2023-2 임베디드 시스템 설계 및 실험
6 주차 보고서

수요일(003) 분반 4 조

개요 라이브러리를 이용해 코드를 작성하고 Clock tree 구조를 이해한다.

오실로스코프를 이용해 Clock 을 확인하고 Clock 을 이용해 시리얼 통신을 한다.

목표

1. 라이브러리를 활용하여 코드 작성
2. Clock Tree 의 이해 및 사용자 Clock 설정
3. 오실로스코프를 이용한 Clock 확인
4. UART 통신의 원리를 배우고 실제 설정 방법 파악

세부 실험 내용

1. DataSheet 및 Reference Manual 을 참고하여 해당 레지스터 및 주소에 대한 설정 이해
2. 예제 코드에서 설정되는 Clock 값을 파악하고, 지정된 Clock 으로 설정
3. 예제 설정 항목에 따라 UART 를 설정하고, 지정된 Baud rate 로 설정

   SYSCLK: 28MHz

   PCLK2: 14MHz

   Baud Rate: 28800
4. User S1 버튼을 누르는 동안 터미널 프로그램(Putty)을 통해 "Hello Team04"을 출력 후 줄 바꿈
5. MCO 를 통해 나오는 System Clock 을 오실로스코프로 수치 확인

실험 과정

1. SysInit 함수로 마이크로 컨트롤러 시스템 클록과 여러 시스템 제어 레지스터 초기화.

2. PC 와 통신

   (1) void SetSysClock(void) 내부 일부분을 고쳐서 목적 Clock 을 생성한다.

```
//@TODO - 1 Set the clock, (///) 주석 표시를 없애고 틀린 값이 있다면 제대로 된 값으로 수정하시오
        /* HCLK = SYSCLK */
        RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;
        /* PCLK2 = HCLK / ?, use PPRE2 */
        /* 분주비를 2로 하여 PCLK2 (APB2 클럭)를 HCLK의 절반이 되도록 수정. */
        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);
        /* 4MHz * PLL(7) = 28MHz */
        RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLXTPRE_PREDIV1 | RCC_CFGR_PLLSRC_PREDIV1 | RCC_CFGR_PLLMULL7);

        RCC->CFGR2 &= (uint32_t)~(RCC_CFGR2_PREDIV2 | RCC_CFGR2_PLL2MUL | RCC_CFGR2_PREDIV1 | RCC_CFGR2_PREDIV1SRC);
        /* HSE(25MHz) / PREDIV2(5) * PLL2(8) / PREDIV1(10) = 4MHz */
        RCC->CFGR2 |= (uint32_t)(RCC_CFGR2_PREDIV2_DIV5 | RCC_CFGR2_PLL2MUL8 | RCC_CFGR2_PREDIV1SRC_PLL2 | RCC_CFGR2_PREDIV1_DIV10);
//@End of TODO - 1
```

그림 1. Clock set 을 위한 코드

```
/* 분주비를 2로 하여 PCLK2 (APB2 클럭)를 HCLK의 절반이 되도록 수정. */
RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE2_DIV2;
```

PCLK2 를 조건에 따르면 SYSCLK 의 절반이 되어야 하므로 RCC_CFGR_PPRE2_DIV2 로 절반이 되도록 만들었다.

