Busan Software Meister High School

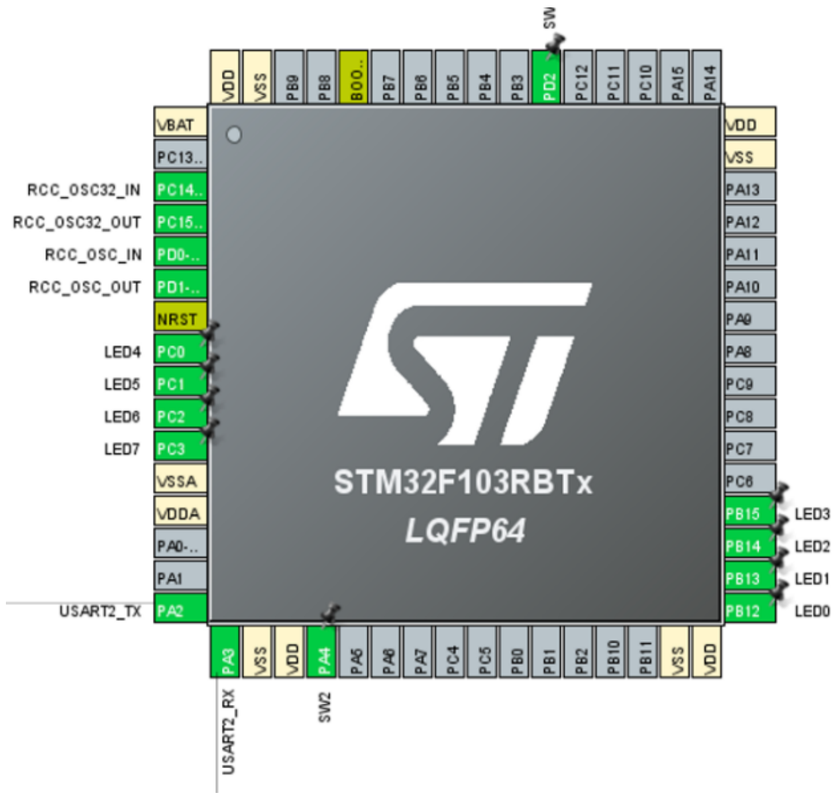
# *MICROPROCESSOR*

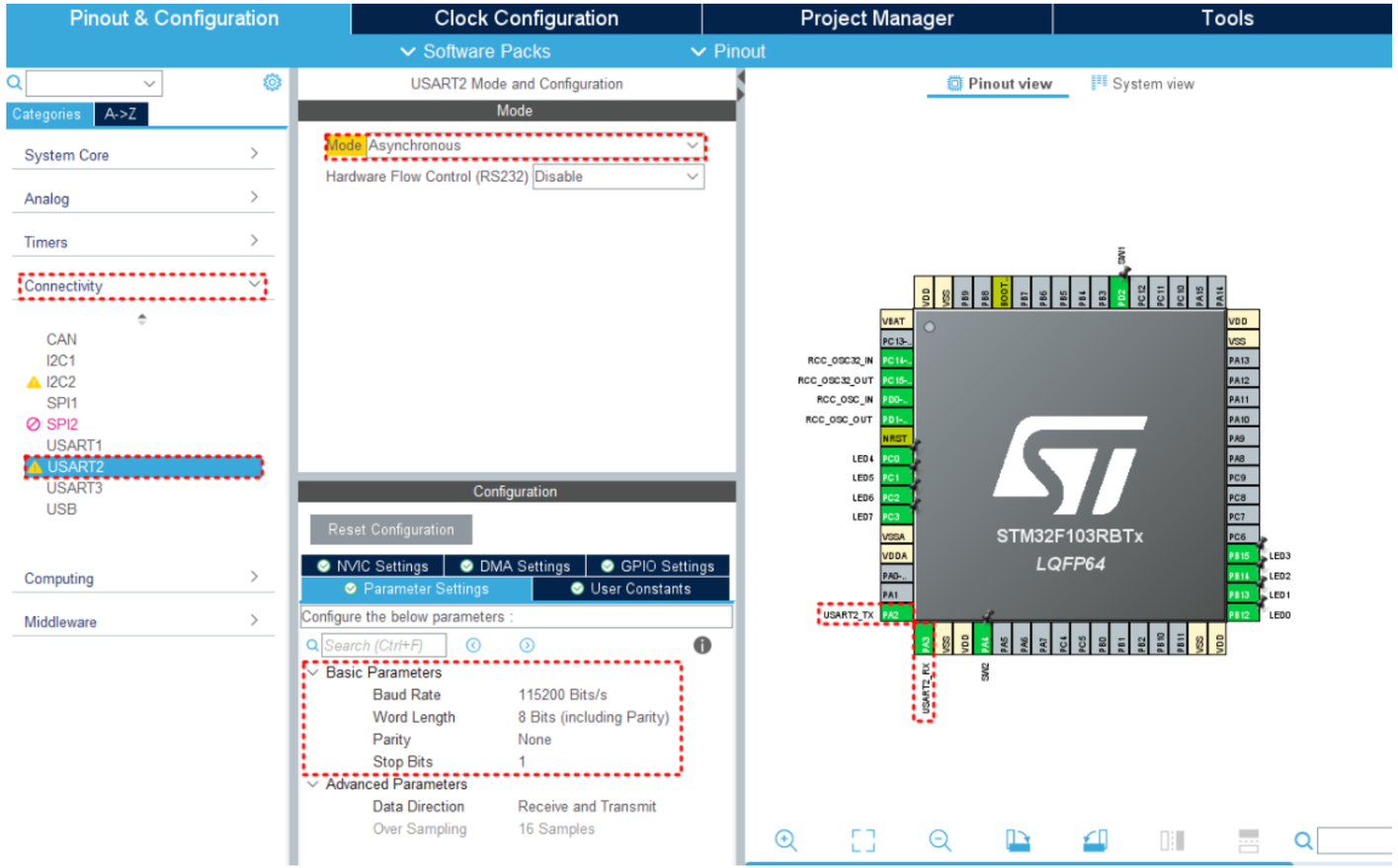
2309 양유빈

20230601 마이크로프로세서

# UART 기본 설정 방법

UART default setting method



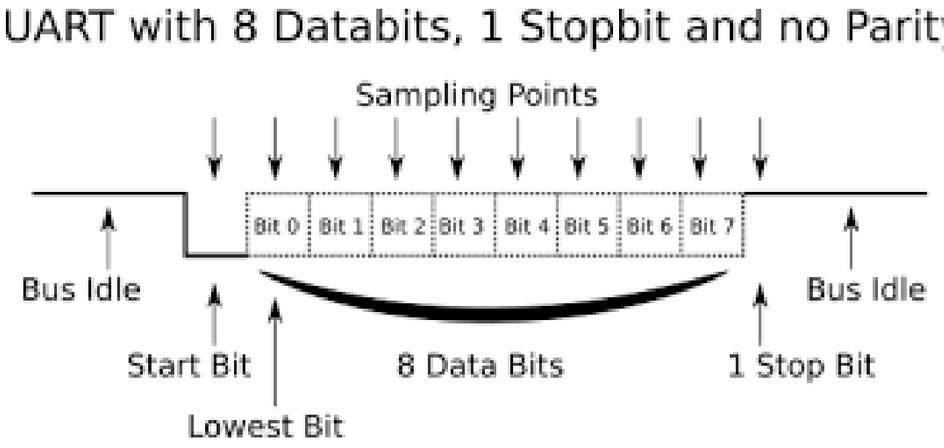



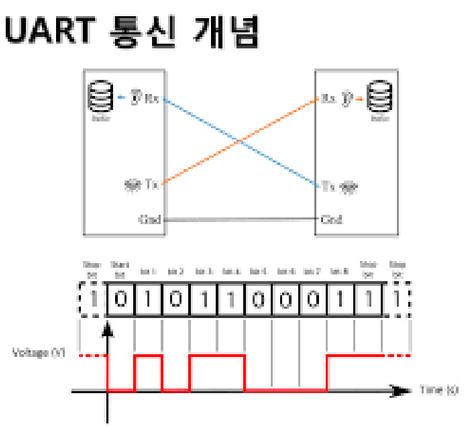
