임베디드시스템 설계 및 실험 보고서(5주차)

8조

작성자: 201824483 박진영

조원:

201602181 정진성

201824523 안혜준

201924660 한병정

202055516 김명서

1. 실험 목표 및 기본 개념

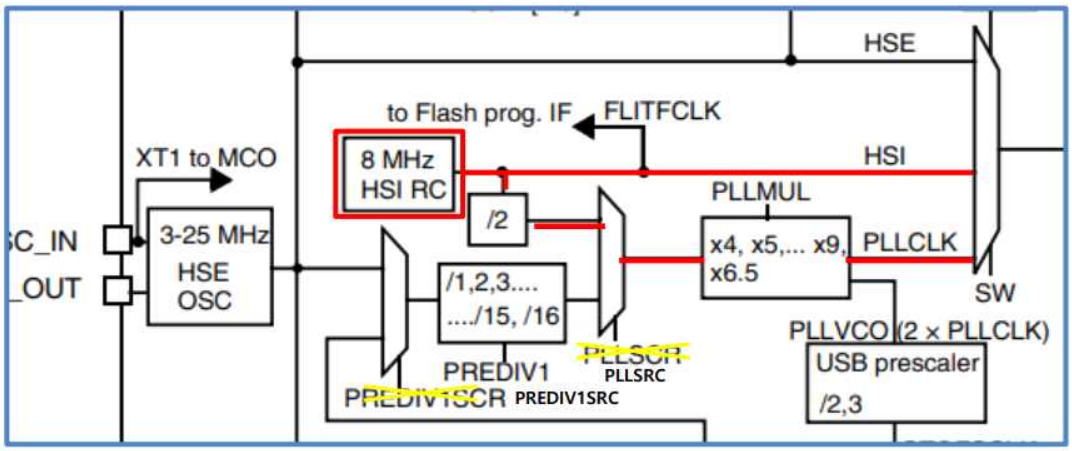
1.1 실험 목표

1. 라이브러리를 활용하여 코드 작성
2. Clock Tree 의 이해 및 사용자 Clock 설정
3. 오실로스코프를 이용한 clock 확인
4. UART 통신의 원리를 배우고 실제 설정 방법 파악

1.2 기본 개념

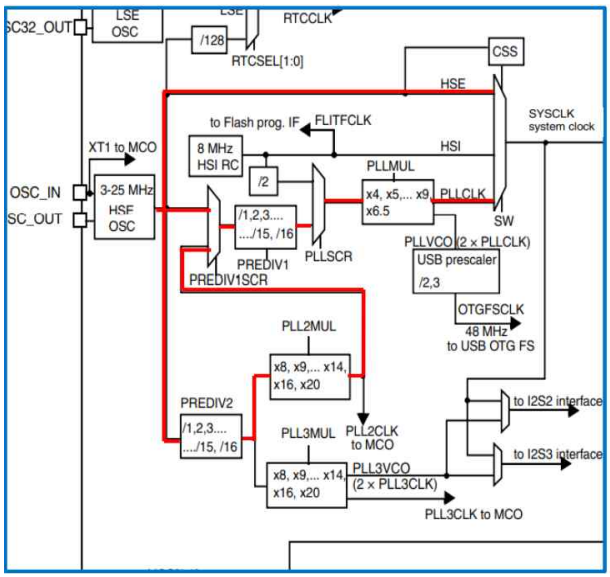
STM32 MCU 의 clock 은 SYSCLK 으로 나타나 있는 system clock 을 기반으로 결정된다. 우리가 사용하는 Cortex-M3 보드에서는 2가지 clock 이 발생된다.

HSI clock(High-speed internal) : 보드의 내부 클락으로 8Mhz 의 값을 가진다.

