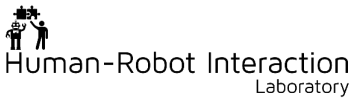
임베디드시스템설계
EMBEDDED SYSTEM DESIGN

CHAPTER 11

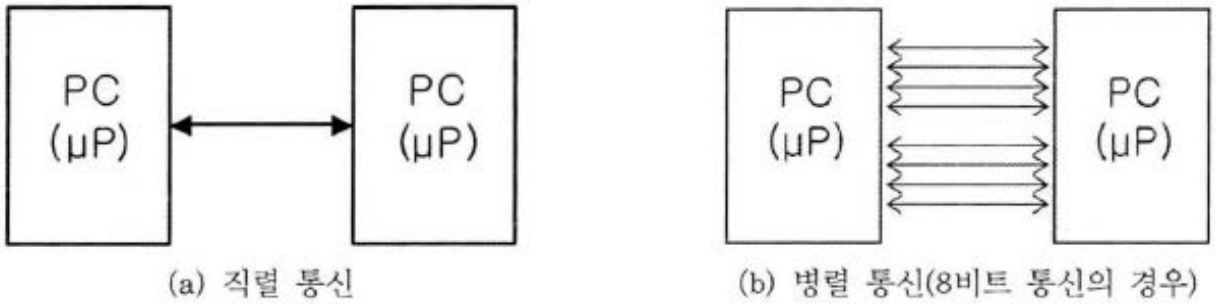
UART를 이용한 PC와의 통신



11.1 직렬 통신 및 UART

직렬(Serial) 통신과 병렬(Parallel) 통신

- 컴퓨터와 마이크로프로세서, 그리고 IC 칩들 사이에서 서로 데이터를 주고 받는 것을 데이터 통신이라고 함
- 직렬 통신은 1개의 라인 만을 이용하여 1 클럭당 1비트의 데이터를 전송함
- 병렬 통신은 여러 개의 라인을 이용하여 1 클럭당 여러 개의 비트를 전송함



직렬 통신과 병렬 통신의 예

KYUNG HEE UNIVERSITY

11.1 직렬 통신 및 UART

직렬(Serial) 통신과 병렬(Parallel) 통신

- 직렬 통신을 이용할 경우 필요한 라인의 수가 작으므로 마이크로프로세서에 내장되는 통신 포트에 필요한 핀 수가 작아짐
- 칩의 크기가 소형화되고 가격이 저렴해지는 장점이 있으므로, 많은 마이크로프로세서나 IC 칩은 외부와의 데이터 통신용으로 직렬 통신 방법을 많이 이용함
- 직렬 통신의 예로는 다음과 같은 것들이 있음
  - 모르스(Morse) 부호를 사용하는 통신 (전보 등)
  - RS-232C, RS-422, RS-485
  - USB, TCP/IP(인터넷), CAN
  - SPI
  - FireWire, PCI Express

11.1 직렬 통신 및 UART

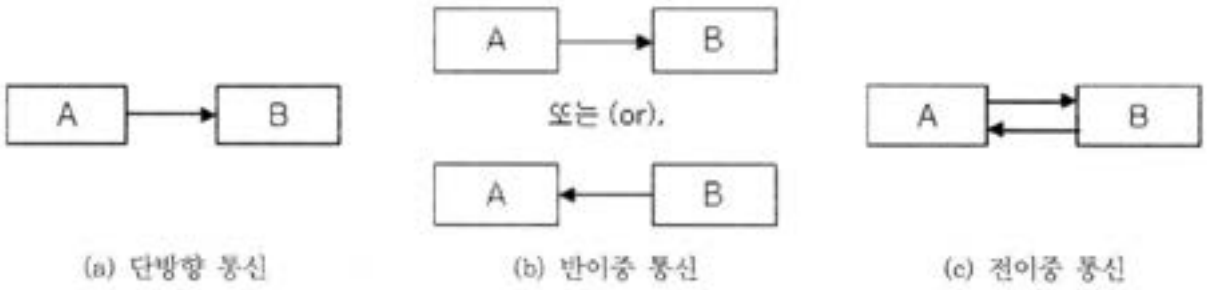
직렬(Serial) 통신과 병렬(Parallel) 통신

- 병렬 통신은 1클럭 시간에 통신 라인 수만큼의 데이터 전송이 가능하므로, 1개의 데이터만 보낼 수 있는 직렬 통신 보다는 개념적으로 통신 속도가 빠름
- 최근에는 직렬 통신이 병렬 통신보다 훨씬 빠른 클럭의 사용이 가능해지면서, 직렬 통신의 속도가 빨라지는 경우도 발생하게 됨
- 직렬 통신에서 고주파수의 클럭의 사용이 가능한 이유는, 통신에 필요한 라인의 개수가 병렬 통신보다 훨씬 적으므로 외부 노이즈의 차폐가 용이하고 크로스토크(crosstalk) 등의 전기적 특성이 유리하기 때문

11.1 직렬 통신 및 UART

단방향 통신과 양방향 통신

- 통신은 데이터의 흐름 방법에 따라 한 방향으로만 통신이 가능한 단방향(Simplex) 통신과 쌍방의 통신이 가능한 양방향 통신으로 나눌 수 있음
- 양방향 통신은 다시 반이중(Half-duplex) 방식과 전이중(Full-duplex) 방식으로 나눌 수 있음



데이터의 흐름 방법에 따른 통신 방식

11.1 직렬 통신 및 UART

UART/USART

- 마이크로프로세서 내에서 데이터 버스는 8비트, 16비트 등의 병렬 통신방식을 사용하므로 직렬 통신 포트를 이용한 데이터 송신 시에는 병렬을 직렬로, 수신시에는 직렬을 병렬로 변환해주어야 함
- UART(Universal Asynchronous Receiver/Transmitter : 범용 비동기 송수신기)는 데이터를 직렬 또는 병렬로 변환시켜주는 물리적 장치임
- PC, 마이크로프로세서와 주변장치들 간에 직렬(Serial) 포트를 이용한 통신에 주로 사용됨
- 최근의 UART는 비동기뿐만 아니라 동기 통신도 가능한 기능을 가지는 것들이 많아졌는데, 이러한 장치를 USART(Universal Synchronous and Asynchronous Receiver/Transmitter : 범용 동기/비동기 송수신기)라고 부름
- 동기와 비동기의 차이는 두 장치 사이에 데이터가 전송될 때 수신부의 클럭이 송신부의 클럭에 동기되는지 여부임

11.1 직렬 통신 및 UART

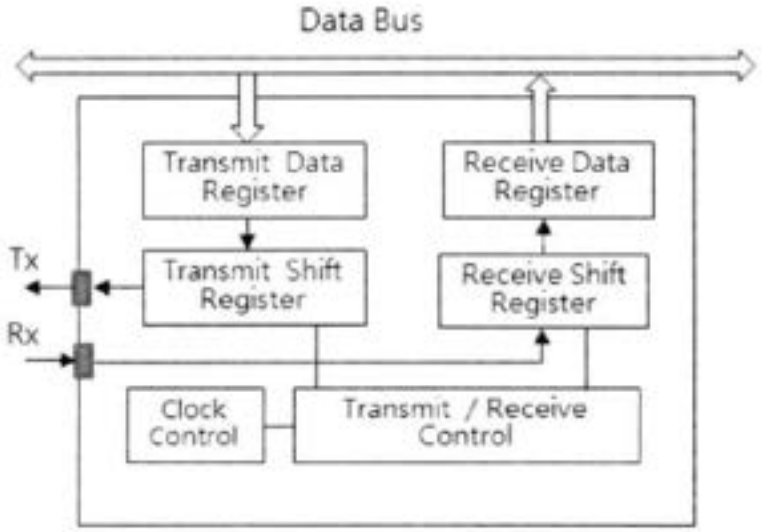
UART의 일반적인 구조

- 송신
  - 송신 데이터 레지스터(Transmit Date Register)는 전송할 데이터를 데이터 버스를 통해 받아 저장함
  - 이 데이터는 송신 쉬프트 레지스터(Transmit Shift Register)로 전달됨
  - 여기까지 데이터는 워드 또는 바이트 단위로 전달됨
  - 송신 쉬프트 레지스터는 데이터를 비트 단위로 TX 단자를 통해 외부로 내보냄
- 수신
  - 수신될 데이터는 외부에서 RX 단자를 통해 수신 쉬프트 레지스터(Receive Shift Register)로 들어옴
  - 수신 쉬프트 레지스터는 비트 단위로 수신된 데이터를 모아서 워드 또는 바이트 단위로 만들어 수신 데이터 레지스터(Receive Data Register)로 전달함
  - 수신 데이터 레지스터에 저장된 데이터는 데이터 버스를 통해 칩의 내부로 전달됨

11.1 직렬 통신 및 UART

UART의 일반적인 구조

- 송수신 제어부(Transmit/Receive Control)는 송신, 수신 과정을 제어해 주는 역할을 함
- 클럭 제어부(Clock Control)는 송신, 수신에 필요한 클럭을 만들어 내거나 제어하는 역할을 함



UART의 일반적인 구조

11.1 직렬 통신 및 UART

데이터 프레임(Data Frame)의 구성

- UART의 데이터 송수신은 프레임(Frame) 단위로 이루어짐
- 1 프레임의 데이터는 일반적으로 다음과 같이 구성됨
  - 1비트의 시작(start) 비트
  - 5~8비트의 데이터(data) 비트
  - 1비트의 패리티(parity) 비트 (옵션 요소)
  - 1~2 비트의 정지(stop) 비트

데이터 프레임의 구성

KYUNG HEE UNIVERSITY

11.1 직렬 통신 및 UART

데이터 전송을 위한 클럭의 조건

- 비동기 통신에서는 송신부와 수신부의 클럭이 동기되지 않으므로 송신부와 수신부는 서로 별개의 클럭으로 동작함
- 따라서 송신부의 클럭과 수신부의 클럭은 서로 일치해야 하며, 실제 구현 시에는 1프레임 송수신시에 양쪽 클럭의 주파수 차이가 10% 이내여야 함

보 레이트(Baud Rate)

- 보 레이트는 데이터의 전송 속도를 나타내는 말
- 1보(baud)는 1초에 전자적인 상태가 1번 변화하는 것을 말함
- 따라서 보 레이트가 1이면 1초에 1비트의 데이터가 전송됨

11.2 USART의 구조 및 기능

STM32F1 시리즈의 USART 개요

- STM32F1 시리즈에 장착된 직렬 통신용 주변 장치의 명칭은 USART임
- 이 USART를 이용하면 UART 통신 및 USART 통신이 모두 가능함
- 강의에서 사용하는 STM32F103C8은 3개(USART1 ~ USART3)의 USART를 가짐

11.2 USART의 구조 및 기능

STM32F10x의 USART의 주요 특징

- 각 USART의 특징
  - USART1 : 고속의 APB2 버스에 연결
  - USART2 ~ USART3 : 저속의 APB1 버스에 연결
- 전이중(Full-duplex) 방식의 동기, 비동기 통신이 가능
- 4.5 Mbits/s까지의 보레이트 구현이 가능
- 데이터의 워드 길이를 8비트 또는 9비트로 설정 가능
- 1비트 또는 2비트의 정지(Stop) 비트의 설정 가능
- 다음의 전송 상태 플래그의 검출이 가능
  - 수신 버퍼 가득 참 (Receive Buffer Full)
  - 송신 버퍼 비어있음 (Transmit Buffer Empty)
  - 송신 완료 (End of Transmission)

11.2 USART의 구조 및 기능

STM32F10x의 USART의 주요 특징

- 패리티(Parity) 제어 가능
  - 패리티 비트 송신, 수신 데이터의 패리티 검사
- 4개의 에러 검출 플래그가 있음
  - 오버런 에러, 노이즈 에러, 프레임 에러, 패리티 에러
- 10개의 인터럽트 소스와 대응되는 플래그 발생
  - 송신 데이터 레지스터 비어있음
  - 송신 완료
  - 수신 데이터 레지스터 가득 참
  - 아이들 라인(Idle Line) 받음
  - 오버런(Over run) 에러
  - 프레임(Frame) 에러
  - 노이즈(Noise) 에러
  - 패리티(Parity) 에러
  - CTS 변화
  - LIN(Local Interconnection Network) 브레이크 검출

11.2 USART의 구조 및 기능

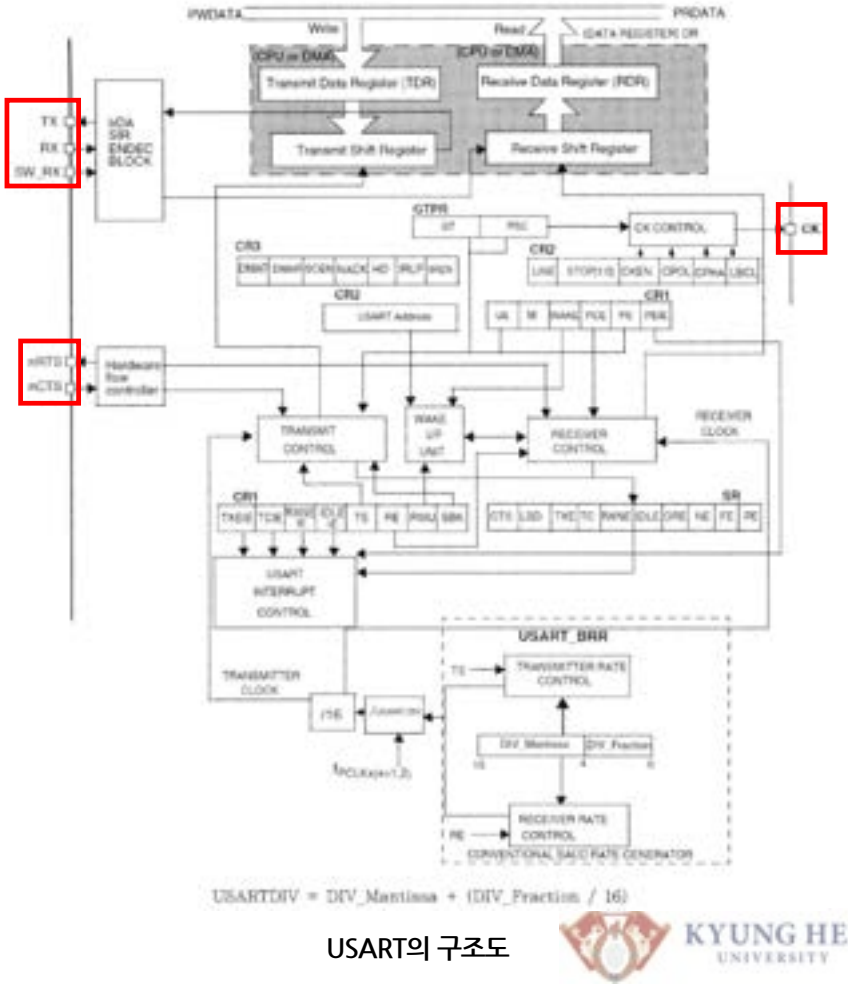
STM32F10x의 USART의 주요 특징

- **스마트 카드 프로토콜 지원**
- **IrDA(Infrared Data Association) 지원**

11.2 USART의 구조 및 기능

USART의 구조도

- 외부와의 통신에 사용되는 핀은 RX와 TX
- 데이터는 이 핀을 통하여 프레임 단위로 송수신됨
- RX는 싱글 와이어 모드나 스마트카드 모드에서 데이터 송수신용 핀으로도 사용되며, 이 경우 USART의 입력 핀으로는 SW_RX 핀이 사용됨
- CK 핀은 동기화 전송시에 필요한 클럭을 출력하는 핀
- nCTS(Clear To Send), nRTS(Request To Send)은 IrDA 데이터 통신에 사용되는 핀



USART의 구조도

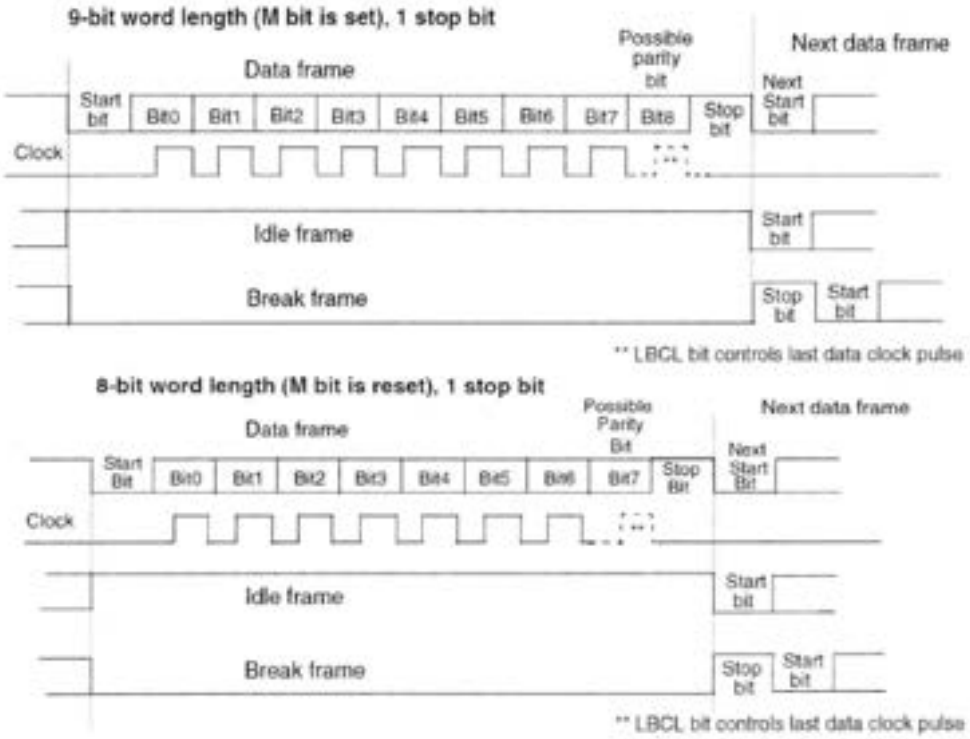
11.2 USART의 구조 및 기능

데이터 프레임의 구성

- 1개의 데이터 프레임은 다음과 같이 구성됨
  - 시작(start) 비트 : 1비트
  - 데이터(data) 비트 : 8~9비트
  - 정지(stop) 비트 : 0.5~2비트
- 시작 비트는 로우(low) 값을, 정지(stop) 비트는 하이(high) 값을 가짐
- 데이터 비트는 8비트 또는 9비트로 설정할 수 있음
- 이는 USART_CR1 레지스터의 M 비트의 설정 값에 따라 결정됨
- 데이터 비트의 제일 마지막은 패리티 비트가 될 수도 있음
- 시작 비트를 포함한 모든 비트가 하이(high)인 프레임을 아이들(idle) 프레임이라고 함
- 모든 비트가 로우(low)인 프레임을 브레이크(break) 프레임이라고 함

11.2 USART의 구조 및 기능

데이터 프레임의 구성



데이터 프레임의 구성

11.2 USART의 구조 및 기능

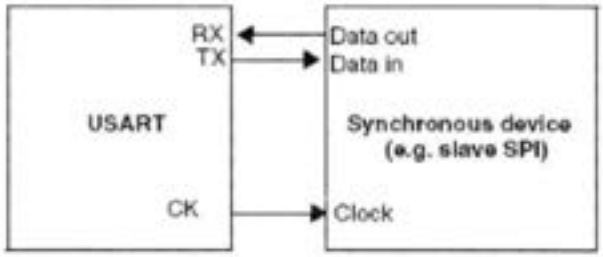
문자 송신 및 수신

- USART는 데이터 송신시에 하위 비트부터 TX 핀을 통해 송신하며 수신시에도 하위 비트부터 RX 핀을 통해 수신함
- 수신시에는 노이즈를 제거하고 정확한 데이터의 수신을 위해 오버 샘플링(over sampling) 방법을 이용함
- 정지 비트는 송수신 모드에 따라 달라지며 다음의 값이 사용됨
  - 1비트의 정지 비트 : 디폴트 값
  - 2비트의 정지 비트 : 싱글 와이어 및 모뎀 모드의 USART 통신
  - 0.5비트의 정지 비트 : 스마트카드 모드의 데이터 수신시
  - 1.5비트의 정지 비트 : 스마트카드 모드의 데이터 송신시

11.2 USART의 구조 및 기능

동기 모드

- USART_CR2 레지스터의 CLKEN 비트를 설정하면 USART는 동기 모드로 동작함
- 이 모드로 송신시에는 TX가 클럭 CK와 동기되며 나머지는 비동기 모드와 동일하게 동작함
- 수신은 오버 샘플링없이 CK와 동기되어 수신됨



동기 송신 모드의 사용 예

11.3 UART 관련 HAL 드라이버

UART 설정용 구조체

- 구조체의 종류
  - UART_InitTypeDef
    - UART 동작조건 설정을 위한 구조체
  - UART_HandleTypeDef
    - UART 설정을 위한 구조체

11.3 UART 관련 HAL 드라이버

UART 설정용 구조체

UART_InitTypeDef

UART의 동작조건을 설정하기 위한 구조체이며 stm32f1xx_hal_uart.h에 정의되어 있다.

[데이터 형]

- uint32_t BaudRate
- uint32_t WordLength
- uint32_t StopBits
- uint32_t Parity
- uint32_t Mode
- uint32_t HwFlowCtl
- uint32_t OverSampling

[데이터 형의 설명]

- BaudRate : UART 통신 보 레이트를 설정한다. 보 레이트는 다음의 식을 이용하여 계산한다.
  - IntegerDivider = ((PCLKx)/(16 * (huart->Init.BaudRate)))
  - FractionalDivider = ((IntegerDivider - ((uint32_t) IntegerDivider)) * 16)+0.5
- WordLength : 송수신 할 데이터 비트의 크기를 지정
  - UART_WORDLENGTH_8B
  - UART_WORDLENGTH_9B
- StopBits : stop 비트의 크기를 지정
  - UART_STOPBITS_1
  - UART_STOPBITS_2

주요 구조체의 상세 설명

11.3 UART 관련 HAL 드라이버

UART 설정용 구조체

- Parity : parity 모드 지정
  - UART_PARITY_NONE
  - UART_PARITY_EVEN
  - UART_PARITY_ODD

참고 패리티가 활성화되면 계산된 패리티가 전송 데이터의 MSB 위치로 들어가게 된다. (9th bit : word length를 9 data bits로 설정 시 삽입, 8th bit : word length를 8 data bits로 설정 시 삽입)

- Mode : 송수신 모드를 활성화 또는 비활성화로 지정
  - UART_MODE_RX
  - UART_MODE_TX
  - UART_MODE_TX_RX
- HwFlowCtl : hardware flow control 모드를 활성화 또는 비활성화로 지정
  - UART_HWCONTROL_NONE
  - UART_HWCONTROL_RTS
  - UART_HWCONTROL_CTS
  - UART_HWCONTROL_RTS_CTS
- OverSampling : Oversampling을 활성화 또는 비활성화로 지정. STM32F1xx 시리즈 디바이스는 적용되지 않으며 OverSampling은 항상 16으로 설정되어야 한다.
  - UART_OVERSAMPLING_1

주요 구조체의 상세 설명

11.3 UART 관련 HAL 드라이버

UART 설정용 구조체

UART_HandleTypeDef

UART 설정을 위한 구조체이며 stm32f1xx_hal_uart.h에 정의되어 있다.

