

# **TI DSP, MCU, Xilinx Zynq FPGA**

## **프로그래밍 전문가 과정**

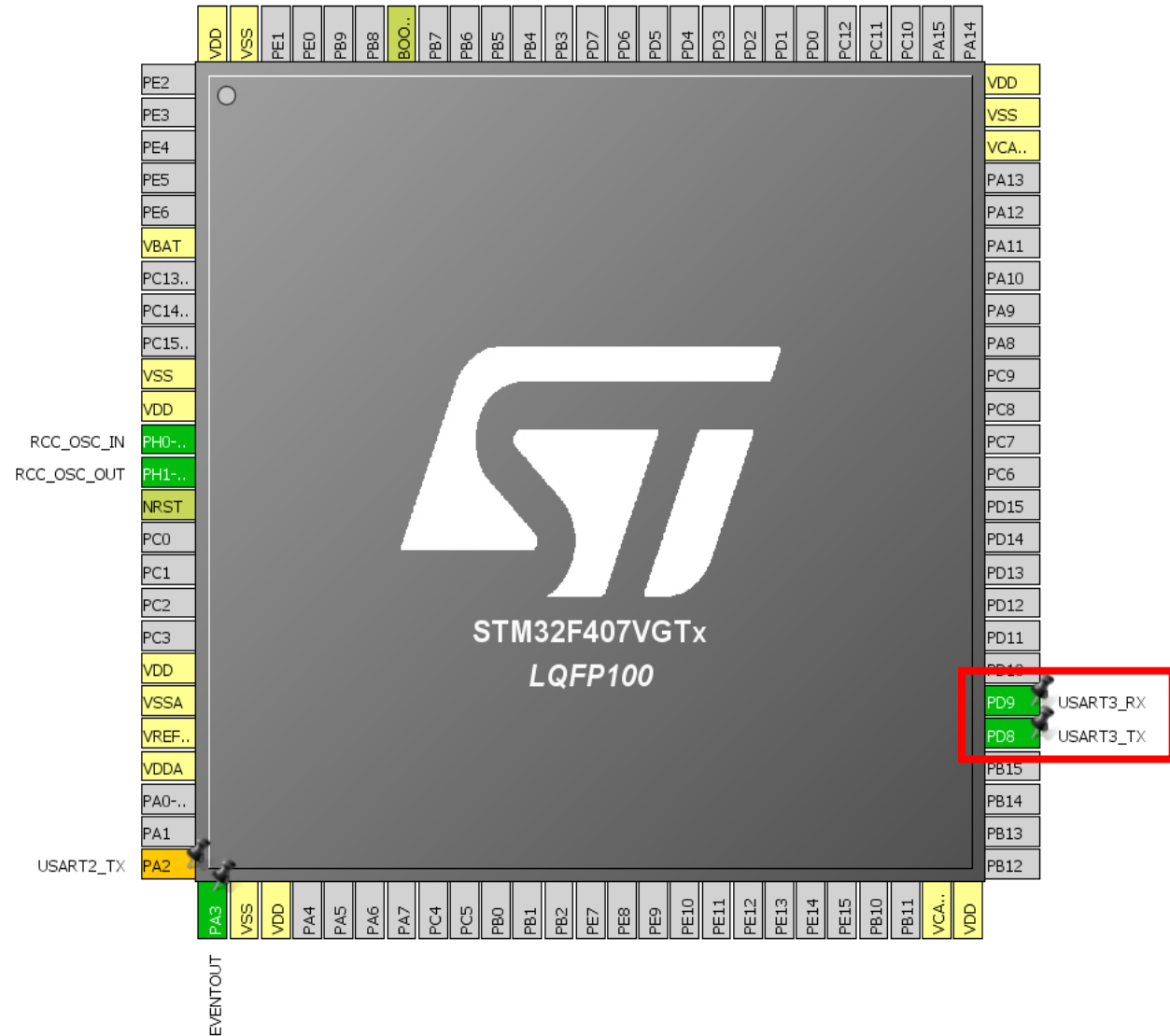
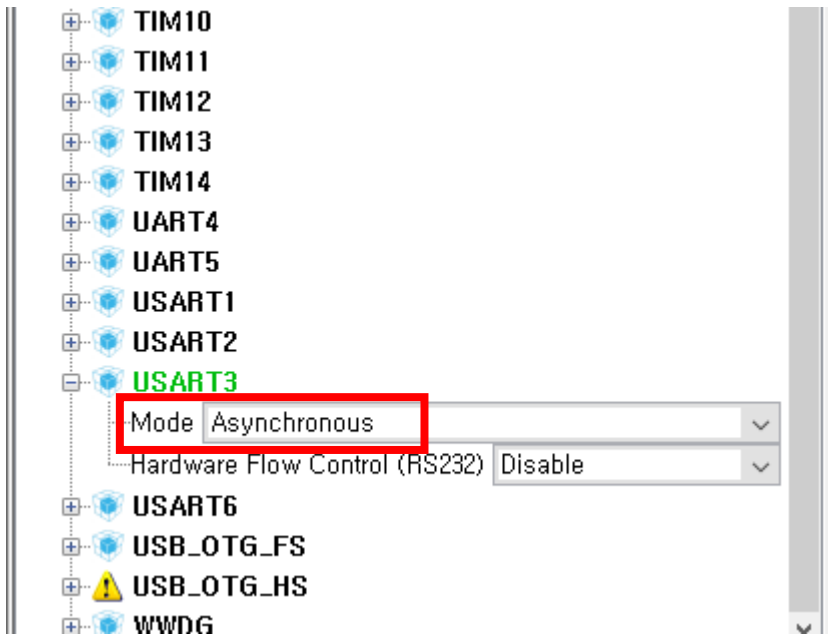
**UART based on STM32F407**

