

การทดลองที่ 3 การใช้งาน UART

วัตถุประสงค์

- 1) เข้าใจการทำงานของ UART
- 2) สามารถเขียนโปรแกรมเพื่อรับส่งข้อความผ่านพอร์ต UART

1. UART / USART

USART (Universal Synchronous/Asynchronous Receiver/Transmitter) เป็นพอร์ตสื่อสารซึ่งสามารถใช้งานได้ทั้งแบบ Asynchronous (UART) และ Synchronous (USART) ไมโครคอนโทรลเลอร์หมายเลข STM32F767 มีพอร์ต UART จำนวน 8 พอร์ต ได้แก่ UART1 – UART8 แต่บอร์ด Nucleo-767ZI ไม่มีไอซี MAX232 จึงไม่สามารถใช้งาน UART ผ่าน RS232 ได้ ทางผู้ผลิตได้เชื่อมต่อ UART3 เข้าไอซี ST-Link บนบอร์ดและเมื่อร่วมกับโปรแกรมไดรเวอร์ ST-Link ที่เครื่องคอมพิวเตอร์จึงสามารถสร้าง Virtual Communication Port เพื่อสื่อสารข้อมูลระหว่างไมโครคอนโทรลเลอร์กับเครื่องคอมพิวเตอร์ผ่าน UART ได้

แต่ละขาของไอซินั้นสามารถกำหนดหน้าที่การทำงานได้สูงสุด 3 รูปแบบ คือ

- 1) Main Function หรือ After Reset
- 2) Alternate Function
- 3) Additional Function

บอร์ด Nucleo-767ZI ได้เชื่อมต่อขา PD8 และ PD9 ซึ่งสามารถทำหน้าที่ UART3_Tx และ UART3_Rx ตามลำดับเข้ากับไอซี ST-Link เพื่อทำเป็นพอร์ต UART เสมือน เพื่อเชื่อมต่อกับเครื่องคอมพิวเตอร์ เมื่อไมโครคอนโทรลเลอร์เริ่มต้นการทำงานขา 77 และ 78 จะทำหน้าที่เป็น GPIO ได้แก่ PD8 และ PD9 ตามลำดับ หากต้องการเปลี่ยนให้ขา 77 และ 78 ทำหน้าที่เป็น UART3 จะต้องเขียนโปรแกรมเพื่อกำหนดให้ทั้งสองขาดังกล่าวทำหน้าที่ Alternate Function หากต้องการใช้ UART ของไมโครคอนโทรลเลอร์เป็นพอร์ตสื่อสารแบบ UART ด้วยมาตรฐาน RS232 ต้องต่อไอซีแปลงระดับแรงดันไฟฟ้าเสียก่อน ตัวอย่างขาของไมโครคอนโทรลเลอร์ที่สามารถทำหน้าที่ UART ได้แสดงดังตารางที่ 1.1

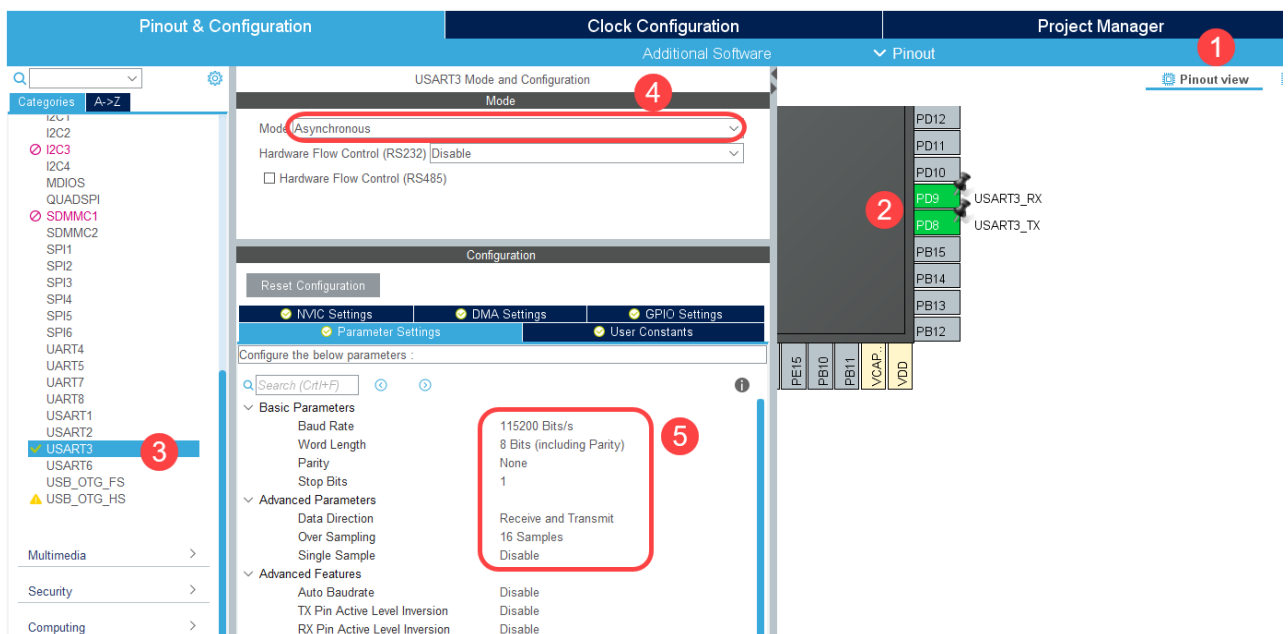
ตารางที่ 1.1 แสดงการทำงานของขาไมโครคอนโทรลเลอร์ที่เกี่ยวกับพอร์ต UART7 UART4 และ UART3

