임베디드 시스템 설계 및 실험

004 분반 - 2 조 6 주차

실험 보고서

Clock Tree

실험자	202055606 주우성 202055623 허치영 202255632 벌드 바타르 아마르투브신 201724637 오치어 자미안퓨레브 202055629 밧툴가 바잘삿
실험날짜	2024-10-10
제출날짜	2024-10-16

1. 실험 제목

Clock Tree

2. 실험 목적

- Clock Tree 의 이해 및 사용자 Clock 설정
- UART 통신의 원리를 배우고, 실제 설정 방법 파악

3. 실험 원리 및 이론

1. 라이브러리 활용

기존에 volatile 로 타입캐스트해 Configuration 이나 enable 을 해줬다면 이제는 헤더파일 안에 들어있는 구조체에 접근하거나 정의된 상수를 이용하여 쉽게 설정할 수 있다.

직접 주소로 접근	(*(volatile unsigned int *) 0x40021018) &= ~0x20; (*(volatile unsigned int *) 0x40021018) |= 0x20; (*(volatile unsigned int *) 0x40011000) &= ~0x00000F00; (*(volatile unsigned int *) 0x40011000) |= 0x00000400;
정의된 주소 값 사용	RCC->APB2ENR &= ~(RCC_APB2ENR_IOPDEN); RCC->APB2ENR |= RCC_APB2ENR_IOPDEN; GPIOD->CRL &= ~(GPIO_CRL_CNF2 | GPIO_CRL_MODE2); GPIOD->CRL |= GPIO_CRL_MODE2_0;

2. Clock 의 개념

Clock 은 HSI Clock 과 HSE Clock 이 있다. 기본으로 HSI Clock 은 8MHz RC 오실레이터에서 생성되고 HSE Clock 은 HSE OSC 25MHz 생성된 Clock 은 시스템 클럭이나 PLL 클럭으로 사용할 수 있다.

Clock Tree 는 HSE, HSI, PLL 중 하나를 MUX 로 선택해 시스템 클럭으로 APB1 과 APB2 에 전달되고 이때 제대로 출력되는지 MCO MUX 를 이용해 MCO 에 출력해 오실로스코프로 확인할 수 있다. APB1 prescaler 을 이용하면 - 2 - 36MHz, APB2 prescaler 은 72MHz 까지의 클럭을 각각의 PCLK1 과 PCLK2 에 출력한다.

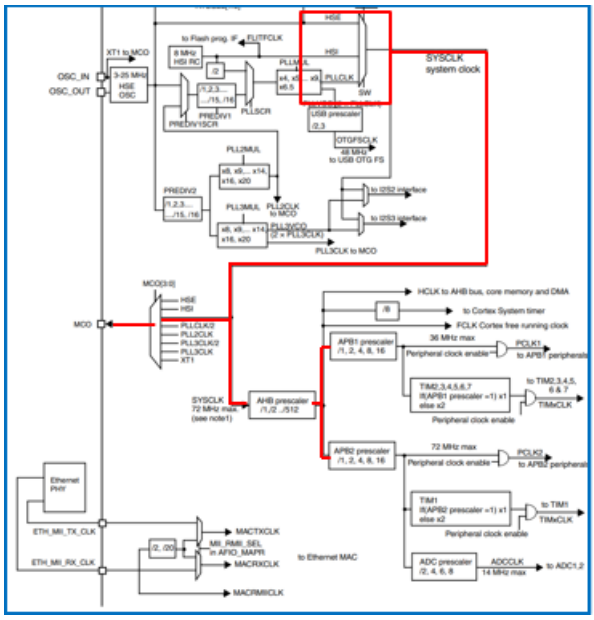


Figure 1. Clock Tree

PLL(Phase-Locked Loop)은 위상 동기 회로이고 MUX, DIV, MUL 을 거치며 입력 신호와 출력 신호를 통해 사용자가 원하는 주파수의 출력 신호를 제어한다.