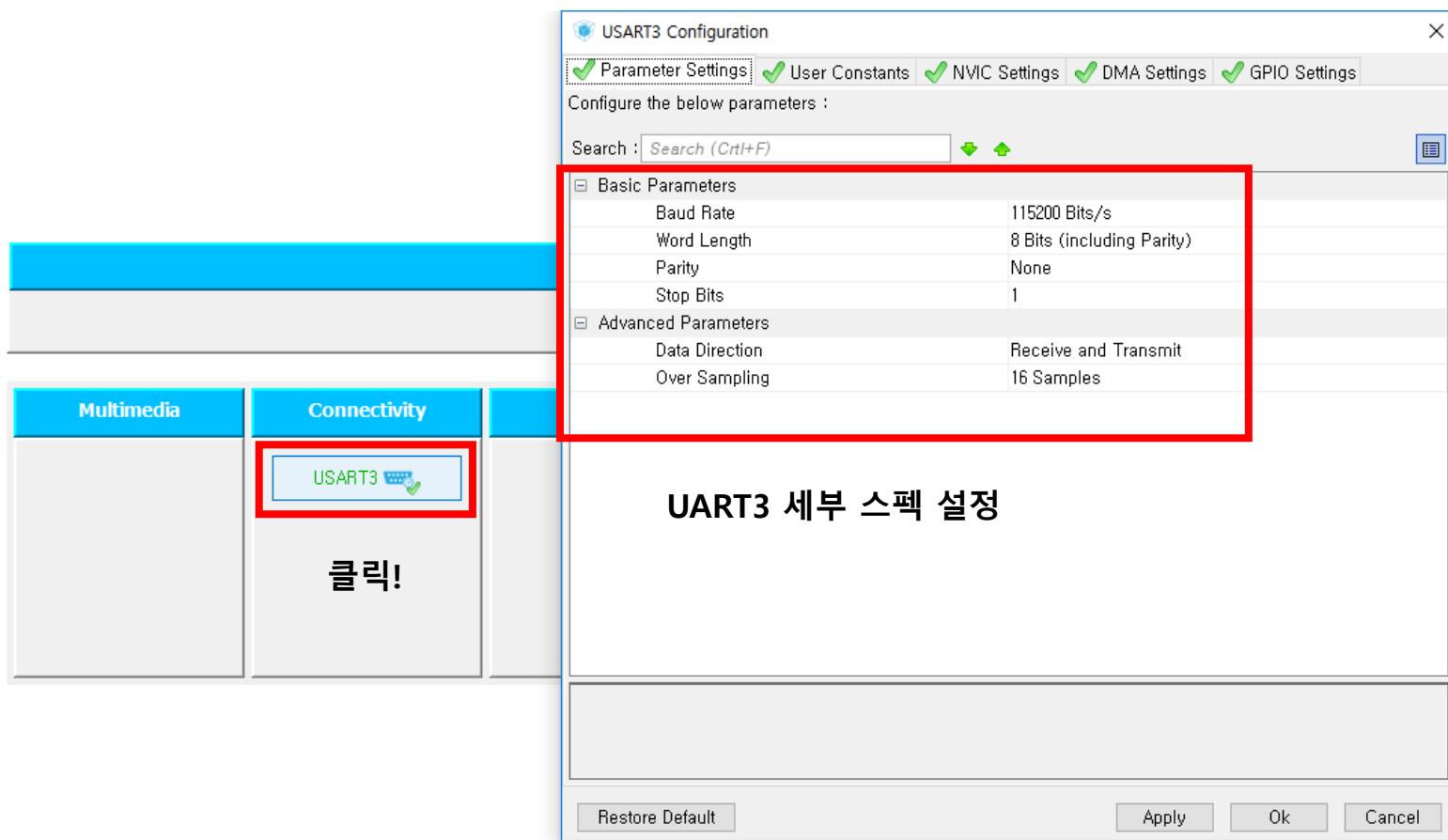
**강사 – Innova Lee(이상훈)**  
gcccompil3r@gmail.com