Pin NO.	Main Function (After Reset)	Alternate Functions	Additional Function
18	PF6	TIM10_CH1, SPI5_NSS, SAI1_SD_B, UART7_RX , QUADSPI_BK1_IO3, EVENTOUT	ADC3_IN4
19	PF7	TIM11_CH1, SPI5_SCK, SAI1_MCLK_B, UART7_TX , QUADSPI_BK1_IO2, EVENTOUT	ADC3_IN5

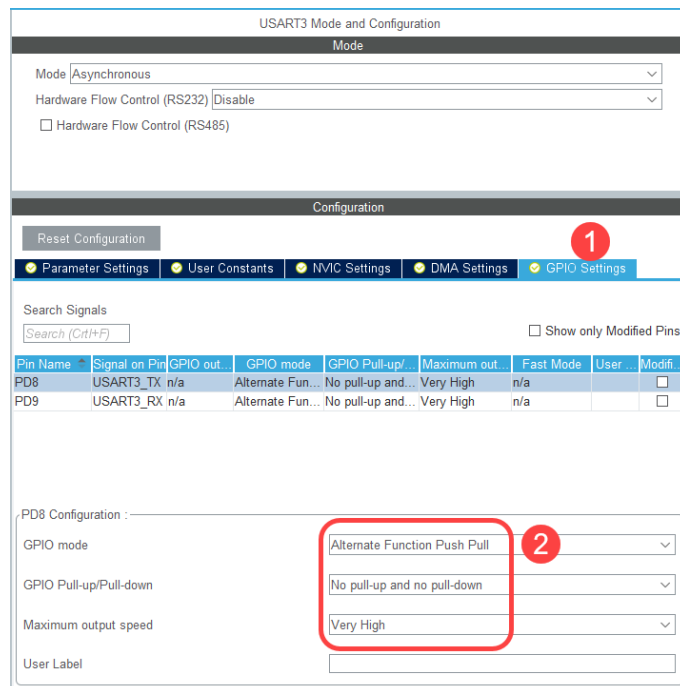
Pin NO.	Main Function (After Reset)	Alternate Functions	Additional Function
34	PA0	TIM2_CH1/TIM2_ETR, TIM5_CH1, TIM8_ETR, USART2_CTS, UART4_TX , SAI2_SD_B, ETH_MII_CRS, EVENTOUT	ADC1_IN0, ADC2_IN0, ADC3_IN0, WKUP1
35	PA1	TIM2_CH2, TIM5_CH2, USART2_RTS, UART4_RX , QUADSPI_BK1_IO3, SAI2_MCLK_B, ETH_MII_RX_CLK/ETH_RMII_ REF_CLK, LCD_R2, EVENTOUT	ADC1_IN1, ADC2_IN1, ADC3_IN1
77	PD8	DFSDM1_CKIN3, USART3_TX , SPDIF_RX1, FMC_D13, EVENTOUT	-
78	PD9	DFSDM1_DATIN3, USART3_RX , FMC_D14, EVENTOUT	-

