# 임베디드 시스템 설계 및 실험

# XX 분반

## 1. 실험 목표

## 2. 배경지식

### 2.1 UART와 USART

UART는 Universal Asynchronous Receiver/Transmitter의 약자로, 마이크로 컨트롤러에서 사용되어 온 통신 방식 중의 하나이다. 속도가 상대적으로 느리고 통신거리가 짧지만, 근거리에서 소량의 데이터를 보낼 때 유용하다. UART는 비동기 통신 방식을 채택하여, 데이터의 원활한 송수신을 위해서 clock을 대체할 수단이 필요하다. Start bit, stop bit, baud rate로 이를 대체한다. USART는 Universal Synchronous and Asynchronous serial Receiver/Transmitter의 약자로, UART이면서 때로는 clock에 따라 데이터를 보내거나 받을 수 있는 차이점이 있다.

### 2.2 Start bit/Stop bit

UART는 일반적으로 데이터를 전송하지 않을 때 데이터 전송 라인이 고전압 상태를 유지한다. 수신 UART가 고전압에서 저전압으로의 변화를 감지했을 때, baud rate의 주파수에서 데이터 프레임의 비트를 읽기 시작하

는데 이 때 쓰이는 것이 start bit이다. stop bit는 전송되는 데이터 패킷의 끝을 신호하기 위해 필요하다.

## 2.3 Baud rate

Baud rate는 1초당 데이터가 얼마만큼 변조되는지를 비율로 나타낸 것을 말한다. 비동기식 통신의 경우, 송신부와 수신부는 같은 속도로 데이터를 보내거나 받아야 하는데, 정확한 데이터 송/수신을 위해서 통신속도를 사용자가 지정하는 것을 baud rate 세팅이라고 한다. 사용자는 공식을 이용하여 baud rate를 계산하여 세팅할 수 있다.

<그림 삽입>

## 2.4 HSE/HIS CLOCK

CLOCK은 주기적인 전기적 펄스로, 정해진 CLOCK에 따라 모든 동작이 발생한다. HSE, HIS 두가지 CLOCK이 발생하는데, HSE의 경우 보드 외부에서 발생하는 CLOCK으로 실험에서 사용하는 보드는 25MHZ로 설정되어 있다. HIS의 경우 보드 내부 CLOCK으로 기본 8MHZ이다. 시스템 내부의 CLOCK은 HSI, HSE, PLL중 하나이다. PLL은 HIS와 HSE CLOCK을 곱하거나 나누는 연산으로 원하는 주파수 값을 생성할 수 있다. 이번 실험에서 맡아야 할 첫 번째 파트이다.

## 2.5 CLOCK TREE

기기의 clock을 어떻게 조정하고 다루는 지를 알기 위해서는 시스템이 제공하는 clock tree를 참고하여 확인할 수 있다. Stm32는 기본적으로 내장 clock이 있다. 필요에 따라 외부의 clock을 사용할 수 있도록 제작되었다. OSC_IN이라는 외부 clock 입력부에 외부 clock을 연결하고 clock tree를 설정해 주면 외부 clock을 이용할 수 있다. MCO는 microcontroller clock output의 약자로, 컨트롤러의 특정 clock을 곧바로 출력해주어 사용자가

clock을 확인할 수 있도록 한다. mco출력 부 바로 앞에 mux가 있어 어떤 clock을 출력을 할 것인지를 mco비트를 통하여 선택할 수 있다. Clock의 주파수를 사용자가 조정하기 위해서는 곱하기와 나누기 연산을 이용하여야 한다.

<그림 삽입>

곱하기, 나누기 연산을 진행할 수 있는 부분은 prediv2, pll2mul, pll3mul, pllmul이 있다. 제일 먼저 클럭은 OSC_IN으로 들어온다. 이를

PREDIV1SCR MUX에서 HSE 클럭을 이용할 지, OSC CLOCK을 사용할 지 결정할 수 있다. OSC클럭을 사용하기로 결정하면, PREDIV1SCR MUX에서 CLOCK을 내보내기 전에 PREDIV2, PRE2MUL에서 클럭값을 배분할 수 있다. 배분된 클럭은 PREDIV1SCR로 그리고 PLLSCR MUX에서 다시 HIS와 해당 주파수 중 하나를 선택하게 된다. 이렇게 나온 최종값이 PLLCLK가 된다. 그 후 PLLMUL에서 CLOCK배분을 하고, 최종적으로 SW MUX에서 HSE,HSI,PLLCLK 중 하나를 선택하게 된다. 이 값이 SYSCLK 시스템 클럭이 된다. MCO MUX 에서 SYSCLK를 선택하고, MCO를 PROBE하여 오실로스코프로 확인하면 시스템 클럭을 관찰할 수 있다.

