REPORT

6주차 결과 보고서

수강과목: 임베디드 시스템 설계 및 설계

담당교수: 백윤주 교수님

학 과: 전기컴퓨터공학부 정보컴퓨터공학전공

조 원: 201724651 김장환 201724648 김경호

201624481 박윤형 201524588 조창흠

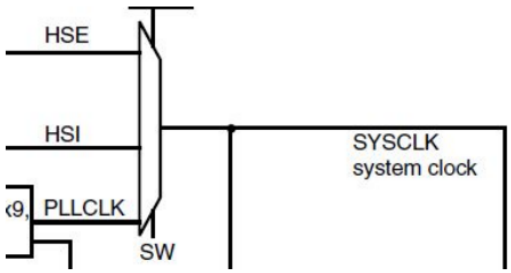
제출일자: 2021. 10. 12

실험 목표

1. Clock Tree의 이해 및 사용자 Clock 설정
2. UART 통신의 원리를 배우고, 실제 설정 방법 파악

1. Clock

System clock(SYSCLK)를 구동하기 위해 사용되는 Clock은 크게 3가지로 분류될 수 있다.



1) HIS Clock (High-speed internal Clock)

   STM32에 내장되어 있는 RC발진 회로로 전원 인가 시 처음 동작하는 Clock이다.

2) HSE Clock (High-speed external Clock)

   STM32의 외부에서 입력되는 높은 주파수의 Clock으로 PLL을 거쳐 System Clock으로 입력된다. Crystal, Resonator와 같은 발진 소자를 사용하거나 외부 Clock Source 또는 Oscilator를 사용한다.

   - HSE external crystal/ceramic resonator
   - HSE user external clock

3) PLL (Phase-Locked Loop)

   HSE와 HIS Clock 중 하나를 선택해서 증폭시켜 준 후 최종적으로 오는 Clock을 의미.

2. UART

병렬 데이터의 형태를 직렬 방식으로 전환하여 데이터를 전송하는 컴퓨터 하드웨어의 일종으로 마치 사람이 대화하는 것과 같은 원리를 가지고 있다. UART를 하기 위해서는 Rx(데이터 수신, Tx(데이터 송신), GND가 서로 연결되어야 하며, 비동기 통신이므로, 둘 사이의 Baud Rate를 일치시켜줘야 한다.

실험 내용

임베디드 시스템 클럭 설정: SetSysClock() 함수에서 RCC->CFGR, RCC->CFGR2를 이용한 클럭설정

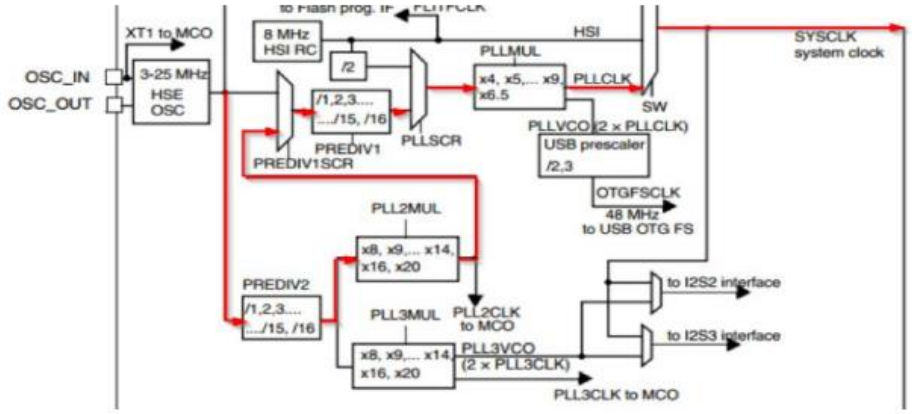
SYSCLK	52MHz
PCLK2	26MHz
Baud Rate	14400

미션 힌트
52 = 25 / 5 * 13 / 5 * 4

- 위의 정보를 통해 아래와 같이 코드를 작성할 수 있다.

```
//@TODO - 1 Set the clock
        /* HCLK = SYSCLK */
        RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;
        /* PCLK2 = HCLK / ?, use PPRE2 */
        //RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE2_DIV1; original
        RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE2_DIV2;
        /* PCLK1 = HCLK */
        RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE1_DIV1;

        /* Configure PLLs ------------------------------------------------------*/
        RCC->CFGR &= (uint32_t)~(RCC_CFGR_PLLXTPRE | RCC_CFGR_PLLSRC | RCC_CFGR_PLLMULL);
        RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLXTPRE_PREDIV1 | RCC_CFGR_PLLSRC_PREDIV1 | RCC_CFGR_PLLMULL4);

        RCC->CFGR2 &= (uint32_t)~(RCC_CFGR2_PREDIV2 | RCC_CFGR2_PLL2MUL | RCC_CFGR2_PREDIV1 | RCC_CFGR2_PREDIV1SRC);
        RCC->CFGR2 |= (uint32_t)(RCC_CFGR2_PREDIV2_DIV5 | RCC_CFGR2_PLL2MUL13 | RCC_CFGR2_PREDIV1SRC_PLL2 | RCC_CFGR2_PREDIV1_DIV5);
//@End of TODO - 1
```