**학생 – 안상재**  
sangjae2015@naver.com

## 1. UART 풀링방식

- CubeMX 설정 (클럭 트리 설정 생략)

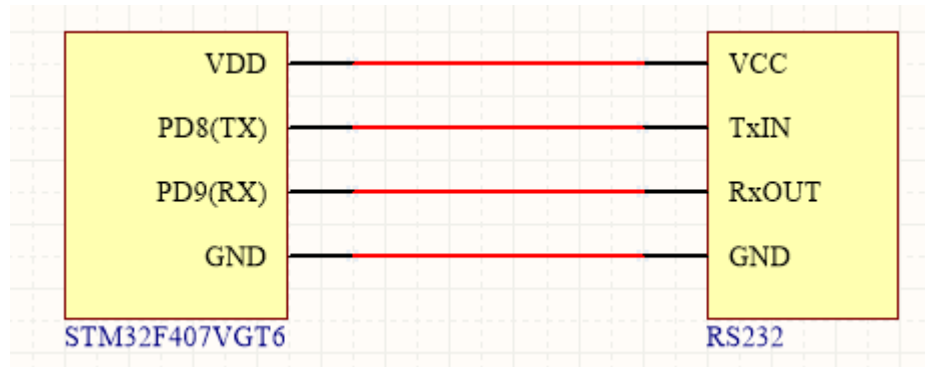




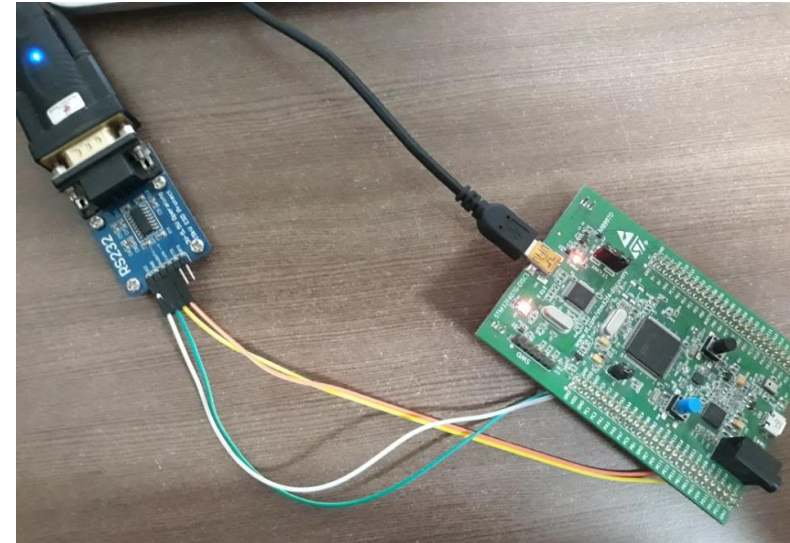
=> 단축키 CTRL+SHIFT+'G' 로 코드 생성

## - 하드웨어 연결 상태

USB to RS232 케이블에 바로 연결해도 되지만, 편의상 RS232 모듈을 사용함!



MCU-RS232 회로도 (UART3)



