펌웨어 개발 및 회로 설계 기초

-3-

2019-02-14

BLUESINK

1. Encoder

1. Encoder

- 엔코더(Encoder)란 회전운동/직선운동을 전기적인 신호로 바꾸어 주는 장치이다.

- 엔코더는 그 구조에 따라 **광학식 / 기계식 / 자기식** 등 다양하게 존재하지만 여러 엔코더로부터 나오는 신호는 **일반적으로 동일**하다.

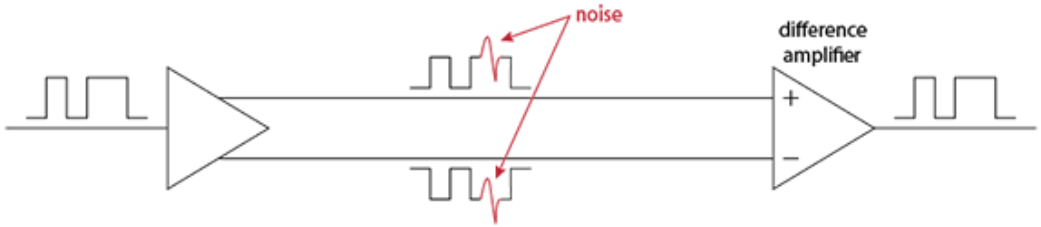
- A / B / (Z)
- A / ~A / B / ~B / (Z / ~Z)

Q. Z상은 무엇인가?

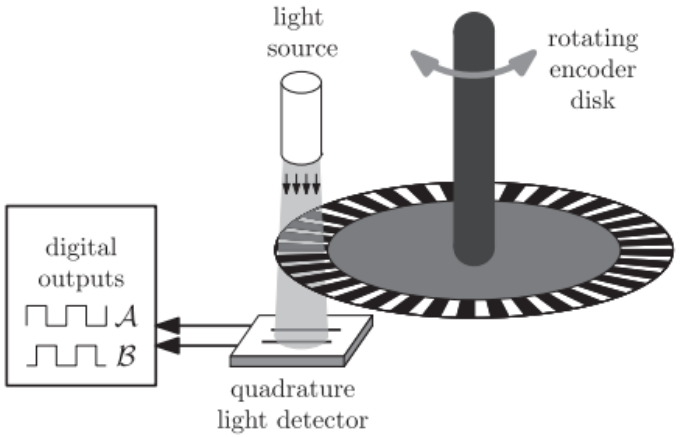
A. Index 신호, 한바퀴마다 한번 Pulse를 생성한다.

Q. 역상의 신호는 왜 필요한가?

A. Differential Signal



광학식 엔코더



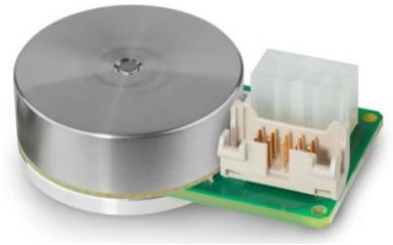
BLUESINK

1. Encoder

MILE Encoder for EC 45 flat – Product Information

Q. 엔코더의 분해능은 어떻게 되는가?

A. CPT(Counts Per Turn)



1.3 Angle Measurement

All values at T = 25°C, n = 1000 rpm, unless otherwise specified.
➔"Definitions" on page 6

Parameter	Conditions	Min.	Typ.	Max.	Unit
Number of channels	ChA, ChB		2		–
Pulse frequency (f_{pulse})	Max. output pulse frequency @ 2048 cpt without virtual backward states				
	@ 25°C	400	550	700	kHz
	@ –40°C	250	365	480	
Resolution (N)	Full period of A, B	256	512	2048	cpt
State length (L_{state})	N≤512 cpt	45	90	135 *1	°el
	N>1024 cpt	36	90	—	

⇒ 엔코더는 **CPT * 4** 만큼 한 바퀴를 나눈다. ***(Why?)***

⇒ 만약 엔코더가 512cpt를 가지고 있다면, 해당 엔코더는 한 바퀴를 2048등분한다.

1. Encoder

• Timer Encoder Mode

- 최신의 MCU들은 이러한 Encoder 신호를 **하드웨어적**으로 처리하는 기능을 가지고 있다.
- STM32의 경우, **타이머의 두 채널을 Combined** 하여 엔코더 신호를 처리할 수 있다.

<표 보는법>

현재 Encoder Mode TI1 and TI2 이라면,

- 만약 T1이 Rising Edge일때, TI2가 High면 카운터 1 감소
- 만약 T1이 Rising Edge일때, TI2가 Low면 카운터 1 증가
- 만약 T1이 Falling Edge일때, TI2가 High면 카운터 1 증가
- 만약 T1이 Falling Edge일때, TI2가 Low면 카운터 1 감소
- TI2에 대해서도 동일

Figure 199. Example of counter operation in encoder interface mode.

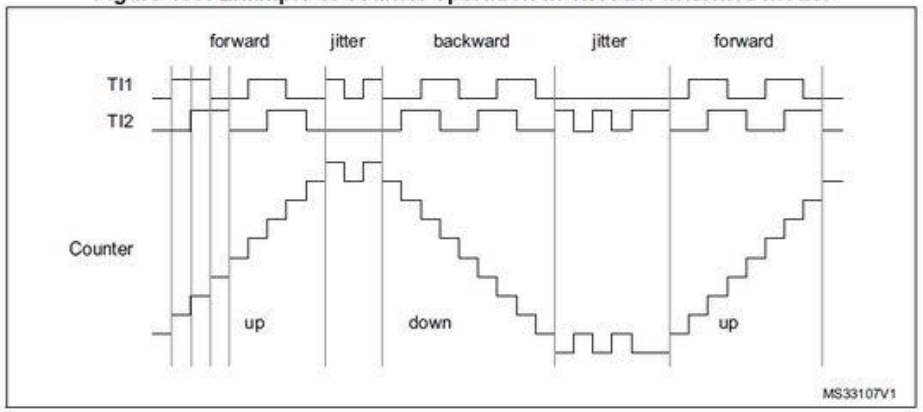
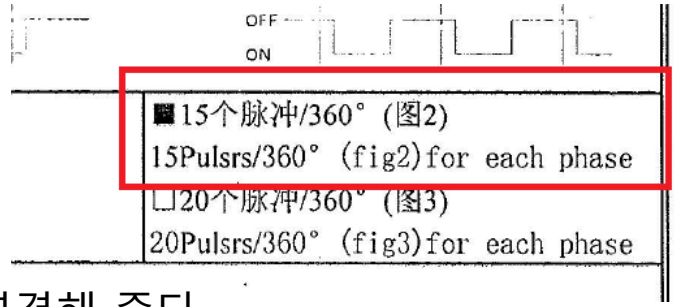


