임베디드 시스템

실험 및 설계

3주차 실험 보고서

2조 노윤정, 박건우, 이동근, 조영진

1. 실험목표

a) 라이브러리를 활용해 코드를 작성

b) Clock Tree 의 이해 및 사용자 Clock 설정

c) UART 통신의 원리를 배우고 실제 설정 방법 파악

2. 실험미션

a) Template.c 에 Todo 부분 라이브러리를 활용해 채워넣기

b) UART 통신을 이용하여 Putty 등의 시리얼 통신 프로그램으로 출력하기

ex) User S1 Button 누르는 동안, Putty에“Hello Team02\r\n”출력하기

3. 주의사항

a) 기기 작동중 케이블 뽑기 금지

b) 실험 후 자리 및 PC 깔끔히 뒷정리

4. 실험과정

1) Clcok Configure ( Todo 1~2 )

- i) Todo 1.

텍스트이(가) 표시된 사진

자동 생성된 설명

< Figure 1. Todo 1 >

RCC(Reset and Clock Control)의 CFGR(Clock Configuration Register)을 이용해 HCLK, PCLK2, PCLK1의 Clock을 조정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

< Figure 2. stm32f10x,h line 1735>

이때 stm32f10x.h 헤더파일을 보면 특정 레지스터의 옵션값을 상수로 선언해 둔 것을 알 수 있다. 이는 이 레지스터 뿐 아니라, 다른 레지스터 또한 마찬가지 이므로 앞으로 있을 코드에서도 이 pre Define 된 상수들을 사용한다면 유용할 것이다.

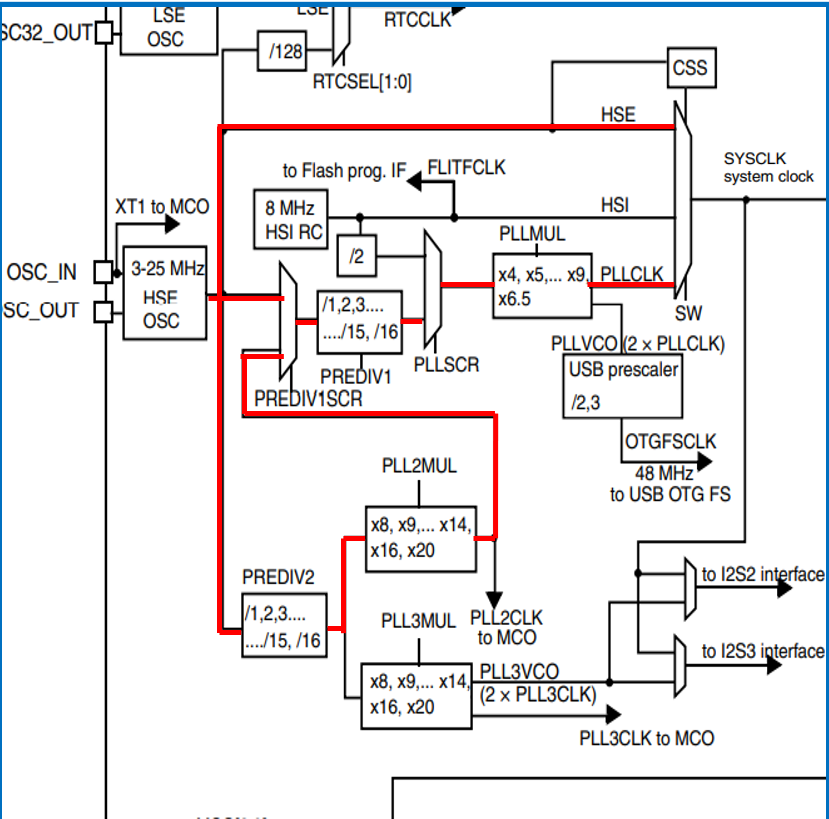
3주차 강의자료에서 제공된 stm32\_ReferenceManual의 140p 부터의 내용을 참고하여 RCC\_CFGR의 값을 조정해 주었다. HCLK는 SYS clock 과 일치시키라고 가이드를 주었으니 RCC\_CFGR\_HPRE \_DIV1 (System clock/1)을 선택했다. PCLK2 는 RCC\_CFGR\_PPRE2\_DIV2 (System clock / 2)를 선택하였다. 마찬가지로 PCLK1 또한 RCC\_CFGR\_PPRE1\_DIV1 (System clock / 1)을 선택해 주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