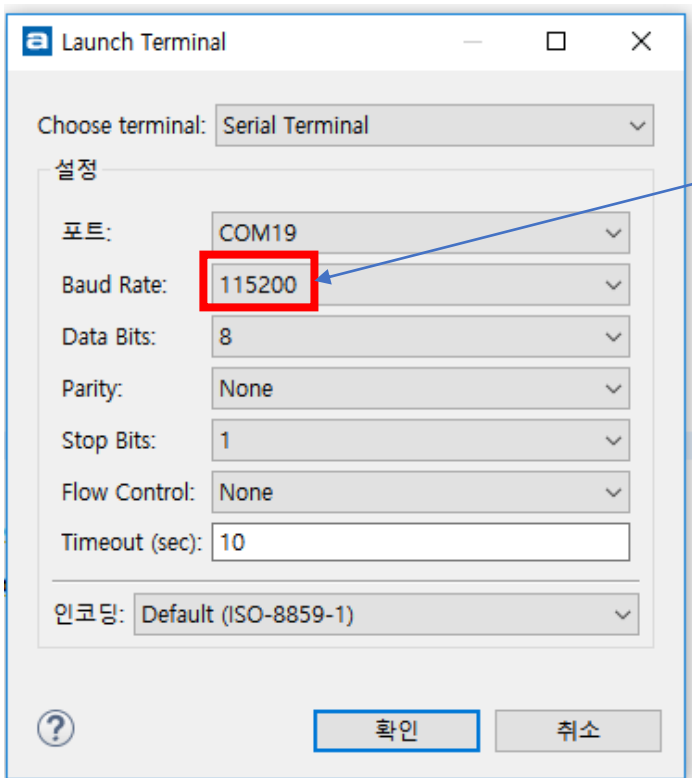
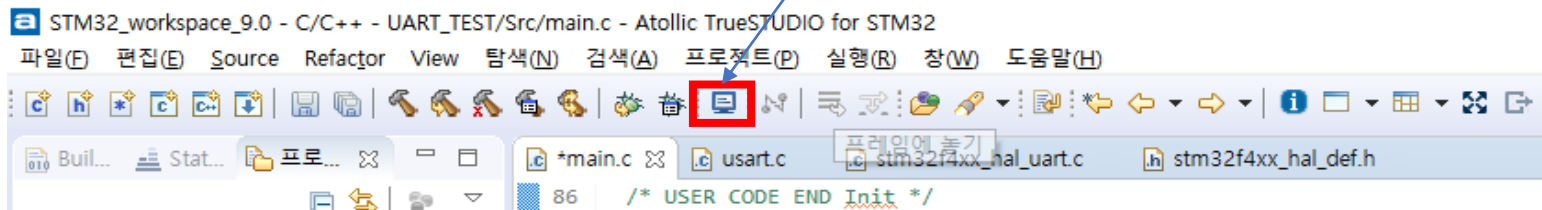
하드웨어 연결 모습

## - 코드

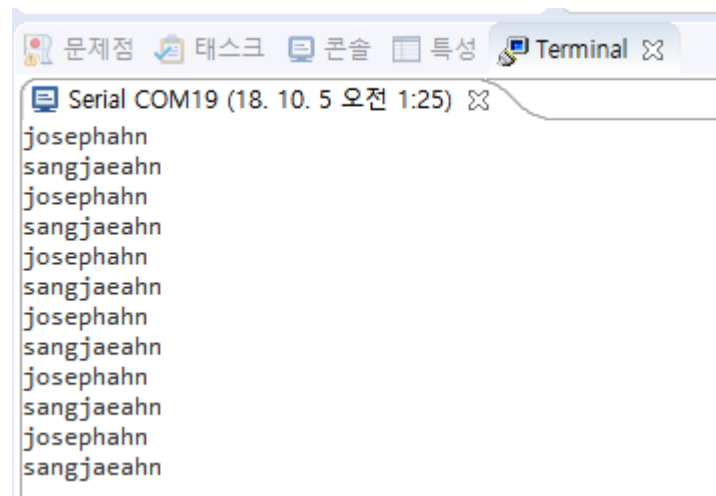
```
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
    if(HAL_UART_Receive(&huart3, &recv, 1, 10) == HAL_OK)    <= 폴링방식일 경우 수신을 성공 했는지 체크해야함!
    {
        if(recv == 'a')
            HAL_UART_Transmit(&huart3, "josephahn\r\n", 11, 10);
        else if(recv == 's')
            HAL_UART_Transmit(&huart3, "sangjaeahn\r\n", 12, 10);
    }
}
/* USER CODE END 3 */
```

터미널 창



장치 관리자 확인



콘솔창 화면

터미널창 통신 설정

## 2. UART printf 사용하기

- printf() 함수 내부에 \_write() 함수가 존재하고, \_write() 함수의 정의부를 바꾸어서 printf() 함수의 출력 대상을 UART로 바꾸어줌!

main 문 위에 \_write() 함수를 정의 함

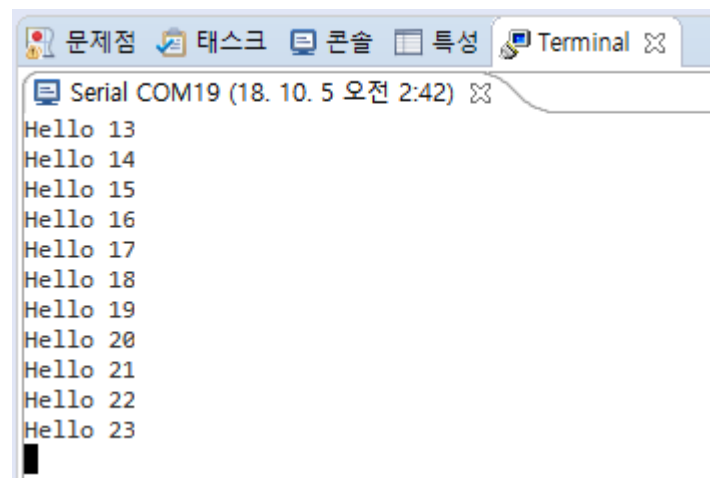
```
int _write(int file, char* p, int len) /* printf 함수 내부의 write 함수 */
{
    HAL_UART_Transmit(&huart3, p, len, 10);
    return len;
}
```