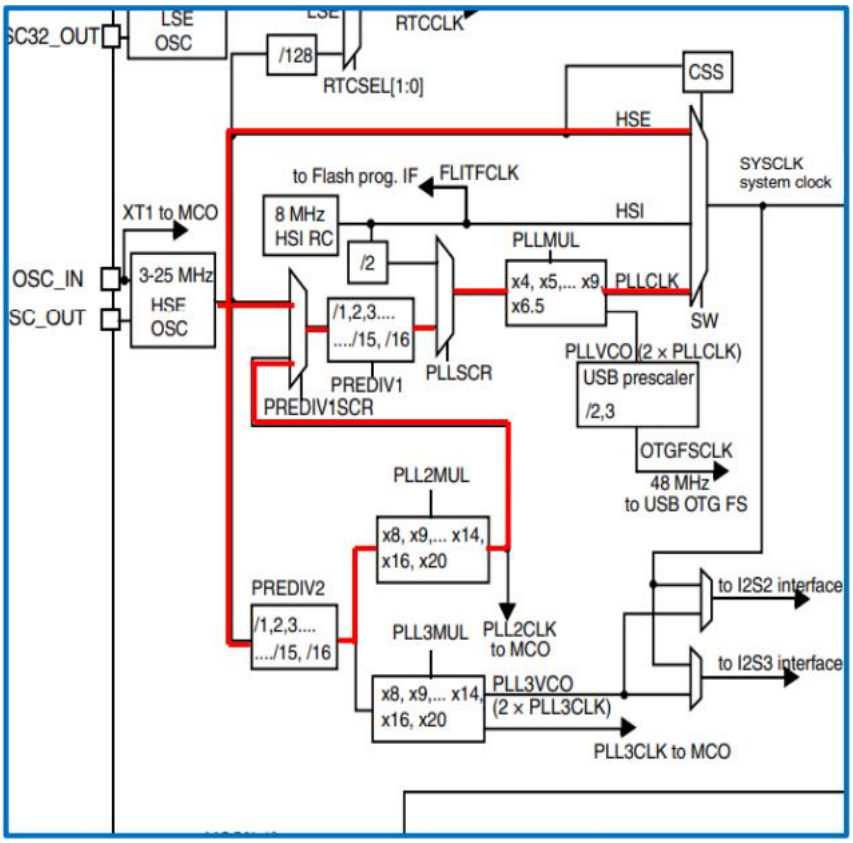

그림 2. Clock Tree 에서 HSE 흐름도

그림 2를 보면 HSE에서 신호가 나오고 PREDIV2를 거쳐 PLL2MUL,PREDIV1SCR,PREDIV1,PLLSCR,PLLMULL,PLLCLK을 지나가는 것을 알 수 있다. 따라서 HSE에서 25MHz의 신호가 나오면 PREDIV2에서 5로 나눈 뒤, PLL2MUL에서 8로 곱한뒤, PREDIV1SCR에서 선택되어 PREDIV1에서 1로 나눈 뒤, PLLSCR을 거쳐 PLLMUL에서 7을 곱해 28MHz를 출력한다. 그러기 위해 CFGR과 CFGR2 레지스터를 설정해 주었다.

이 과정을 통해 PLLCLK은 28MHz의 clock이 흐른다.TODO에는 없지만 해당 코드에 의해서 PLLCLK이 SYSCLK으로 지정된다.

```
        /* Select PLL as system clock source */
        RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
        RCC->CFGR |= (uint32_t)RCC_CFGR_SW_PLL;
        /* Wait till PLL is used as system clock source */
        while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS) != (uint32_t)0x08)
        {
        }
        /* Select System Clock as output of MCO */
//@TODO - 2 Set the MCO port for system clock output
        /* MCO 설정 초기화 */
        RCC->CFGR &= ~(uint32_t)RCC_CFGR_MCO;
        /* MCO 포트가 SYSCLK(시스템 클럭) 출력하도록 설정 */
        RCC->CFGR |= RCC_CFGR_MCO_SYSCLK;
//@End of TODO - 2
```

그림 3. PLL, MCO 설정코드

따라서 CFGR 에 RCC_CFGR_MCO_SYSCLK 을 입력해 주면 28MHZ 의 Clock 이 MCO 로 지정된다

(2) GPIOA 포트, USART1 클럭, GPIOC 포트 활성화

```
void RCC_Enable(void) {
//@TODO - 3 RCC Setting
    /*---------------------------- RCC Configuration -----------------------------*/
    /* GPIO RCC Enable  */
    /* UART Tx, Rx, MCO port */
    /* UART Tx, Rx, MCO 포트가 위치한 GPIOA 포트 클럭 활성화 */
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
    /* USART RCC Enable */
    /* UART 통신을 위해 USART1 클럭 활성화 */
    RCC->APB2ENR |= RCC_APB2ENR_USART1EN;
    /* User S1 버튼이 위치한 GPIOC 포트 클럭 활성화 */
    RCC->APB2ENR |= (uint32_t)RCC_APB2ENR_IOPCEN;
}
```

그림 4. 클럭, 포트 활성화 코드

RCC_APB2ENR 레지스터를 수정하여 GPIOA , GPIOC포트 클럭을 활성화하고, USART1에 Clock을 인가해준다.