# UART 통신

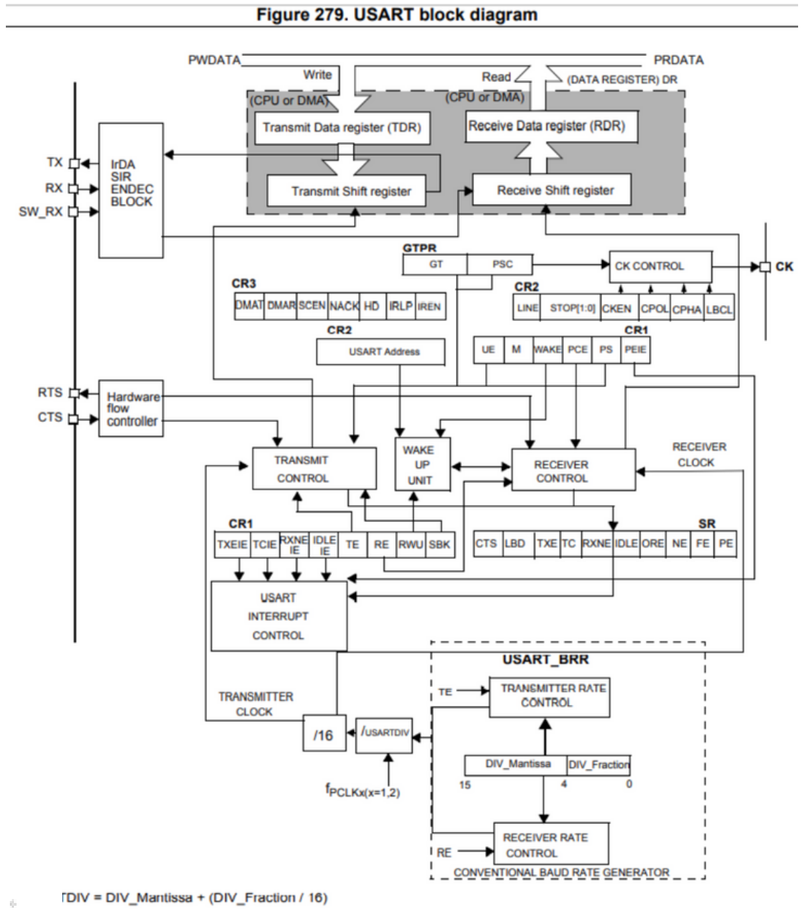
UART communication

- Universal Asynchronous Receiver/Transmitter - 범용 비동기 송수신기

  (참고)USART(Universal Synchronous/Asynchronous Receiver/Transmitter)
- 가장 오래되고, 가장 기본이 되는 시리얼 통신 방식
- 3선(TxD, RxD, GND) 연결 방식이 보통
- Baud Rate : 전송속도를 의미하며, 보통 9600, 115200이 가장 많이 사용됨




// STM32F103 USART 통신 블록다이어그램

# 터미널 에뮬레이터(Tera Term)

Terminal Emulator (Tera Term)

-사용자가 PC를 통하여 마이크로컨트롤러와 UART 통신을 하기 위하여 필요한 통신 프로그램

-종류

- Tera Term : https://osdn.net/projects/ttssh2/releases/
- Hyper Terminal, Hercules, Putty 등

-일반적인 초기 설정

- 데이터 비트 수 : 8
- 패리티 사용 여부 : 사용하지 않음
- 스톱 비트 수 : 1
- Baud Rate : 9600/38400/115200/… 등