2. การตั้งค่า UART

การจะใช้งานพอร์ต UART บนบอร์ดการทดลองผ่าน Virtual Com Port (VCP) ต้องตั้งค่าขาของไมโครคอนโทรลเลอร์ที่เชื่อมต่ออยู่กับไอซี ST-Link ด้วยโปรแกรม STM32CubeMX ให้ถูกต้อง ดังรูปที่ 2.1 และ รูปที่ 2.2



รูปที่ 2.1 แสดงการตั้งค่าพอร์ต UART3 ที่แท้บ PINOUT



รูปที่ 2.2 แสดงการตั้งค่าพอร์ต UART3

3. อธิบายการทำงาน

การเริ่มต้นใช้งาน UART ต้องกำหนดค่าต่างๆ ที่เกี่ยวข้องกับ UART เสียก่อน เช่น

- Baud rate (ความเร็วในการรับส่งข้อมูล) เช่น 115200, 57600 หรือ 38400 Bits/sec
- จำนวนบิตใน 1 เฟรมว่าจะเป็น 7, 8 หรือ 9 บิต (รวม parity bit แล้ว)
- จำนวน Stop bit
- Parity ที่จะใช้ตรวจสอบความถูกต้องของการรับส่งข้อมูล ได้แก่ even/odd/no parity
- การเลือกจะใช้โปรโตคอลเกี่ยวกับ hardware flow control หรือไม่
- โหมดการทำงานว่าจะให้รับหรือส่งข้อมูล หรือทั้งรับและส่ง เป็นต้น

จากนั้นทำการตั้งค่า Alternate Function ให้ขาไมโครคอนโทรลเลอร์จากที่ทำหน้าที่เป็น GPIO PD8 และ PD9 ให้ทำหน้าที่เป็น USART3_TX และ USART3_RX ตามลำดับ

การตั้งค่าการทำงาน UART จากโปรแกรม STM32CubeMX จะถูกกำหนดไว้ที่ไฟล์ 3 ไฟล์ ได้แก่

- `usart.c`
- `gpio.c`
- `main.c`

ไฟล์ `usart.c`

เป็นไฟล์ที่รวมการตั้งค่า GPIO ให้ทำหน้าที่ Alternate function และการตั้งค่า UART

Global variables

- จะทำการประกาศตัวแปร `huart3` เพื่อที่จะใช้เป็นตัวแทนของโมดูล UART3 ดังรูปที่ 3.1
`UART_HandleTypeDef huart3;`
- และเพื่อให้สามารถใช้ตัวแปร `huart3` นี้ในไฟล์อื่นๆ เช่น `main.c` ได้ จึงได้ทำการประกาศตัวแปรแบบ `extern` ไว้ในไฟล์ `usart.h`
`extern UART_HandleTypeDef huart3;`

ฟังก์ชัน `MX_USART2_UART_Init()`

- เป็นฟังก์ชันที่โปรแกรม STM32CubeMX สร้างขึ้นมา เพื่อตั้งค่า UART3 บนไมโครคอนโทรลเลอร์ให้สอดคล้องกับค่าที่กำหนดไว้ในโปรแกรม ดังรูปที่ 3.1
- เริ่มต้นด้วยการกำหนดให้ตัวแปร `huart3` เป็นตัวแทนของ UART3
`huart3.Instance = USART3;`
- แล้วทำการตั้งค่าการทำงานของ USART ดังนี้
 - Baud rate = 115,200 bits/second
 - ใช้ 8 บิตใน 1 เฟรม
 - ใช้ 1 stop bit
 - ไม่ใช้ Parity bit
 - กำหนดให้ทำงานทั้งรับและส่งข้อมูล
 - ไม่ใช้ Hardware Flow Control
 - กำหนดให้ทำการ Oversampling 16 เท่า

```
huart3.Init.BaudRate      = 115200;  
huart3.Init.WordLength    = UART_WORDLENGTH_8B;  
huart3.Init.StopBits      = UART_STOPBITS_1;  
huart3.Init.Parity        = UART_PARITY_NONE;  
huart3.Init.Mode          = UART_MODE_TX_RX;  
huart3.Init.HwFlowCtl     = UART_HWCONTROL_NONE;  
huart3.Init.OverSampling  = UART_OVERSAMPLING_16;  
HAL_UART_Init(&huart3);
```

```

UART_HandleTypeDef huart3;

/* USART3 init function */

void MX_USART3_UART_Init(void)
{
    huart3.Instance = USART3;
    huart3.Init.BaudRate = 115200;
    huart3.Init.WordLength = UART_WORDLENGTH_8B;
    huart3.Init.StopBits = UART_STOPBITS_1;
    huart3.Init.Parity = UART_PARITY_NONE;
    huart3.Init.Mode = UART_MODE_TX_RX;
    huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart3.Init.OverSampling = UART_OVERSAMPLING_16;
    huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
    huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
    if (HAL_UART_Init(&huart3) != HAL_OK)
    {
        Error_Handler();
    }
}

