



Pusan
National
University



Oct. 11, 2023

조교
안현기

tykr0001@pusan.ac.kr

임베디드 시스템 설계 및 실험

수요일 분반

6주차
Clock Tree

Contents

6주차 실험 내용

- 라이브러리를 활용하여 코드 작성
- Clock Tree 의 이해 및 사용자 Clock 설정
- 오실로스코프를 이용한 clock 확인
- UART 통신의 원리를 배우고 실제 설정 방법 파악

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

```
#include "stm32f10x.h"

int main(){
    // RCC CLOCK enable Port D

    *((volatile unsigned int *)0x40021018) |= 0x20;
```

어렵게 주소, 값 일일이 입력하지 말고 라이브러리에 정의된 상수 및 구조체 이용하기!!!

```
RCC->APB2ENR |= RCC_APB2ENR_IOPEN
```

↓

- RCC_APB1RSTR_WWDGRST
- RCC_APB2ENR_ADC1EN
- RCC_APB2ENR_ADC2EN
- RCC_APB2ENR_AFIOEN
- RCC_APB2ENR_IOPAEN
- RCC_APB2ENR_IOPBEN
- RCC_APB2ENR_IOPCEN
- RCC_APB2ENR_IOPDEN**
- RCC_APB2ENR_IOPEEN
- RCC_APB2ENR_SPI1EN

```
/*< MCO configuration */
#define RCC_CFGR_MCO ((uint32_t)0x0F000000) /*< MCO[3:0] bits (Microcontroller Clock)
#define RCC_CFGR_MCO_0 ((uint32_t)0x01000000) /*< Bit 0 */
#define RCC_CFGR_MCO_1 ((uint32_t)0x02000000) /*< Bit 1 */
#define RCC_CFGR_MCO_2 ((uint32_t)0x04000000) /*< Bit 2 */
#define RCC_CFGR_MCO_3 ((uint32_t)0x08000000) /*< Bit 3 */

#define RCC_CFGR_MCO_NOCLOCK ((uint32_t)0x00000000) /*< No clock */
#define RCC_CFGR_MCO_SYSCLK ((uint32_t)0x04000000) /*< System clock selected as MCO source
#define RCC_CFGR_MCO_HSI ((uint32_t)0x05000000) /*< HSI clock selected as MCO source */
#define RCC_CFGR_MCO_HSE ((uint32_t)0x06000000) /*< HSE clock selected as MCO source */
#define RCC_CFGR_MCO_PLLCLK_Div2 ((uint32_t)0x07000000) /*< PLL clock divided by 2 selected as MCO source
#define RCC_CFGR_MCO_PLL3CLK ((uint32_t)0x08000000) /*< PLL3 clock selected as MCO source */
#define RCC_CFGR_MCO_PLL3CLK_Div2 ((uint32_t)0x09000000) /*< PLL3 clock divided by 2 selected as MCO source
#define RCC_CFGR_MCO_Ext_HSE ((uint32_t)0x0A000000) /*< XT1 external 3-25 MHz oscillator clock selected as MCO source
#define RCC_CFGR_MCO_PLL3CLK ((uint32_t)0x0B000000) /*< PLL3 clock selected as MCO source */
```

```
void RCC_Enable(void) {
    //TODO - 3 RCC Setting
    /*----- RCC Configuration -----*/
    /* GPIO RCC Enable */
    /* UART Tx, Rx, MCO */
    //RCC->APB2ENR |= ??
    /* USART RCC Enable */
    //RCC->APB2ENR |= ??
}

void PortConfiguration(void) {
    //TODO - 4 GPIO Configuration
    /* Reset Port A CRH - MCO, USART1 TX, RX */
    //GPIOA->CRH |= ??
    /* Reset Port D CRH - User SI Button */
    //GPIOD->CRH |= ??

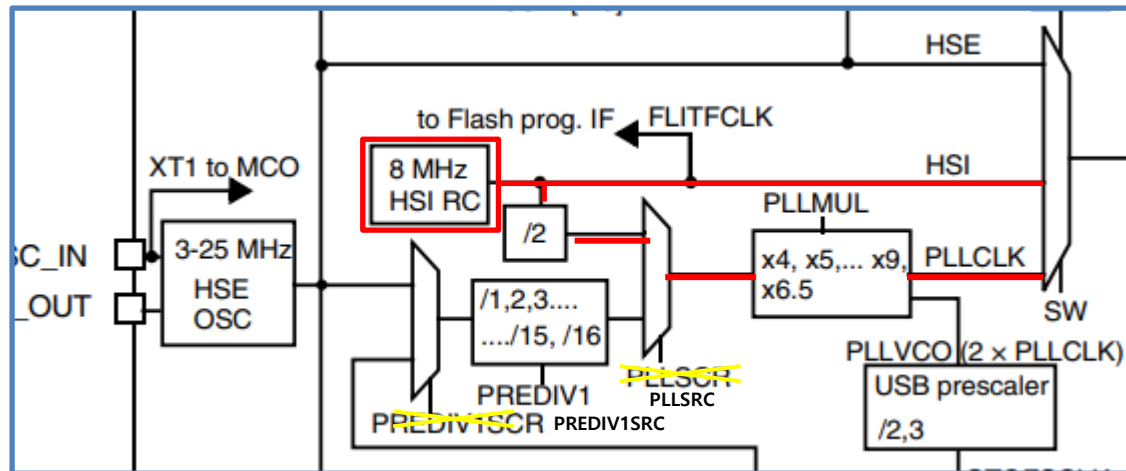
    /* MCO Pin Configuration */
    //GPIOC->CRH |= ??
}
```

Go to Definition of 'RCC_CFGR_MCO'

정의된 변수명 또는 함수명 : stm32f10x.h 또는 다른 라이브러리 파일에서 찾기 (F12)

코드에서 이름을 어느 정도 입력 후
ctrl+spacebar 또는 ctrl+alt+spacebar 를 누르면 정의된 이름들로 바로 변경되기도 함

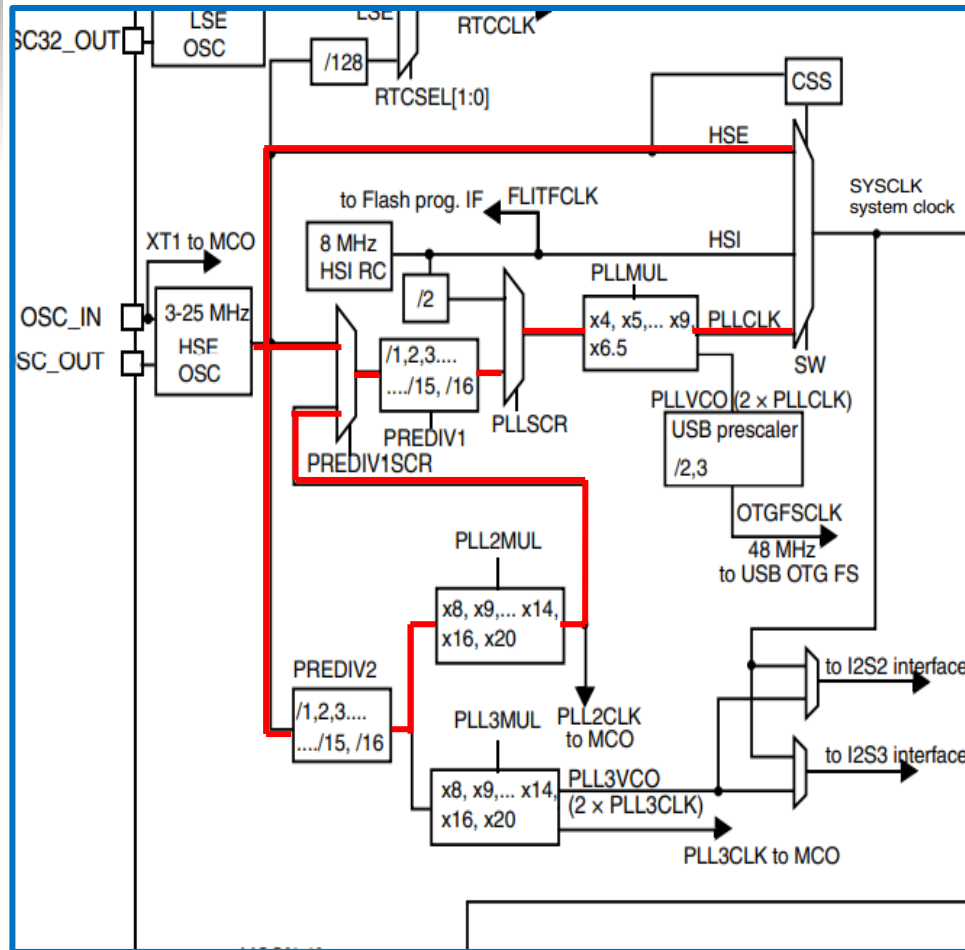
HSI Clock (High-speed internal)



