REPORT

제목 : 9조 6주차 실험결과 보고서

수강과목 :	임베디드 시스템 설계 및 실험	
조 원 :	201524414	기호영
	20152437	김요한
	201624410	권선근
	201824461	남지원
제출일자 :	2020.10.19	

1. 실험목표

A. Clock Tree의 이해 및 사용자 Clock 설정

B. UART 통신의 원리를 배우고, 실제 설정 방법 파악

2. 배경지식

A. Clock Tree

STM32의 내부 Clock의 흐름

HSI, HSE, PPL의 출력중 하나를 사용하여 System Clock을 출력

PLL은 HSI와 HSE를 곱하거나 나누어서 원하는 주파수를 출력

B. UART(Universal Asynchronous Receiver/Transmitter)

비동기 통신 프로토콜

Rx(데이터 수신)와 Tx(데이터 송신)의 교차 연결

서로의 Ground를 연결

baud rate를 일치시켜야 통신 가능

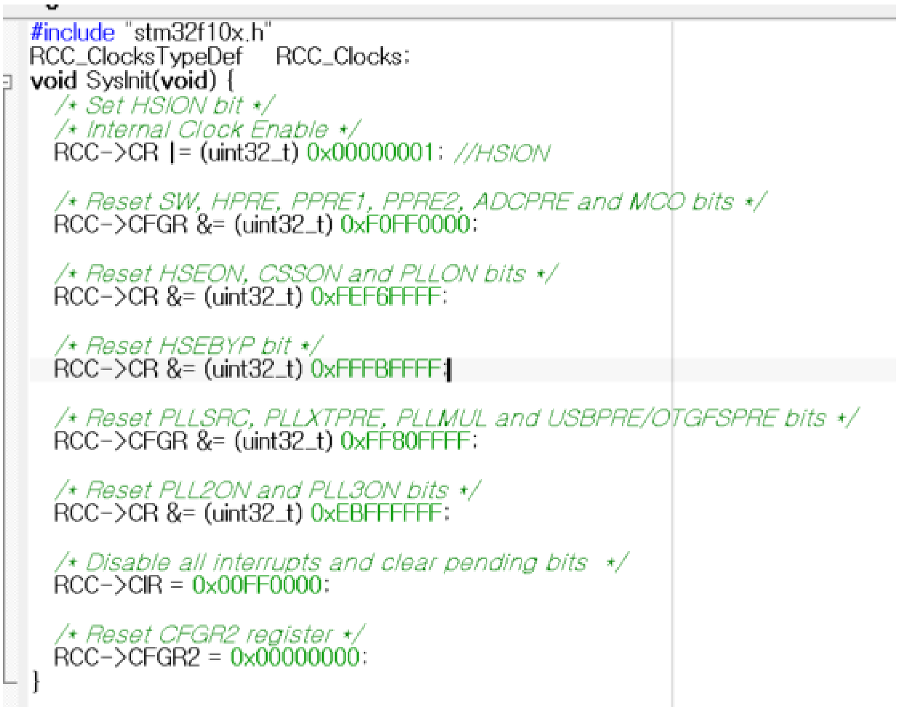
3. 실험방법

Clock 을 이용한 시리얼 통신

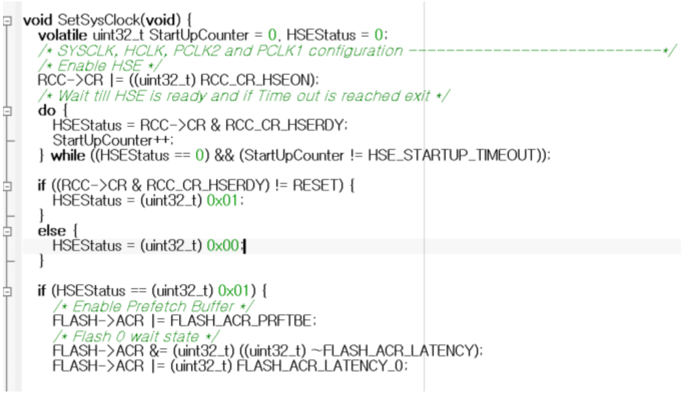
- PuTTY

PuTTY는 SSH, 텔넷, rlogin, raw TCP를 위한 클라이언트로 동작하는 자유 및 오픈 소스 단말 에뮬레이터 응용 프로그램이다. PuTTY라는 이름에는 특별한 뜻이 없으나 tty는 유닉스 전통의 터미널의 이름을 가리키며 teletype를 짧게 줄인 것이다.

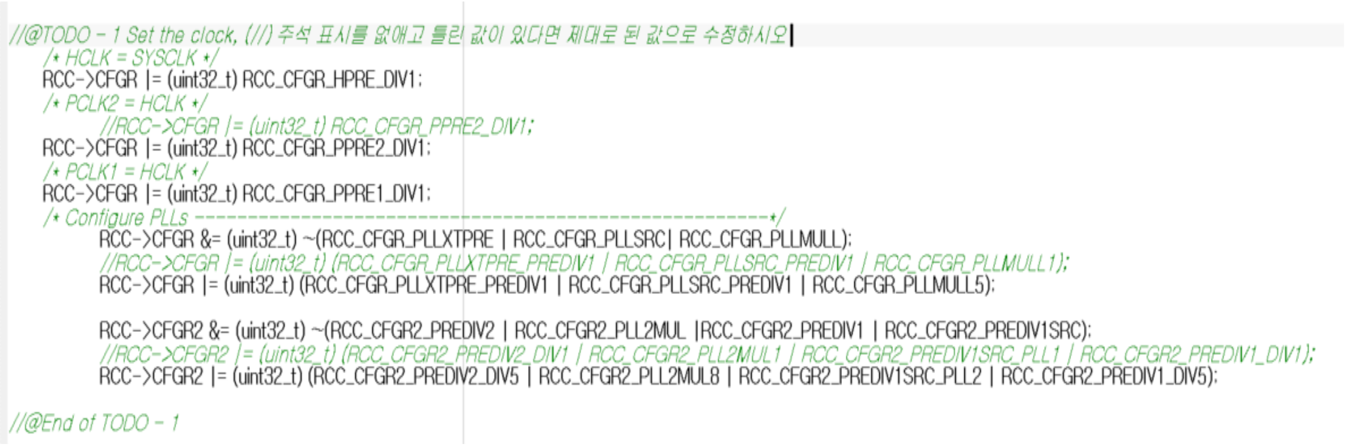
1) 주어진 Template.c 코드 작성
   todo 부분의 코드를 수정



