`

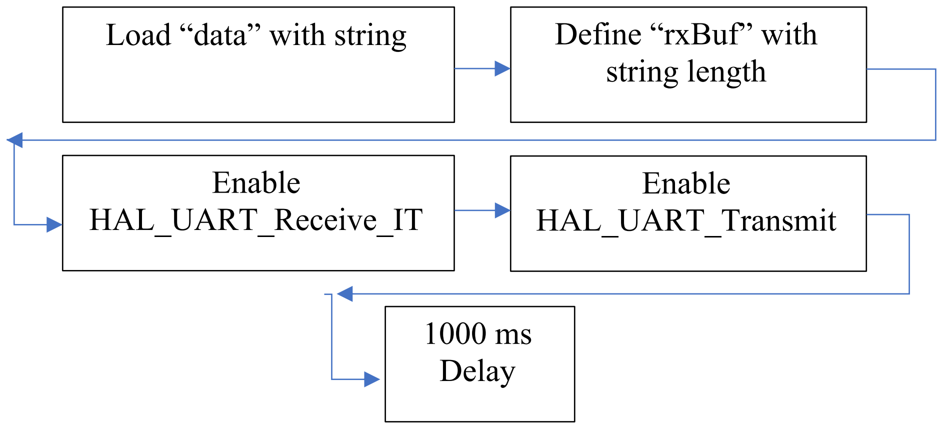
Upon changing the above code, the debugging session is begun and the "rxBuf" variable is observed for proper transmission of the string as the code is stepped through.

Results

The above procedure was followed exactly as it is listed in order to configure the STM32F103C8 for USART transmission. Upon copying the provided code from the lab handout for transmission of "Hello World!", the code was compiled and uploaded to the STM32F103C8 for debugging. Pins A2 and A3 on the STM32F103C8 were connected and the program was stepped through while observing the "rxBuf" variable in the STM32CubeIDE. Initially, while stepping through the program the string was not being transmitted into rxBuf. After reviewing the code and configuration settings, it could not be determined as to where the issue was. Further research online occurred to understand any potential issues. It was determined that the data was not being transmitted due to the USART2 global interrupt not being enabled. Upon enabling the USART2 global interrupt, the program was once again uploaded to the STM32F103C8 for debugging. The program was stepped through and while observing the "rxBuf" variable, it was determined that the program worked as expected and without error as the "Hello World!" string was transmitted through the loopback connection.

For the unique string of "Cameron" being transmitted through the loopback connection, the code was changed accordingly and uploaded to the STM32F103C8. As the program was stepped through, it was determined that the program worked as expected and without error as "Cameron" was successfully transmitted to the "rxBuf" variable through the loopback connection.

Answers to Posted Questions

1. Explain what the code does in the main function.
   - The variable "data" is loaded with the desired string. The variable "rxBuf" is defined as the length of the string. Within the forever loop, HAL_UART_Receive_IT is set to receive the number of characters in the "data" string into "rxBuf" while in interrupt mode. HAL_UART_Transmit transmits the number of characters in the "data" string with a 10,000 ms timeout. Finally, HAL_Delay(1000) provides a 1,000 ms delay before continually repeating the process.
2. Create a block diagram for the code.
   - 

3. What does HAL stand for? Why do you think abstraction is valuable in programming microcontrollers?

- Hardware Abstraction Library. This allows for the ability to write one program and upload the code to a variety of microcontrollers without having to alter the code for the respective controller. Essentially creates a universal language.

4. What is the difference between Synchronous and Asynchronous mode in USART?
   - Synchronous mode allows for the data transfer to be clocked where the bits synchronize with the clock signal. Synchronous mode does not use start and stop bits and a higher baud rate can be achieved. Asynchronous mode requires start and stop bits and the data transferred is synchronized with the start and stop bits.
5. Why is a loopback connection useful? What are the shortcomings of testing a connection this way?
   - Loopback connections are useful as testing can be accomplished using only one device. This allows for quicker and easier debugging with programming. The shortcomings of testing with a loopback are that there may be unforeseen issues when testing with other devices that do not appear when testing on the same device. While these issues cannot always be known until testing with the desired devices occurs, loopbacks provide at the very least a method to verify that data is being transferred and received properly on the test device.