(3) GPIO 포트설정

```
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 */
    /* MCO(PA8) 핀을 Alternative Function push-pull 모드, 출력속도 최대속도로 설정 */
    GPIOA->CRH |= (GPIO_CRH_CNF8_1 | GPIO_CRH_MODE8);
    /* USART Pin Configuration */
    /* TX(PA9) 핀을 Alternative Function push-pull 모드, 출력속도 최대속도로 설정 */
    /* RX(PA10) 핀을 Alternate Function floating input 모드, 입력 모드로 설정 */
    GPIOA->CRH |=  ((GPIO_CRH_CNF9_1 | GPIO_CRH_MODE9) | (GPIO_CRH_CNF10_1));

    /* Reset(Clear) Port C CRL - User S1 Button */
    /* User S1 버튼(PC4) 핀 설정 초기화 */
    GPIOC->CRL &= ~(GPIO_CRL_CNF4 | GPIO_CRL_MODE4);
    /* User S1 Button Configuration */
    /* User S1 버튼(PC4) 핀을 input floating mode로 설정 */
    GPIOC->CRL |= GPIO_CRL_CNF4_1;
}
```

그림 5. GPIO 포트 설정 코드

GPIO 포트들의 속도, 모드들을 설정 해준다.

(4) Putty를 이용해서 port값, Baud rate값을 설정해준다.

Fractional baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the Mantissa and Fraction values of USARTDIV.

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

그림 6. Baud rate 계산식

PCLK2: 14MHz

Baud Rate: 28800

주어진 조건에 따라 USARTDIV= 30.3819 가 나온다.

Mantisa 값이 30 이므로 이를 16 진수로 고치면 0x1E 이 나오고 Fraction 부분이 0.3819 이므로 0.3819*16=6=0x6 이므로 BRR=0x1E6 이된다.

```
//@TODO - 11: Calculate & configure BRR
    USART1->BRR |= 0x1E6;
    /* 14,000,000 / (28800 * 16) = 30.3819 */
    /* Mantisa : 30 = 0x1E */
    /* Fraction : 0.3819 * 16 = 6 = 0x6 */
    /* BRR = 0x1E6 */

    /*---------------------------- USART Enable ----------------------------------*/
```

(5) 버튼을 누르면 “Hello Team04”가 나올수 있도록 delay 함수, SendData 함수, main 함수를 작성

```
void delay(void){
    int i = 0;
    for(i=0;i<1000000;i++);
}

void SendData(uint16_t data) {
    /* Transmit Data */
   USART1->DR = data;

   /* Wait till TC is set */
   while ((USART1->SR & USART_SR_TC) == 0);
}

int main() {
   int i;
   char msg[] = "Hello Team04\r\n";

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

    // if you need, init pin values here


    while (1) {
      //@TODO - 13: Send the message when button is pressed
      if(!(GPIOC -> IDR & GPIO_IDR_IDR4)){
            for(i = 0; i < 14; i++) {
                SendData(msg[i]);
            }
        }
   }
```

```
}// end main
```

3. 오실로스코프 아날로그 측정

(1) 신호를 Analog로 받기 위해 설정한다.

(2) Meas 버튼을 눌러 측정을 시작한다.

(3) frequency 모드로 변경한다.

(4) 화면에 frequency를 출력한다.

(5) Auto scale 버튼을 누르면 그래프가 정돈된다.

