**6주차 결과 보고서**

9조 팀장 김주송

팀원 강선호

박형주

손봉국

1. **실험 목적**
   1. **라이브러리를 활용하여 코드 작성**
   2. **Clock Tree의 이해 및 사용자 Clock 설정**
   3. **오실로스코프를 이용한 clock 확인**
   4. **UART 통신의 원리를 배우고 실제 설정 방법 파악**
2. **배경 지식**
3. HIS(High Speed Internal)  
   STM32에 내장되어 있는 clock으로 8MHz RC 오실레이터에서 생성된다. 생성된 clock은 시스템 클럭으로 사용하거나, PLL clock으로 사용 가능하다. 오차 발생확률이 높은 방식이다.
4. HSE(High Speed External)  
   STM32 외부에서 입력되는 높은 주파수 clock으로 HSE OSC에서 25 MHz clock을 생성한다. 생성된 clock은 바로 system clock으로 사용하거나, PLL clock으로 사용할 수 있다.
5. Clock Tree  
   STM32 내부의 Clock 흐름을 보여준다. HIS, HSE, PLL중 SW MUX에 의해 system clock(최대 72MHz)으로 설정될 수 있다. 시스템 클럭은 APB1, APB2에 전달된다. 시스템 클럭은 MCO MUX를 통해서 MCO에 출력해 오실로스코프로 확인이 가능하다.  
   System Clock을 prescaler를 이용하여 APB1에는 최대 36MHz를 PCLK1에, APB2에는 최대 72MHz를 PCLK2에 제공 가능하다.
   1. FCLK  
      Cortex System(CPU)에서 사용되는 clock
   2. HCLK  
      DMA, Core memory, AHB Bus에서 사용되는 clock
   3. PCLK  
      APB Bus에 사용되는 clock
6. PLL(Phase-Locked Loop)  
   PLL은 위상 동기 회로이며, 입력 신호와 출력신호를 이용해 출력신호를 제어하는 시스템을 말한다. 입력된 신호에 맞추어 출력 신호의 주파수 조절이 목적이다.
7. MCO  
   Clock을 외부에서 측정할 수 있게 하는 기능을 제공한다. MCO핀을 통해 MCO Clock Source를 출력한다.
8. Serial Communication(직렬 통신)  
   하나의 데이터 선을 이용해 비트를 차례로 보내는 방법이다. 속도는 느리나, 데이터 선을 연장하기 위한 비용이 적다.
9. USART(Universal Sync/Async Receiver/Transmitter)  
   동기식 통신을 지원하는 UART이다. 데이터 동기화를 위해 같은 클럭을 사용하고 클럭 전송을 위한 별도의 클럭 선이 필요하다. Start bits/Stop bits가 없고 Data bits를 클럭에 동기화해 전송한다.
10. UART(Universal Asynchronous Receiver/Transmitter)  
    비동기 통신 프로토콜로 Rx(수신)와 Tx(송신)을 교차 연결한다. 비동기 통신이기 때문에 Baud Rate를 일치시켜야 한다.
11. Baud Rate  
    초당 얼마나 많은 심볼(의미 있는 데이터 묶음)을 전송할 수 있는가를 정한 비율
12. Start bit  
    통신의 시작을 의미하며 1bit 길이만큼 유지.
13. Data bit  
    8~9 bits의 데이터를 전송한다. Bit의 개수는 레지스터 설정에 따른다.
14. Parity bit  
    에러 검사를 위한 값. 수신 측에서 이 bit를 이용해 에러를 검사한다. 에러 발생 여부는 확인이 가능하나, 오류 수정은 불가하다. 수신 측에서 에러 발생 여부 확인 후, 데이터 재요청이 가능하다.
    1. Even Parity: 전송하고자 하는 데이터 + parity bit 중 1인 bit의 개수가 짝수
    2. Odd Parity: 전송하고자 하는 데이터 + parity bit 중 1인 bit의 개수가 홀수

Even, Odd 설정은 송수신 측이 미리 동기화되어 있어야 한다.