Conclusion

Lab 3.3 introduced the STM32CubeIDE and the "Hardware Abstraction Library" in order to generate a program that takes advantage of one of the devices USART channels. The USART on the STM32F103C8 was configured to transmit the string "Hello World!" from the transmit pin to the receive pin on the same device through a loopback connection. The loopback connection provides the ability to test and debug a program without the use of external hardware and proper operation can be determined through the STM32CubeIDE debugging tools. Following the configuration and implementation of code, the "Hello World!" loopback program was uploaded to the STM32F103C8 and tested for accuracy. It was determined that the "Hello World!" program operated as expected and without error as the string was transmitted. Additionally, a second program was to be implemented that transmitted the string "Cameron" using the same process. Upon changing the string and the size of the string, the other portions of code remained the same. The second program was uploaded to the STM32F103C8 and testing began through the debugging tools in STM32CubeIDE. It was determined that the second program worked as expected as the string “Cameron” was transmitted successfully to the receive pin. Following the completion of Lab 3.4, a greater understanding of the STM32CubeIDE software was gained and the increased ability to control the STM32F103C8 microcontroller. With this knowledge, proper configuration of the STM32F103C8 for control of the USART channels was learned. The information gained within this lab will prove to be valuable as the STM32CubeIDE and use of the USART will be used in future labs in ECEN 3320 as well as future courses.

Appendix

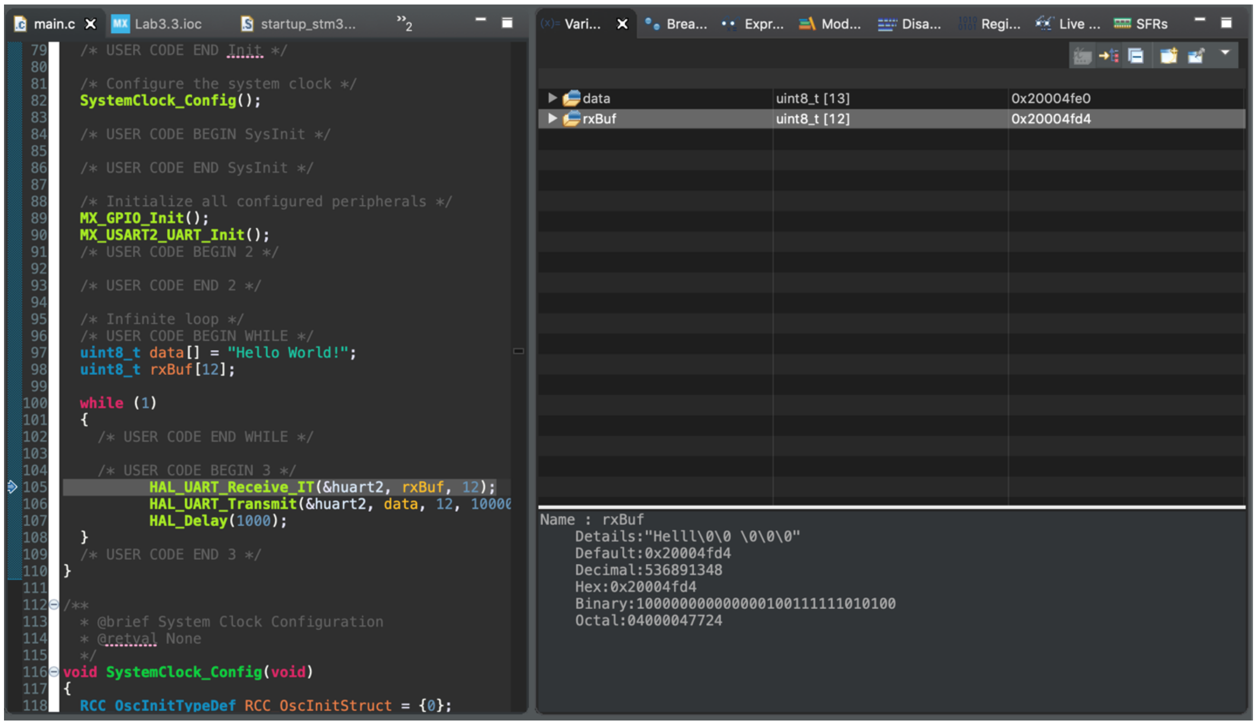
Figure I: UART Loopback "Hello World!"