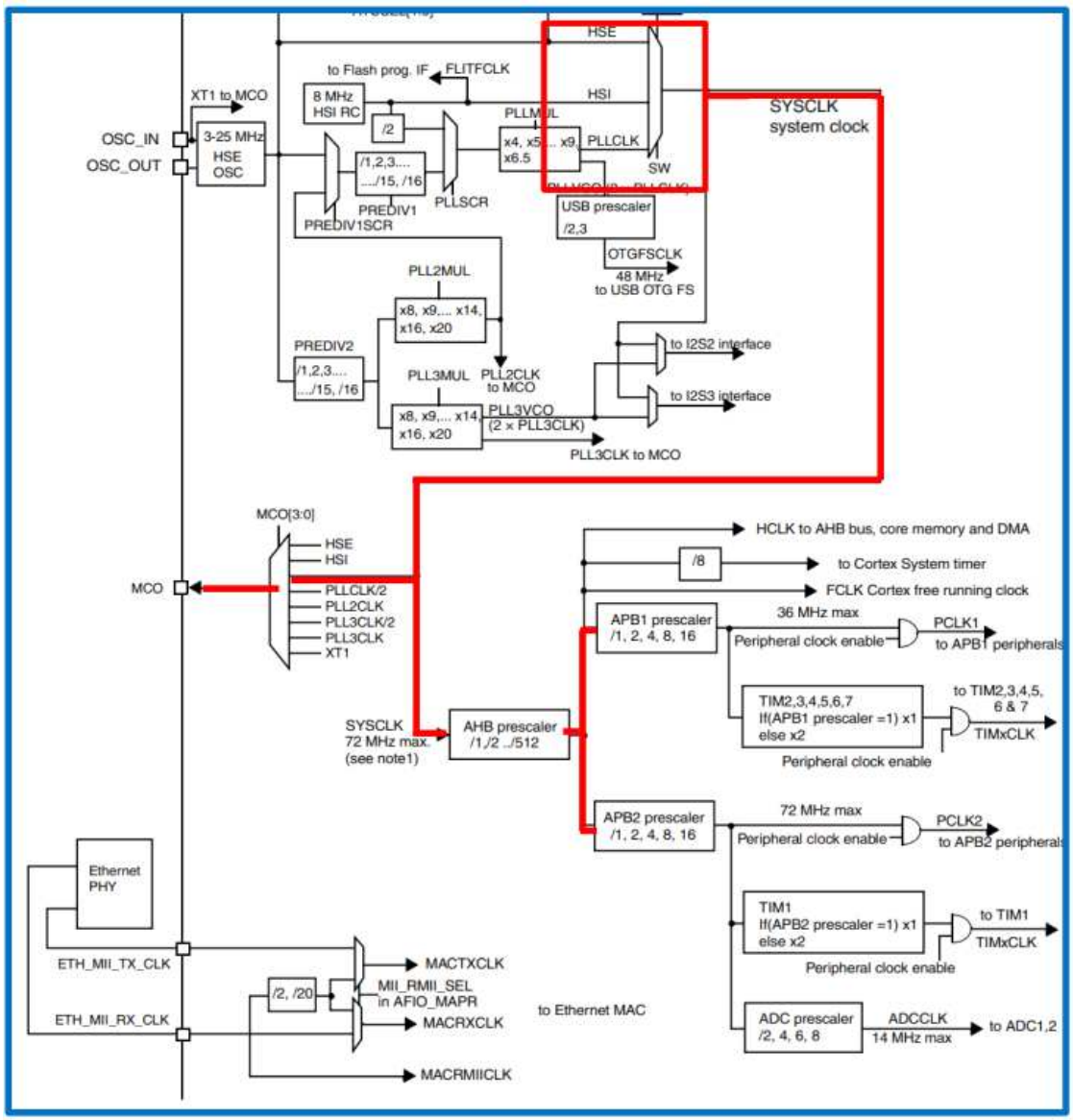


HSE clock (High-speed external)