**-포트: COM4, 속도: 115200, 수신: AUTO**

# 문자 전송 프로그램

texting program

```
#include "main.h"
/* USER CODE BEGIN PV */
uint8_t TxBuffer[] = "Embedded-Software\n";
/* USER CODE END PV */

int main(void) {
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    HAL_UART_Transmit(&huart2, TxBuffer, sizeof(TxBuffer), 100);
    HAL_Delay(1000);
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

# 송수신 프로그램

sending and receiving program

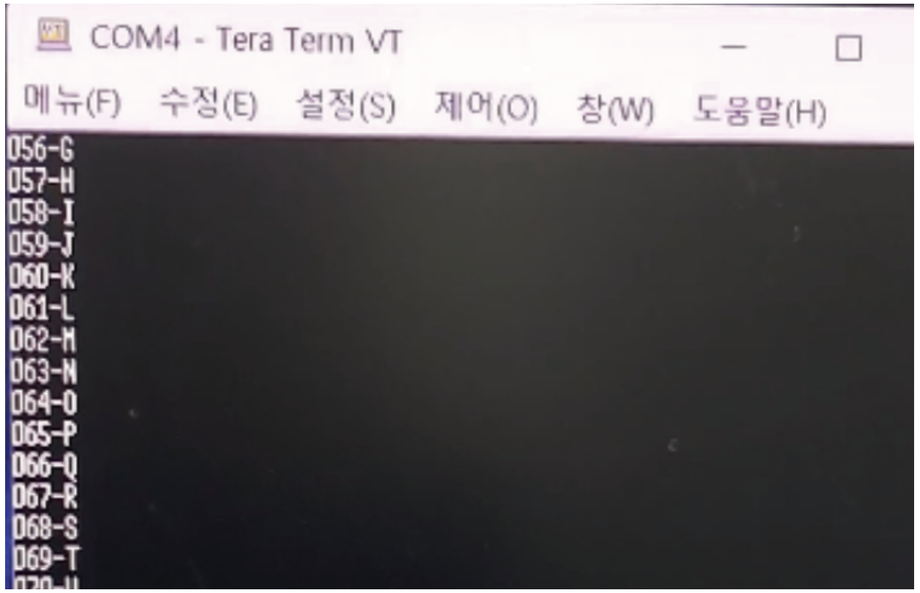
```
#include "main.h"
/* USER CODE BEGIN PV */
uint8_t TxBuffer[] = "Embedded-Software\n";
uint8_t TxBuffer_else[] = "1을 눌려주세요\n";
uint8_t RxBuffer[1];
/* USER CODE END PV */
int main(void) {
  /* USER CODE BEGIN WHILE */
  while (1) {
    if (HAL_UART_Receive(&huart2, RxBuffer, 1, 1) == HAL_OK) {
      if(RxBuffer[0] == 1)
        HAL_UART_Transmit(&huart2, TxBuffer, sizeof(TxBuffer), 100);
      else
        HAL_UART_Transmit(&huart2, TxBuffer_else, sizeof(TxBuffer_else), 100);
    }
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

# 순서가 있는 문자 전송 프로그램

Sequential text transmission program

```
#include "main.h"
/* USER CODE BEGIN PV */
uint8_t TxBuffer[6];
uint8_t num = 1, num100, num10, num1;
uint8_t alphabet = 'A';
/* USER CODE END PV */

int main(void) {
  /* USER CODE BEGIN WHILE */
  while (1) {
    num100 = num / 100;
    num10 = (num % 100) / 10;
    num1 = (num % 10);
    TxBuffer[0] = num100 + '0';
    TxBuffer[1] = num10 + '0';
    TxBuffer[2] = num1 + '0';
    TxBuffer[3] = '-';
    TxBuffer[4] = alphabet;
    TxBuffer[5] = '\n';
    HAL_UART_Transmit(&huart2, TxBuffer, sizeof(TxBuffer), 100);
    HAL_Delay(1000);
    if (++num > 100)  num = 1;
    if (++alphabet > 'Z')  alphabet = 'A';
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```



# UART 인터럽트

UART interrupt

-UART 송신 : 프로그램에서 필요한 시점에 바로 실행하면 되므로 인터럽트 처리의 필요성은 크지 않음

-UART 수신

- 필요한 데이터가 언제 수신될 지 모름
- UART Rx 포트의 수신 상태를 검사하여 데이터가 들어왔는지를 검사 한다는 것(폴링)은 시간적인 낭비이며, 언제 어느 곳에서 하여야 할 지를 정해야 하는 것도 쉽지 않음
- 그래서, UART의 수신 데이터의 처리는 폴링이 아닌 인터럽트를 이용하여 처리하는 것이 일반적임

+ UART 인터럽트 관련 HAL 라이브러리

| | |
|---|---|
| 형태 | HAL_StatusTypeDef HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size) |
| 설명 | 원하는 크기(Size)의 데이터(*pData)를 수신 |
| 파라미터 | huart : UART 핸들(handle)을 선언한 포인터 값 (&huart, &huart2, …)pData : 수신할 데이터를 저장할 위치(포인터) 지정Size : 수신할 데이터의 최대 크기 |
| 리턴값 | 수신 결과(HAL_OK, HAL_ERROR, HAL_BUSY, HAL_TIMEOUT) |
| 예시 | huart1 핸들을 통하여 인터럽트 방식으로 10개의 문자를 수신하고 싶을 때ex) HAL_UART_Receive_IT(&huart1, pt, 10); // pt는 수신 데이터를 저장할 장소를 가리키는 포인터 |

# 터미널 제어 비상등 프로그램

## Terminal control emergency light program