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

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

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */
```

```
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* 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();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  uint8_t data[] = "Hello World!";
  uint8_t rxBuf[12];

  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
              HAL_UART_Receive_IT(&huart2, rxBuf, 12);
              HAL_UART_Transmit(&huart2, data, 12, 10000);
              HAL_Delay(1000);
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{

  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 9600;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */

  /* USER CODE END USART2_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();

}
```

```
/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */

  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

Figure II: Value Before Transmission

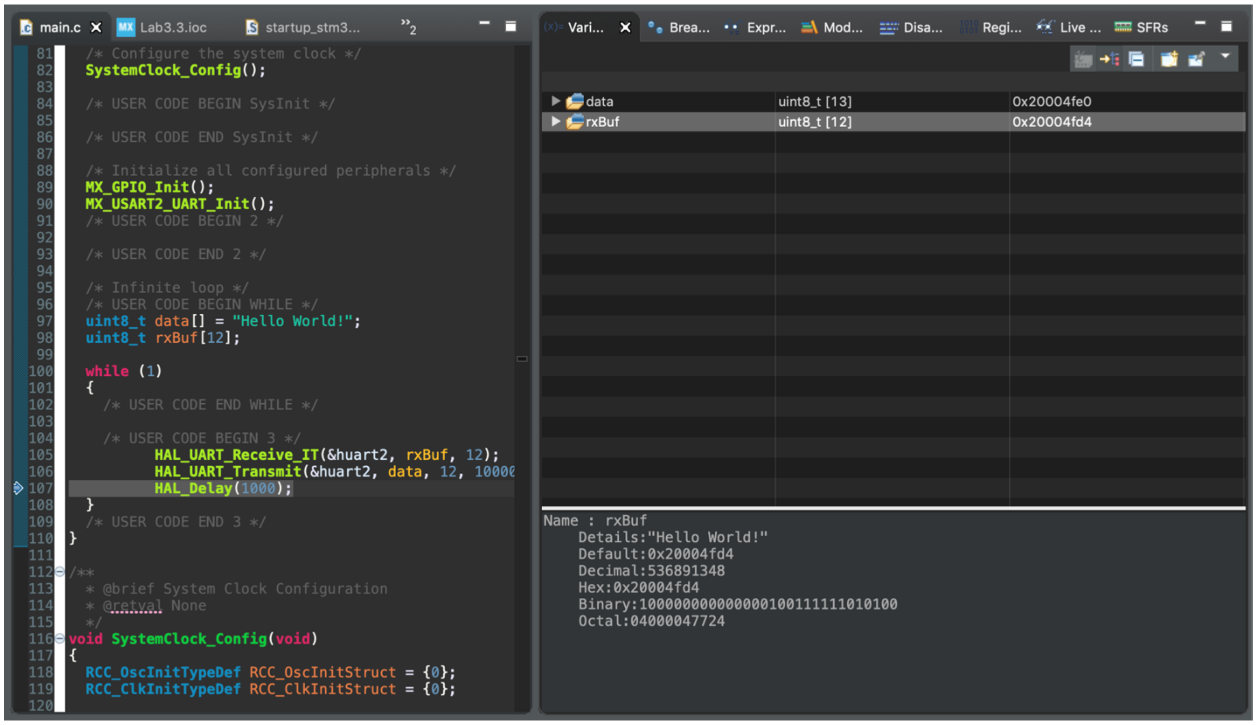


```
  /* USER CODE END Init */

  /* 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();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  uint8_t data[] = "Hello World!";
  uint8_t rxBuf[12];

  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
          HAL_UART_Receive_IT(&huart2, rxBuf, 12);
          HAL_UART_Transmit(&huart2, data, 12, 1000
          HAL_Delay(1000);
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
```

```
data     uint8_t [13]     0x20004fe0
rxBuf    uint8_t [12]     0x20004fd4
```

```
Name : rxBuf
    Details:"Helll\0\0 \0\0\0"
    Default:0x20004fd4
    Decimal:536891348
    Hex:0x20004fd4
    Binary:100000000000000100111111010100
    Octal:04000047724
```