main 문

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

    /* USER CODE BEGIN 3 */
    printf("Hello %d\r\n", a++);
    HAL_Delay(1000);
}
/* USER CODE END 3 */
```

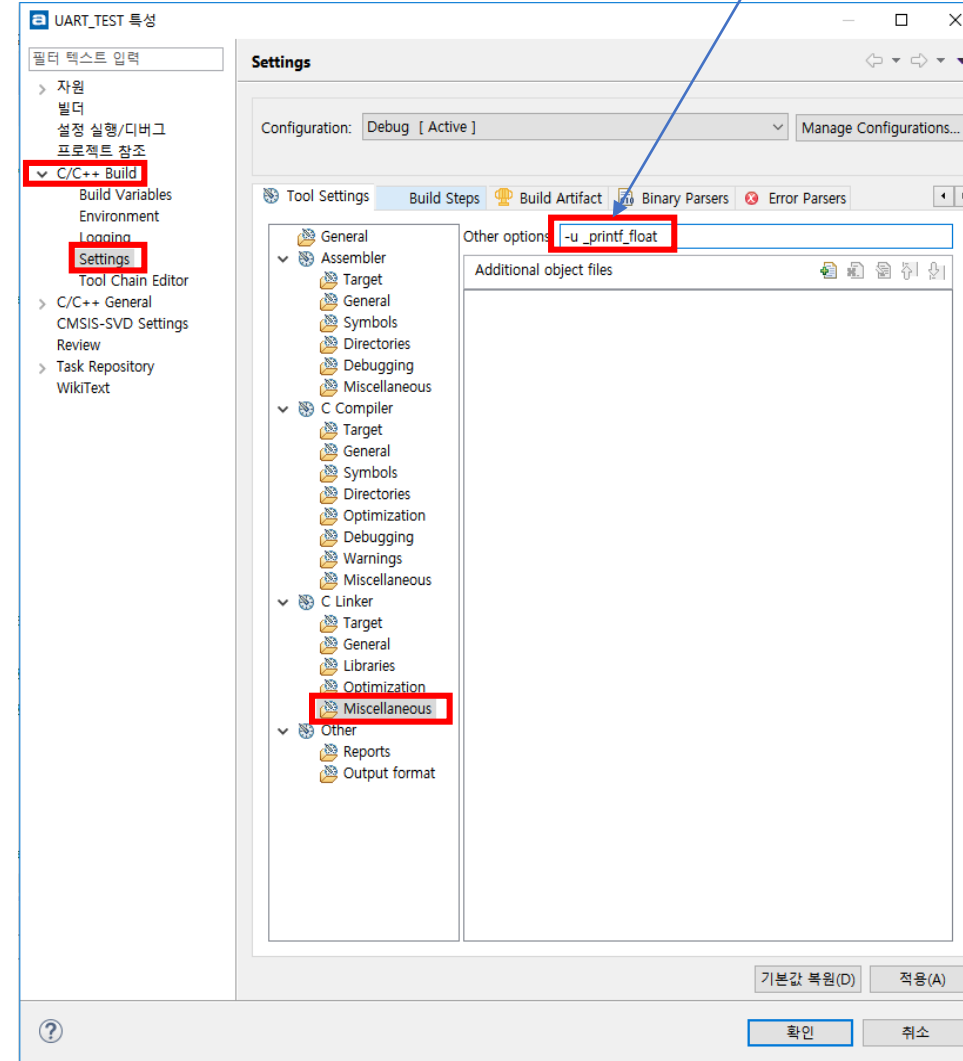