1. Stop bit: 통신의 종료를 알림. 레지스터에 따라 1, 1.5, 2bit으로 설정한다.
2. **실험 내용**

**지정된 설정**

|  |  |
| --- | --- |
| SYSCLK | 28MHz |
| PCLK2 | 14MHz |
| aud Rate | 28800 |

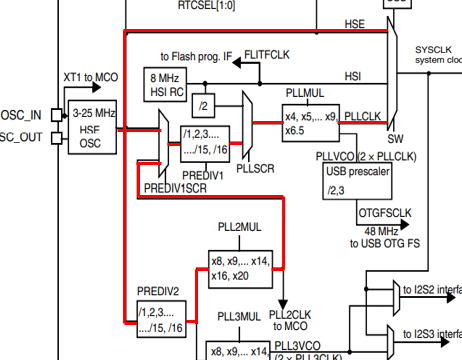
**코드 설명**

**-Todo 1텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

****

SYSCLK을 28MHz로 만들려고 HSE OSC에서 나오는 25MHz를 /10, \*8, /5, \*7의 루트를 타게 만들었다.

RCC->CFGR2 &= (uint32\_t)~(RCC\_CFGR2\_PREDIV2 | RCC\_CFGR2\_PLL2MUL | RCC\_CFGR2\_PREDIV1 | RCC\_CFGR2\_PREDIV1SRC);

RCC->CFGR &= (uint32\_t)~( RCC\_CFGR\_PLLSRC | RCC\_CFGR\_PLLMULL);

먼저 CFGR, CFGR2에서 PREDIV2, PLL2MUL, PREDIV1SCR, PREDIV1, PLLSCR, PLLMUL을 초기화한다.

RCC->CFGR2 |= (uint32\_t)(RCC\_CFGR2\_PREDIV2\_DIV10 | RCC\_CFGR2\_PLL2MUL8 | RCC\_CFGR2\_PREDIV1SRC\_PLL2 | RCC\_CFGR2\_PREDIV1\_DIV5);

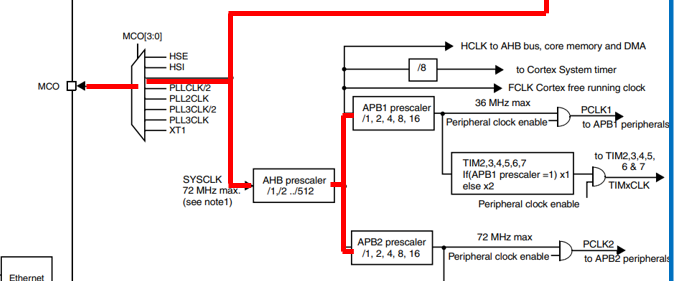
RCC->CFGR |= (uint32\_t)( RCC\_CFGR\_PLLSRC\_PREDIV1 | RCC\_CFGR\_PLLMULL7);

이후, PREDIV1SCR에서 RCC\_CFGR2\_PREDIV1SRC\_PLL2로 PLL2를 선택하게 하여 PREDIV2, PLL2MUL을 지날 수 있게 한다. PREDIV2는 RCC\_CFGR\_PREDIV2\_DIV10을 이용하고 PLL2MULL은 RCC\_CFGR2\_PLL2MUL8을 이용하여 /10, \*8을 수행한다. PLLSCR에서 RCC\_CFGR\_PLLSRC\_PREDIV1으로 PREDIV1을 선택하게 하고 RCC\_CFGR2\_PREDIV1\_DIV5로 PREDIV1에서 /5를 수행한다. RCC\_CFGR\_PLLMULL7으로 PLLMUL에서 \*7을 수행한다.

RCC->CFGR &= (uint32\_t)((uint32\_t)~(RCC\_CFGR\_SW));

RCC->CFGR |= (uint32\_t)RCC\_CFGR\_SW\_PLL;

마지막으로 주어진 코드에서 SW MUX를 최기화한 후, RCC\_CFGR\_SW\_PLL로 PLL을 선택하게 한다.



System clock으로 나온 28MHz는 PCLK2에서 14MHz로 나와야 한다.

RCC->CFGR |= (uint32\_t)RCC\_CFGR\_HPRE\_DIV1;

먼저 AHB prescaler에서 28MHz를 그대로 통과시킨다.

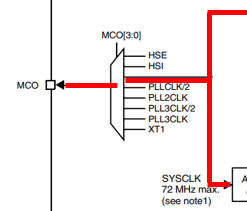
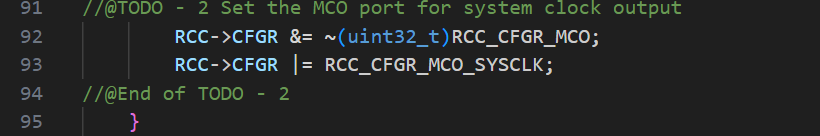
RCC->CFGR |= (uint32\_t)RCC\_CFGR\_PPRE2\_DIV2;

APB2 prescaler에서 /2를 통해 PCLK2에서 14MHz가 나오게 만든다.