: Cortex-M3 보드의 Oscillator에서 방생되는 clock으로 25Mhz 의 값을 가짐

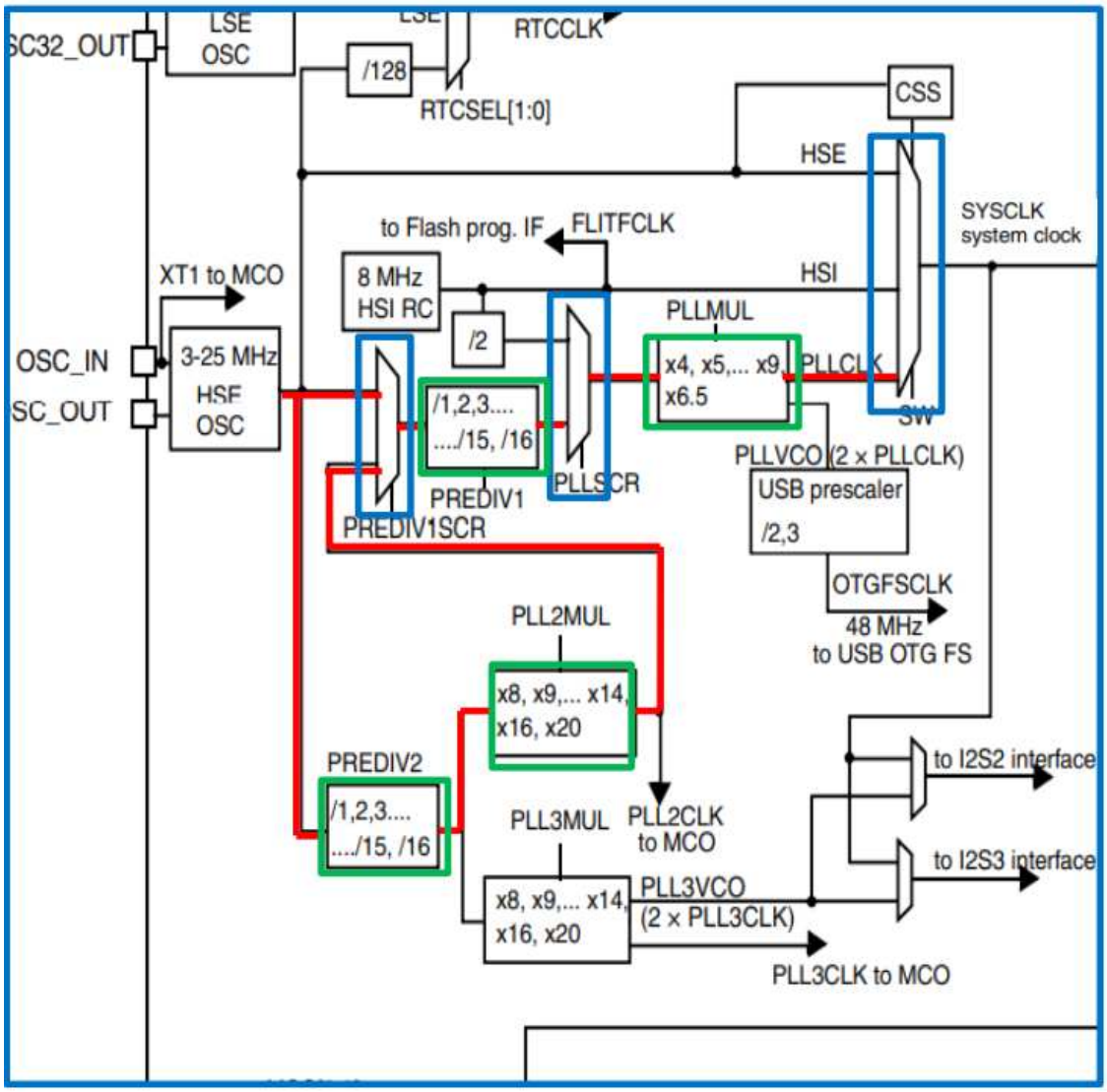


Clock tree



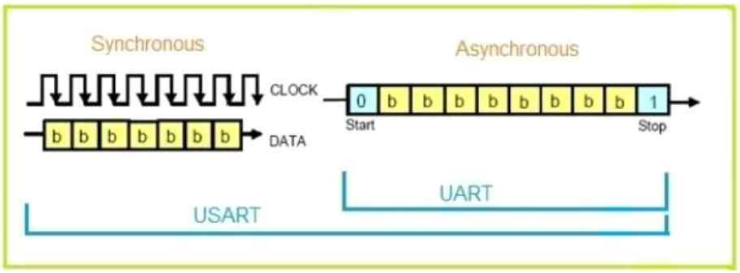
- HSI, HSE, PLL 중 SW MUX에 의해 시스템 클럭(최대72MHz)으로 설정될 수 있다.
- 시스템 클럭은 APB1, APB2에 전달된다.
- 시스템 클럭은 MCO MUX를 통해서 MCO에 출력해 오실로스코프로 확인이 가능함.

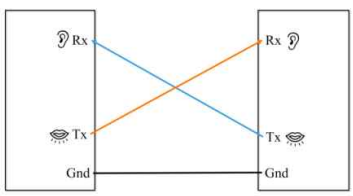
PLL(Phase-Locked Loop)



• PLL은 위상 동기 회로이며, 입력 신호와 출력신호를 이용해 출력신호를 제어한다.
• 입력된 신호에 맞추어 출력 신호의 주파수 조절이 목적이다.

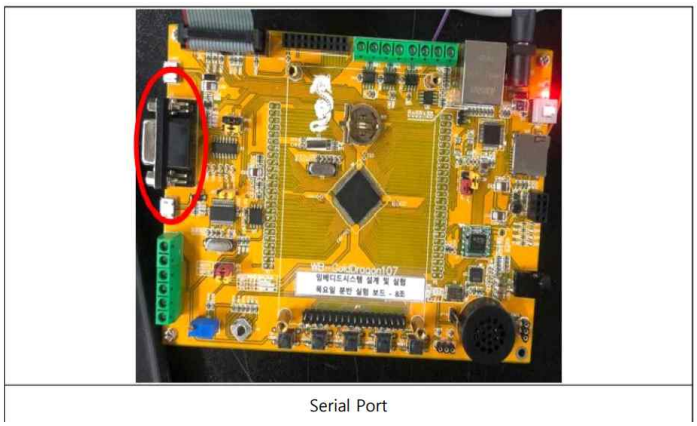
Universal Asynchronous Receiver/Transmitter (UART)