```

รูปที่ 3.1 ฟังก์ชันตั้งค่า UART3

ฟังก์ชัน HAL_UART_MspInit()

- เป็นฟังก์ชันที่ใช้ในการตั้งค่าขา GPIO ที่จะนำมาใช้เป็นขา TX และ RX ของ UART ดังรูปที่ 3.2
- เริ่มต้นการตั้งค่า UART3 ด้วยการจ่ายสัญญาณนาฬิกาให้กับโมดูล UART และ GPIOD


```

HAL_RCC_USART3_CLK_ENABLE();
HAL_RCC_GPIOD_CLK_ENABLE();

```
- กำหนดให้ PD8 และ PD9 ทำหน้าที่ UART โดยจะตั้งค่าให้เป็น alternate function แบบพุชพูล โดยไม่ใช้ตัวต้านทานภายใน ที่ความเร็วแบบ Very High Speed

```

GPIO_InitStruct.Pin      = GPIO_PIN_8|GPIO_PIN_9;
GPIO_InitStruct.Mode     = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull     = GPIO_NOPULL;
GPIO_InitStruct.Speed    = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF7_USART3;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

```

```

void HAL_UART_MspInit(UART_HandleTypeDef* uartHandle)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    if(uartHandle->Instance==USART3)
    {
        /* USER CODE BEGIN USART3_MspInit 0 */

        /* USER CODE END USART3_MspInit 0 */
        /* USART3 clock enable */
        __HAL_RCC_USART3_CLK_ENABLE();

        __HAL_RCC_GPIOD_CLK_ENABLE();
        /**USART3 GPIO Configuration
        PD8      -----> USART3_TX
        PD9      -----> USART3_RX
        */
        GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9;
        GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
        GPIO_InitStruct.Pull = GPIO_NOPULL;
        GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
        GPIO_InitStruct.Alternate = GPIO_AF7_USART3;
        HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

        /* USER CODE BEGIN USART3_MspInit 1 */

        /* USER CODE END USART3_MspInit 1 */
    }
}

```

รูปที่ 3.2 ฟังก์ชันตั้งค่า UART3

ไฟล์ gpio.c

ฟังก์ชัน MX_GPIO_Init()

- เป็นฟังก์ชันที่โปรแกรม STM32CubeMX สร้างขึ้นมา เพื่อจ่ายสัญญาณนาฬิกาให้กับ GPIO พอร์ต D ด้วยคำสั่ง

```
__HAL_GPIO_CLK_ENABLE();
```

ไฟล์ main.c

ฟังก์ชัน main()

- เริ่มต้นการทำงานด้วยฟังก์ชันเพื่อตั้งค่าโมดูลต่างๆ ได้แก่
 - HAL_Init()
 - SystemClock_Config()
 - MX_GPIO_Init()
 - MX_USART3_UART_Init()

4. การส่งและรับข้อมูลผ่าน UART

ก่อนการส่งหรือการรับข้อมูลผ่าน UART จำเป็นต้องตรวจสอบสถานะการทำงานของโมดูล UART เสียก่อนว่ามีสถานะที่พร้อมรับข้อมูลเพื่อส่งออก (send) หรือว่าพร้อมที่จะให้อ่านข้อมูลที่รับเข้ามาหรือไม่ (receive) ด้วยการตรวจสอบบางบิตในรีจิสเตอร์ของโมดูล (flag) เช่น ตรวจสอบแฟล็ก Transmission Control (TC) เพื่อตรวจสอบว่าการส่งข้อมูลก่อนหน้าดำเนินการเสร็จสิ้นหรือยัง ถ้ายังไม่เสร็จ UART ก็ไม่สามารถส่งข้อมูลใหม่ได้ หรือตรวจสอบแฟล็ก Read Data