- HSI Clock은 8 MHz RC 오실레이터에서 생성됨.
- 생성된 Clock은 시스템 클럭으로 사용하거나, PLL Clock으로 사용 가능

$$\text{PLLCLK} = \text{HSI RC} / 2 * \text{PLLMUL}$$

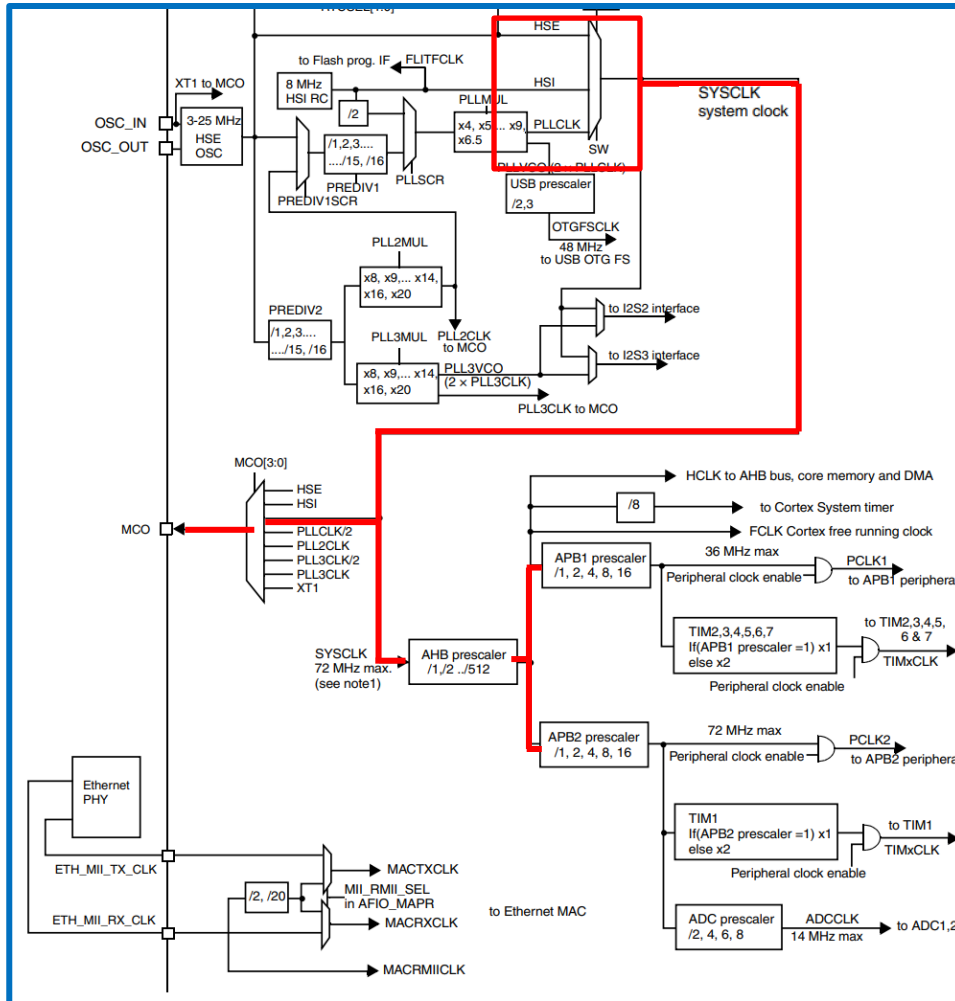
HSE Clock (High-speed external)



- HSE OSC에서 25 MHz 클럭을 생성
- 생성된 클럭은 바로 시스템 클럭으로 사용하거나, PLL Clock으로 사용할 수 있음

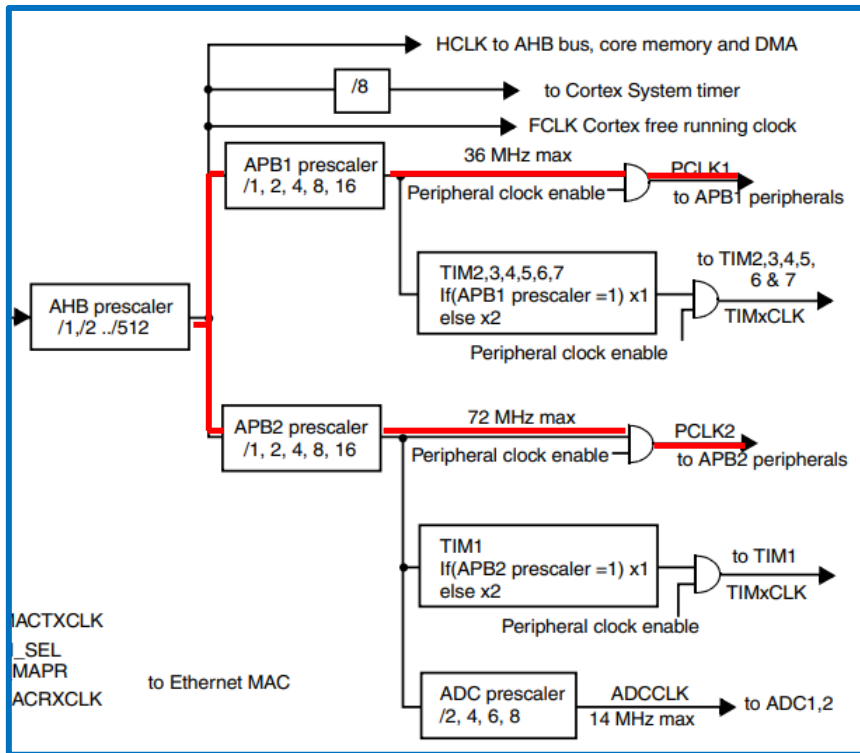
1. $PLLCLK = HSE\ OSC / PREDIV1 * PLLMUL$
2. $PLLCLK = HSE\ OSC / PREDIV2 * PLL2MUL / PREDIV1 * PLLMUL$