[데이터 형]

• USART_TypeDef *	Instance
• UART_InitTypeDef	Init
• uint8_t *	pTxBuffPtr
• uint16_t	TxXferSize
• uint16_t	TxXferCount
• uint8_t *	pRxBuffPtr
• uint16_t	RxXferSize
• uint16_t	RxXferCount
• DMA_HandleTypeDef *	hdmatx
• DMA_HandleTypeDef *	hdmarx
• HAL_LockTypeDef	Lock
• __IO HAL_UART_StateTypeDef	State
• __IO uint32_t	ErrorCode

주요 구조체의 상세 설명

11.3 UART 관련 HAL 드라이버

UART 설정용 구조체

[데이터 형의 설명]

- Instance : UART 레지스터 base address
- Init : UART 통신 파라미터
- pTxBuffPtr : UART Tx 버퍼
- TxXferSize : UART Tx 크기
- TxXferCount : UART Tx 카운터
- pRxBuffPtr : UART Rx 버퍼
- RxXferSize : UART Rx 크기
- RxXferCount : UART Rx 카운터
- hdmatx : UART Tx DMA 핸들러 파라미터
- hdmarx : UART Rx DMA 핸들러 파라미터
- Lock : Locking object
- State : UART 통신 상태
- ErrorCode : UART 에러 코드

주요 구조체의 상세 설명

11.3 UART 관련 HAL 드라이버

UART 구동용 HAL 함수

- **초기화 및 설정용 함수**
  - **HAL_UART_Init(), HAL_UART_DeInit()**
  - **UART의 초기화, 초기화 해제**
  - **HAL_UART_MspInit(), HAL_UART_MspDeInit()**
  - **UART Msp의 초기화, 초기화 해제**

11.3 UART 관련 HAL 드라이버

UART 구동용 HAL 함수

- 입출력용 함수
  - HAL_UART_Transmit()
    - 블로킹 모드로 데이터 송신
  - HAL_UART_Receive()
    - 블로킹 모드로 데이터 수신
  - HAL_UART_Transmit_IT()
    - 비 블로킹 모드로 데이터 송신
  - HAL_UART_Receive_IT()
    - 비 블로킹 모드로 데이터 수신

11.3 UART 관련 HAL 드라이버

UART 구동용 HAL 함수

- 입출력용 함수
  - HAL_UART_Transmit_DMA()
    - DMA를 사용하여 비 블로킹 모드로 데이터 송신
  - HAL_UART_Receive_DMA()
    - DMA를 사용하여 비 블로킹 모드로 데이터 수신
  - HAL_UART_IRQHandler()
    - UART 인터럽트 처리 핸들러 함수

11.3 UART 관련 HAL 드라이버

UART 구동용 HAL 함수

- 주변장치 상태 함수
  - HAL_UART_GetState()
    - Run-time 중 UART 주변장치 상태 확인
  - HAL_UART_GetError()
    - 통신 시 일어날 수 있는 run-time 에러 확인

11.3 UART 관련 HAL 드라이버

구동용 함수의 사용 방법

a. 클럭을 활성화함

- _HAL_RCC_USARTx_CLK_ENABLE() 함수를 이용하여 사용할 USARTx의 클럭을 활성화시킴
- _HAL_RCC_GPIOx_CLK_ENABLE() 함수를 이용하여 USART 핀으로 사용할 GPIO의 클럭을 활성화시킴

b. UART로 사용할 GPIO 핀을 대체 기능(Alternate function) 모드로 하고 UART 핀으로 사용하도록 설정한 다음 HAL_GPIO_Init() 함수를 이용하여 초기화

c. UART_HandlerTypeDef 구조체 변수를 이용하여 UART를 설정하고 HAL_UART_Init() 함수를 이용하여 동작조건을 설정

d. 통신 모드에 따라 다음의 함수를 이용하여 데이터를 송수신

- Blocking 모드
  - HAL_UART_Transmit(), HAL_UART_Receive()
- Non Blocking 모드 (인터럽트)
  - HAL_UART_Transmit_IT(), HAL_UART_Receive_IT()
  - HAL_UART_IRQHandler()
- DMA 모드
  - HAL_UART_Transmit_DMA(), HAL_UART_Receive_DMA()

11.3 UART 관련 HAL 드라이버

구동용 함수의 사용 방법

e. UART에서 인터럽트를 발생시키는 경우는 다음과 같이 처리함

- HAL_NVIC_SetPriority() 함수를 이용하여 해당 인터럽트의 우선 순위를 설정하고 HAL_NVIC_SetPriority() 함수를 이용하여 인터럽트를 활성화시킴
- 해당 인터럽트의 ISR(Interrupt Service Routine)을 작성함 (stm32f1xx_it.c에서 작성)
  - ISR의 예 : USART2_IRQHandler()
- 인터럽트의 발생시에 처리해야 할 내용을 해당 인터럽트의 콜백함수에 작성함
  - 콜백 함수의 예 : HAL_UART_RxCpltCallback(), HAL_UART_TxCpltCallback()

11.3 UART 관련 HAL 드라이버

구동용 함수의 동작 모드

a. 폴링(polling) 모드 입출력
   - HAL_UART_Transmit() 함수를 이용하여 블로킹 모드에서 데이터를 송신함
   - HAL_UART_Receive() 함수를 이용하여 블로킹 모드에서 데이터를 수신함

b. 인터럽트 모드 입출력
   - HAL_UART_Transmit_IT() 함수를 이용하여 비 블로킹 모드에서 데이터를 송신함
   - 송신이 끝나면 HAL_UART_TxCpltCallback() 콜백 함수가 호출됨
   - 그러므로 송신 완료시 수행되어야 하는 작업의 코드를 이 콜백 함수 내에 작성하면 됨

   - HAL_UART_Receive_IT() 함수를 이용하여 비 블로킹 모드에서 데이터를 수신함
   - 수신이 끝나면 HAL_UART_RxCpltCallback() 콜백 함수가 호출됨
   - 그러므로 수신 완료시 수행되어야 하는 작업의 코드를 이 콜백 함수 내에 작성하면 됨

   - 송수신 에러가 발생하면 HAL_UART_ErrorCallback() 콜백 함수가 호출됨
   - 그러므로 에러 발생 시 수행되어야 하는 작업의 코드를 이 콜백 함수 내에 작성하면 됨

11.3 UART 관련 HAL 드라이버

구동용 함수의 동작 모드

c. DMA 모드 입출력
- HAL_UART_Transmit()_DMA 함수를 이용하여 비 블로킹 모드에서 데이터를 송신함
  - Half transfer 송신이 완료되면 HAL_UART_TxHalfCpltCallback() 콜백 함수가 호출됨
  - 사용자는 송신이 완료되면 수행해야 하는 작업의 코드를 이 콜백 함수 내에 작성하면 됨

- HAL_UART_Receive() 함수를 이용하여 블로킹 모드에서 데이터를 수신함
  - Half transfer 수신이 완료되면 HAL_UART_RxHalfCpltCallback() 콜백 함수가 호출됨
  - 사용자는 수신이 완료되면 수행해야 하는 작업의 코드를 이 콜백 함수 내에 작성하면 됨

- 송수신 에러가 발생하면 HAL_UART_ErrorCallback() 콜백 함수가 호출됨
- 그러므로 에러 발생 시 수행되어야 하는 작업의 코드를 이 콜백 함수 내에 작성하면 됨

11.3 UART 관련 HAL 드라이버

UART 구동용 함수

- **초기화 및 설정용 함수**

HAL_UART_Init (UART_HandleTypeDef * huart)

UART_InitTypeDef에 명시된 파라미터에 따라 UART 모드를 초기화한다.

[파라미터]
- huart : UART handle

[반환 값] HAL status

HAL_UART_DeInit (UART_HandleTypeDef * huart)

UART 주변장치를 해제한다.

[파라미터]
- huart : UART handle

[반환 값] HAL status

11.3 UART 관련 HAL 드라이버

UART 구동용 함수

- **입출력용 함수**

HAL_UART_Receive (UART_HandleTypeDef* huart, uint8_t* pData, uint16_t Size, uint32_t Timeout)

데이터를 블로킹 모드로 수신한다.

[파라미터]
- huart : UART handle
- pData : 데이터 버퍼
- Size : 입력받을 데이터의 크기
- Timeout : 지연시간

[반환 값] HAL status

HAL_UART_Receive_IT (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)

데이터를 비 블로킹 모드로 수신한다.

[파라미터]
- huart : UART handle
- pData : 데이터 버퍼
- Size : 입력받을 데이터의 크기

[반환 값] HAL status

11.3 UART 관련 HAL 드라이버

UART 구동용 함수

- **입출력용 함수**

HAL_UART_Transmit (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size, uint32_t Timeout)

데이터를 블로킹 모드로 송신한다.

[파라미터]
- huart : UART handle
- pData : 데이터 버퍼
- Size : 입력받을 데이터의 크기
- Timeout : 지연 시간

[반환 값] HAL status

HAL_UART_Transmit_IT (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)

데이터를 비 블로킹 모드로 송신한다.

[파라미터]
- huart : UART handle
- pData : 데이터 버퍼
- Size : 입력받을 데이터의 크기

[반환 값] HAL status

11.3 UART 관련 HAL 드라이버

UART 구동용 함수

- **입출력용 함수**

HAL_UART_Transmit_DMA (UART_HandleTypeDef * huart, uint32_t* pData, uint32_t Size)

데이터를 비 블로킹 모드로 DMA를 통해 송신한다.

[파라미터]
- huart : UART handle
- pData : 해당 데이터
- Size : 데이터 크기

[반환 값] 없음

HAL_UART_Receive_DMA (UART_HandleTypeDef * huart, uint32_t* pData, uint32_t Size)

데이터를 비 블로킹 모드로 DMA를 통해 수신한다.

[파라미터]
- huart : UART handle
- pData : 해당 데이터
- Size : 데이터 크기

[반환 값] 없음

11.3 UART 관련 HAL 드라이버

UART 구동용 함수

- **입출력용 함수**

HAL_UART_IRQHandler (UART_HandleTypeDef * huart)

UART 인터럽트를 요청한다.

[파라미터]
- huart : UART handle

[반환 값] 없음

HAL_UART_TxCpltCallback (UART_HandleTypeDef * huart)

송신완료시 호출되는 callback 함수이다.

[파라미터]
- huart : UART handle

[반환 값] 없음

HAL_UART_RxCpltCallback (UART_HandleTypeDef * huart)

수신완료시 호출되는 callback 함수이다.

[파라미터]
- huart : UART handle

[반환 값] 없음

11.3 UART 관련 HAL 드라이버

UART 구동용 함수

- **주변 장치 상태 함수**

HAL_UART_GetState (UART_HandleTypeDef * huart)

UART의 동작 상태를 읽어온다.

[파라미터]
- huart : UART handle

[반환 값] HAL state

HAL_UART_GetError (UART_HandleTypeDef * huart)

UART의 에러 상태를 읽어온다.

[파라미터]
- huart : UART handle

[반환 값] ART 에러 코드

11.3 UART 관련 HAL 드라이버

UART 구동용 매크로(macro) 및 정의(define)

- **UART 에러 코드**

```
HAL_UART_ERROR_NONE
HAL_UART_ERROR_PE
HAL_UART_ERROR_NE
HAL_UART_ERROR_FE
HAL_UART_ERROR_ORE
HAL_UART_ERROR_DMA
```

- **UART 상태**

```
UART_STATE_DISABLE
UART_STATE_ENABLE
```

11.3 UART 관련 HAL 드라이버

UART 구동용 매크로(macro) 및 정의(define)

- **UART 매크로**

```
__HAL_UART_RESET_HANDLE_STATE
__HAL_UART_GET_FLAG
__HAL_UART_CLEAR_FLAG
__HAL_UART_CLEAR_PEFLAG
__HAL_UART_CLEAR_FEFLAG
__HAL_UART_CLEAR_NEFLAG
__HAL_UART_ENABLE_IT
__HAL_UART_DISABLE_IT
__HAL_UART_GET_IT_SOURCE
__HAL_UART_ENABLE
__HAL_UART_DISABLE
```

11.3 UART 관련 HAL 드라이버

UART 구동용 매크로(macro) 및 정의(define)

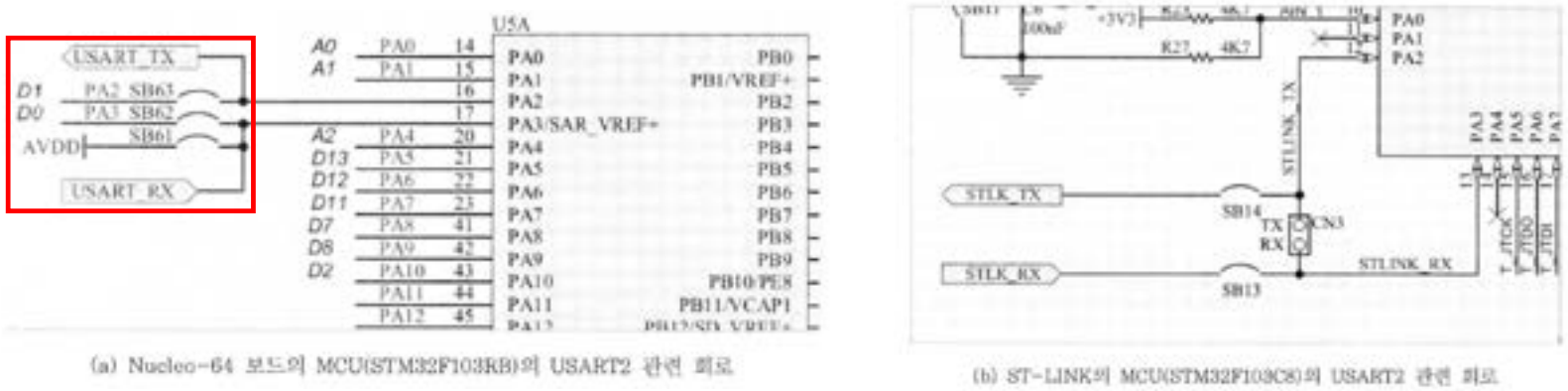
- **UART 인터럽트 정의**

```
UART_IT_PE
UART_IT_TXE
UART_IT_TC
UART_IT_RXNE
UART_IT_IDLE
UART_IT_LBD
UART_IT_CTS
UART_IT_ERR
```

11.4 UART 활용 예제

UART 관련 회로도

- Nucleo-F103 확장보드에서는 UART 통신을 위해 내장된 MCU인 STM32F103RB의 USART2를 사용함
- USART2_TX 핀은 PA2이고, USART2_RX 핀은 PA3
- USART2는 ST-LINK의 MCU와 연결되어 있으므로 이를 이용하면 PC와 UART 통신이 가능함



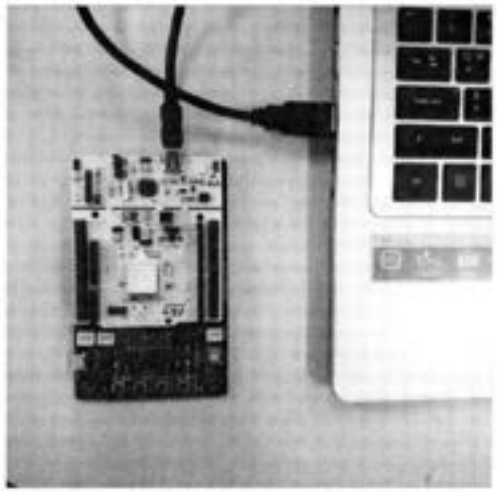
(a) Nucleo-64 보드의 MCU(STM32F103RB)의 USART2 관련 회로 (b) ST-LINK의 MCU(STM32F103C8)의 USART2 관련 회로

Nucleo-64 보드의 USART2 관련 회로

11.4 UART 활용 예제

예제 실행을 위한 PC와 실습 보드의 연결

- PC와 Nucleo-F103 확장보드의 USB 포트를 USB 케이블로 연결하기만 하면 됨
- 이 경우 PC와 Nucleo-F103 확장보드는 물리적으로는 USB 케이블로 연결되었지만, PC에서는 이를 가상 COM 포트로 인식함
- 따라서 PC나 Nucleo-F103 확장보드 양쪽 모두에서 UART 통신 프로그램으로 상호 통신이 가능함



PC와 Nucleo-F103 확장보드의 연결

11.4 UART 활용 예제

UART 예제 1 : UART를 이용하여 PC에 메시지 보내기

- UART를 이용하여 실습 보드에서 PC로 메시지를 보내는 예제
- UART의 TX, RX 핀을 이용하여 PC와 UART 통신을 함
- UART는 보 레이트 9600, 데이터 비트 8bit, 정지 비트 1bit, 패리티 없음, 흐름제어 없음으로 설정함
- PC에서는 하이퍼 터미널 프로그램을 이용하여 통신함
- 폴더 명 : [Examples_Nucleo F103] - [UART] - [UART 1_F103] - [MDK-ARM]
- 위의 폴더에 있는 [Project.uvprojx] 파일을 더블클릭하여 실행함

11.4 UART 활용 예제

UART 예제 1 : UART를 이용하여 PC에 메시지 보내기

[main.c]

```
#include "main.h"
#include "Nucleo_F103.h"              // Nucleo-F103 확장보드용 헤더 파일
// #include "Nucleo_F429.h"           // Nucleo-F429 확장보드용 헤더 파일

// -- <1> UartHandle 변수를 외부정의 변수로 선언
extern UART_HandleTypeDef UartHandle;

// -- <2> UART 통신을 위한 정의
#define TxBufferSize   (countof(TxBuffer) - 1)      // 송신 버퍼 사이즈 정의
#define RxBufferSize   0xFF                         // 수신 버퍼 사이즈를 0xFF로 정의
#define countof(a)     (sizeof(a) / sizeof(*(a)))   // 데이터 사이즈

// -- <3> UART 통신용 변수 선언
uint8_t TxBuffer[] = "\n\rUART Example 1 (Transmission  Success !!)\n\r\n\r";
uint8_t RxBuffer[RxBufferSize];

// ------------------------------------------------------------------------- //
```

```
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  LED_Config();

  // -- <4> UART의 초기설정용 함수를 호출
  UART2_Config();        // Nucleo-F103 확장보드 사용 시 함수 호출
  // UART3_Config();     // Nucleo-F429 확장보드 사용 시 함수 호출

  LED_OnOff(GPIO_PIN_LedAll, 500);

  // -- <5> TxBuffer에 저장되어 있는 내용을 PC로 보낸다.
  HAL_UART_Transmit(&UartHandle, (uint8_t*)TxBuffer, TxBufferSize , 0xFFFF);

  // -- <6> 무한 루프
  while (1) { }
}
```

UART를 이용하여 PC에 메시지 보내기 예제 코드

11.4 UART 활용 예제

UART 예제 1 : UART를 이용하여 PC에 메시지 보내기

- 타이머 인터럽트를 사용하기 위해서는 인터럽트 발생시에 처리되어야하는 동작이 정의되어 있는 인터럽트 핸들러(handler) 함수의 작성이 필요함
- 인터럽트 핸들러 함수 작성을 위해서 [Example/User] 폴더 내의 [stm32f1xx.it.c] 파일을 더블클릭하여 연 후에 다음과 같이 소스 코드를 작성

[stm32f1xx_it.c]

```
#include "main.h"
#include "stm32f1xx_it.h"            // 인터럽트 사용에 필요한 헤더 파일

// -- <1> UART 인터럽트 ISR을 위한 UartHandle 변수를 외부정의 변수로 선언
extern UART_HandleTypeDef   UartHandle;

// ------------------------------------------------------------------------- //

void SysTick_Handler(void)
{
    HAL_IncTick();
}
// ------------------------------------------------------------------------- //
```

```
void EXTI15_10_IRQHandler(void)
{
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_10);
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);
}
// ------------------------------------------------------------------------- //

void EXTI4_IRQHandler(void)
{
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_4);
}

// ------------------------------------------------------------------------- //

void EXTI9_5_IRQHandler(void)
{
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_5);
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_8);
}

// ------------------------------------------------------------------------- //
// -- <2> UART 인터럽트 Callback 함수
void USART2_IRQHandler(void)
{
    // -- <2-1> UART 인터럽트 Callback 함수
    HAL_UART_IRQHandler(&UartHandle);
}
```

인터럽트 함수 코드

11.4 UART 활용 예제

UART 예제 1 : UART를 이용하여 PC에 메시지 보내기

- **소스 코드 주석 설명**

[main.c]

<1> UART의 초기화를 위하여 구조체 Uart_HandleTypeDef 형의 변수 UartHandle를 외부변수로 선언한다. 이 변수는 <Nucleo_F103.c> / <Nucleo_F429.c>에 선언되어 있다.

<2> UART 통신을 위한 송신, 수신 버퍼의 사이즈를 정의 한다.

```
#define TxBufferSize (countof(TxBuffer) - 1)          // 송신 버퍼 사이즈 정의
#define RxBufferSize 0xFF                             // 수신 버퍼 사이즈를 0xFF로 정의
#define countof(a)      (sizeof(a) / sizeof(*(a)))    // 데이터 사이즈
```

<3> UART 통신의 송신용 변수 TxBuffer[]와 수신용 변수 RxBuffer[]를 선언한다. 본 예제는 수신만 하므로 송신용 변수 TxBuffer[]는 사용되지 않는다.

```
uint8_t TxBuffer[] = "\n\r\UART Example 1 (Transmission  Success !!)\n\r\n\r";
uint8_t RxBuffer[RxBufferSize];
```

<4> UART2_Config( ) 함수 : Nucleo-F103 확장보드는 UART2를 이용하여 통신을 한다. 이 함수는 UART2의 초기설정을 위한 것이다.

UART3_Config( ) 함수 : Nucleo-F429 확장보드는 UART3를 이용하여 통신을 한다. 이 함수는 UART3의 초기설정을 위한 것이다.

이 함수들은 각각 <Nucleo_F103.c> / <Nucleo_F429.c>에 정의되어 있다. [14.3절. 예제 소스코드 추가 설명 : Nucleo_F103.c 및 Nucleo_F429.c]을 참고하기 바란다.

** 이 함수에는 UART의 초기 동작을 설정하는 소스코드가 있다. 이 소스코드는 UART의 이해를 위해서 매우 중요한 코드이므로 반드시 찾아서 살펴보기 바란다.

<5> HAL_UART_Transmit() 함수 : TxBuffer에 저장되어 있는 내용을 UART를 통해 송신한다. 타임아웃 시간은 0xFF이다. .

<6> 프로그램은 어두운 작업 없이 무한 루프로 동작한다.

[stm32f1xx_it.c], [stm32f4xx_it.c]

<1> 구조체 Uart_HandleTypeDef 형의 변수 UartHandle를 외부변수로 선언한다. 이 변수는 USART2_IRQHandler( ) / USART3_IRQHandler( ) 함수에서 사용한다.

<2> UART2 / UART3 인터럽트가 발생하면 이를 처리하는 callback 함수이다.

<2-1> UART에서 발생한 여러 가지 인터럽트를 처리하는 Handler 함수이다. 이 함수의 내부에서는 발생된 UART 인터럽트의 종류에 맞는 콜백함수를 다시 호출해준다.

11.4 UART 활용 예제

UART 예제 1 : UART를 이용하여 PC에 메시지 보내기

- PC에서 하이퍼터미널 설정하는 방법

1. 다음의 폴더에서 하이퍼터미널 프로그램(hypertrm.exe)을 실행
   - 폴더 명 : [Utilities_2] - [hypertrm]
   - 위 폴더의 'hypertrm'을 더블클릭하여 실행

2. 그러면 다음과 같은 화면이 나오는데, 이 화면에서 연결할 [이름] (예 : test)를 입력하고 [확인]을 클릭함

하이퍼터미널 연결 창

KYUNG HEE UNIVERSITY

11.4 UART 활용 예제

UART 예제 1 : UART를 이용하여 PC에 메시지 보내기

- **PC에서 하이퍼터미널 설정하는 방법**

  3. **[연결에 사용할 모뎀]을 설정**
     COM 포트 번호는 각자의 PC에서 ST-Link가 설치된 COM 포트 번호로 선택하면 됨

연결에 사용할 모뎀 설정

KYUNG HEE UNIVERSITY

11.4 UART 활용 예제

UART 예제 1 : UART를 이용하여 PC에 메시지 보내기

- **PC에서 하이퍼터미널 설정하는 방법**

  4. 포트를 다음과 같이 설정하고 [확인]을 클릭함

포트 설정

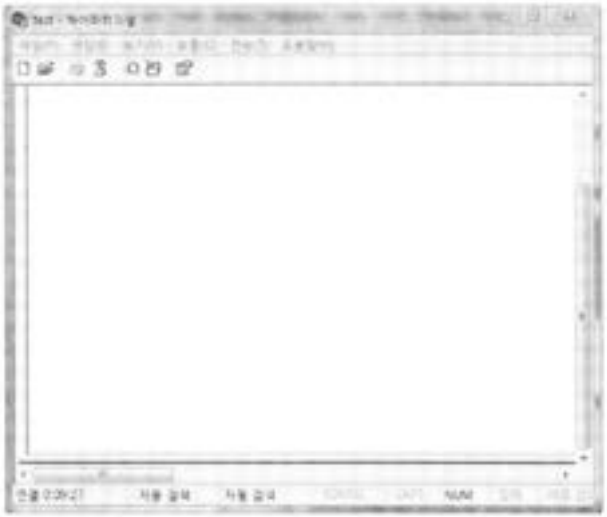
KYUNG HEE UNIVERSITY

11.4 UART 활용 예제

UART 예제 1 : UART를 이용하여 PC에 메시지 보내기

- **PC에서 하이퍼터미널 설정하는 방법**

  5. 다음과 같은 하이퍼터미널 창이 뜨면 통신할 준비가 완료됨



하이퍼터미널 창

KYUNG HEE UNIVERSITY

11.4 UART 활용 예제

UART 예제 1 : UART를 이용하여 PC에 메시지 보내기

- **프로그램 실행**

1. PC에서 하이퍼터미널 프로그램을 실행
   하이퍼터미널은 실습보드와 동일하게 보 레이트 9600, 데이터 비트 8bit, 정지 비트 1bit, 패리티 없음, 흐름제어 없음으로 설정

2. 위에서 작성한 소스 코드를 컴파일하고 실습보드로 다운로드한 후에 프로그램을 실행

3. 하이퍼터미널 창에서 “UART Example 1 (Transmission Success !!)” 초기메시지를 확인할 수 있음

예제 1 실행 결과

KYUNG HEE UNIVERSITY

11.4 UART 활용 예제

UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

- 사용자가 스위치를 누르면 UART를 통하여 PC로 대응되는 메시지를 보내는 예제
- 스위치 SW가 눌러지면 대응하는 외부 인터럽트를 발생시키고, 인터럽트 서비스 루틴에서는 해당 EXTI에 대응하는 메시지를 PC로 보냄
- UART는 보 레이트 9600, 데이터 비트 8bit, 정지 비트 1bit, 패리티 없음, 흐름제어 없음으로 설정
- PC에서는 하이퍼터미널 프로그램을 이용하여 통신함

11.4 UART 활용 예제

UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

- 폴더 명 : [Examples_Nucleo F103] - [UART] - [UART 2_F103] - [MDK-ARM]
- 위의 폴더에 있는 [Project.uvprojx] 파일을 더블클릭하여 실행함