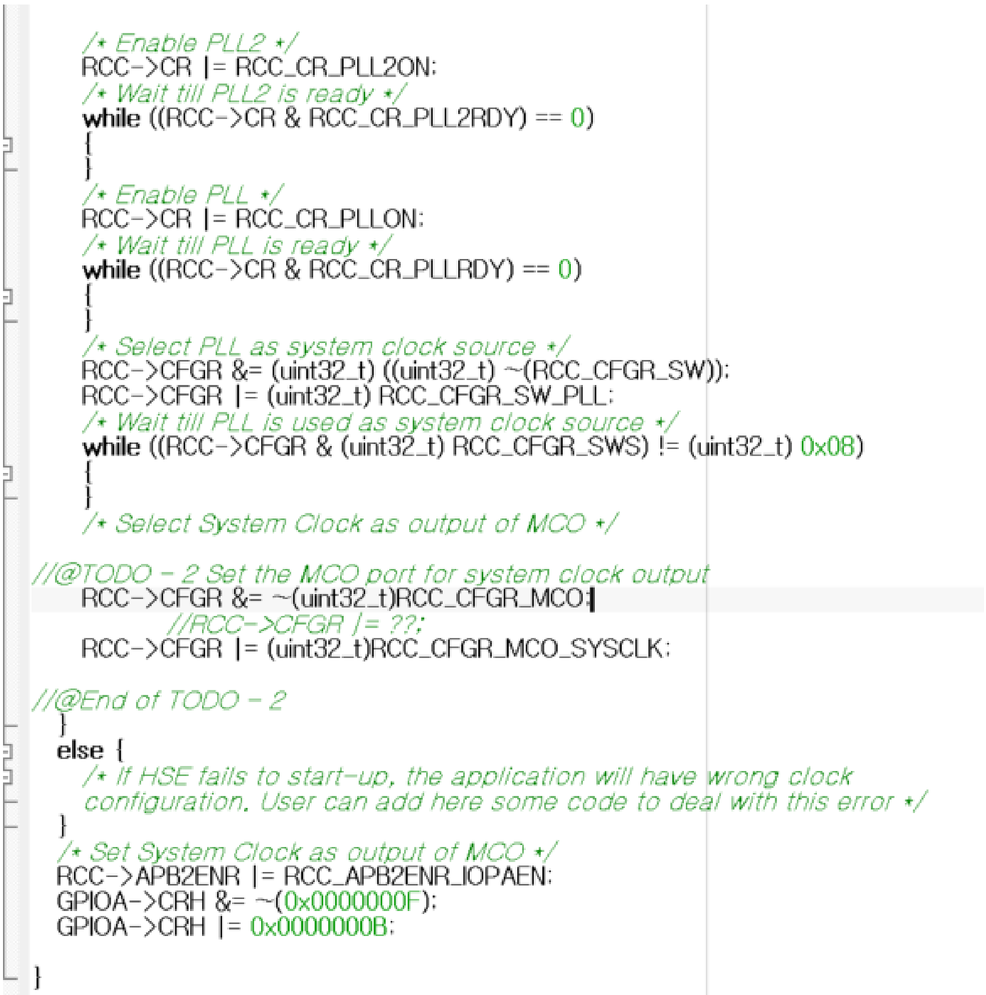
```c
#include "stm32f10x.h"
RCC_ClocksTypeDef    RCC_Clocks;
void SysInit(void) {
    /* Set HSION bit */
    /* Internal Clock Enable */
    RCC->CR |= (uint32_t) 0x00000001; //HSION

    /* 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;
}
```



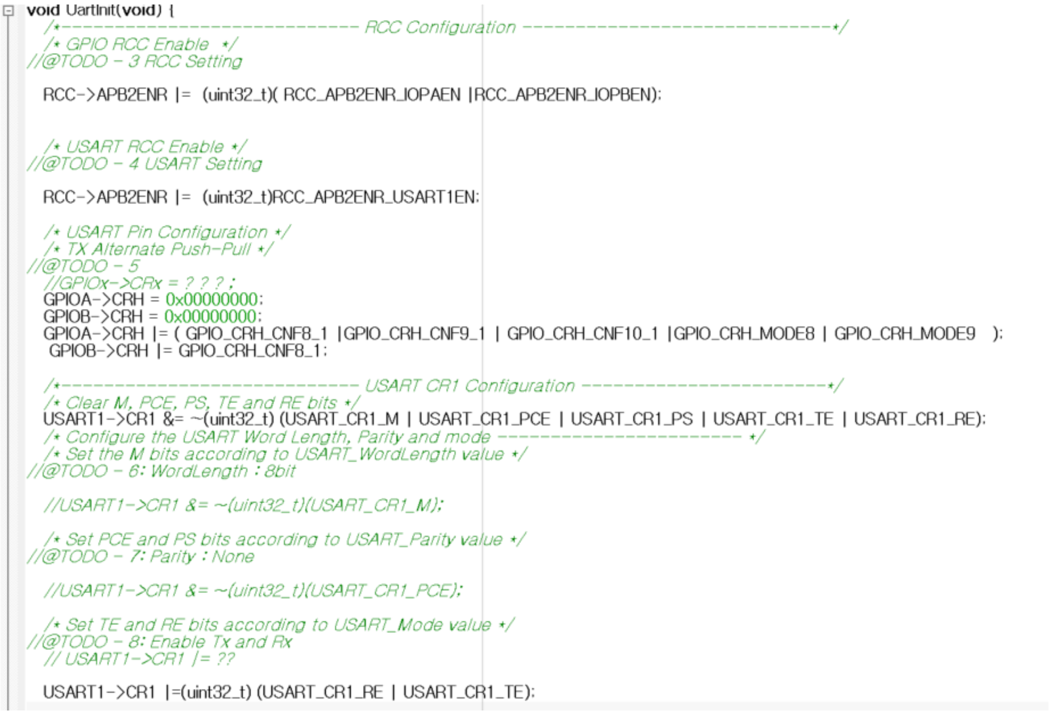
```c
void SetSysClock(void) {
   volatile uint32_t StartUpCounter = 0, HSEStatus = 0;
   /* SYSCLK, HCLK, PCLK2 and PCLK1 configuration ---------------------------*/
   /* Enable HSE */
   RCC->CR |= ((uint32_t) RCC_CR_HSEON);
   /* Wait till HSE is ready and if Time out is reached exit */
   do {
      HSEStatus = RCC->CR & RCC_CR_HSERDY;
      StartUpCounter++;
   } while ((HSEStatus == 0) && (StartUpCounter != HSE_STARTUP_TIMEOUT));

   if ((RCC->CR & RCC_CR_HSERDY) != RESET) {
      HSEStatus = (uint32_t) 0x01;
   }
   else {
      HSEStatus = (uint32_t) 0x00;
   }

   if (HSEStatus == (uint32_t) 0x01) {
      /* Enable Prefetch Buffer */
      FLASH->ACR |= FLASH_ACR_PRFTBE;
      /* Flash 0 wait state */
      FLASH->ACR &= (uint32_t) ((uint32_t) ~FLASH_ACR_LATENCY);
      FLASH->ACR |= (uint32_t) FLASH_ACR_LATENCY_0;
```