< Figure 3. Configure PLLs >

위의 코드와는 별개로, System clcok을 조정해야 한다. 이때 PLL(Phase-Locked Loop)를 이용하는데 PLL의 주 용도는 입력된 신호에 맞추어 출력 Clock 주파수를 조절하는 것이다. 때문에 PLL을 적절히 이용한다면 원하는 Clock 주파수를 얻을 수 있다.



< Figure 4. PLLs Circuit >

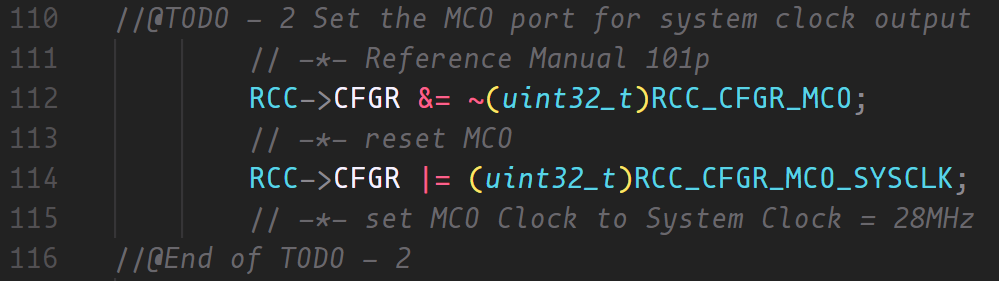
System clock 은 기본적으로 HSF OSC에서 생성된 25MHz를 사용하므로, Default Value를 25MHz로 맞추고, PLL을 조정해야 한다. Clock 은 HSF OSC에서 생성되어 PREDIV2, PLL2MUL, PREDIV1SCR, PREDIV1, PLLSCR, PLLMUL, SW를 거치게 된다. 따라서 이 값에 해당하는 레지스터를 조정하여야 한다. 우선 강의자료 ppt에서 힌트를 준 “28 = 25 /5 \*4 /5 \*7”을 변형해“28 = 25 /5 \*8 /10 \*7”로 Clock을 28MHz 로 조정해 주었다.

Clock의 주파수를 조절해주는 방법은 위에서 언급한 순서대로 PREDIV2, PLL2MUL, PREDIV1, PLLMUL의 값을 조정하는 것이다. 따라서 PREDIV2\_DIV5, PLL2MUL\_MUL8, PREDIV1\_DIV10, PLLMUL\_ MUL7의 상수를 이용해 설정해 주었다. 이때 DIV는 divide(나누기), MUL은 multiply(곱하기)를 뜻한다. DIV나 MUL 뒤에 오는 수는 얼마를 나누고 곱할 것인지를 뜻한다.

28MHz를 만들어준 후, 우리 팀은 ppt에 추가로 적힌 52MHz의 Clock을 만들어 보기로 했다. 본래 System clock 주파수는 하나만 써야 하므로 이는 쓰이지는 않겠지만 추가로 적어본 코드이다. 실험 제공 자료 ppt에 적힌 “52 = 25 /5 \*13 /5 \*4”를 이용해 PREDIV2\_DIV5, PLL2MUL\_MUL13, PREDIV1\_DIV5, PLLMUL\_MUL4 로 52MHz의 Clock을 구현하였다.

28MHz로 설정하든 52MHz로 설정하든 PREDIV1SCR은 PLL2로 설정해 주어야 하는데 이는 PREDIV1SCR가 Figure 4의 회로에서 HSF OSC에서 나온 25MHz Clock을 사용할지 아니면 PLL 회로들을 거쳐서 조정된 Clock을 사용할지 결정하는 회로이기 때문이다. 따라서 이 값은 공통적으로 PREDIV1SCR\_PLL2 로 설정해 주었다.