Clock Tree



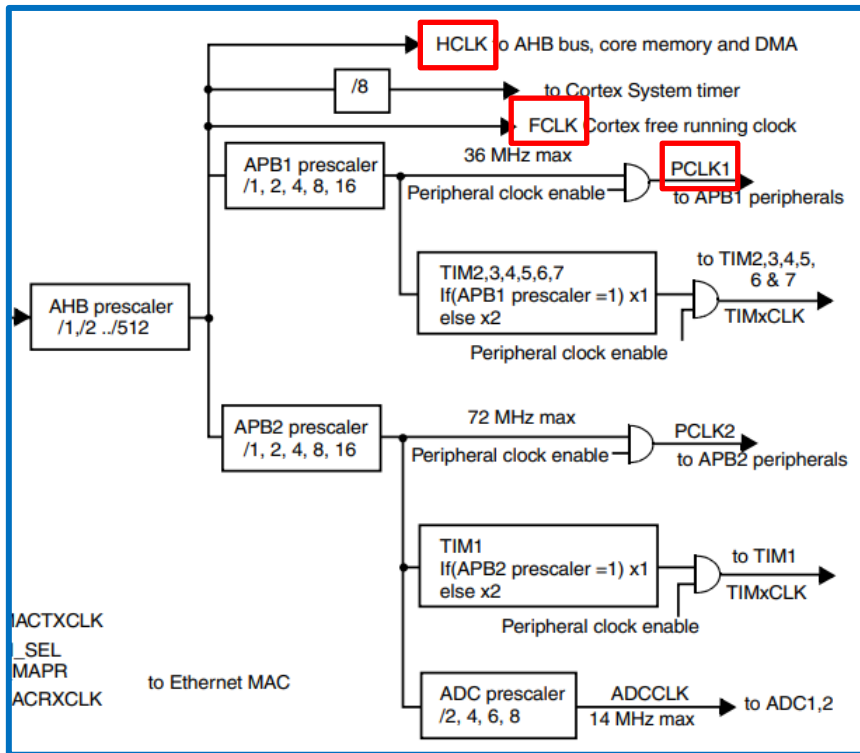
- HSI, HSE, PLL 중 SW MUX에 의해 시스템 클럭(최대 72MHz)으로 설정될 수 있다.
- 시스템 클럭은 APB1, APB2에 전달된다.
- 시스템 클럭은 MCO MUX를 통해서 MCO에 출력해 오실로스코프로 확인이 가능하다.

Clock Tree



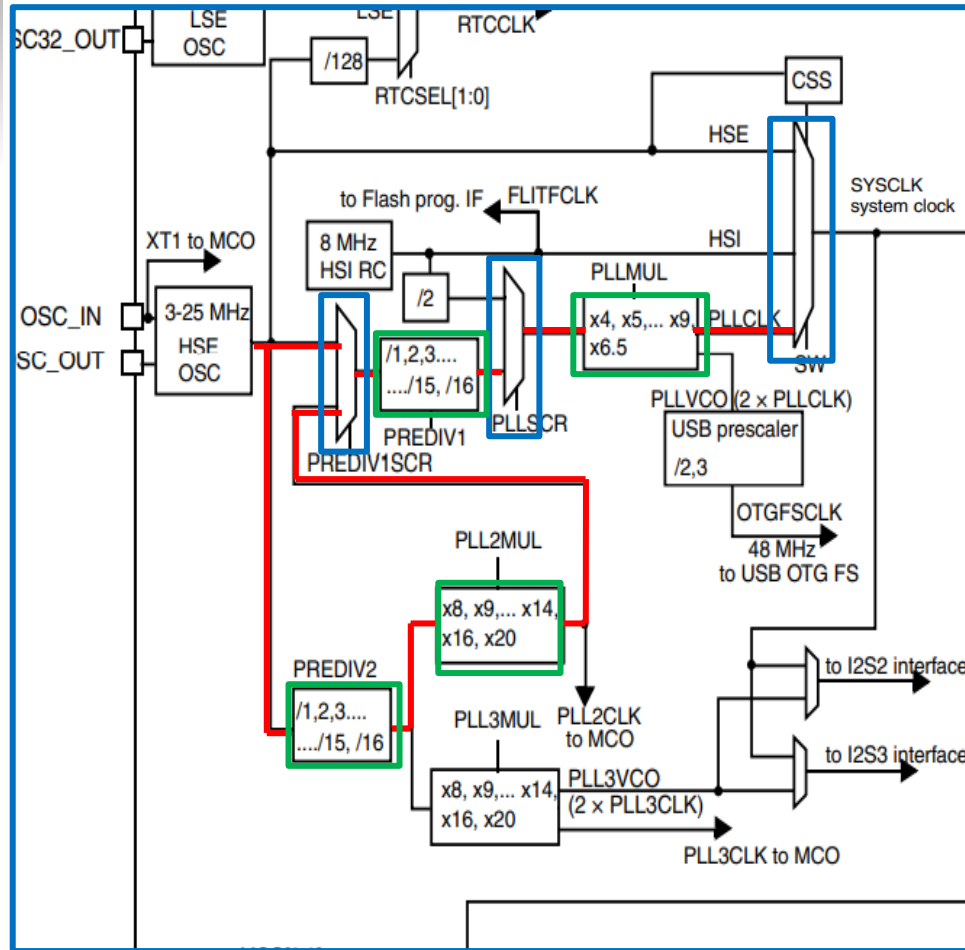
- 시스템 클럭을 APB1 prescaler을 이용해 최대 36MHz 클럭을 PCLK1에 제공 가능하다.
- 시스템 클럭을 APB2 prescaler을 이용해 최대 72MHz 클럭을 PCLK2에 제공 가능하다.

Clock Tree



- **FCLK** : Cortex System(CPU)에서 사용되는 Clock
- **HCLK** : DMA, Core memory, AHB Bus에서 사용되는 Clock
- **PCLK** : APB Bus에 사용되는 Clock

PLL(Phase-Locked Loop)



- PLL은 위상 동기 회로이며, 입력 신호와 출력신호를 이용해 출력신호를 제어하는 시스템을 말한다.
 - 입력된 신호에 맞추어 출력 신호의 주파수 조절이 목적이다.
1. HSE 오실레이터에 Clock(25MHz)을 생성
 2. 주어진 MUX와 DIV, MUL을 이용해 원하는 시스템 클럭을 만든다.

PLL(Phase-Locked Loop)

Clock 제어에 사용되는 여러가지 레지스터

8.3.2 Clock configuration register (RCC_CFGR)

Address offset: 0x04

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

1 or 2 wait states inserted only if the access occurs during a clock source switch.

8.3.12 Clock configuration register2 (RCC_CFGR2)

Address offset: 0x2C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

8.3.1 Clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

Access: no wait state, word, half-word and byte access

예) APB2에서 2로 나누기

8.3.2 Clock configuration register (RCC_CFGR)

Address offset: 0x04

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

1 or 2 wait states inserted only if the access occurs during a clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved				MCO[3:0]				Res.	OTGFS PRE		PLLMUL[3:0]			PLL XTPRE		PLL SRC
				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ADC PRE[1:0]				PPRE2[2:0]				HPRE[3:0]				SWS[1:0]		SW[1:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw	

Bits 13:11 **PPRE2[2:0]: APB high-speed prescaler (APB2)**

Set and cleared by software to control the division factor of the APB High speed clock (PCLK2).

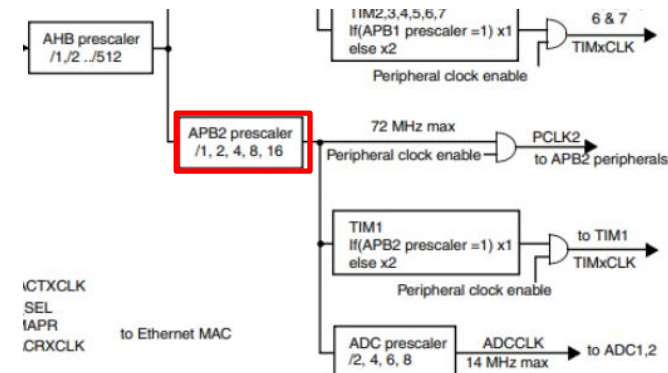
0xx: HCLK not divided

100: HCLK divided by 2

101: HCLK divided by 4

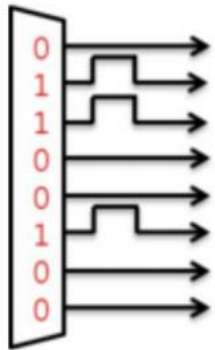
110: HCLK divided by 8

111: HCLK divided by 16

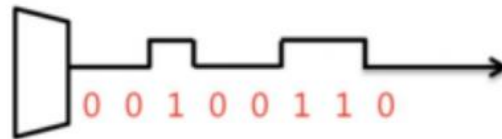


RCC->CFGR |= RCC_CFGR_PPRE2_DIV2

100(10진수) → 0x64(16진수) → 01100100(2진수)