RCC->CFGR |= (uint32\_t)RCC\_CFGR\_PPRE1\_DIV1;

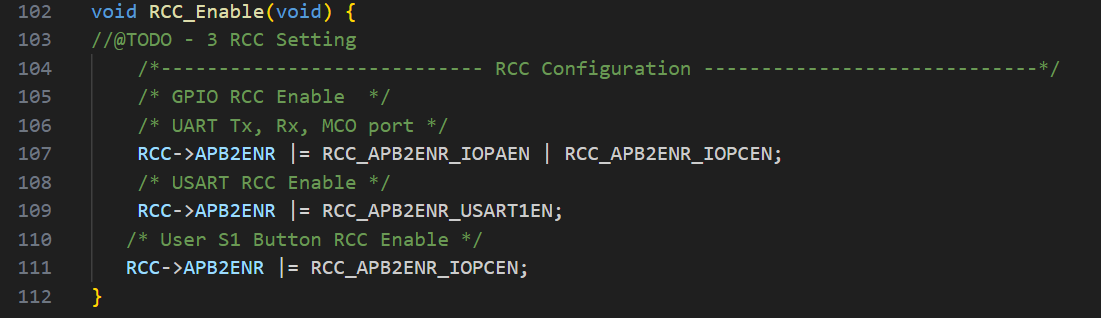
APB1 prescaler에서는 28MHz를 그대로 내보내어 PCLK1으로 나가게 한다.

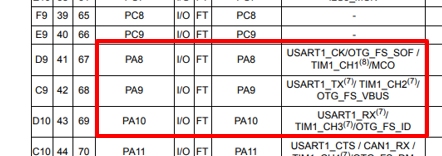
-Todo 2



MCO MUX에서 SYSCLK을 선택하도록 하여 MCO에서 28MHz가 나오게 한다.

-Todo 3



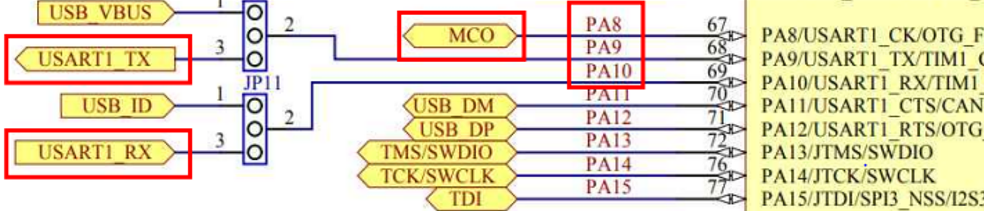
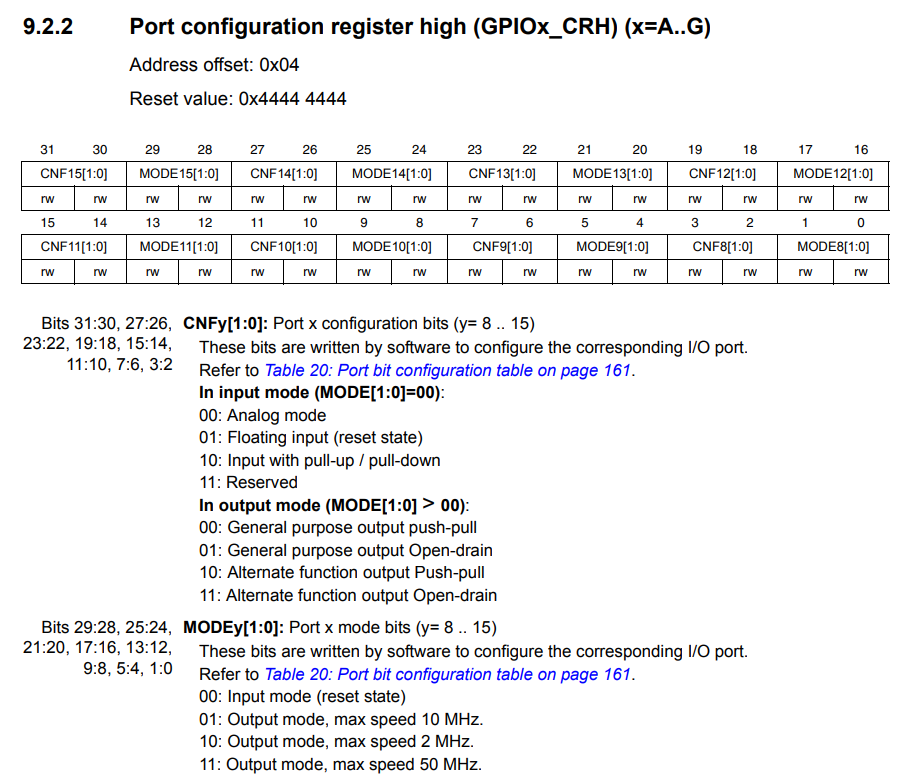