실험 결과

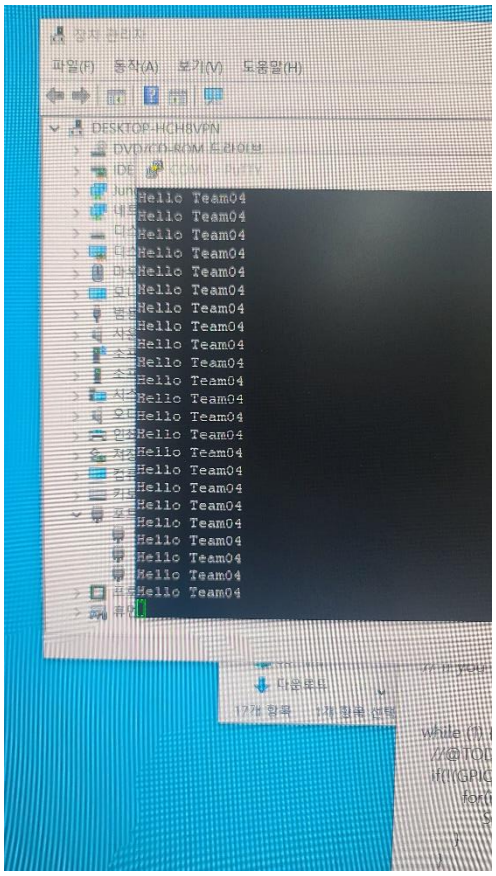

그림 6. Putty 실험 결과화면

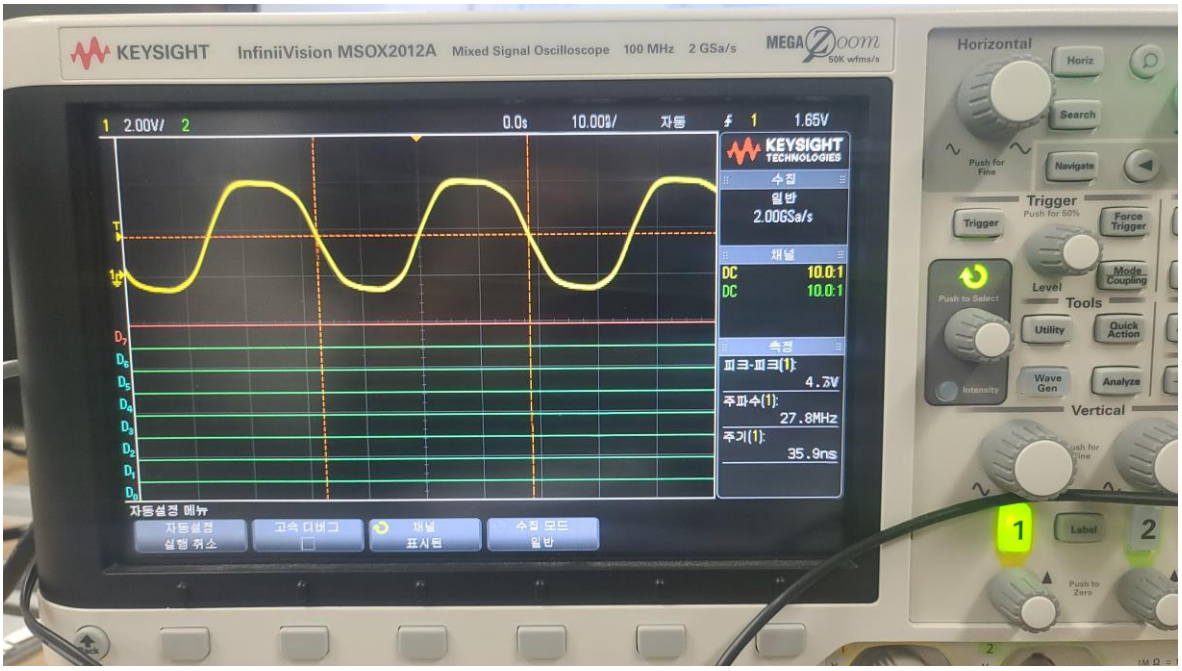

그림 7. 오실로스코프 결과화면

Cortex-m3 stm32f 보드와 PC를 연결하여 'hello world'를 putty 화면에 출력하였다. 오실로스코프와 보드를 연결하여 시스템 Clock 주파수가 28MHz로 설정된 걸 확인 할 수 있었다.