Table 145. Counting direction versus encoder signals

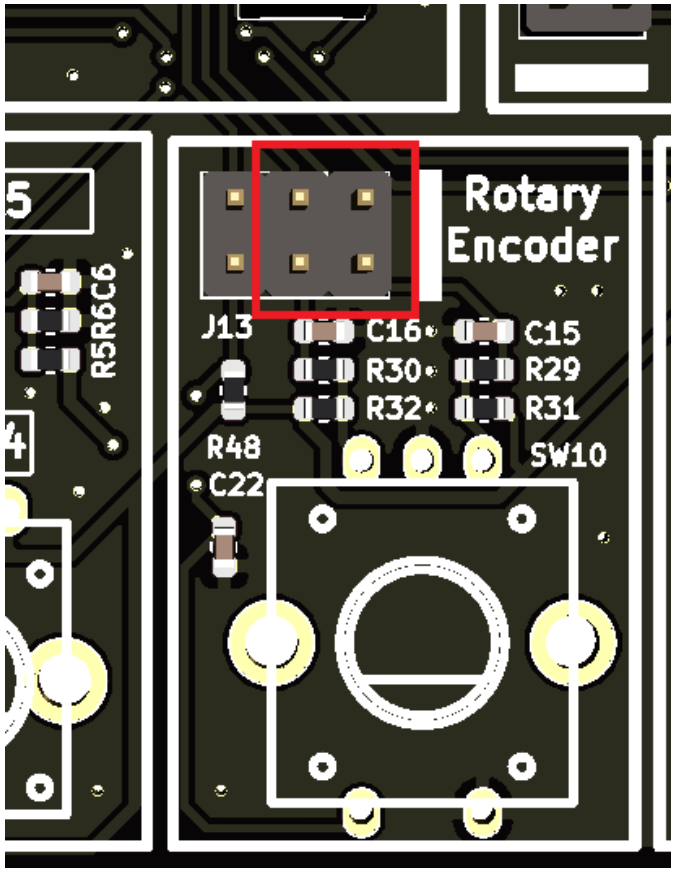
Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

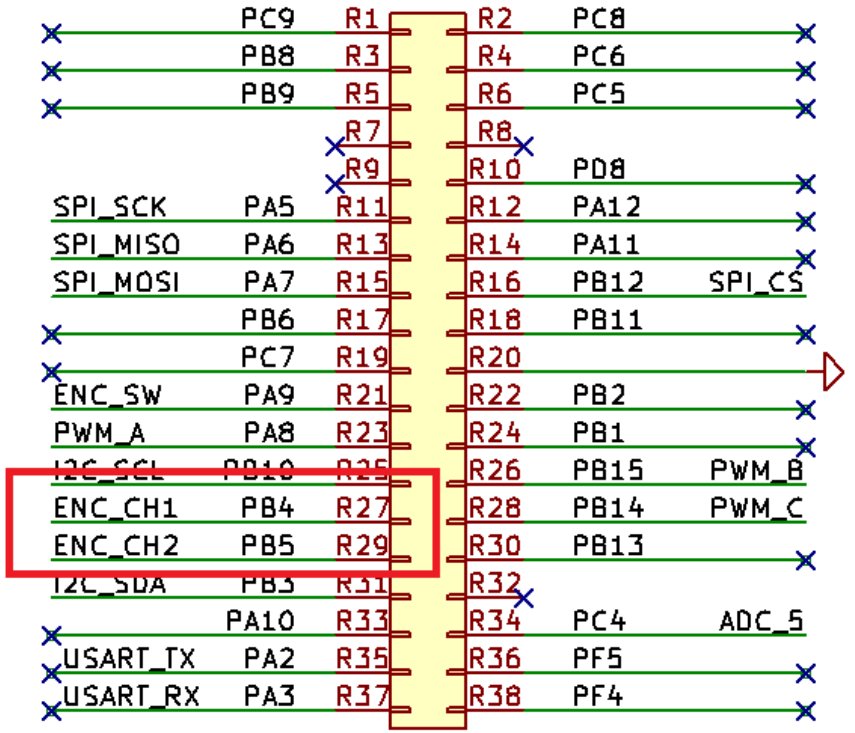
1. Encoder

• Timer Encoder Mode



- 보드의 Rotary Encoder 블록을 점퍼선으로 연결해, Nucleo에 연결해 준다.
- 엔코더의 A/B상 신호는 **Nucleo의 PB4, PB5**로 들어간다.
- 현재 보드에 연결 되어있는 엔코더는 **15cpt** 이다.