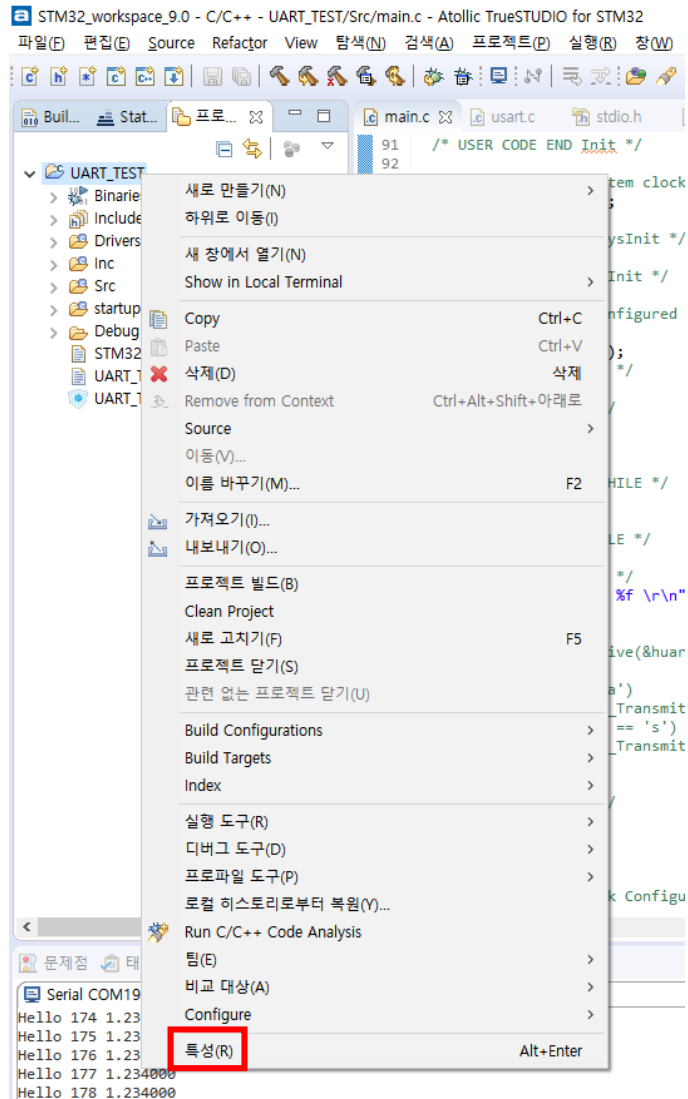
터미널 창



## \* 소숫점 출력 방법

-> 대부분의 마이크로프로세서에서는 printf()함수의 출력이 정수 형태로 출력이 되도록 되어 있음!

옵션 추가





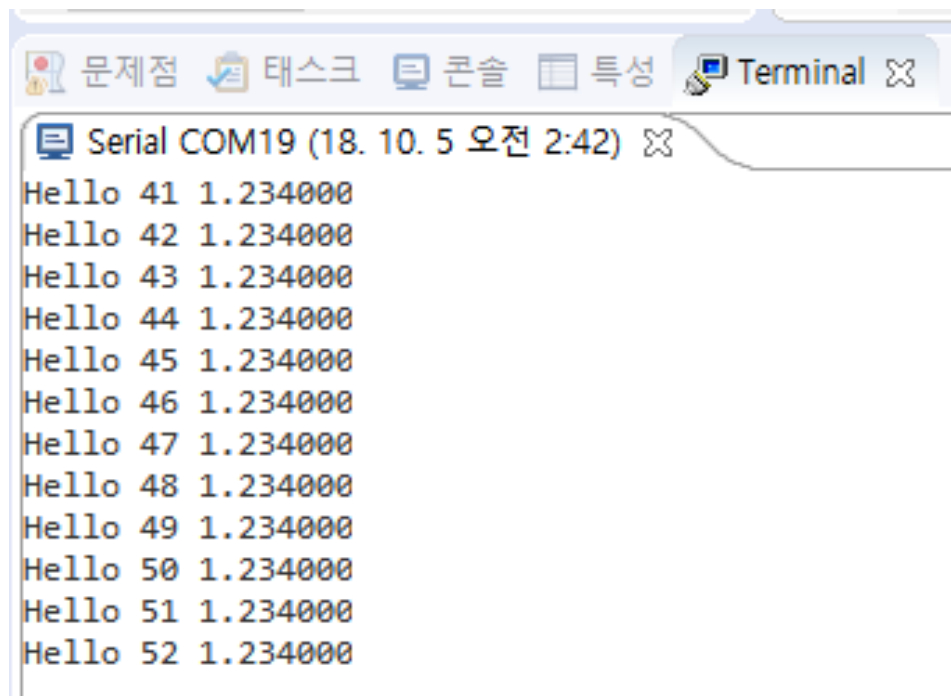
## - 소숫점 출력

### main 문