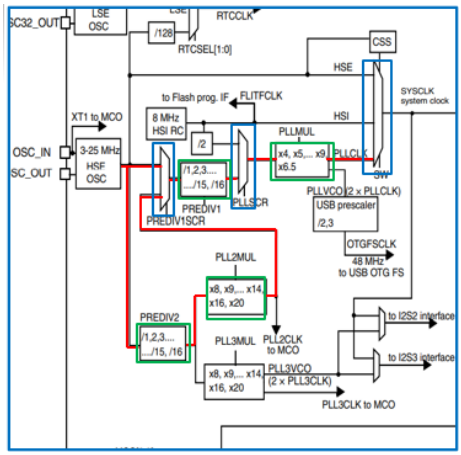


Figure 2. Phase-Locked Loop

3. 시리얼 통신

시리얼통신은 직렬 통신과 병렬 통신이 있다. 병렬 통신은 여러 개의 데이터 선을 이용해 비트를 하나씩 보내지만 비용이 많이 들고, 직렬 통신은 하나의 데이터 선으로 전송의 시작을 의미하는 Start bit, 정보를 담고 있는 Databit, 오류의 여부를 확인하는 Parity bit 와 데이터 통신의 종료를 의미하는 Stop bit 로 구분하는 방식이다.

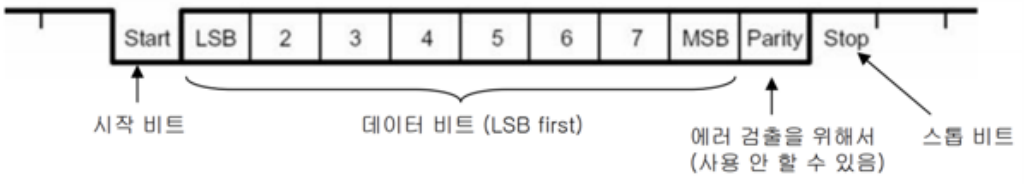


Figure 3. UART serial bit

4. 세부 실험 내용

1. TODO 1: Set the clock

RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE2_DIV2;

이 라인은 프리스케일러가 클록 주파수를 2 로 나누도록 설정하여 STM32 마이크로컨트롤러의 APB2 주변 장치 클록을 구성합니다.

RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_PREDIV1 | RCC_CFGR_PLLMULL4);

이 라인은 PREDIV1 을 PLL 소스로 선택하고 PLL 증폭 계수를 4 로 설정하여 STM32 마이크로컨트롤러에서 PLL(위상 잠금 루프)을 구성합니다.

RCC->CFGR2 |= (uint32_t)(RCC_CFGR2_PREDIV2_DIV5 | RCC_CFGR2_PLL2MUL13 | RCC_CFGR2_PREDIV1SRC_PLL2 | RCC_CFGR2_PREDIV1_DIV5);

PREDIV2 분배기를 5 로, PLL2 곱셈기를 13 으로, PREDIV1 소스를 PLL2 로, PREDIV1 분배기를 5 로 설정합니다. (52 = 25 / 5 * 13 / 5 * 4).

2. TODO 2: Set the MCO port for system clock output

RCC->CFGR |= (uint32_t)RCC_CFGR_MCO_SYSCLK;

STM32 마이크로컨트롤러의 MCO 핀에서 시스템 클록(SYSCLK)을 출력하도록 클록 출력(MCO)을 구성합니다.

3. TODO 3: RCC Setting

RCC->APB2ENR |= (uint32_t)(RCC_APB2ENR_IOPAEN | RCC_APB2ENR_AFIOEN);

APB2 버스의 포트 A 및 AFIO(Alternate Function I/O) 주변 장치의 클록을 활성화합니다.

RCC->APB2ENR |= (uint32_t)(RCC_APB2ENR_USART1EN);

USART1 주변장치에 대한 클록을 활성화합니다.

4. TODO 4: GPIO Configuration

GPIOA->CRH |= (GPIO_CRH_CNF8_1 | GPIO_CRH_MODE8_1 | GPIO_CRH_MODE8_0);

PA8 을 대체 기능 푸시풀 출력으로 구성합니다.