11.4 UART 활용 예제

UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

[main.c]

```
#include "main.h"
#include "Nucleo_F103.h"            // Nucleo-F103 확장보드용 헤더 파일
// #include "Nucleo_F429.h"         // Nucleo-F429 확장보드용 헤더 파일

// UartHandle 변수를 외부참조 변수로 선언
extern UART_HandleTypeDef UartHandle;

// -- UART 통신을 위한 정의
#define TxBufferSize     (countof(TxBuffer) - 1)
#define TxBufferSize_2   (countof(TxBuffer_2) - 1)
#define TxBufferSize_3   (countof(TxBuffer_3) - 1)
#define TxBufferSize_4   (countof(TxBuffer_4) - 1)
#define TxBufferSize_5   (countof(TxBuffer_5) - 1)
#define RxBufferSize     0xFF
#define countof(a)       (sizeof(a) / sizeof(*(a)))

// UART 통신용 변수 선언
uint8_t TxBuffer[]   = "\n\rUART Example 2 (Transmission  Success !!)\n\r\n\r";
uint8_t TxBuffer_2[] = "\n\r Button1 is pressed !! \n\r";
uint8_t TxBuffer_3[] = "\n\r Button2 is pressed !! \n\r";
uint8_t TxBuffer_4[] = "\n\r Button3 is pressed !! \n\r";
uint8_t TxBuffer_5[] = "\n\r Button4 is pressed !! \n\r";
uint8_t RxBuffer[RxBufferSize];
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  SwEXTI_Config();
  LED_Config();
  // UART의 초기설정용 함수를 호출
  UART2_Config();       // Nucleo-F103 확장보드 사용 시 함수 호출
  // UART3_Config();    // Nucleo-F429 확장보드 사용 시 함수 호출

  LED_OnOff(GPIO_PIN_LedAll, SOO);

  // TxBuffer에 저장되어 있는 내용을 PC로 보낸다.
  HAL_UART_Transmit(&UartHandle, (uint8_t*)TxBuffer, TxBufferSize, 0xFFFF);

  while (1) { }
}

// ------------------------------------------------------------------------ //
// -- <1> SW의 EXTI 입력 처리용 callback 함수
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  // -- <2> Nucleo-64 보드의 SW(B1), SW1 ~ SW4를 누르면 해당 메시지를 PC로 송신한다.
  if ( GPIO_Pin == GPIO_PIN_Nucleo_Sw)
      HAL_UART_Transmit(&UartHandle, (uint8_t*)TxBuffer_2, TxBufferSize_2 , 0xFFFF);
  if ( GPIO_Pin == GPIO_PIN_Sw1)
      HAL_UART_Transmit(&UartHandle, (uint8_t*)TxBuffer_2, TxBufferSize_2 , 0xFFFF);
  if ( GPIO_Pin == GPIO_PIN_Sw2)
      HAL_UART_Transmit(&UartHandle, (uint8_t*)TxBuffer_3, TxBufferSize_3 , 0xFFFF);
  if ( GPIO_Pin == GPIO_PIN_Sw3)
      HAL_UART_Transmit(&UartHandle, (uint8_t*)TxBuffer_4, TxBufferSize_4 , 0xFFFF);
  if ( GPIO_Pin == GPIO_PIN_Sw4)
      HAL_UART_Transmit(&UartHandle, (uint8_t*)TxBuffer_5, TxBufferSize_5 , 0xFFFF);
}
```

스위치를 누르면 PC로 메시지 보내기 예제 코드

11.4 UART 활용 예제

UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

- **UART 인터럽트 핸들러 함수는 앞의 [UART 예제 1]과 동일**

[stm32f1xx_it.c]

UART 예제 1의 < stm32F1xx_it.c >와 동일함

KYUNG HEE UNIVERSITY

11.4 UART 활용 예제

UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

- **소스 코드 주석 설명**

[main.c]

```
<1> SW의 EXTI 입력 처리용 callback 함수이다.
<2> SW를 누르면 해당 메시지를 PC로 송신한다.
        B1(User Button, 파란색)을 누르면 "Button1 is pressed !! " 메시지가 전송된다.
        SW1을 누르면 "Button1 is pressed !! " 메시지가 전송된다.
        SW2를 누르면 "Button2 is pressed !! " 메시지가 전송된다.
        SW3을 누르면 "Button3 is pressed !! " 메시지가 전송된다.
        SW4를 누르면 "Button4 is pressed !! " 메시지가 전송된다.
```

11.4 UART 활용 예제

UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

- **프로그램 실행**

  1. **PC에서 하이퍼터미널 프로그램을 실행**
     하이퍼 터미널은 실습보드와 동일하게 보 레이트 9600, 데이터 비트 8bit, 정지 비트 1bit, 패리티 없음, 흐름제어 없음으로 설정함

  2. **위에서 작성한 소스 코드를 컴파일하고 실습보드로 다운로드한 후에 프로그램을 실행**

  3. **하이퍼터미널 창에서 “UART Example 2 (Transmission Success !!)”라는 메시지를 확인할 수 있음**

  4. **B1, SW1~SW4를 누르면 각각 이에 대응되는 메시지가 하이퍼터미널 창에 나타나는 것을 알 수 있음**

11.4 UART 활용 예제

UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

예제 2의 실행 결과

KYUNG HEE UNIVERSITY

11.4 UART 활용 예제

UART 예제 3 : UART를 이용하여 PC와 통신하기(폴링 방식)

- PC에서 실습보드로 메시지를 보내고, 실습보드는 받은 메시지를 다시 PC로 보내는 예제
- 실습 보드는 UART의 TX, RX 핀을 이용하여 PC와 UART 통신을 함
- UART는 보 레이트 9600, 데이터 비트 8bit, 정지 비트 1bit, 패리티 없음, 흐름제어 없음으로 설정함
- PC에서는 하이퍼터미널 프로그램을 이용하여 통신을 함
- 이 예제는 UART 통신과 관련된 인터럽트를 사용하지 않는 폴링 방식으로 동작함

예제 동작 순서

1) 실습 보드에서 프로그램이 실행되거나 리셋 스위치를 누르면 다음의 메시지가 PC의 하이퍼터미널로 전송된다.

UART Example 3 (Transmission Sucess!!)

2) PC의 하이퍼터미널 프로그램에서 키보드를 이용하여 1개의 문자를 실습 보드로 보내면, 실습 보드는 받은 문자를 그대로 다시 PC로 전송한다.

11.4 UART 활용 예제

UART 예제 3 : UART를 이용하여 PC와 통신하기(폴링 방식)

- 폴더 명 : [Examples_Nucleo F103] - [UART] - [UART 3_F103] - [MDK-ARM]
- 위의 폴더에 있는 [Project.uvprojx] 파일을 더블클릭하여 실행함

11.4 UART 활용 예제

UART 예제 3 : UART를 이용하여 PC와 통신하기(폴링 방식)

[main.c]

```
#include "main.h"
#include "Nucleo_F103.h"              // Nucleo-F103 확장보드용 헤더 파일
// #include "Nucleo_F429.h"           // Nucleo-F429 확장보드용 헤더 파일

// UartHandle 변수를 외부정의 변수로 선언
extern UART_HandleTypeDef UartHandle;

// UART 통신을 위한 정의
#define TxBufferSize      (countof(TxBuffer) - 1)
#define RxBufferSize      0xFF
#define countof(a)        (sizeof(a) / sizeof(*(a)))

// UART 통신용 변수 선언
uint8_t TxBuffer[] = "\n\rUART Example 3 (Transmission  Success !!)\n\r\n\r";
uint8_t RxBuffer[RxBufferSize];

//-------------------------------------------------------------------------------
//
```

```
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  LED_Config();

  // UART의 초기설정용 함수를 호출
  UART2_Config();        // Nucleo-F103 확장보드 사용 시 함수 호출
  // UART3_Config();     // Nucleo-F429 확장보드 사용 시 함수 호출

  LED_OnOff(GPIO_PIN_LedAll, 500);

  // -- <1> TxBuffer에 저장되어 있는 내용을 PC로 보낸다.
  HAL_UART_Transmit(&UartHandle, (uint8_t*)TxBuffer, TxBufferSize, 0xFFFF);

  // -- <2> PC에서 데이터를 받아 그대로 PC로 전송(echo) 한다.
  while (1)  {
    // -- <3> PC로부터 받은 데이터를 RxBuffer에 저장한다.
    if(HAL_UART_Receive(&UartHandle, (uint8_t*)RxBuffer,1, 5000)== HAL_OK)
    {
      // --<4> RxBuffer를 PC로 다시 보낸다.
      HAL_UART_Transmit(&UartHandle, (uint8_t*)RxBuffer, 1, 5000);
    }
  }
}
```

UART를 이용하여 PC와 통신하기(폴링 방식) 예제 코드

11.4 UART 활용 예제

UART 예제 3 : UART를 이용하여 PC와 통신하기(폴링 방식)

- **UART 인터럽트 핸들러 함수는 앞의 [UART 예제 1]과 동일**

[stm32f1xx_it.c]

UART 예제 1의 < stm32F1xx_it.c >와 동일함

KYUNG HEE UNIVERSITY

11.4 UART 활용 예제

UART 예제 3 : UART를 이용하여 PC와 통신하기(폴링 방식)

- **소스 코드 주석 설명**

[main.c]

<1> : TxBuffer에 저장되어 있는 내용을 PC로 보낸다.
<2> : PC에서 데이터를 받아 그대로 PC로 전송(echo)하는 무한루프이다.
<3> : PC로부터 받은 데이터를 RxBuffer에 담는다.
<4> : RxBuffer에 저장되어 있는 내용을 PC로 다시 보낸다.

11.4 UART 활용 예제

UART 예제 3 : UART를 이용하여 PC와 통신하기(폴링 방식)

- **프로그램 실행**

1. **PC에서 하이퍼터미널 프로그램을 실행**
   하이퍼 터미널은 실습보드와 동일하게 보 레이트 9600, 데이터 비트 8bit, 정지 비트 1bit, 패리티 없음, 흐름제어 없음으로 설정함

2. **위에서 작성한 소스 코드를 컴파일하고 실습보드로 다운로드한 후에 프로그램을 실행**

3. **하이퍼터미널 창에서 초기 메시지를 확인한 후, 'A'를 입력하면 실습보드가 메시지를 입력 받고 다시 PC로 메시지를 보냄**
   화면에 나타나는 'A'라는 메시지는 "PC→실습보드→PC"의 경로로 UART 통신을 한 결과 값

11.4 UART 활용 예제

UART 예제 3 : UART를 이용하여 PC와 통신하기(폴링 방식)

예제 3의 실행 결과

KYUNG HEE UNIVERSITY

11.4 UART 활용 예제

UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)

- PC에서 실습보드로 메시지를 보내고, 실습보드는 받은 메시지를 다시 PC로 보내는 예제
- 실습 보드는 UART의 TX, RX 핀을 이용하여 PC와 UART 통신을 함
- UART는 보 레이트 9600, 데이터 비트 8bit, 정지 비트 1bit, 패리티 없음, 흐름제어 없음으로 설정함
- PC에서는 하이퍼터미널 프로그램을 이용하여 통신을 함
- 이 예제는 UART 통신과 관련된 인터럽트를 사용하는 인터럽트 방식으로 동작함

11.4 UART 활용 예제

UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)

- 폴더 명 : [Examples_Nucleo F103] - [UART] - [UART 4_F103] - [MDK-ARM]
- 위의 폴더에 있는 [Project.uvprojx] 파일을 더블클릭하여 실행함

11.4 UART 활용 예제

UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)

[main.c]

```
#include "main.h"
#include "Nucleo_F103.h"            // Nucleo-F103 확장보드용 헤더 파일
// #include "Nucleo_F429.h"         // Nucleo-F429 확장보드용 헤더 파일

// UartHandle 변수를 외부정의 변수로 선언
extern UART_HandleTypeDef  UartHandle;

// UART 통신용 변수 선언
#define TxBufferSize       (countof(TxBuffer) - 1)
#define TxBufferSize_3     (countof(TxBuffer_3) - 1)
#define RxBufferSize       0xFF
#define countof(a)         (sizeof(a) / sizeof(*(a)))

// UART 통신을 위한 전역
uint8_t TxBuffer[] = "\n\rUART Example 4 (Transmission  Success !!)\n\r\n\r";
uint8_t TxBuffer_3[] = "\n\r A Message from HAL_UART_TxCpltCallback() !!\n\r\n\r";
uint8_t RxBuffer[RxBufferSize];

// ------------------------------------------------------------------------------ //

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  LED_Config();
  UART2_Config();       // Nucleo-F103 확장보드 사용 시 함수 호출
  // UART3_Config();    // Nucleo-F429 확장보드 사용 시 함수 호출
```

```
  // TxBuffer에 저장되어 있는 내용을 PC로 보낸다.
  HAL_UART_Transmit(&UartHandle, (uint8_t*)TxBuffer, TxBufferSize , 0xFFFF);

  while (1)
  {
    // -- <1> PC로부터 데이터를 수신할 때 인터럽트로 구현
    HAL_UART_Receive_IT(&UartHandle, (uint8_t*)RxBuffer,1);
  }

}

// ------------------------------------------------------------------------------ //
// -- <2> Interrupt 모드 송신이 완료되면 호출되는 callback 함수
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *UartHandler)
{
  // -- <3> 송신(INT 모드)이 완료되면 TxBuffer_3에 저장된 메시지를 PC로 송신
  HAL_UART_Transmit(UartHandler, (uint8_t*)TxBuffer_3, TxBufferSize_3 , 0xFFFF);
}

// ------------------------------------------------------------------------------ //
// -- <4> Interrupt 모드 수신이 완료되면 호출되는 callback 함수
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *UartHandler)
{
  /* -- <5> 수신(INT 모드)이 완료되면 수신된 데이터(RxBuffer)를 그대로 송신 */
  HAL_UART_Transmit_IT(UartHandler, (uint8_t*)RxBuffer, 1);
}
```

UART를 이용하여 PC와 통신하기(인터럽트 방식) 예제 코드

11.4 UART 활용 예제

UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)

- UART 인터럽트 핸들러 함수는 앞의 [UART 예제 1]과 동일

[stm32f1xx_it.c]

UART 예제 1의 < stm32F1xx_it.c >와 동일함

11.4 UART 활용 예제

UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)

- **소스 코드 주석 설명**

[main.c]

<1> HAL_UART_Receive_IT() 함수 : PC로부터 데이터를 수신할 때 버퍼 사이즈는 1이며 Interrupt 모드로 구현이 된다.
<2> Interrupt 모드 송신이 완료되면 호출되는 callback 함수이다.
<3> 송신(INT 모드)이 완료되면 TxBuffer_3에 저장된 메시지를 PC로 송신한다.
<4> Interrupt 모드 수신이 완료되면 호출되는 callback 함수이다.
<5> HAL_UART_Transmit_IT() 함수 : 수신(INT 모드)이 완료되면 수신된 데이터(RxBuffer)를 그대로 송신한다. 송신할 때 버퍼 사이즈는 1이며 Interrupt모드로 구현이 된다.

11.4 UART 활용 예제

UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)

- **프로그램 실행**

  1. **PC에서 하이퍼터미널 프로그램을 실행**
     하이퍼 터미널은 실습보드와 동일하게 보 레이트 9600, 데이터 비트 8bit, 정지 비트 1bit, 패리티 없음, 흐름제어 없음으로 설정함

  2. **위에서 작성한 소스 코드를 컴파일하고 실습보드로 다운로드한 후에 프로그램을 실행**

  3. **하이퍼터미널 창에서 초기 메시지를 확인한 후, 'A'를 입력하면 실습보드가 메시지를 입력 받고 송신 인터럽트를 발생하여 해당 메시지를 보냄**

11.4 UART 활용 예제

UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)

예제 4의 실행 결과

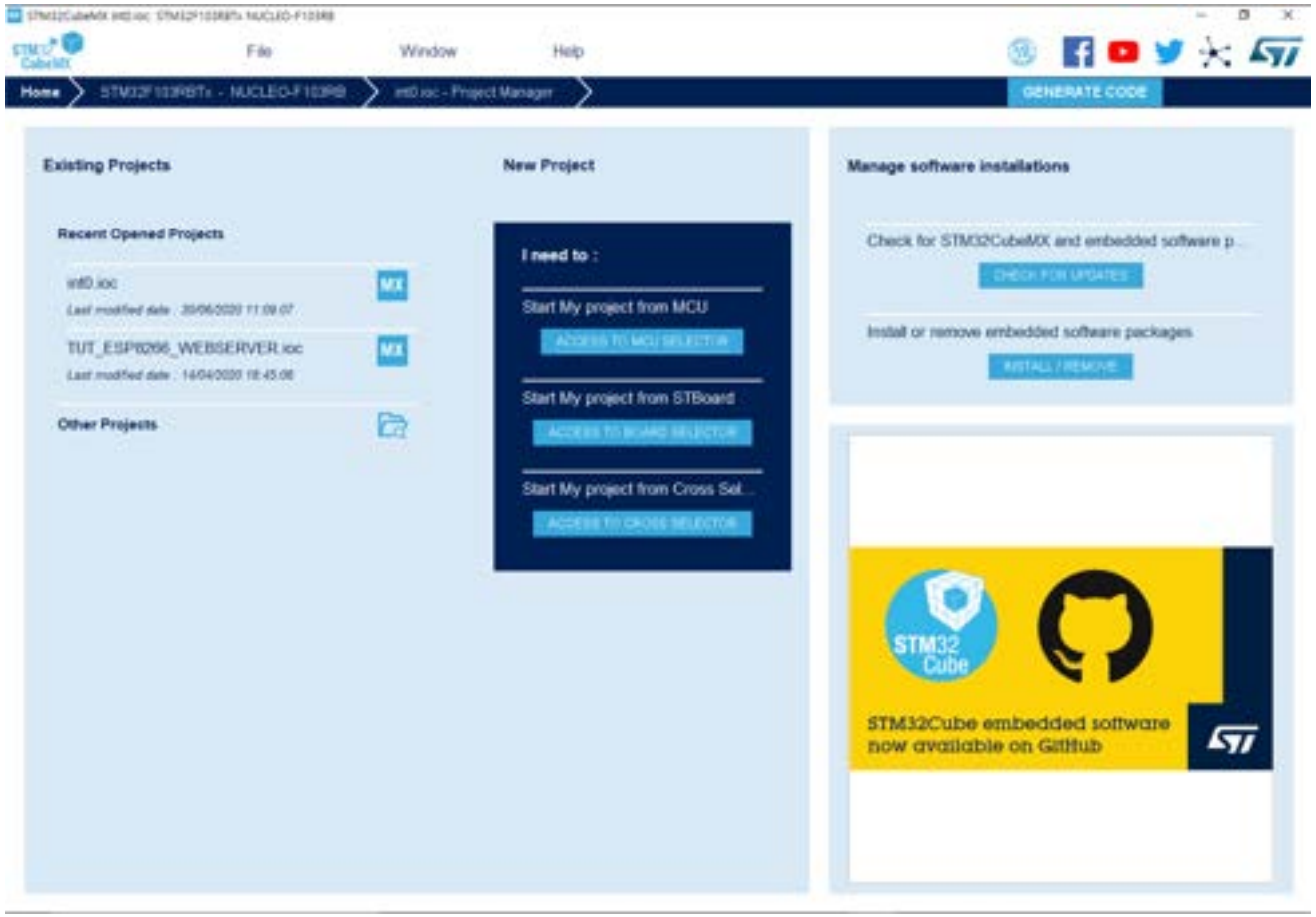
KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 예제 11.1 구현하기