- ii) Todo 2.



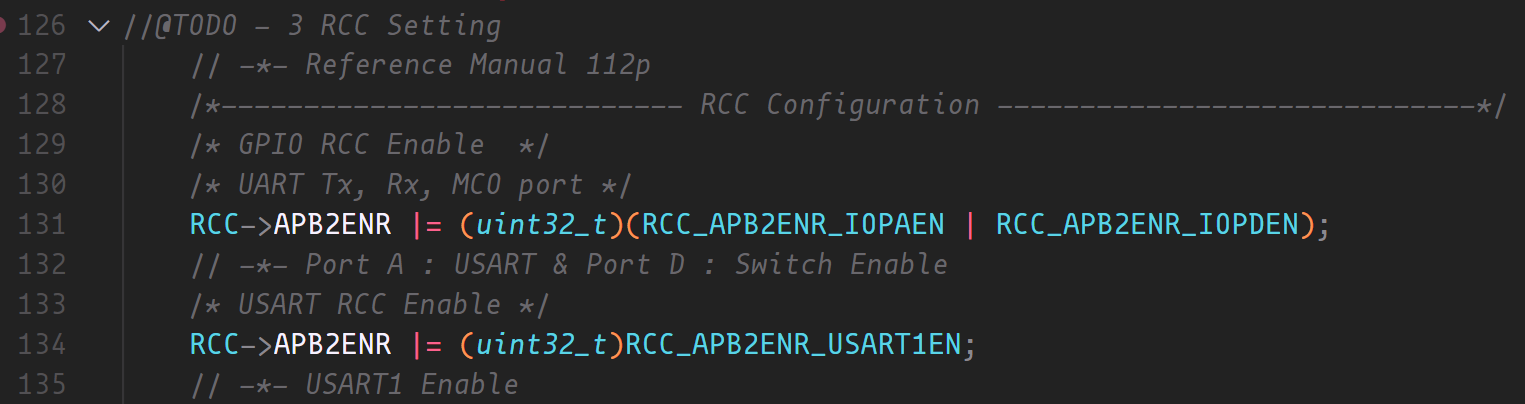
< Figure 5. Todo 2 >

MCO(Microcontroller Clock Output)을 초기화 시키고, 이를 System clock 으로 설정해 주었다. 이때 우리가 설정한 System clock 의 주파수는 28MHz 이므로 MCO의 주파수 또한 28MHz가 된다.

2) RCC & GPIO Configure ( Todo 3~4 )

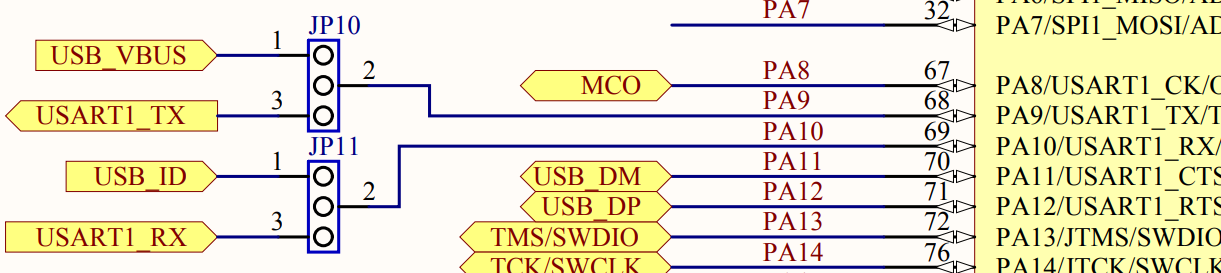
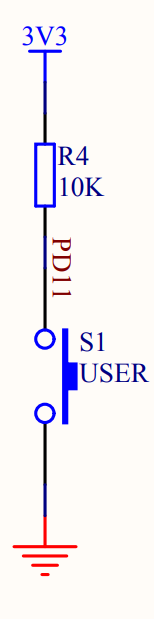
- i) Todo 3.