UART 통신과 버튼을 사용하기 위해 A, C포트에 Clock을 넣고, USART를 enable 시켜준다. USART1을 사용하기 위해서 RCC\_APB2ENR\_USART1EN을 이용한다.

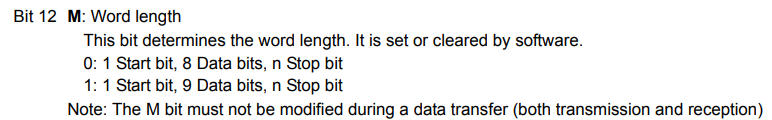
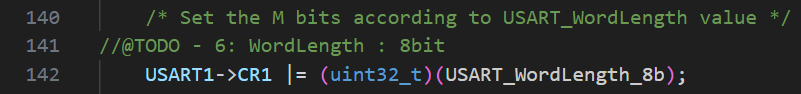
-Todo 4

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

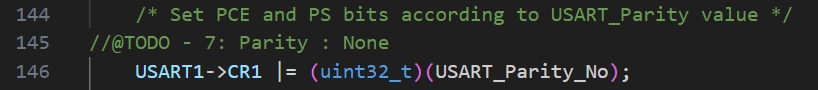
자동 생성된 설명

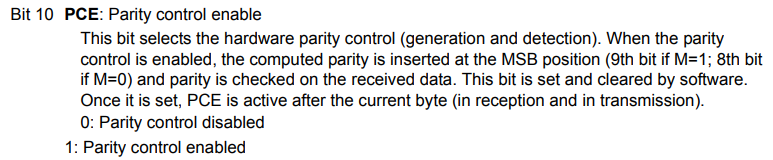


MCO, Tx, Rx를 설정하기 위해 PA8, 9, 10을 설정하고, 1번 버튼을 이용하기 위해서 PC4를 설정해준다. MCO, Tx는 output모드로, Rx와 버튼은 input pull-up/pull-down모드로 설정한다.

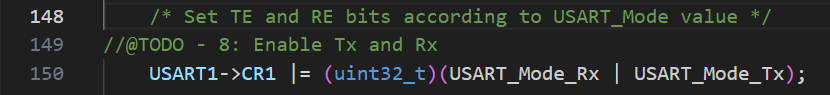
-Todo 6

Word length를 8bit로 선택한다.

-Todo 7



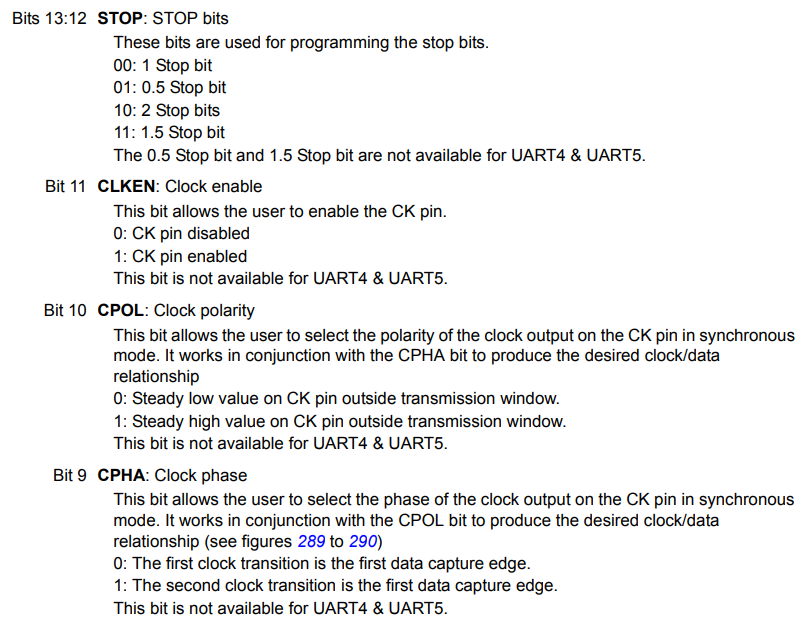
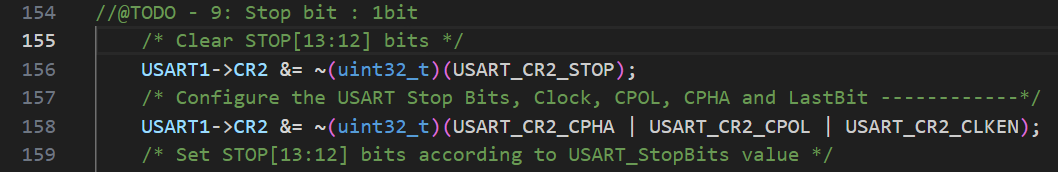
Parity bit을 사용하지 않도록 설정한다.