```
uint8_t a=0;
float f = 1.234;
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
    printf("Hello %d %f\r\n", a++, f);
    HAL_Delay(1000);
}
/* USER CODE END 3 */
```

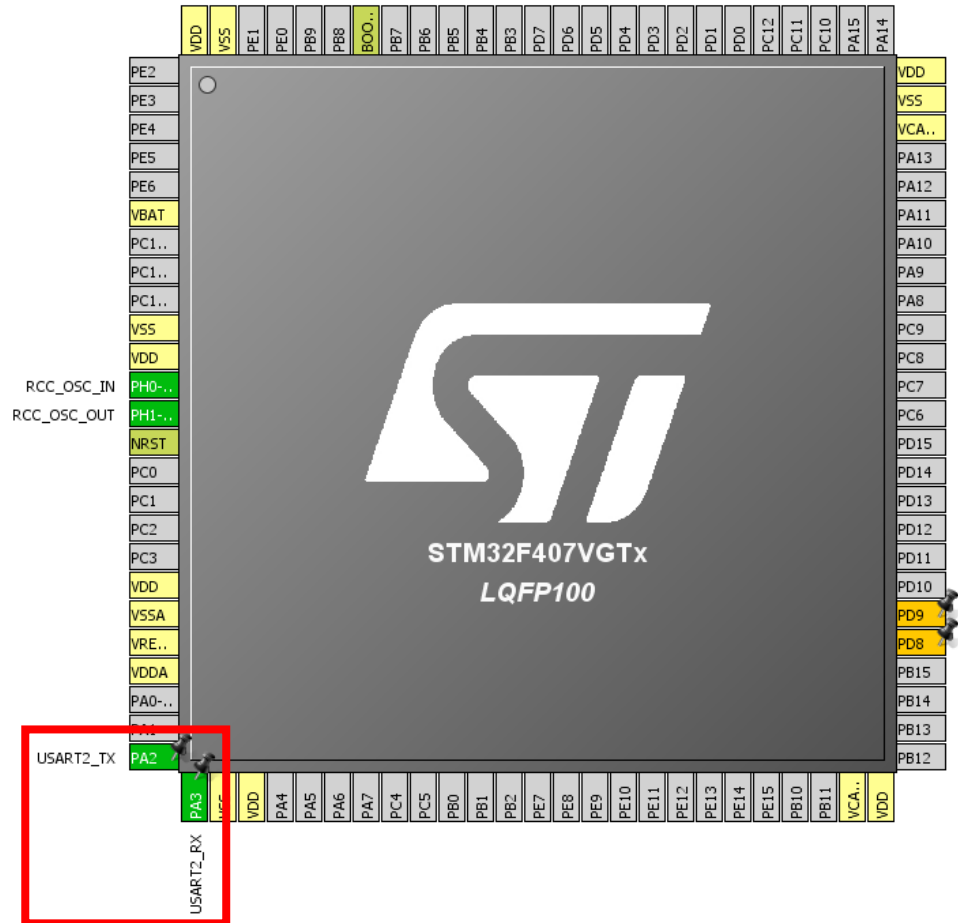
### 터미널 창

A screenshot of a terminal window with a light blue header bar containing icons and labels for '문제점', '태스크', '콘솔', '특성', and 'Terminal'. The 'Terminal' tab is active. Below the header, the title bar reads 'Serial COM19 (18. 10. 5 오전 2:42)'. The terminal content shows a series of lines: 'Hello 41 1.234000' through 'Hello 52 1.234000', where the integer part of the float increases by 1 each line, while the float part remains constant at 1.234000.

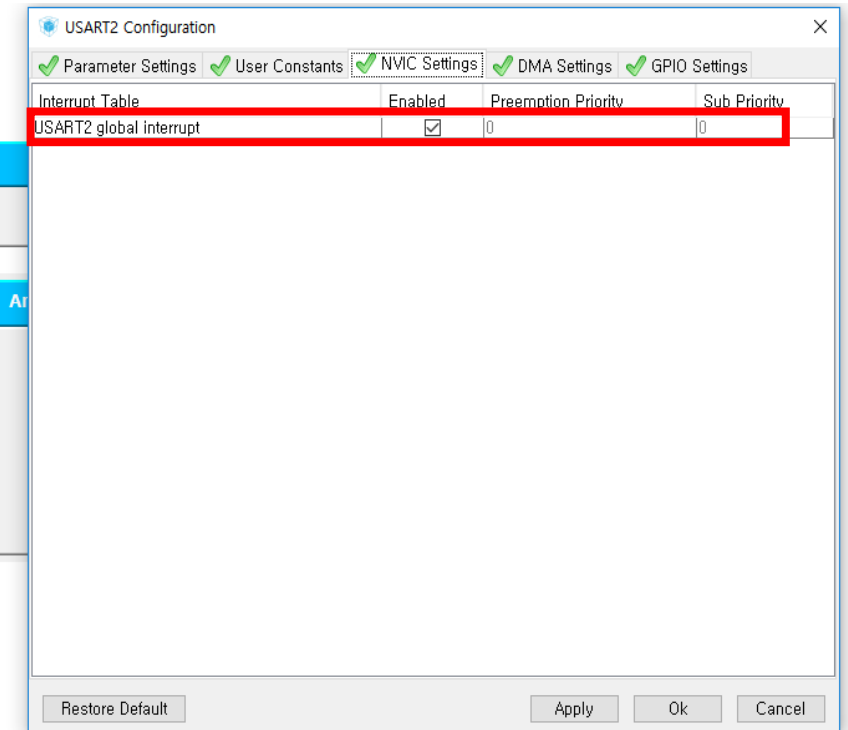
```
문제점 태스크 콘솔 특성 Terminal
Serial COM19 (18. 10. 5 오전 2:42)
Hello 41 1.234000
Hello 42 1.234000
Hello 43 1.234000
Hello 44 1.234000
Hello 45 1.234000
Hello 46 1.234000
Hello 47 1.234000
Hello 48 1.234000
Hello 49 1.234000
Hello 50 1.234000
Hello 51 1.234000
Hello 52 1.234000
```

### 3. UART 인터럽트

- CubeMX 설정



configuration 탭



# configuration 탭

Middlewares

Multimedia

Connectivity

USART2

Analog

System

DMA

GPIO

NVIC

RCC

Control

Security

NVIC Configuration

✓ NVIC

✓ Code generation

Priority Group 4 bits for pre-emption priority 0 bits for ...

☐ Sort by Preemption Priority and Sub Priority

Search 

☐ Show only enabled interrupts

Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
USART2 global interrupt	<input checked="" type="checkbox"/>	0	0
FSMC global interrupt	<input type="checkbox"/>	0	0

☐ Enabled    Preemption Priority 

▼

    Sub Priority 

▼

Apply

Ok

Cancel

NVIC Configuration

✓ NVIC

✓ Code generation

Enabled interrupt table

☒ Select for init sequence ordering

☒ Generate IRQ handler

Non maskable interrupt	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Hard fault interrupt	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Memory management fault	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Pre-fetch fault, memory access fault	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Undefined instruction or illegal state	<input type="checkbox"/>	<input checked="" type="checkbox"/>
System service call via SWI instruction	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Debug monitor	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Pendable request for system service	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Time base: System tick timer	<input type="checkbox"/>	<input checked="" type="checkbox"/>
USART2 global interrupt	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Interrupt unmasking ordering table (interrupt init code is moved after all the peripheral init code)

Rank	Interrupt name
1	USART2 global interrupt

↓

↑

Apply

Ok

Cancel

## - 소스 코드