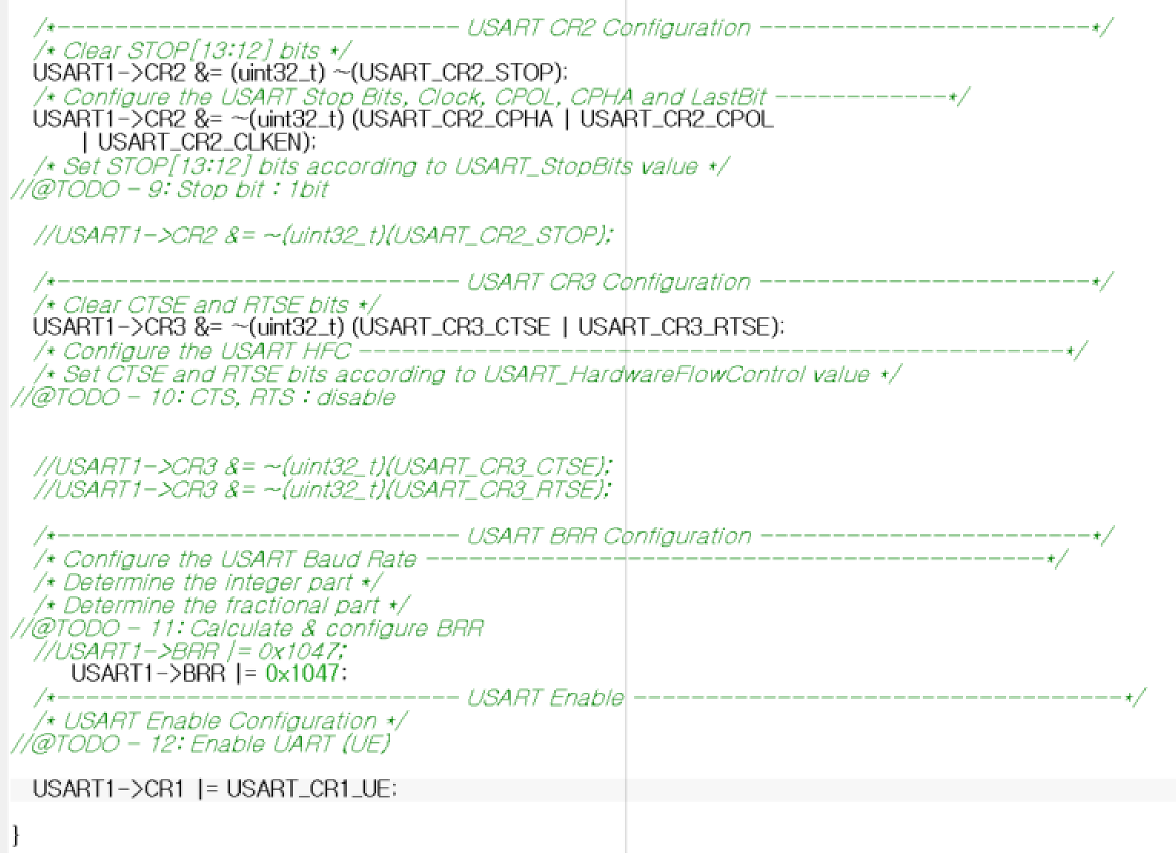
```
//@TODO - 1 Set the clock, (//) 주석 표시를 없애고 틀린 값이 있다면 제대로 된 값으로 수정하시오
    /* HCLK = SYSCLK */
    RCC->CFGR |= (uint32_t) RCC_CFGR_HPRE_DIV1;
    /* PCLK2 = HCLK */
        //RCC->CFGR |= (uint32_t) RCC_CFGR_PPRE2_DIV1;
    RCC->CFGR |= (uint32_t) RCC_CFGR_PPRE2_DIV1;
    /* PCLK1 = HCLK */
    RCC->CFGR |= (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->CFGR |= (uint32_t) (RCC_CFGR_PLLXTPRE_PREDIV1 | RCC_CFGR_PLLSRC_PREDIV1 | RCC_CFGR_PLLMULL5);

        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_PLL1 | RCC_CFGR2_PREDIV1_DIV1);
        RCC->CFGR2 |= (uint32_t) (RCC_CFGR2_PREDIV2_DIV5 | RCC_CFGR2_PLL2MUL8 | RCC_CFGR2_PREDIV1SRC_PLL2 | RCC_CFGR2_PREDIV1_DIV5);

//@End of TODO - 1
```



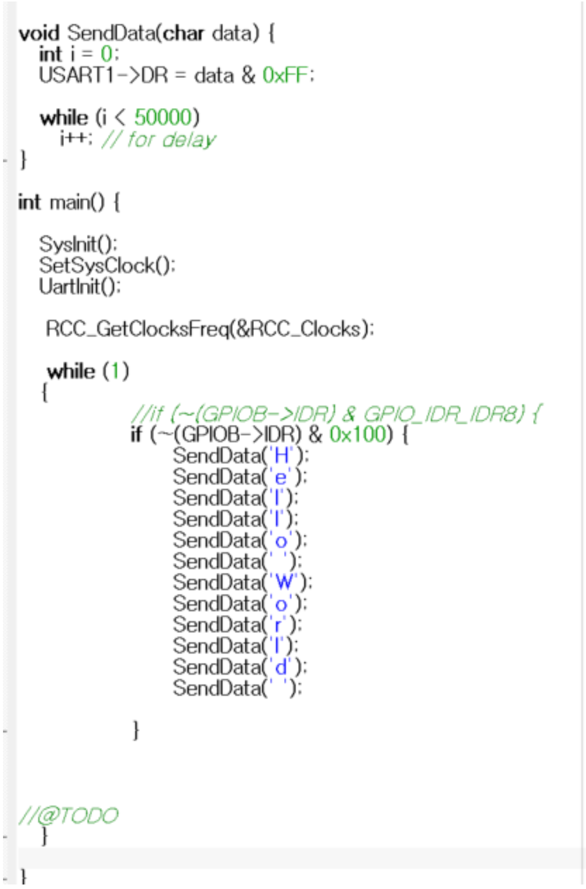
```
        /* Enable PLL2 */
        RCC->CR |= RCC_CR_PLL2ON;
        /* Wait till PLL2 is ready */
        while ((RCC->CR & RCC_CR_PLL2RDY) == 0)
        {
        }
        /* Enable PLL */
        RCC->CR |= RCC_CR_PLLON;
        /* Wait till PLL is ready */
        while ((RCC->CR & RCC_CR_PLLRDY) == 0)
        {
        }
        /* Select PLL as system clock source */
        RCC->CFGR &= (uint32_t) ((uint32_t) ~(RCC_CFGR_SW));
        RCC->CFGR |= (uint32_t) RCC_CFGR_SW_PLL;
        /* Wait till PLL is used as system clock source */
        while ((RCC->CFGR & (uint32_t) RCC_CFGR_SWS) != (uint32_t) 0x08)
        {
        }
        /* 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 |= ??;
        RCC->CFGR |= (uint32_t)RCC_CFGR_MCO_SYSCLK;

//@End of TODO - 2
    }
    else {
        /* If HSE fails to start-up, the application will have wrong clock
        configuration. User can add here some code to deal with this error */
    }
    /* Set System Clock as output of MCO */
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
    GPIOA->CRH &= ~(0x0000000F);
    GPIOA->CRH |= 0x0000000B;

}
```



```
void UartInit(void) {
  /*---------------------------- RCC Configuration -----------------------------*/
  /* GPIO RCC Enable  */
//@TODO - 3 RCC Setting

  RCC->APB2ENR |=  (uint32_t)( RCC_APB2ENR_IOPAEN |RCC_APB2ENR_IOPBEN);


  /* USART RCC Enable */
//@TODO - 4 USART Setting

  RCC->APB2ENR |=  (uint32_t)RCC_APB2ENR_USART1EN;

  /* USART Pin Configuration */
  /* TX Alternate Push-Pull */
//@TODO - 5
  //GPIOx->CRx = ? ? ? ;
  GPIOA->CRH = 0x00000000;
  GPIOB->CRH = 0x00000000;
  GPIOA->CRH |= ( GPIO_CRH_CNF8_1 |GPIO_CRH_CNF9_1 | GPIO_CRH_CNF10_1 |GPIO_CRH_MODE8 | GPIO_CRH_MODE9   );
   GPIOB->CRH |= GPIO_CRH_CNF8_1;

  /*---------------------------- 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

  //USART1->CR1 &= ~(uint32_t)(USART_CR1_M);

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

  //USART1->CR1 &= ~(uint32_t)(USART_CR1_PCE);

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

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



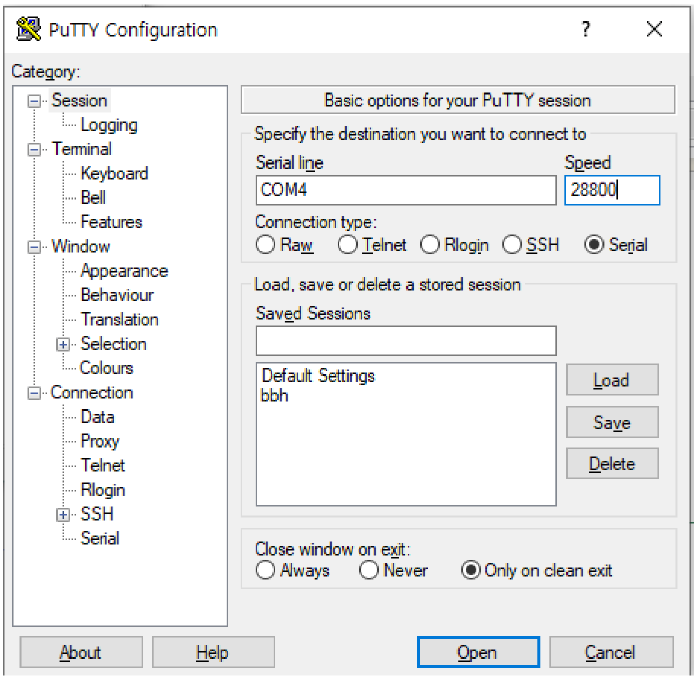
```
  /*---------------------------- USART CR2 Configuration ----------------------*/
  /* Clear STOP[13:12] bits */
  USART1->CR2 &= (uint32_t) ~(USART_CR2_STOP);
  /* Configure the USART Stop Bits, Clock, CPOL, CPHA and LastBit ------------*/
  USART1->CR2 &= ~(uint32_t) (USART_CR2_CPHA | USART_CR2_CPOL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

