
ROBIT
ROBOT SPORT GAME TEAM

Team. ROBIT | # STM32 LL Driver

16th 백종욱

# INDEX

# STM

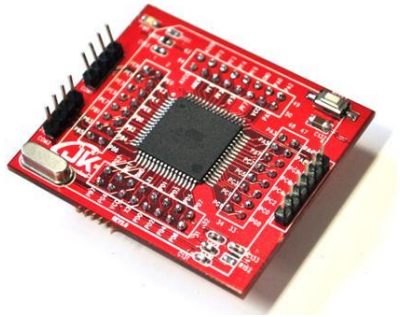
# STM32F446RE


ST 마이크로 일렉트로닉스 제품의 마이크로 칩

제품의 성능

- 512 Kbytes of Flash memory
- 128 Kbytes of SRAM
- Flexible external memory controller with up to 16-bit data bus: SRAM, PSRAM, SDRAM/LPSDR SDRAM, NOR/NAND Flash memories
- All devices offer three 12-bit ADC
- 12 general-purpose 16-bit timers including two PWM timers for motor control,
- two general-purpose 32-bit timers.

# STM32cubeIDE



Atmel Studio

eSoftner

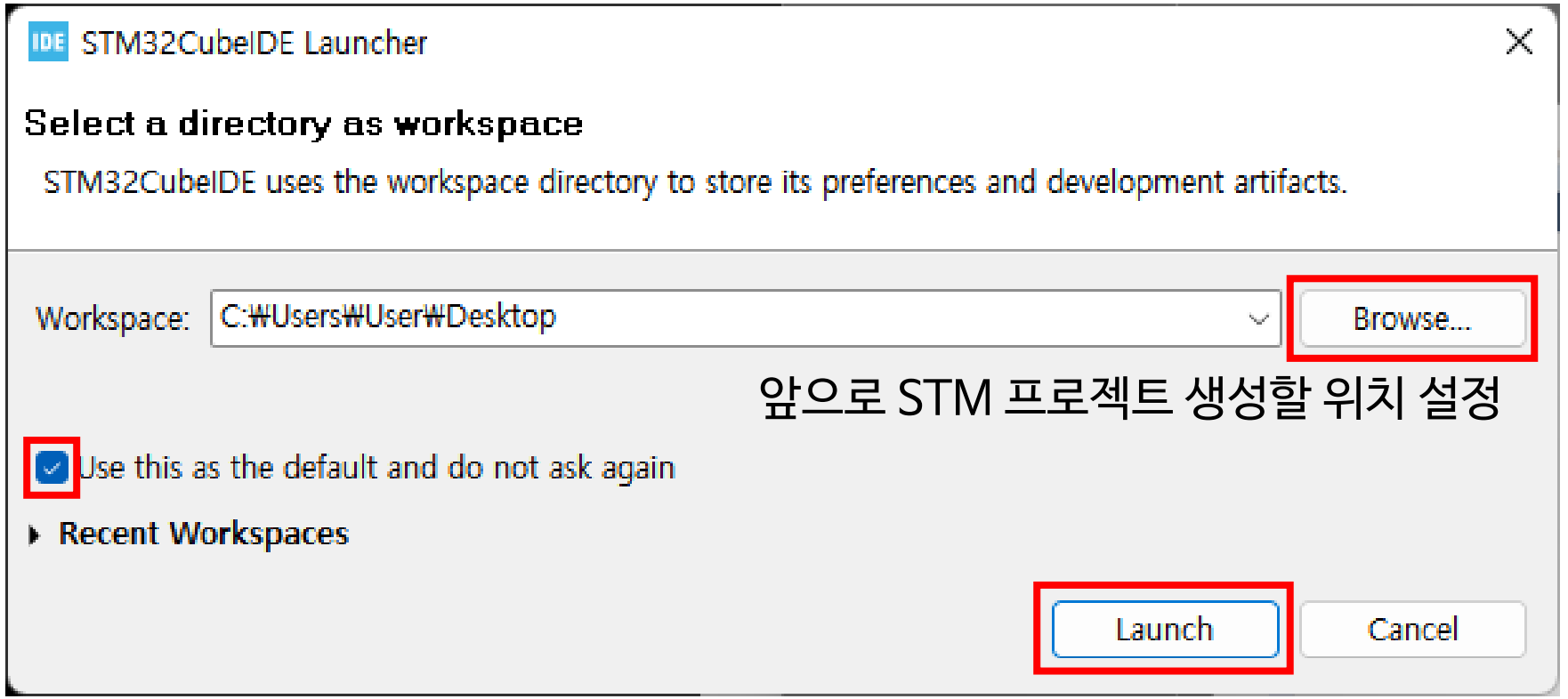
WINGKIT


STM32 CubeIDE

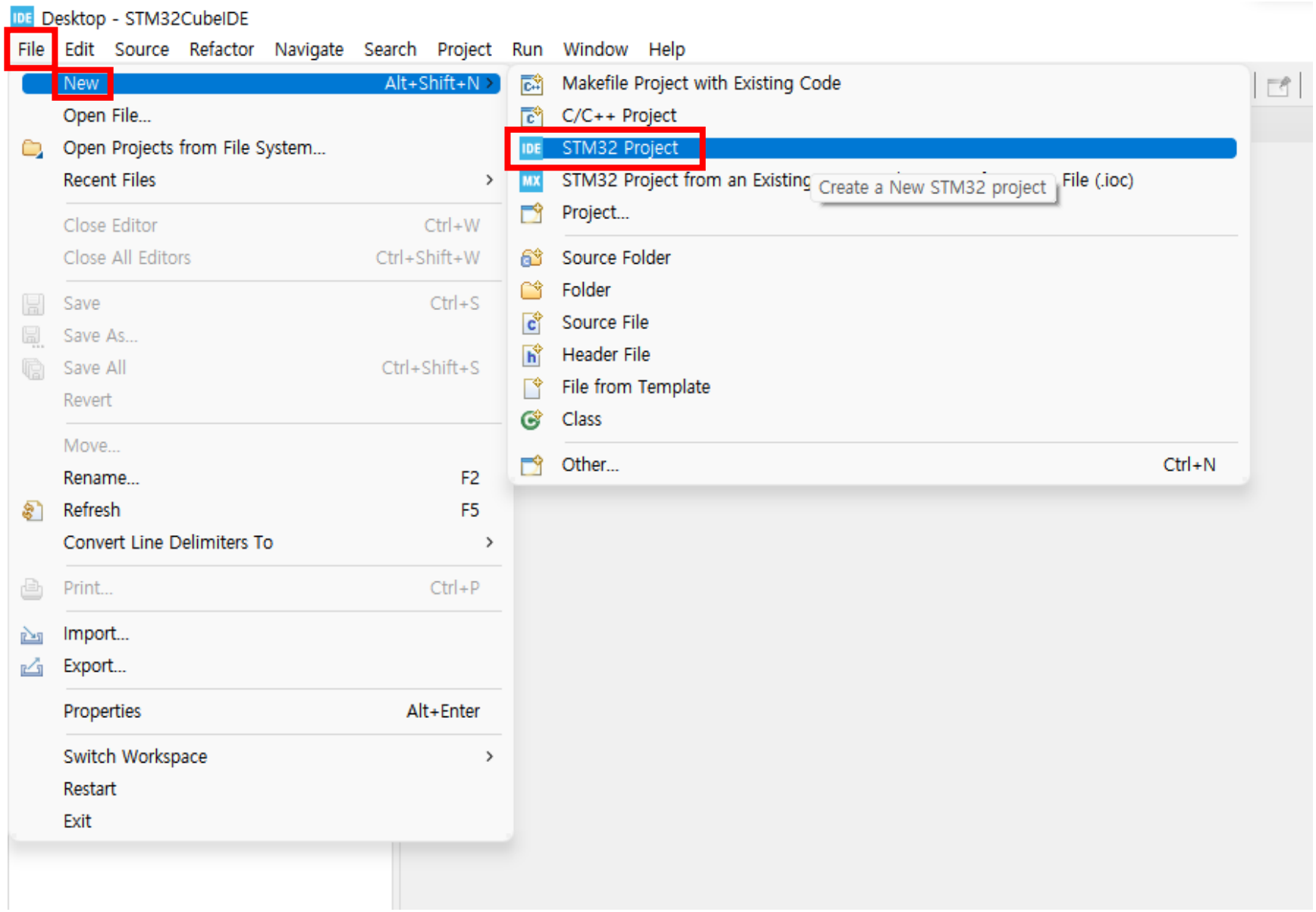
# STM 프로젝트 생성

# 프로젝트 생성


IDE
STM32Cub...
1.10.1
STMStudio

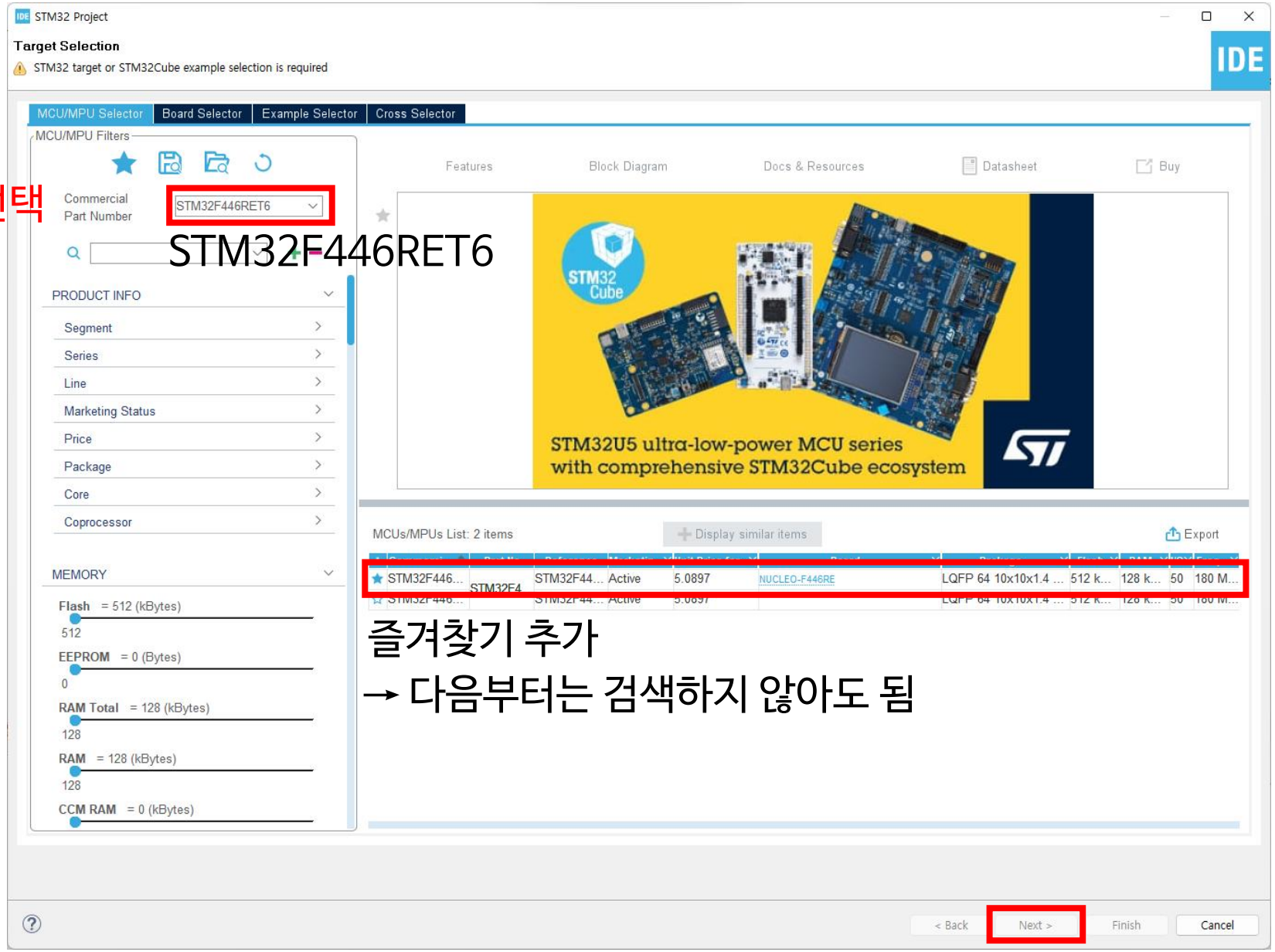

STM32CubeIDE Launcher
Select a directory as workspace
STM32CubeIDE uses the workspace directory to store its preferences and development artifacts.
Workspace:
C:₩Users₩User₩Desktop
Browse...
앞으로 STM 프로젝트 생성할 위치 설정
Use this as the default and do not ask again
Recent Workspaces
Launch
Cancel

# 프로젝트 생성


Desktop - STM32CubeIDE
File
Edit
Source
Refactor
Navigate
Search
Project
Run
Window
Help
New Alt+Shift+N
Open File...
Open Projects from File System...
Recent Files
Close Editor Ctrl+W
Close All Editors Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Import...
Export...
Properties Alt+Enter
Switch Workspace
Restart
Exit
Makefile Project with Existing Code
C/C++ Project
STM32 Project
STM32 Project from an Existing
Create a New STM32 project
File (.ioc)
Project...
Source Folder
Folder
Source File
Header File
File from Template
Class
Other... Ctrl+N