병렬 통신



직렬 통신

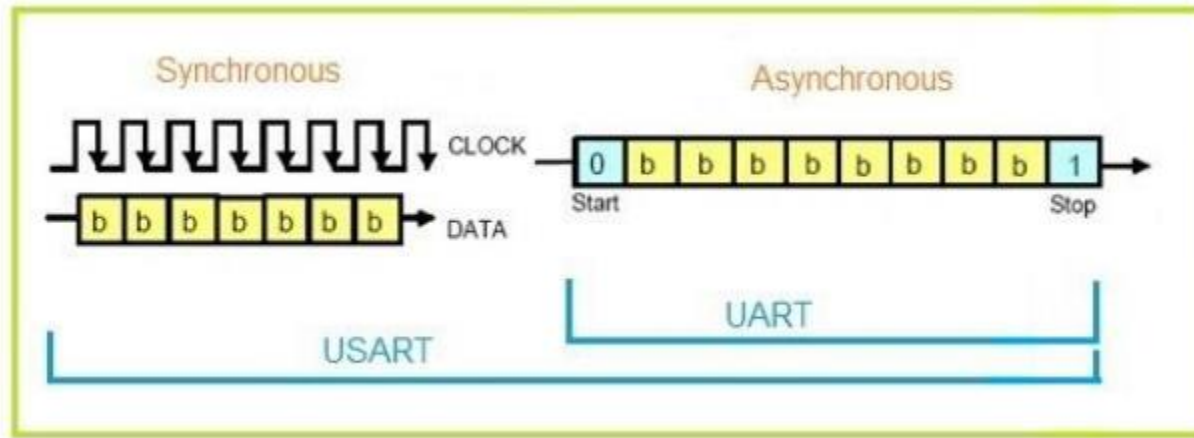
1. Parallel Communication(병렬 통신)

- 여러 개의 데이터 선을 이용해, 각각의 선에 비트를 하나씩 보내는 방법

2. Serial Communication(직렬 통신)

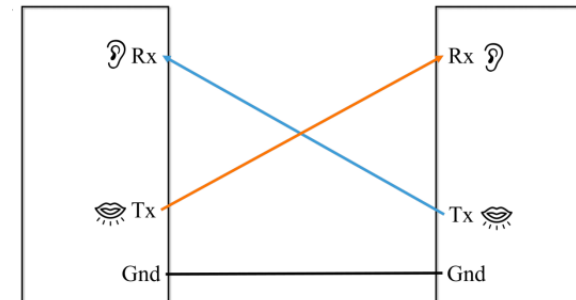
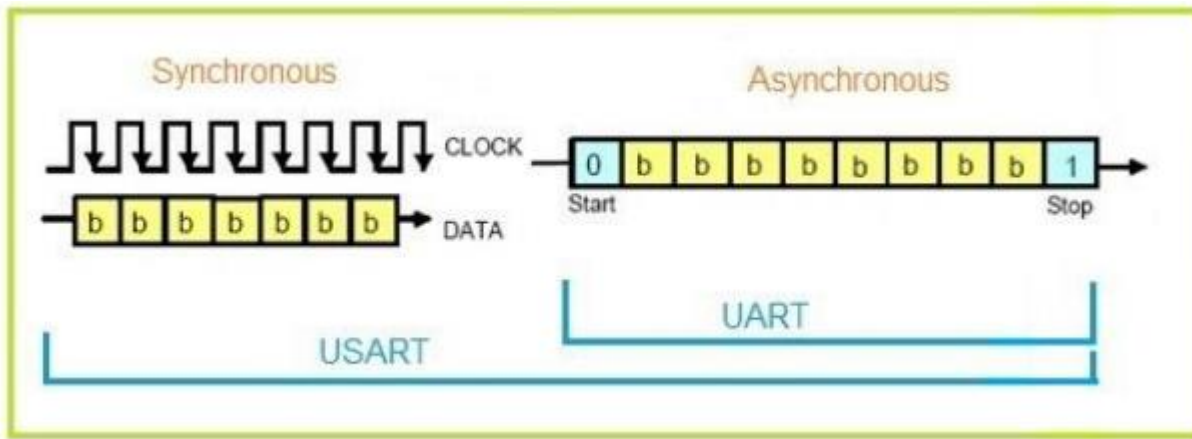
- 하나의 데이터 선을 이용해 비트를 차례로 보내는 방법
- 속도는 느리나, 데이터 선을 연장하기 위한 비용이 적음

Universal Sync/Async Receiver/Transmitter (USART)



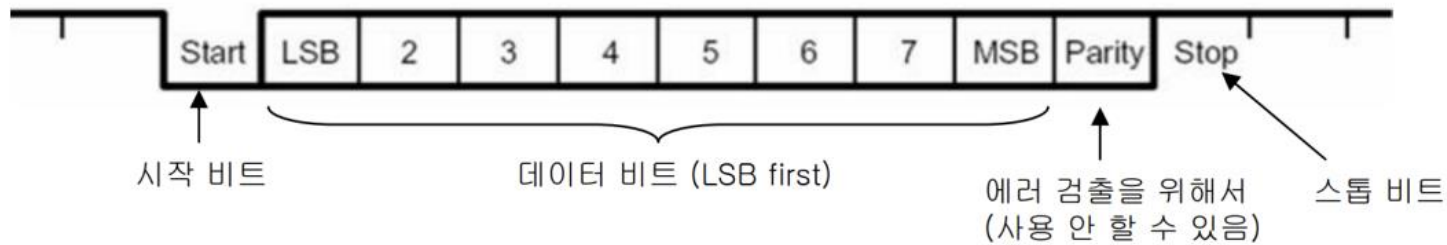
- 동기식 통신을 지원하는 UART
- 데이터 동기화를 위해 같은 클럭을 사용함
- 클럭 전송을 위한 별도의 클럭선 필요
- Start bits/ Stop bits가 없고 Data bits를 클럭에 동기화해 전송

Universal Asynchronous Receiver/Transmitter (UART)



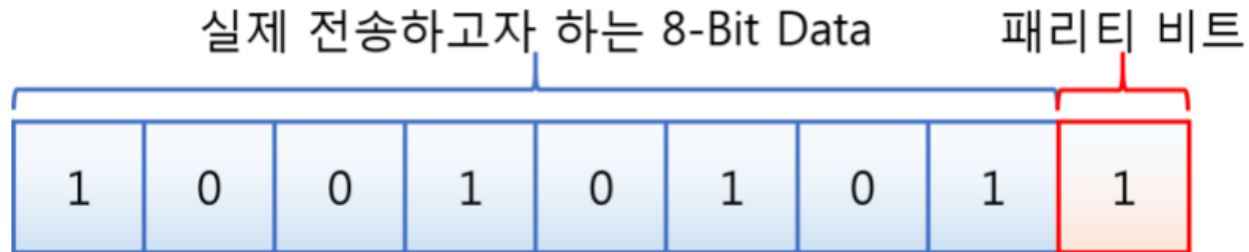
- 비동기 통신 프로토콜
- Rx(데이터 수신)와 Tx(데이터 송신) 교차연결
- 비동기 통신이기 때문에 Baud Rate를 일치 시켜야 한다.
- Baud Rate란 초당 얼마나 많은 심볼(의미 있는 데이터 묶음)을 전송할 수 있는가

Universal Asynchronous Receiver/Transmitter



- Start bit : 통신의 시작을 의미하며 1bit 길이 만큼 유지. "이제부터 정해진 약속에 따라 통신을 시작한다" 라는 의미
- Data bit : 8~9 bit의 데이터를 전송. Bit의 개수는 레지스터 설정에 따라 다름
- Parity bit : 에러 검사를 위한 값. 수신 측에서 이 bit를 이용해 에러 검사
- Stop bit : 통신의 종료를 알림. 레지스터에 따라 1, 1.5, 2bit로 설정