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

  /*---------------------------- 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


  //USART1->CR3 &= ~(uint32_t)(USART_CR3_CTSE);
  //USART1->CR3 &= ~(uint32_t)(USART_CR3_RTSE);

  /*---------------------------- USART BRR Configuration ----------------------*/
  /* Configure the USART Baud Rate -------------------------------------------*/
  /* Determine the integer part */
  /* Determine the fractional part */
//@TODO - 11: Calculate & configure BRR
  //USART1->BRR |= 0x1047;
      USART1->BRR |= 0x1047;
  /*---------------------------- USART Enable ---------------------------------*/
  /* USART Enable Configuration */
//@TODO - 12: Enable UART (UE)

  USART1->CR1 |= USART_CR1_UE;

}
```



```
void SendData(char data) {
  int i = 0;
  USART1->DR = data & 0xFF;

  while (i < 50000)
    i++; // for delay
}

int main() {

  SysInit();
  SetSysClock();
  UartInit();

   RCC_GetClocksFreq(&RCC_Clocks);

   while (1)
  {
          //if (~(GPIOB->IDR) & GPIO_IDR_IDR8) {
          if (~(GPIOB->IDR) & 0x100) {
              SendData('H');
              SendData('e');
              SendData('l');
              SendData('l');
              SendData('o');
              SendData(' ');
              SendData('W');
              SendData('o');
              SendData('r');
              SendData('l');
              SendData('d');
              SendData(' ');