Clock 설정은 끝났으니, 이제 사용할 포트에 Clock을 인가해야 한다.



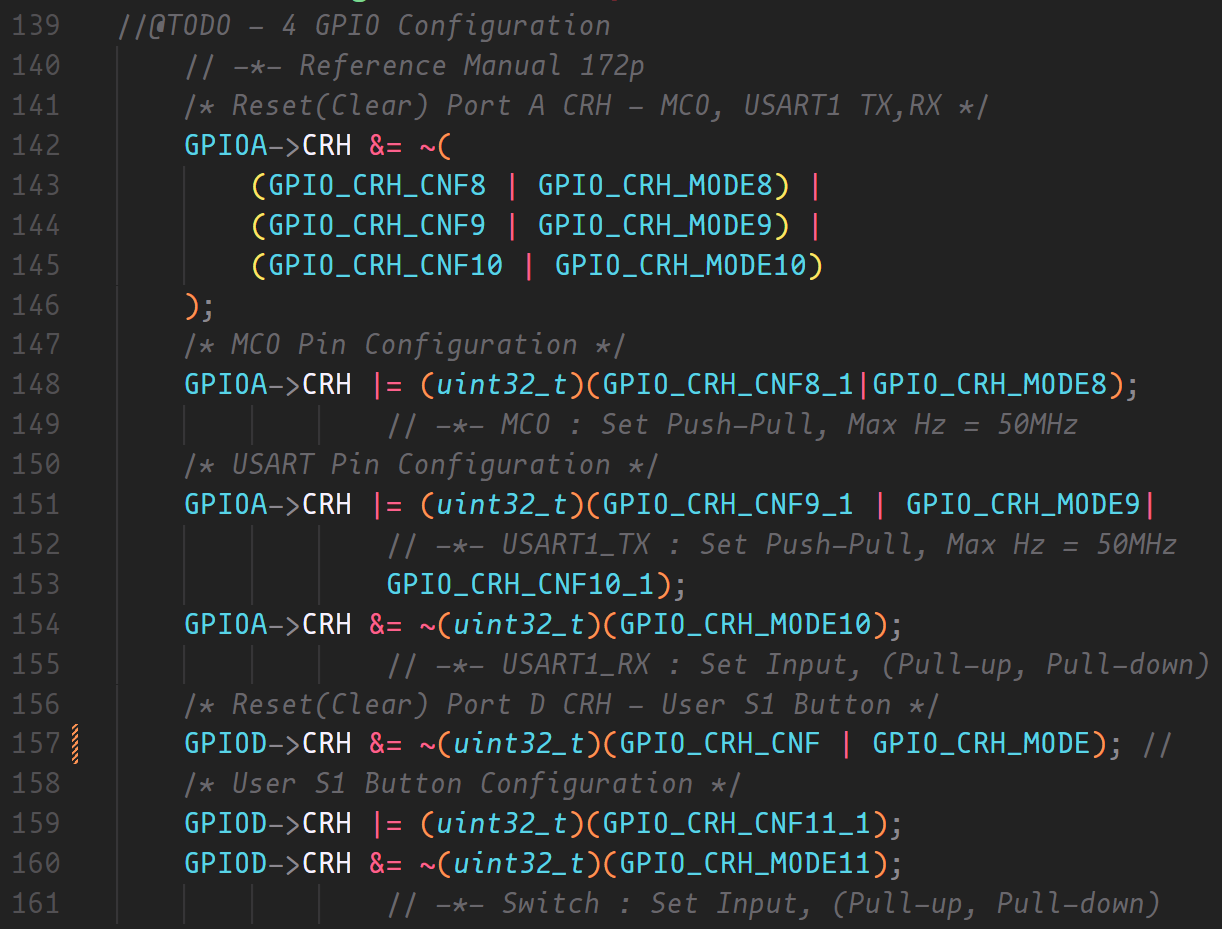
< Figure 6. Todo 3 >

Clock 인가는 RCC\_APB2ENR를 이용한다. 사용할 포트는 GPIOA(USART) 와 GPIOD(Switch) 이기때문에 각각의 포트를 Enable 시켜 주었다. 또한, USART1을 사용해야 하므로 USART1도 Enable로 설정했다.