GPIOA->CRH |= (GPIO_CRH_CNF9_1 | GPIO_CRH_MODE9_1 | GPIO_CRH_MODE9_0 | GPIO_CRH_CNF10_1);

PA9 을 대체 기능 푸시풀 출력으로 구성합니다. PA10 을 대체 기능이 있는 입력으로 구성합니다.

GPIOA->CRL &= ~(GPIO_CRL_CNF0 | GPIO_CRL_MODE0);

PA0 의 구성 및 모드 비트를 지우고 재설정합니다.

GPIOA->CRL |= (GPIO_CRL_CNF0_1);

구성 비트를 설정하여 PA0 을 플로팅 입력으로 구성합니다.

5. TODO 8: Enable Tx and Rx

USART1->CR1 |= (uint32_t)(USART_CR1_TE | USART_CR1_RE);

USART1 제어 레지스터(CR1)의 해당 비트를 설정하여 USART1 주변 장치의 송신기(TE)와 수신기(RE)를 모두 활성화합니다.

6. TODO 11: Calculate & configure BRR

USART1->BRR |= (uint32_t)(0xA94)

0xA94 값을 통신 속도 레지스터(BRR)에 씁니다.

BRR= PCLK / (16 * Baud rate)

PCLK = 26Mhz, Baud rate = 9600 -> BRR = 169 (0xA9) & Fraction = 4 (0x4) -> 0xA94

7. TODO 12: Enable UART UE

```
USART1->CR1 |= (uint32_t)(USART_CR1_UE);
```

USART1 제어 레지스터(CR1)의 UE(USART Enable) 비트를 설정하여 USART1 주변 장치를 활성화합니다.

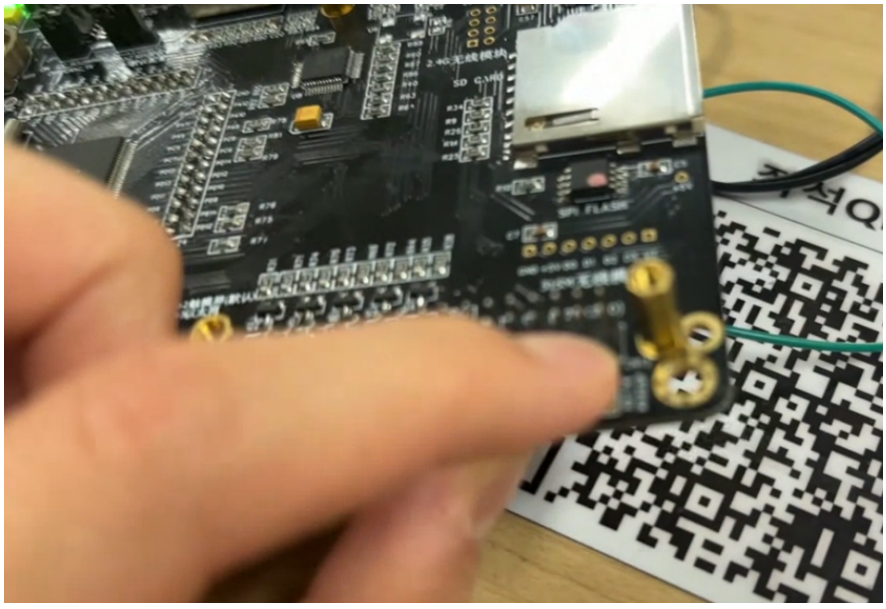
8. TODO 13: Send the message when button is pressed

```
if (~GPIOA->IDR & (GPIO_IDR_IDR0)) {
    for (i = 0; i < sizeof(msg); i++) {
        SendData(msg[i]);
    }
    delay();
}
```