## 2.6 RS-232C

이번 실험에서 추가적으로 사용할 보드 포트인 RS-232C 포트는 양방향 통신이 가능하고, TX,RX쌍이 있다. 즉, 한방향으로 송신, 수신이 모두 가능하다. 직렬 통신이라고도 하며, 신호 저항성이 매우 높다. 비동기 통신이므로 통신속도가 기기간에 다르게 설정되어 있으면 통신이 불가능하다. 그래서 이번 실험에서는 BAUD RATE를 조절하여 통신속도를 일치시키는 방법을 배운다.

# 3. 실험방법

## 3.1 putty 설정

실험결과를 확인하기 위해서는 컴퓨터에 putty가 설치되어 있어야 하며, 이번 실험에서는 baud rate가 115200으로 통일되어야 한다.

## 3.2 사용한 레지스터 설명

### 3.2.1 control register 1(USART_CR1)

control register 1에서 M은 word length이고, PCE, PS는 오류 검출을 위한 parity bit를 enable, select 하는 역할을 한다. 이번 실험에서는 parity control disabled이므로 사용되지 않았다. TE, RE는 각각 transmitter enable, receiver enable이며, 0은 disable, 1은 enable을 가리킨다.

### 3.2.2 Control register 2(USART_CR2)

Control register 2의 clken는 clock enable이고, cpol은 clock polarity, cpha는 clock phase이다. Cr2는 stop bit를 초기화하며, clock사용 설정에 이용된다.

### 3.2.3 Control register 3(USART_CR3)

Control register 3의 ctse는 cts enable을 뜻하며, RTSE는 RTS enab

le을 뜻한다. Cr1, cr2로 clock 세팅이 완료되면 초당 전송되는 baud rate를 조절할 때 이용되는 레지스터이다.

### 3.2.4 Baud rate register(USART_BRR)

Baud rate와 PCLK2 값을 이용하여 계산된 USARTDIV을 정수, 실수 부분으로 나눈 다음, DIV_Mantissa, DIV_Fraction에 각각 16진수로 변환하여 적용한 값을 전송하는데 필요한 레지스터로, data register인 ustar_dr을 적용하여 msb 12bit 에 정수부, lsb 12bit에 소수부의 값을 적용하여 전송하는 역할을 한다.

## 4.2 코드 설명

전체 시스템 클럭 설정을 위한 setsysclock()함수 수정

**Todo-1: 클럭을 52으로 바꾼다.**

**TODO-2: MCO 포트를 시스템 클럭 출력으로 설정한다.**

RCC->CFGR |= (UINT32_T)RCC_CFGR_MCO_SYSCLK;를 이용하여 설정한다.

**TODO-3: RCC설정**

**TODO-4: USART SETTING**

**Todo-5:**

**Todo-6 워드비트 8자리로 설정**

USART_CR1 레퍼런스를 참고하면, WORDLENGTH에 관한 비트 M의 비트값 설정을 보면 비트값이 0일 때, WORD_LENGTH가 8BIT임을 알 수 있다.

USARTINIT()함수의 초기화부분(맨 첫줄)에서 USART_CR1_M을 이용하여 비트를 0으로 초기화 하였으므로, 추가 코드 입력이 굳이 필요하지 않겠다.

**TODO7 : parity:none(disable)**

USART_CR1의 레퍼런스를 확인해 보면 비트값이 0일 때, PARITY기능을 disable 한다고 나와 있다. uartinit()함수 초기화 부분에서 usart_cr1_pce를 이용한 초기화를 진행하였으므로, 불필요한 추가 코드는 필요없다.

**Todo8: enable tx and rx**

```
USART1->CR1 |= USART_CR1_UE | USART_CR1_TE | USART_CR1_RE;
```

레퍼런스 27.6.4를 참고하여 연산을 진행하였다.

**Todo9: stop bit 1로 초기화**

```
USART1->CR2 &= (uint32_t) ~(USART_CR2_STOP);
```

위 코드로 이미 stop bit를 초기화 한 상태이다. 레퍼런스 27.6.5를 보면, usart_cr2가 나오는데, stop bit가 1이기 위해서는 비트의 값이 0이어야 한다. 초기화 해놓은 상태와 같은 상태이기 때문에 추가적인 코드작성이 필요하지 않다.

<출력 결과와 보드 사진 삽입>

결론

실험과정에서 baud rate계산 시 실수를 하여 값을 잘못 입력하였지만, 다시 계산하여 결과값을 도출해 냈다. 이번 실험으로 단거리 비동기 통신인 uart방식으로 컴퓨터와 보드간의 통신과정 및 결과를 확인할 수 있었고, 근거리에서의 uart통신의 신속성과 정확성을 알 수 있었다. 처음으로 헤더파일 참조를 사용하여서 불편하기도 했지만, 조금 더 써보면 코드의 작업속도를 증진시킬 수 있는 방법이 될 거 같다는 것을 알 수도 있을 것 같았다.