-Todo 8텍스트, 스크린샷, 폰트, 대수학이(가) 표시된 사진

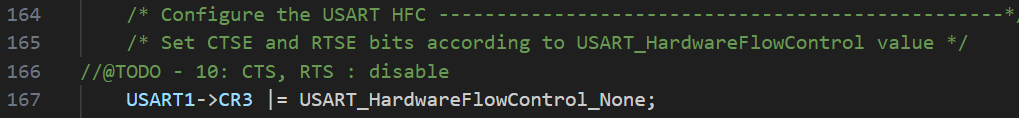
자동 생성된 설명

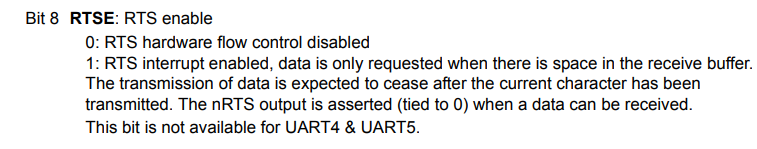
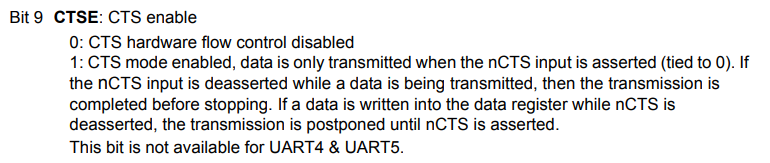
Tx, Rx를 사용하도록 enable로 설정한다.

-Todo 9



Stop bit을 1bit으로 설정한다.

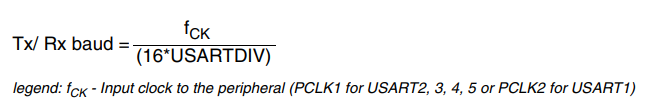
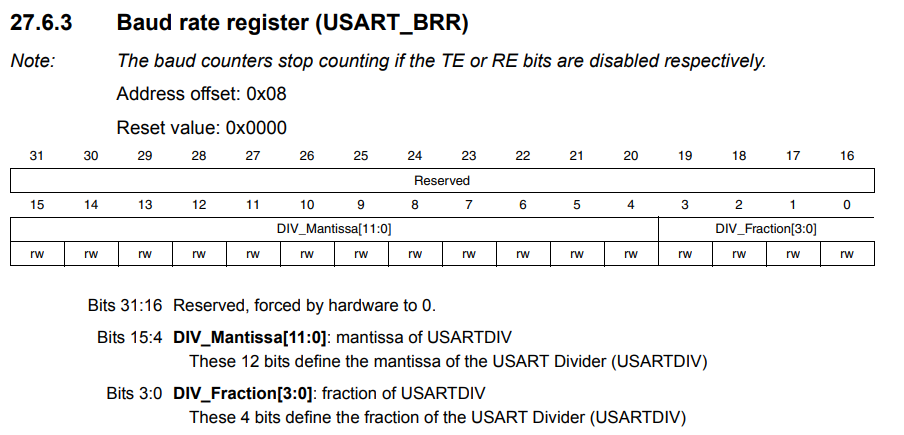
-Todo 10



CTSE와 RTSE를 disable로 설정한다.

-Todo 11

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Baud rate를 구하여 추가한다.