Figure III: Value After Transmission



```
  /* 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();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  uint8_t data[] = "Hello World!";
  uint8_t rxBuf[12];

  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
          HAL_UART_Receive_IT(&huart2, rxBuf, 12);
          HAL_UART_Transmit(&huart2, data, 12, 1000
          HAL_Delay(1000);
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
```

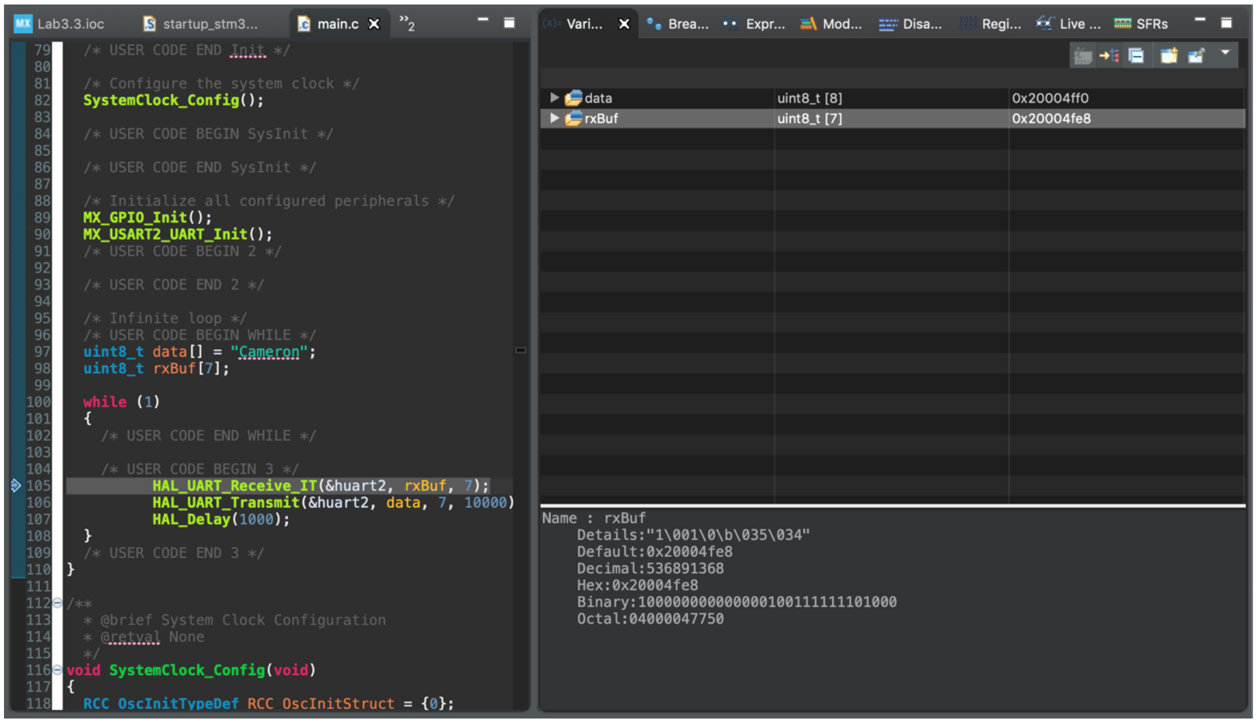
```
data     uint8_t [13]     0x20004fe0
rxBuf    uint8_t [12]     0x20004fd4
```

```
Name : rxBuf
    Details:"Hello World!"
    Default:0x20004fd4
    Decimal:536891348
    Hex:0x20004fd4
    Binary:100000000000000100111111010100
    Octal:04000047724
```

Figure IV: UART Loopback "Cameron"

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

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

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* 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();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  uint8_t data[] = "Cameron";
  uint8_t rxBuf[7];

  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
              HAL_UART_Receive_IT(&huart2, rxBuf, 7);
              HAL_UART_Transmit(&huart2, data, 7, 10000);
              HAL_Delay(1000);
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{

  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 9600;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */

  /* USER CODE END USART2_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();