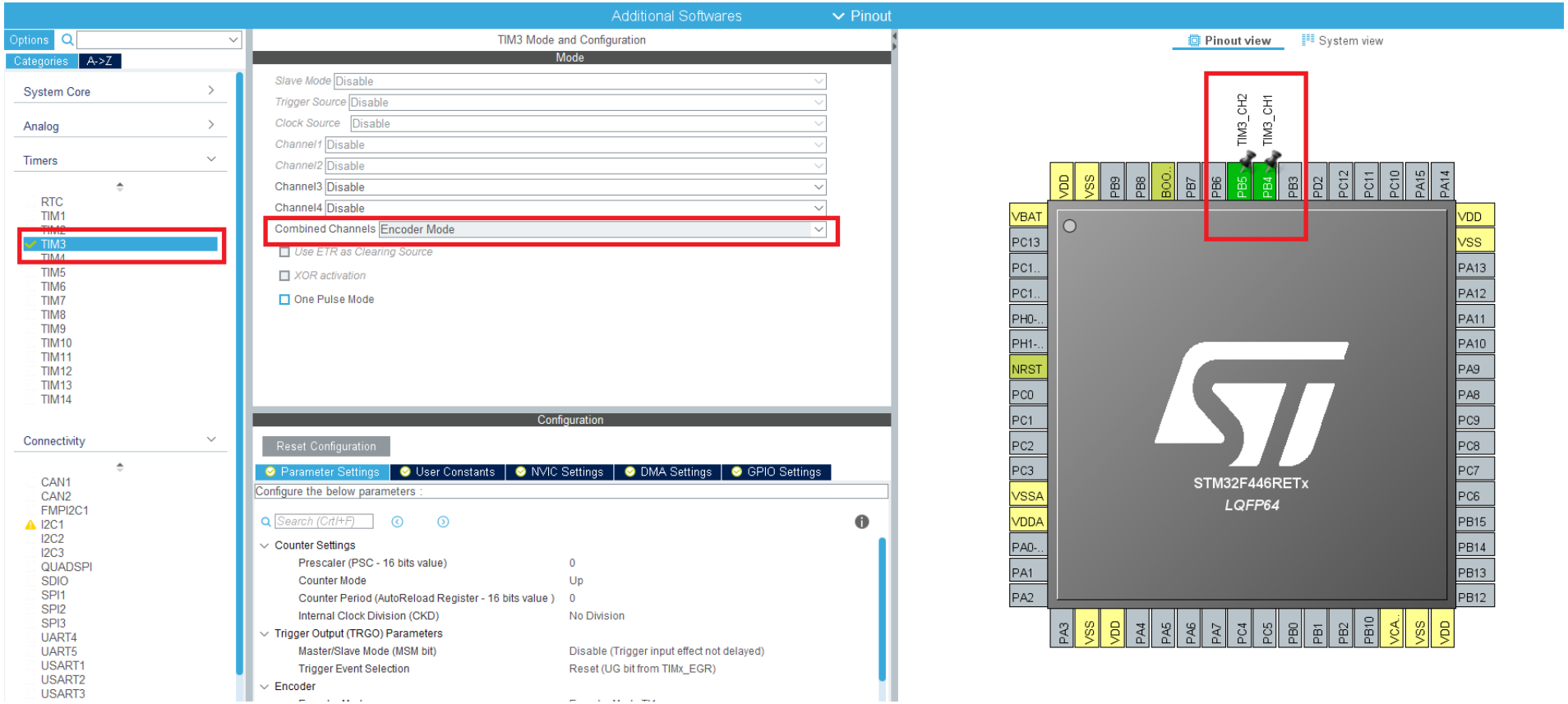




1. Encoder

• Timer Encoder Mode

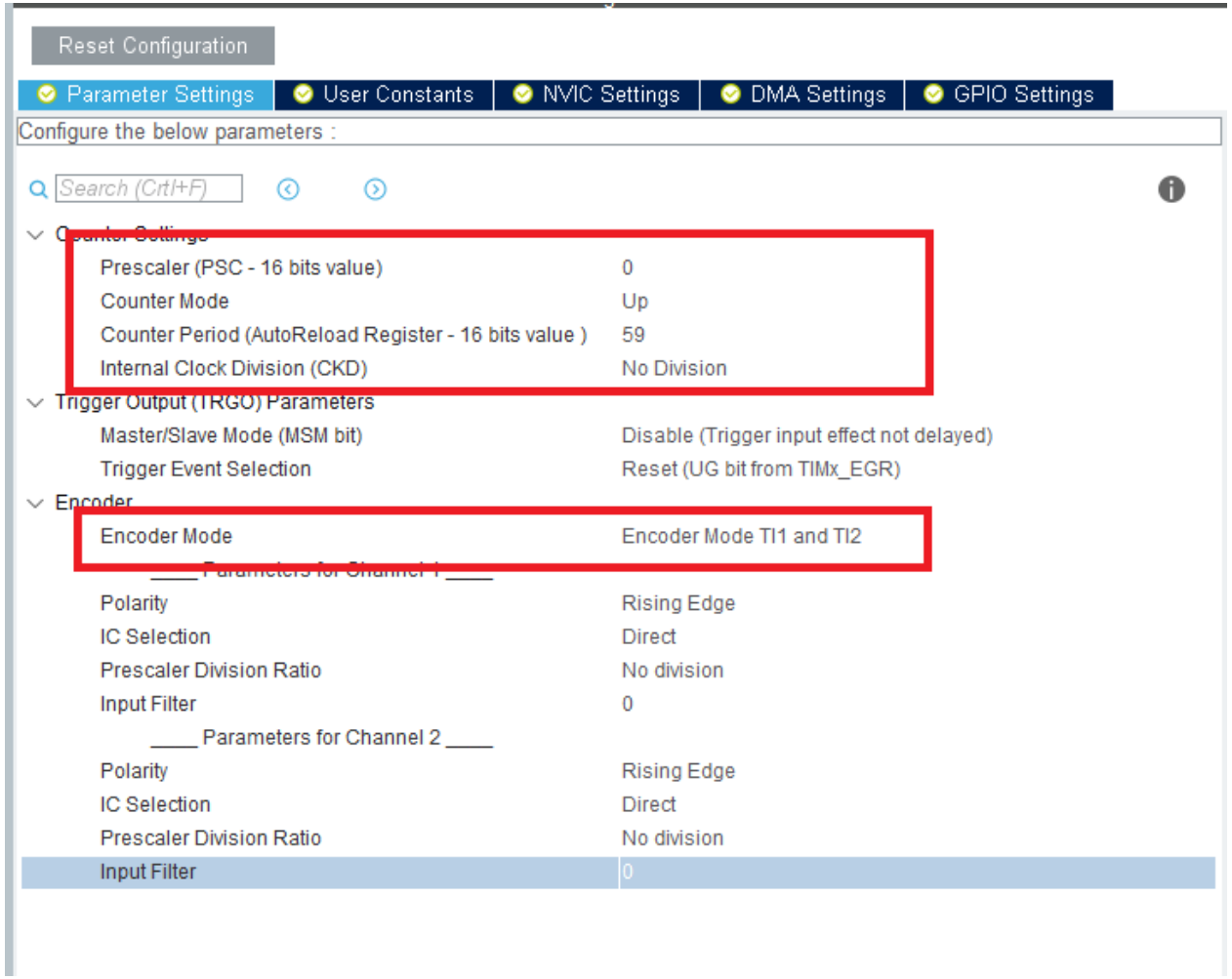
- PB4, PB5를 TIM3의 CH1, CH2로 설정한다.
- 그 다음, TIM3 탭에서, **Encoder Mode**를 선택한다.



1. Encoder

• Timer Encoder Mode

- Encoder Mode를 **Encoder Mode T1 and T2**로 선택한다.
- Counter Period를 **59**로 설정한다. ***(Why?)***
- 코드를 생성한다.



1. Encoder

• Timer Encoder Mode

- Main.c 상단에 엔코더 카운터 값을 저장할 변수를 선언한다.

```
/* Private user code ---------------
/* USER CODE BEGIN 0 */
uint16_t encoderCnt;
/* USER CODE END 0 */
```

1. Encoder

• Timer Encoder Mode

- **HAL_TIM_Encoder_Start**(Timer Instance, Timer channel)
- 특정 타이머의 엔코더 모드를 시작한다.