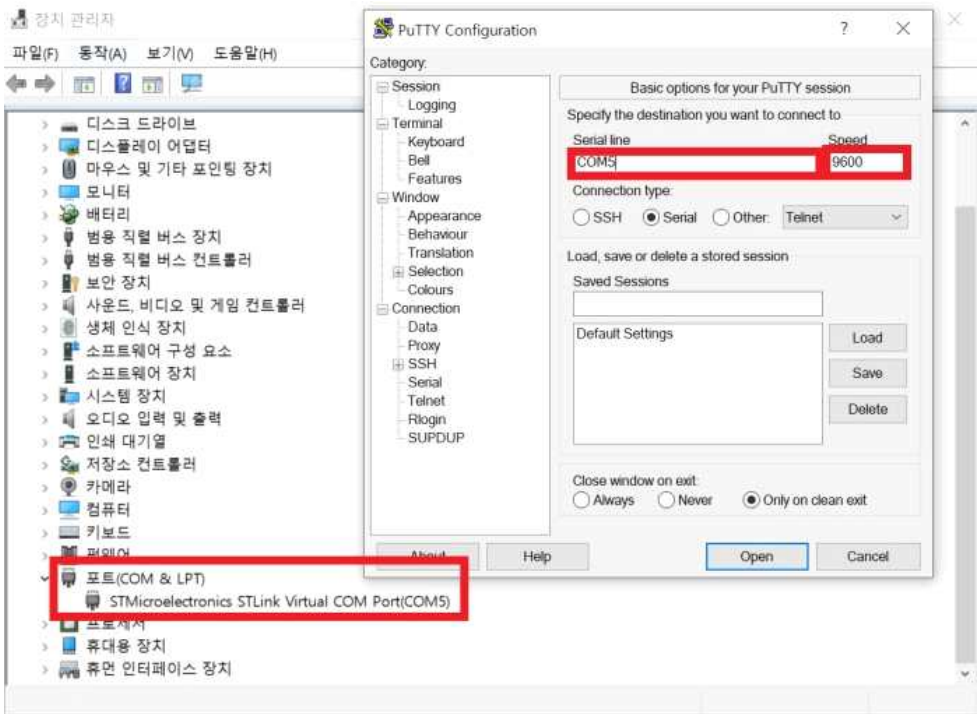
• 비동기 통신 프로토콜
• Rx(데이터 수신)와 Tx(데이터 송신) 교차연결
• 비동기 통신이기 때문에 Baud Rate를 일치 시켜야 한다.
• Baud rate는 초당 얼마나 많은 심볼(Symbol, 의미 있는 데이터 묶음)을 전송할 수 있는가를 나타낸다.

이번 실험에서 시리얼 통신을 하기 위한 시리얼 포트이다. 이 포트를 PC와 연결하여 시리얼 통신을 할 수 있다.



Serial Port

시리얼 통신을 하기 위해서 Putty 라는 프로그램을 사용하였다. connection type을 serial로 선택하고 컴퓨터에서 시리얼 포트를 확인하여 입력하고 Baud rate를 입력하면 시리얼 통신이 가능하다.