< Figure 7. GPIO Port >

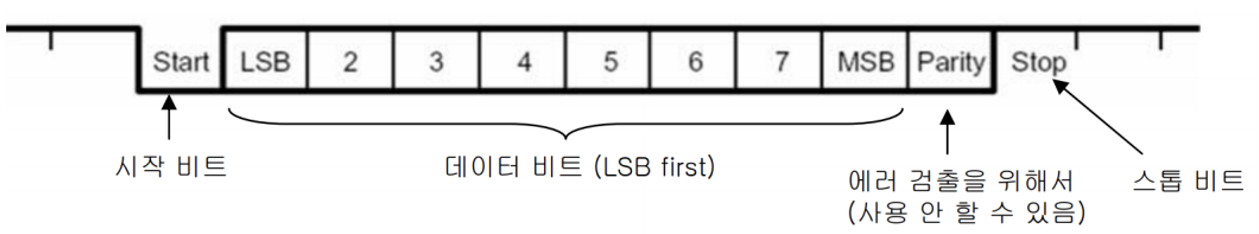
- ii) Todo 4.



< Figure 8. Todo 4 >

포트에 Clock을 인가한 다음에는 몇 번째 포트를 쓸 건지 설정하는 과정이 필요하다. USART를 사용하기 위해 활성화 해야 할 포트는 GPIOA8(MCO), GPIOA9(USART\_TX), GPIOA10(USART\_RX) 이므로 GPIOA\_CRH에 이들을 각각 설정해주었다.(line 148~154) Switch를 사용하기 위해 활성화 해야하는 포트는 GPIOD11 이므로 우선 GPIOD를 초기화 시켜준 후, GPIOD11를 설정하였다(line 157~160). (자세한건 Figure 7 내 주석 참고.)

3) UART Configure ( Todo 6~12 )



< Figure 9. UART data transmission >

- i) Todo 6.

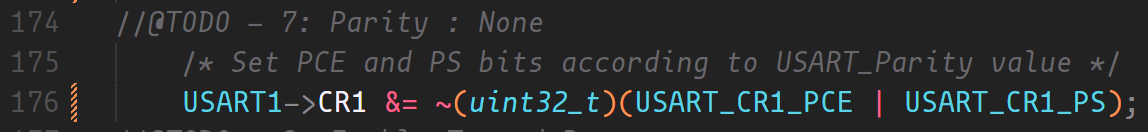
텍스트이(가) 표시된 사진

자동 생성된 설명

< Figure 10. Todo 6 >

USART1\_CR1을 통해 데이터를 주고받는 단위인 Word의 size를 설정할 수 있다. 가이드에서 8bit를 사용하라 하였기에 8bit로 설정하였다.

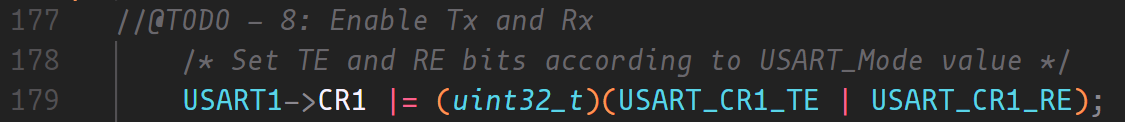
- ii) Todo 7.



< Figure 11. Todo 7 >

Word의 data 끝에 parity bit의 추가 여부를 CR1\_PCE로 설정할 수 있다. CR1\_PS는 PCE가 Enable일 때, parity bit를 even parity로 할지, odd parity로 할지 설정하는 값이다.