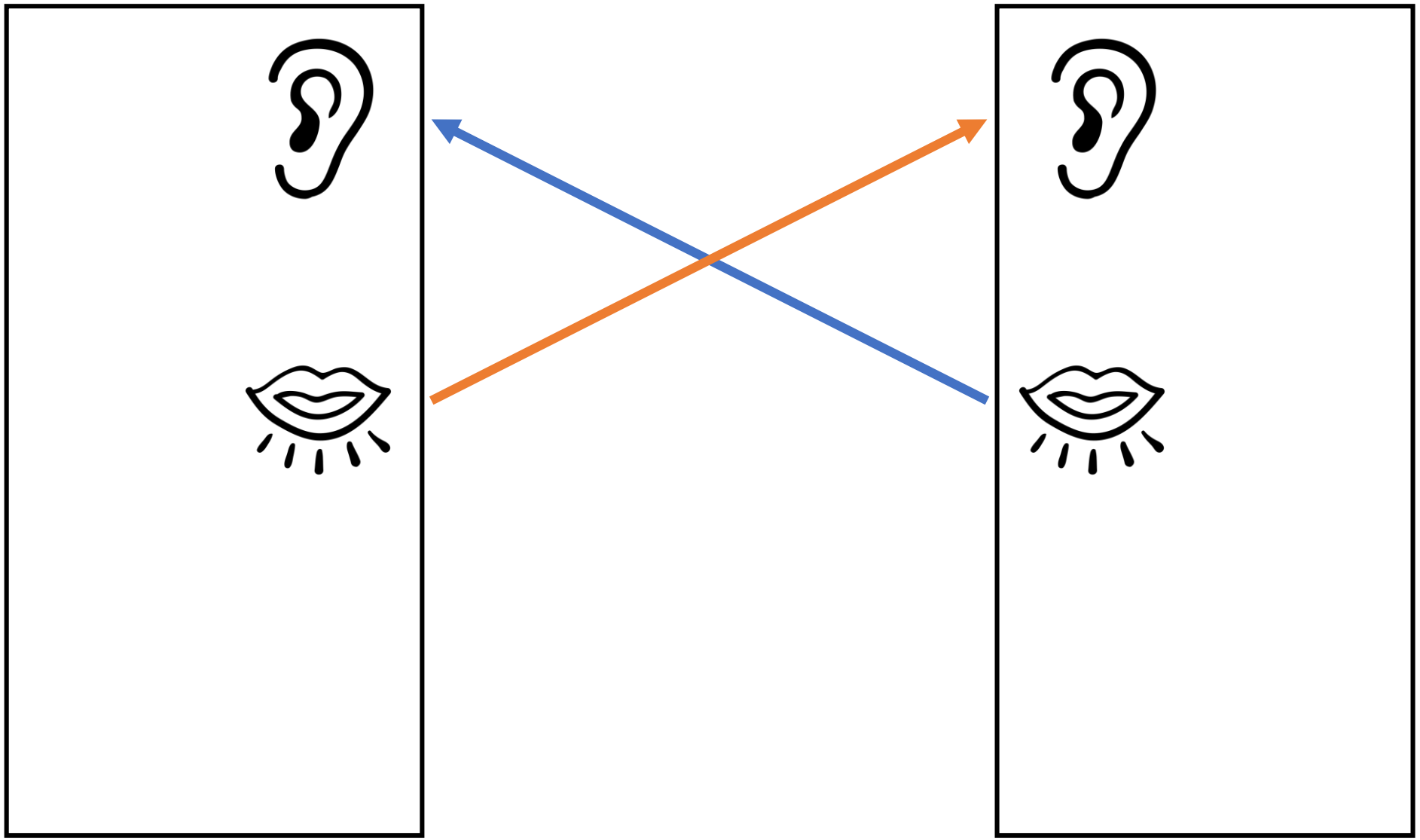
- 아래와 같이 입력한 후, Live Watch를 통해 encoderCnt값의 변화를 보자.

```
/* USER CODE BEGIN 2 */
HAL_TIM_Encoder_Start(&htim3, TIM_CHANNEL_ALL);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  encoderCnt = htim3.Instance->CNT;
  HAL_Delay(10);
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

2. UART

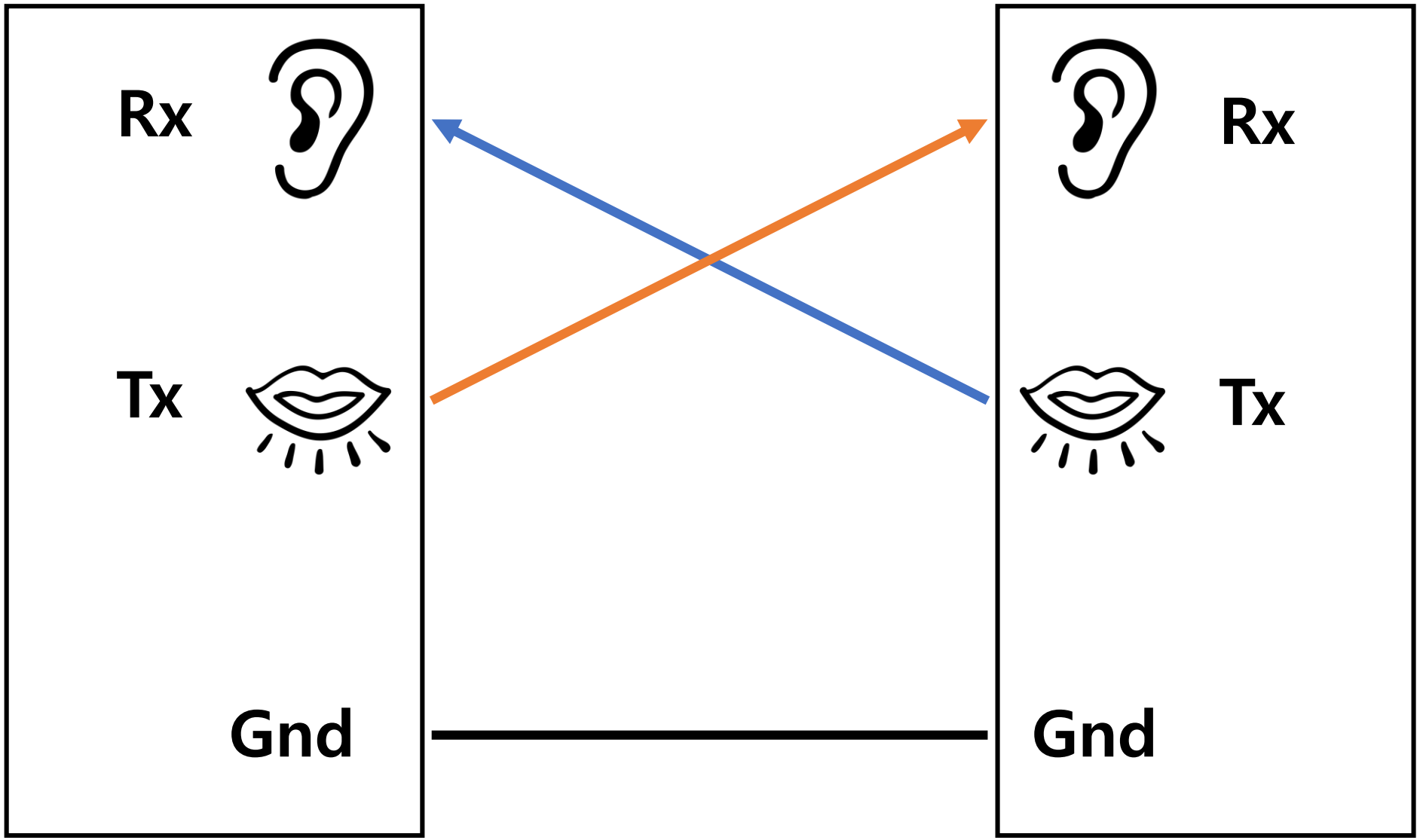
2. UART

- 사람은 대화를 어떻게 할까?



2. UART

- UART에서 연결은 다음과 같다.



2. UART

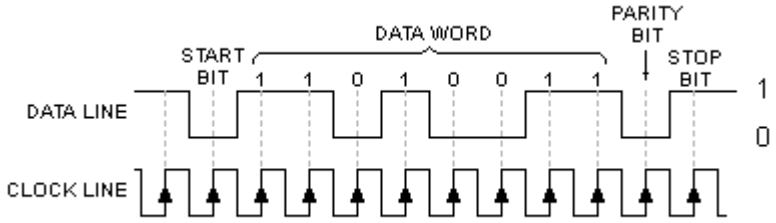