          }



//@TODO
  }

}
```

2) PuTTY 실행

pc 장치 관리자에 보드와 연결된 serial port를 확인하고 Connection Type을 serial로 선택하고 port와 Baud rate를 입력 후 Open을 클릭



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM4

Speed: 28800

Connection type: ○ Raw ○ Telnet ○ Rlogin ○ SSH ● Serial

Load, save or delete a stored session

Saved Sessions

Default Settings
bbh

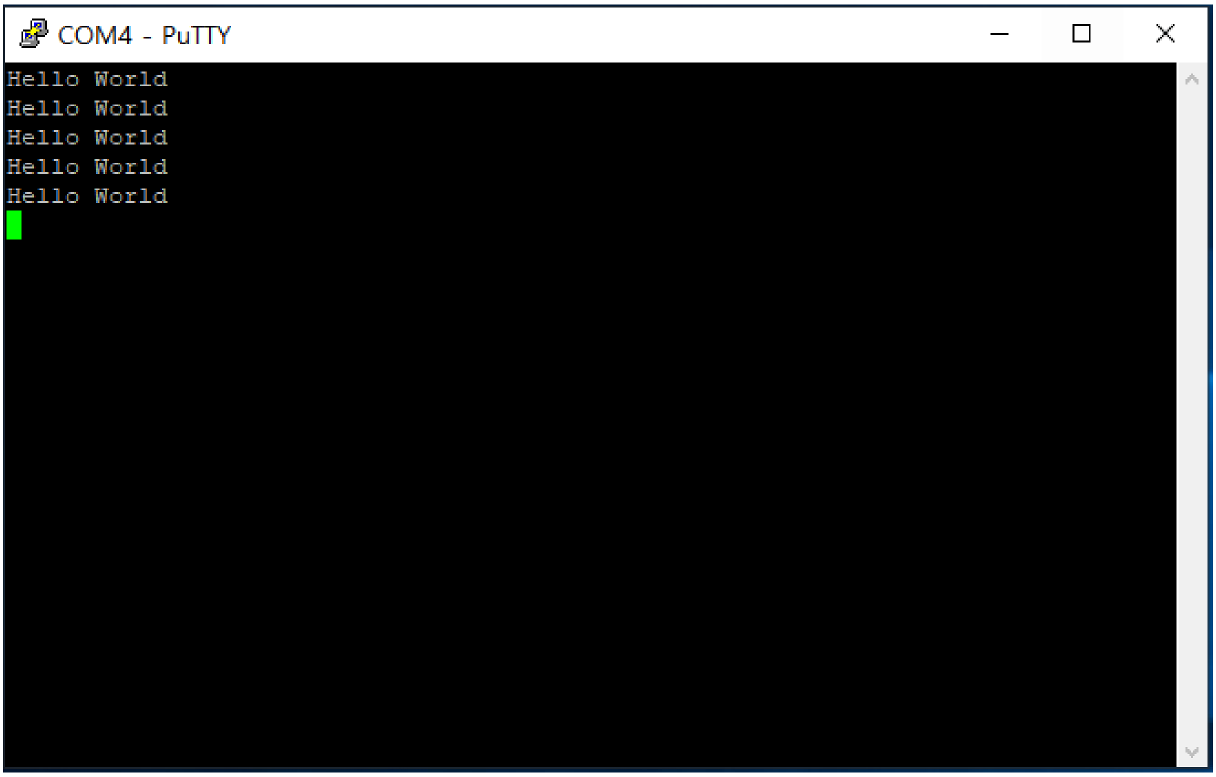
Close window on exit: ○ Always ○ Never ● Only on clean exit

4. 실험 결과

A. 결과

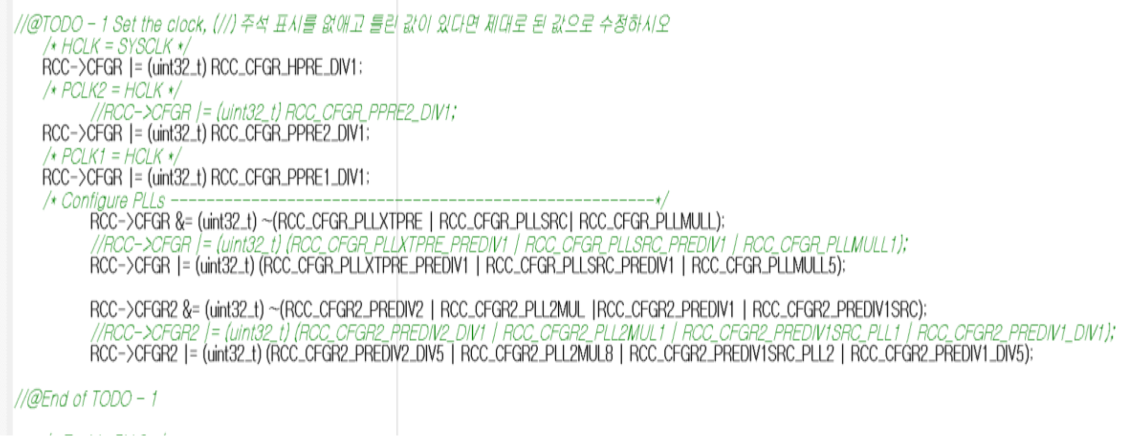
버튼을 누를 때마다 Putty를 통해 Hello world을 출력.

전체 화면

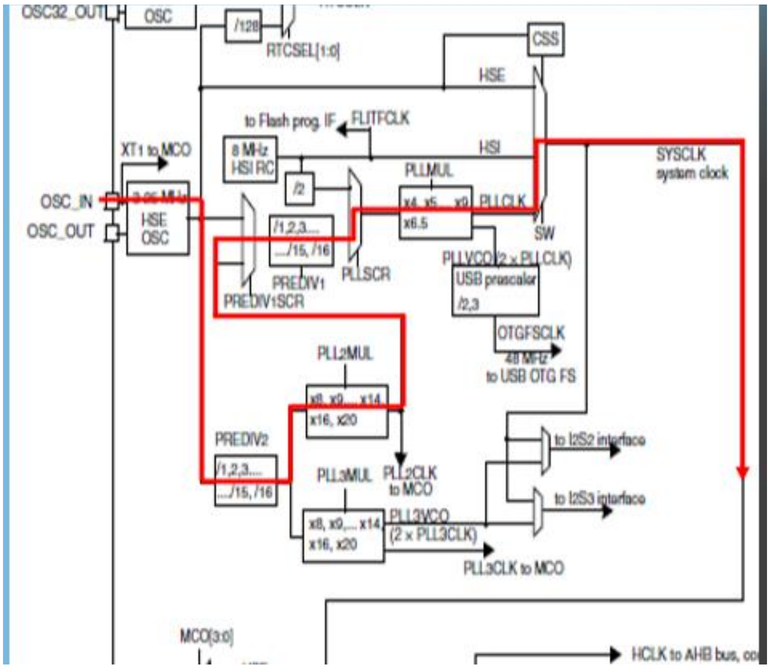


```
Hello World
Hello World
Hello World
Hello World
Hello World
```

B. 코드 분석



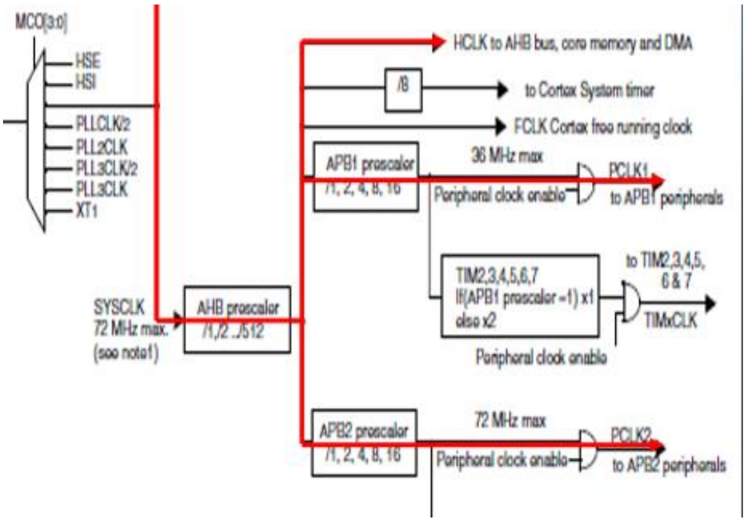
```
//@TODO - 1 Set the clock, (//) 주석 표시를 없애고 틀린 값이 있다면 제대로 된 값으로 수정하시오
    /* HCLK = SYSCLK */
    RCC->CFGR |= (uint32_t) RCC_CFGR_HPRE_DIV1;
    /* PCLK2 = HCLK */
        //RCC->CFGR |= (uint32_t) RCC_CFGR_PPRE2_DIV1;
    RCC->CFGR |= (uint32_t) RCC_CFGR_PPRE2_DIV1;
    /* PCLK1 = HCLK */
    RCC->CFGR |= (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->CFGR |= (uint32_t) (RCC_CFGR_PLLXTPRE_PREDIV1 | RCC_CFGR_PLLSRC_PREDIV1 | RCC_CFGR_PLLMULL5);

        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_PLL1 | RCC_CFGR2_PREDIV1_DIV1);
        RCC->CFGR2 |= (uint32_t) (RCC_CFGR2_PREDIV2_DIV5 | RCC_CFGR2_PLL2MUL8 | RCC_CFGR2_PREDIV1SRC_PLL2 | RCC_CFGR2_PREDIV1_DIV5);