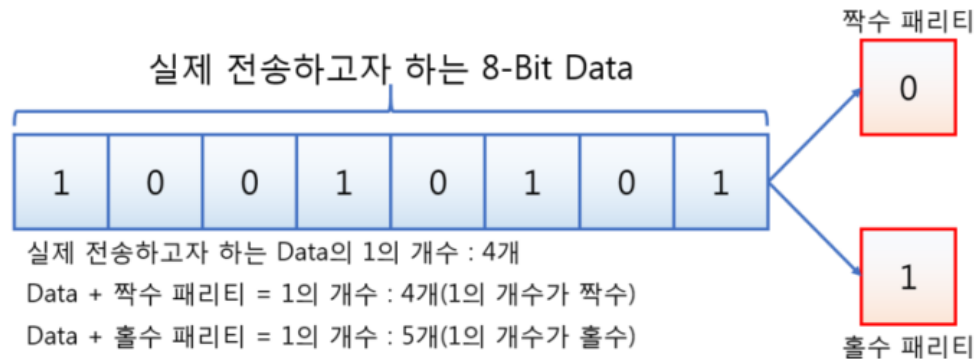
Universal Asynchronous Receiver/Transmitter



패리티 비트

- 데이터 송수신 시 오류로 인해 bit가 1 -> 0 또는 0 -> 1로 바뀌었을 때 확인할 수 있는 보호장치
- 에러 발생 여부는 확인 가능하나, 오류 수정은 불가
- 수신 측에서 에러 발생 여부 확인 후, 데이터 재요청 가능

Universal Asynchronous Receiver/Transmitter



Even parity, Odd Parity 설정에 따라, Parity bit의 값이 다름

- Even Parity : (전송하고자 하는 데이터 + parity bit) 중 1인 bit의 개수가 짝수
- Odd Parity : (전송하고자 하는 데이터 + parity bit) 중 1인 bit의 개수가 홀수
- Even, Odd 설정은 송수신 측이 미리 동기화되어 있어야 함.

System Clock Generate

```
//@TODO - 1 Set the clock
/* HCLK = SYSCLK */
RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;
/* PCLK2 = HCLK */
RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE2_DIV1;
/* PCLK1 = HCLK */
RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE1_DIV1;

/* Configure PLLs -----*/
RCC->CFGR &= (uint32_t)~(RCC_CFGR_PLLXTPRE | RCC_CFGR_PLLSRC | RCC_CFGR_PLLMULL);
RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLXTPRE_PREDIV1 | RCC_CFGR_PLLSRC_PREDIV1 |
RCC_CFGR_PLLMULL1);

RCC->CFGR2 &= (uint32_t)~(RCC_CFGR2_PREDIV2 | RCC_CFGR2_PLL2MUL |
RCC_CFGR2_PREDIV1 | RCC_CFGR2_PREDIV1SRC);
RCC->CFGR2 |= (uint32_t)(RCC_CFGR2_PREDIV2_DIV1 | RCC_CFGR2_PLL2MUL1 |
RCC_CFGR2_PREDIV1SRC_PLL2 | RCC_CFGR2_PREDIV1_DIV1);

//@End of TODO - 1

/* Enable PLL2 */
RCC->CR |= RCC_CR_PLL2ON;
/* Wait till PLL2 is ready */
while ((RCC->CR & RCC_CR_PLL2RDY) == 0)
{
}

/* Enable PLL */
RCC->CR |= RCC_CR_PLLON;
/* Wait till PLL is ready */
while ((RCC->CR & RCC_CR_PLLRDY) == 0)
{
}

/* Select PLL as system clock source */
RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_
RCC->CFGR |= (uint32_t)RCC_CFGR_SW_PLL;
/* Wait till PLL is used as system clock sour
while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS) !
{
}

/* Select System Clock as output of MCO */
//@TODO - 2 Set the MCO port for system clock output
RCC->CFGR &= ~(uint32_t)RCC_CFGR_MCO;
// RCC->CFGR |= ??

//@End of TODO - 2

void RCC_Enable(void) {
//@TODO - 3 RCC Setting
/*----- RCC Configur
/* GPIO RCC Enable */
/* UART Tx, Rx, MCO */
//RCC->APB2ENR |= ??
/* USART RCC Enable */
// RCC->APB2ENR |= ??

void PortConfiguration(void) {
//@TODO - 4 GPIO Configuration
/* Reset Port A CRH */
// GPIOA->CRH &= ??
/* Reset Port B CRH */
// GPIOB->CRH &= ??

/* MCO Pin Configuration */
// GPIOA->CRH |= ??
/* USART Pin Configuration */
// GPIOA->CRH |= ??
/* User S1 Button Configuration */
// GPIOD->CRH |= ??
}
```

TODO 부분

- reference 및 datasheet 참고하여 코드 채워 넣기
- 주소 및 대입 값 정의된 이름으로 입력하기

```
void UartInit(void) {
/*----- USART CR1 Configuration -----*/
/* Clear M, PCE, PS, TE and RE bits */
USART1->CR1 &= ~(uint32_t)(USART_CR1_M | USART_CR1_PCE | USART_CR1_PS | USART_CR1_TE | USART_CR1_RE);
/* Configure the USART Word Length, Parity and mode -----*/
/* Set the M bits according to USART_WordLength value */
//@TODO - 6: WordLength : 8bit

/* Set PCE and PS bits according to USART_Parity value */
//@TODO - 7: Parity : None

/* Set TE and RE bits according to USART_Mode value */
//@TODO - 8: Enable Tx and Rx
// USART1->CR1 |= ??

/*----- USART CR2 Configuration -----*/
/* Clear STOP[13:12] bits */
USART1->CR2 &= ~(uint32_t)(USART_CR2_STOP);
/* Configure the USART Stop Bits, Clock, CPOL, CPHA and LastBit -----*/
USART1->CR2 &= ~(uint32_t)(USART_CR2_CPHA | USART_CR2_CPOL | USART_CR2_CLKEN);
/* Set STOP[13:12] bits according to USART_StopBits value */
//@TODO - 9: Stop bit : 1bit

/*----- USART CR3 Configuration -----*/
/* Clear CTSE and RTSE bits */
USART1->CR3 &= ~(uint32_t)(USART_CR3_CTSE | USART_CR3_RTSE);
/* Configure the USART HFC -----*/
/* Set CTSE and RTSE bits according to USART_HardwareFlowControl value */
//@TODO - 10: CTS, RTS : disable

/*----- USART BRR Configuration -----*/
/* Configure the USART Baud Rate -----*/
/* Determine the integer part */
/* Determine the fractional part */
//@TODO - 11: Calculate & configure BRR
// USART1->BRR |= ??

/*----- USART Enable -----*/
/* USART Enable Configuration */
//@TODO - 12: Enable UART
// USART1->CR1 |= ??
}
```

GPIO 설정

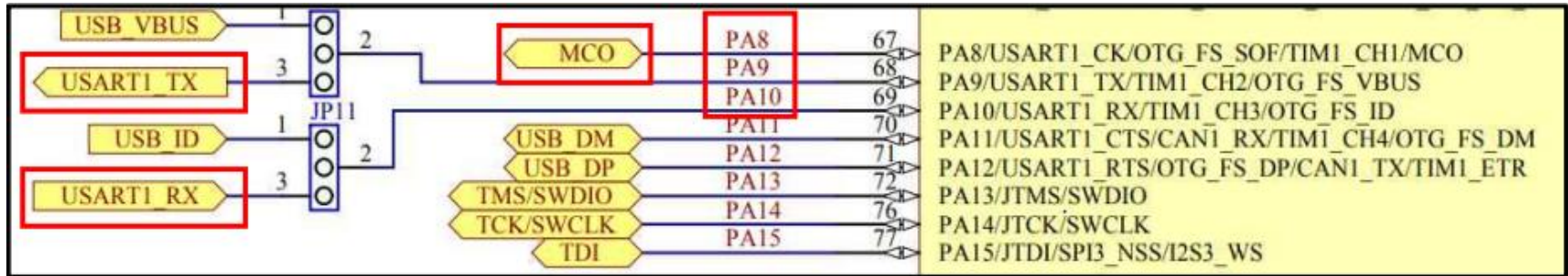


Table 5. Pin definitions (continued)