- UART(Universal Asynchronous Receiver/Transmitter)

- UART의 가장 간단한 통신 형태는 **시리얼 통신**으로 Rx, Tx, Gnd를 연결하여 구성한다.
- 이때, Rx와 Tx는 **서로 교차하여** 연결한다.

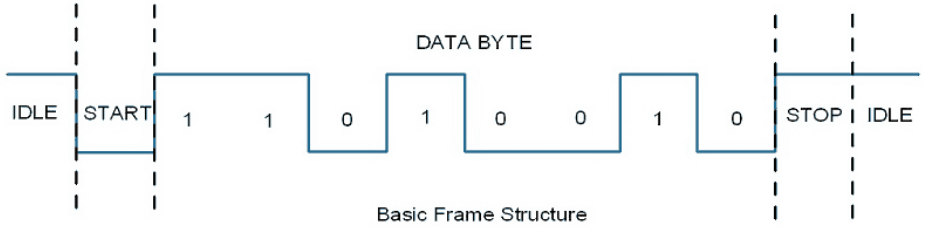
Q. Synchronous(동기)/Asynchronous(비동기) 통신이란?

A. 서로가 **동일한 클럭에 맞추어 통신**하는 방식을 **Synchronous 통신**이라 한다.
이러한 **동기 클럭원 없이 통신**하는 방식을 **Asynchronous 통신**이라 한다. 비동기 통신의 경우, 서로 간에 통신속도를 알 수 없기 때문에 **통신속도를 서로 미리 약속해야 한다.**

시리얼 통신에서는 이러한 통신속도를 **Baud Rate(보드 레이트)**라고 하며, 서로간에 미리 맞추어 주어야 정상적인 통신이 된다.



Synchronous