Register Not Empty (RXNE) เพื่อดูว่าข้อมูลที่รับเข้ามาที่ละบิตได้ถูกเลื่อนบิต (shift) เข้ามาในบัฟเฟอร์ (buffer) จนครบแล้วหรือไม่ โดยมีฟังก์ชันที่เกี่ยวข้องในการรับส่งข้อมูลดังนี้

มาโคร `__HAL_UART_GET_FLAG (__HANDLE__ , __FLAG__)`

- ใช้เพื่ออ่านค่าแฟล็กต่างๆ ของ UART
- `__HANDLE__` : ระบุ UART ที่ต้องการ เช่น &huart3 เป็นต้น
- `__FLAG__` : ระบุแฟล็กที่ต้องการทราบค่า เช่น `UART_FLAG_RXNE` และ `UART_FLAG_TC` เป็นต้น

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

- ใช้เพื่อส่งข้อมูลที่ต้องการผ่านทาง UART
- `huart` : ระบุ UART ที่จะใช้ส่งข้อมูล เช่น &huart3 เป็นต้น
- `pData` : คือ Pointer ที่ชี้ไปยังตำแหน่งเริ่มต้นของข้อมูลที่จะส่ง
- `Size` : ความยาวของข้อมูลที่จะส่งในหน่วย Byte
- `Timeout` : ระยะเวลาที่ฟังก์ชันสามารถใช้เพื่อส่งข้อมูลมีหน่วยเป็น millisecond หากไม่สามารถส่งข้อมูลเสร็จภายในระยะเวลาที่กำหนด ฟังก์ชันจะส่งค่ากลับเป็น `HAL_TIMEOUT`
- ฟังก์ชันจะส่งค่ากลับเป็นสถานะการทำงาน เช่น `HAL_OK` เมื่อส่งข้อมูลสำเร็จ หรือ `HAL_BUSY` ถ้าโมดูล UART ไม่พร้อมทำงาน

```
char str[] = "Hello, World!!\r\n";

while(__HAL_UART_GET_FLAG(&huart3, UART_FLAG_TC) == RESET) {}
HAL_UART_Transmit(&huart3, (uint8_t*) str, strlen(str), 1000);

HAL_Delay(500);
```

รูปที่ 4.1 ตัวอย่างการส่งข้อมูลผ่าน UART

รูปที่ 4.1 แสดงตัวอย่างการส่งข้อมูลของตัวแปรข้อความ `str` ผ่าน UART ตัวแปร `str` ถูกปิดท้ายข้อความ "Hello, World!!\r\n" ด้วยตัวอักษรพิเศษ 2 ตัว ได้แก่ ตัวอักษร Carriage Return หรือ '\r' ใช้สำหรับเลื่อนเคอร์เซอร์ให้กลับสู่ตำแหน่งแรกของบรรทัดปัจจุบัน และตัวอักษร Line Feed หรือ '\n' ใช้สำหรับขึ้นบรรทัดใหม่โดยเคอร์เซอร์จะอยู่ตำแหน่งเดียวกันกับบรรทัดบน ก่อนการเรียกใช้ฟังก์ชันเพื่อส่งข้อมูลต้องรอให้ UART พร้อมรับข้อมูลใหม่ที่จะส่งออกไปด้วยการวนลูปรอจนกระทั่งแฟล็ก TC ถูกเซต

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

- ใช้เพื่ออ่านข้อมูลที่รับเข้ามาทาง UART
- `huart` : ระบุ UART ที่จะใช้ส่งข้อมูล เช่น &huart3 เป็นต้น
- `pData` : คือ Pointer ที่ชี้ไปยังตำแหน่งที่อยู่ของตัวแปรที่รับข้อมูล
- `Size` : ระบุความยาวของข้อมูลที่จะรับในหน่วย Byte

- Timeout : ระยะเวลาที่ฟังก์ชันสามารถใช้เพื่ออ่านข้อมูลที่รับเข้ามาจากบัฟเฟอร์มีหน่วยเป็น millisecond หากไม่สามารถทำงานเสร็จภายในระยะเวลาที่กำหนด ฟังก์ชันจะส่งค่ากลับเป็น HAL_TIMEOUT
- โดยตัวฟังก์ชันจะรีเทิร์นค่าออกเป็นสถานะการทำงาน เช่น HAL_OK หรือ HAL_BUSY

```
char ch1;

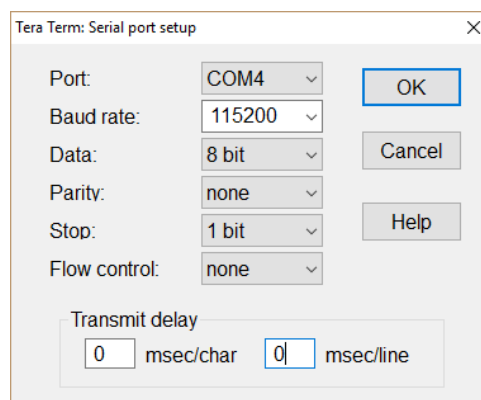
while(__HAL_UART_GET_FLAG(&huart3,UART_FLAG_RXNE) == RESET) {}
HAL_UART_Receive(&huart3, (uint8_t*) &ch1, 1, 1000);
```