USARTDIV = 30.38

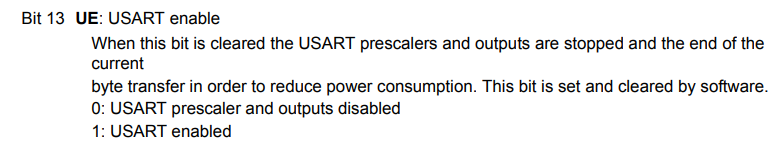
DIV\_Mantissa = 0d30 = 0x1E

DIV\_Fraction = 0.38 => 0x6

USART\_BRR = 0x1E6

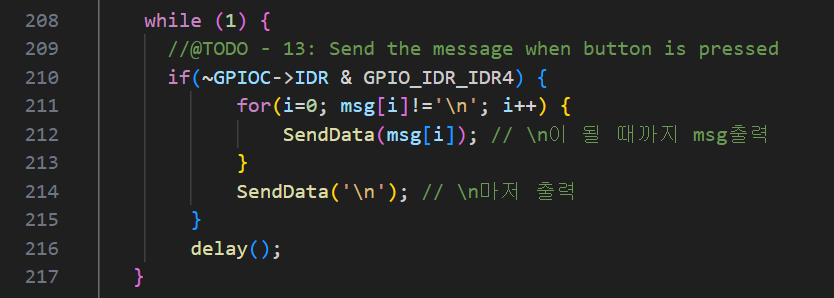
-Todo 12텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명



USART enable.

-Todo 13



버튼 1번을 감지해서 메시지를 출력하게 만든다.

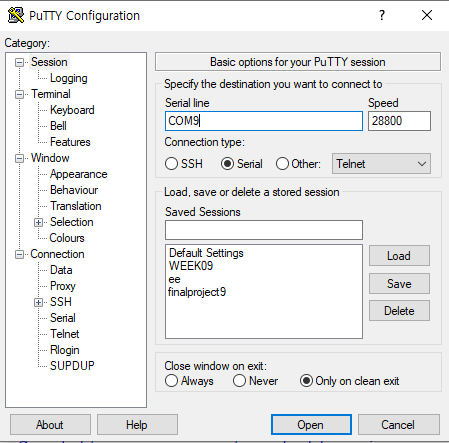
**MCO에서 나오는 System Clock 확인**

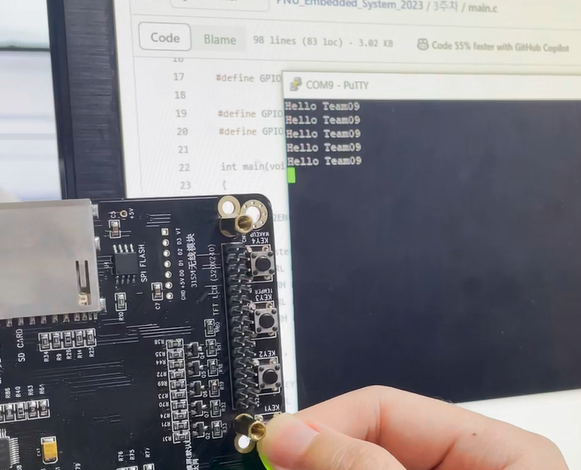
기계, 전자제품, 텍스트, 제어판이(가) 표시된 사진

자동 생성된 설명

MCO 출력 단자 PA8과 보드 GND를 오실로스코프에 연결하여 28MHz를 확인했다.

**Putty에서 출력확인**

****



1. **실험 결과**

이번 실험을 통해서 라이브러리를 활용하여 설정을 직관적으로 실행할 수 있었다. 앞으로 직관적으로 보드를 활용할 수 있게 되었다. 클럭에 대한 이해를 기반으로 커스텀 클럭을 각각 다른 포트에 집어넣고 활용할 수 있도록 클럭 트리를 이해하는 방법을 알게 되었다.

/\* HCLK = SYSCLK \*/

        RCC->CFGR |= (uint32\_t)RCC\_CFGR\_HPRE\_DIV1;

        /\* PCLK2 = HCLK / ?, use PPRE2 \*/

        RCC->CFGR |= (uint32\_t)RCC\_CFGR\_PPRE2\_DIV2;

        /\* PCLK1 = HCLK \*/

        RCC->CFGR |= (uint32\_t)RCC\_CFGR\_PPRE1\_DIV1;

이 부분에서 PCLK1,2에 다른 클럭을 집어넣을 수 있다는 개념이 수업시간에는 잘 잡히지 않았었는데 보고서를 쓰면서 이해할 수 있게 되었다.

시리얼 통신을 통해 컴퓨터와 통신을 진행했는데 앞으로 배울 블루투스 통신 등에 도움이 많이 될 것 같다.