# 프로젝트 생성

사용하는 MCU 선택



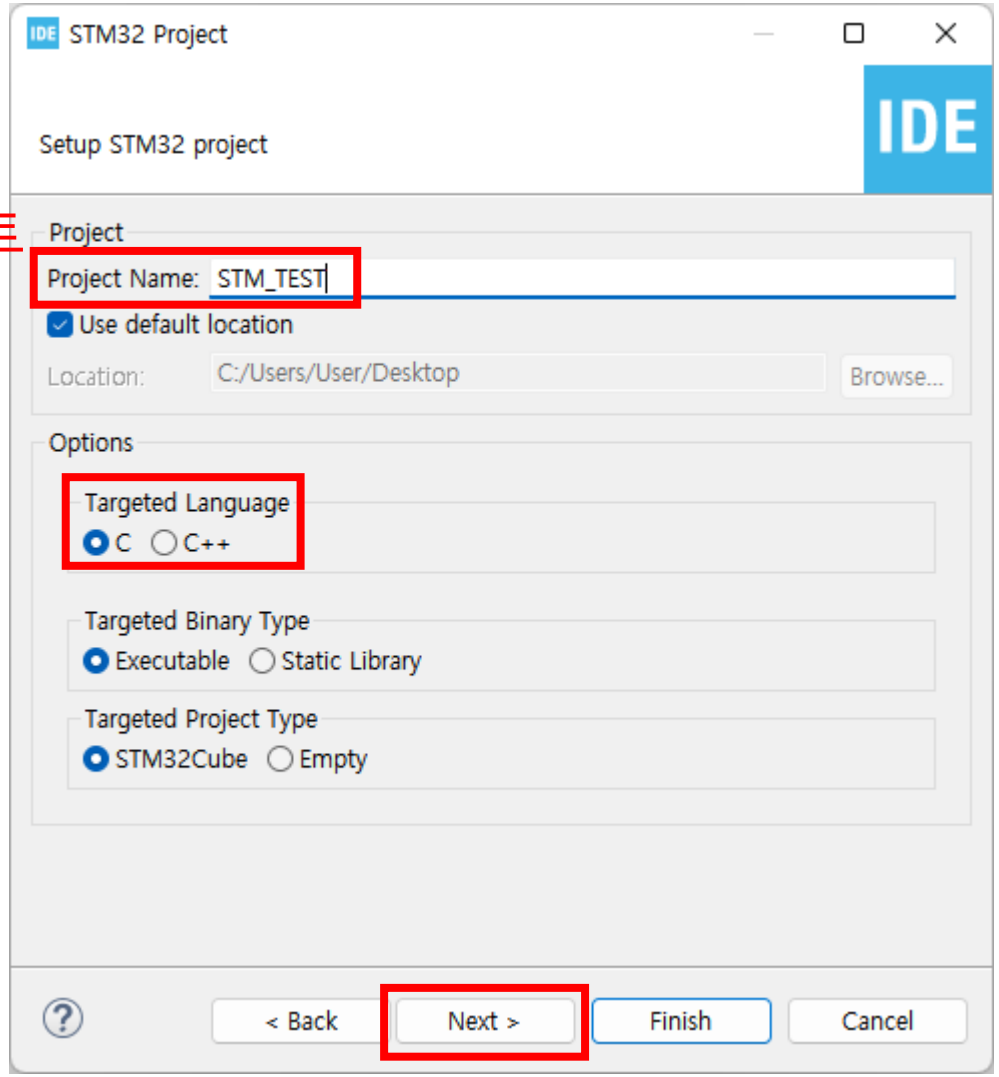

STM32F446RET6

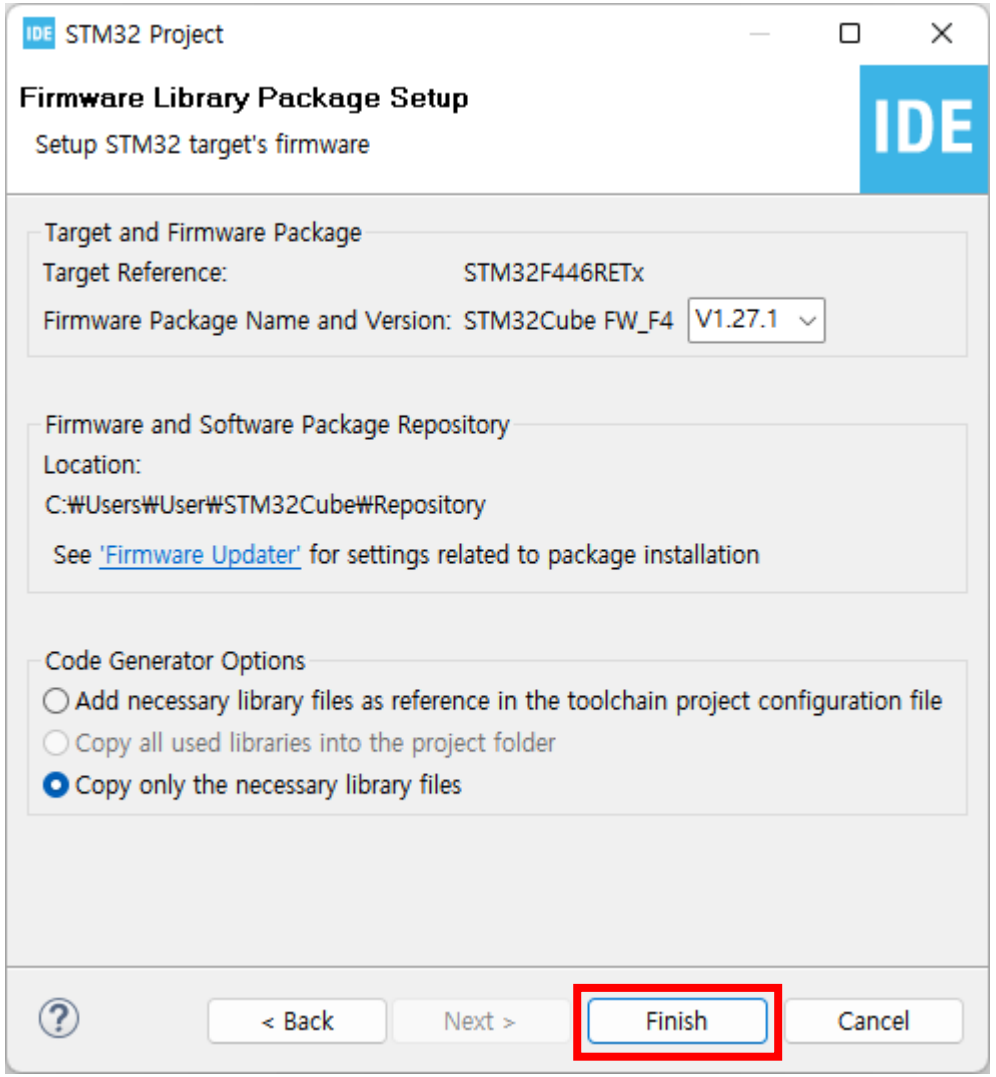
즐겨찾기 추가

→ 다음부터는 검색하지 않아도 됨

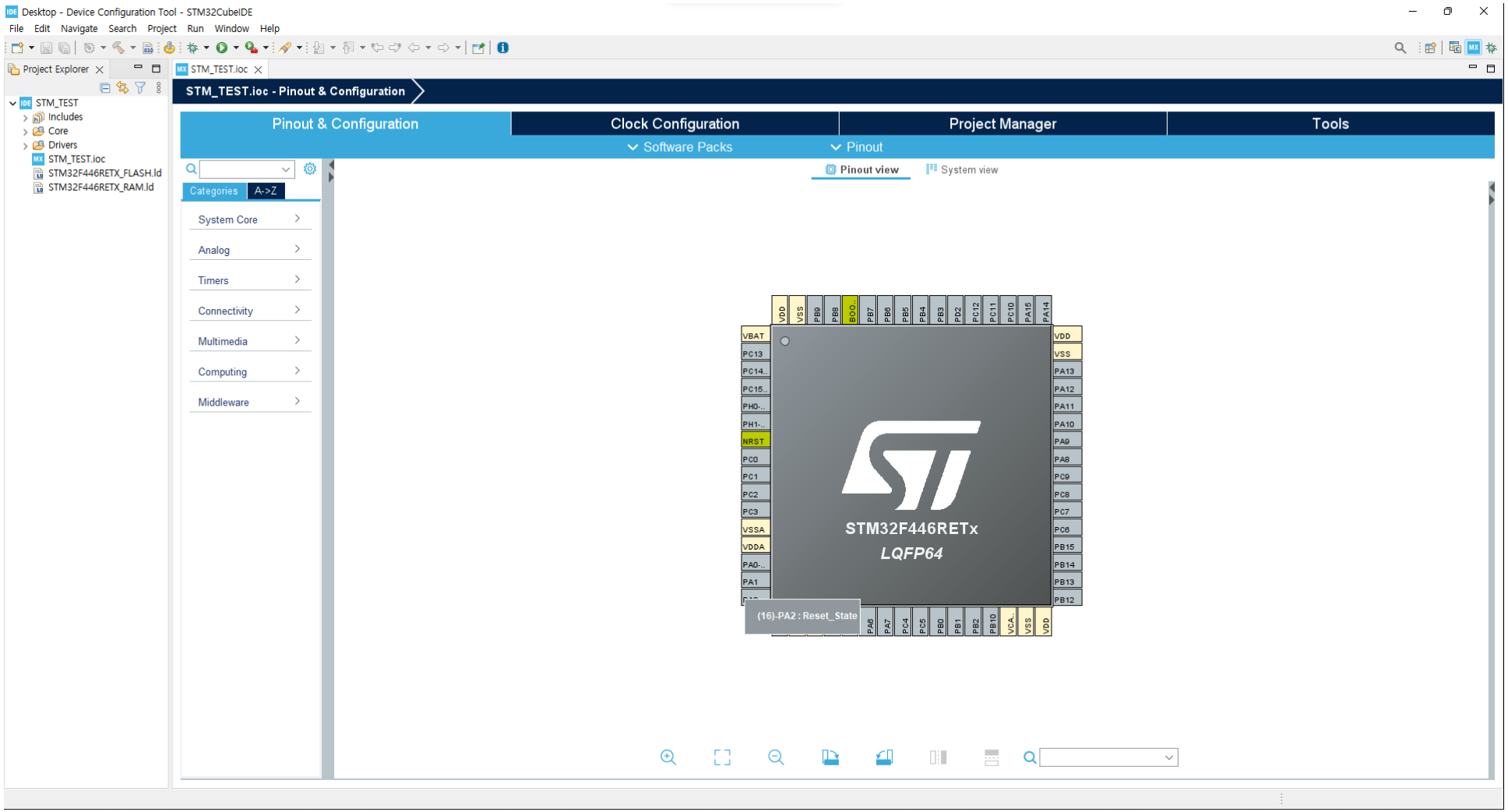
# 프로젝트 생성

프로젝트 이름



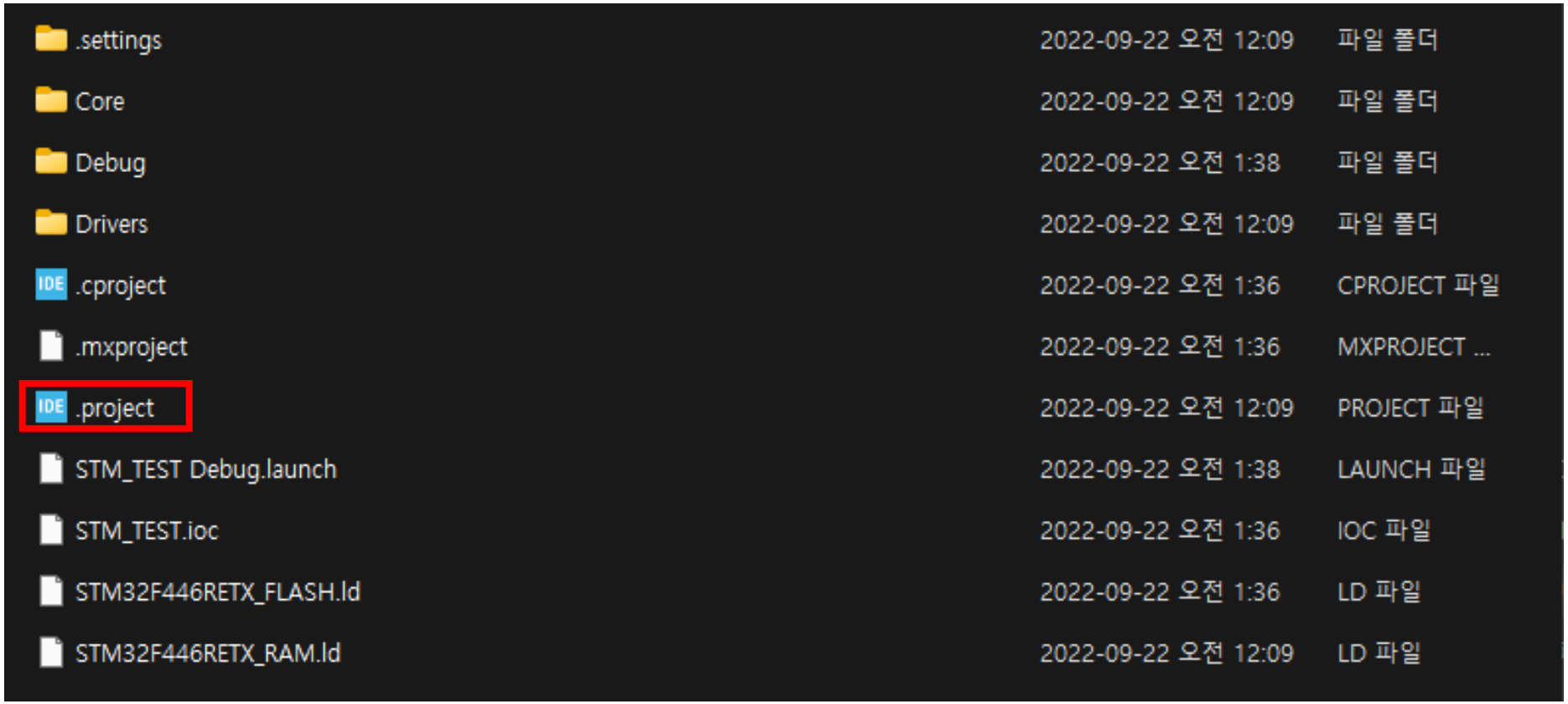



# 프로젝트 생성



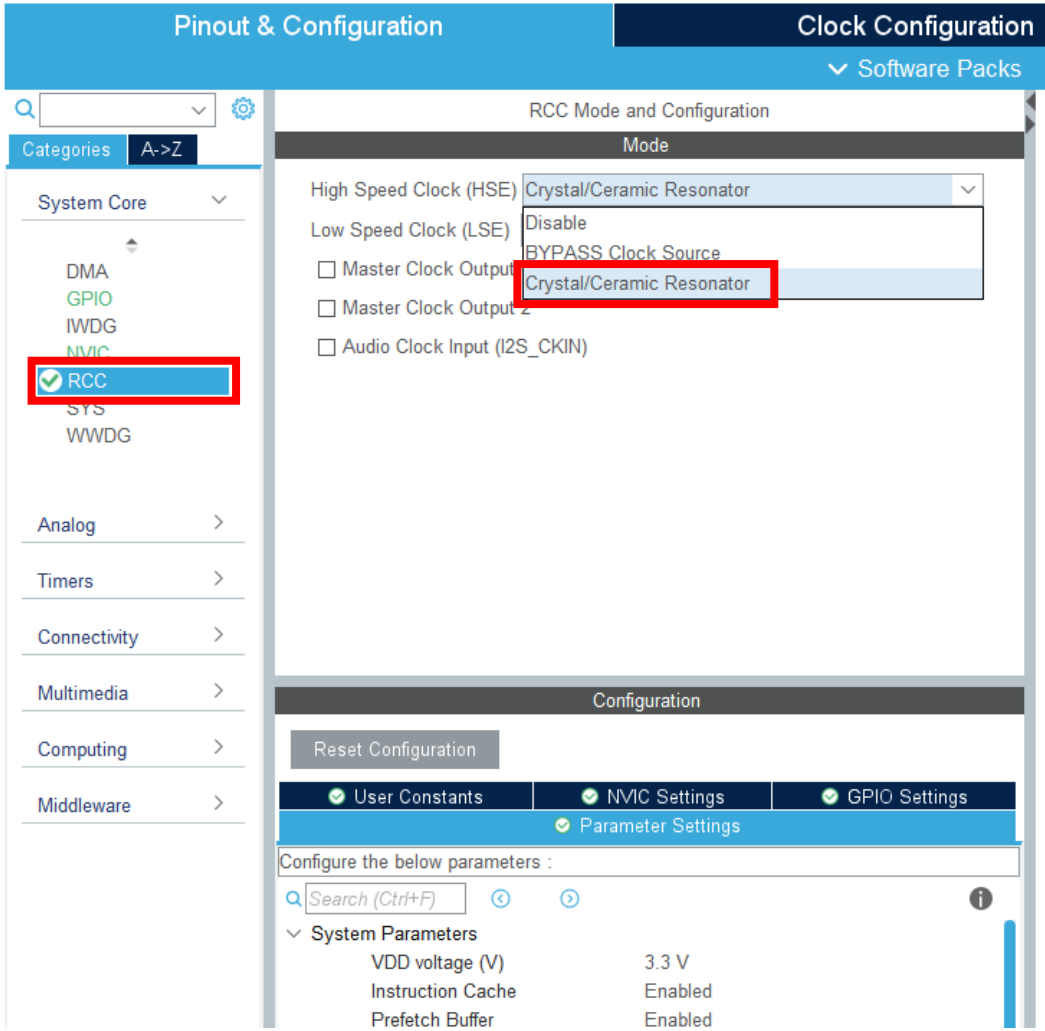

이렇게 뜨면 성공

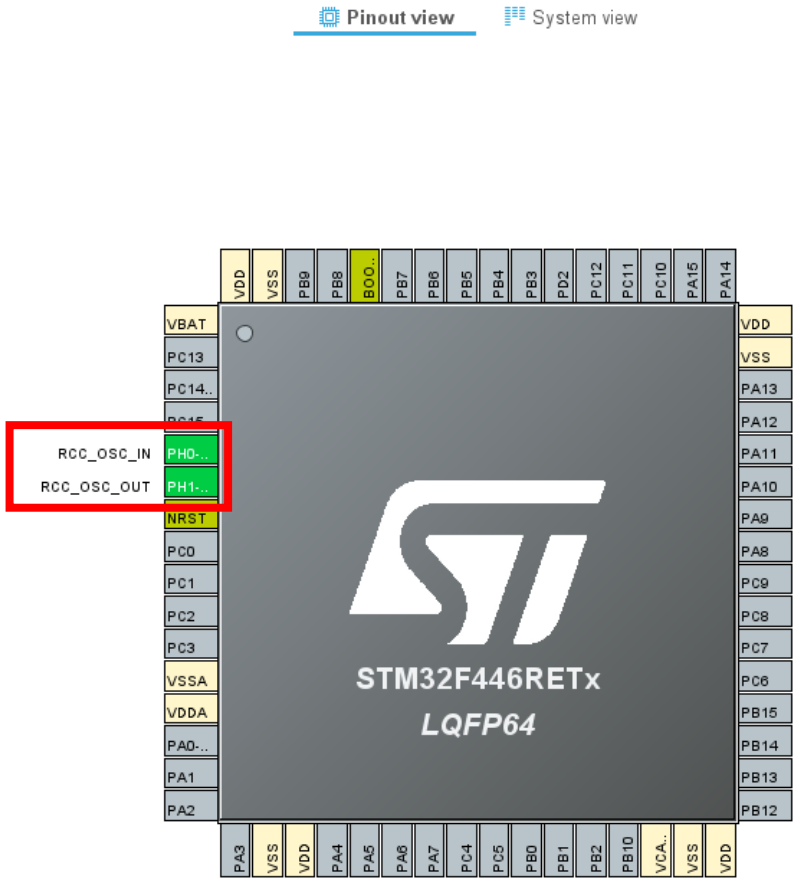
# 프로젝트 실행


.settings 2022-09-22 오전 12:09 파일 폴더
Core 2022-09-22 오전 12:09 파일 폴더
Debug 2022-09-22 오전 1:38 파일 폴더
Drivers 2022-09-22 오전 12:09 파일 폴더
.cproject 2022-09-22 오전 1:36 CPROJECT 파일
.mxproject 2022-09-22 오전 1:36 MXPROJECT ...
.project 2022-09-22 오전 12:09 PROJECT 파일
STM_TEST Debug.launch 2022-09-22 오전 1:38 LAUNCH 파일
STM_TEST.ioc 2022-09-22 오전 1:36 IOC 파일
STM32F446RETX_FLASH.ld 2022-09-22 오전 1:36 LD 파일
STM32F446RETX_RAM.ld 2022-09-22 오전 12:09 LD 파일

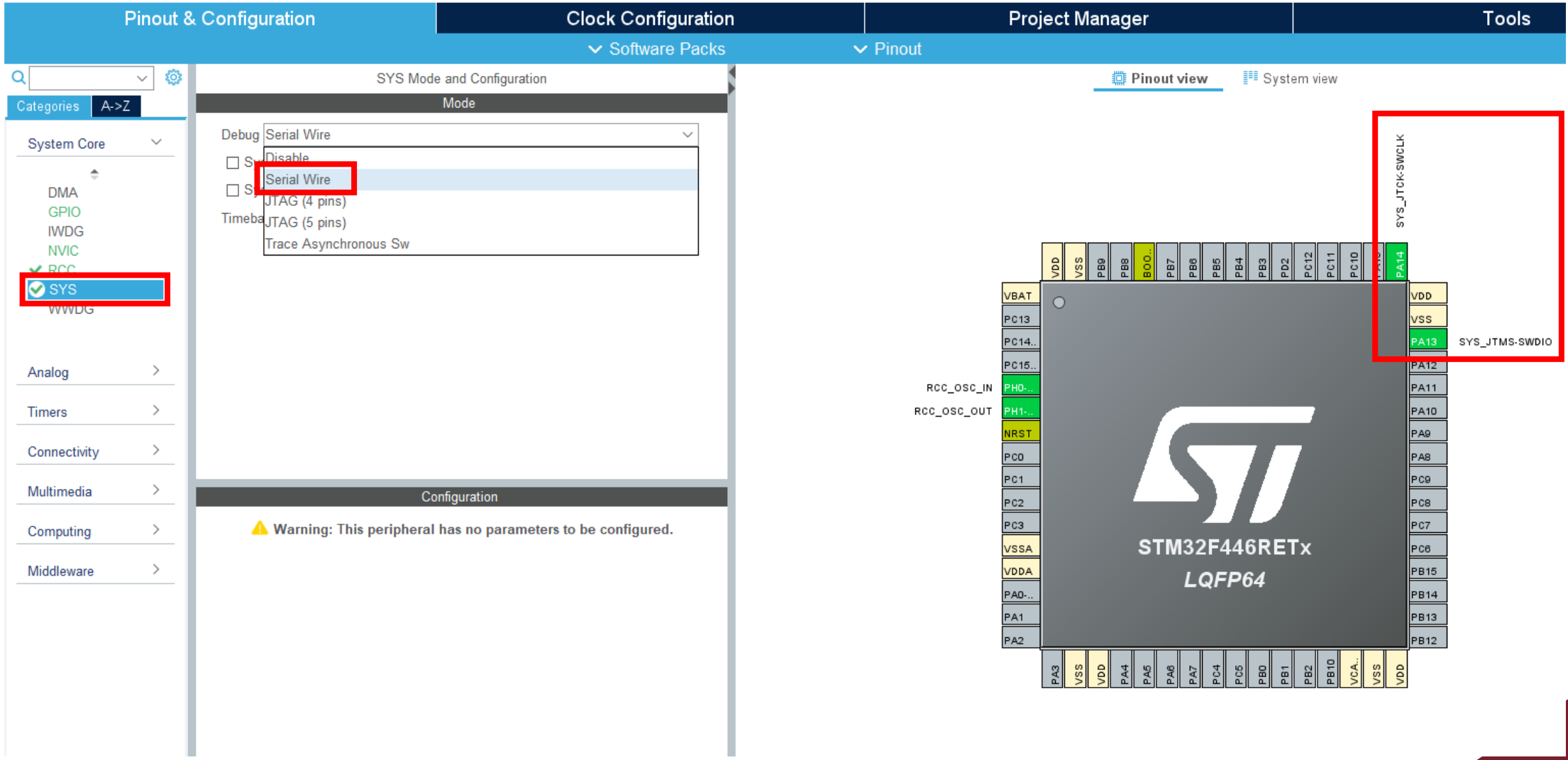
# 초기 설정

# 오실레이터 설정

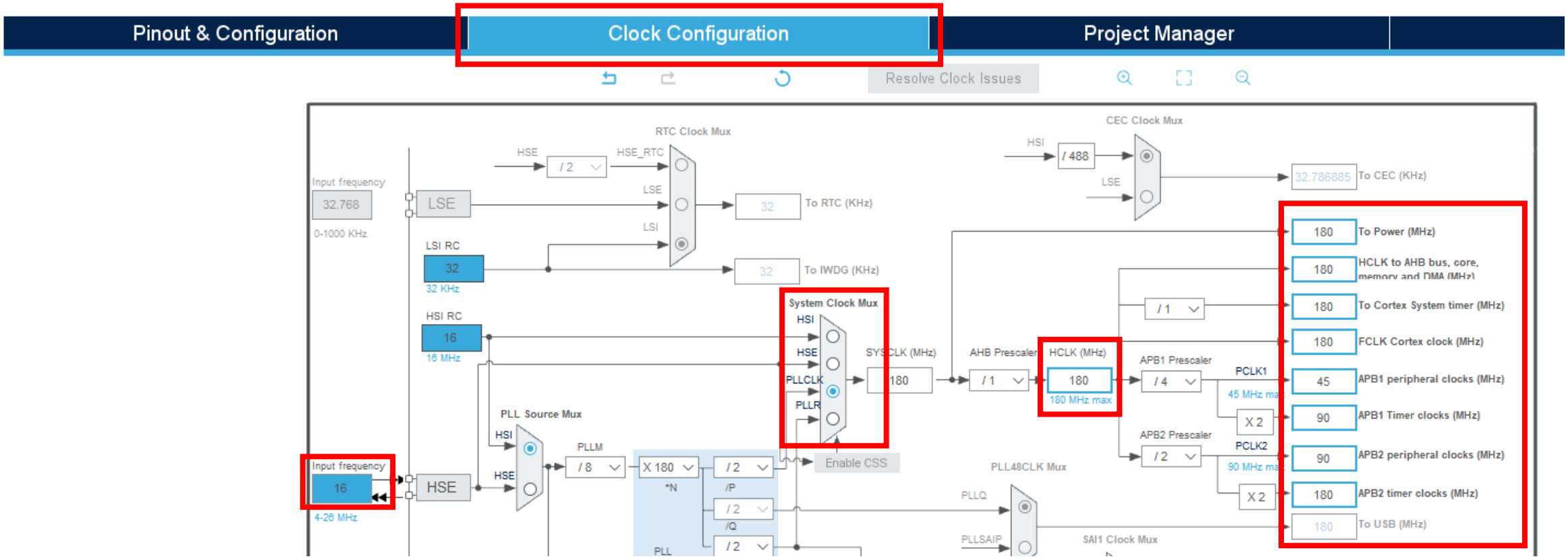

Pinout & Configuration
Clock Configuration
Project Manager
To
Software Packs
Pinout
Categories
A->Z
System Core
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG
Analog
Timers
Connectivity
Multimedia