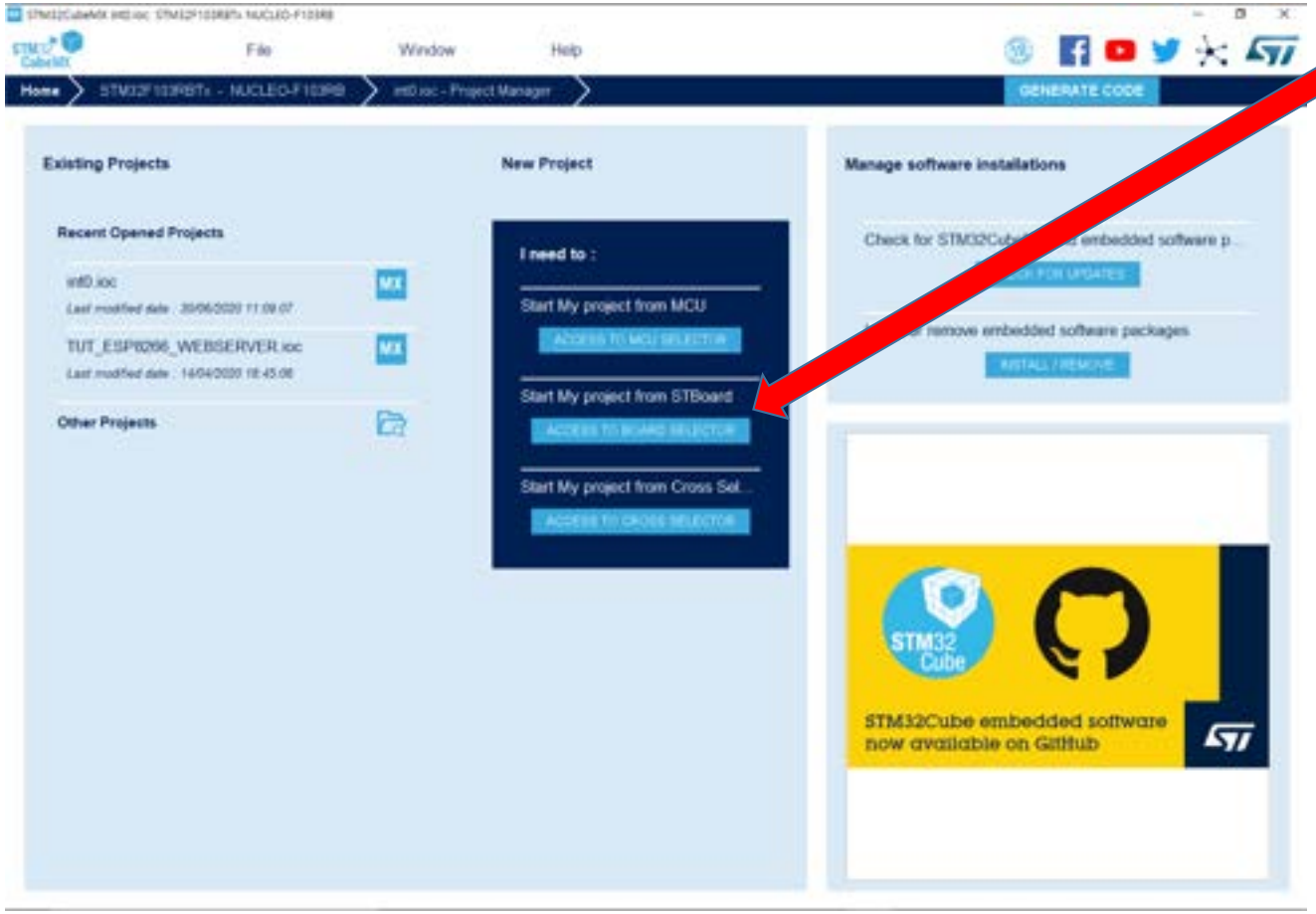
CubeMX 실행

초기화면



11.5 UART CubeMX과제

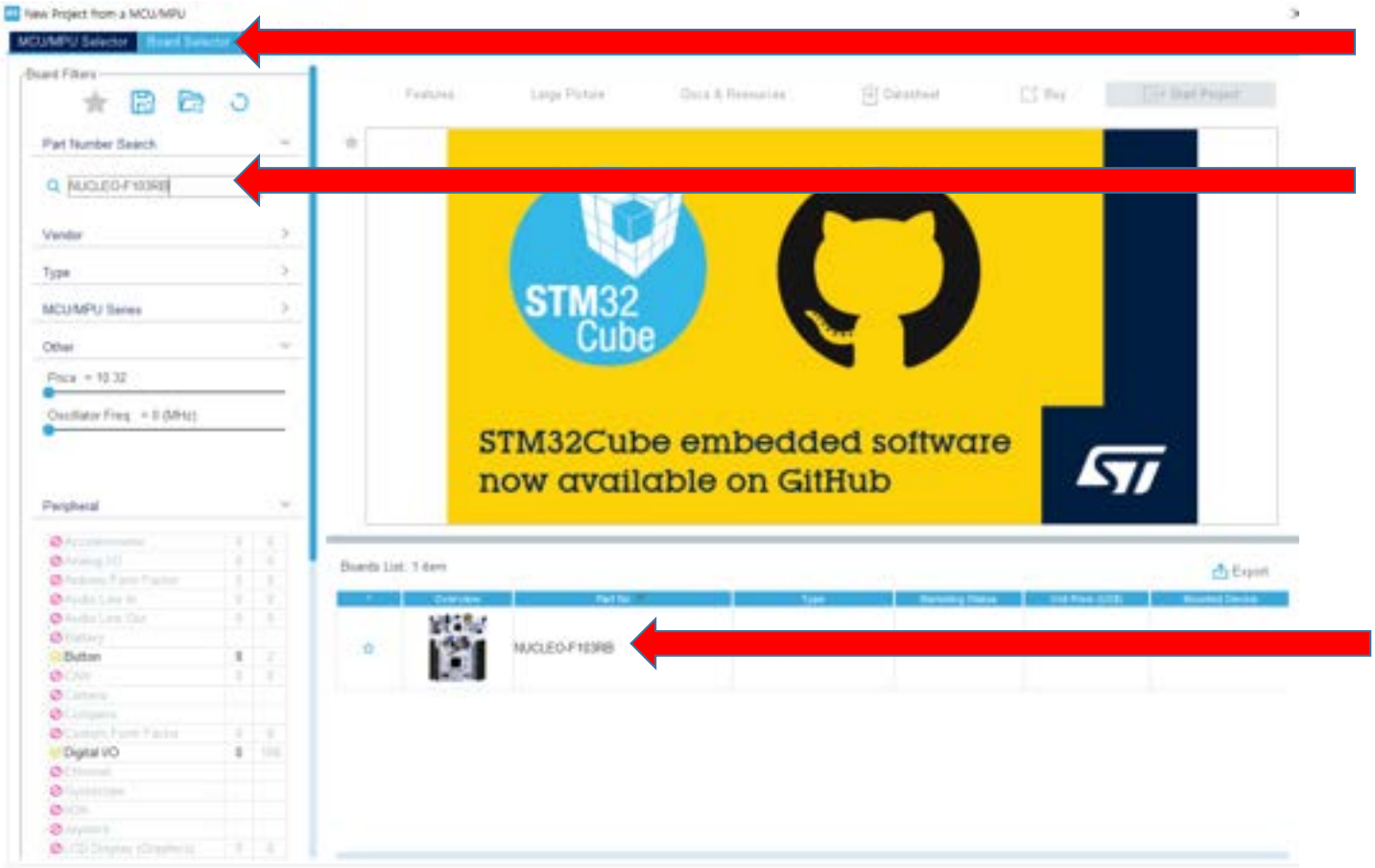
CubeMX로 예제 11.1 구현하기



보드 선택

11.5 UART CubeMX과제

CubeMX로 예제 11.1 구현하기



보드 선택

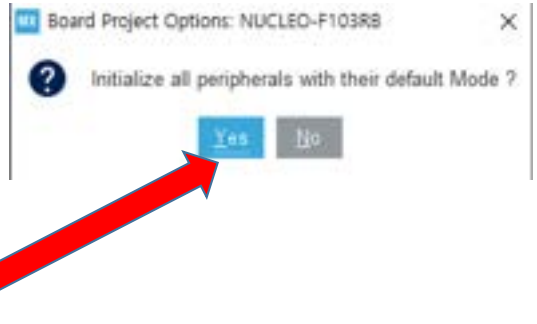
NUCLEO-F103RB 선택

NUCLEO-F103RB 더블클릭

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

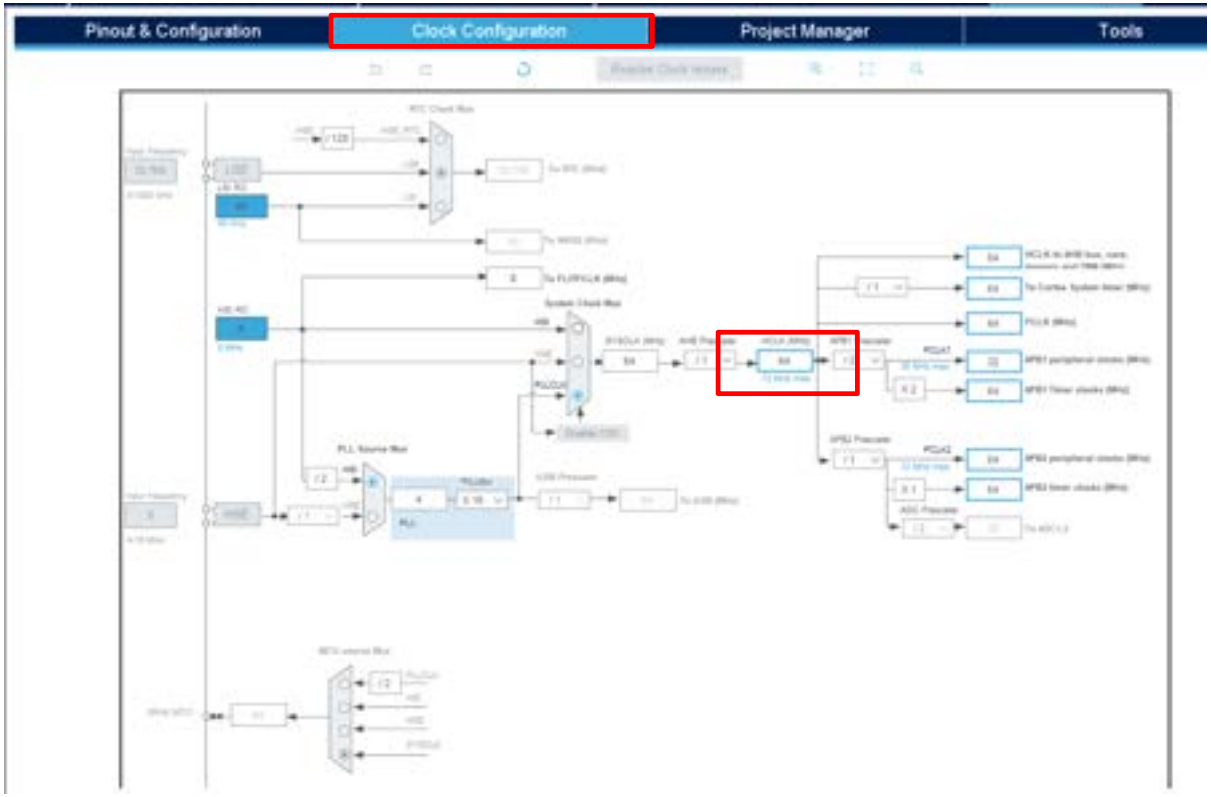
CubeMX로 예제 11.1 구현하기



Yes 선택하면 오른쪽과 같이 칩이 보임

11.5 UART CubeMX과제

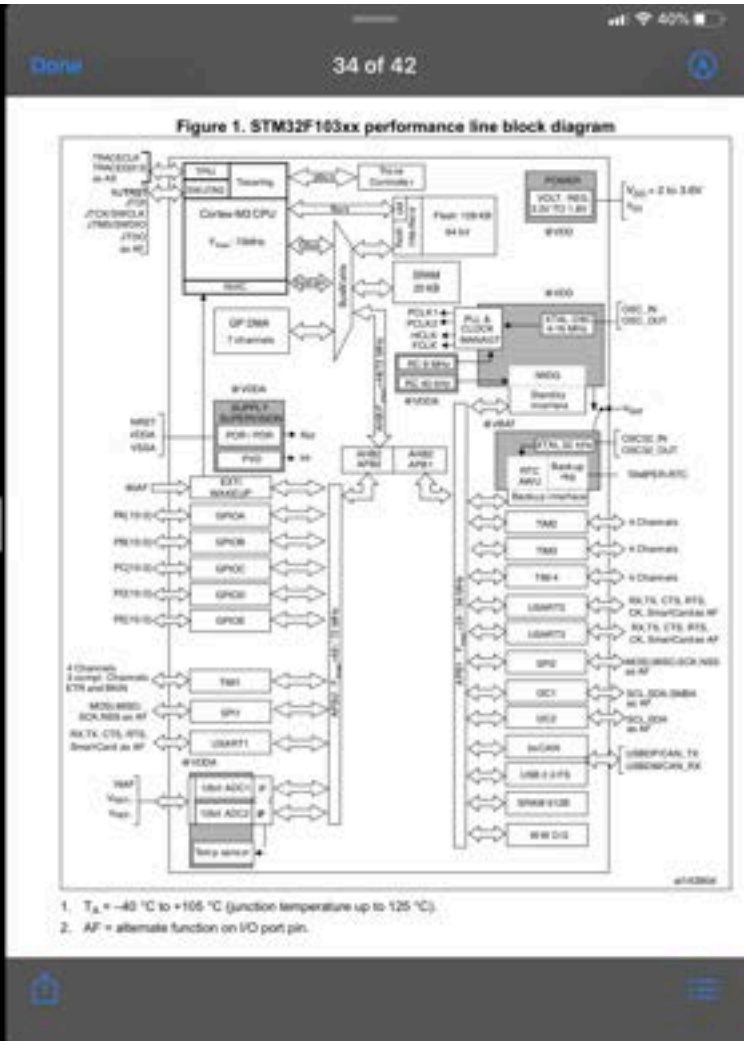
CubeMX로 UART 예제 1 : UART를 이용하여 PC에 메시지 보내기



Clock configuration 탭에서 Main Clk를 설정 및 확인 → 64MHz

11.5 UART CubeMX과제

```
 *          PLLMUL                  = 16
 *          Flash Latency(WS)       = 2
 */
// ------------------------------------------------------------------------- //

void  SystemClock_Config(void)
{
          RCC_ClkInitTypeDef clkinitstruct = {0};
          RCC_OscInitTypeDef oscinitstruct = {0};

          /* Configure PLL ------------------------------------------------------*/
          /* PLL configuration: PLLCLK = (HSI / 2) * PLLMUL = (8 / 2) * 16 = 64 MHz */
          /* PREDIV1 configuration: PREDIV1CLK = PLLCLK / HSEPredivValue = 64 / 1 = 64
MHz */
          /* Enable HSI and activate PLL with HSI_DIV2 as source */
          oscinitstruct.OscillatorType  = RCC_OSCILLATORTYPE_HSI;
          oscinitstruct.HSEState        = RCC_HSE_OFF;
          oscinitstruct.LSEState        = RCC_LSE_OFF;
          oscinitstruct.HSIState        = RCC_HSI_ON;
          oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
          oscinitstruct.HSEPredivValue    = RCC_HSE_PREDIV_DIV1;
          oscinitstruct.PLL.PLLState    = RCC_PLL_ON;
          oscinitstruct.PLL.PLLSource   = RCC_PLLSOURCE_HSI_DIV2;
          oscinitstruct.PLL.PLLMUL      = RCC_PLL_MUL16;

          if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)  {
                    /* Initialization Error */
                    while(1);
          }

          /* Select PLL as system clock source and configure the HCLK, PCLK1 and
PCLK2
              clocks dividers */
          clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
          clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
          clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
          clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
          clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;

          if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)  {
                    /* Initialization Error */
                    while(1);
          }
}

// ------------------------------------------------------------------------- //
//  --  <18>  Clock 설정시 에러가 발생하면 처리해주는 함수
/**
```



KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

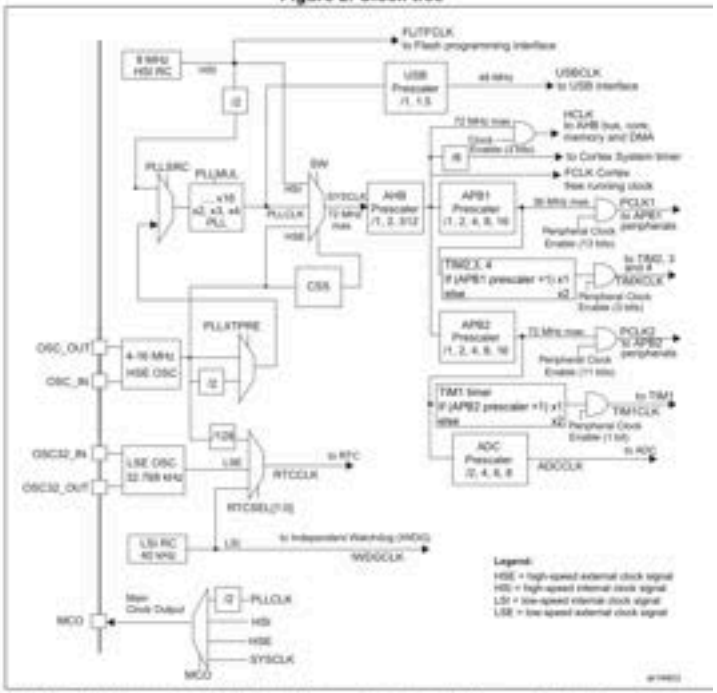
8:41 PM Mon Oct 28

Done 2 of 4

```
void SystemClock_Config(void)
{
  RCC_ClkInitTypeDef clkinitstruct = {0};
  RCC_OscInitTypeDef oscinitstruct = {0};

  /* Configure PLL ------------------------------------------------------*/
  /* PLL configuration: PLLCLK = (HSI / 2) * PLLMUL = (8 / 2) * 16 = 64 MHz */
  /* PREDIV1 configuration: PREDIV1CLK = PLLCLK / HSEPredivValue = 64 / 1 = 64
MHz */
  /* Enable HSI and activate PLL with HSi_DIV2 as source */
  oscinitstruct.OscillatorType  = RCC_OSCILLATORTYPE_HSI;
  oscinitstruct.HSEState        = RCC_HSE_OFF;
  oscinitstruct.LSEState        = RCC_LSE_OFF;
  oscinitstruct.HSIState        = RCC_HSI_ON;
  oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  oscinitstruct.HSEPredivValue    = RCC_HSE_PREDIV_DIV1;
  oscinitstruct.PLL.PLLState    = RCC_PLL_ON;
  oscinitstruct.PLL.PLLSource   = RCC_PLLSOURCE_HSI_DIV2;
  oscinitstruct.PLL.PLLMUL      = RCC_PLL_MUL16;

  if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)  {
      /* Initialization Error */
      while(1);
  }

  /* Select PLL as system clock source and configure the HCLK, PCLK1 and
PCLK2
      clocks dividers */
  clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
  clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
  clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)  {
      /* Initialization Error */
      while(1);
  }
}
```

Done 34 of 42

Figure 2. Clock tree

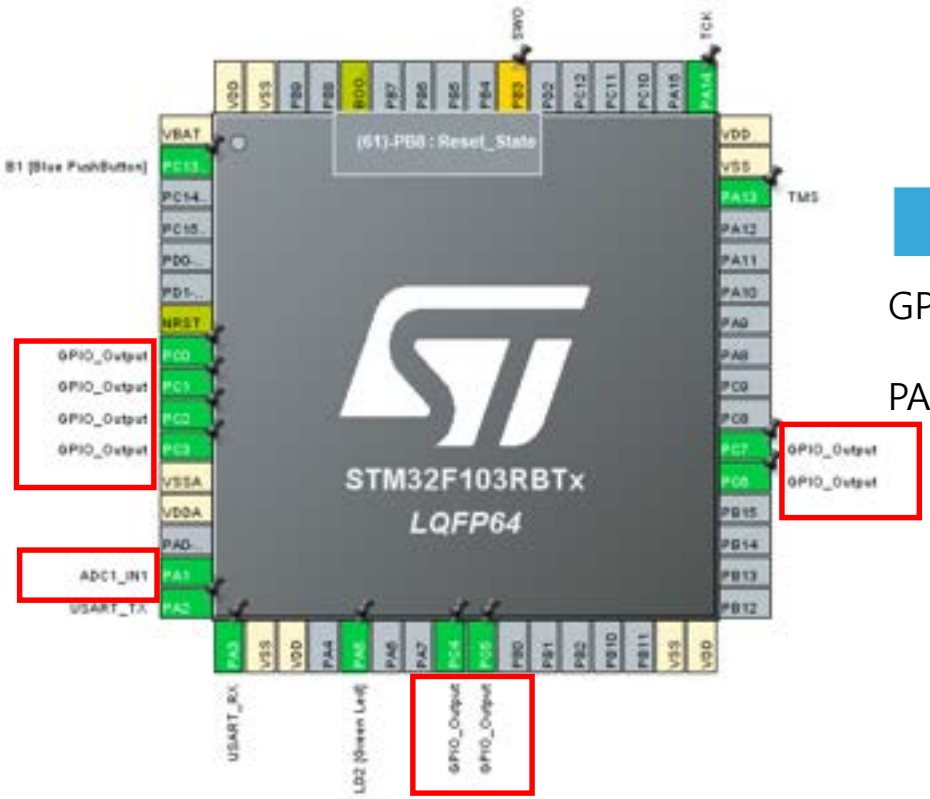


1. When the HSI is used as a PLL clock input, the maximum system clock frequency that can be achieved is 64 MHz.
2. For the USB function to be available, both HSE and PLL must be enabled, with USBCLK running at 48 MHz.
3. To have an ADC conversion time of 1 µs, APB2 must be at 14 MHz, 28 MHz or 56 MHz.

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 1 : UART를 이용하여 PC에 메시지 보내기



Pinout & Configuration 탭에서

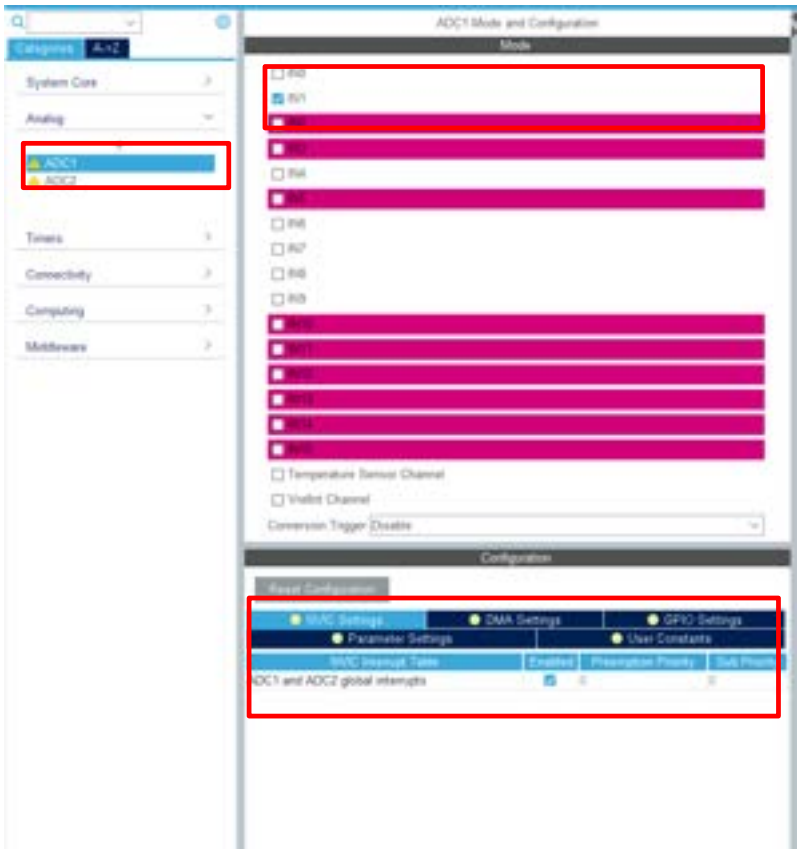
GPIO 출력핀 설정 : PC0~7에 LED 연결됨

PA1을 ADC1_IN1으로 설정함

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 1 : UART를 이용하여 PC에 메시지 보내기



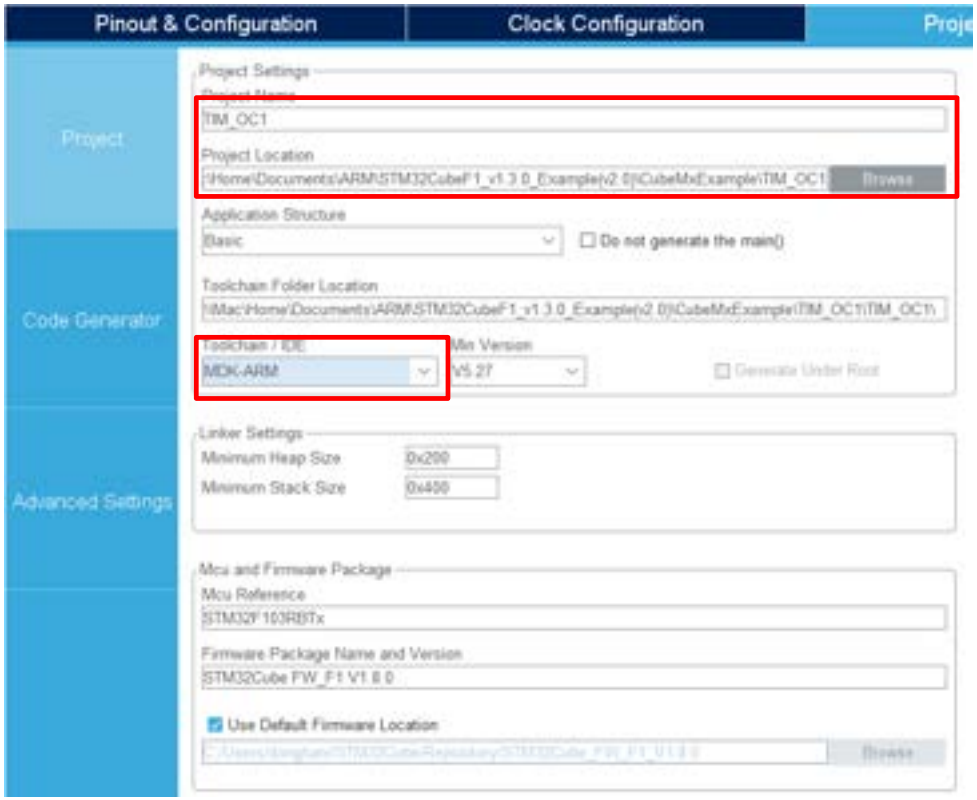
ADC1 중 IN1을 선택

Configuration에서 Parameter Settings 탭은 기본적으로 Scan, Continuous, Discontinuous Conversion Mode가 Disabled되어 있으므로 그대로 둠
NVIC 탭에서 ADC1 and ADC2 global interrupts를 Enabled 함

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 1 : UART를 이용하여 PC에 메시지 보내기



Project Manager 탭을 누르면, 왼쪽의 그림처럼 project 생성 단계가 되며, project name과 location을 정해줌.

Toolchain / IDE는 꼭 MDK-ARM을 선택

마지막으로 GENERATE CODE 탭을 누르면, 코드가 생성되면서 MDK uVision이 실행됨

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 1 : UART를 이용하여 PC에 메시지 보내기

```
60  void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
61  {
62      adcValue = HAL_ADC_GetValue(&hadc1) * 6000;
63      HAL_ADC_Start_IT(&hadc1);
64  }
```

→ main() 함수 앞 쪽에 ADC 변환 완료 콜백함수 선언하고 인터럽트 걸리면 해야 할 일 정의

```
117   while (1)
118   {
119     /* USER CODE END WHILE */
120
121     /* USER CODE BEGIN 3 */
122
123     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0 |  GPIO_PIN_1 |  GPIO_PIN_2 |  GPIO_PIN_3 |  GPIO_PIN_4 |
124                       GPIO_PIN_5 |  GPIO_PIN_6 |  GPIO_PIN_7, GPIO_PIN_SET);
125     for(delay_time = 0; delay_time < adcValue; delay_time++);
126     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0 |  GPIO_PIN_1 |  GPIO_PIN_2 |  GPIO_PIN_3 |  GPIO_PIN_4 |
127                       GPIO_PIN_5 |  GPIO_PIN_6 |  GPIO_PIN_7, GPIO_PIN_RESET);
128
129
130     HAL_Delay(1000);
131
132   }
```

→ 예제 9.1과 비교해보세요.

→ 최종 코드(교과서의 코드와 CubeMX의 코드)를 비교해보세요

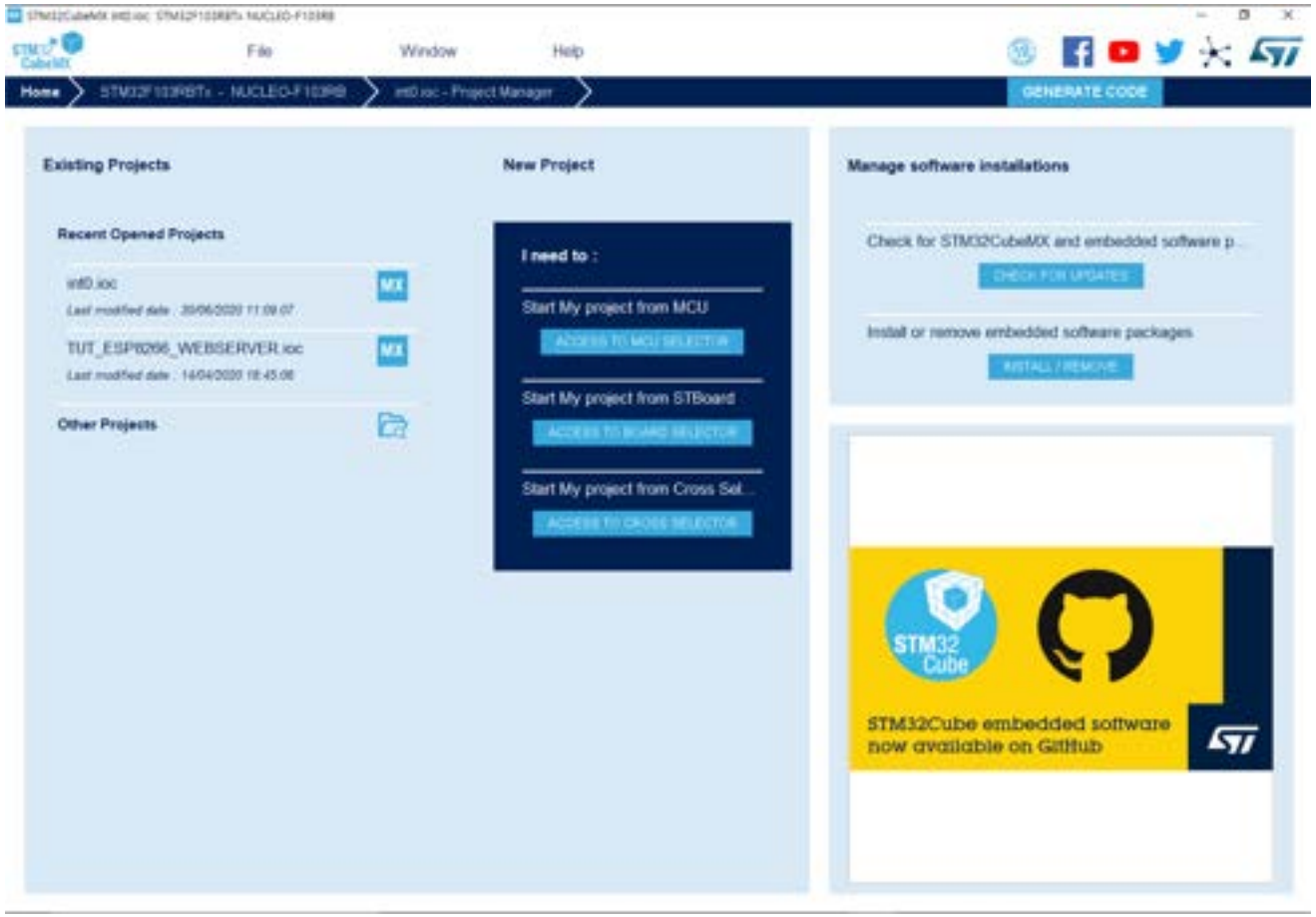
KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 예제 11.2 구현하기