이번 실험에서는 오실로스코프를 이용해서 System clock을 확인한다. 오실로스코프를 사용하는 방법은

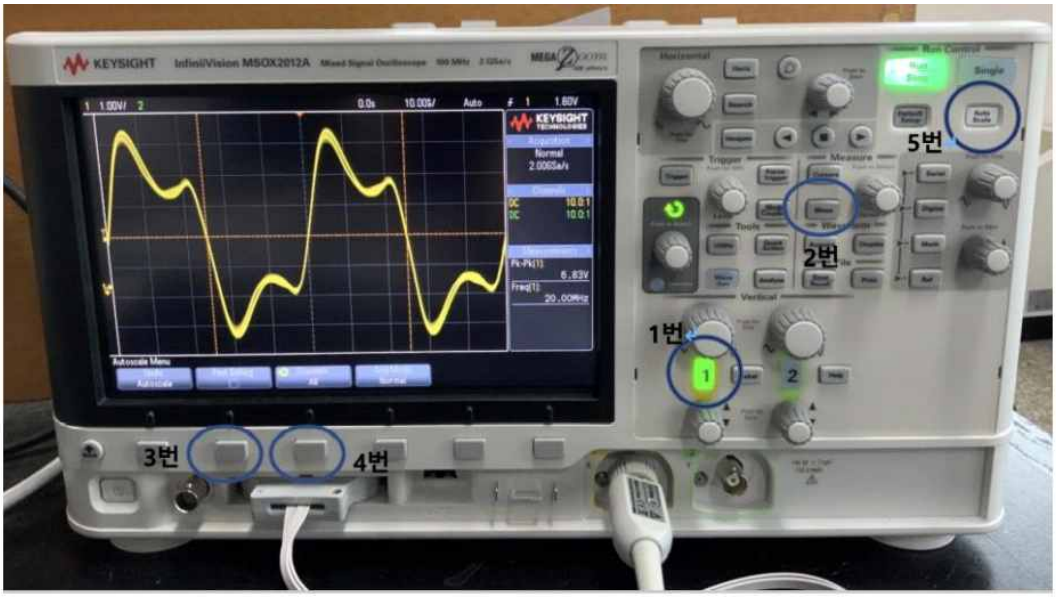
1번 : 신호를 Analog로 받기 위해 설정

2번 : Meas 버튼 눌러 측정 시작

3번 : frequency 모드로 변경

4번 : 화면에 frequency 출력

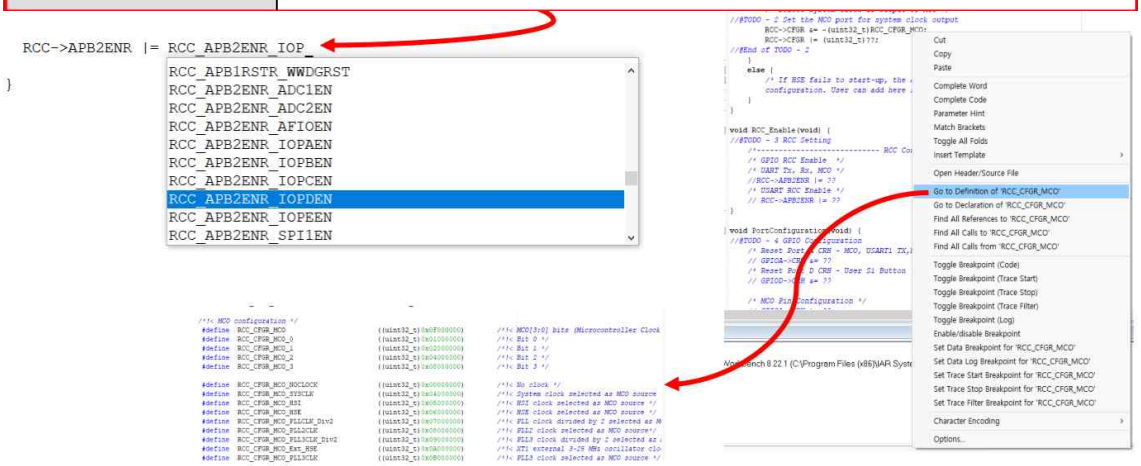
5번 : Auto scale 버튼을 누르면 그래프가 깔끔해진다.



2. 실험 과정

이전 실험까지는 직접 주소로 접근하여 프로그래밍을 진행하였으나 이번 실험부터는 라이브러리를 활용하여 정의된 주솟값을 사용하여 프로그래밍을 진행하였다.

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;



프로그래밍시에 위의 사진과 같이 라이브러리 목록과 “stm32f10x.h”또는 다른 라이브러리 파일에서 라이브러리에 대한 정의를 확인할 수 있다.

2.1. 레지스터 초기화

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		PLL3 RDY	PLL3 ON	PLL2 RDY	PLL2 ON	PLLRD Y	PLLON	Reserved				CSSON	HSEBY P	HSERDY	HSEON
		r	rw	r	rw	r	rw					rw	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSIRDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

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
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC PRE[1:0]		PPRE2[2:0]			PPRE1[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

RCC_CR 에서는 HSI를 Enable 상태, HSE,PLL,CSS를 OFF 상태로 초기화.

RCC_CFGR 에서는 HSI와 HSI 오실레이터를 system clock으로 선택.

곱셈과 나눗셈 연산이 일어나지 않도록 설정.

MCO는 clock을 받지 않는 상태로 설정.

HSE 오실레이터가 BYPASS 하지않도록 설정.

PLL input clock을 HSI 오실레이터/2로 선택.

PLL2와 PLL3는 OFF 상태로 설정.

```
void SysInit(void) {
    /* Set HSION bit */
    /* Internal Clock Enable */
    RCC->CR |= (uint32_t)0x00000001; //HSION

    /* Reset SW, HPRE, PPRE1, PPRE2, ADCPRE and MCO bits */
    RCC->CFGR &= (uint32_t)0xF0FF0000;

    /* Reset HSEON, CSSON and PLLON bits */
    RCC->CR &= (uint32_t)0xFEF6FFFF;

    /* Reset HSEBYP bit */
    RCC->CR &= (uint32_t)0xFFFBFFFF;

    /* Reset PLLSRC, PLLXTPRE, PLLMUL and USBPRE/OTGFSPRE bits */
    RCC->CFGR &= (uint32_t)0xFF80FFFF;

    /* Reset PLL2ON and PLL3ON bits */
    RCC->CR &= (uint32_t)0xEBFFFFFF;

    /* Disable all interrupts and clear pending bits  */
    RCC->CIR = 0x00FF0000;

    /* Reset CFGR2 register */
    RCC->CFGR2 = 0x00000000;
}
```

2.2. 클럭값 설정

SYSCLK	52MHz
PCLK2	26MHz

SYSCLK을 설정하기 위해서

1. PREDIV2 : /5
2. PLL2MUL : *13
3. PREDIV1 : /5
4. PLLMUL : *4
5. PLLCLK 값 선택하기

```
//@TODO - 1 Set the clock,
      /* HCLK = SYSCLK */
      RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;
      /* PCLK2 = HCLK / ?, use PPRE2 */
      RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE2_DIV2; // PCLK2 = SYSCLK / 2
      /* PCLK1 = HCLK */
      RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE1_DIV1;

      /* Configure PLLs ------------------------------------------------------*/
      RCC->CFGR &= (uint32_t)~( RCC_CFGR_PLLSRC | RCC_CFGR_PLLMULL);
      RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_PREDIV1 | RCC_CFGR_PLLMULL4);

      RCC->CFGR2 &= (uint32_t)~(RCC_CFGR2_PREDIV2 | RCC_CFGR2_PLL2MUL | RCC_CFGR2_PREDIV1 | RCC_CFGR2_PREDIV1SRC);
      RCC->CFGR2 |= (uint32_t)(RCC_CFGR2_PREDIV2_DIV5 | RCC_CFGR2_PLL2MUL13 | RCC_CFGR2_PREDIV1SRC_PLL2 | RCC_CFGR2_PREDIV1_DIV5);
      // 25 / 5(PREDIV2_DIV5) * 13(PLL2MUL13) -> 이제 선택 -> / 5(PREDIV1_DIV5) * 4(PLLMULL4) = 52
//@End of TODO - 1
```