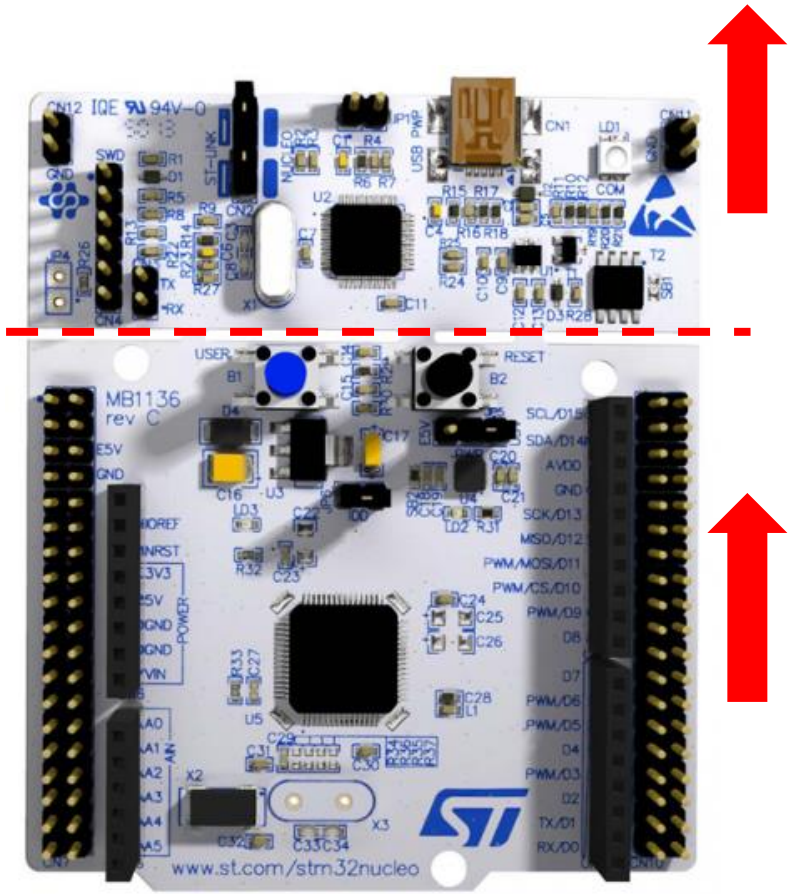


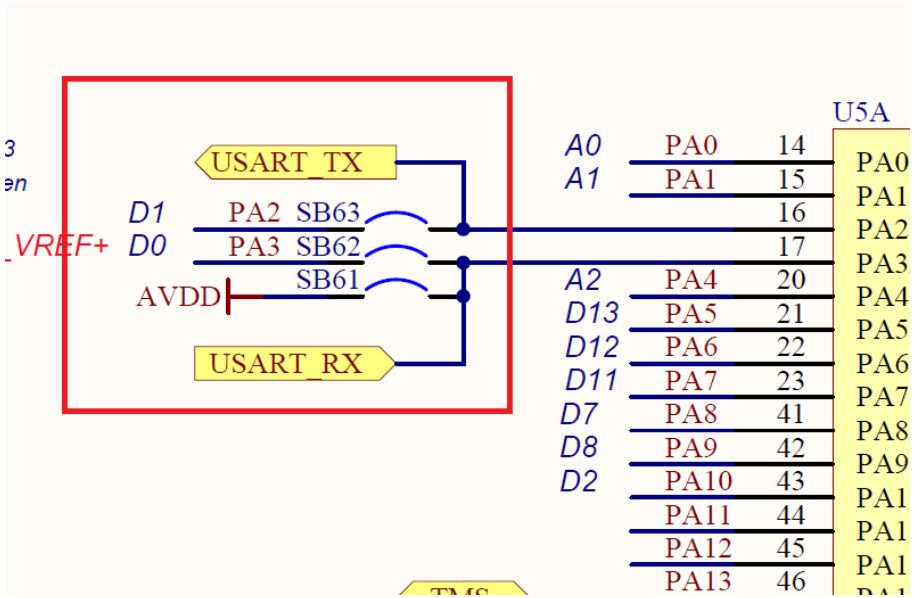
Asynchronous (UART Frame)

2. UART

• UART 실습

- 누클레오 보드의 특정 UART(PA2/PA3)는 내부적으로 디버거와 연결되어 있다.
- 디버거는 USB to UART 기능을 내장하고 있어, 추가적인 장치 없이 컴퓨터와 바로 UART 통신을 할 수 있다.

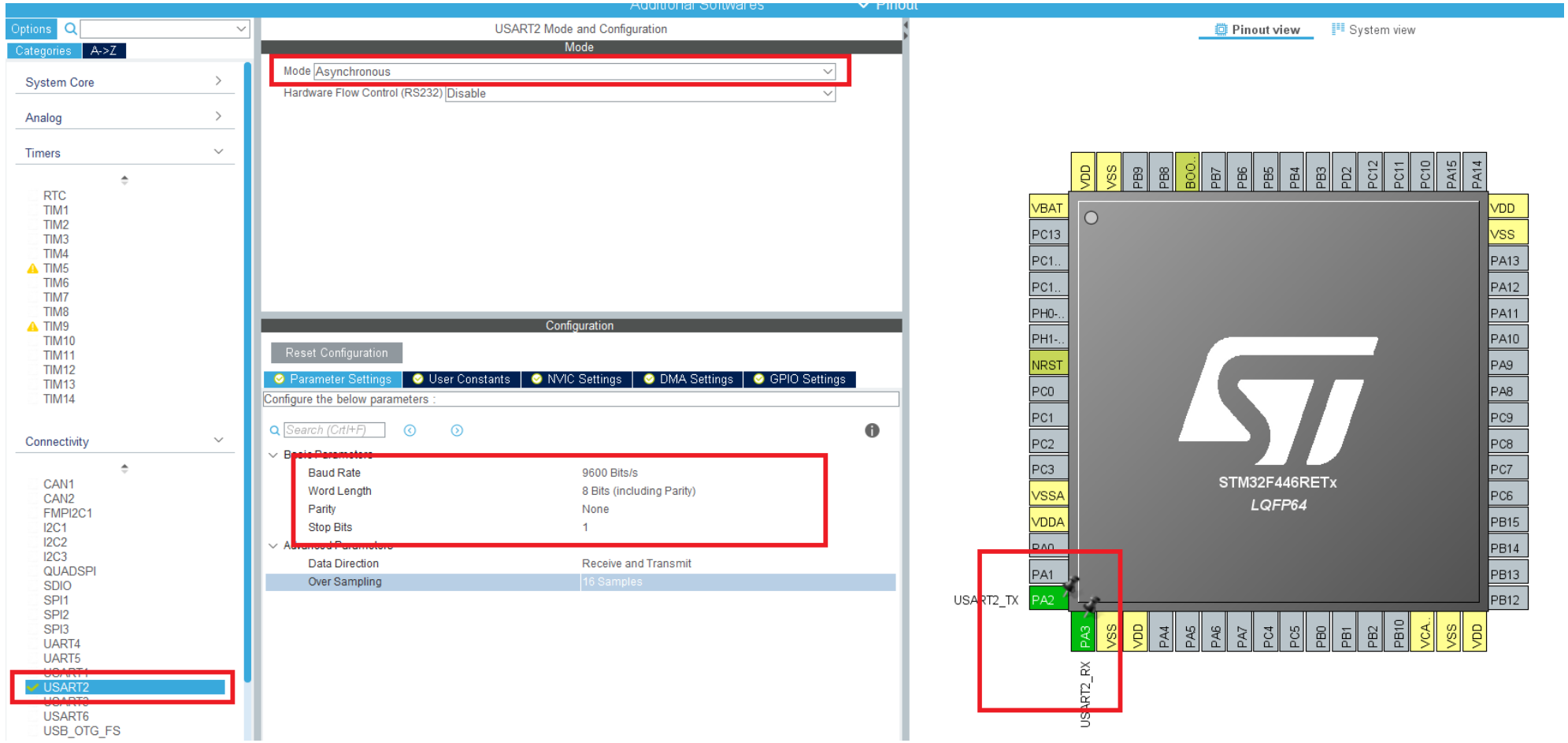




2. UART

- **UART 실습**

- PA2/PA3을 UART로 설정한 후, 아래와 같이 설정한다.
- 그 후에 코드를 생성한다.



2. UART

• UART 실습

- 사용자가 컴퓨터에서 보낸 텍스트를 그대로 되돌려 주는 에코(Echo)기능을 구현할 것이다.
- Main.c 상단에 주고 받은 데이터를 저장할 변수를 선언한다.