รูปที่ 4.2 ตัวอย่างการรับข้อมูลผ่าน UART

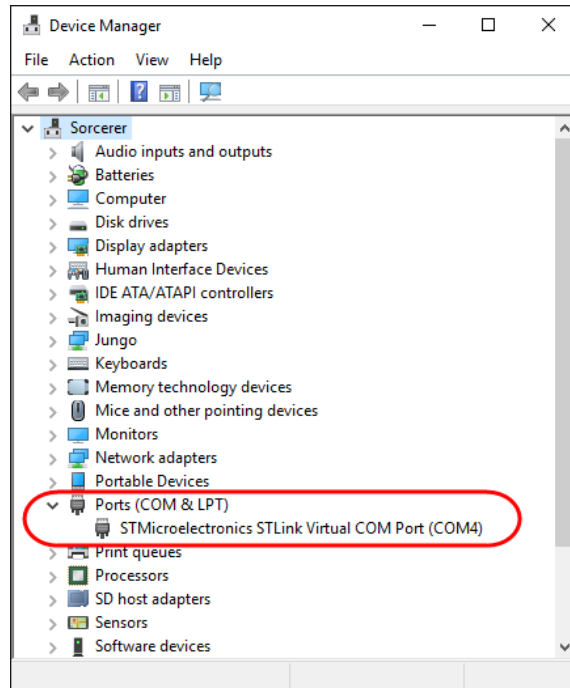
5. โปรแกรม Tera Term

โปรแกรม Tera Term ใช้สำหรับการรับส่งข้อมูลผ่านทางพอร์ตการสื่อสาร เช่น พอร์ตอนุกรม Serial Port (COM Port) ด้วยรหัส ASCII โดยเมื่อเชื่อมต่อเครื่องคอมพิวเตอร์เข้ากับไมโครคอนโทรลเลอร์ผ่านทางพอร์ตอนุกรมหรือพอร์ตอนุกรมเสมือนแล้ว หากผู้ใช้กดปุ่มใดๆ บนคีย์บอร์ดที่เครื่องคอมพิวเตอร์ในโปรแกรม Tera Term โปรแกรมจะส่งรหัส ASCII ของตัวอักษรที่ผู้ใช้กดออกไปทางพอร์ตอนุกรมส่งไปยังไมโครคอนโทรลเลอร์ และถ้าหากไมโครคอนโทรลเลอร์ส่งข้อมูลรหัส ASCII ผ่านทางพอร์ต UART ออกมา โปรแกรมนี้ก็จะรับข้อมูลแล้วแสดงผลตัวอักษรที่มีรหัส ASCII ตรงกันบนหน้าจอของโปรแกรม

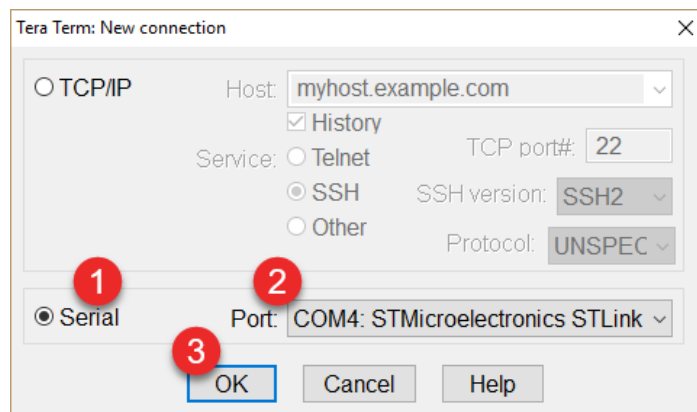
- ดาวน์โหลดและติดตั้งโปรแกรม Tera Term ได้ที่ url <https://ttssh2.osdn.jp/index.html.en>
- เปิดโปรแกรมแล้วตั้งค่าพอร์ตอนุกรมโดยเลือกเมนู File -> Setup -> Serial Port แล้วตั้งค่าให้ตรงกันกับการตั้งค่าของ UART บนไมโครคอนโทรลเลอร์ ดังรูปที่ 5.1 โดยใช้หมายเลข Virtual COM port ตามที่ปรากฏใน Device Manager รูปที่ 5.2
- เชื่อมต่อเครื่องคอมพิวเตอร์กับไมโครคอนโทรลเลอร์โดยเลือกเมนู File -> New connection แล้วเลือก COM port ที่ต้องการ ดังรูปที่ 5.3
- สามารถตัดการเชื่อมต่อ ที่เมนู File -> Disconnect
- เคลียร์หน้าจอที่เมนู Edit -> Clear screen



