

## Experiment 5

**Title:** Transmitting and Receiving Messages using UART communication

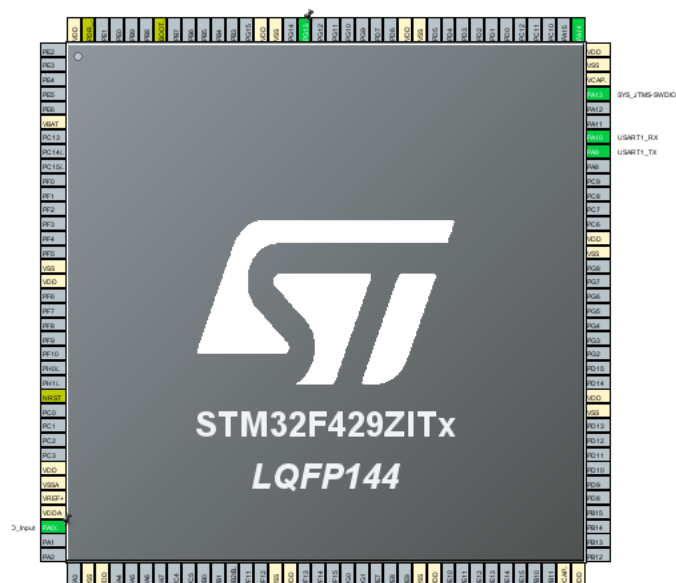
**Aim:** Transmitting and Receiving Messages using UART communication (If switch is pressed display message LED is on otherwise display message LED is off)

**Tool used:** Tool used for this assignment is STM32CubeIDE.

### **Procedure:**

1. Create a new STM32 project with a suitable project name.
2. IOC UI will open in that configure desired pins as input/output.
3. In system core select SYS.
4. In Mode select debug as serial wire.
5. In connectivity select USART1(mode asynchronous) and in parameter settings make baud rate 9600 bits/sec.
6. Select PA0 as input(switch) and PG13 as LED.
7. Press Ctrl+S to generate the code.
8. In the main.c file add the desired code.
9. Go to Project-> Build Project
10. Connect the discovery Board and go to Run-> Run.

### **CubeMX pin diagram:**



## Code:

```
/* USER CODE BEGIN Header */
/**
```

```
*****
***
```

```
 * @file      : main.c
 * @brief     : Main program body
```

```
*****
***
```

```
 * @attention
 *
 * Copyright (c) 2023 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
```

```
*****
***
```

```
 */
/* 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 huart1;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code -----*/
/* USER CODE BEGIN 0 */
unsigned char MSG1[]="STM32 Sending Data \r\n";
unsigned char MSG2[]="Received the string \r\n";
unsigned char MSG3[]="\r\n";
unsigned char MSG4[]="Switch is pressed \r\n";
unsigned char MSG5[]="LED is On \r\n";
unsigned char MSG6[]="LED is Off \r\n";
unsigned char RX_BUFFER[30];
unsigned char RX_Success;
/* 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_USART1_UART_Init();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
//HAL_UART_Transmit(&huart1,MSG1,21,100);
while (1)
{
    /* USER CODE END WHILE */
    if(HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_0)==1)
    {
        HAL_UART_Transmit(&huart1,MSG4,20,100);
    }
    RX_Success=HAL_UART_Receive(&huart1,RX_BUFFER,1,500);

    /*if(RX_Success==HAL_OK)
    {
        HAL_UART_Transmit(&huart1,MSG2,22,100);
        HAL_UART_Transmit(&huart1,RX_BUFFER,10,100);
        HAL_UART_Transmit(&huart1,MSG3,2,100);
    }*/

    if((RX_Success==HAL_OK)&&(RX_BUFFER[0]=='1'))
    {
        HAL_GPIO_WritePin(GPIOG,GPIO_PIN_13,1);
        HAL_UART_Transmit(&huart1,MSG5,13,100);
    }
    if((RX_Success==HAL_OK)&&(RX_BUFFER[0]=='0'))
    {
        HAL_GPIO_WritePin(GPIOG,GPIO_PIN_13,0);
        HAL_UART_Transmit(&huart1,MSG6,13,100);
    }

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)

```

```

{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Configure the main internal regulator output voltage
    */
    __HAL_RCC_PWR_CLK_ENABLE();

    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE
3);

    /** Initializes the RCC Oscillators according to the specified parameters
    * in the RCC_OscInitTypeDef structure.
    */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
    RCC_OscInitStruct.HSISState = 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 USART1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_USART1_UART_Init(void)
{

    /* USER CODE BEGIN USART1_Init 0 */

    /* USER CODE END USART1_Init 0 */

```

```

/* USER CODE BEGIN USART1_Init 1 */

/* USER CODE END USART1_Init 1 */
huart1.Instance = USART1;
huart1.Init.BaudRate = 9600;
huart1.Init.WordLength = UART_WORDLENGTH_8B;
huart1.Init.StopBits = UART_STOPBITS_1;
huart1.Init.Parity = UART_PARITY_NONE;
huart1.Init.Mode = UART_MODE_TX_RX;
huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart1.Init.OverSampling = UART_OVERSAMPLING_16;
if (HAL_UART_Init(&huart1) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN USART1_Init 2 */

/* USER CODE END USART1_Init 2 */

}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

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

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_RESET);

    /*Configure GPIO pin : PA0 */
    GPIO_InitStruct.Pin = GPIO_PIN_0;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    /*Configure GPIO pin : PG13 */
    GPIO_InitStruct.Pin = GPIO_PIN_13;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;

```

```

GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* 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 */
    __disable_irq();
    while (1)
    {
    }
    /* 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,
       ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
    /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

```

## Output:

```
COM7

11:25:19.307 -> Switch is pressed
11:25:29.543 -> LED is On
11:25:33.557 -> LED is Off

COM7

11:26:01.330 -> Switch is pressed
11:26:10.864 -> LED is On
```



## Functions used:

HAL\_GPIO\_ReadPin(GPIOx,GPIO Pin Number);

HAL\_UART\_Transmit (&handle\_variable\_name,MSG,strlen(MSG),Timeout);

HAL\_UART\_Receive (&handle\_variable\_name,RX\_BUFFER,strlen(RX),Timeout);

HAL\_GPIO\_WritePin(GPIOx,GPIO Pin Number,1/0);

**Result:** Basic STM32Cube project for transmitting and receiving messages with the help of UART communication was built using STM32CubeIDE.