```
#include "main.h"
/* USER CODE BEGIN PD */
#define MODE_A   1
#define MODE_B   2
#define MODE_OFF  3
/* USER CODE END PD */
/* USER CODE BEGIN PV */
volatile uint8_t ambulance_mode;
uint8_t RxBuffer[1];
uint8_t message_1[] = "\n\n**********\n";
uint8_t message_2[] = "Emergency Light Control\n";
uint8_t message_3[] = "**********\n\n";
uint8_t message_4[] = "1. Set to [Ambulance_A] mode\n";
uint8_t message_5[] = "2. Set to [Ambulance_B] mode\n";
uint8_t message_6[] = "3. Set to [Off] mode\n";
uint8_t message_7[] = "4. Inquire current Emergency mode\n\n";
uint8_t message_8[] = "Type number : ";
uint8_t message_9[] = "\n\nNow, [Ambulance_A] mode\n";
uint8_t message_10[] = "\n\nNow, [Ambulance_B] mode\n";
uint8_t message_11[] = "\n\nNow, [Off] mode\n";
uint8_t message_12[] = "\n\nCurrent mode is [Ambulance_A] mode\n";
uint8_t message_13[] = "\n\nCurrent mode is [Ambulance_B] mode\n";
uint8_t message_14[] = "\n\nCurrent mode is [Off] mode\n";
uint8_t message_15[] = "\n\nSelect number : 1 ~ 4\n";
/* USER CODE END PV */
```

```
/* USER CODE BEGIN 0 */
void display_menu() {
  HAL_UART_Transmit(&huart2, message_1, sizeof(message_1), 100);
  HAL_UART_Transmit(&huart2, message_2, sizeof(message_2), 100);
  HAL_UART_Transmit(&huart2, message_3, sizeof(message_3), 100);
  HAL_UART_Transmit(&huart2, message_4, sizeof(message_4), 100);
  HAL_UART_Transmit(&huart2, message_5, sizeof(message_5), 100);
  HAL_UART_Transmit(&huart2, message_6, sizeof(message_6), 100);
  HAL_UART_Transmit(&huart2, message_7, sizeof(message_7), 100);
  HAL_UART_Transmit(&huart2, message_8, sizeof(message_8), 100);
}

typedef struct led {
  GPIO_TypeDef *port;
  uint16_t pin;
   } LED;
LED led[8] = {
  {GPIOC, GPIO_PIN_3},
  {GPIOC, GPIO_PIN_2},
  {GPIOC, GPIO_PIN_1},
  {GPIOC, GPIO_PIN_0},
  {GPIOB, GPIO_PIN_15},
  {GPIOB, GPIO_PIN_14},
  {GPIOB, GPIO_PIN_13},
  {GPIOB, GPIO_PIN_12},
};
```