รูปที่ 5.1 การตั้งค่าโปรแกรม Tera Term



รูปที่ 5.2 แสดงหมายเลข COM Port ใน Device Manager



รูปที่ 5.3 การสร้างการเชื่อมต่อไปยังไมโครคอนโทรลเลอร์

6. การทดลอง

1. จงเขียนโปรแกรมรับส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์และเครื่องคอมพิวเตอร์ผ่าน UART3 โดยใช้โปรแกรม STM32CubeIDE สร้างไฟล์โปรเจกต์ขึ้นมารูปที่ 2.1 และรูปที่ 2.2

จากนั้นให้เขียนโปรแกรมในลูป while ในฟังก์ชัน main() เพื่อส่งข้อความ "Hello World!!" ไปแสดงในโปรแกรม Tera Term ผ่านทางพอร์ต UART โดยใช้ฟังก์ชันส่งข้อมูลดังรูปที่ 4.1 โดยให้ include string.h เพิ่มเติมเข้ามาในไฟล์ main.c เพื่อให้สามารถใช้งานฟังก์ชัน strlen เพื่อหาความยาวของ string ได้ดังนี้

```
/* Includes -----  
#include "main.h"  
#include "usart.h"  
#include "gpio.h"  
  
/* Private includes -----  
/* USER CODE BEGIN Includes */  
#include "string.h"  
/* USER CODE END Includes */
```

รูปที่ 6.1 การสร้างการเชื่อมต่อไปยังไมโครคอนโทรลเลอร์

2. จงเขียนคำสั่งเพื่อส่งข้อมูลออกจาก UART3 โดยใช้ตัวแปร char

```
char ch1 = 'A';
```

```
while(_HAL_UART_GET_FLAG(&huart3, UART_FLAG_TC) == RESET){}
```

```
HAL_UART_Transmit(&huart3, (uint8_t*) &ch1, 1, 1000);
```

คำสั่งเพื่อส่งข้อมูลออกจาก UART3 โดยใช้ตัวแปร char

3. จงเขียนโปรแกรมให้ไมโครคอนโทรลเลอร์แสดงข้อความดังต่อไปนี้ในโปรแกรม Tera Term

Input =>

จากนั้นรอให้ผู้ป้อนตัวอักษร 1 ตัว เมื่อผู้ใช้ป้อนตัวอักษรใดให้ไมโครคอนโทรลเลอร์นำตัวอักษรที่ถูกป้อนนั้นกลับมาแสดงในโปรแกรม Tera Term จากนั้นให้แสดงข้อความเดิม (**Input =>**) ในบรรทัดใหม่ แล้วรอให้ผู้ป้อนตัวอักษรตัวต่อไป ให้โปรแกรมทำงานแบบนี้ซ้ำไปเรื่อยๆ จนกระทั่งผู้ใช้กดปุ่ม 'q' ให้ไมโครคอนโทรลเลอร์แสดงข้อความ "QUIT" แล้วหยุดการทำงานไม่รับและไม่ส่งข้อมูลใดๆ อีก ดังรูปที่ 6.2

```
Input => a  
Input => 1  
Input => D  
Input => q  
QUIT
```

รูปที่ 6.2 แสดงตัวอย่างการทำงานของการทำงานทดลองข้อ 2 ในโปรแกรม Tera Term

4. ต่อ LED จำนวน 2 ดวง เข้ากับขา GPIO ของไมโครคอนโทรลเลอร์ตามต้องการ จากนั้นให้ไมโครคอนโทรลเลอร์แสดงข้อความเมนูผ่านทาง UART3 เพื่อไปแสดงในโปรแกรม Tera Term ดังรูปที่ 6.3

```
Display Blinking LED PRESS (1, 2)
Display Group Members PRESS m
Quit PRESS q
Input =>
```