```
/* Private user code ---------------------------
/* USER CODE BEGIN 0 */
uint8_t txData, rxData;
/* USER CODE END 0 */
```

2. UART

- **UART 실습**

- **HAL_UART_Receive**(UART Instance, rxData, size, TimeOut)
- **HAL_UART_Transmit**(UART Instance, txData, size, TimeOut)

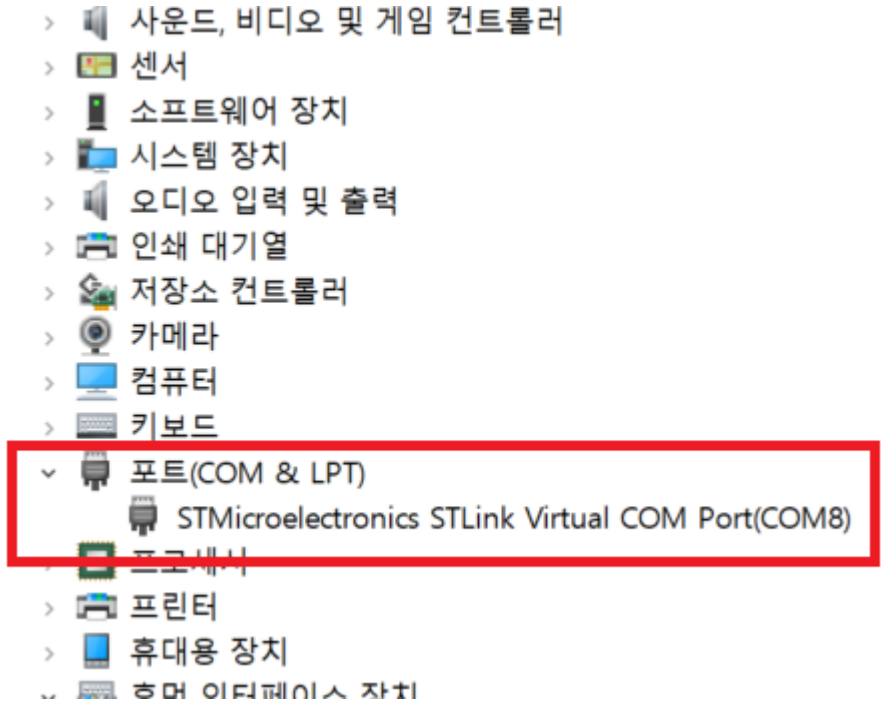
- 아래와 같이 입력한 후, 업로드를 하자.

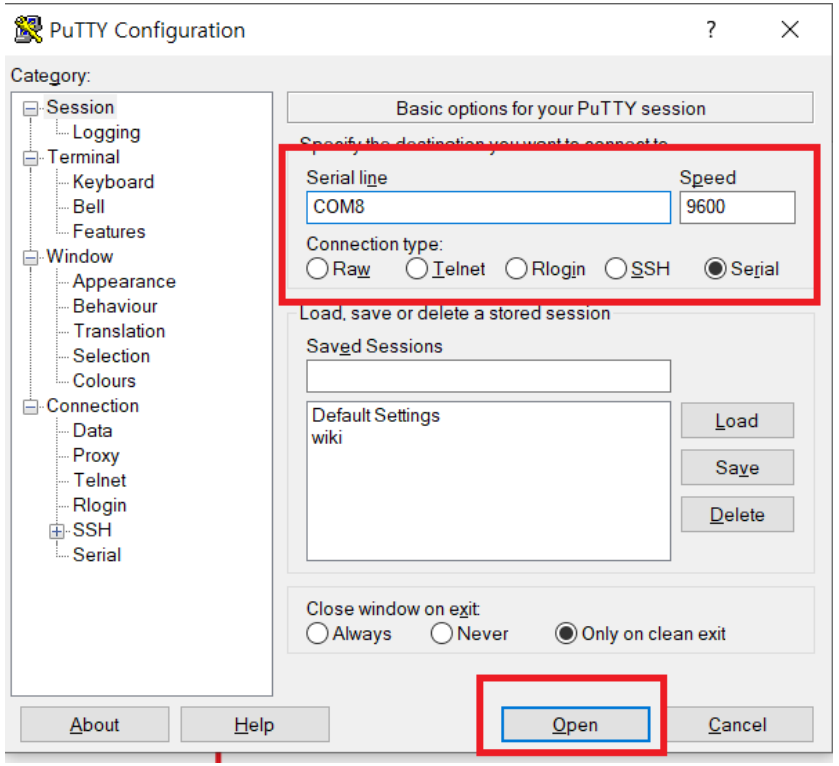
```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  if(HAL_UART_Receive(&huart2, &rxData, 1, 1000) == HAL_OK){
    txData = rxData;
    HAL_UART_Transmit(&huart2, &txData, 1, 1000);
  }
  /* USER CODE END WHILE */
```

2. UART

• UART 실습

- Putty를 설치한다.
- 장치관리자에 들어가 Nucleo의 포트 번호를 알아낸다.
- Putty에서 Serial을 선택하고, 해당 포트 번호를 입력한다. 또한 보드레이트도 맞춰준다.
- Open을 누른 후, 키보드로 입력한 글자가 그대로 되돌아 오는지 확인해보자.