2.3. MCO값 설정

다음으로 SYSCLK를 출력하기 위해 MCO값을 설정해야한다.
MCO값으로 SYSCLK을 선택하도록 설정한다.

```
//@TODO - 2 Set the MCO port for system clock output ★★★★★★★★★★★★★★★★★★★★
      RCC->CFGR &= ~(uint32_t)RCC_CFGR_MCO;
      RCC->CFGR |= (uint32_t)RCC_CFGR_MCO_SYSCLK; // MCO가 SYSCLK 값을 출력한다 (SYSCLK 값은 위에서 계산된 52Mhz 로 지정되있다)
//@End of TODO - 2 ★★★★★★★★★★★★★★★★★★★★
```

2.4. PORT/PIN GPIO 설정

MCO 및 UART 사용을 위해서 PORT/PIN의 GPIO 설정이 필요하다.
GPIO 와 USART RCC를 enable 상태로 설정한다.

```
void RCC_Enable(void) {
//@TODO - 3 RCC Setting ★★★★★★★★★★★★★★★★★★★★
   /*---------------------------- RCC Configuration -----------------------------*/
   /* GPIO RCC Enable  */
   /* UART Tx, Rx, MCO port */
   RCC->APB2ENR |= (RCC_APB2ENR_IOPAEN | RCC_APB2ENR_AFIOEN | RCC_APB2ENR_IOPDEN); // GPIOA, D 활성화, Alternative function enable
   /* USART RCC Enable */
   RCC->APB2ENR |= RCC_APB2ENR_USART1EN; // USART1  enable
}
```

MCO, USART1, TX, RX를 초기화한 다음 MCO 와 UART TX 는 Alternate function output Push-pull 으로 설정하고 UART RX
는 Input with pull-up/pull-down으로 설정한다.
USER S1 버튼 초기화 후 input mode로 설정한다.

```
//@TODO - 4 GPIO Configuration ★★★★★★★★★★★★★★★★★★★★
   /* Reset(Clear) Port A CRH - MCO, USART1 TX,RX*/
   GPIOA->CRH &= ~(
      (GPIO_CRH_CNF8 | GPIO_CRH_MODE8) |
      (GPIO_CRH_CNF9 | GPIO_CRH_MODE9) |
      (GPIO_CRH_CNF10 | GPIO_CRH_MODE10)
   );
   /* MCO Pin Configuration */
   GPIOA->CRH |= (GPIO_CRH_CNF8_1 | GPIO_CRH_MODE8); // MCO Pin의 CNF와 MODE를 1011 로 맞춘다. (OUTPUT, Alternative pull in - pull down)
   /* USART Pin Configuration */
   GPIOA->CRH |= (
            (GPIO_CRH_CNF9_1 | GPIO_CRH_MODE9) | // USART Tx 와 Rx 의 CNF와 MODE 를 설정한다 (1011, 1000)
            (GPIO_CRH_CNF10_1));

   /* Reset(Clear) Port D CRH - User S1 Button */
   GPIOD->CRH &= ~(GPIO_CRH_CNF11 | GPIO_CRH_MODE11);
   /* User S1 Button Configuration */
   GPIOD->CRH |= GPIO_CRH_CNF11_1; // S1 스위치를 input mode로 설정한다
```

2.5. USART 설정

USART_CR1 레지스터에서 먼저 USART Word Length, Parity, Parity, Mode를 초기화한 후 Word Length, PCE, PS와 Parity는 초기값이 설정값이랑 동일하므로 따로 설정하지 않는다. TE와 RE는 Enable 상태로 설정한다.

27.6.4 Control register 1 (USART_CR1)

Address offset: 0x0C

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
Reserved		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

```
  /*---------------------------- USART1 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 */
  // PCE와 PS는 disable 이므로 설정 필요 X (초기값 0)
//@TODO - 7: Parity : None ★★★★★★★★★★★★★★★★★★★★
  /* Set TE and RE bits according to USART_Mode value */
  // 설정 필요 X
//@TODO - 8: Enable Tx and Rx ★★★★★★★★★★★★★★★★★★★★
  USART1->CR1 |= (USART_CR1_TE | USART_CR1_RE); // Tx와 Rx를 enable 한다. [reference USART_CR1 참조]
```

USART_CR2 레지스터에서도 초기화를 진행하고 stop bit 는 이미 1bit로 설정되어 있기 때문에 따로 설정하지 않는다.