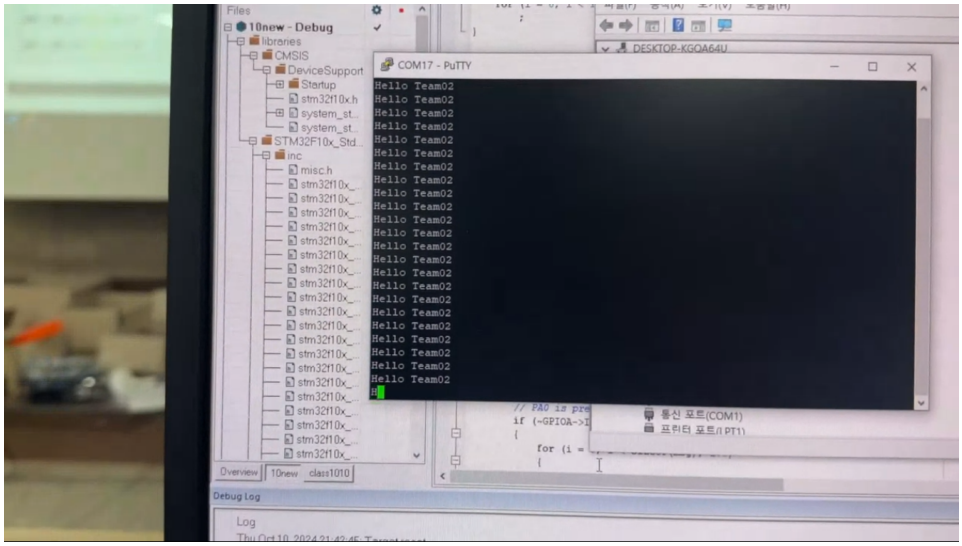
버튼을 누르면 SendData 함수를 호출하여 비트 단위로 USART 에 데이터를 전송합니다.
"Hello Team02₩r₩n"이 인쇄됩니다.

5. 실험 결과

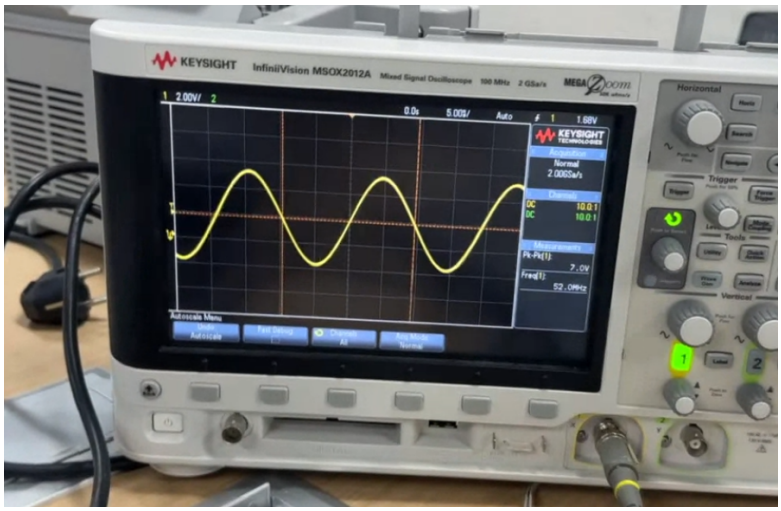
버튼을 누르다.



버튼을 누르면 PuTTY 에 "Hello Team02"가 출력됩니다.



모든 핀을 오실로스코프에 연결한 후 MCO 출력이 52Mhz 인 것을 볼 수 있으며 이는 실험이 제대로 작동하고 있음을 보여줍니다.



6. 분석 및 고찰

기존에 비트를 16 진수로 계산하고 volatile 로 타입 캐스트해 대입하는 귀찮은 과정 대신. 기존 라이브러리의 구조체나 상수를 가져오는 것으로 코드를 작성할 수 있어 한결 수월했다.

저번 실험까지는 STM32 보드 안에서 입력과 출력이 일어났는데, 이번 실험은 외부와 시리얼통신을 이용해 출력을 내보냈다.

7. 참고자료

- STM32107VCT6 schematic
- stm32 Datasheet
- stm32 Reference Manual