Computing
Middleware
RCC Mode and Configuration
Mode
High Speed Clock (HSE)
Crystal/Ceramic Resonator
Low Speed Clock (LSE)
Disable
BYPASS Clock Source
Crystal/Ceramic Resonator
Master Clock Output
Master Clock Output 2
Audio Clock Input (I2S_CKIN)
Configuration
Reset Configuration
User Constants
NVIC Settings
GPIO Settings
Parameter Settings
Configure the below parameters :
Search (Ctrl+F)
System Parameters
VDD voltage (V)
3.3 V
Instruction Cache
Enabled
Prefetch Buffer
Enabled


Pinout view
System view
RCC_OSC_IN
RCC_OSC_OUT
STM32F446RETx
LQFP64

# 시리얼 라인 설정

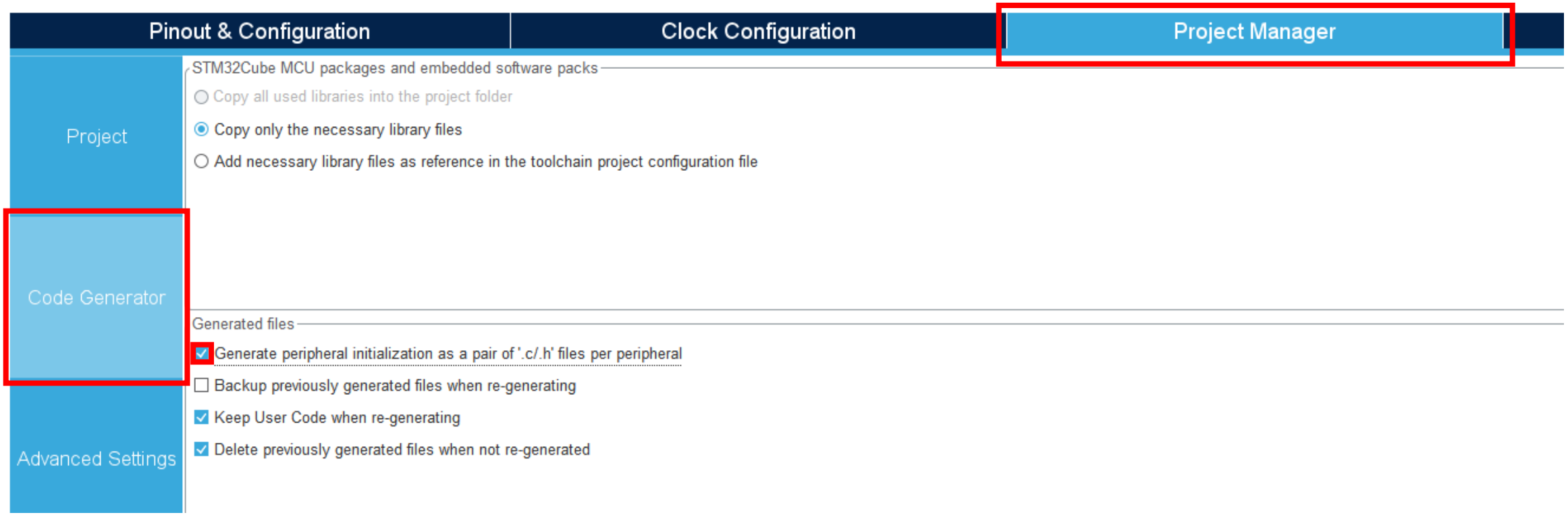

Pinout & Configuration
Clock Configuration
Project Manager
Tools
Software Packs
Pinout
Categories
A->Z
System Core
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG
Analog
Timers
Connectivity
Multimedia
Computing
Middleware
SYS Mode and Configuration
Mode
Debug
Serial Wire
Disable
Serial Wire
JTAG (4 pins)
JTAG (5 pins)
Trace Asynchronous Sw
Configuration
Warning: This peripheral has no parameters to be configured.
Pinout view
System view
SYS_JTCK-SWCLK
SYS_JTMS-SWDIO
RCC_OSC_IN
RCC_OSC_OUT
STM32F446RETx
LQFP64