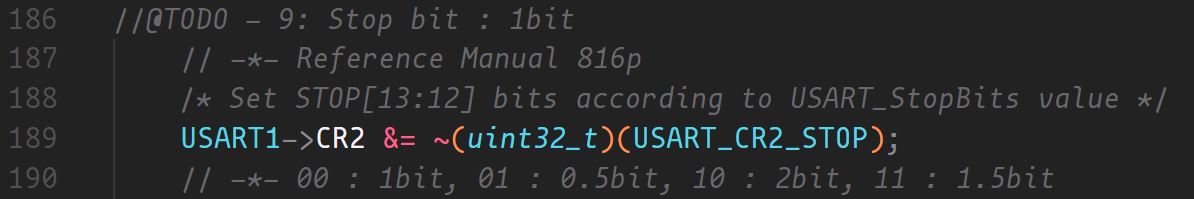
- iii) Todo 8.



< Figure 12. Todo 8 >

USART의 Tx와 Rx를 Enable 시키기 위해 CR1\_TE와 CR1\_RE를 설정해 주었다.

- iV) Todo 9.



< Figure 13. Todo 9 >

CR2\_STOP은 USART의 Stop bit를 설정하는 값이다. 가이드에서 이를 1bit 로 설정하라 하였으니, Reference를 참고해 값을 설정해 주었다.

- V) Todo 10.

텍스트이(가) 표시된 사진

자동 생성된 설명

< Figure 14. Todo 10 >

CR3\_CTSE 와 CR3\_RTSE를 이용해 CTS와 RTS를 disable 시킨다. 둘 다 interrupt 와 관련된 값이다.

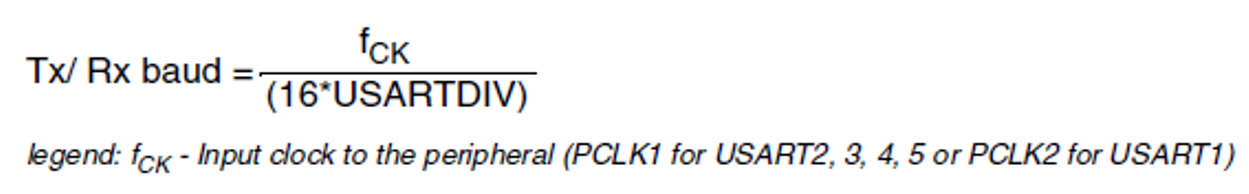
- Vi) Todo 11.

텍스트이(가) 표시된 사진

자동 생성된 설명

< Figure 15. Todo 11 >

BRR(Baud Rate Register)를 이용해 Baud Rate관련 값을 설정할 수 있다.



< Figure 16. Baud Rate >

위 수식은 Baud Rate를 계산하는 수식이다. 우리는 USART1를 사용하므로 f\_ck = PCLK2 = 28/2 MHz = 14MHz (1-i 참고) 이다. 목표 하고자 하는 Baud Rate 는 28800Hz 이므로 위 수식에 대입하면 USARTDIV = f\_ck / (baud \* 16) = 14MHz / (28.8KHz\*16) 30.38 이다.

USARTDIV(=BRR)은 상위 4bit가 Fraction, 하위 12bit가 Mantissa이다. 30.38을 Fraction과 Mantissa로 나눈다면 0.38 부분이 Fraction이 되고, 30이 Mantissa가 된다.

Fraction = x/16(x는 BRR의 상위 4bit)로 계산하고, 이 값이 0.38이 되어야 하므로 x/16 = 0.38, x = 0.38 \* 16 = 6.08 6 = 0x6 이다.

Mantissa = x(x는 BRR의 하위 12bit)로 계산하고, 이 값이 30이 되어야 하므로 x = 30 = 0x1E 이다.

위에서 구한 두 값을 토대로 설정해야 하는 USARTDIV(=BRR) 값은 0x1E6 이다. 이 값을 적용시켰지만 후술할 문제로 인해 실험 결과가 제대로 나오지 않아, 해당 값을 0x1E4로 수정하였다.

- Vii) Todo 12.

텍스트이(가) 표시된 사진

자동 생성된 설명

< Figure 17. Todo 12 >

마지막으로, CR1\_UE 를 이용해 UART를 Enable 시켜주었다.

5. 실험결과

6. 결론 및 제언