}

/* USER CODE BEGIN 4 */
/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */

  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

Figure V: Value Before Transmission

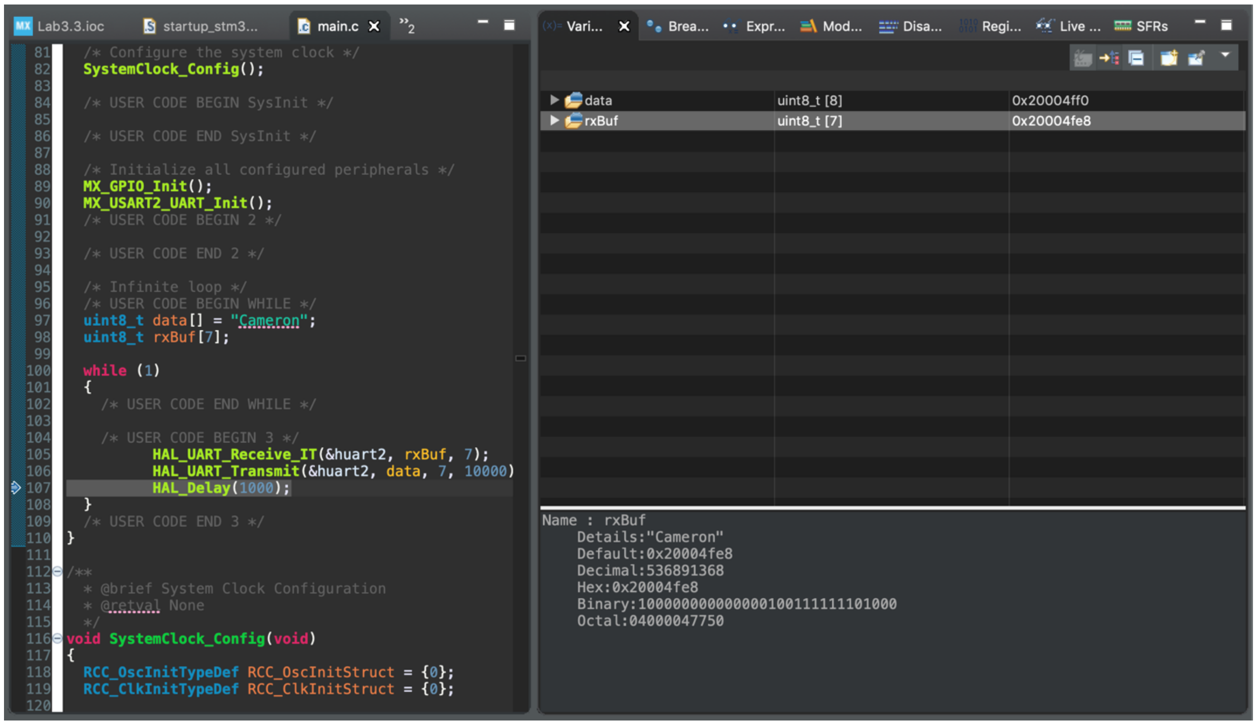


```
  /* USER CODE END Init */

  /* 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();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  uint8_t data[] = "Cameron";
  uint8_t rxBuf[7];

  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
        HAL_UART_Receive_IT(&huart2, rxBuf, 7);
        HAL_UART_Transmit(&huart2, data, 7, 10000)
        HAL_Delay(1000);
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
```

data	uint8_t [8]	0x20004ff0
rxBuf	uint8_t [7]	0x20004fe8

```
Name : rxBuf
    Details:"1\001\0\b\035\034"
    Default:0x20004fe8
    Decimal:536891368
    Hex:0x20004fe8
    Binary:100000000000000100111111101000
    Octal:04000047750
```

