6 주차 실험 결과 보고서

2 조 (이지수, 박은재, 유승훈, 고상현, 강민수)

1. 실험 목표

- Clock Tree 의 이해 및 사용자 Clock 설정
- UART 통신의 원리 이해 및 실제 설정 방법 파악

2. 이론

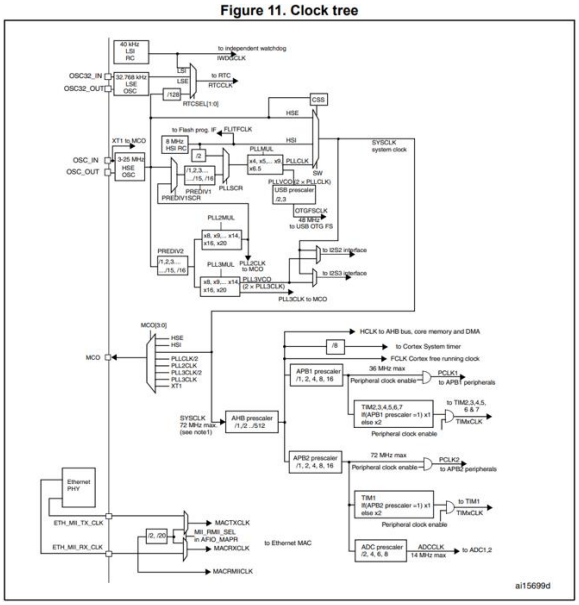
- Clock

HSI, HSE 라는 클럭 소스가 존재하며, 이를 나누고 곱하는 조작의 과정을 거쳐 PLLCLK 라는 신호를 만들어낼 수 있습니다.

최종적으로 셋 중 하나를 골라 System Clock 을 결정하게 됩니다.

아래의 Clock Tree 를 통해 구조를 확인할 수 있습니다

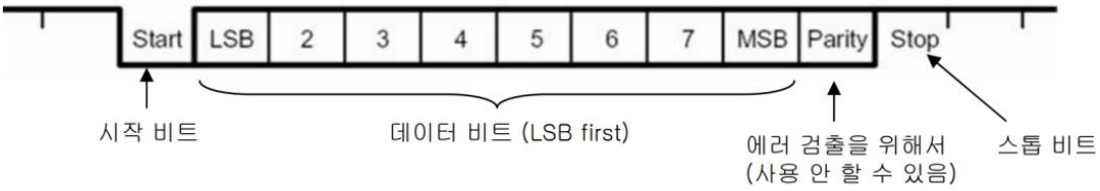
Figure 11. Clock tree



이번 실험에서 저희 조는 52Mhz 의 System Clock, 26Mhz 의 PLCK2 를 만들었습니다

- UART 통신

통신 프로토콜의 일종입니다. Rx, Tx 간의 교차 연결을 통해 이루어지며, 비동기 통신이기 때문에 Baud Rate 를 일치시켜야 합니다.



전송하는 데이터의 구조는 위의 그림과 같이 통신이 시작됨을 알리는 시작 비트, 실제 데이터가 전송되는 데이터 비트, 에러 검출을 위한 패리티 비트, 그리고 통신이 끝났음을 알리는 스탑 비트로 이루어집니다.

이번 실험에서 저희 조의 Baud Rate 는 9600 이었습니다.

3. 실험 내용 및 결과

3-1. 코드 작성

코드에는 5 개의 주요 함수가 정의되어 있습니다.

1) void SysInit(void);

```
RCC->CR |= (uint32_t)0x00000001; // Reset HSION
RCC->CFGR &= (uint32_t)0xF0FF0000; // Reset SW, HPRE, PPRE1, PPRE2, ADCPRE, MCO
RCC->CR &= (uint32_t)0xFEF6FFFF; // Reset HSEON, CSSON, PLLON
RCC->CR &= (uint32_t)0xFFFBFFFF; // Reset HSEBYP
RCC->CFGR &= (uint32_t)0xFF80FFFF; // Reset PLLSRC, PLLXTPRE, PLLMUL, USBPRE/OTGFSPRE
RCC->CR &= (uint32_t)0xEBFFFFFF; // Reset PLL2ON, PLL3ON
RCC->CIR = 0x00FF0000; // Disable all interrupts and clear pending bits
RCC->CFGR2 = 0x00000000; // Reset CFGR2 register
```