Pins			Pin name	Type ⁽¹⁾	I/O Level ⁽²⁾	Main function ⁽³⁾ (after reset)	Alternate functions ⁽⁴⁾	
							Default	Remap
J9	-	56	PD9	I/O	FT	PD9	-	USART3_RX/ ETH_MII_RXD0/ ETH_RMII_RXD0
H9	-	57	PD10	I/O	FT	PD10	-	USART3_CK/ ETH_MII_RXD1/ ETH_RMII_RXD1
G9	-	58	PD11	I/O	FT	PD11	-	USART3_CTS/ ETH_MII_RXD2
K10	-	59	PD12	I/O	FT	PD12	-	TIM4_CH1 / USART3_RTS/ ETH_MII_RXD3
J10	-	60	PD13	I/O	FT	PD13	-	TIM4_CH2
H10	-	61	PD14	I/O	FT	PD14	-	TIM4_CH3
G10	-	62	PD15	I/O	FT	PD15	-	TIM4_CH4
F10	37	63	PC6	I/O	FT	PC6	I2S2_MCK/	TIM3_CH1
E10	38	64	PC7	I/O	FT	PC7	I2S3_MCK	TIM3_CH2
F9	39	65	PC8	I/O	FT	PC8	-	TIM3_CH3
E9	40	66	PC9	I/O	FT	PC9	-	TIM3_CH4
D9	41	67	PA8	I/O	FT	PA8	USART1_CK/OTG_FS_SOF / TIM1_CH1 ⁽⁷⁾ /MCO	-
C9	42	68	PA9	I/O	FT	PA9	USART1_TX ⁽⁷⁾ /TIM1_CH2 ⁽⁷⁾ / OTG_FS_VBUS	-
D10	43	69	PA10	I/O	FT	PA10	USART1_RX ⁽⁷⁾ / TIM1_CH3 ⁽⁷⁾ /OTG_FS_ID	-
C10	44	70	PA11	I/O	FT	PA11	USART1_CTS / CAN1_RX / TIM1_CH4 ⁽⁷⁾ /OTG_FS_DM	-
B10	45	71	PA12	I/O	FT	PA12	USART1_RTS / OTG_FS_DP / CAN1_TX ⁽⁷⁾ / TIM1_ETR ⁽⁷⁾	-
A10	46	72	PA13	I/O	FT	JTMS-SWDIO	-	PA13
F8	-	73	Not connected					-
E6	47	74	VSS_2	S	-	VSS_2	-	-
F6	48	75	VDD_2	S	-	VDD_2	-	-
A9	49	76	PA14	I/O	FT	JTCK-SWCLK	-	PA14

- MCO 및 UART 사용을 위한 PORT/PIN 의 GPIO 설정이 필요함
- Datasheet의 Table 5. Pin definitions (p27~32) 참고
 - MCO 및 UART TX: Alternate function output Push-pull 으로 설정
 - UART RX: Input with pull-up / pull-down 으로 설정

Baud rate Example

Baud rate 계산 예제 ReferenceManual 에 있음

27.6.3 Baud rate register (USART_BRR)

Note: The baud counters stop counting if the TE or RE bits are disabled respectively.

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 Reserved, forced by hardware to 0.

Bits 15:4 **DIV_Mantissa[11:0]**: mantissa of USARTDIV
These 12 bits define the mantissa of the USART Divider (USARTDIV)

Bits 3:0 **DIV_Fraction[3:0]**: fraction of USARTDIV
These 4 bits define the fraction of the USART Divider (USARTDIV)

Fractional baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the Mantissa and Fraction values of USARTDIV.

$$\text{Tx/ Rx baud} = \frac{f_{\text{CK}}}{(16 \cdot \text{USARTDIV})}$$

legend: f_{CK} - Input clock to the peripheral (PCLK1 for USART2, 3, 4, 5 or PCLK2 for USART1)

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

The baud counters are updated with the new value of the Baud registers after a write to USART_BRR. Hence the Baud rate register value should not be changed during communication.

How to derive USARTDIV from USART_BRR register values

Example 1:

If DIV_Mantissa = 0d27 and DIV_Fraction = 0d12 (USART_BRR = 0x1BC), then

Mantissa (USARTDIV) = 0d27

Fraction (USARTDIV) = $12/16 = 0d0.75$

Therefore USARTDIV = 0d27.75

Example 2:

To program USARTDIV = 0d25.62

This leads to:

DIV_Fraction = $16 \cdot 0d0.62 = 0d9.92$

The nearest real number is 0d10 = 0xA

DIV_Mantissa = mantissa (0d25.620) = 0d25 = 0x19

Then, USART_BRR = 0x19A hence USARTDIV = 0d25.625

Example 3:

To program USARTDIV = 0d50.99

This leads to:

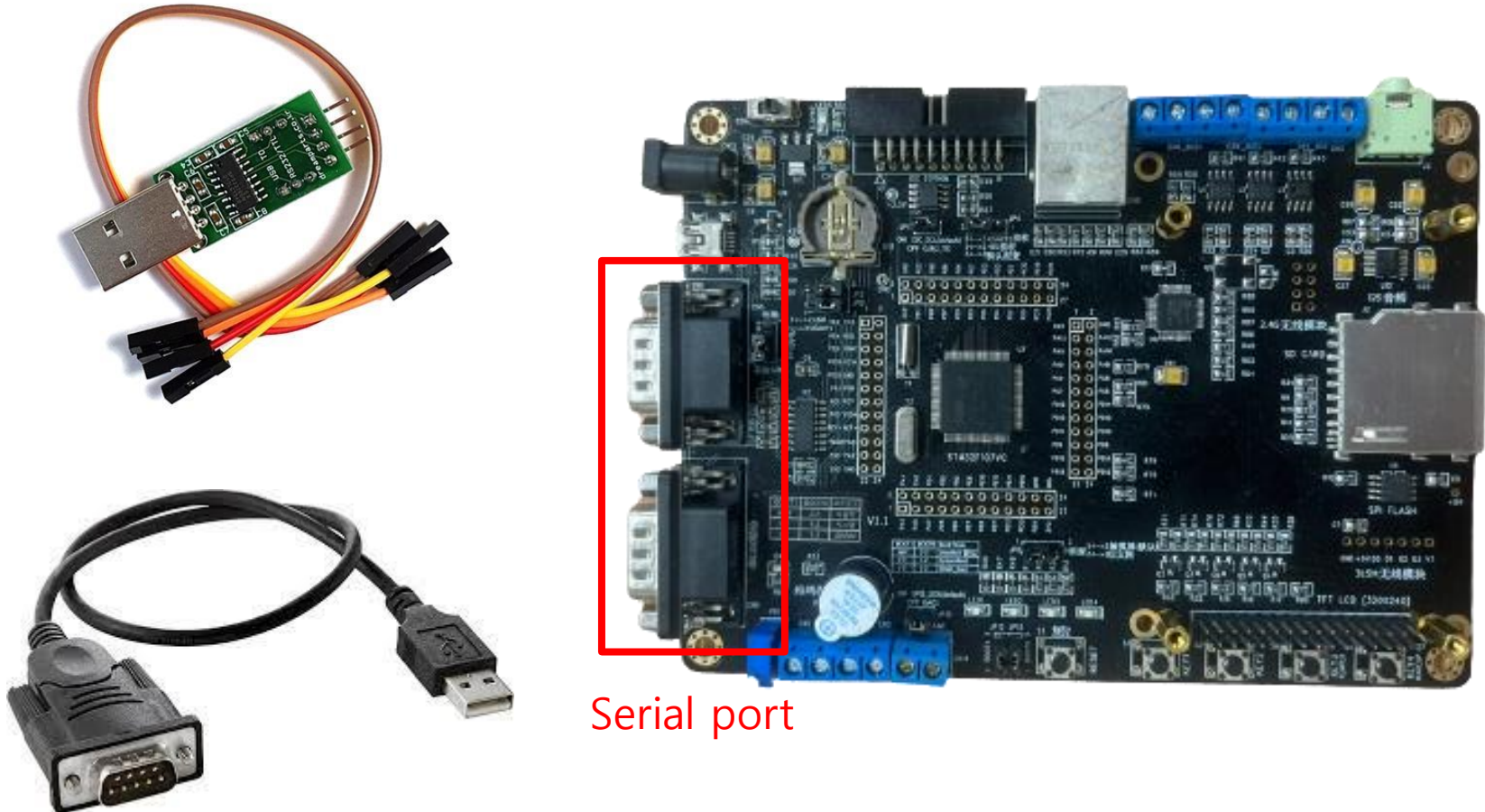
DIV_Fraction = $16 \cdot 0d0.99 = 0d15.84$