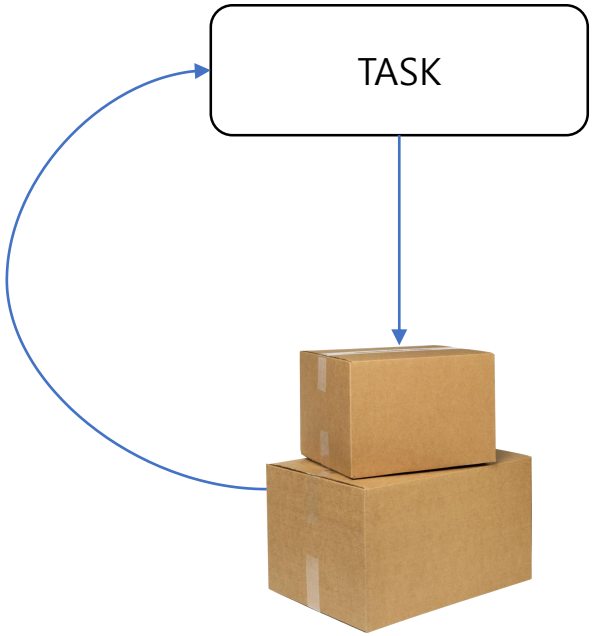


3. DMA

3. DMA

- 나는 지금 너무 바쁜데 지난주에 시킨 택배가 언제 올지 모른다.
- 택배가 왔는지 아닌지 확인하기 위하여 어떻게 해야 할까?
- **근데 나는 지금 너무 바빠서 하고 있는 일을 도저히 중단할 수가 없다.**

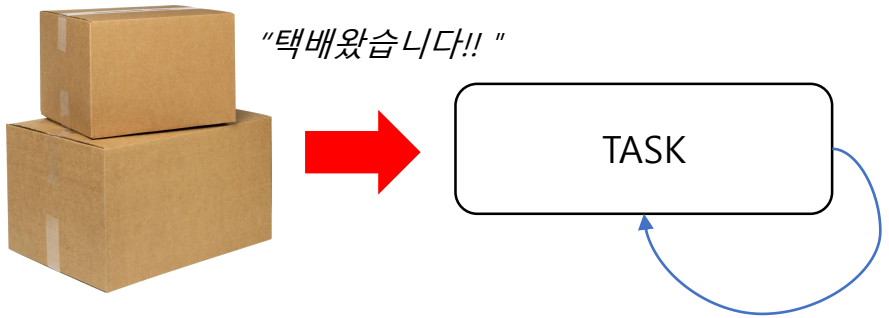
3. DMA



[Polling]

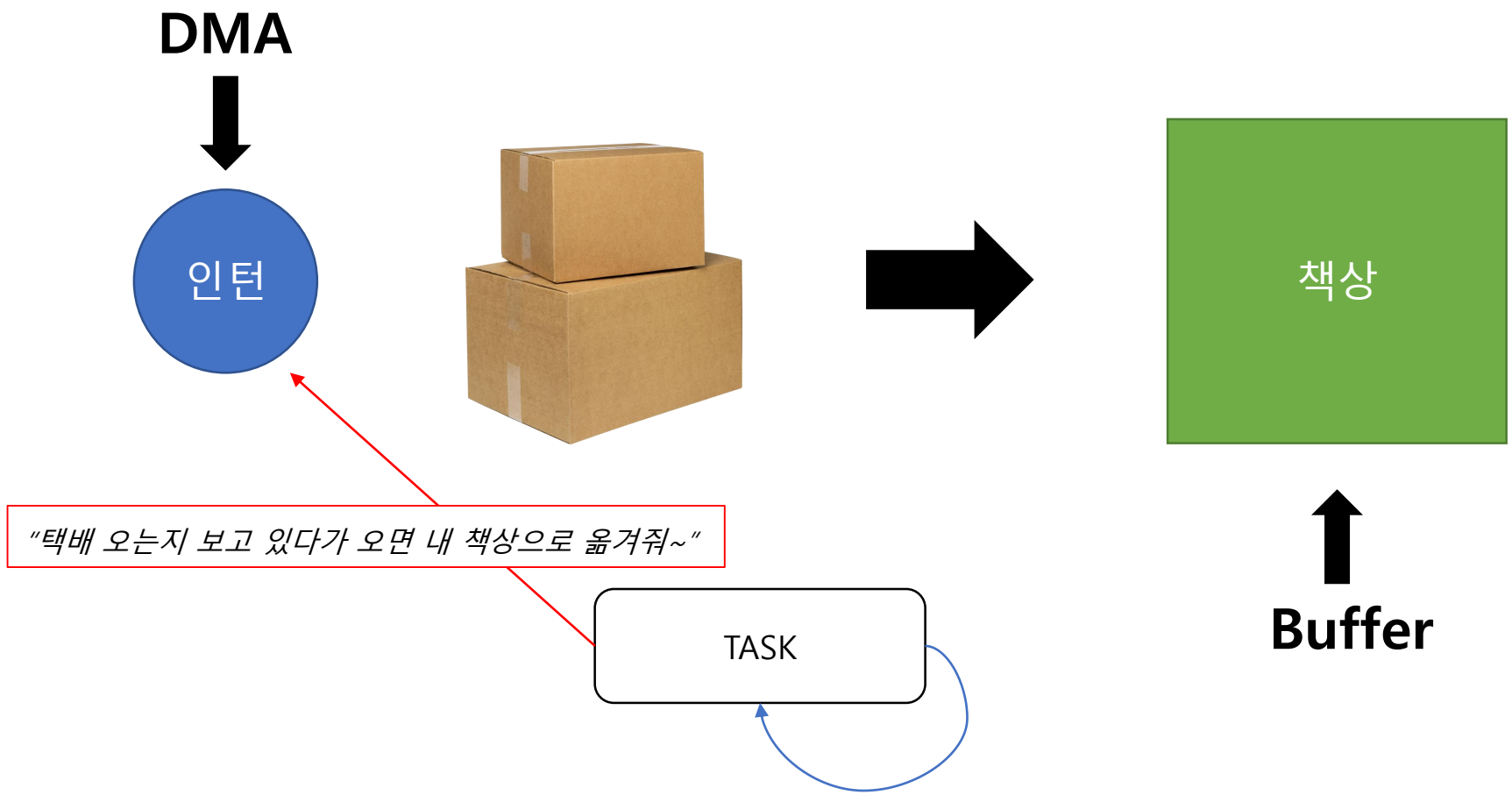
MCU에선 누구에게 이 사실을 알릴까? -> NVIC

"아저씨 택배 도착하면 010-XXXX-XXXX로 문자남겨주세요 "



[Interrupt]

3. DMA



[제 3의 방법?]

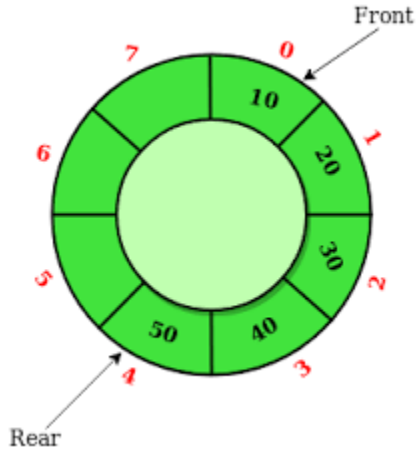
3. DMA

- DMA(Direct Memory Access)

- DMA는 다음과 같은 세가지 모드가 존재한다.
  - ➢ Memory To Peripheral
  - ➢ Peripheral To Memory
  - ➢ Memory To Memory

Q. DMA가 이러한 역할을 하기 위해 알아야 할 정보는 무엇인가?

A. 출발지 주소, 도착지 주소, 데이터 타입



Q. 데이터가 너무 많이 들어와서 버퍼가(책상이) 다 차버리면 어떻게 되는가?

A. Normal Type, Circular Type Buffer

Q. 데이터가 얼마나 들어왔는지, 혹은 얼마나 나갔는지는 어떻게 알 수 있나?

A. NDTR Value : DMA의 레지스터 값으로, 현재 버퍼가 얼마나 남았는지를 알려준다.