# 클럭 설정

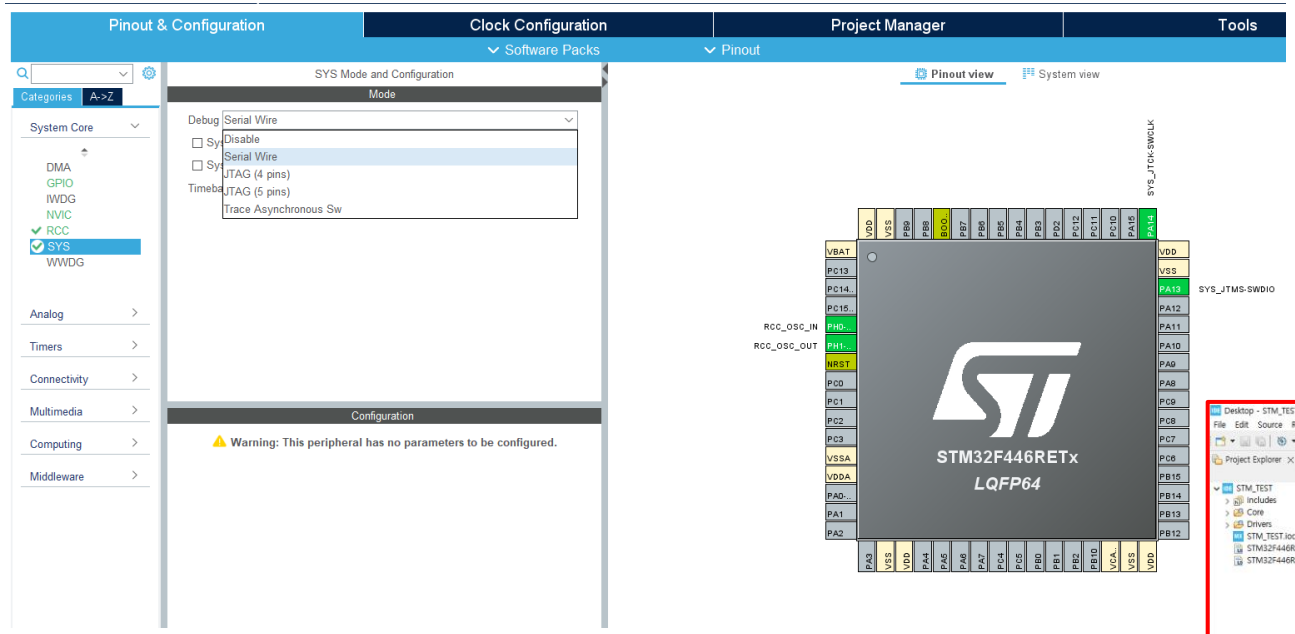



HCLK 변경하면
자동으로 바뀜

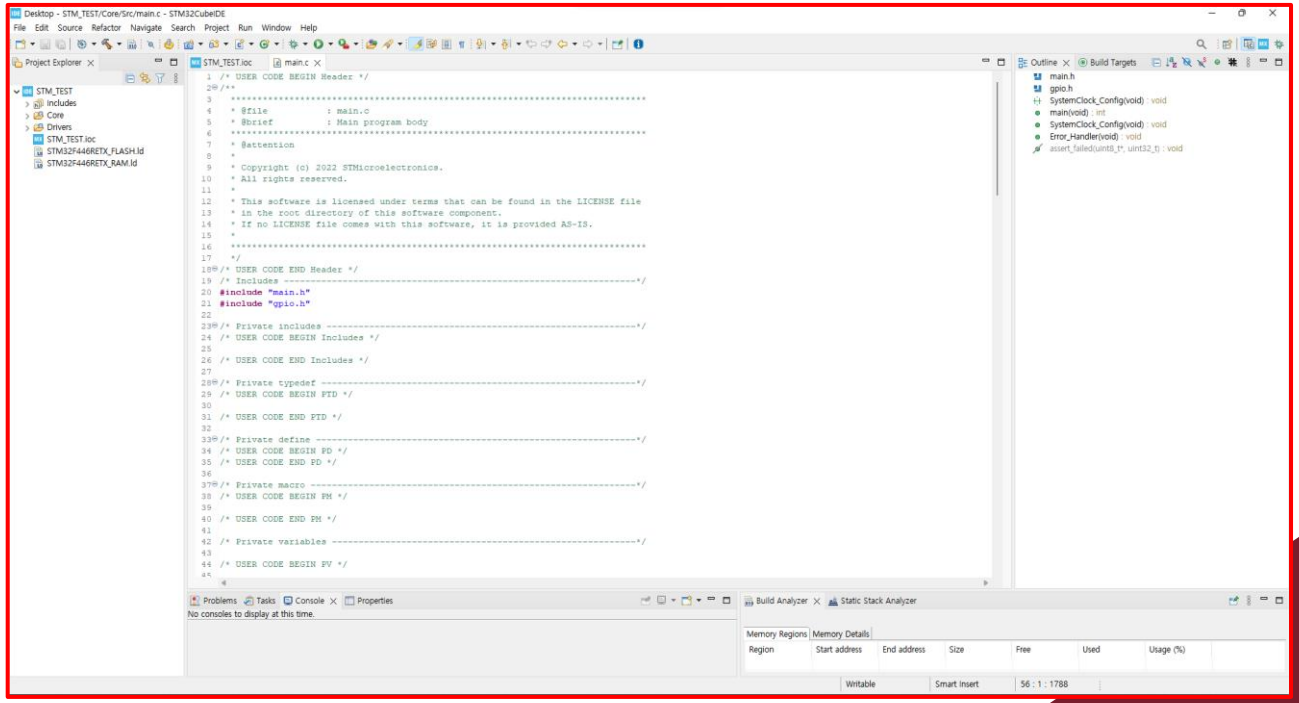
# 코드 생성 설정


Pinout & Configuration
Clock Configuration
Project Manager
Project
Code Generator
Advanced Settings
STM32Cube MCU packages and embedded software packs
Copy all used libraries into the project folder
Copy only the necessary library files
Add necessary library files as reference in the toolchain project configuration file
Generated files
Generate peripheral initialization as a pair of '.c/.h' files per peripheral
Backup previously generated files when re-generating
Keep User Code when re-generating
Delete previously generated files when not re-generated

# 코드 생성




Ctrl + S (저장)
→ 코드 자동 생성

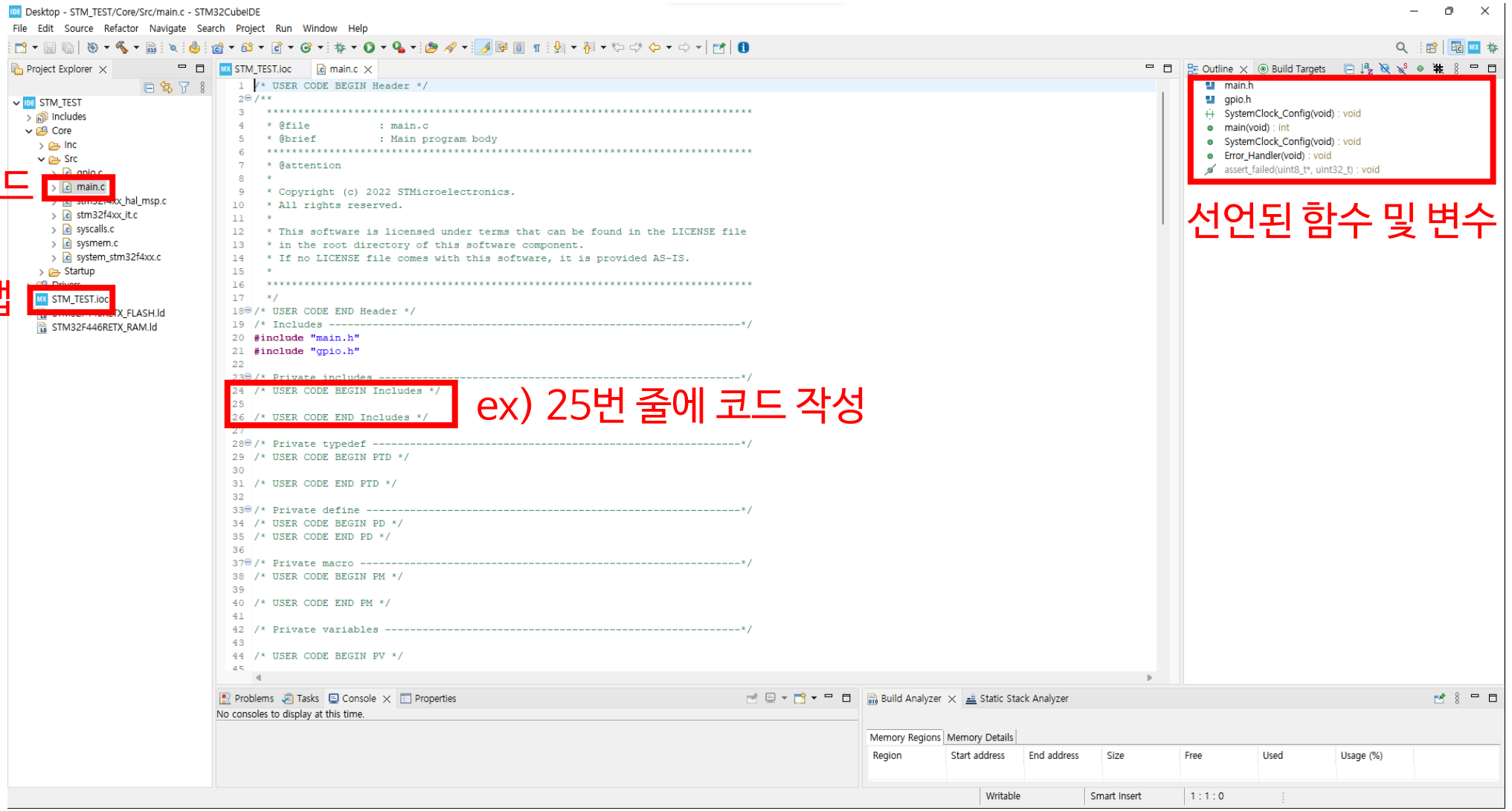



생성된 코드

# 코드 생성

생성된 코드

핀 맵




선언된 함수 및 변수

ex) 25번 줄에 코드 작성

코드는 자동 생성되므로 BEGIN 과 END 사이에 작성한 코드만 저장됨!

# 코드 생성

Includes : 라이브러리 추가
PD : 상수 선언
PV : 전역변수 선언

main 함수 내부
Init : 초기 설정 등의 함수 실행
2 : 타이머 시작 및 설정 관련 명령어 실행
3 : 무한 반복문 내부 코드

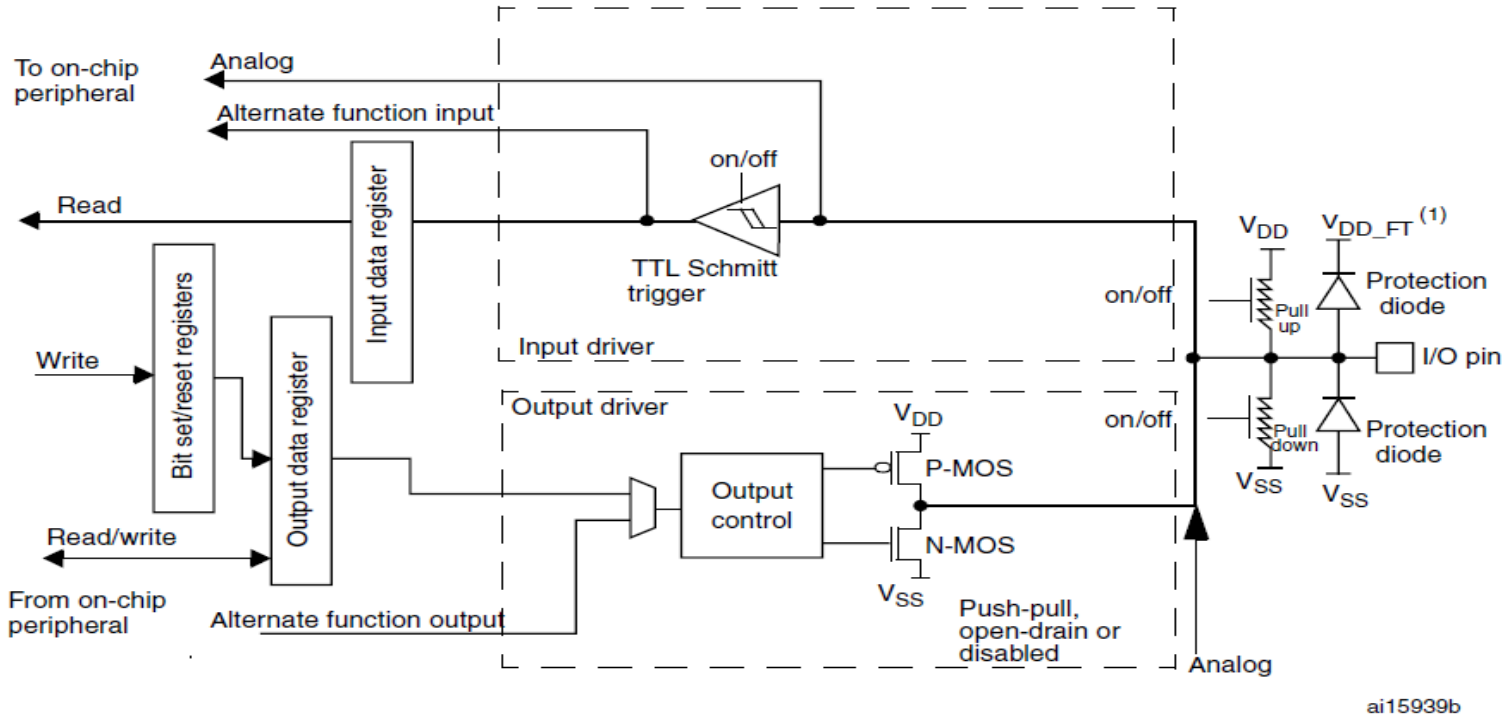
main 함수 외부
4 : 타이머 관련 함수, 사용자 정의 함수

# GPIO

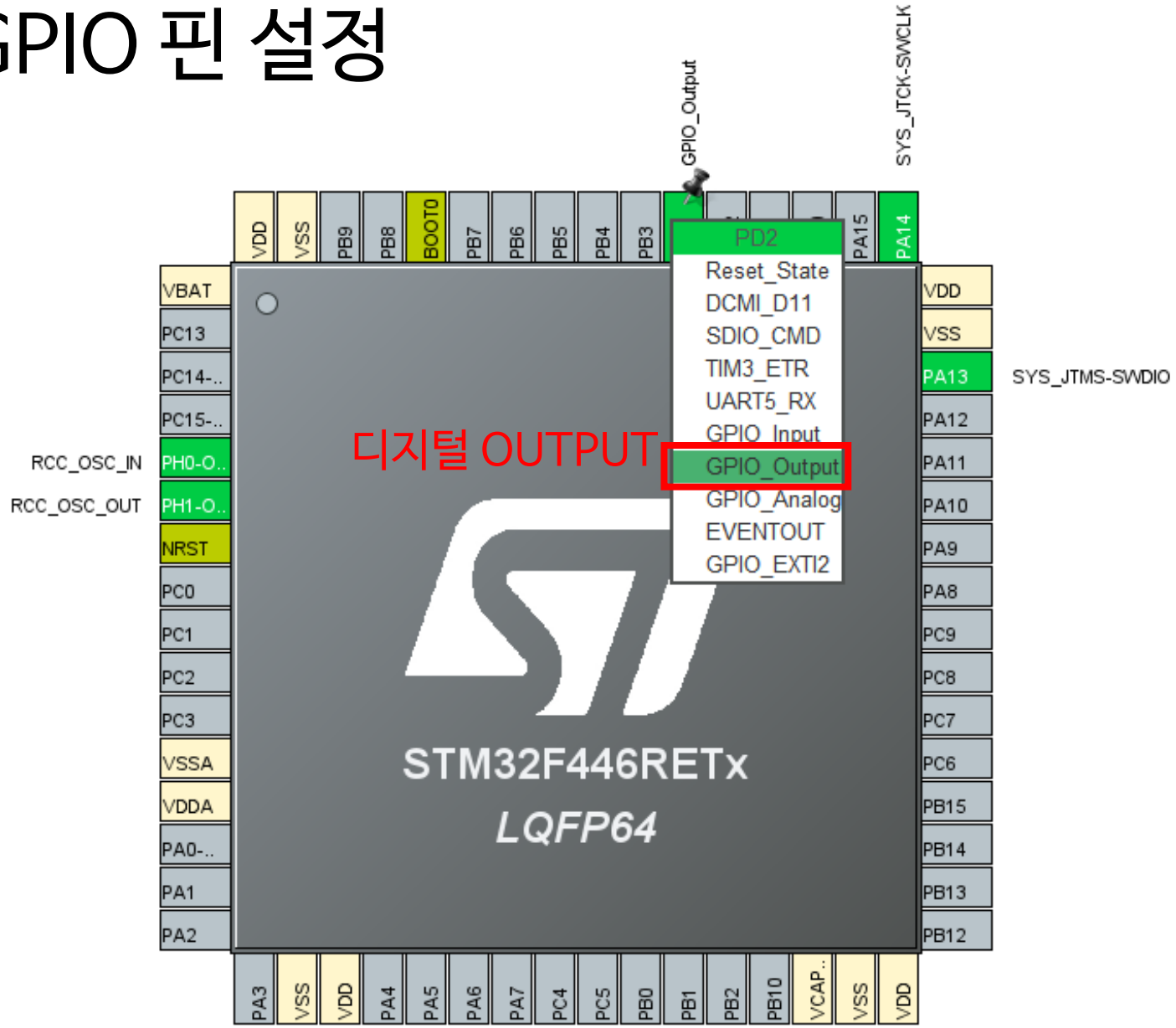
# GPIO

: GPIO란 General Purpose input/output의 약자로 **사용자가 그 포트에 대해 입력인지 출력인지 설정**할 수 있고 마음대로 변형하면서 제어할 수 있도록 제공해주고 범용적인 목적으로 입출력을 담당하는 포트이다.