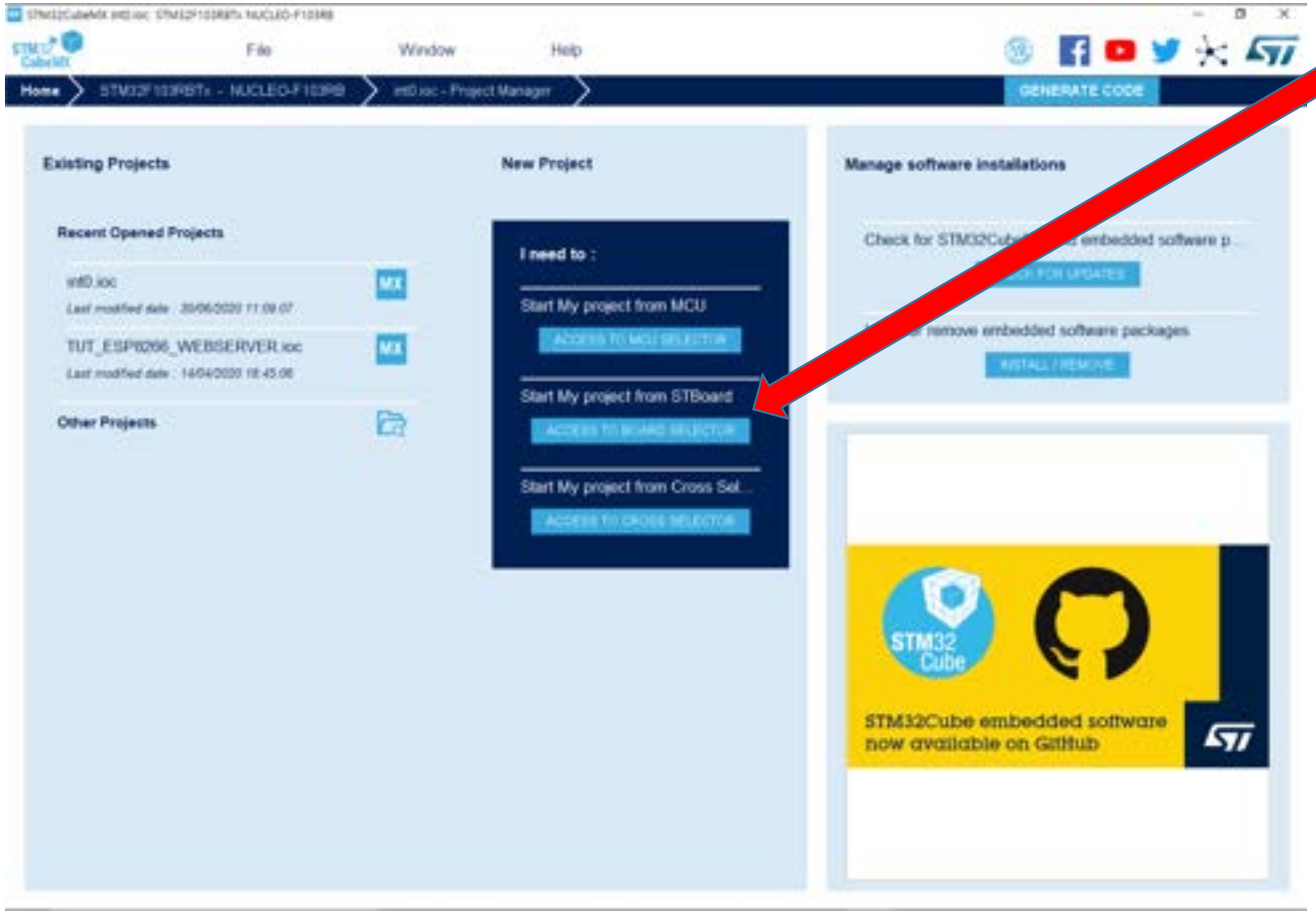
CubeMX 실행

초기화면



11.5 UART CubeMX과제

CubeMX로 예제 11.2구현하기

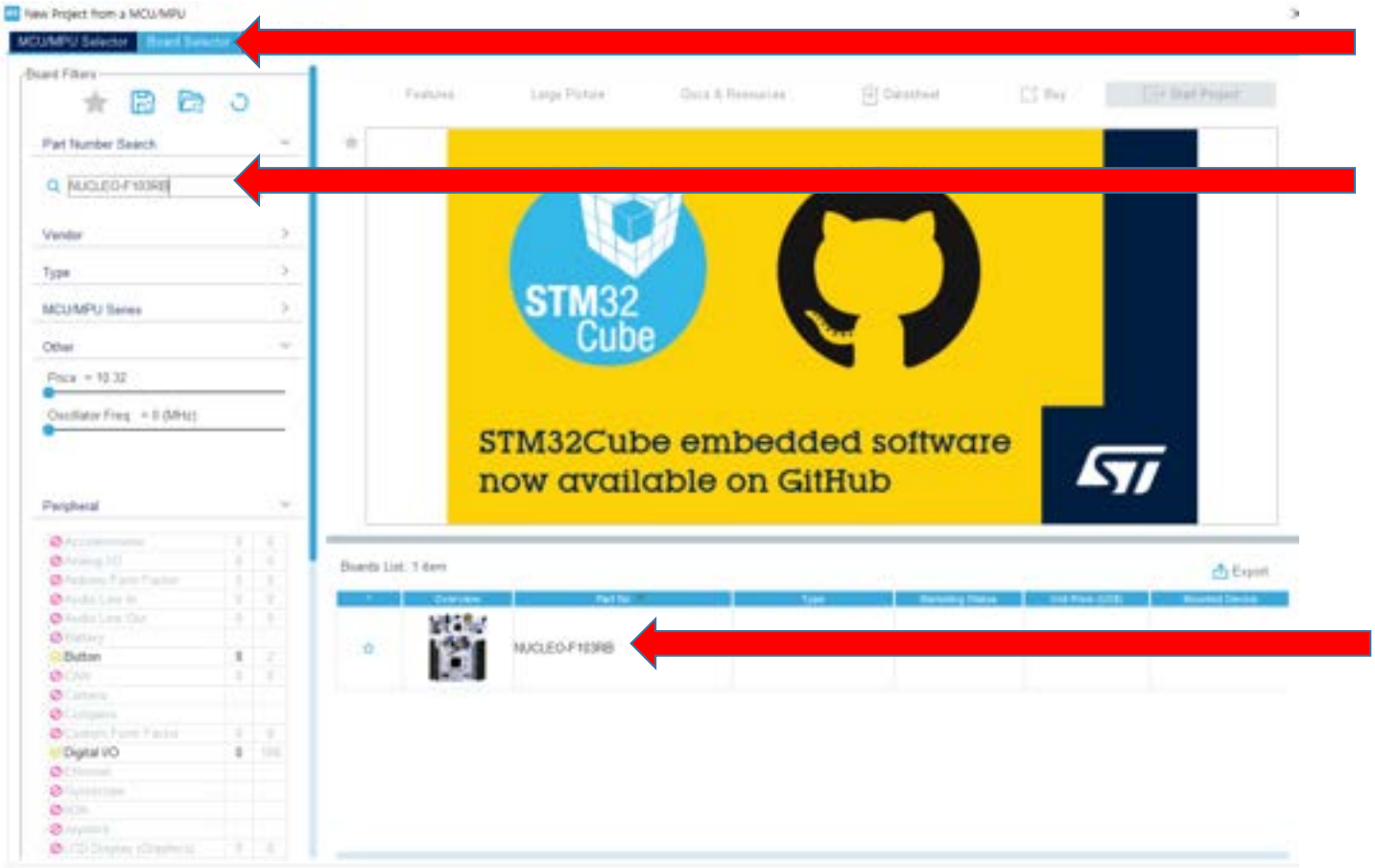


보드 선택

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 예제 11.2 구현하기



보드 선택

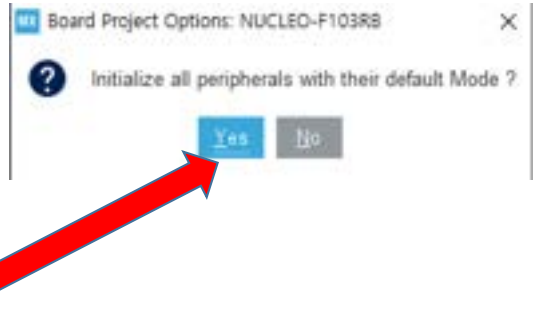
NUCLEO-F103RB 선택

NUCLEO-F103RB 더블클릭

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

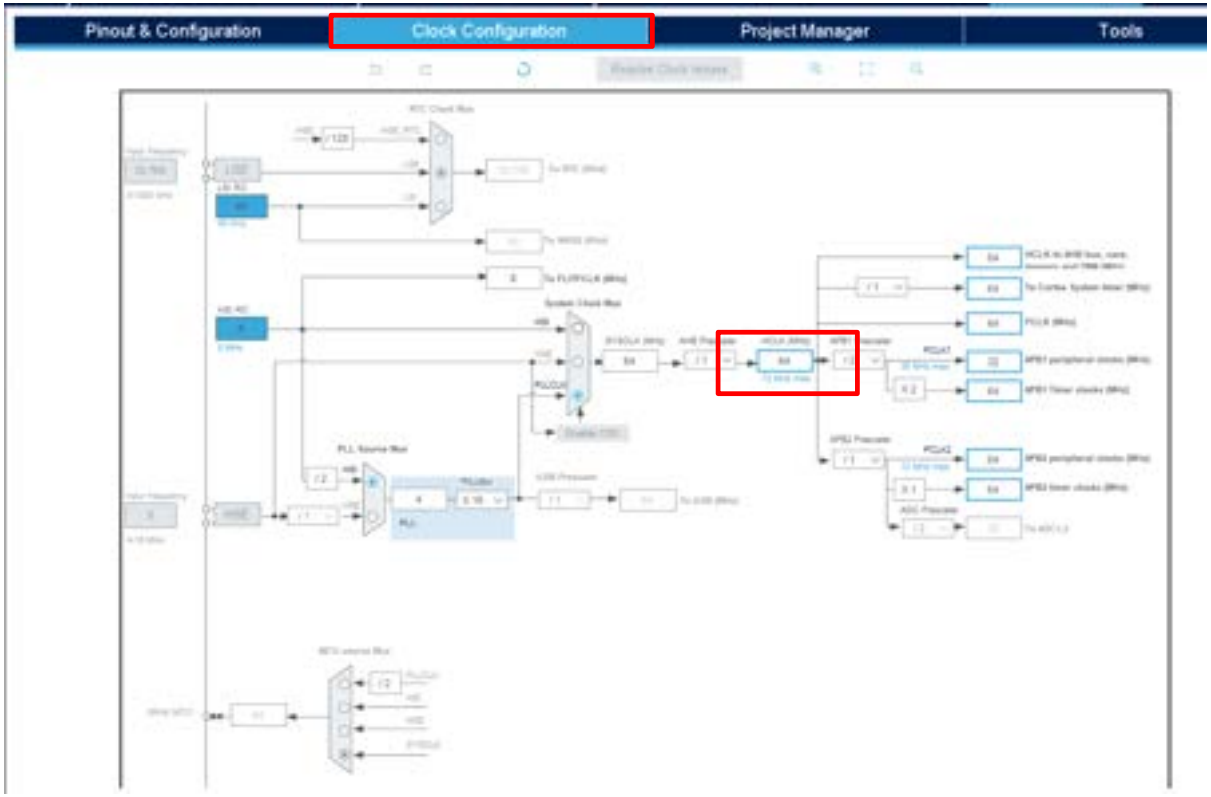
CubeMX로 예제 11.2 구현하기



Yes 선택하면 오른쪽과 같이 칩이 보임

11.5 UART CubeMX과제

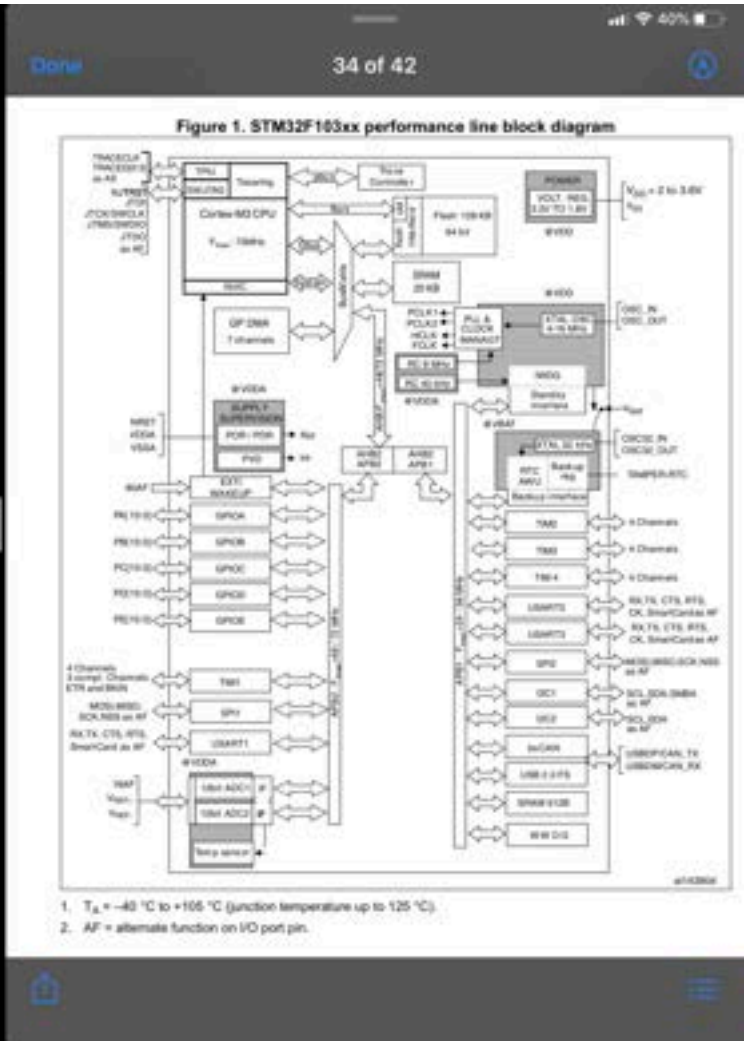
CubeMX로 UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기



Clock configuration 탭에서 Main Clk를 설정 및 확인 → 64MHz

11.5 UART CubeMX과제

```
 *          PLLMUL                  = 16
 *          Flash Latency(WS)       = 2
 */
// ------------------------------------------------------------------------- //

void  SystemClock_Config(void)
{
          RCC_ClkInitTypeDef clkinitstruct = {0};
          RCC_OscInitTypeDef oscinitstruct = {0};

          /* Configure PLL ------------------------------------------------------*/
          /* PLL configuration: PLLCLK = (HSI / 2) * PLLMUL = (8 / 2) * 16 = 64 MHz */
          /* PREDIV1 configuration: PREDIV1CLK = PLLCLK / HSEPredivValue = 64 / 1 = 64
MHz */
          /* Enable HSI and activate PLL with HSI_DIV2 as source */
          oscinitstruct.OscillatorType  = RCC_OSCILLATORTYPE_HSI;
          oscinitstruct.HSEState        = RCC_HSE_OFF;
          oscinitstruct.LSEState        = RCC_LSE_OFF;
          oscinitstruct.HSIState        = RCC_HSI_ON;
          oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
          oscinitstruct.HSEPredivValue    = RCC_HSE_PREDIV_DIV1;
          oscinitstruct.PLL.PLLState    = RCC_PLL_ON;
          oscinitstruct.PLL.PLLSource   = RCC_PLLSOURCE_HSI_DIV2;
          oscinitstruct.PLL.PLLMUL      = RCC_PLL_MUL16;

          if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)  {
                    /* Initialization Error */
                    while(1);
          }

          /* Select PLL as system clock source and configure the HCLK, PCLK1 and
PCLK2
              clocks dividers */
          clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
          clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
          clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
          clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
          clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;

          if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)  {
                    /* Initialization Error */
                    while(1);
          }
}

// ------------------------------------------------------------------------- //
//  --  <18>  Clock 설정시 에러가 발생하면 처리해주는 함수
/**
```



KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

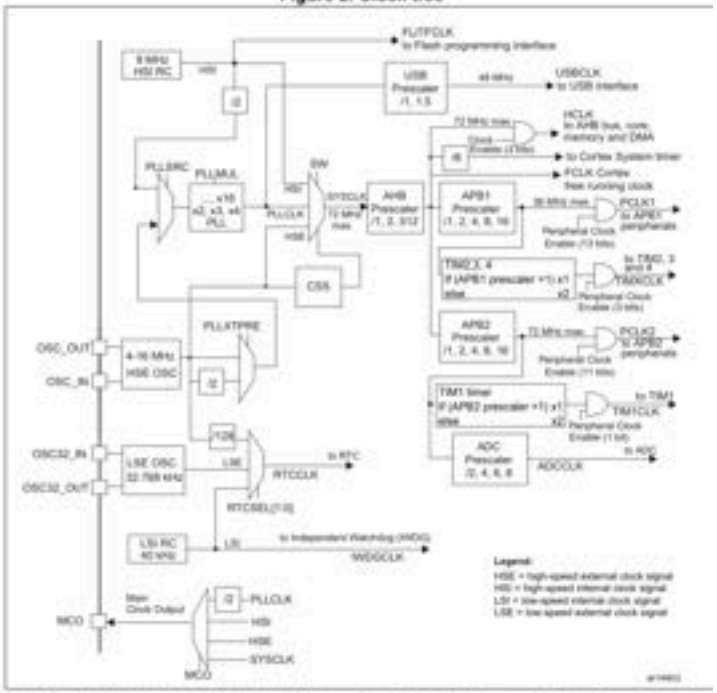
8:41 PM Mon Oct 28

Done 2 of 4

```
void SystemClock_Config(void)
{
  RCC_ClkInitTypeDef clkinitstruct = {0};
  RCC_OscInitTypeDef oscinitstruct = {0};

  /* Configure PLL ------------------------------------------------------*/
  /* PLL configuration: PLLCLK = (HSI / 2) * PLLMUL = (8 / 2) * 16 = 64 MHz */
  /* PREDIV1 configuration: PREDIV1CLK = PLLCLK / HSEPredivValue = 64 / 1 = 64
MHz */
  /* Enable HSI and activate PLL with HSi_DIV2 as source */
  oscinitstruct.OscillatorType  = RCC_OSCILLATORTYPE_HSI;
  oscinitstruct.HSEState        = RCC_HSE_OFF;
  oscinitstruct.LSEState        = RCC_LSE_OFF;
  oscinitstruct.HSIState        = RCC_HSI_ON;
  oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  oscinitstruct.HSEPredivValue    = RCC_HSE_PREDIV_DIV1;
  oscinitstruct.PLL.PLLState    = RCC_PLL_ON;
  oscinitstruct.PLL.PLLSource   = RCC_PLLSOURCE_HSI_DIV2;
  oscinitstruct.PLL.PLLMUL      = RCC_PLL_MUL16;

  if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)  {
      /* Initialization Error */
      while(1);
  }

  /* Select PLL as system clock source and configure the HCLK, PCLK1 and
PCLK2
      clocks dividers */
  clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
  clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
  clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)  {
      /* Initialization Error */
      while(1);
  }
}
```

Done 34 of 42

Figure 2. Clock tree

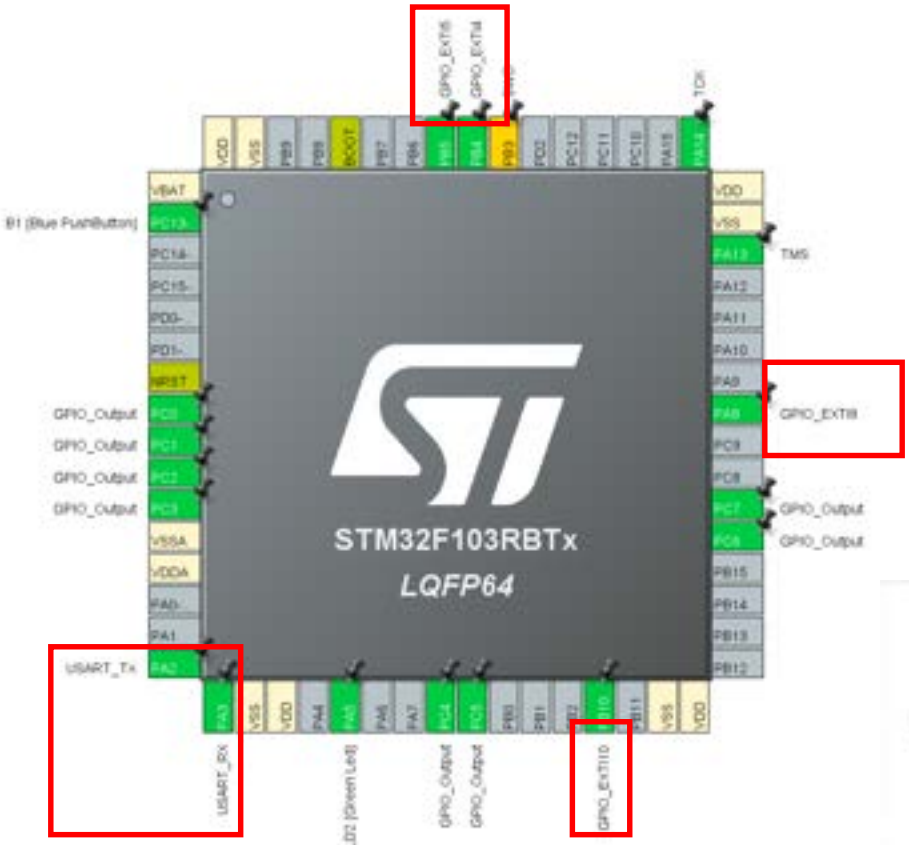


1. When the HSI is used as a PLL clock input, the maximum system clock frequency that can be achieved is 64 MHz.
2. For the USB function to be available, both HSE and PLL must be enabled, with USBCLK running at 48 MHz.
3. To have an ADC conversion time of 1 µs, APB2 must be at 14 MHz, 28 MHz or 56 MHz.

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

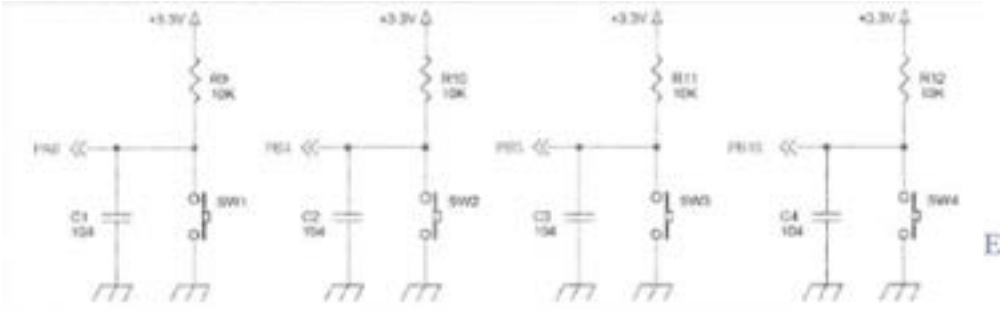


Pinout & Configuration 탭에서

GPIO 출력핀 설정 : PC0~7에 LED 연결됨

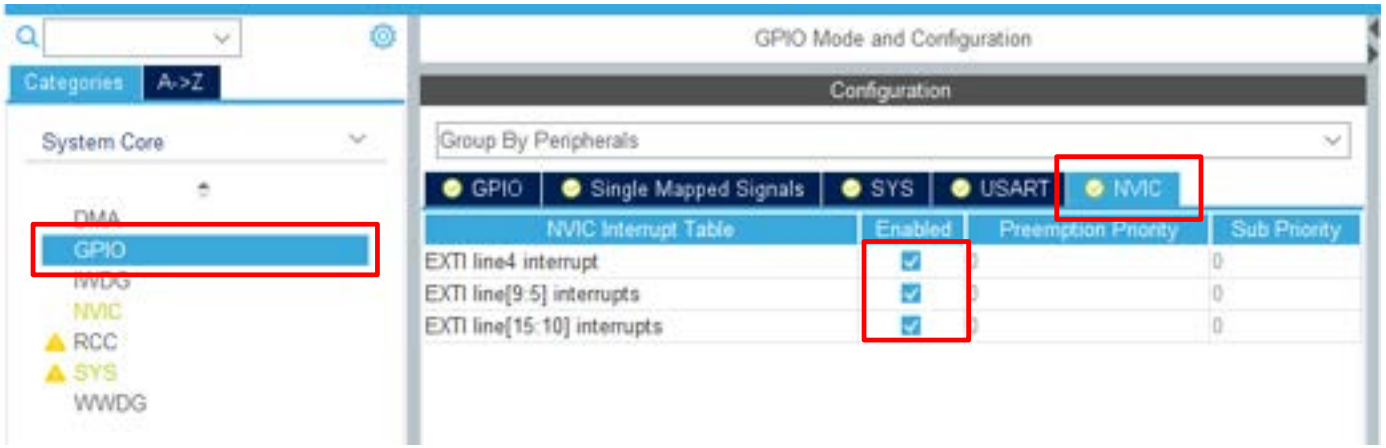
PA8, PB4, PB5, PB10을 GPIO_EXT로 설정함

UART2(PA2, PA3)를 설저함



11.5 UART CubeMX과제

CubeMX로 UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

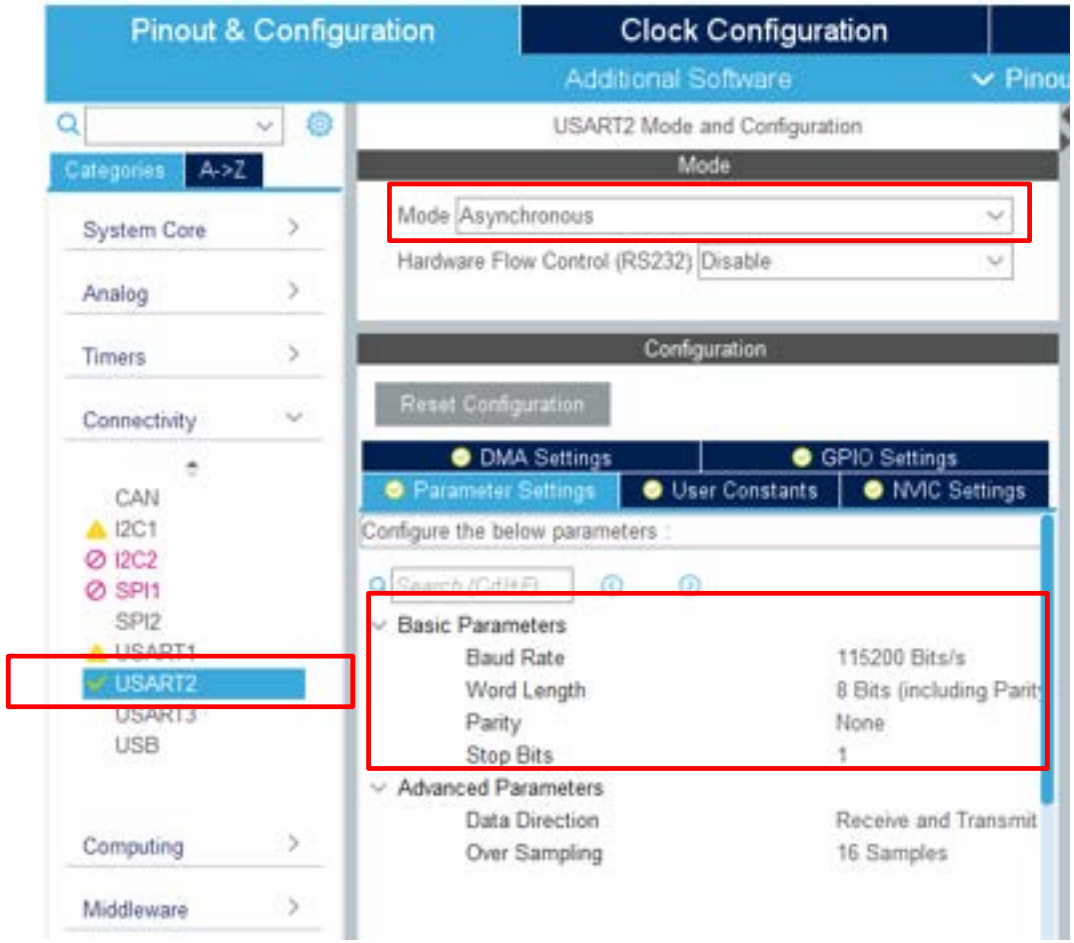


왼쪽 윈도우의 System Core 중에 GPIO를 선택하고, 왼쪽 윈도우의 Configuration에서 NVIC 탭에서 모든 interrupts를 Enabled 설정함