- 위의 코드는 임베디드 시스템 클럭을 설정하는 코드 부분인데, 이 코드 이해를 위해 아래의 클럭 트리를 살펴보면, 다음과 같은 경로로 클럭이 설정이 된다.

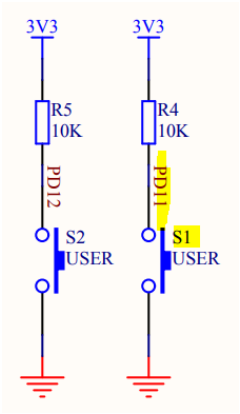


위에서 주어진 정보처럼 시스템 클럭을 52MHz로 맞추기 위해서는, 미션 힌트처럼 기존 클럭 25MHz에서 25MHz / 5 * 13 / 5 * 4 = 52MHz가 되어야 한다.

그 다음 추가로, PCLK2는 52MHz의 절반인 26MHz가 되어야 하므로 DIV2를 한 번 더 넣어준다.

그럼 이제, User S1 버튼을 누르는 동안 터미널 프로그램(Putty)을 통해 "Hello Team08"을 출력하기 위해, User S1 버튼을 설정해주어야 한다.

STM32의 스키메틱의 사진 중에, 오른쪽의 사진을 찾을 수 있는데, 거기서 User S1이 PD11인 것을 확인할 수 있다. 이것은 Port D의 11번을 쓴다는 얘기이다.



9.2.2 Port configuration register high (GPIOx_CRH) (x=A..G)

Address offset: 0x04

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2 **CNFy[1:0]:** Port x configuration bits (y= 8 .. 15)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table on page 161.*
In input mode (MODE[1:0]=00):
00: Analog mode
01: Floating input (reset state)
10: Input with pull-up / pull-down
11: Reserved
In output mode (MODE[1:0] > 00):
00: General purpose output push-pull
01: General purpose output Open-drain
10: Alternate function output Push-pull
11: Alternate function output Open-drain

Bits 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0 **MODEy[1:0]:** Port x mode bits (y= 8 .. 15)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table on page 161.*
00: Input mode (reset state)
01: Output mode, max speed 10 MHz.
10: Output mode, max speed 2 MHz.
11: Output mode, max speed 50 MHz.

그리고, 위의 사진을 보자. 우리는 GPIOD의 CRH를 쓰며, Configuration을 할 거기 때문에, Reference의 CRH 부분에서 11번을 쓰기 때문에, GPIO_CRH_CNF11 중에서, GPIO_CRH_CNF11_0 혹은 GPIO_CRH_CNF11_1 둘 중 하나가 들어오는 신호를 S1 신호로 받아야한다. 여기서 S1은 스위치이기 때문에 CNFy[1:0] (Configuration) 부분의 Input mode이며, Input with pull-up / pull-down 인 10으로 넣어줘야 하므로, GPIO_CRH_CNF11_1을 넣어줘야 한다.

물론, 초기화 부분은, _0과 _1이 둘 다 꺼져야 하므로, GPIO_CRH_CNF11와 GPIO_CRH_MODE11가 (11)이므로 ~(11)하면 (00)이므로, 아무런 입력이 없는 상태로 만들어줘야 한다.

```
#define  GPIO_CRH_CNF11                ((uint32_t)0x0000C000)      /*!< CNF11[1:0] bits (Port x configuration bits, pin 11) */
#define  GPIO_CRH_CNF11_0              ((uint32_t)0x00004000)      /*!< Bit 0 */
#define  GPIO_CRH_CNF11_1              ((uint32_t)0x00008000)      /*!< Bit 1 */
#define  GPIO_CRH_MODE11               ((uint32_t)0x00003000)      /*!< MODE11[1:0] bits (Port x mode bits, pin 11) */
#define  GPIO_CRH_MODE11_0             ((uint32_t)0x00001000)      /*!< Bit 0 */
#define  GPIO_CRH_MODE11_1             ((uint32_t)0x00002000)      /*!< Bit 1 */
```