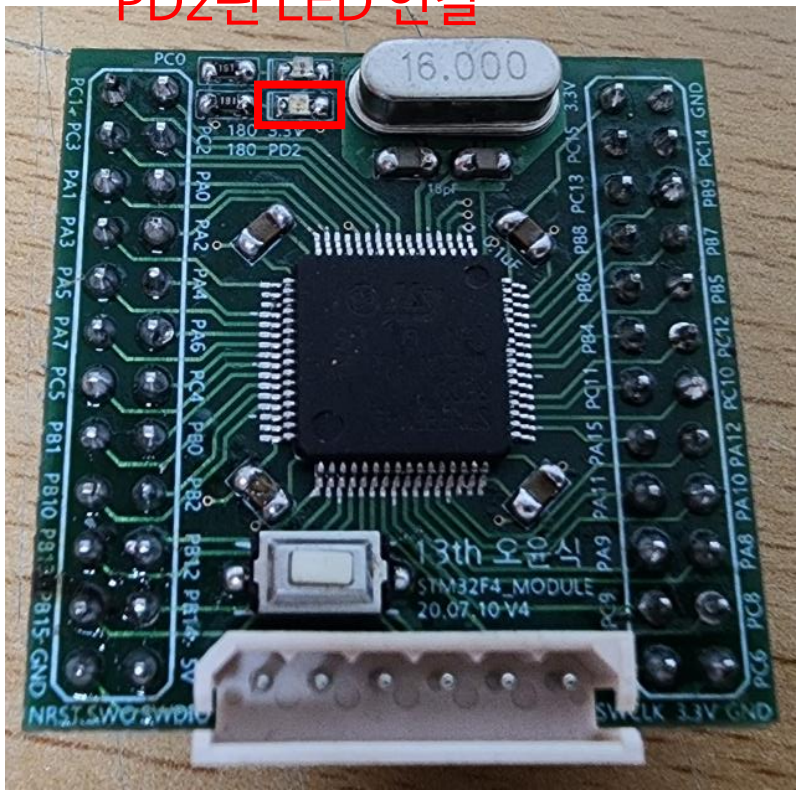
출력으로 사용시 0V(Low), 3.3V(High)를 출력
입력으로 사용시 0V 입력할 때 0(Low), 3.3V로 입력할 때 1(High)로 인식



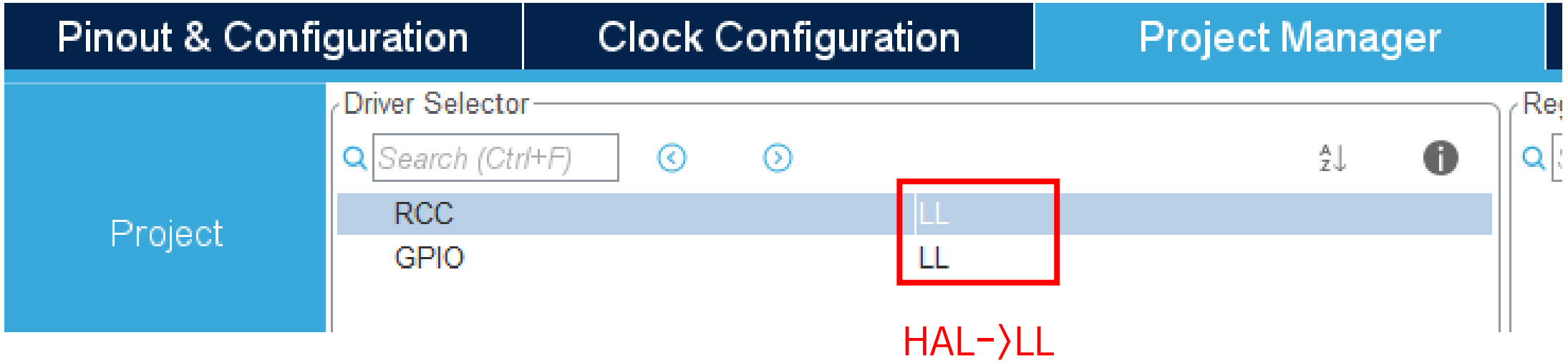
# GPIO 핀 설정




PD2핀 LED 연결



# GPIO 핀 설정


Pinout & Configuration
Clock Configuration
Project Manager
Project
Driver Selector
Search (Ctrl+F)
RCC
LL
GPIO
LL
HAL->LL

# GPIO 실습

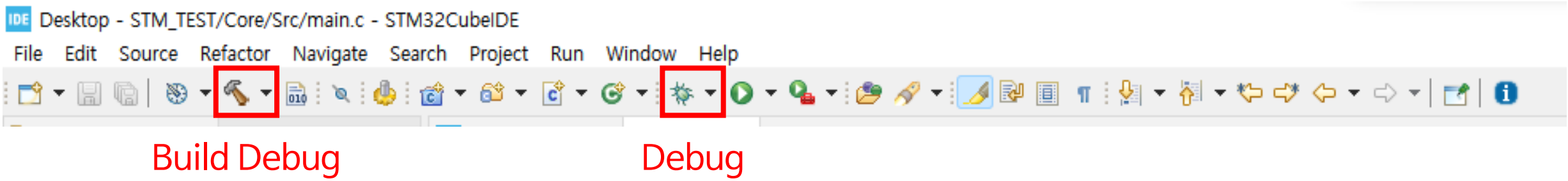
```
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
  LL_GPIO_SetOutputPin(GPIOD, LL_GPIO_PIN_2);
  LL_mDelay(50);
  LL_GPIO_ResetOutputPin(GPIOD, LL_GPIO_PIN_2);
  LL_mDelay(50);
}
/* USER CODE END 3 */
```

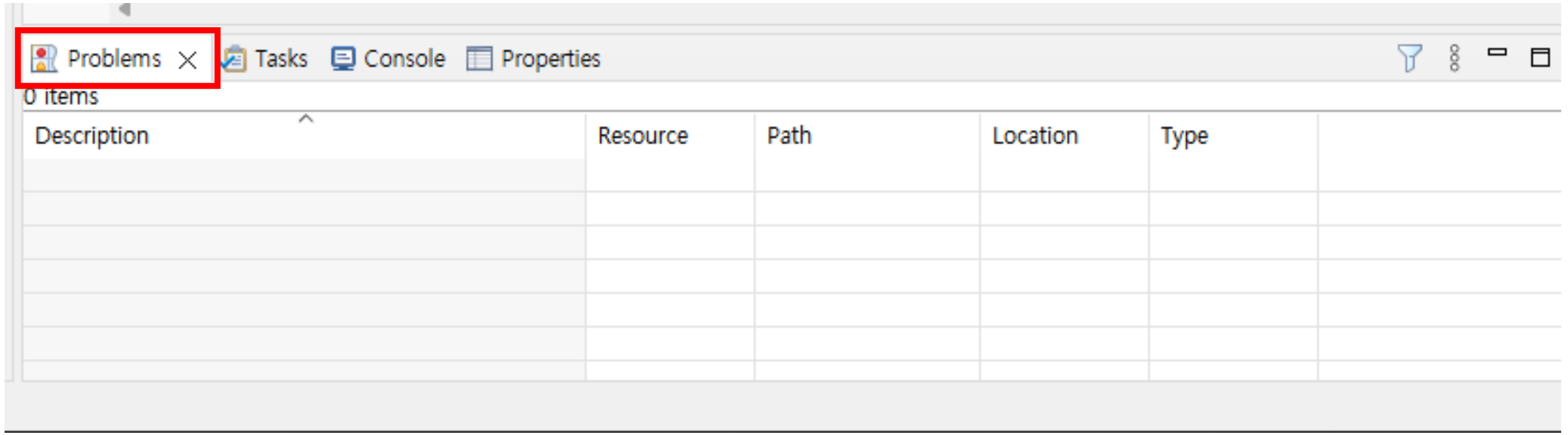
LED ON 3.3V OUTPUT

LED OFF 0V OUTPUT

# 실행


Desktop - STM_TEST/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help
Build Debug
Debug

오류 발생 시

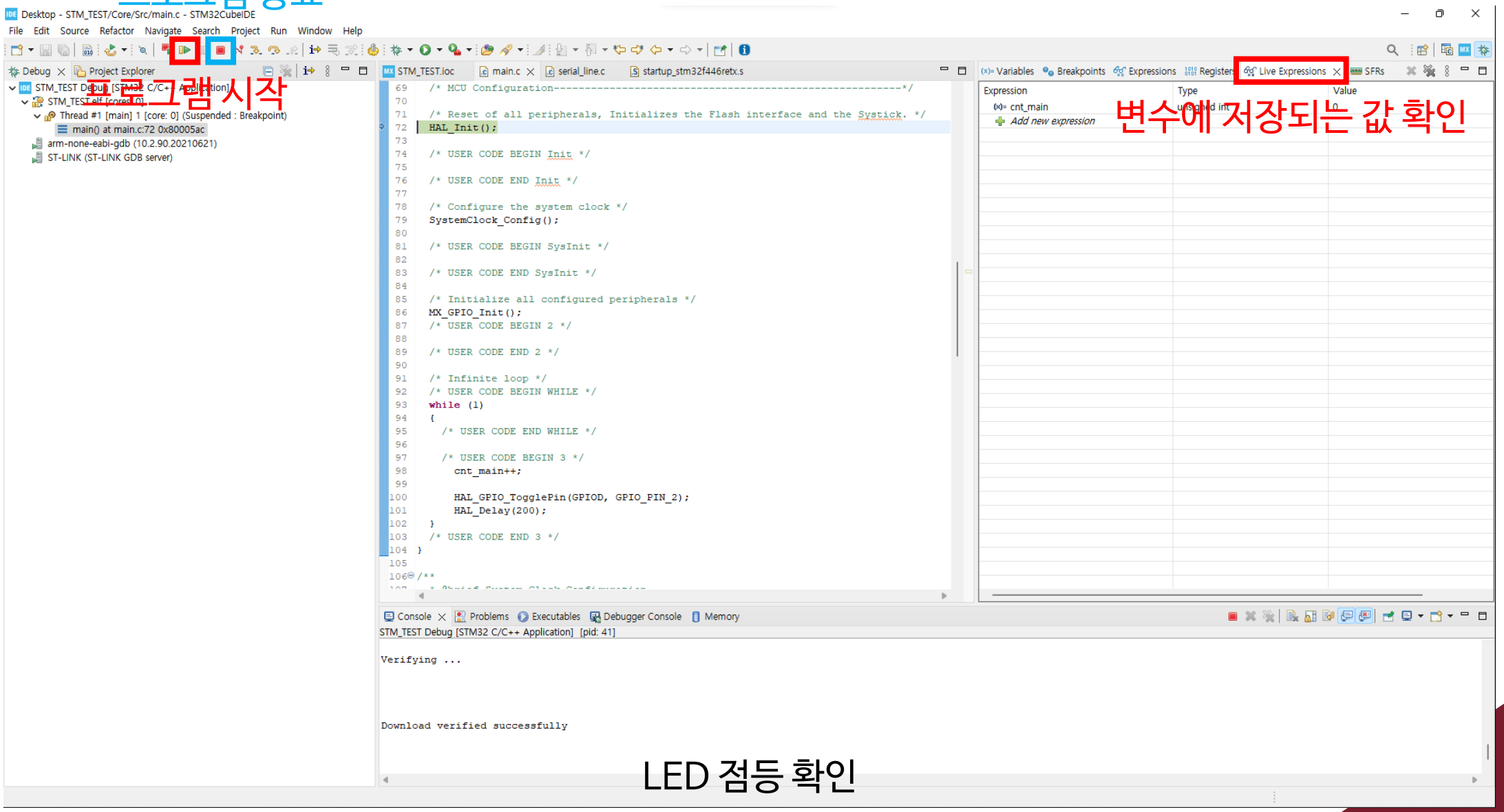

Problems
Tasks
Console
Properties
0 items
Description
Resource
Path
Location
Type