```
void led_all_off() {
  for (uint8_t k = 0; k < 8; k++) {
    HAL_GPIO_WritePin(led[k].port, led[k].pin, GPIO_PIN_RESET);
  }
}
void ambulance_a() {
  for (uint8_t i = 0; i < 4; i++)  {
    HAL_GPIO_WritePin(led[i].port, led[i].pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(led[i+4].port, led[i+4].pin, GPIO_PIN_RESET);
    }
  HAL_Delay(500);
  if (ambulance_mode == MODE_B || ambulance_mode == MODE_OFF) {
    led_all_off();
    return;
  }
  for (uint8_t j = 0; j < 4; j++)  {
    HAL_GPIO_WritePin(led[j].port, led[j].pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(led[j+4].port, led[j+4].pin, GPIO_PIN_SET);
    }
  HAL_Delay(500);
  if (ambulance_mode == MODE_B || ambulance_mode == MODE_OFF) {
    led_all_off();
    return;
  }
}
```

# 터미널 제어 비상등 프로그램

## Terminal control emergency light program

```
void ambulance_b() {
  for (uint8_t i = 0; i < 8; i++)  {
    HAL_GPIO_WritePin(led[i].port, led[i].pin, GPIO_PIN_SET);
    HAL_Delay(100);
    HAL_GPIO_WritePin(led[i].port, led[i].pin, GPIO_PIN_RESET);
    HAL_Delay(100);
    if (ambulance_mode == MODE_A || ambulance_mode == MODE_OFF) {
      led_all_off();
      return;
    }
  }
  for (uint8_t j = 6; j > 0; j--)  {
    HAL_GPIO_WritePin(led[j].port, led[j].pin, GPIO_PIN_SET);
    HAL_Delay(100);
    HAL_GPIO_WritePin(led[j].port, led[j].pin, GPIO_PIN_RESET);
    HAL_Delay(100);
    if (ambulance_mode == MODE_A || ambulance_mode == MODE_OFF) {
      led_all_off();
      return;
    }
  }
}
/* USER CODE END 0 */
```

```
int main(void) {
  /* USER CODE BEGIN 2 */
  display_menu();
  HAL_UART_Receive_IT(&huart2, RxBuffer, 1);
  /* USER CODE END 2 */
  /* USER CODE BEGIN WHILE */
  while (1) {
    if (ambulance_mode == MODE_A)
      ambulance_a();
    else if (ambulance_mode == MODE_B)
      ambulance_b();
    else
      led_all_off();
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/* USER CODE BEGIN 4 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
  switch (RxBuffer[0]) {
    case '1':
      ambulance_mode = MODE_A;
      HAL_UART_Transmit(&huart2, RxBuffer, sizeof(RxBuffer), 100);
      HAL_UART_Transmit(&huart2, message_9, sizeof(message_9), 100);
      display_menu();
      break;
    case '2':
      ambulance_mode = MODE_B;
      HAL_UART_Transmit(&huart2, RxBuffer, sizeof(RxBuffer), 100);
      HAL_UART_Transmit(&huart2, message_10, sizeof(message_10), 100);
      display_menu();
      break;
    case '3':
      ambulance_mode = MODE_OFF;
      HAL_UART_Transmit(&huart2, RxBuffer, sizeof(RxBuffer), 100);
      HAL_UART_Transmit(&huart2, message_11, sizeof(message_11), 100);
      display_menu();
      break;
    case '4':
      HAL_UART_Transmit(&huart2, RxBuffer, sizeof(RxBuffer), 100);
      if (ambulance_mode == MODE_A)
        HAL_UART_Transmit(&huart2, message_12, sizeof(message_12), 100);
      else if (ambulance_mode == MODE_B)
        HAL_UART_Transmit(&huart2, message_13, sizeof(message_13), 100);
      else
        HAL_UART_Transmit(&huart2, message_14, sizeof(message_14), 100);
      display_menu();
      break;
    default :
      RxBuffer[0] = '\a';
      HAL_UART_Transmit(&huart2, RxBuffer, sizeof(RxBuffer), 100);
      HAL_UART_Transmit(&huart2, message_15, sizeof(message_15), 100);
      display_menu();
      break;
  }
  HAL_UART_Receive_IT(&huart2, RxBuffer, 1);
}
/* USER CODE END 4 */
```