2) void SetSysClock(void);

```
//Set the Clock
        /* HCLK = SYSCLK */
        RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;
        /* PCLK2 = HCLK / 2, use PPRE2 */
        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_PLLSRC | RCC_CFGR_PLLMULL);
```

```
        RCC->CFGR |= (uint32_t)(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);

//Set MCO port for system clock output
        RCC->CFGR &= ~(uint32_t)RCC_CFGR_MCO;
        RCC->CFGR |= RCC_CFGR_MCO_SYSCLK;
```

3) void RCC_Enable(void);

```
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN; // port A clock enable
    RCC->APB2ENR |= RCC_APB2ENR_USART1EN; // USART1 clock enable
    RCC->APB2ENR |= (uint32_t)RCC_APB2ENR_IOPDEN; // port D clock enable
```

4) void PortConfiguration(void);

```
    GPIOA->CRH &= ~(
        (GPIO_CRH_CNF8 | GPIO_CRH_MODE8) |
        (GPIO_CRH_CNF9 | GPIO_CRH_MODE9) |
        (GPIO_CRH_CNF10 | GPIO_CRH_MODE10)
    ); // Reset Port A CRH - MCO, USART1 TX,RX

    GPIOA->CRH |= (GPIO_CRH_CNF8_1 | GPIO_CRH_MODE8_1); // MCO Pin Configuration
    GPIOA->CRH |=   ((GPIO_CRH_CNF9_1 | GPIO_CRH_MODE9_1) | (GPIO_CRH_CNF10_1 |
GPIO_CRH_MODE10_1));
    // USART Pin Configuration

    GPIOC->CRL &= ~(GPIO_CRL_CNF4 | GPIO_CRL_MODE4); // Reset Port C CRL
    GPIOC->CRL |= GPIO_CRL_CNF4_1; // Button Configuration
```

5) void UartInit(void);

```
/*---------------------------- USART CR1 Configuration -----------------------*/
    USART1->CR1 &= ~(uint32_t)(USART_CR1_M | USART_CR1_PCE | USART_CR1_PS | USART_CR1_TE |
USART_CR1_RE); // Clear M, PCE, PS, TE and RE bits
    USART1->CR1 |= (uint32_t)(USART_CR1_M); // Word Length
    USART1->CR1 |= ~(uint32_t)(USART_CR1_PCE); // Parity
    USART1->CR1 |= (uint32_t)(USART_CR1_TE | USART_CR1_RE) // Enable Tx, Rx

    /*---------------------------- USART CR2 Configuration -----------------------*/
    USART1->CR2 &= ~(uint32_t)(USART_CR2_STOP); // Clear STOP[13:12] bits
```

```
    USART1->CR2 &= ~(uint32_t)(USART_CR2_CPHA | USART_CR2_CPOL | USART_CR2_CLKEN); // Configure
the USART Stop Bits, Clock, CPOL, CPHA and LastBit
    USART1->CR2 |= (uint32_t)(0x0000); // Set Stop bit : 1 bit

    /*---------------------------- USART CR3 Configuration -----------------------*/
    USART1->CR3 &= ~(uint32_t)(USART_CR3_CTSE | USART_CR3_RTSE); // Clear CTSE and RTSE bits
    USART1->CR3 |= ~(uint32_t)(USART_CR3_CTSE | USART_CR3_RTSE); // Disable CTS, RTS

    USART1->BRR |= 0xA94; // Configure BRR
    USART1->CR1 |= (uint32_t)(USART_CR1_UE); // USART Enable
```

main 함수 내부의 흐름에 따라 정리하자면,
1 번 함수로 Clock 신호 조작을 위해 필요한 레지스터들을 설정하고,
2 번 함수로 Clock 신호를 조작하여 원하는 System clock 을 만들어 냅니다.
이까지가 Clock 신호 실험 내용에 해당합니다.

이후, 3, 4 번 함수로 UART 통신을 위해 필요한 레지스터들을 설정하고,
5 번 함수로 UART 통신 관련 설정을 조작합니다.