# 디버깅

프로그램 종료

프로그램 시작

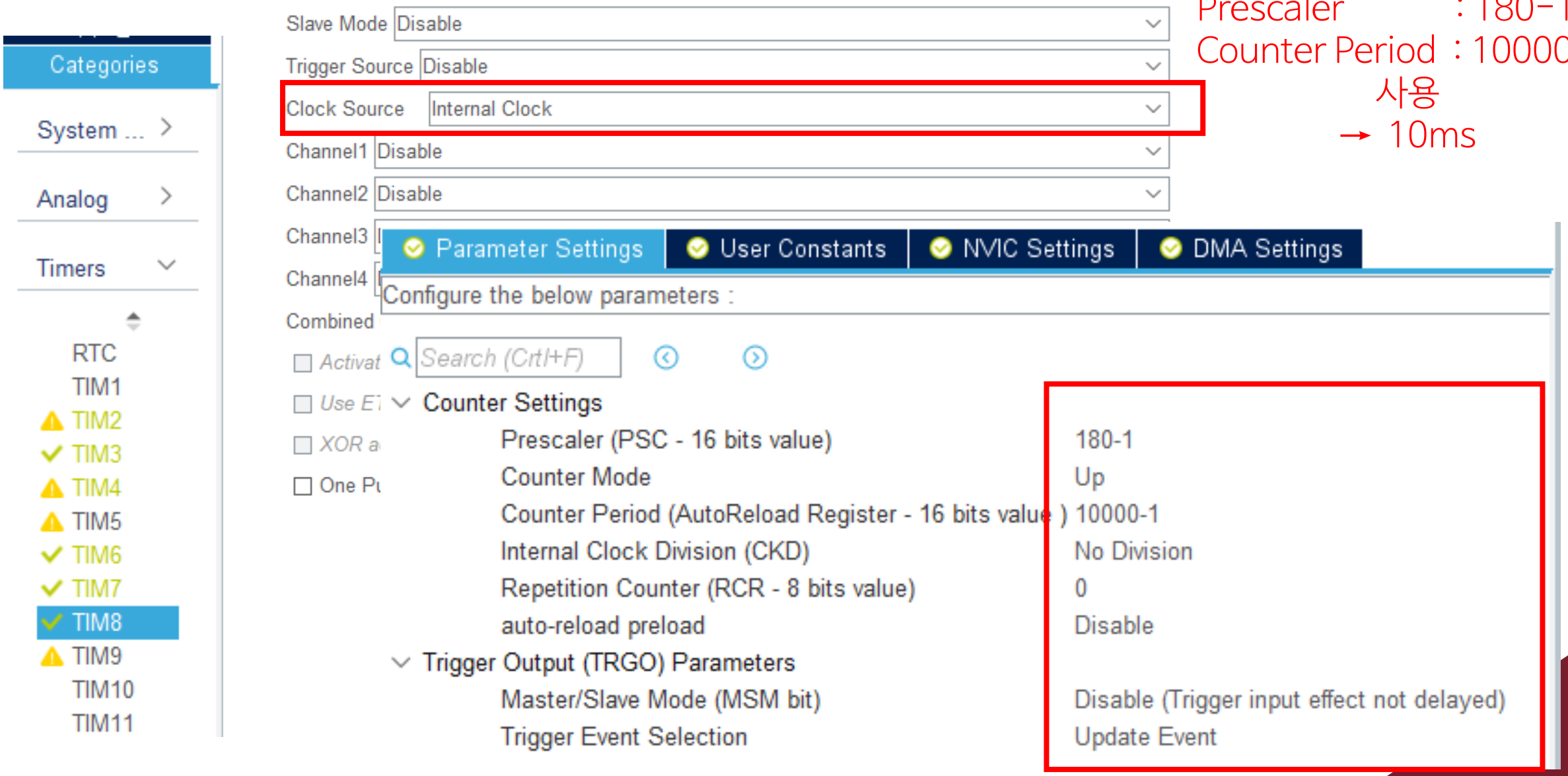
변수에 저장되는 값 확인




LED 점등 확인

# ADC

# ADC 설정




Prescaler : 180-1
Counter Period : 10000-1
사용
→ 10ms

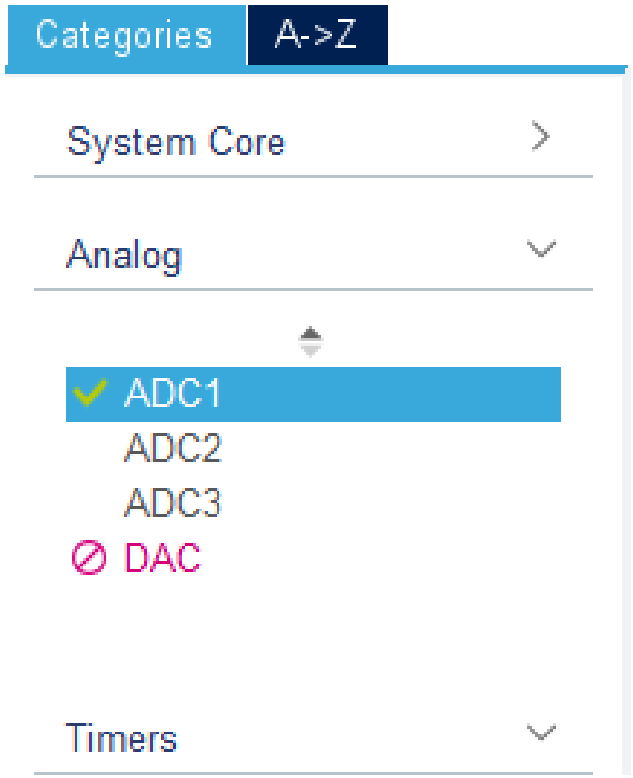
# ADC 설정

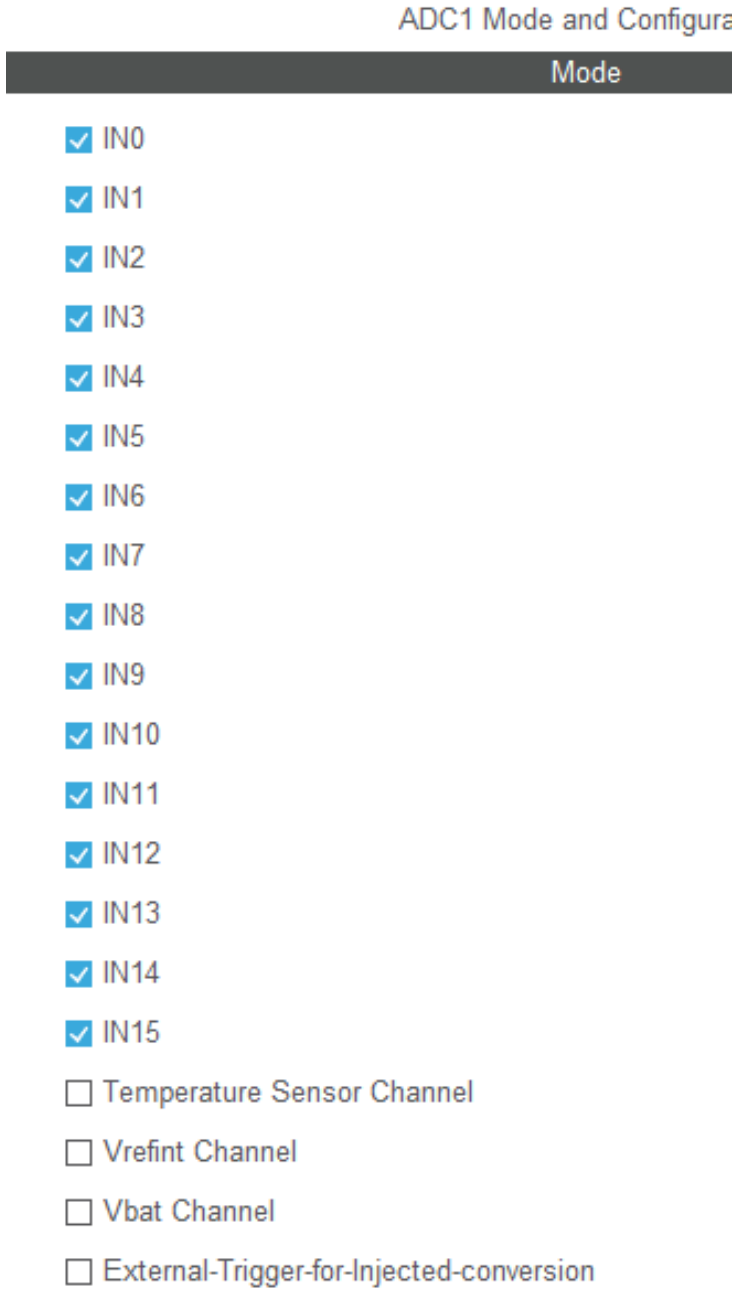
ROBIT ROBOT SPORT GAME TEAM

Parameter Settings | User Constants | NVIC Settings | DMA Settings

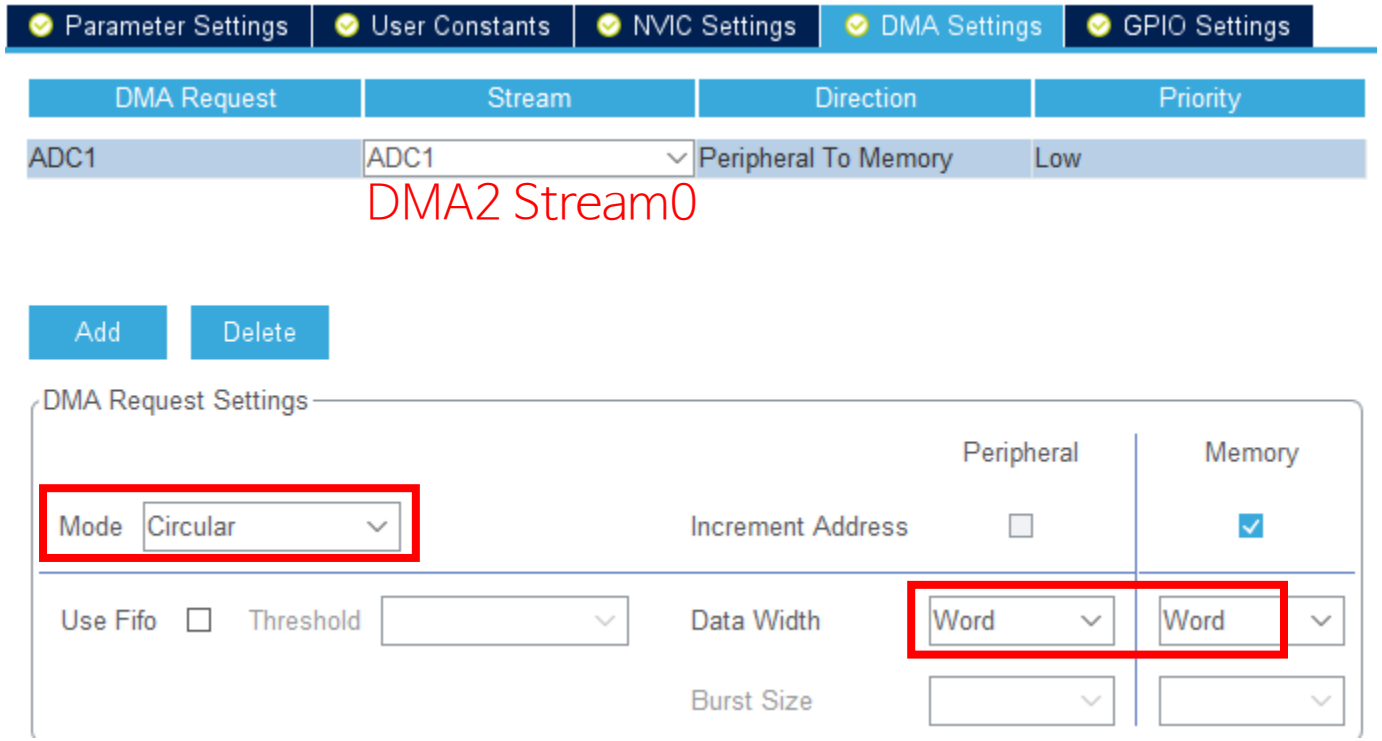
| NVIC Interrupt Table | Enabled | Preemption Priority | Sub Priority |
|---|---|---|---|
| TIM8 break interrupt and TIM12 global interrupt | ☐ | 0 | 0 |
| TIM8 update interrupt and TIM13 global interrupt | ☑ | 0 | 0 |
| TIM8 trigger and commutation interrupts and TIM14 global interrupt | ☐ | 0 | 0 |
| TIM8 capture compare interrupt | ☐ | 0 | 0 |

# ADC 설정

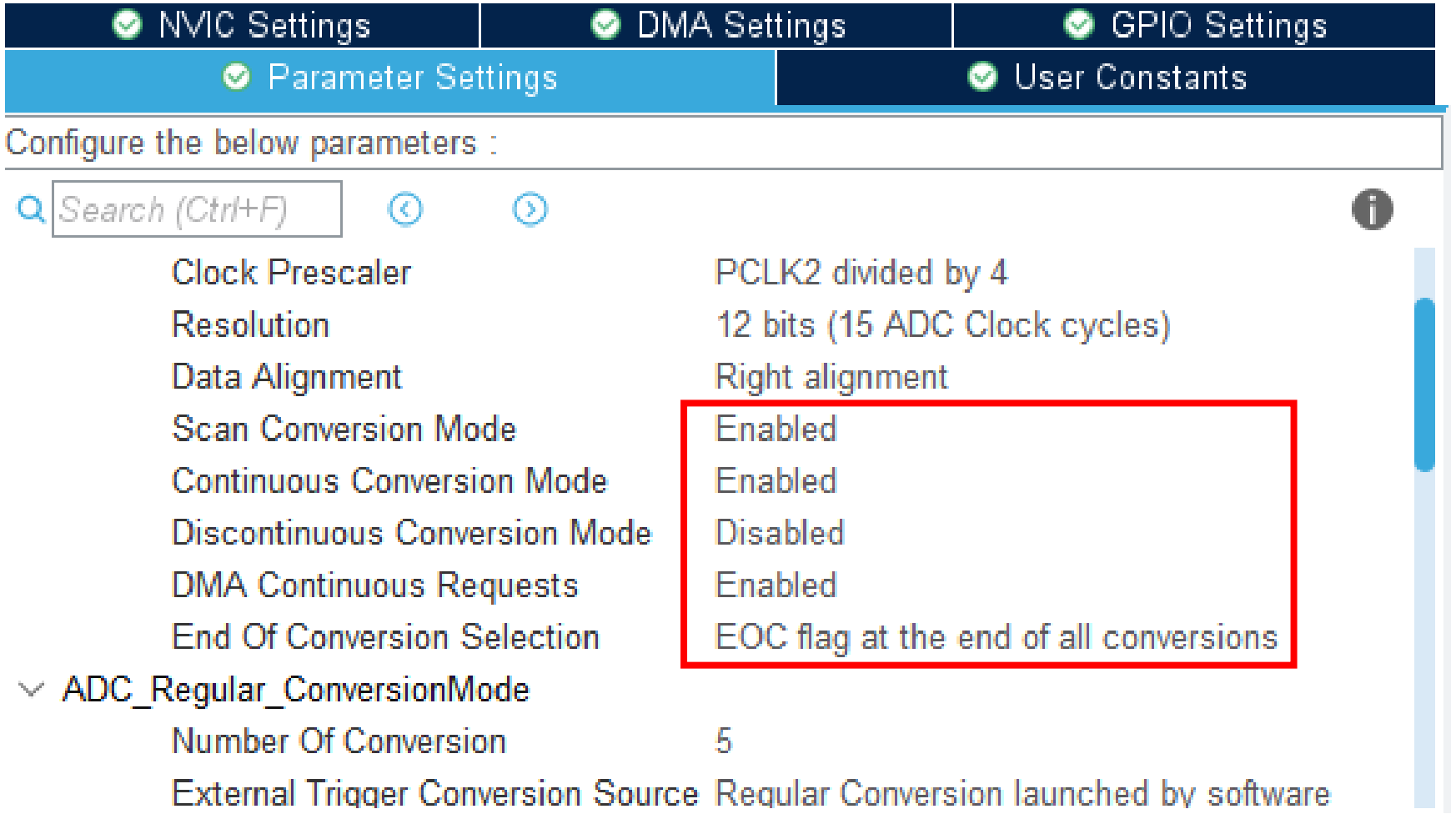

Categories
A->Z
System Core
Analog
ADC1
ADC2
ADC3
DAC
Timers


ADC1 Mode and Configura
Mode
IN0
IN1
IN2
IN3
IN4
IN5
IN6
IN7
IN8
IN9
IN10
IN11
IN12
IN13
IN14
IN15
Temperature Sensor Channel
Vrefint Channel
Vbat Channel
External-Trigger-for-Injected-conversion

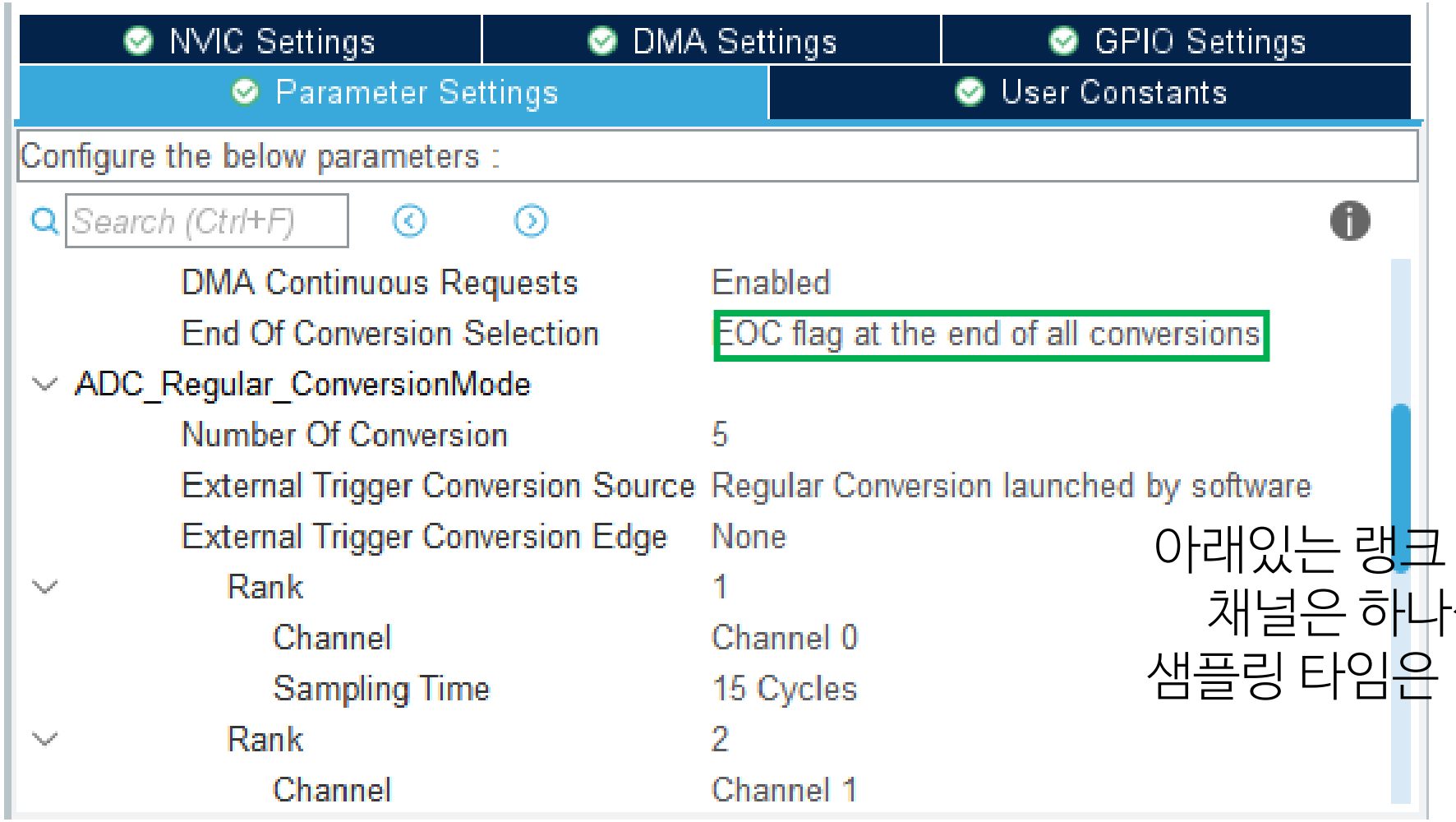
# ADC 설정


Parameter Settings
User Constants
NVIC Settings
DMA Settings
GPIO Settings
DMA Request
Stream
Direction
Priority
ADC1
ADC1
Peripheral To Memory
Low
DMA2 Stream0
Add
Delete
DMA Request Settings
Peripheral
Memory
Mode
Circular
Increment Address
Use Fifo
Threshold
Data Width
Word
Word
Burst Size