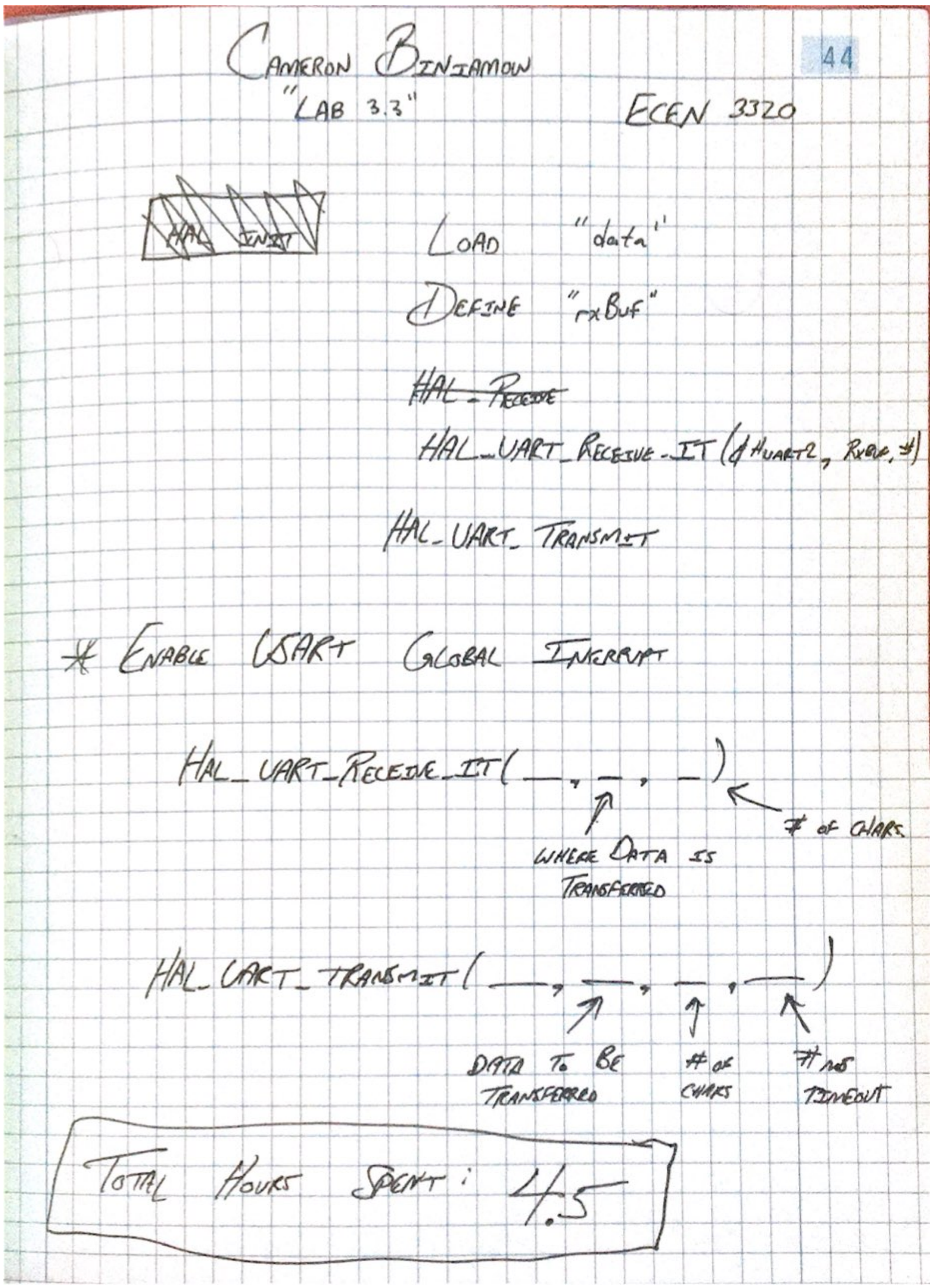
Figure VI: Value After Transmission



```
  /* 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();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  uint8_t data[] = "Cameron";
  uint8_t rxBuf[7];

  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
        HAL_UART_Receive_IT(&huart2, rxBuf, 7);
        HAL_UART_Transmit(&huart2, data, 7, 10000)
        HAL_Delay(1000);
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
```

data	uint8_t [8]	0x20004ff0
rxBuf	uint8_t [7]	0x20004fe8

```
Name : rxBuf
    Details:"Cameron"
    Default:0x20004fe8
    Decimal:536891368
    Hex:0x20004fe8
    Binary:100000000000000100111111101000
    Octal:04000047750
```



Cameron Biniamow 44

"Lab 3.3" ECEN 3320

~~HAL_INIT~~

Load "data"

Define "rxBuf"

~~HAL_Receive~~

HAL_UART_Receive_IT(&Huart2, RxBuf, #)

HAL_UART_Transmit

* Enable USART Global Interrupt

HAL_UART_Receive_IT(__ , __ , __)

Where data is transferred

# of chars

HAL_UART_Transmit(__ , __ , __ , __)

Data to be transferred

# of chars

ms timeout

$\boxed{\text{Total Hours Spent: 4.5}}$