รูปที่ 6.3 แสดงเมนูในโปรแกรม Tera Term

จากนั้นรอให้ผู้ใช้งานป้อนตัวอักษร 1 ตัว แสดงตัวอักษรที่ผู้ใช้ป้อนในโปรแกรม Tera Term แล้วให้โปรแกรมทำงานตามที่กำหนดในตารางที่ 6.1 หลังจากทำงานเสร็จสิ้นแล้ว ให้ไมโครคอนโทรลเลอร์ **รอรับคำสั่งถัดไป** ด้วยการแสดงเมนูเฉพาะบรรทัด **Input =>**

ตารางที่ 6.1 แสดงการทำงานของโปรแกรมสำหรับการทดลองข้อ 3

ปุ่มที่กด	การทำงานของ LED
1	ให้ LED ดวงที่ 1 ที่เชื่อมต่อกับไมโครคอนโทรลเลอร์กระพริบ 3 ครั้ง โดยหน่วงเวลา 300 ms
2	ให้ LED ดวงที่ 2 ที่เชื่อมต่อกับไมโครคอนโทรลเลอร์กระพริบ 3 ครั้ง โดยหน่วงเวลา 300 ms
m	แสดงชื่อ และรหัสของสมาชิกในกลุ่ม (ชื่อ 1 บรรทัดและรหัส 1 บรรทัดต่อสมาชิก 1 คน) <div>63xxxxxx</div> <div>First1 Last1</div> <div>63xxxxxx</div> <div>First2 Last2</div>
q	แสดงคำว่า Quit แล้วให้ไมโครคอนโทรลเลอร์จบการทำงาน ไม่รับหรือส่งข้อมูลใดๆ อีก
ปุ่มอื่นๆ	แสดงคำว่า Unknown Command แล้วรอรับข้อมูลใหม่

ใบตรวจการทดลองที่ 3

Microcontroller Application and Development 2566

วัน/เดือน/ปี 7/สิงหาคม/2566 กลุ่มที่ 8

1. รหัสนักศึกษา 64010131 ชื่อ-นามสกุล จิรัฐตะวัน ลาดร

2. รหัสนักศึกษา 64010315 ชื่อ-นามสกุล ธนธร แดงอ่อน

64010441 นิภาภัทร คำมัน

ลายเซ็นผู้ตรวจ

การทดลองข้อ 1-3 ผู้ตรวจ _____ วันที่ตรวจ ☐ W ☐ W+1

การทดลองข้อ 4 ผู้ตรวจ _____ วันที่ตรวจ ☐ W ☐ W+1

คำถามท้ายการทดลอง

1. หากตัดคำสั่งต่อไปนี้ออกจากโค้ดของการทดลองข้อ 1 โปรแกรมจะสามารถทำงานได้สมบูรณ์เหมือนเดิมหรือไม่? เพราะเหตุใด?

```
while(__HAL_UART_GET_FLAG(&huart3,UART_FLAG_TC)==RESET){}
```

โปรแกรมจะสามารถทำงานได้สมบูรณ์เหมือนเดิม เพราะคำสั่งที่ตัดออกจากโค้ดการทดลองข้อที่ 1 ทำหน้าที่เช็คสถานะของการส่งข้อมูลแต่ข้อมูลที่ส่งมีทั้งขนาดเล็กและขนาดใหญ่ก็สามารถส่งได้ปกติ

2. หากตัดคำสั่งต่อไปนี้ออกจากโค้ดรูปที่ 4.2 โปรแกรมจะสามารถทำงานได้สมบูรณ์เหมือนเดิมหรือไม่? เพราะเหตุใด?

```
while(__HAL_UART_GET_FLAG(&huart3,UART_FLAG_RXNE)==RESET){}
```

โปรแกรมจะไม่สามารถทำงานได้สมบูรณ์เหมือนเดิม เพราะคำสั่งที่ตัดออกจากโค้ดรูปที่ 4.2 มีหน้าที่เช็คสถานะการรับข้อมูลว่ามีข้อมูลส่งมาใหม่ จึงทำให้เมื่อตัดออกไปนั้น จะรับข้อมูลเป็น null รัวๆ