//@End of TODO - 1
```



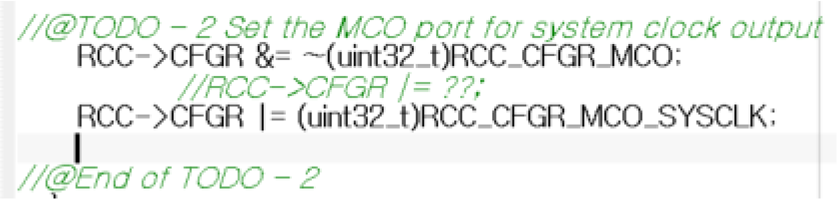
설정하고자 하는 SYSCLK이 40MHz이고 HSE OSC에서 25MHz가 나오므로 PREDIV2에서 /5를 PLL2MUL에서 *8을 PREDIV1에서 /5를 계산해 주어야 하므로 주어진 template.c의

RCC->CFGR2 |= (uint32_t)(RCC_CFGR2_PREDIV2_DIV1 | RCC_CFGR2_PLL2MUL1 | RCC_CFGR2_PREDIV1SRC_PLL2 | RCC_CFGR2_PREDIV1_DIV1);를

RCC->CFGR2 |= (uint32_t) (RCC_CFGR2_PREDIV2_DIV5 | RCC_CFGR2_PLL2MUL8 | RCC_CFGR2_PREDIV1SRC_PLL2 | RCC_CFGR2_PREDIV1_DIV5);로 수정합니다.



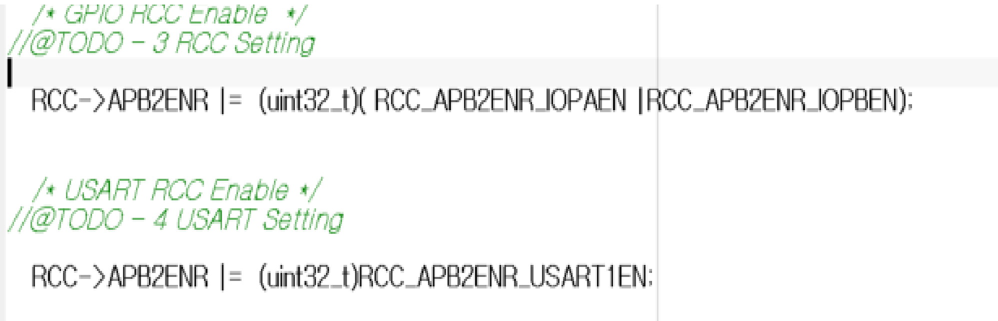
PCLK2로 40MHz를 내보내야 합니다.



```
//@TODO - 2 Set the MCO port for system clock output
    RCC->CFGR &= ~(uint32_t)RCC_CFGR_MCO;
          //RCC->CFGR |= ??;
    RCC->CFGR |= (uint32_t)RCC_CFGR_MCO_SYSCLK;

//@End of TODO - 2
```

MCO port를 SYSCLK의 output으로 set하기 위해

RCC->CFGR |= RCC_CFGR_MCO_SYSCLK; 를 추가합니다.

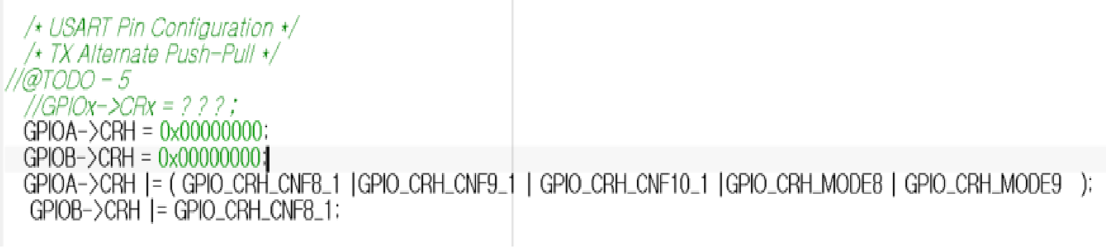


```
  /* GPIO RCC Enable  */
//@TODO - 3 RCC Setting

  RCC->APB2ENR |=  (uint32_t)( RCC_APB2ENR_IOPAEN |RCC_APB2ENR_IOPBEN);

  /* USART RCC Enable */
//@TODO - 4 USART Setting

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

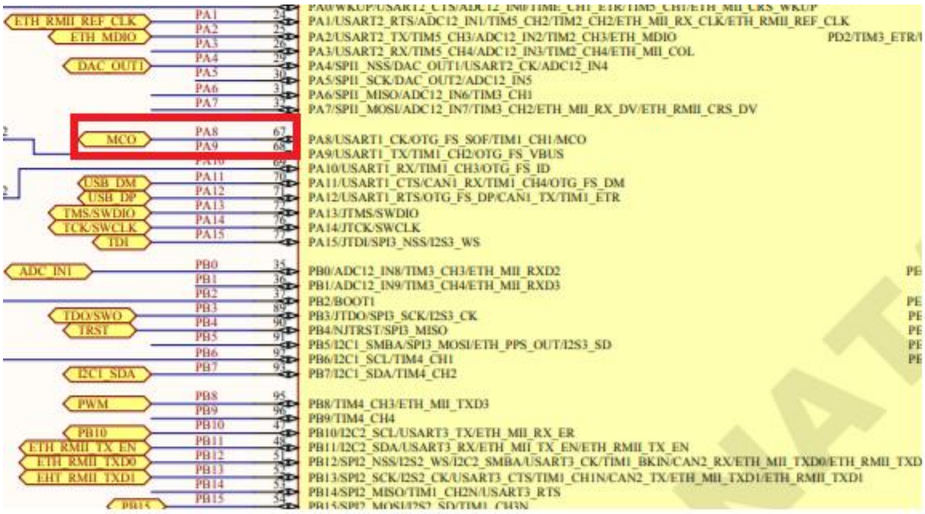
SYSCLK를 출력할 Port A와 버튼 입력을 받을 Port D 를 통신을 위해 USART 1을 enable 시키기 위해서

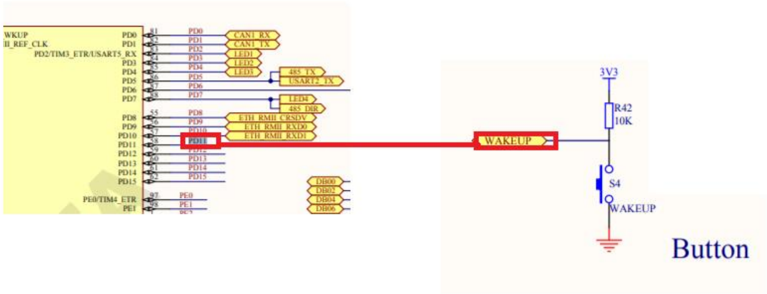
RCC->APB2ENR |= (RCC_APB2ENR_IOPAEN | RCC_APB2ENR_IOPDEN);

RCC->APB2ENR |= RCC_APB2ENR_USART1EN; 를 추가합니다.



```
 /* USART Pin Configuration */
 /* TX Alternate Push-Pull */
//@TODO - 5
 //GPIOx->CRx = ? ? ? ;
 GPIOA->CRH = 0x00000000;
 GPIOB->CRH = 0x00000000;
 GPIOA->CRH |= ( GPIO_CRH_CNF8_1 |GPIO_CRH_CNF9_1 | GPIO_CRH_CNF10_1 |GPIO_CRH_MODE8 | GPIO_CRH_MODE9  );
  GPIOB->CRH |= GPIO_CRH_CNF8_1;
```