따라서, 아래와 같은 코드로 표현할 수 있다.

```
/* Reset(Clear) Port D CRH - User S1 Button */
// GPIOD->CRH &= ??
GPIOD->CRH &= ~(GPIO_CRH_CNF11 | GPIO_CRH_MODE11);
/* User S1 Button Configuration */
// GPIOD->CRH |= ??
GPIOD->CRH |= GPIO_CRH_CNF11_1;
```

우리는, 코드에서 A Port를 UART Tx, Rx, MCO를 위해 사용하고, D Port를 S1 Button을 위해 사용한다. 그러기 때문에, 아래의 코드와 같이 RCC->APB2ENR로 GPIOA (Port A)와 GPIOD (Port D)를 enable해주었다. 마찬가지로 USART RCC도 enable해준다.

```
void RCC_Enable(void) {
//@TODO - 3 RCC Setting
    /*---------------------------- RCC Configuration ----
    /* GPIO RCC Enable  */
    /* UART Tx, Rx, MCO port */
    // RCC->APB2ENR |= ??
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
    /* USART RCC Enable */
    // RCC->APB2ENR |= ??
    RCC->APB2ENR |= RCC_APB2ENR_USART1EN;
    /* User S1 Button RCC Enable */
    // RCC->APB2ENR |= ??
    RCC->APB2ENR |= RCC_APB2ENR_IOPDEN;
}
```

Port Configuration 부분인데, MCO가 PA8이고, USART Tx가 PA9, USART Rx가 PA10이기 때문에, 포트 번호에 맞춰 configuration해주었고, 주어진 대로 MCO 및 UART Tx는 Alternate function output Push-pull, UART Rx는 Input with pull-up / pull-down에 맞게 설정했다. 물론, 위에서와 같이 초기화도 not (11)처럼 초기화 해준다.

```
void PortConfiguration(void) {
//@TODO - 4 GPIO Configuration
    /* Reset(Clear) Port A CRH - MCO, USART1 TX,RX*/
    GPIOA->CRH &= ~(
            (GPIO_CRH_CNF8 | GPIO_CRH_MODE8) |
            (GPIO_CRH_CNF9 | GPIO_CRH_MODE9) |
            (GPIO_CRH_CNF10 | GPIO_CRH_MODE10)
        );
    /* MCO Pin Configuration */
    GPIOA->CRH |= (GPIO_CRH_CNF8_1 | GPIO_CRH_MODE8_0);
    //GPIOA->CRH |= 0x000000BB; // MCO, USART TX Pin Configuration
    /* USART Pin Configuration */
    // GPIOA->CRH |= ??
    GPIOA->CRH |= (GPIO_CRH_CNF9_1 | GPIO_CRH_MODE9_0 |
                   GPIO_CRH_CNF10_1 | GPIO_CRH_MODE10_0);
    /* Reset(Clear) Port D CRH - User S1 Button */
    // GPIOD->CRH &= ??
    GPIOD->CRH &= ~(GPIO_CRH_CNF11 | GPIO_CRH_MODE11);
    /* User S1 Button Configuration */
    // GPIOD->CRH |= ??
    GPIOD->CRH |= GPIO_CRH_CNF11_1;
}
```

27.6.4 Control register 1 (USART_CR1)

Address offset: 0x0C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

위의 표를 참조하여, 아래와 같이 M, PCE, PS, TE, RE를 Clear해주는 코드와, Tx와 Rx를 Enable해주도록 USART의 Control Register로 transmitter, receiver, USART를 enable 시켜주는 코드를 작성할 수 있다.

```
void UartInit(void) {
    /*---------------------------- USART CR1 Configuration -----------------------*/
    /* Clear M, PCE, PS, TE and RE bits */
    USART1->CR1 &= ~(uint32_t)(USART_CR1_M | USART_CR1_PCE | USART_CR1_PS | USART_CR1_TE | USART_CR1_RE);
    /* Configure the USART Word Length, Parity and mode ----------------------- */
    /* Set the M bits according to USART_WordLength value */
//@TODO - 6: WordLength : 8bit

    /* Set PCE and PS bits according to USART_Parity value */
//@TODO - 7: Parity : None

    /* Set TE and RE bits according to USART_Mode value */
//@TODO - 8: Enable Tx and Rx
    // USART1->CR1 |=
    USART1->CR1 |= USART_CR1_TE | USART_CR1_RE;
```

그리고, Baud Rate를 구하는 부분인데, 빨간 네모 안의 수식들을 이용하여 USARTDIV를 구해야 한다.