# ADC 설정

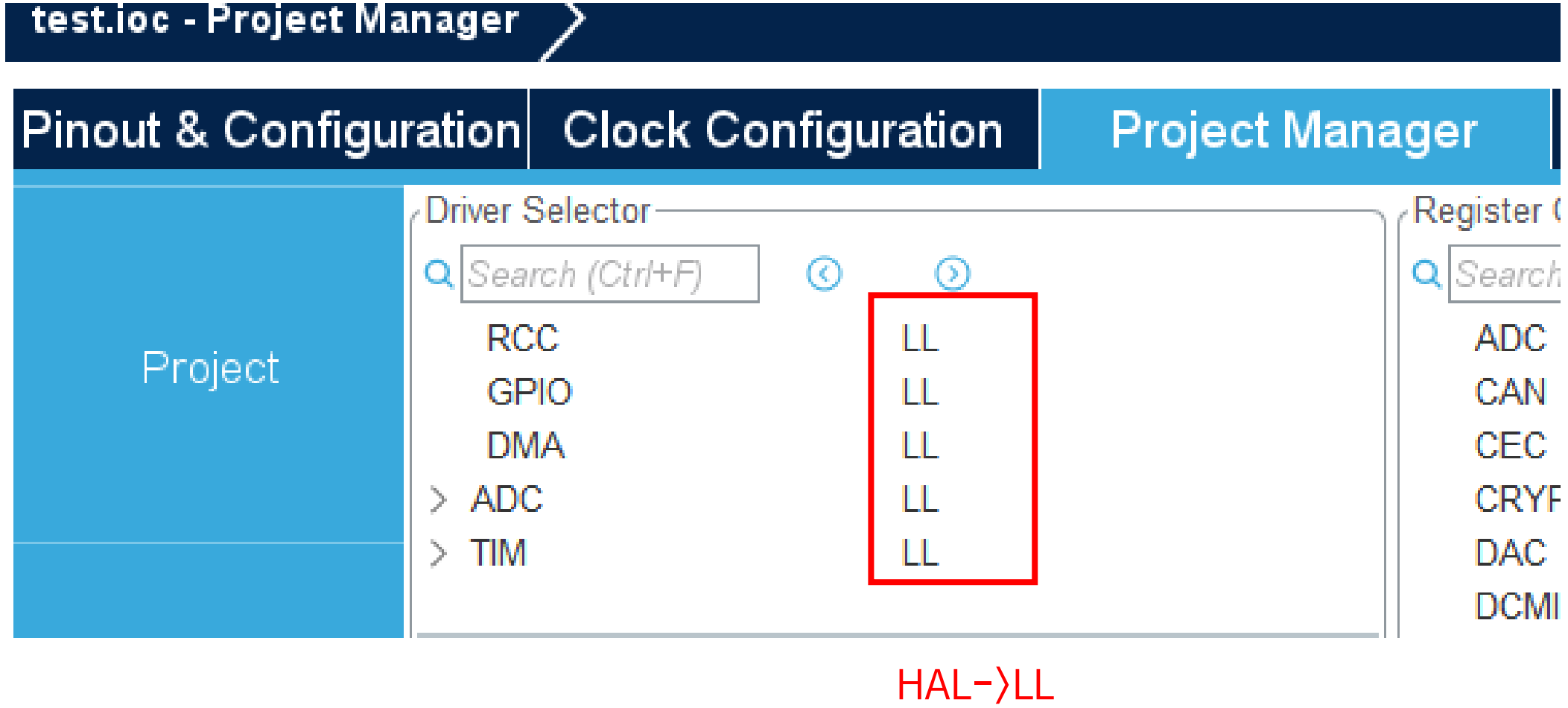

NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
Configure the below parameters :
Search (Ctrl+F)
Clock Prescaler
PCLK2 divided by 4
Resolution
12 bits (15 ADC Clock cycles)
Data Alignment
Right alignment
Scan Conversion Mode
Enabled
Continuous Conversion Mode
Enabled
Discontinuous Conversion Mode
Disabled
DMA Continuous Requests
Enabled
End Of Conversion Selection
EOC flag at the end of all conversions
ADC_Regular_ConversionMode
Number Of Conversion
5
External Trigger Conversion Source
Regular Conversion launched by software

# ADC 설정




아래있는 랭크 전부 똑같이
채널은 하나씩 늘리고
샘플링 타임은 15 사이클로

# GPIO 핀 설정




HAL->LL

# ADC 실습

```
/* USER CODE BEGIN PV */
uint32_t adcVal[5] = {0, };
int adc[5] = {0, };
/* USER CODE END PV */
  /* USER CODE BEGIN 2 */

  //HAL_TIM_Base_Start_IT(&htim8);
  LL_TIM_EnableIT_UPDATE(TIM8);
  LL_TIM_EnableCounter(TIM8);

  //HAL_ADC_Start_DMA(&hadc1, adc1_buffer, 1);
  LL_DMA_ConfigAddresses(DMA2, LL_DMA_STREAM_0,
          LL_ADC_DMA_GetRegAddr(ADC1, LL_ADC_DMA_REG_REGULAR_DATA),
          (uint32_t)adcVal,
          LL_DMA_DIRECTION_PERIPH_TO_MEMORY);
  LL_DMA_SetDataLength(DMA2, LL_DMA_STREAM_0, 5);
  LL_DMA_EnableIT_TC(DMA2, LL_DMA_STREAM_0);
  LL_DMA_EnableStream(DMA2, LL_DMA_STREAM_0);
  LL_ADC_Enable(ADC1);
  LL_ADC_REG_StartConversionSWStart(ADC1);

  /* USER CODE END 2 */
```

사용하는 ADC 개수만큼

# ADC 실습

ROBIT ROBOT SPORT GAME TEAM

## stm32f4xx_it

```
/* Private variables ---------------
/* USER CODE BEGIN PV */
extern uint32_t adcVal[5];
extern int adc[5];


void TIM8_UP_TIM13_IRQHandler(void)
{
  /* USER CODE BEGIN TIM8_UP_TIM13_IRQn 0 */
    if(LL_TIM_IsActiveFlag_UPDATE(TIM8))
    {
        for(int i = 0; i < 5; i++)
        {
            adc[i] = adcVal[i];
        }

        LL_TIM_ClearFlag_UPDATE(TIM8);
    }
  /* USER CODE END TIM8_UP_TIM13_IRQn 0 */

  /* USER CODE BEGIN TIM8_UP_TIM13_IRQn 1 */

  /* USER CODE END TIM8_UP_TIM13_IRQn 1 */
}
```

# PWM

# TIMER



### 3.21.1 Advanced-control timers (TIM1, TIM8)

The advanced-control timers (TIM1, TIM8) can be seen as three-phase PWM generators multiplexed on 6 channels. They have complementary PWM outputs with programmable inserted dead times. They can also be considered as complete general-purpose timers. Their 4 independent channels can be used for:

- Input capture
- [illegible]
- PWM generation (edge- or center-aligned modes)
- One-pulse mode output

If configured as standard 16-bit timers, they have the same features as the general-purpose TIMx timers. If configured as 16-bit PWM generators, they have full modulation capability (0-100%).

The advanced-control timer can work together with the TIMx timers via the Timer Link feature for synchronization or event chaining.

TIM1 and TIM8 support independent DMA request generation.

### 3.21.2 General-purpose timers (TIMx)

There are ten synchronized general-purpose timers embedded in the STM32F446xC/E devices (see *Table 6* for differences).

- **TIM2, TIM3, TIM4, TIM5**

  The STM32F446xC/E include four full-featured general-purpose timers: TIM2, TIM5, TIM3, and TIM4.The TIM2 and TIM5 timers are based on a 32-bit auto-reload up/downcounter and a 16-bit prescaler. The TIM3 and TIM4 timers are based on a 16-bit auto-reload up/downcounter and a 16-bit prescaler. They all feature 4 independent channels for input capture/output compare, PWM or one-pulse mode output. This gives up to 16 input capture/output compare/PWMs on the largest packages.

  The TIM2, TIM3, TIM4, TIM5 general-purpose timers can work together, or with the other general-purpose timers and the advanced-control timers TIM1 and TIM8 via the Timer Link feature for synchronization or event chaining.

  Any of these general-purpose timers can be used to generate PWM outputs.

  TIM2, TIM3, TIM4 [illegible] generation. They are capable of handling quadrature (incremental) encoder signals and the digital outputs from one to four [illegible].

- **TIM9, TIM10, TIM11, TIM12, TIM13, and TIM14**

  These timers are based on a 16-bit auto-reload upcounter and a 16-bit prescaler. TIM10, TIM11, TIM13, and TIM14 feature one independent channel, whereas TIM9 and TIM12 have two independent channels for input capture/output compare, PWM or one-pulse mode output. They can be synchronized with the TIM2, TIM3, TIM4, TIM5 full-featured general-purpose timers. They can also be used as simple time bases.

### 3.21.3 Basic timers TIM6 and TIM7

These timers are mainly used for DAC trigger and waveform generation. They can also be used as a generic 16-bit time base.

TIM6 and TIM7 support independent DMA request generation.