The nearest real number is 0d16 = 0x10 => overflow of DIV_frac[3:0] => carry must be added up to the mantissa

DIV_Mantissa = mantissa (0d50.990 + carry) = 0d51 = 0x33

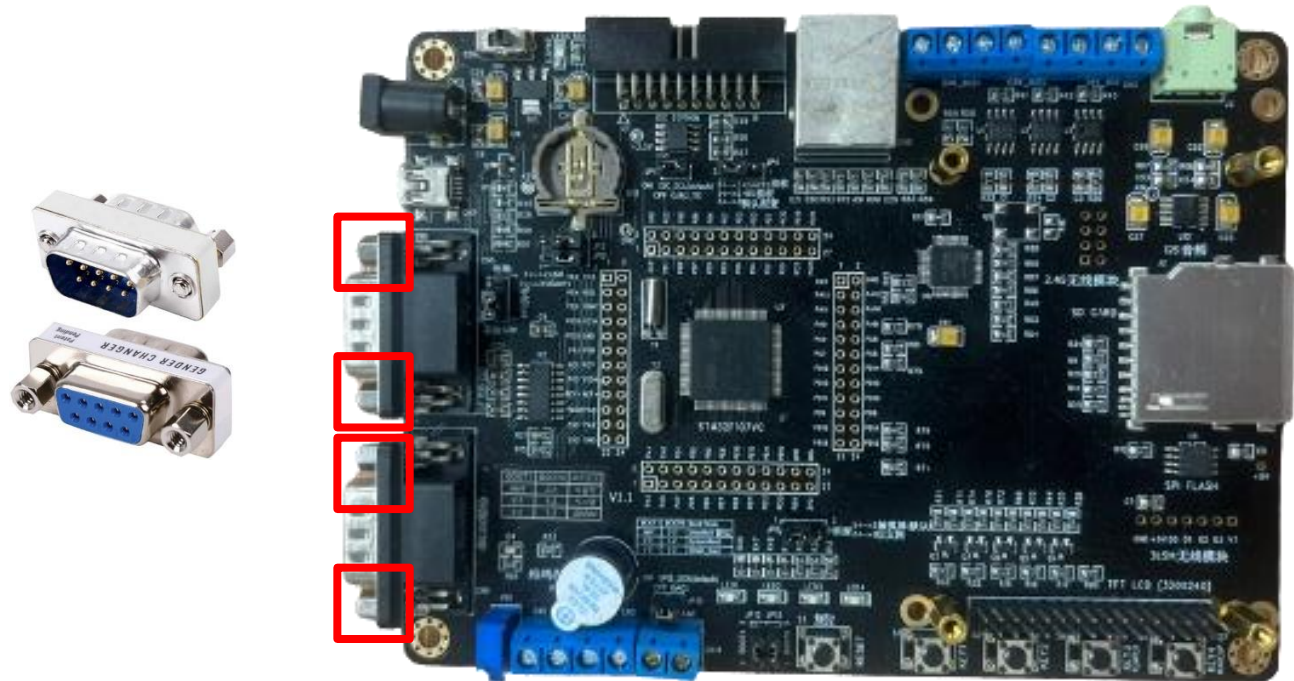
Then, USART_BRR = 0x330 hence USARTDIV = 0d51.000

Clock을 이용한 시리얼 통신



Serial port

해당 Serial Port를 PC와 연결해 Serial 통신
(Serial to USB cable 사용)

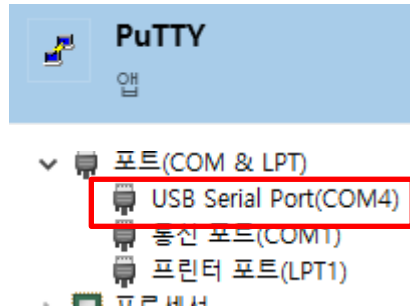


나사 탈착 후 커넥터 재결합

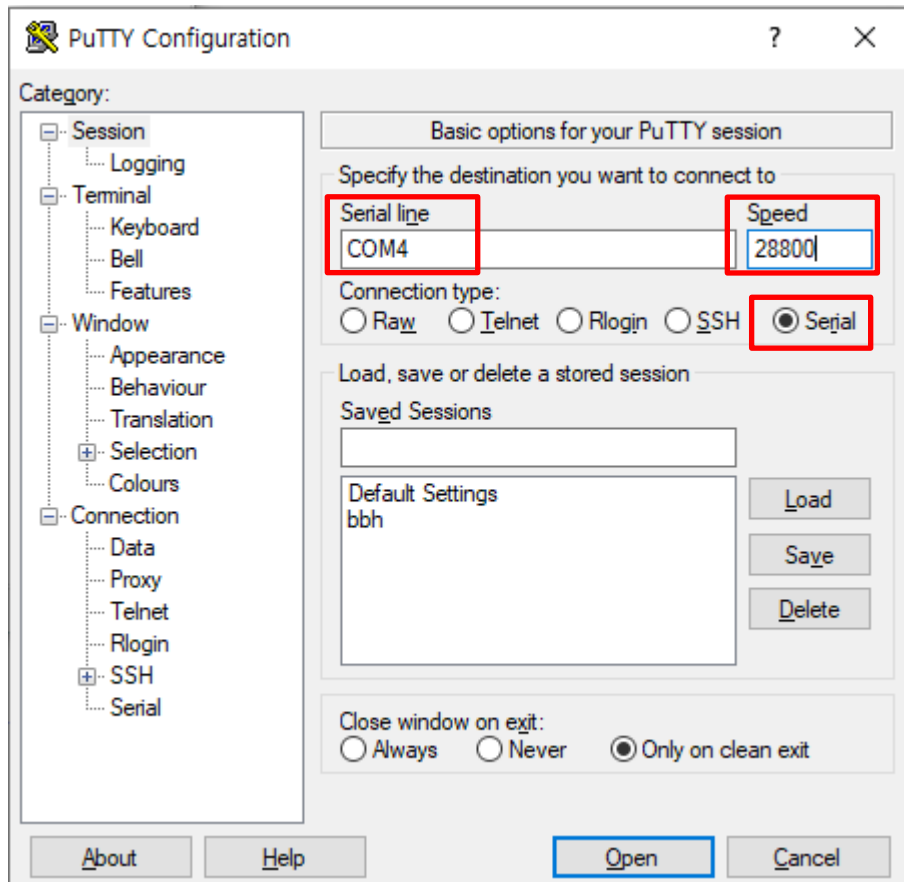
제공되는 보드의 Male/Female과 케이블의 Male/Female이 서로 일치하지 않음
보드의 Serial port를 살짝 풀어서 탈착 후 커넥터 재결합 필요