f_CK 가 Input clock으로, PCLK2 for USART1이기 때문에 f_CK는 26MHz이다. (26,000,000Hz)

- ➔ 16 * USARTDIV = f_CK / Baud Rate
- ➔ USARTDIV = 26,000,000 / 14,400 / 16 = 112.847222….
- ➔ 112 = 0x70
- ➔ 0.84722 * 16 = 13.55555… = 약 14 (가장 가까운 실수) = 0xE
- ➔ USARTDIV = **0x70E**

$$\text{Tx/ Rx baud} = \frac{f_{CK}}{(16*\text{USARTDIV})}$$

legend: f_{CK} - Input clock to the peripheral (PCLK1 for USART2, 3, 4, 5 or PCLK2 for USART1)

Example 2:
To program USARTDIV = 0d25.62
This leads to:
DIV_Fraction = 16*0d0.62 = 0d9.92
The nearest real number is 0d10 = 0xA
DIV_Mantissa = mantissa (0d25.620) = 0d25 = 0x19
Then, USART_BRR = 0x19A hence USARTDIV = 0d25.625

SYSCLK	52MHz
PCLK2	26MHz
Baud Rate	14400

```
    /*---------------------------- USART BRR Configuration -----------------------*/
    /* Configure the USART Baud Rate -----------------------------------------------*/
    /* Determine the integer part */
    /* Determine the fractional part */
//@TODO - 11: Calculate & configure BRR
    // USART1->BRR |= ??
    USART1->BRR |= 0x70E;
```

그리고, 마지막으로 "Hello Team08" 문자열을 Putty로 출력해주기 위한 문자 배열 선언과, S1버튼을 누르고 있는 동안에 반복적으로 출력되도록 하고, 한 번 클릭했을 때, 하나만 출력되도록 출력 후, 딜레이를 주었다.

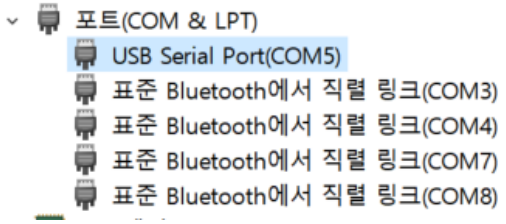
```
int main() {
        int i;
        char msg[] = "Hello Team08\r\n";

    SysInit();
    SetSysClock();
    RCC_Enable();
    PortConfiguration();
    UartInit();

    // if you need, init pin values here


    while (1) {
      if (!(GPIOD->IDR & (GPIO_IDR_IDR11))) {

        for (i = 0; i < strlen(msg); i++) {
          SendData(msg[i]);

        }
        delay();
      }
    }
}// end main
```