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

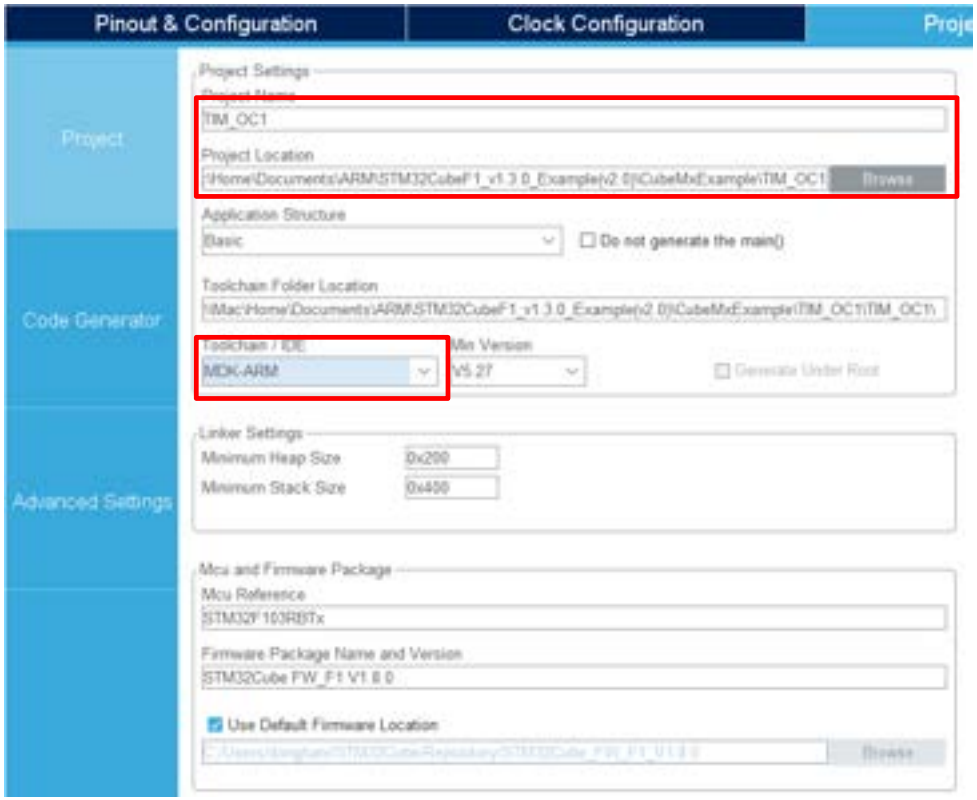


왼쪽 윈도우의 Connectivity중에 USART2를 선택하고, 왼쪽 윈도우의 Mode에서 Asynchronous 선택하고, Configuration에서 기본 설정된 값(Baud Rate, Word Length, Parity, Stop Bits)를 기억함

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기



Project Manager 탭을 누르면, 왼쪽의 그림처럼 project 생성 단계가 되며, project name과 location을 정해줌.

Toolchain / IDE는 꼭 MDK-ARM을 선택

마지막으로 GENERATE CODE 탭을 누르면, 코드가 생성되면서 MDK uVision이 실행됨

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

```
34  /* Private define ------------------------------------------------------------*/
35  /* USER CODE BEGIN PD */
36  #define TxBufferSize  (countof(TxBuffer) - 1)
37  #define TxBufferSize_2  (countof(TxBuffer_2) - 1)
38  #define TxBufferSize_3  (countof(TxBuffer_3) - 1)
39  #define TxBufferSize_4  (countof(TxBuffer_4) - 1)
40  #define TxBufferSize_5  (countof(TxBuffer_5) - 1)
41  #define RxBufferSize  0xFF
42  #define countof(a)  (sizeof(a) / sizeof(*(a)))
43  /* USER CODE END PD */
44
45  /* Private macro -------------------------------------------------------------*/
46  /* USER CODE BEGIN PM */
47
48  /* USER CODE END PM */
49
50  /* Private variables ---------------------------------------------------------*/
51  UART_HandleTypeDef huart2;
52
53  /* USER CODE BEGIN PV */
54  uint8_t TxBuffer[] = "\n\rUART Example 2 (Transmission  Success !!)\n\r\n\r";
55  uint8_t TxBuffer_2[] = "\n\r Button1 is pressed !! \n\r";
56  uint8_t TxBuffer_3[] = "\n\r Button2 is pressed !! \n\r";
57  uint8_t TxBuffer_4[] = "\n\r Button3 is pressed !! \n\r";
58  uint8_t TxBuffer_5[] = "\n\r Button4 is pressed !! \n\r";
59  uint8_t RxBuffer[RxBufferSize];
60  /* USER CODE END PV */
```

→ 변수와 정의 선언. 이 부분은 기존의 예제 2와 동일

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

```
67  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
68  {
69      switch(GPIO_Pin)
70      {
71            case GPIO_PIN_13:
72               HAL_UART_Transmit(&huart2, (uint8_t*)TxBuffer_2, TxBufferSize_2 , 0xFFFF);
73            break;
74            case GPIO_PIN_8:
75               HAL_UART_Transmit(&huart2, (uint8_t*)TxBuffer_2, TxBufferSize_2 , 0xFFFF);
76            break;
77            case GPIO_PIN_4:
78               HAL_UART_Transmit(&huart2, (uint8_t*)TxBuffer_3, TxBufferSize_3 , 0xFFFF);
79            break;
80            case GPIO_PIN_5:
81               HAL_UART_Transmit(&huart2, (uint8_t*)TxBuffer_4, TxBufferSize_4 , 0xFFFF);
82            break;
83            case GPIO_PIN_10:
84               HAL_UART_Transmit(&huart2, (uint8_t*)TxBuffer_5, TxBufferSize_5 , 0xFFFF);
85            break;
86      }
87  }
```

→ 외부 인터럽트 콜백함수. 이 부분은 기존의 예제 2와 동일

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 2 : 스위치를 누르면 PC로 메시지 보내기

```
122   MX_GPIO_Init();
123   MX_USART2_UART_Init();
124   /* USER CODE BEGIN 2 */
125   HAL_UART_Transmit(&huart2, (uint8_t*)TxBuffer, TxBufferSize, 0xFFFF);
126   /* USER CODE END 2 */
127
128   /* Infinite loop */
129   /* USER CODE BEGIN WHILE */
130   while (1)
131   {
132     /* USER CODE END WHILE */
133
134     /* USER CODE BEGIN 3 */
135   }
136   /* USER CODE END 3 */
```

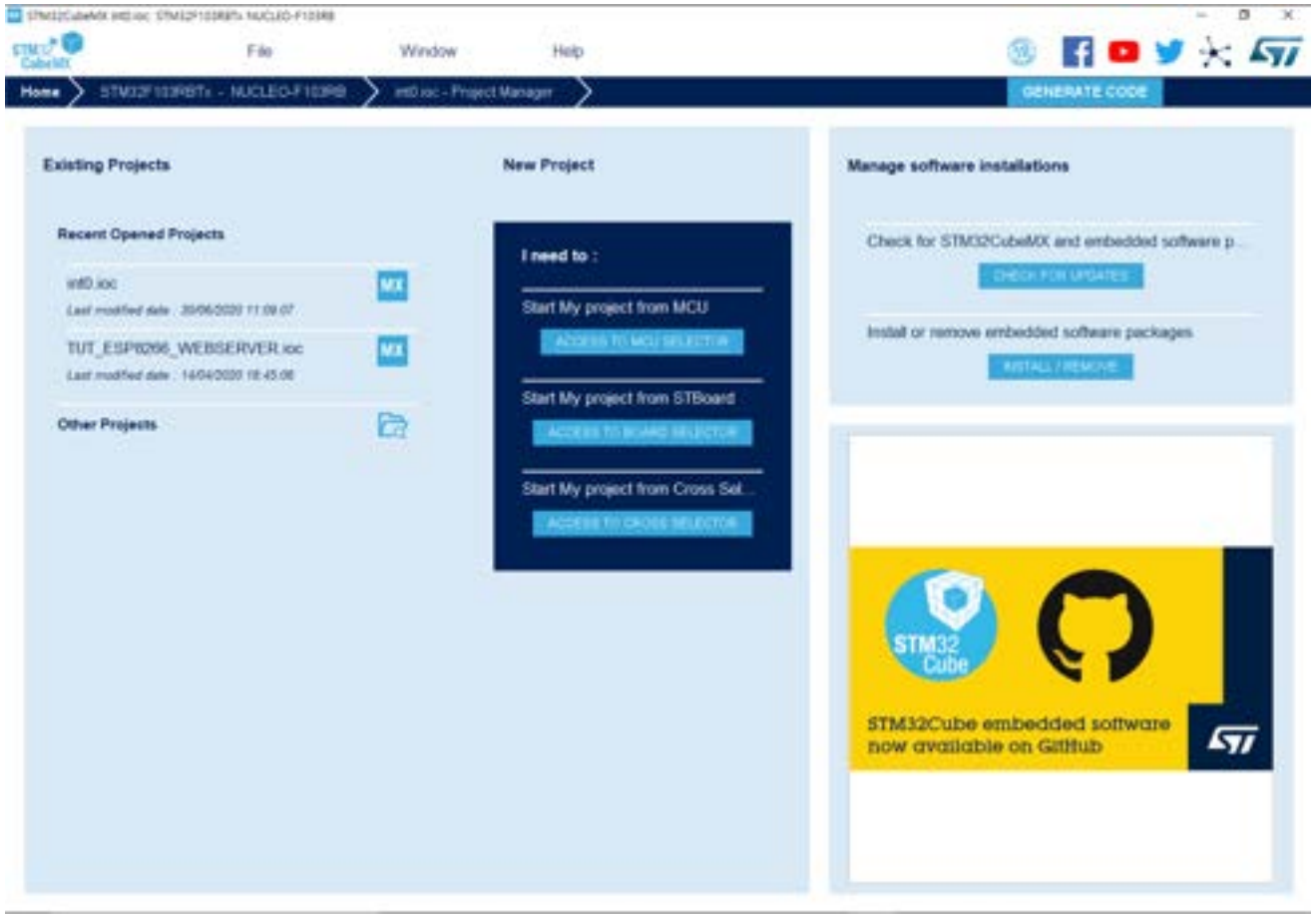
→ 125줄은 보드를 리셋하면 처음 나오는 문구표시
→ 그 외에는 스위치를 누르면 외부 인터럽트가 걸림

11.5 UART CubeMX과제

CubeMX로 예제 11.3 구현하기

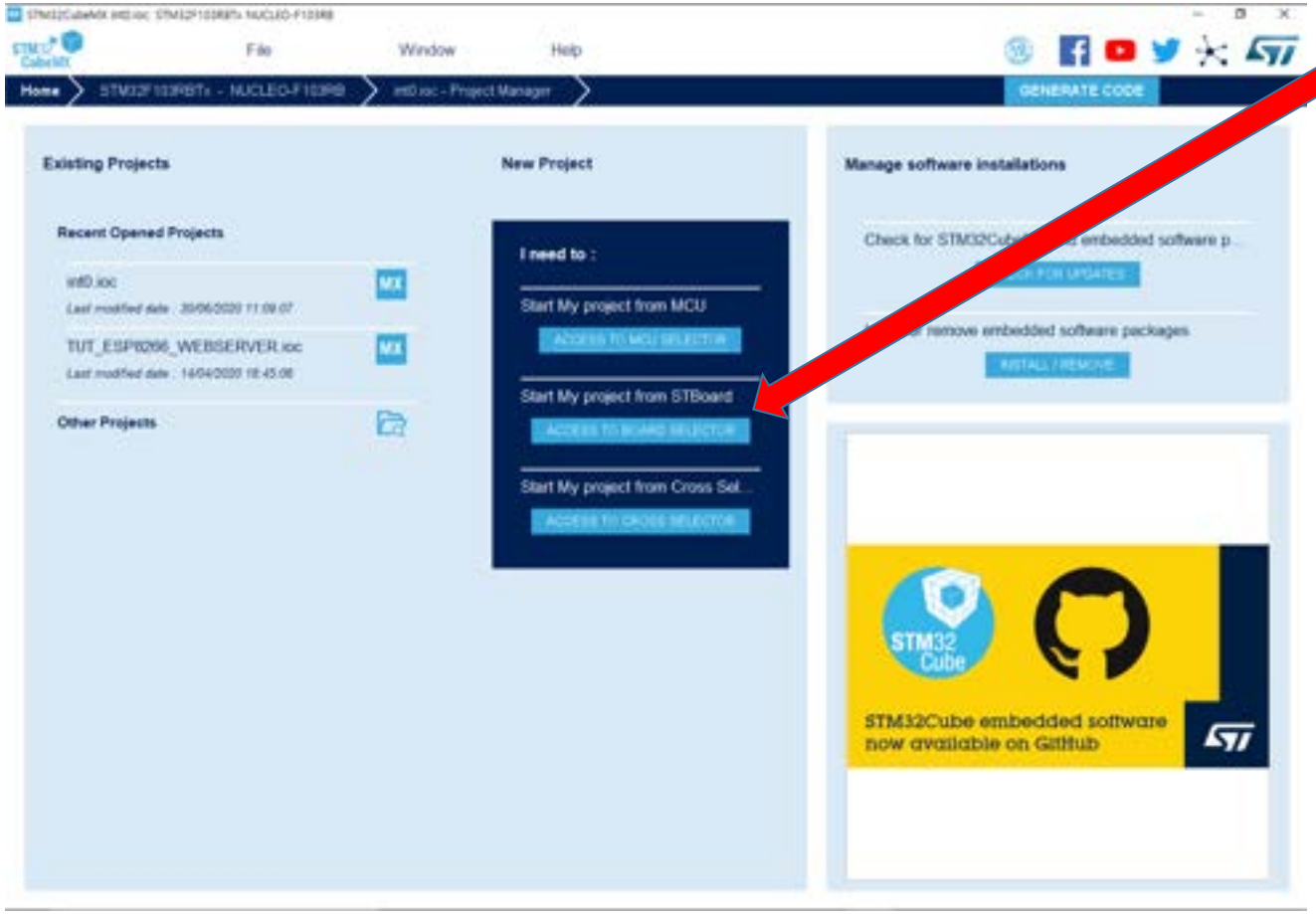
CubeMX 실행

초기화면



11.5 UART CubeMX과제

CubeMX로 예제 11.3 구현하기

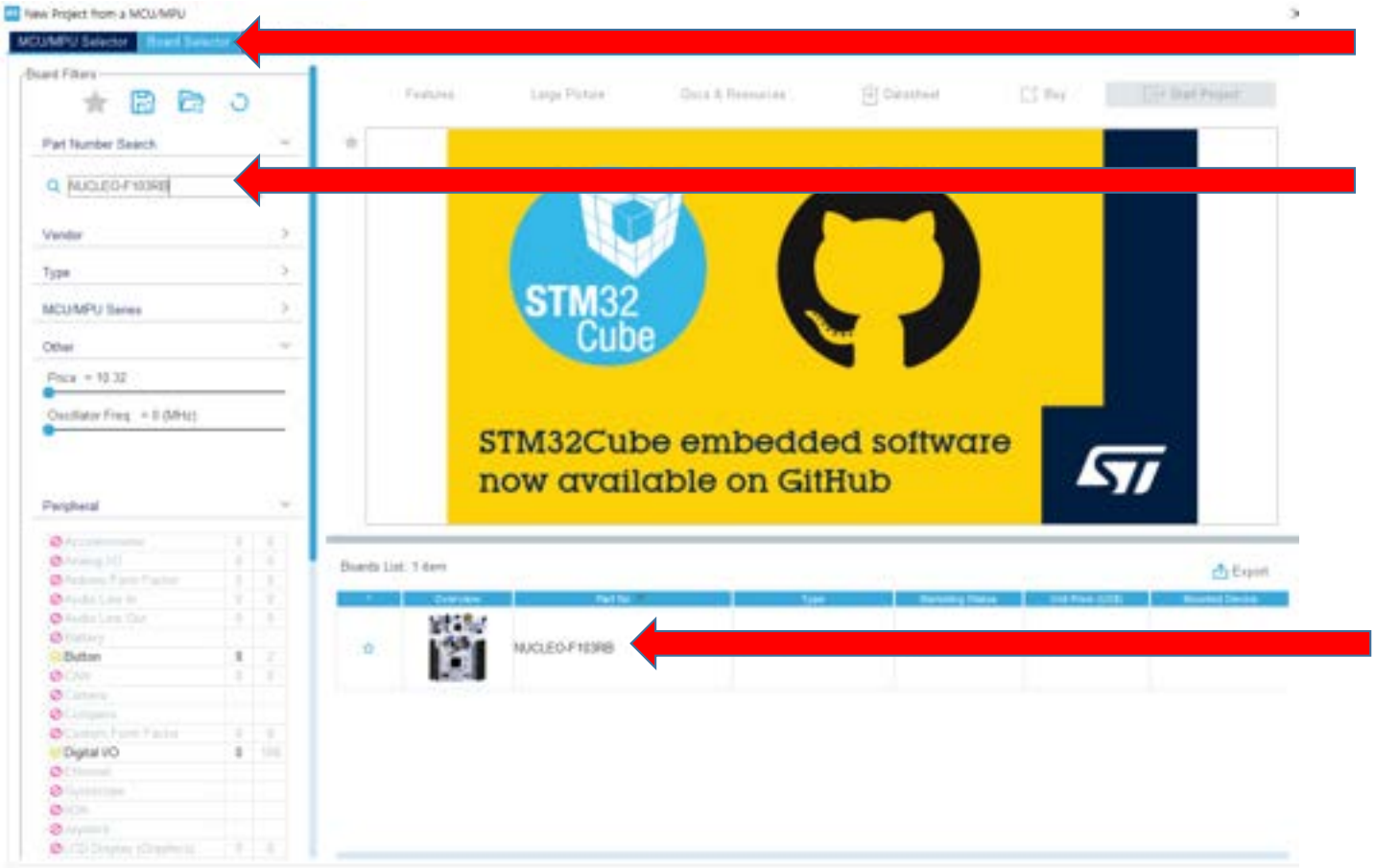


보드 선택

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 예제 11.3 구현하기



보드 선택

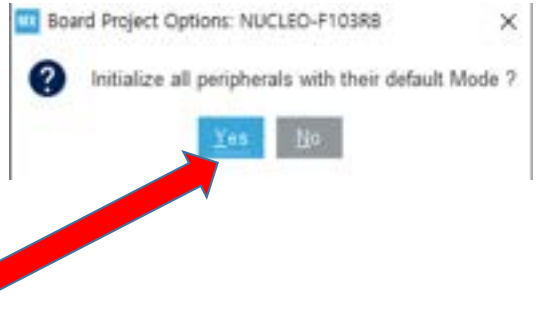
NUCLEO-F103RB 선택

NUCLEO-F103RB 더블클릭

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

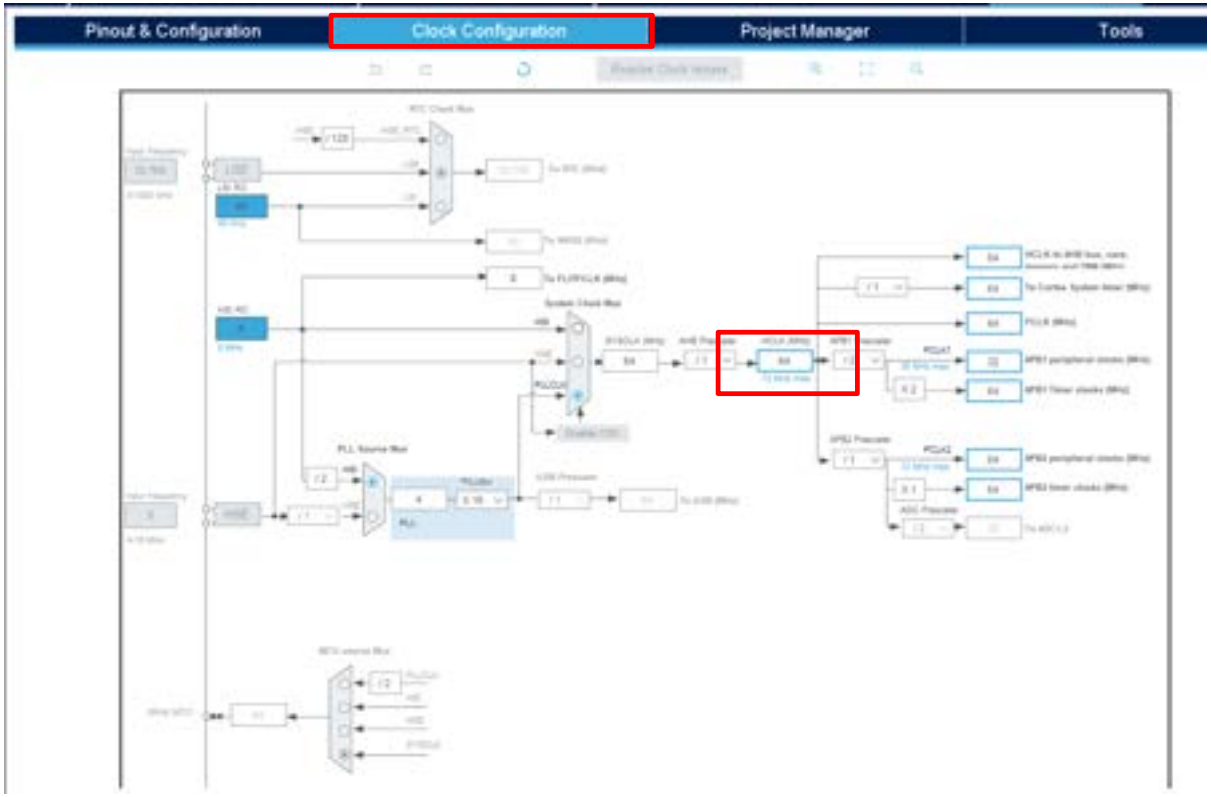
CubeMX로 예제 11.3 구현하기



Yes 선택하면 오른쪽과 같이 칩이 보임

11.5 UART CubeMX과제

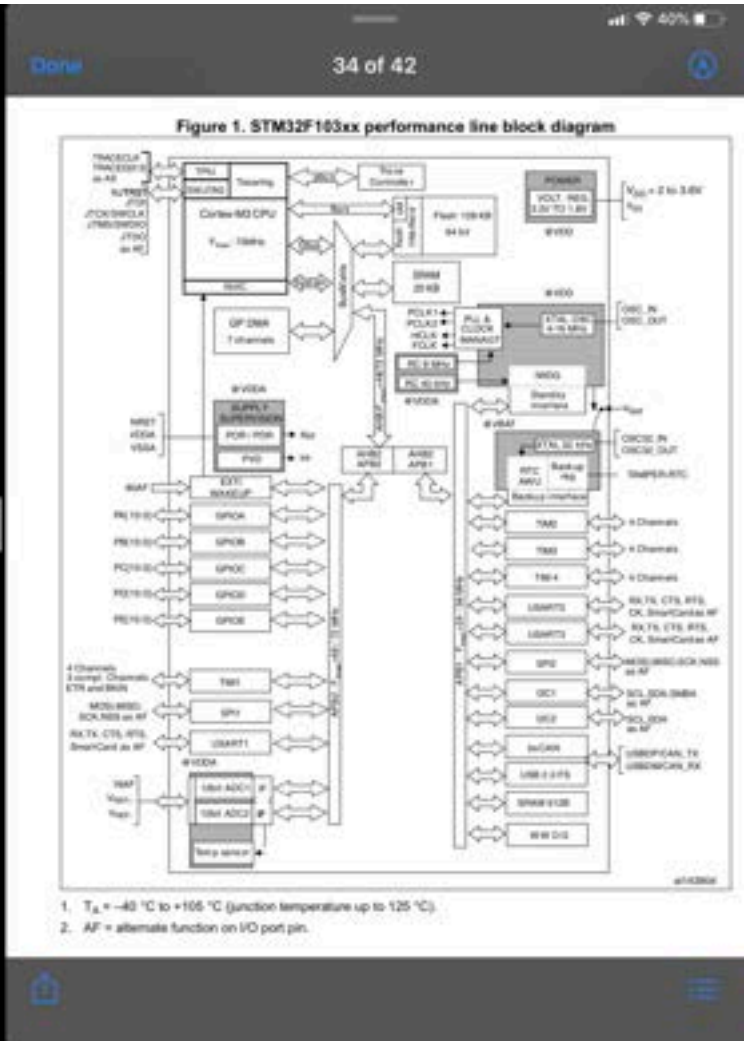
CubeMX로 UART를 이용하여 PC와 통신하기(폴링 방식)