Clock을 이용한 시리얼 통신

PC에서 시리얼 통신을 확인할 때 쓰는 프로그램
PC에 없으면 인터넷에서 최신버전 설치하세요



PC 장치 관리자에서 보드와 연결된 Serial Port 확인

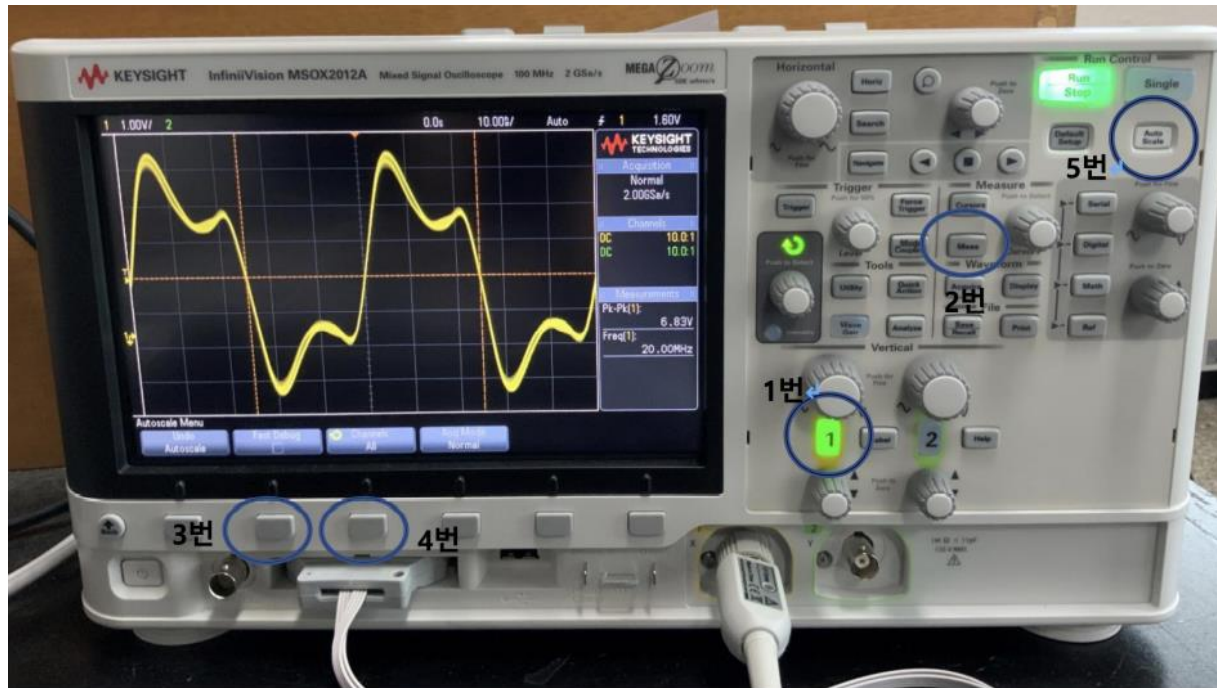


Connection type - Serial 선택 후
확인한 Port 입력, Baud rate 입력

Open

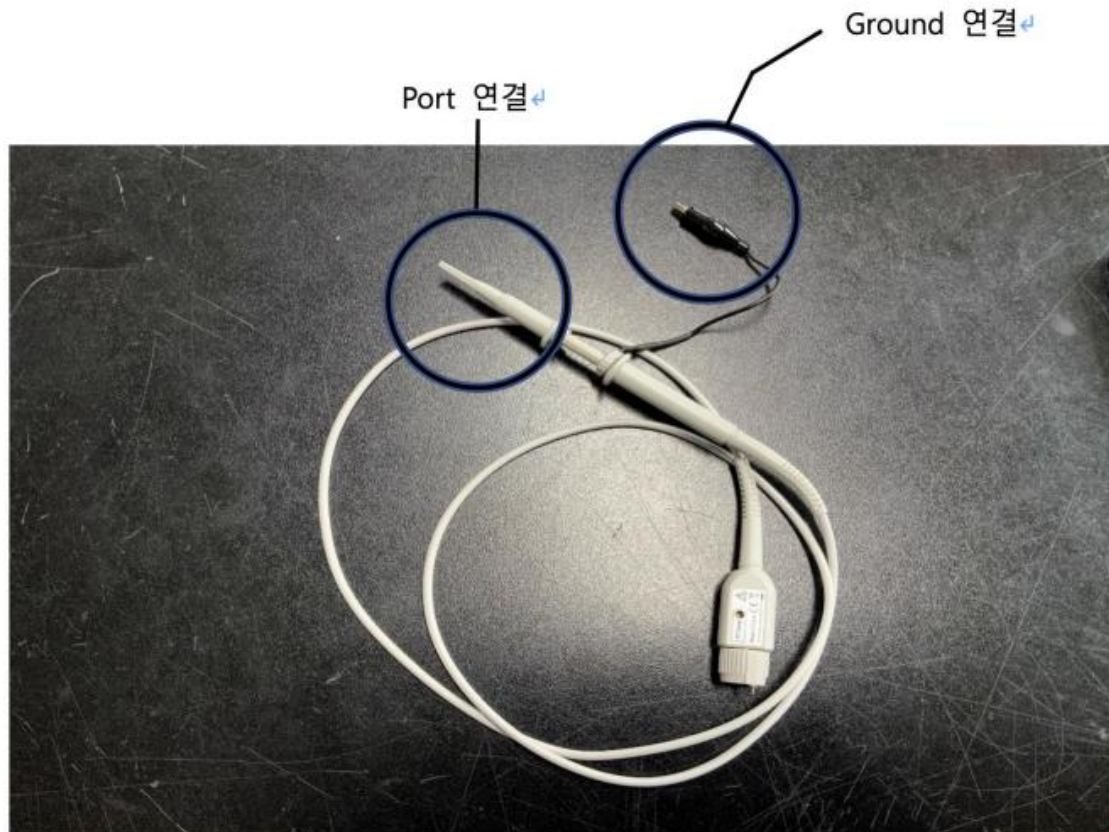


오실로스코프 아날로그 측정 방법



- 1번 : 신호를 Analog로 받기 위해 설정
- 2번 : Meas 버튼 눌러 측정 시작
- 3번 : frequency 모드로 변경
- 4번 : 화면에 frequency 출력
- 5번 : Auto scale 버튼 누르면 그래프가 깔끔해짐

오실로스코프 아날로그 측정 방법



레퍼런스 매뉴얼

~~Low-, medium-, high- and XLA connectivity reset and clock control (RCC)~~

Connectivity line devices: reset and clock control (RCC) .

- 실험 장비들을 연결 및 분리할 때 반드시 모든 전원을 끄고 연결해주세요.
 - 장비사용시 충격이 가해지지 않도록 주의해주세요.
 - 자리는 항상 깔끔하게 유지하고 반드시 정리 후 퇴실해주세요.
 - 실험 **소스 코드와 프로젝트 폴더**는 **백업** 후 반드시 **삭제**해주세요.
 - 장비 관리, 뒷정리가 제대로 되지 않을 경우 해당 조에게 감점이 주어집니다.
-
- **동작 중 케이블 절대 뽑지말것**
 - **보드는 전원으로 USBPort나 어댑터(5V,1A)를 사용할것 (5V 5A 어댑터(비슷하게 생김)와 혼동하지 말 것, 사용시 보드가 타버림 -> 감점)**
 - **디버깅 모드 중에 보드 전원을 끄거나 연결 케이블을 분리하지 말 것!!!**
-
- **-> 지켜지지 않을 시 해당 조 감점**

미션 ! 별도 미션지 참고

1. week06_template.c 에 TODO 부분 채워 넣기, 라이브러리 활용할 것
2. 각 조별 목적 Clock MCO로 출력 -> 오실로스코프로 Hz 검사
3. UART 통신을 이용하여 Putty로 출력
ex) user S1 Button 누르는 동안, "Hello TeamXXWrWn"

미션 힌트

$$28 = 25 / 5 * 4 / 5 * 7$$
$$52 = 25 / 5 * 13 / 5 * 4$$

Wr, Wn 둘 다 입력해야 line이 바뀝니다

실험 검사

오늘 검사 받을 수 있는 조는 오늘 받고 못 받는 조는 따로 미션 수행 후 다음 주 수업 시작할 때 검사
이번 주 실험 결과 보고서 및 소스 코드 및 실험 동작 영상

- A. 이론부터 실습까지 전반적인 내용을 포함하도록 작성 (실험 과정 사진 찍으시면 좋아요)
- B. 다음 실험시간 전까지 PLATO 제출
- C. 소스 코드는 직접 작성 및 수정한 파일만 제출

나가실 때, 만드신 코드 및 프로젝트 폴더는 모두 백업하시고 삭제해주세요.
다른 분반 파일은 만지지 마시고 조교에게 알려주세요.
자리 정리정돈 안 되어 있으면 **감점**합니다!!!