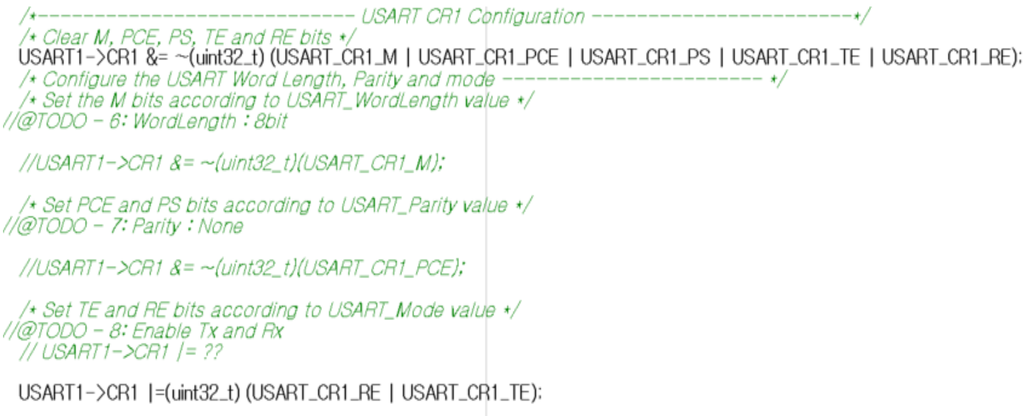

ETH_RMII_REF_CLK
ETH_MDIO
DAC_OUT1
MCO
USB_DM
USB_DP
TMS/SWDIO
TCK/SWCLK
TDI
ADC_IN1
TDO/SWO
TRST
I2C1_SDA
PWM
PB10
ETH_RMII_TX_EN
ETH_RMII_TXD0
EHT_RMII_TXD1
PB15
PA1 24
PA2 25
PA3 26
PA4 29
PA5 30
PA6 31
PA7 32
PA8 67
PA9 68
PA10 69
PA11 70
PA12 71
PA13 72
PA14 76
PA15 77
PB0 35
PB1 36
PB2 37
PB3 89
PB4 90
PB5 91
PB6 92
PB7 93
PB8 95
PB9 96
PB10 47
PB11 48
PB12 51
PB13 52
PB14 53
PB15 54
PA0/WKUP/USART2_CTS/ADC12_IN0/TIM5_CH1_ETR/TIM5_CH1/ETH_MII_CRS_WKUP
PA1/USART2_RTS/ADC12_IN1/TIM5_CH2/TIM2_CH2/ETH_MII_RX_CLK/ETH_RMII_REF_CLK
PA2/USART2_TX/TIM5_CH3/ADC12_IN2/TIM2_CH3/ETH_MDIO
PA3/USART2_RX/TIM5_CH4/ADC12_IN3/TIM2_CH4/ETH_MII_COL
PA4/SPI1_NSS/DAC_OUT1/USART2_CK/ADC12_IN4
PA5/SPI1_SCK/DAC_OUT2/ADC12_IN5
PA6/SPI1_MISO/ADC12_IN6/TIM3_CH1
PA7/SPI1_MOSI/ADC12_IN7/TIM3_CH2/ETH_MII_RX_DV/ETH_RMII_CRS_DV
PA8/USART1_CK/OTG_FS_SOF/TIM1_CH1/MCO
PA9/USART1_TX/TIM1_CH2/OTG_FS_VBUS
PA10/USART1_RX/TIM1_CH3/OTG_FS_ID
PA11/USART1_CTS/CAN1_RX/TIM1_CH4/OTG_FS_DM
PA12/USART1_RTS/OTG_FS_DP/CAN1_TX/TIM1_ETR
PA13/JTMS/SWDIO
PA14/JTCK/SWCLK
PA15/JTDI/SPI3_NSS/I2S3_WS
PB0/ADC12_IN8/TIM3_CH3/ETH_MII_RXD2
PB1/ADC12_IN9/TIM3_CH4/ETH_MII_RXD3
PB2/BOOT1
PB3/JTDO/SPI3_SCK/I2S3_CK
PB4/NJTRST/SPI3_MISO
PB5/I2C1_SMBA/SPI3_MOSI/ETH_PPS_OUT/I2S3_SD
PB6/I2C1_SCL/TIM4_CH1
PB7/I2C1_SDA/TIM4_CH2
PB8/TIM4_CH3/ETH_MII_TXD3
PB9/TIM4_CH4
PB10/I2C2_SCL/USART3_TX/ETH_MII_RX_ER
PB11/I2C2_SDA/USART3_RX/ETH_MII_TX_EN/ETH_RMII_TX_EN
PB12/SPI2_NSS/I2S2_WS/I2C2_SMBA/USART3_CK/TIM1_BKIN/CAN2_RX/ETH_MII_TXD0/ETH_RMII_TXD
PB13/SPI2_SCK/I2S2_CK/USART3_CTS/TIM1_CH1N/CAN2_TX/ETH_MII_TXD1/ETH_RMII_TXD1
PB14/SPI2_MISO/TIM1_CH2N/USART3_RTS
PB15/SPI2_MOSI/I2S2_SD/TIM1_CH3N
PD2/TIM3_ETR/



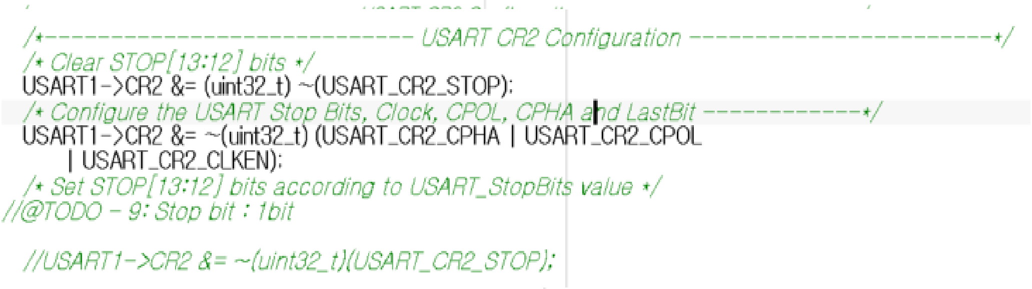
S1버튼의 input 값을 받아드리고 MCO의 output 값을 내보내기 위해서 A와 D PORT를 reset하는

```
GPIOA->CRH = 0x00000000;

GPIOB->CRH = 0x00000000;

GPIOA->CRH |= ( GPIO_CRH_CNF8_1 |GPIO_CRH_CNF9_1 | GPIO_CRH_CNF10_1 |GPIO_CRH_MODE8 | GPIO_CRH_MODE9 );

GPIOB->CRH |= GPIO_CRH_CNF8_1;
```