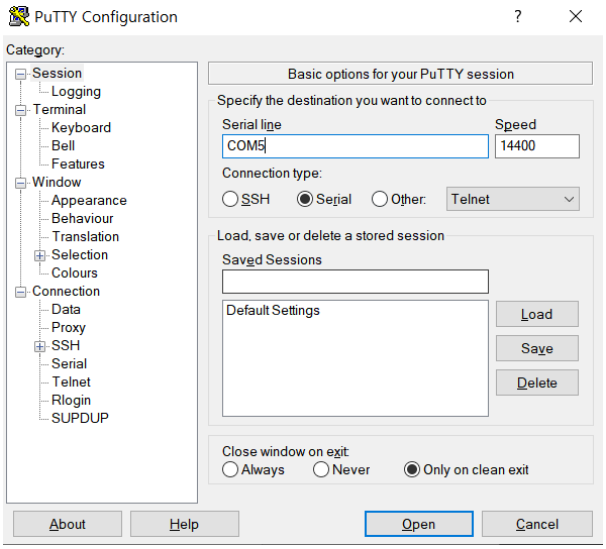
결 론

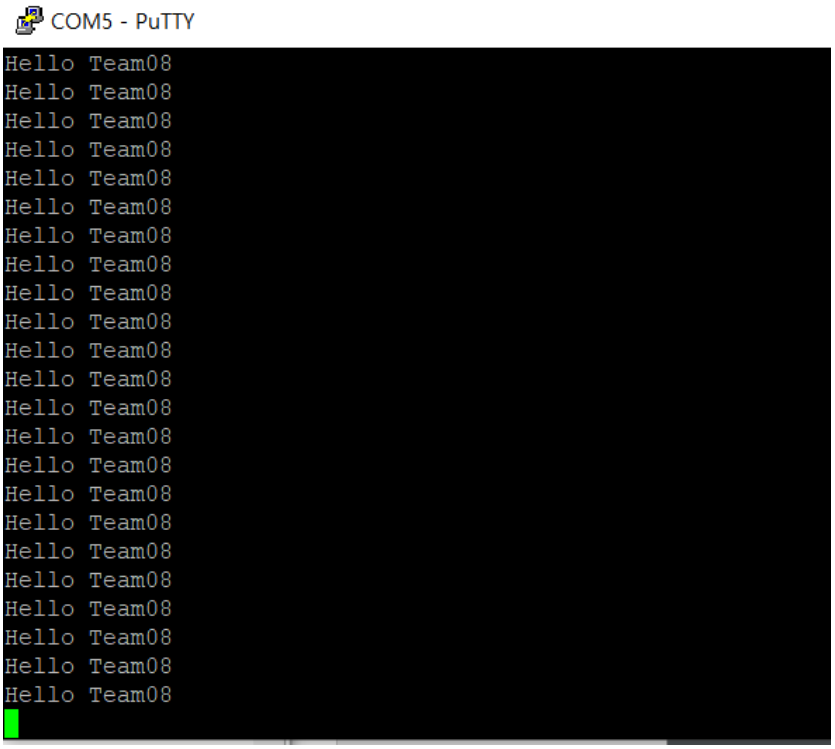
자 그럼 이제, User S1 버튼을 누르는 동안 터미널 프로그램(Putty)을 통해 "Hello Team08"을 출력 후 줄 바꿈 작업을 해보도록 하자.

우선, STM32를 컴퓨터와 연결한 포트번호를 알아야한다. 장치 관리자를 통해 USB Serial Port를 확인하면 쉽게 알 수 있으며, Putty Configuration을 할 때, 해당 시리얼 라인(COM5)를 입력해주고, Speed는 Baud Rate를 동일하게 14400으로 맞춰주고 Open을 누르면 검정색 화면이 나타난다.

그런 뒤, S1 버튼을 누르면 아래의 사진과 같이 Hello Team08이 반복적으로 출력이 된다.

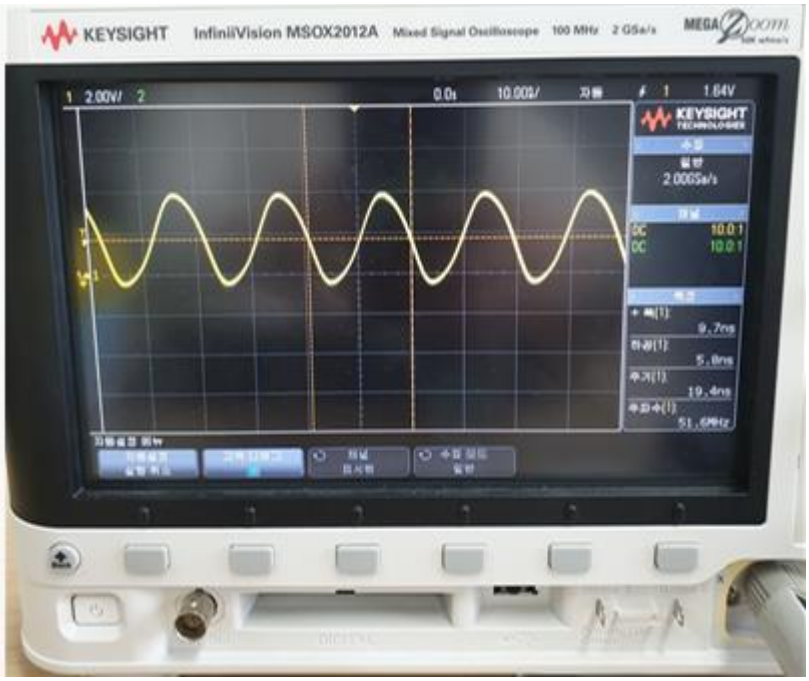






마지막으로, MCO를 통해 나오는 System Clock을 오실로스코프로 수치를 확인해보았다.

(이때, GND와 Port로 연결해주면 되는데, MCO가 PA8이므로, PA8과 GND(ground)에 각각 꽂으면 아래의 사진과 같은 결과를 확인할 수 있다.)



느낀 점

이번 실험을 통해 컴퓨터 구조 시간에 배웠던 클락의 작동과 값을 넘기는 방법을 눈으로 직접 보고 확인할 수 있게 되었고, 처음에 clock tree가 뭔지도 몰랐는데, 스키메틱과 reference, Datasheet 내용을 차근차근 훑어보니 이해할 수 있었다. 특히, Baud rate의 용도와 값을 계산하는 법도 직접 계산해보고 왜 그렇게 되는지 알고나니 굉장히 뿌듯하고 도움이 되었다.