```
79 int main(void)
80 {
81     /* USER CODE BEGIN 1 */
82     uint8_t rcv = 0;
83     /* USER CODE END 1 */
84
85     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
86     HAL_Init();
87
88     /* Configure the system clock */
89     SystemClock_Config();
90
91     /* Initialize all configured peripherals */
92     MX_GPIO_Init();
93     MX_USART2_UART_Init();
94
95     /* Initialize interrupts */
96     MX_NVIC_Init();
97     /* USER CODE BEGIN 2 */
98
99     // rx3_data에 1바이트 채워지면 인터럽트를 호출하겠다!
100    // 이 코드가 없으면 인터럽트 핸들러 자체가 안걸림!
101    HAL_UART_Receive_IT(&huart2, &rx, 1);
102    /* USER CODE END 2 */
103
104    /* Infinite loop */
105    /* USER CODE BEGIN WHILE */
106    while (1)
107    {
108        /* USER CODE END WHILE */
109
110        /* USER CODE BEGIN 3 */
111        HAL_Delay(1000);
112    }
113    /* USER CODE END 3 */
114
115 }
```

← main 문 안에서 반드시 써줘야함!

## IRQ 핸들러

```
void USART2_IRQHandler(void)
{
    /* USER CODE BEGIN USART2_IRQn 0 */

    /* USER CODE END USART2_IRQn 0 */
    HAL_UART_IRQHandler(&huart2);
    /* USER CODE BEGIN USART2_IRQn 1 */

    /* USER CODE END USART2_IRQn 1 */
}
```

IRQ 핸들러에서 이 함수를 호출하게 됨!

=> main 문 밑에서 정의부를 다시 해주어야함!

```
187
188 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
189 {
190     if(huart->Instance == USART2)
191     {
192         HAL_UART_Receive_IT(&huart2, &rx, 1);
193         if(rx == 'a')
194             HAL_UART_Transmit(&huart2, "josephahn\r\n", 11, 10);
195         else if(rx == 's')
196             HAL_UART_Transmit(&huart2, "sangjaeahn\r\n", 12, 10);
197     }
198 }
```

- 결과 화면

필자는 점퍼선을 사용하기 귀찮아서,

확장 보드와 RS232 모듈을 연결해서 UART를 구현했음! (UART2 사용)