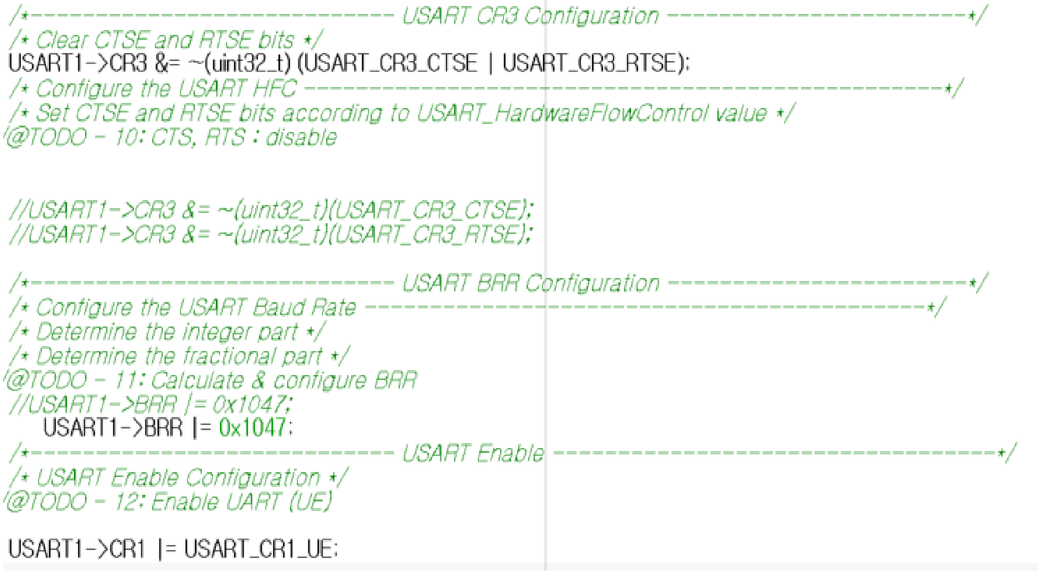
코드를 추가 합니다.



```
  /*---------------------------- 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

  //USART1->CR1 &= ~(uint32_t)(USART_CR1_M);

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

  //USART1->CR1 &= ~(uint32_t)(USART_CR1_PCE);

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

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



```
  /*---------------------------- 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

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



```
 /*---------------------------- 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


 //USART1->CR3 &= ~(uint32_t)(USART_CR3_CTSE);
 //USART1->CR3 &= ~(uint32_t)(USART_CR3_RTSE);

 /*---------------------------- USART BRR Configuration -----------------------*/
 /* Configure the USART Baud Rate -------------------------------------------*/
 /* Determine the integer part */
 /* Determine the fractional part */
'@TODO - 11: Calculate & configure BRR
 //USART1->BRR |= 0x1047;
    USART1->BRR |= 0x1047;
 /*---------------------------- USART Enable ---------------------------------*/
 /* USART Enable Configuration */
'@TODO - 12: Enable UART (UE)

 USART1->CR1 |= USART_CR1_UE;
```

$$\frac{T_x}{R_x}\text{ Baud } = \frac{f_{ck}}{16\times \text{USARTDIV}}$$

$$\text{USARTDIV} = 169.27083$$

$$DIV\_\text{Mantissa} = 1028 = 0X104$$

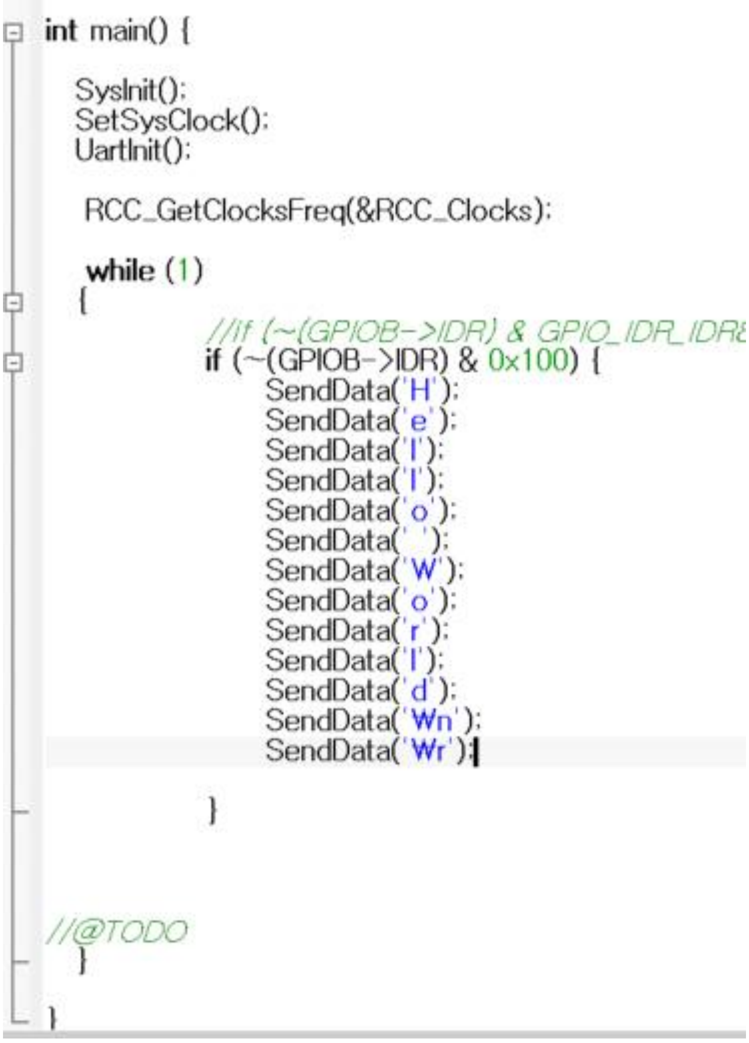
$$DIV\_\text{Fraction} = 16 \times 0d0.27083 = 0X7$$

$$\text{USART\_BR}R = 0X1047$$

이므로 `USART1->BRR |= 0x1047;`를 추가합니다.

UART를 enable하기 위해서

`USART1->CR1 |= USART_CR1_UE;`를 추가합니다.



```
int main() {

    SysInit();
    SetSysClock();
    UartInit();

     RCC_GetClocksFreq(&RCC_Clocks);

     while (1)
    {
                    //if (~(GPIOB->IDR) & GPIO_IDR_IDR8
                    if (~(GPIOB->IDR) & 0x100) {
                            SendData('H');
                            SendData('e');
                            SendData('l');
                            SendData('l');
                            SendData('o');
                            SendData(' ');
                            SendData('W');
                            SendData('o');
                            SendData('r');
                            SendData('l');
                            SendData('d');
                            SendData('₩n');
                            SendData('₩r');

                    }



//@TODO
    }

}
```

버튼을 눌러주는 동안 "Hello World"을 보내주어야 하므로 위와 같이 while문을 채웠습니다.

5. 결론

Clock tree의 그림을 보고 이를 이용하여 SYSCLK과 PCLK2를 직접 설정하여 주고 Baud Rate 식을 이용하여 UART통신을 하기 위한 코드를 직접 짜봄으로써 Clock과 UART에 대한 이해와 지식이 더욱 늘어났습니다.