27.6.5 Control register 2 (USART_CR2)

Address offset: 0x10

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
Res.	LINEN	STOP[1:0]		CLK EN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

```
  /*---------------------------- 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 ★★★★★★★★★★★★★★★★★★★★
  // 이미 설정되있어서 설정 필요 X [reference USART_CR2 참조]
```

USART_CR3 에서도 초기화 진행 후 CTS 와 RTS를 disable 상태로 설정해야 하는데 초기값이 disable 상태이기 때문에 따로 설정하지 않는다.

```
  /*---------------------------- 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 ★★★★★★★★★★★★★★★★★★★★
  // PCE와 PS는 disable 이므로 설정 필요 X (초기값 0)
```

Baud rate 레지스터에서는 요구조건에 맞게 Baud rate를 설정해줬다.

27.6.6 Control register 3 (USART_CR3)

Address offset: 0x14

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
Reserved					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Baud rate 계산법은 아래 사진에 있는 식을 사용하여 계산하였다.
제시된 baud rate는 9600이므로 아래 식에 따라서 USARTDIV를 구해주면
26,000,000 / (16 * 9600) ~= 169.27이고
밑의 Example2 에 따라 Integer part와 Fractional part를 구해주면
DIV_Mantissa = 0d169 = 0xA9 이고
DIV_Fraction = 16*0d0.27 = 0d4.32 반올림하여 0d4 = 0x4이다
따라서 USART_BRR = 0xA94이다.

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]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

```
  /*---------------------------- USART BRR Configuration -----------------------*/
  /* Configure the USART Baud Rate -----------------------------------------------*/
  /* Determine the integer part */
  /* Determine the fractional part */
//@TODO - 11: Calculate & configure BRR ★★★★★★★★★★★★★★★★★★★★★
  USART1->BRR |= 0xA94; // 26000000/(16*9600) 해서 여러 과정을 거친 뒤 구한것.
```

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.

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

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*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*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

마지막으로 USART를 Enable 해준다.

```
  /*---------------------------- USART Enable ----------------------------------*/
  /* USART Enable Configuration */
//@TODO - 12: Enable UART (UE) ★★★★★★★★★★★★★★★★★★★★★
  USART1->CR1 |= USART_CR1_UE; // UE를 활성화한다.
}
```

2.6. 버튼 설정

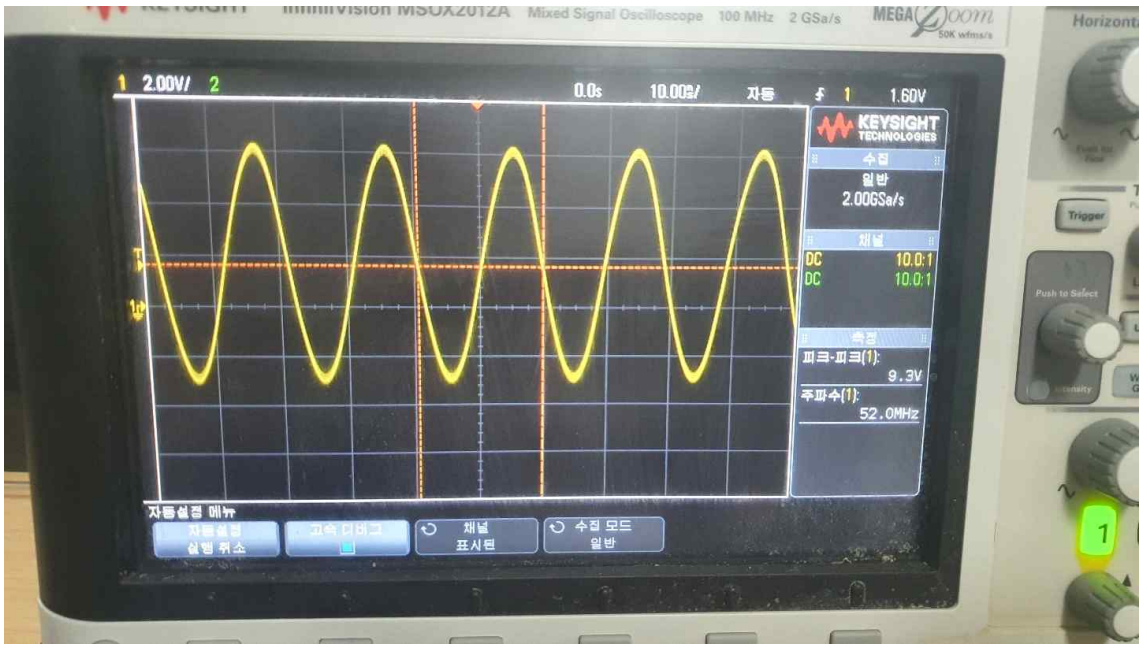
GPIO port A를 enable 상태로 바꾼 뒤에 check_bit 함수에서 GPIOD_IDR의 상태를 읽고 1에서 0으로 바뀌었다면 메시지를 전송한다. 전송시에 msg의 길이만큼 SendData 함수를 호출하여 글자를 출력해준다.