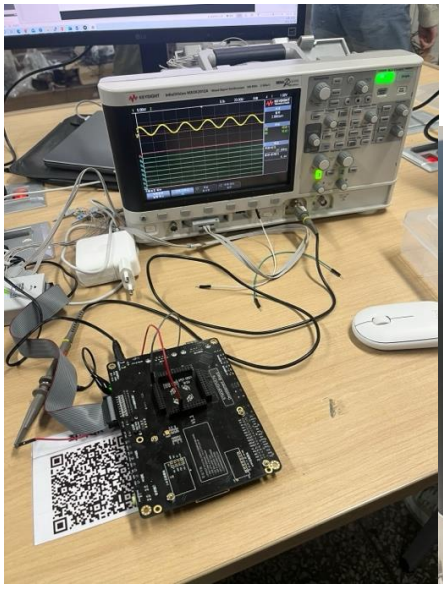
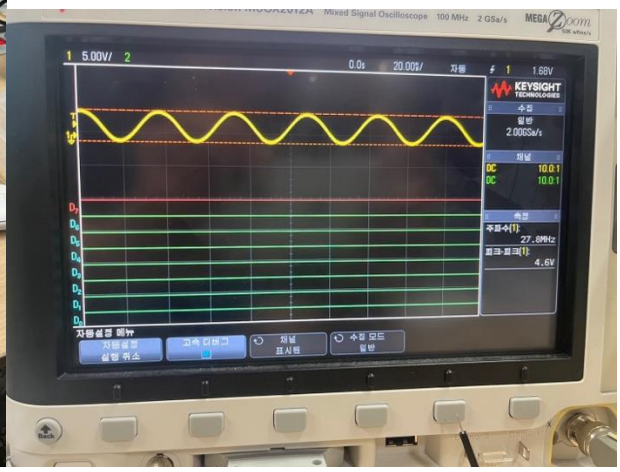
```
    char msg[] = "Hello Team2\r\n";

    while (1) {
      if(!(GPIOD -> IDR & GPIO_IDR_IDR11))
        for(i = 0; i < 14; i++) SendData(msg[i]);
    }
```

그 결과, 위의 코드로 UART 통신이 이루어집니다.

3-2. 오실로스코프를 통한 Clock 신호 확인

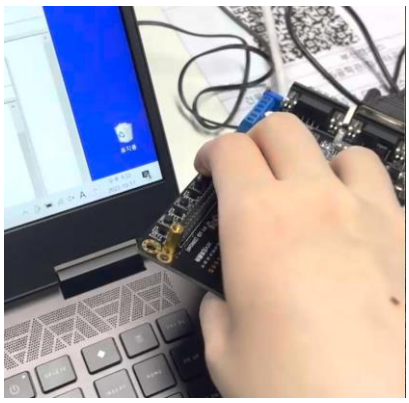
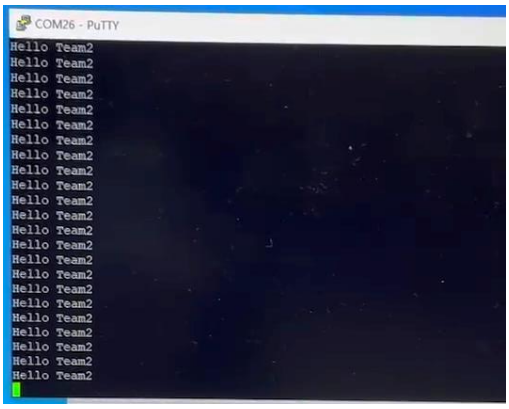
Clock 신호 설정이 제대로 되었는지 확인하기 위해 보드와 오실로스코프를 연결하여 Clock 신호 출력을 확인하였습니다.

위와 같이 오실로스코프에 연결 후 측정하자 System Clock 값이 측정되었습니다.

3-3. putty 를 통한 UART 통신 결과 확인

UART 통신이 제대로 이루어졌는지 확인하기 위해 putty 를 통해 통신 결과를 확인하였습니다.

위의 그림과 같이 버튼을 누르면 "Hello Team2"라는 문자열이 출력되는 것을 확인할 수 있었습니다.