Clock configuration 탭에서 Main Clk를 설정 및 확인 → 64MHz

11.5 UART CubeMX과제

```
 *          PLLMUL                  = 16
 *          Flash Latency(WS)       = 2
 */
// ------------------------------------------------------------------------- //

void  SystemClock_Config(void)
{
          RCC_ClkInitTypeDef clkinitstruct = {0};
          RCC_OscInitTypeDef oscinitstruct = {0};

          /* Configure PLL ------------------------------------------------------*/
          /* PLL configuration: PLLCLK = (HSI / 2) * PLLMUL = (8 / 2) * 16 = 64 MHz */
          /* PREDIV1 configuration: PREDIV1CLK = PLLCLK / HSEPredivValue = 64 / 1 = 64
MHz */
          /* Enable HSI and activate PLL with HSI_DIV2 as source */
          oscinitstruct.OscillatorType  = RCC_OSCILLATORTYPE_HSI;
          oscinitstruct.HSEState        = RCC_HSE_OFF;
          oscinitstruct.LSEState        = RCC_LSE_OFF;
          oscinitstruct.HSIState        = RCC_HSI_ON;
          oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
          oscinitstruct.HSEPredivValue    = RCC_HSE_PREDIV_DIV1;
          oscinitstruct.PLL.PLLState    = RCC_PLL_ON;
          oscinitstruct.PLL.PLLSource   = RCC_PLLSOURCE_HSI_DIV2;
          oscinitstruct.PLL.PLLMUL      = RCC_PLL_MUL16;

          if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)  {
                    /* Initialization Error */
                    while(1);
          }

          /* Select PLL as system clock source and configure the HCLK, PCLK1 and
PCLK2
              clocks dividers */
          clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
          clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
          clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
          clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
          clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;

          if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)  {
                    /* Initialization Error */
                    while(1);
          }
}

// ------------------------------------------------------------------------- //
//  --  <18>  Clock 설정시 에러가 발생하면 처리해주는 함수
/**
```



KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

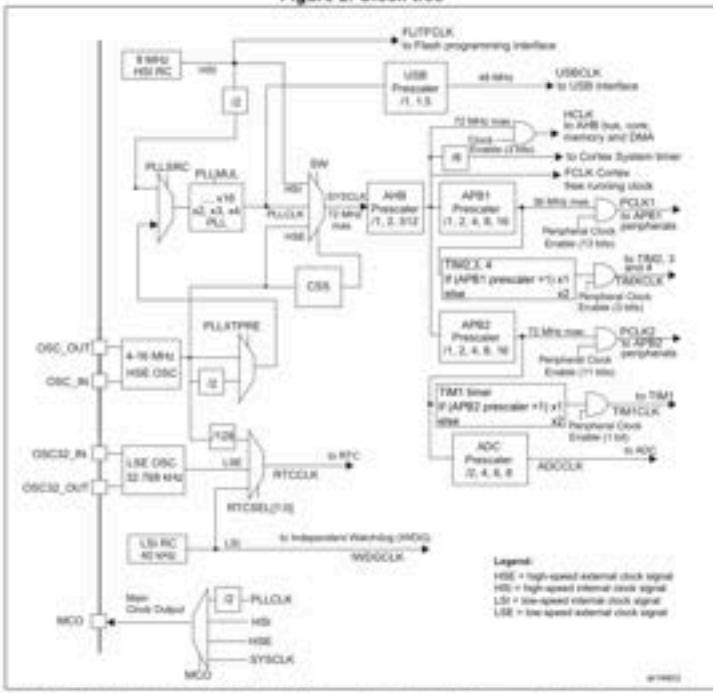
8:41 PM Mon Oct 28

Done 2 of 4

```
void SystemClock_Config(void)
{
  RCC_ClkInitTypeDef clkinitstruct = {0};
  RCC_OscInitTypeDef oscinitstruct = {0};

  /* Configure PLL ------------------------------------------------------*/
  /* PLL configuration: PLLCLK = (HSI / 2) * PLLMUL = (8 / 2) * 16 = 64 MHz */
  /* PREDIV1 configuration: PREDIV1CLK = PLLCLK / HSEPredivValue = 64 / 1 = 64
MHz */
  /* Enable HSI and activate PLL with HSi_DIV2 as source */
  oscinitstruct.OscillatorType  = RCC_OSCILLATORTYPE_HSI;
  oscinitstruct.HSEState        = RCC_HSE_OFF;
  oscinitstruct.LSEState        = RCC_LSE_OFF;
  oscinitstruct.HSIState        = RCC_HSI_ON;
  oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  oscinitstruct.HSEPredivValue    = RCC_HSE_PREDIV_DIV1;
  oscinitstruct.PLL.PLLState    = RCC_PLL_ON;
  oscinitstruct.PLL.PLLSource   = RCC_PLLSOURCE_HSI_DIV2;
  oscinitstruct.PLL.PLLMUL      = RCC_PLL_MUL16;

  if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)  {
      /* Initialization Error */
      while(1);
  }

  /* Select PLL as system clock source and configure the HCLK, PCLK1 and
PCLK2
      clocks dividers */
  clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
  clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
  clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)  {
      /* Initialization Error */
      while(1);
  }
}
```

Done 34 of 42

Figure 2. Clock tree

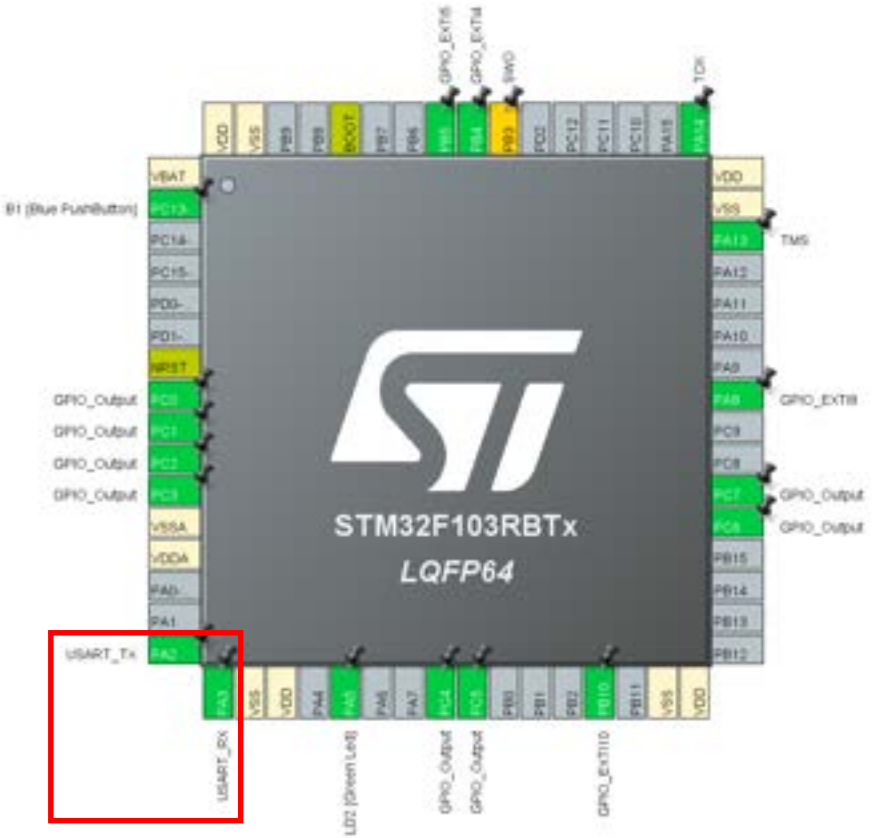


1. When the HSI is used as a PLL clock input, the maximum system clock frequency that can be achieved is 64 MHz.
2. For the USB function to be available, both HSE and PLL must be enabled, with USBCLK running at 48 MHz.
3. To have an ADC conversion time of 1 µs, APB2 must be at 14 MHz, 28 MHz or 56 MHz.

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 3 : UART를 이용하여 PC와 통신하기(폴링 방식)



Pinout & Configuration 탭에서

~~GPIO 출력핀 설정 : PC0~7에 LED 연결됨~~

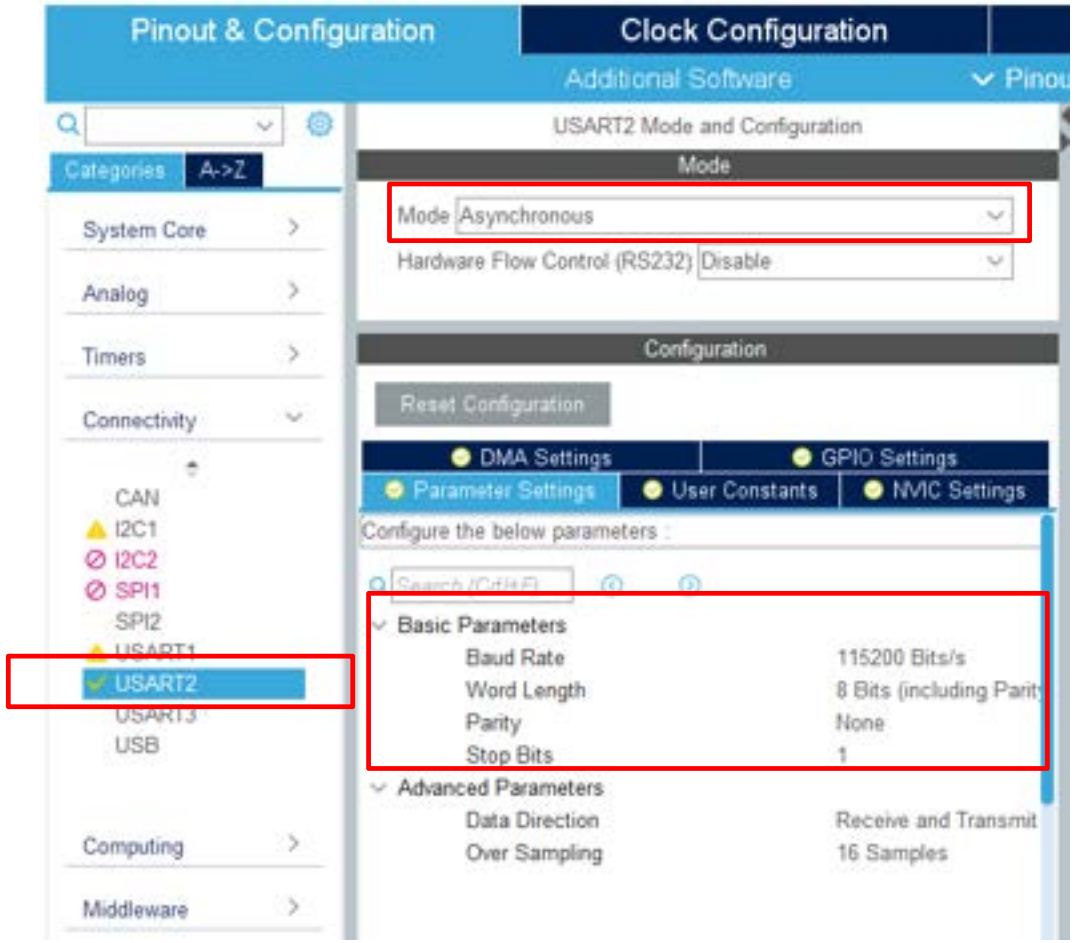
~~PA8, PB4, PB5, PB10을 GPIO_EXT로 설정함~~

UART2(PA2, PA3)를 설정함

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 3 : UART를 이용하여 PC와 통신하기(폴링 방식)

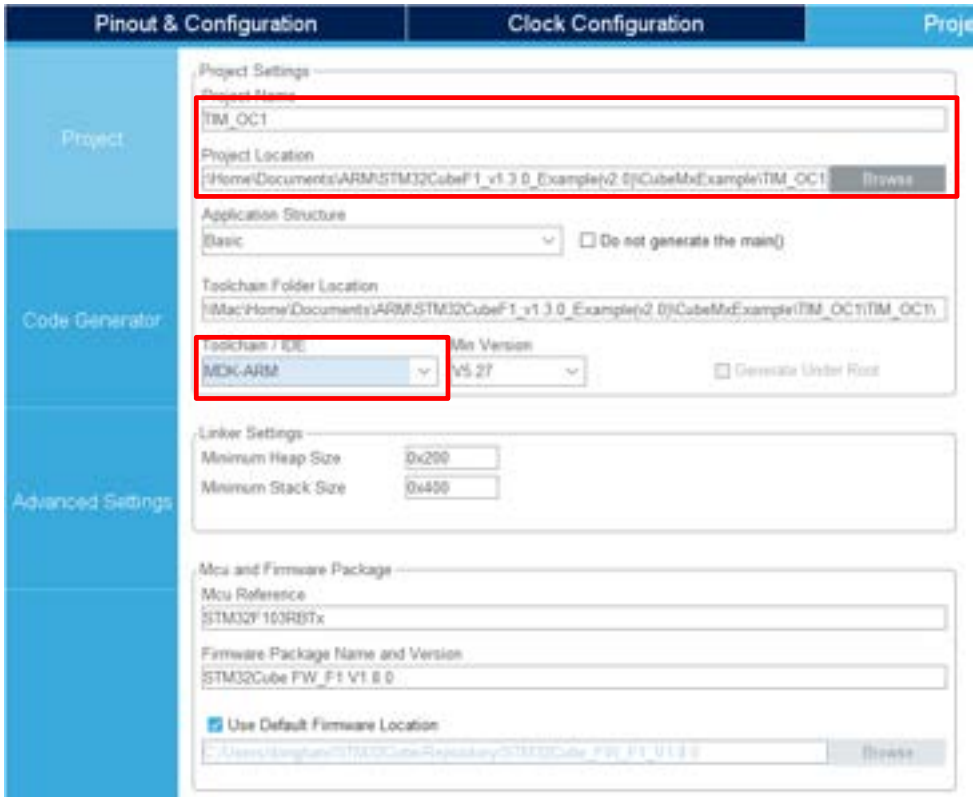


왼쪽 윈도우의 Connectivity중에 USART2를 선택하고, 왼쪽 윈도우의 Mode에서 Asynchronous 선택하고, Configuration에서 기본 설정된 값(Baud Rate, Word Length, Parity, Stop Bits)를 기억함

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 3 : UART를 이용하여 PC와 통신하기(폴링 방식)



Project Manager 탭을 누르면, 왼쪽의 그림처럼 project 생성 단계가 되며, project name과 location을 정해줌.

Toolchain / IDE는 꼭 MDK-ARM을 선택

마지막으로 GENERATE CODE 탭을 누르면, 코드가 생성되면서 MDK uVision이 실행됨

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 3 : UART를 이용하여 PC와 통신하기(폴링 방식)

```
34  /* Private define ------------------------------------------------------------*/
35  /* USER CODE BEGIN PD */
36  #define TxBufferSize  (countof(TxBuffer) - 1)
37  #define RxBufferSize  0xFF
38  #define countof(a)  (sizeof(a) / sizeof(*(a)))
39  /* USER CODE END PD */
40
41  /* Private macro -------------------------------------------------------------*/
42  /* USER CODE BEGIN PM */
43
44  /* USER CODE END PM */
45
46  /* Private variables ---------------------------------------------------------*/
47  UART_HandleTypeDef huart2;
48
49  /* USER CODE BEGIN PV */
50  uint8_t TxBuffer[] = "\n\rUART Example 3 (Transmission  Success !!)\n\r\n\r";
51  uint8_t RxBuffer[RxBufferSize];
52  /* USER CODE END PV */
```

→ 변수와 정의 선언. 이 부분은 기존의 예제 3과 동일

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 3 : UART를 이용하여 PC와 통신하기(폴링 방식)

```
96    /* USER CODE BEGIN 2 */
97    HAL_UART_Transmit(&huart2, (uint8_t*)TxBuffer, TxBufferSize, 0xFFFF);
98
99    /* USER CODE END 2 */
100
101   /* Infinite loop */
102   /* USER CODE BEGIN WHILE */
103   while (1)
104   {
105     /* USER CODE END WHILE */
106
107     /* USER CODE BEGIN 3 */
108     if(HAL_UART_Receive(&huart2, (uint8_t*)RxBuffer, 1, 5000) == HAL_OK)
109     {
110       HAL_UART_Transmit(&huart2, (uint8_t*)RxBuffer, 1, 5000);
111     }
112   }
113   /* USER CODE END 3 */
```

→ 이 부분은 기존의 예제 3과 동일

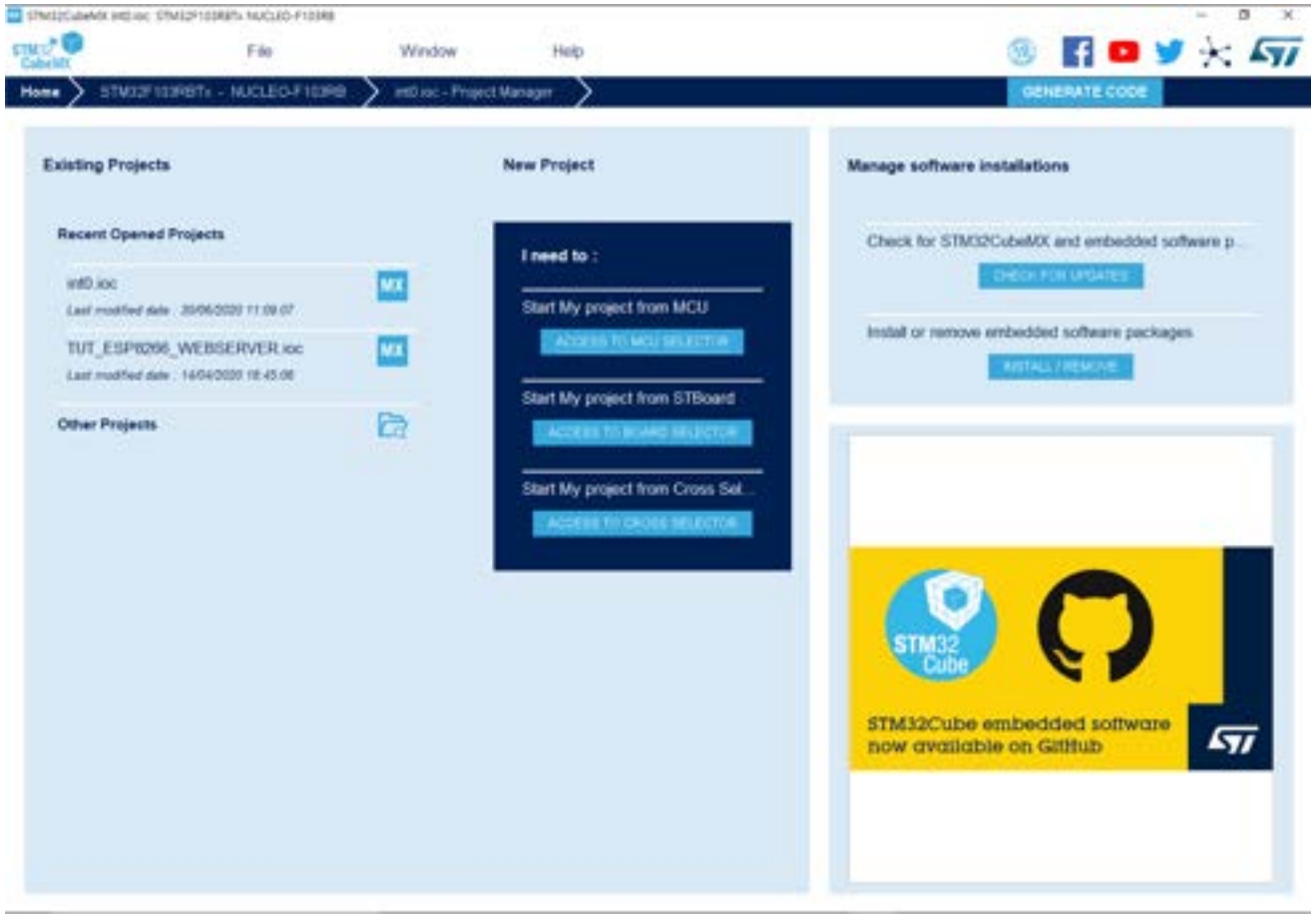
KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 예제 11.4 구현하기