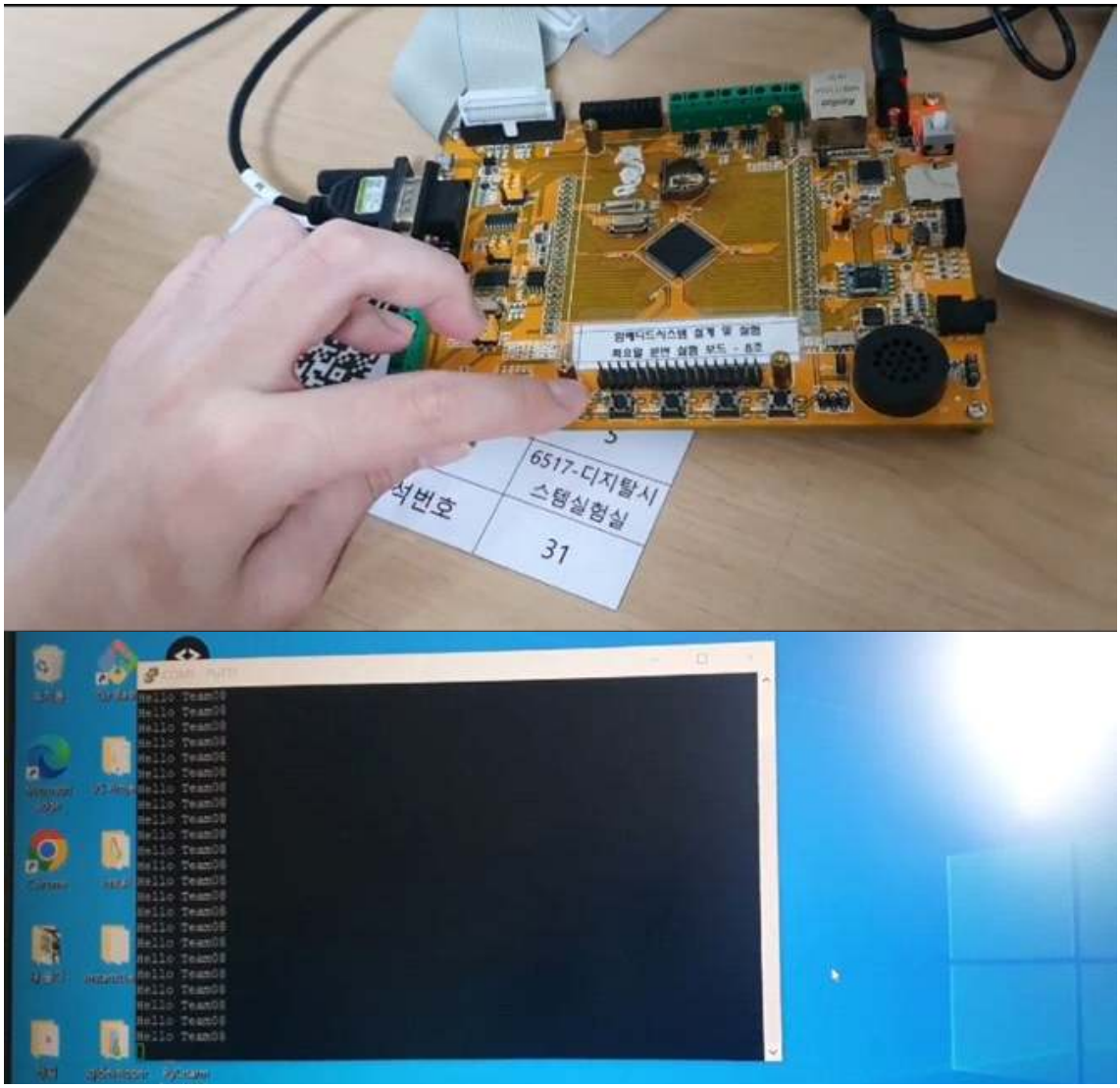
```
int main() {
  int i;
  char msg[] = "Hello Team08₩r₩n";

  RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;


  SysInit();
  SetSysClock();
  RCC_Enable();
  PortConfiguration();
  UartInit();

  // if you need, init pin values here


  while (1) {
    //@TODO - 13: Send the message when button is pressed ★★★★★★★★★★★★★★★★★★★★★★
    if (check_bit(GPIOD_IDR, 11) == 0) { // GPIOD_IDR 이 1에서 0으로 바뀌었는지 확인 후 바뀌었다면 메세지 전송
      for(int i = 0; i < 14; i++) {
        SendData(msg[i]);
      }
    }
  }

}// end main
```

3. 실험 결과

오실로스코프 상에서 MCO를 통해서 나오는 System Clock 수치를 확인했을 때 요구사항인 52MHz가 출력되는 것을 확인할 수 있었고

User S1 버튼을 누르는 동안 터미널 프로그램을 통해서 “Hello Team08”이 출력되는 것을 확인 할 수 있었다.




6517-디지탈시
스템실험실
31