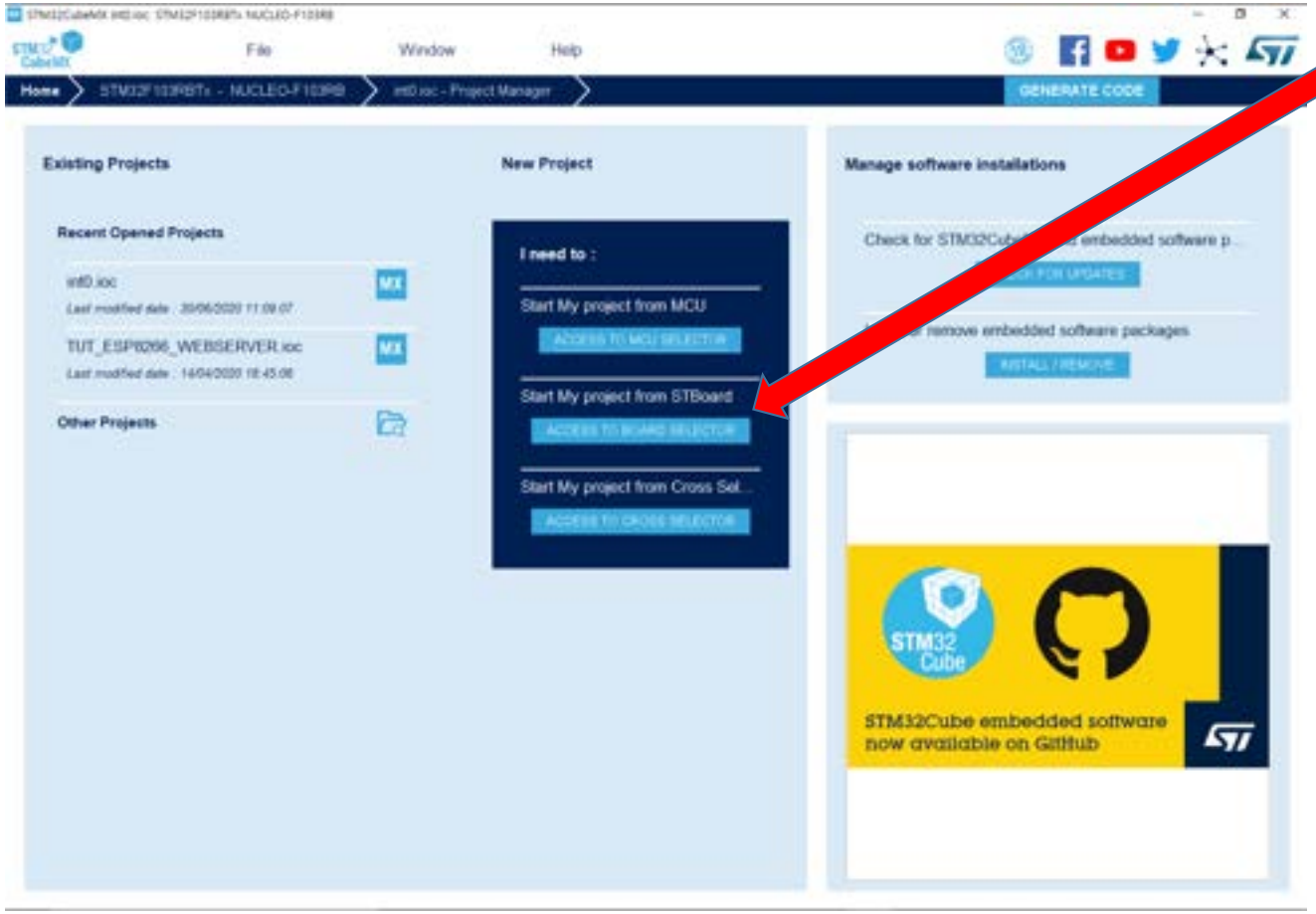
CubeMX 실행

초기화면



11.5 UART CubeMX과제

CubeMX로 예제 11.4 구현하기

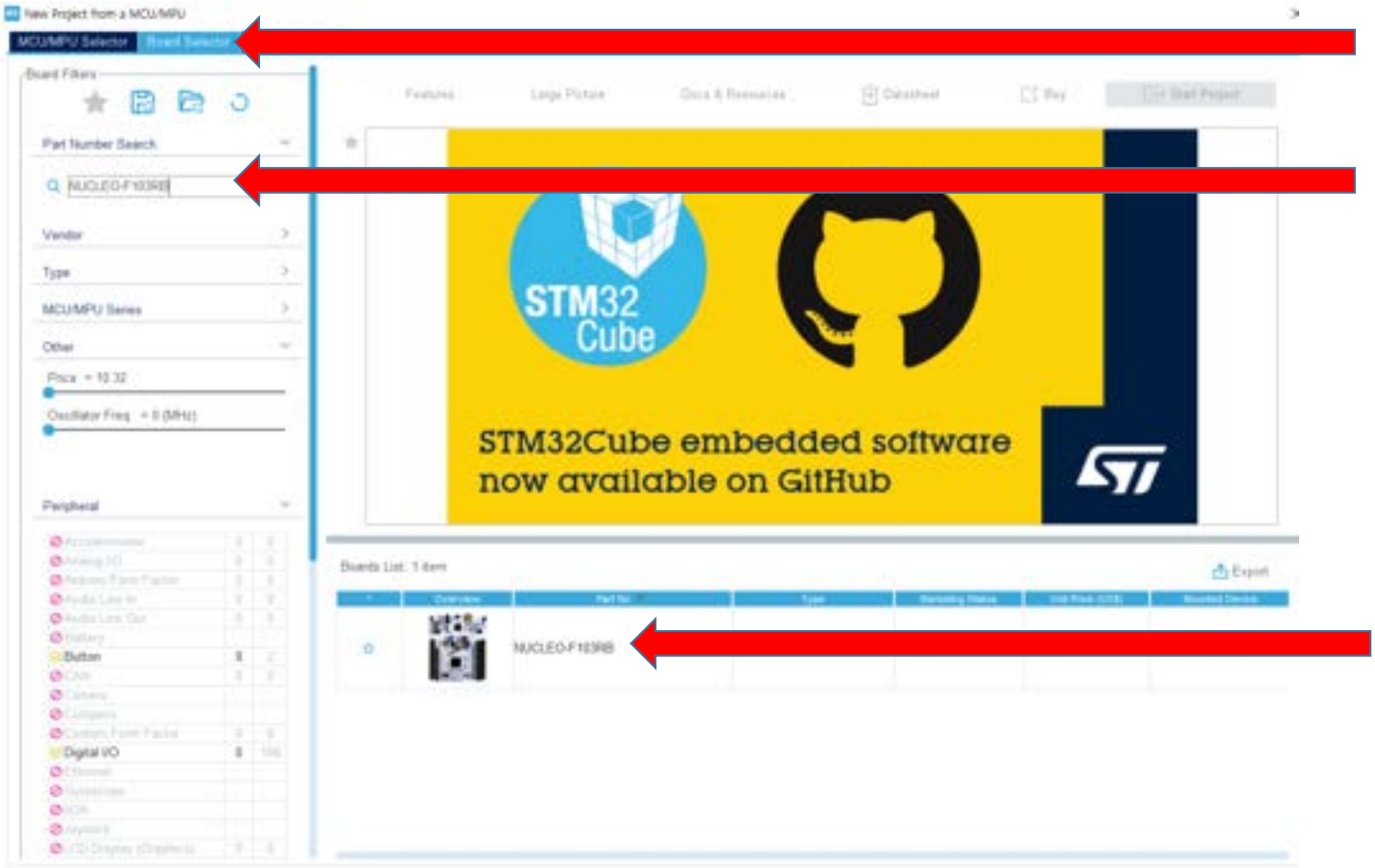


보드 선택

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 예제 11.4 구현하기



보드 선택

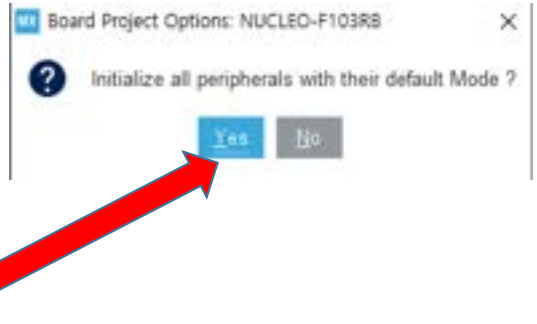
NUCLEO-F103RB 선택

NUCLEO-F103RB 더블클릭

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

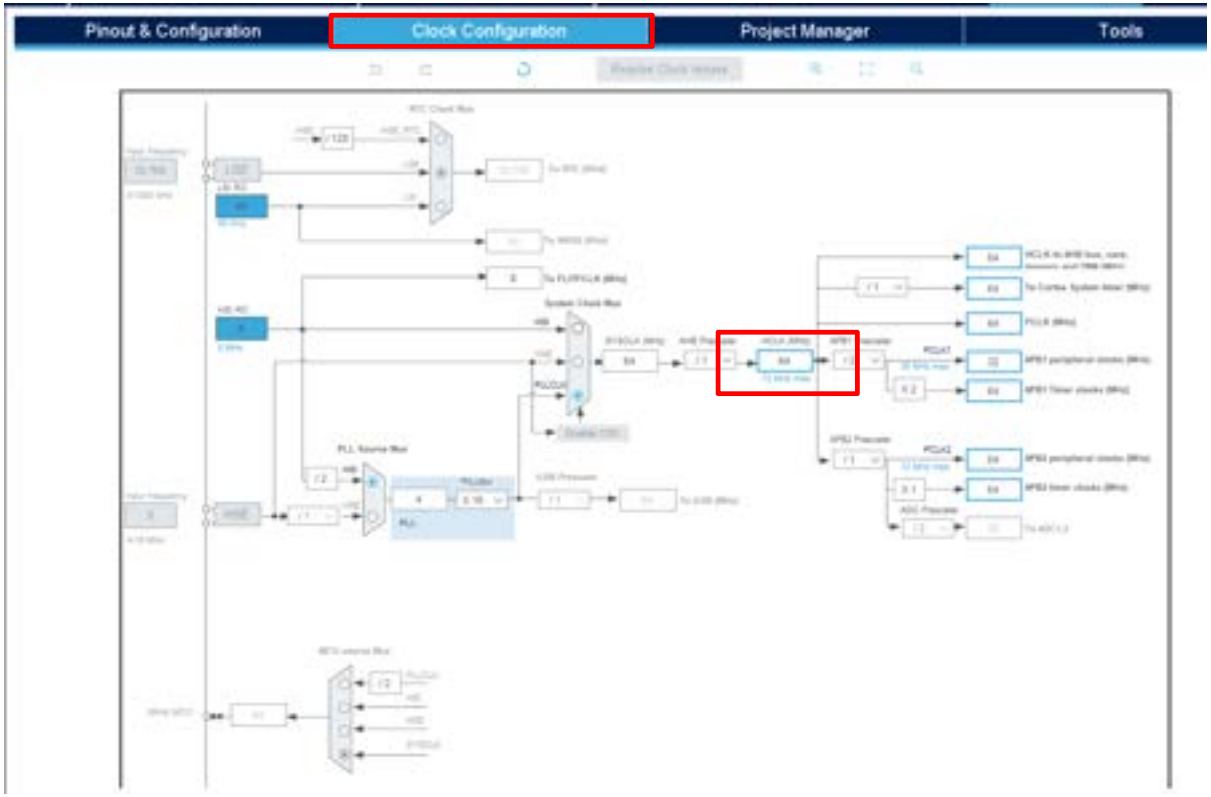
CubeMX로 예제 11.4 구현하기



Yes 선택하면 오른쪽과 같이 칩이 보임

11.5 UART CubeMX과제

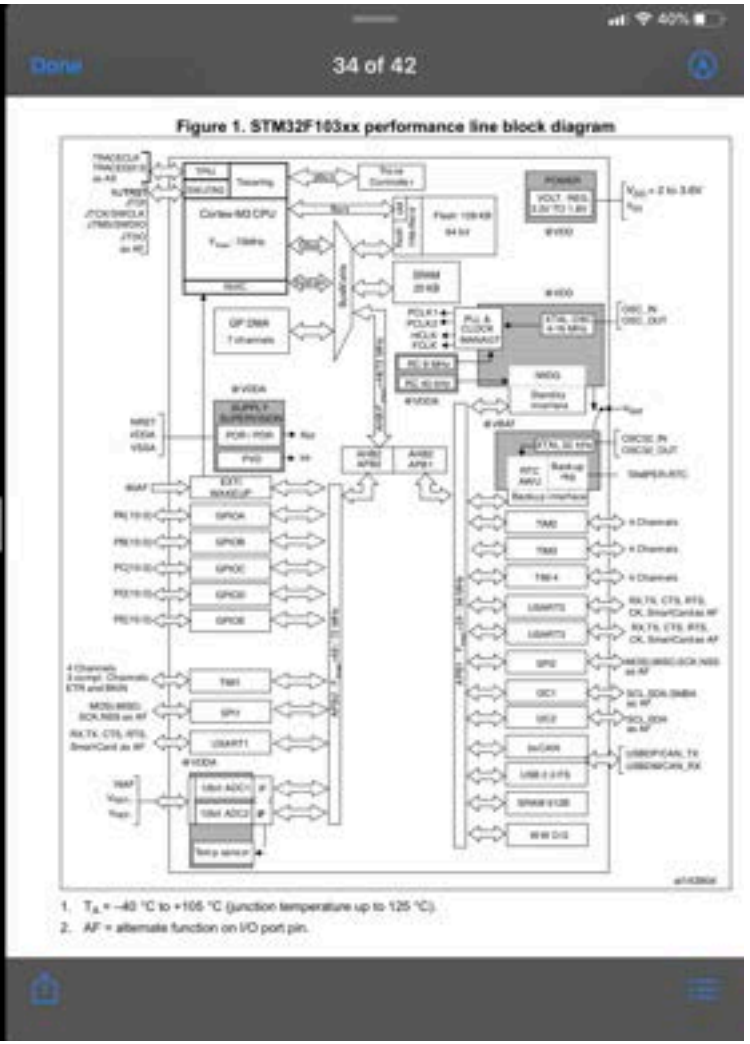
CubeMX로 UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)



Clock configuration 탭에서 Main Clk를 설정 및 확인 → 64MHz

11.5 UART CubeMX과제

```
 *          PLLMUL                  = 16
 *          Flash Latency(WS)       = 2
 */
// ------------------------------------------------------------------------- //

void  SystemClock_Config(void)
{
          RCC_ClkInitTypeDef clkinitstruct = {0};
          RCC_OscInitTypeDef oscinitstruct = {0};

          /* Configure PLL ------------------------------------------------------*/
          /* PLL configuration: PLLCLK = (HSI / 2) * PLLMUL = (8 / 2) * 16 = 64 MHz */
          /* PREDIV1 configuration: PREDIV1CLK = PLLCLK / HSEPredivValue = 64 / 1 = 64
MHz */
          /* Enable HSI and activate PLL with HSI_DIV2 as source */
          oscinitstruct.OscillatorType  = RCC_OSCILLATORTYPE_HSI;
          oscinitstruct.HSEState        = RCC_HSE_OFF;
          oscinitstruct.LSEState        = RCC_LSE_OFF;
          oscinitstruct.HSIState        = RCC_HSI_ON;
          oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
          oscinitstruct.HSEPredivValue    = RCC_HSE_PREDIV_DIV1;
          oscinitstruct.PLL.PLLState    = RCC_PLL_ON;
          oscinitstruct.PLL.PLLSource   = RCC_PLLSOURCE_HSI_DIV2;
          oscinitstruct.PLL.PLLMUL      = RCC_PLL_MUL16;

          if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)  {
                    /* Initialization Error */
                    while(1);
          }

          /* Select PLL as system clock source and configure the HCLK, PCLK1 and
PCLK2
              clocks dividers */
          clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
          clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
          clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
          clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
          clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;

          if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)  {
                    /* Initialization Error */
                    while(1);
          }
}

// ------------------------------------------------------------------------- //
//  --  <18>  Clock 설정시 에러가 발생하면 처리해주는 함수
/**
```



KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

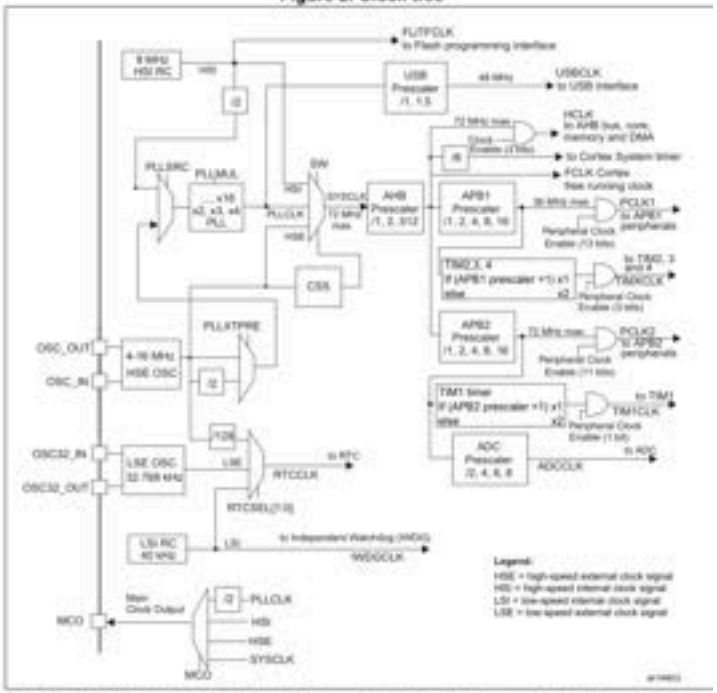
8:41 PM Mon Oct 28

Done 2 of 4

```
void SystemClock_Config(void)
{
  RCC_ClkInitTypeDef clkinitstruct = {0};
  RCC_OscInitTypeDef oscinitstruct = {0};

  /* Configure PLL ------------------------------------------------------*/
  /* PLL configuration: PLLCLK = (HSI / 2) * PLLMUL = (8 / 2) * 16 = 64 MHz */
  /* PREDIV1 configuration: PREDIV1CLK = PLLCLK / HSEPredivValue = 64 / 1 = 64
MHz */
  /* Enable HSI and activate PLL with HSi_DIV2 as source */
  oscinitstruct.OscillatorType  = RCC_OSCILLATORTYPE_HSI;
  oscinitstruct.HSEState        = RCC_HSE_OFF;
  oscinitstruct.LSEState        = RCC_LSE_OFF;
  oscinitstruct.HSIState        = RCC_HSI_ON;
  oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  oscinitstruct.HSEPredivValue    = RCC_HSE_PREDIV_DIV1;
  oscinitstruct.PLL.PLLState    = RCC_PLL_ON;
  oscinitstruct.PLL.PLLSource   = RCC_PLLSOURCE_HSI_DIV2;
  oscinitstruct.PLL.PLLMUL      = RCC_PLL_MUL16;

  if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)  {
      /* Initialization Error */
      while(1);
  }

  /* Select PLL as system clock source and configure the HCLK, PCLK1 and
PCLK2
      clocks dividers */
  clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK
| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
  clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
  clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)  {
      /* Initialization Error */
      while(1);
  }
}
```

Done 34 of 42

Figure 2. Clock tree

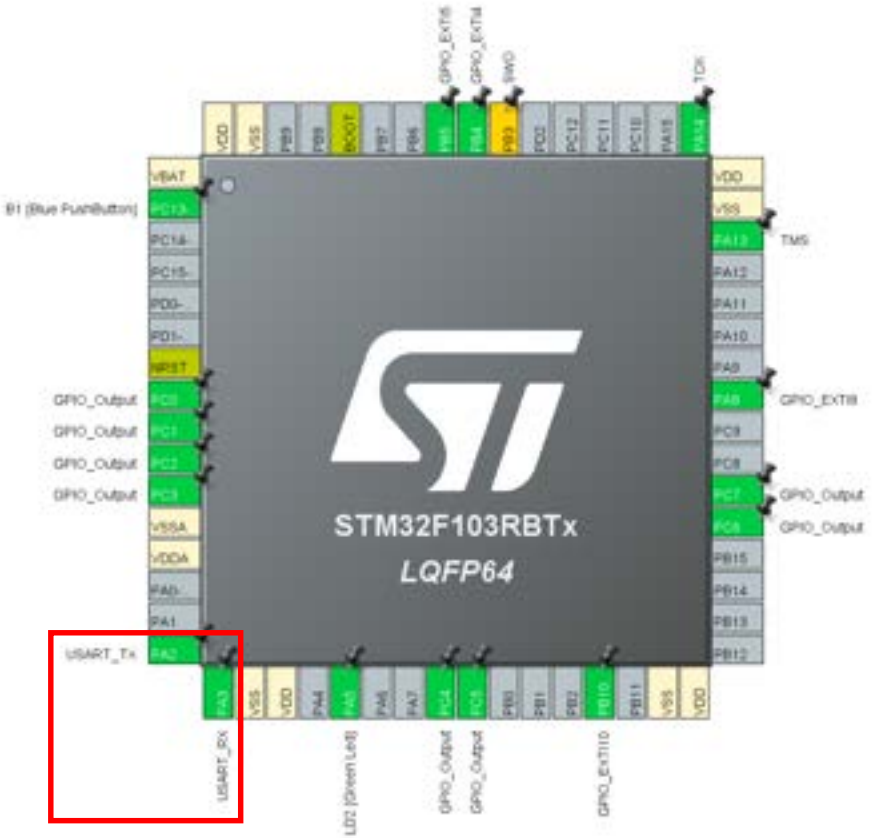


1. When the HSI is used as a PLL clock input, the maximum system clock frequency that can be achieved is 64 MHz.
2. For the USB function to be available, both HSE and PLL must be enabled, with USBCLK running at 48 MHz.
3. To have an ADC conversion time of 1 µs, APB2 must be at 14 MHz, 28 MHz or 56 MHz.

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)



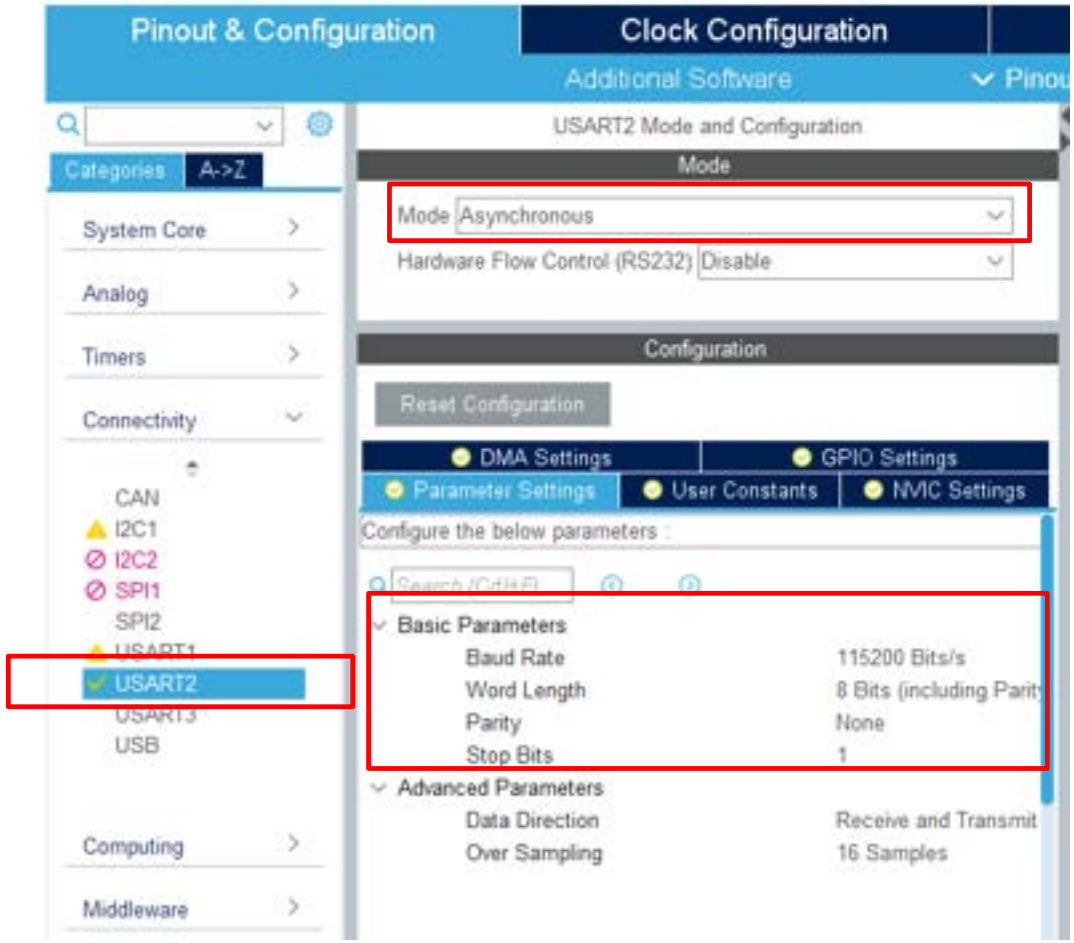
Pinout & Configuration 탭에서

UART2(PA2, PA3)를 설정함

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)



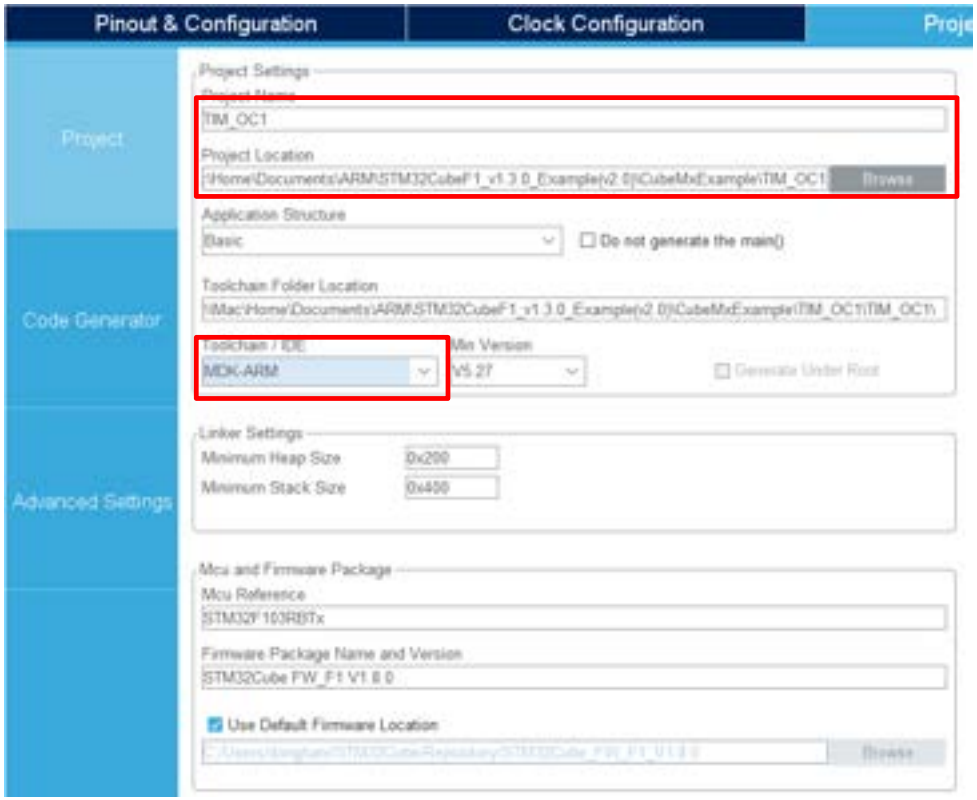
왼쪽 윈도우의 Connectivity중에 USART2를 선택하고, 왼쪽 윈도우의 Mode에서 Asynchronous 선택하고, Configuration에서 기본 설정된 값(Baud Rate, Word Length, Parity, Stop Bits)를 기억함

UART와 관련된 인터럽트를 사용하기 위해서 Configuration의 NVIC Settings 탭을 열어서 USART2 global interrupt를 Enabled 함

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)



Project Manager 탭을 누르면, 왼쪽의 그림처럼 project 생성 단계가 되며, project name과 location을 정해줌.

Toolchain / IDE는 꼭 MDK-ARM을 선택

마지막으로 GENERATE CODE 탭을 누르면, 코드가 생성되면서 MDK uVision이 실행됨

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)

```
34  /* Private define ------------------------------------------------------------*/
35  /* USER CODE BEGIN PD */
36  #define TxBufferSize    (countof(TxBuffer) - 1)
37  #define TxBufferSize_3  (countof(TxBuffer_3) - 1)
38  #define RxBufferSize    0xFF
39  #define countof(a)    (sizeof(a) / sizeof(*(a)))
40
41  /* USER CODE END PD */
42
43  /* Private macro -------------------------------------------------------------*/
44  /* USER CODE BEGIN PM */
45
46  /* USER CODE END PM */
47
48  /* Private variables ---------------------------------------------------------*/
49  UART_HandleTypeDef huart2;
50
51  /* USER CODE BEGIN PV */
52  uint8_t TxBuffer[] = "\n\rUART Example 4 (Transmission  Success !!)\n\r\n\r";
53  uint8_t TxBuffer_3[] = "\n\r A Message from HAL_UART_TxCpltCallback() !!\n\r\n\r";
54  uint8_t RxBuffer[RxBufferSize];
55  /* USER CODE END PV */
```

→ 변수와 정의 선언. 이 부분은 기존의 예제 4와 동일

11.5 UART CubeMX과제

CubeMX로 UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)

```
61  /* USER CODE BEGIN PFP */
62  void HAL_UART_TxCpltCallback(UART_HandleTypeDef *UartHandler)
63  {
64    HAL_UART_Transmit(UartHandler, (uint8_t*)TxBuffer_3, TxBufferSize_3 , 0xFFFF);
65  }
66
67
68  void HAL_UART_RxCpltCallback(UART_HandleTypeDef *UartHandler)
69  {
70    HAL_UART_Transmit_IT(UartHandler, (uint8_t*)RxBuffer, 1);
71  }
72
73  /* USER CODE END PFP */
```

→ UART2를 통해서 PC에서 데이터를 받았을 때와 보낼 때 발생하는 interrupt callback 함수

KYUNG HEE UNIVERSITY

11.5 UART CubeMX과제

CubeMX로 UART 예제 4 : UART를 이용하여 PC와 통신하기(인터럽트 방식)

```
109  /* USER CODE BEGIN 2 */
110  HAL_UART_Transmit(&huart2, (uint8_t*)TxBuffer, TxBufferSize , 0xFFFF);
111
112  /* USER CODE END 2 */
113
114  /* Infinite loop */
115  /* USER CODE BEGIN WHILE */
116  while (1)
117  {
118    /* USER CODE END WHILE */
119
120    /* USER CODE BEGIN 3 */
121    HAL_UART_Receive_IT(&huart2, (uint8_t*)RxBuffer, 1);
122
123  }
124  /* USER CODE END 3 */
```

→ 기존 예제 4와 동일함