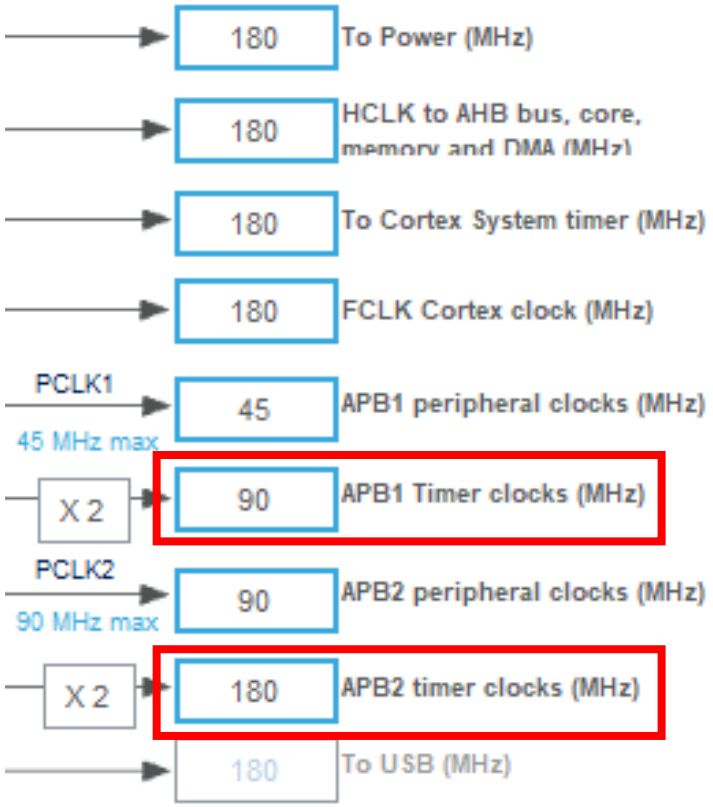


TIM1, TIM8
: PWM에 사용 가능

TIM2, TIM3, TIM4, TIM5
: Encoder에 사용 가능

TIM6, TIM7
: 타이머 인터럽트를 통한 제어주기 만들 때 사용 가능

# Timer



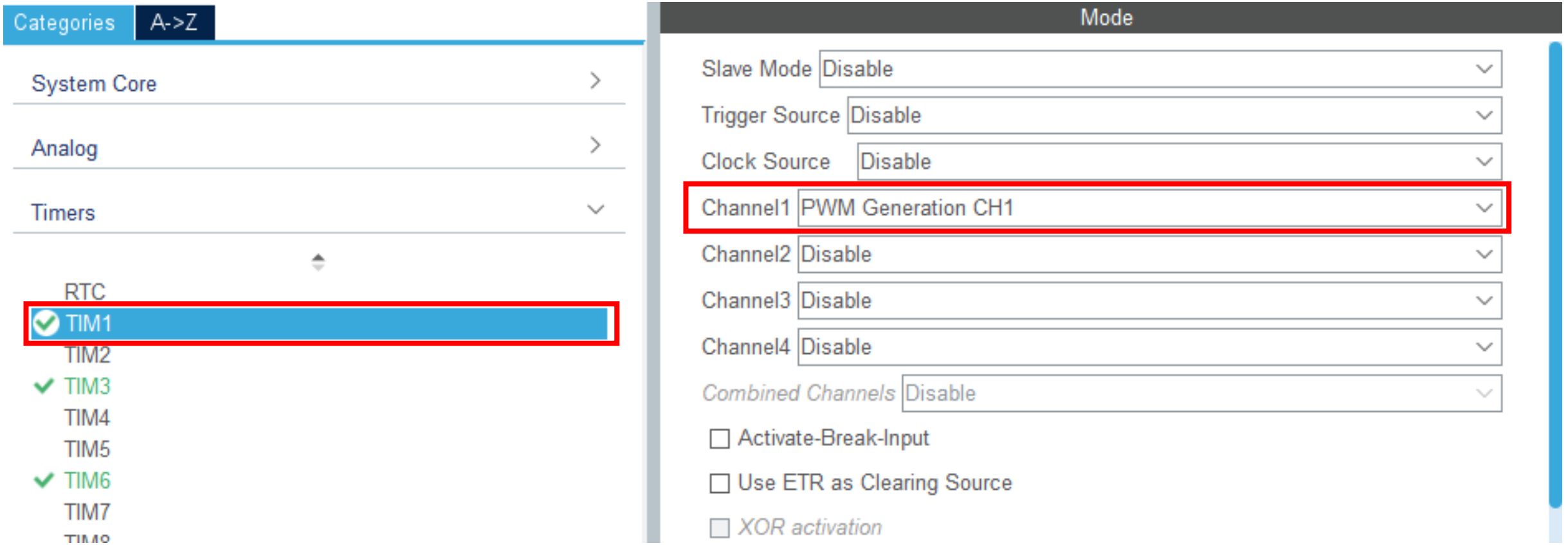

TIM2
TIM3
TIM4
TIM5
TIM6
TIM7
TIM12
TIM13
TIM14

APB1는 90MHz

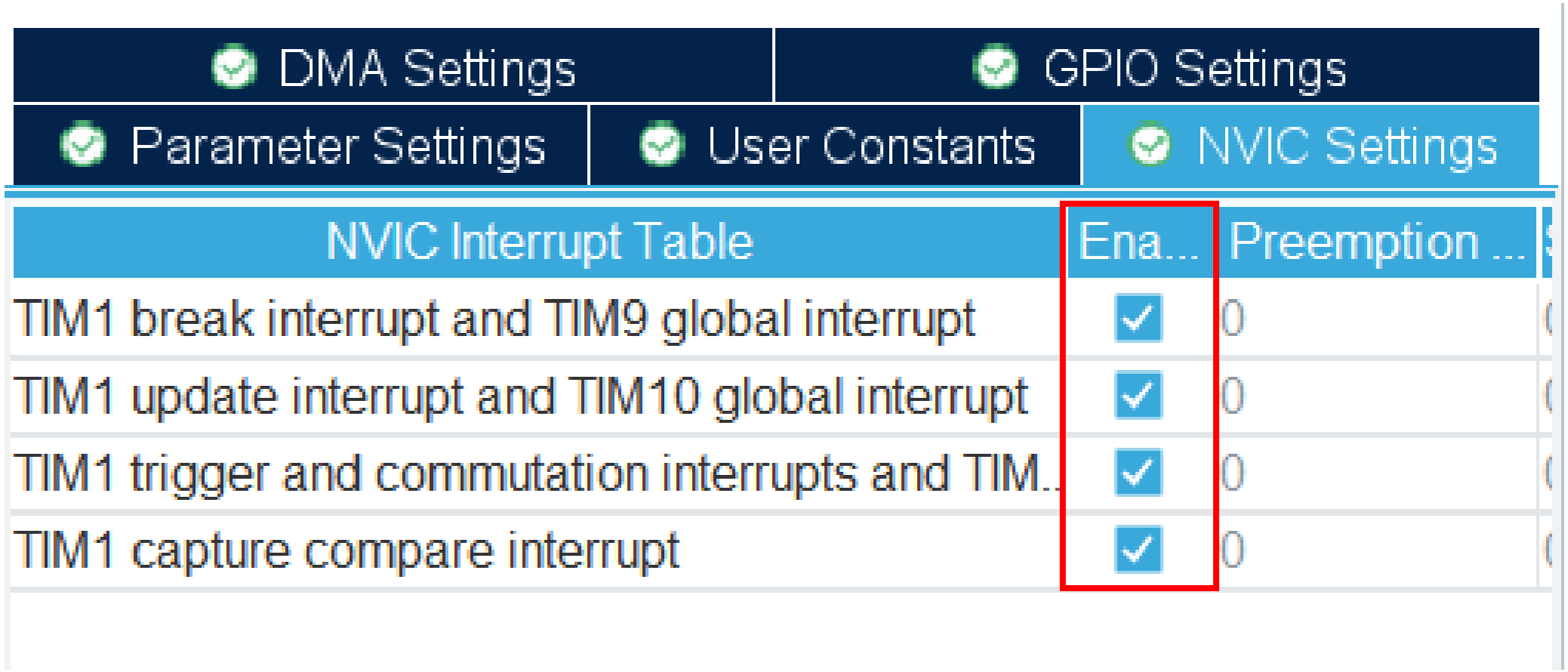
TIM1
TIM8
TIM9
TIM10
TIM11

APB2는 180MHz

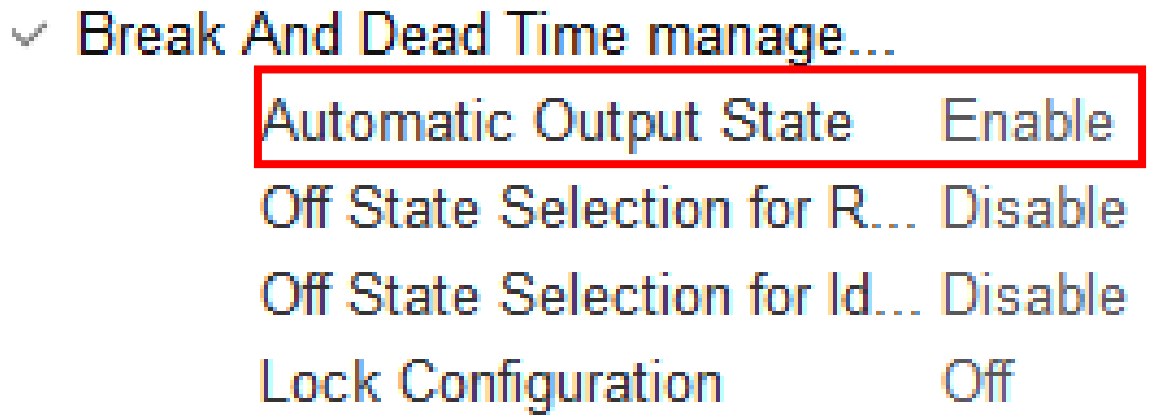
# PWM


Categories
A->Z
System Core
Analog
Timers
RTC
TIM1
TIM2
TIM3
TIM4
TIM5
TIM6
TIM7
Mode
Slave Mode
Disable
Trigger Source
Disable
Clock Source
Disable
Channel1
PWM Generation CH1
Channel2
Disable
Channel3
Disable
Channel4
Disable
Combined Channels
Disable
Activate-Break-Input
Use ETR as Clearing Source
XOR activation

# PWM


DMA Settings
GPIO Settings
Parameter Settings
User Constants
NVIC Settings
NVIC Interrupt Table
Ena...
Preemption ...
TIM1 break interrupt and TIM9 global interrupt
0
TIM1 update interrupt and TIM10 global interrupt
0
TIM1 trigger and commutation interrupts and TIM...
0
TIM1 capture compare interrupt
0

# PWM


Break And Dead Time manage...
Automatic Output State Enable
Off State Selection for R... Disable
Off State Selection for Id... Disable
Lock Configuration Off

# PWM

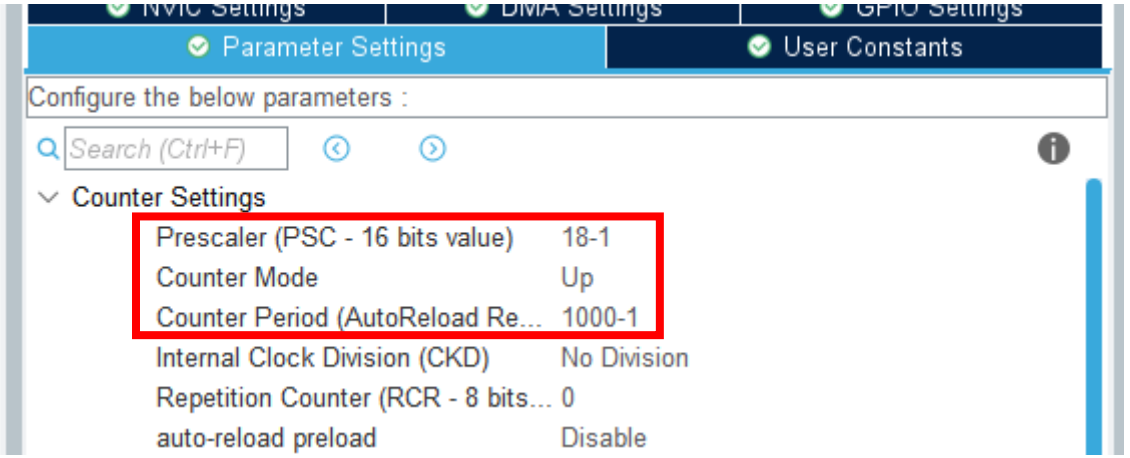
180MHz(180000000Hz)를 10000Hz로 만드려면?

180000000 / 18 / 1000 =10000

▲ 모터드라이버 Max PWM Frequency의 약 절반 정도

10000Hz이면
1000(ms) / 10000(Hz) = 0.1(ms)
0.1ms 주기

```
//pwm
LL_TIM_CC_EnableChannel(TIM1, LL_TIM_CHANNEL_CH1);
LL_TIM_EnableCounter(TIM1);
```



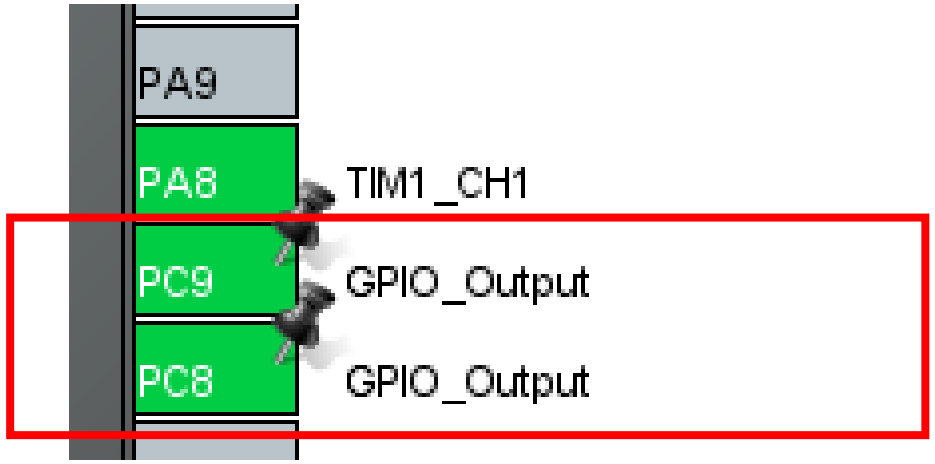

```
void TIM6_DAC_IRQHandler(void)
{
  /* USER CODE BEGIN TIM6_DAC_IRQn 0 */
    if(LL_TIM_IsActiveFlag_UPDATE(TIM6))
    {
        LL_GPIO_SetOutputPin(GPIOC, LL_GPIO_PIN_8);
        LL_TIM_OC_SetCompareCH1(TIM1, duty);

        LL_TIM_ClearFlag_UPDATE(TIM6);
    }
  /* USER CODE END TIM6_DAC_IRQn 0 */
```

Duty rate 설정하기

Counter period가 1000 일 때
TIM1->CCRn = 0~1000 범위
(0이면 duty 0%, 1000이면 duty 100%)

# PWM




DIR

```
void TIM6_DAC_IRQHandler(void)
{
  /* USER CODE BEGIN TIM6_DAC_IRQn 0 */
    if(LL_TIM_IsActiveFlag_UPDATE(TIM6))
    {
        LL_GPIO_SetOutputPin(GPIOC, LL_GPIO_PIN_8);
        LL_TIM_OC_SetCompareCH1(TIM1, duty);

        LL_TIM_ClearFlag_UPDATE(TIM6);
    }
  /* USER CODE END TIM6_DAC_IRQn 0 */
```

Set, Reset에 따라 모터회전 방향바뀜

# TIMER

# TIMER



### 3.21.1 Advanced-control timers (TIM1, TIM8)

The advanced-control timers (TIM1, TIM8) can be seen as three-phase PWM generators multiplexed on 6 channels. They have complementary PWM outputs with programmable inserted dead times. They can also be considered as complete general-purpose timers. Their 4 independent channels can be used for:

- Input capture
- [illegible]
- PWM generation (edge- or center-aligned modes)
- One-pulse mode output

If configured as standard 16-bit timers, they have the same features as the general-purpose TIMx timers. If configured as 16-bit PWM generators, they have full modulation capability (0-100%).

The advanced-control timer can work together with the TIMx timers via the Timer Link feature for synchronization or event chaining.

TIM1 and TIM8 support independent DMA request generation.

### 3.21.2 General-purpose timers (TIMx)

There are ten synchronized general-purpose timers embedded in the STM32F446xC/E devices (see *Table 6* for differences).

- **TIM2, TIM3, TIM4, TIM5**

  The STM32F446xC/E include four full-featured general-purpose timers: TIM2, TIM5, TIM3, and TIM4.The TIM2 and TIM5 timers are based on a 32-bit auto-reload up/downcounter and a 16-bit prescaler. The TIM3 and TIM4 timers are based on a 16-bit auto-reload up/downcounter and a 16-bit prescaler. They all feature 4 independent channels for input capture/output compare, PWM or one-pulse mode output. This gives up to 16 input capture/output compare/PWMs on the largest packages.

  The TIM2, TIM3, TIM4, TIM5 general-purpose timers can work together, or with the other general-purpose timers and the advanced-control timers TIM1 and TIM8 via the Timer Link feature for synchronization or event chaining.

  Any of these general-purpose timers can be used to generate PWM outputs.

  TIM2, TIM3, TIM4 [illegible] generation. They are capable of handling quadrature (incremental) encoder signals and the digital outputs from one to four [illegible].

- **TIM9, TIM10, TIM11, TIM12, TIM13, and TIM14**

  These timers are based on a 16-bit auto-reload upcounter and a 16-bit prescaler. TIM10, TIM11, TIM13, and TIM14 feature one independent channel, whereas TIM9 and TIM12 have two independent channels for input capture/output compare, PWM or one-pulse mode output. They can be synchronized with the TIM2, TIM3, TIM4, TIM5 full-featured general-purpose timers. They can also be used as simple time bases.

### 3.21.3 Basic timers TIM6 and TIM7

These timers are mainly used for DAC trigger and waveform generation. They can also be used as a generic 16-bit time base.

TIM6 and TIM7 support independent DMA request generation.

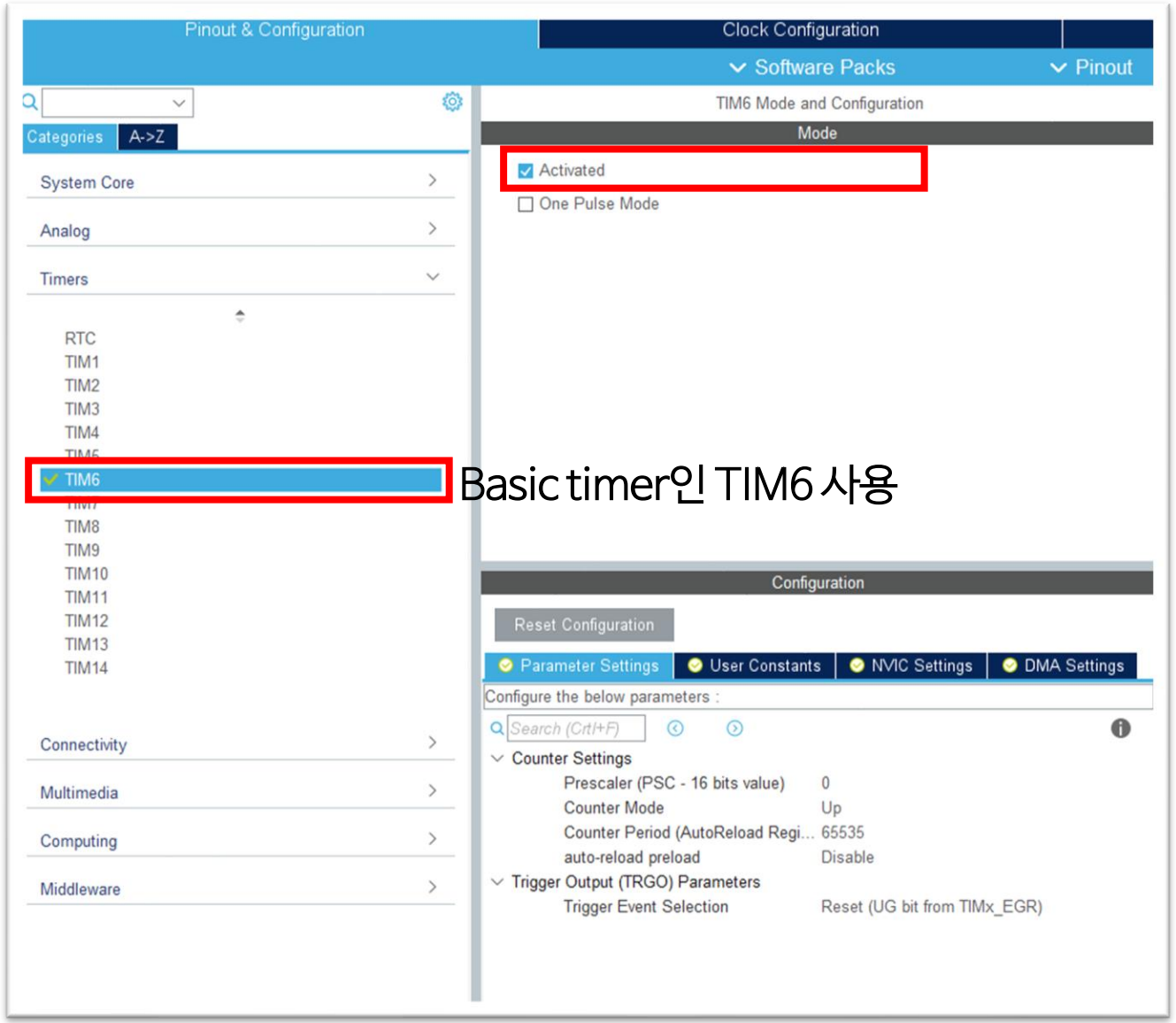


TIM1, TIM8
: PWM에 사용 가능

TIM2, TIM3, TIM4, TIM5
: Encoder에 사용 가능

TIM6, TIM7
: 타이머 인터럽트를 통한 제어주기 만들 때 사용 가능

# Timer




Basic timer인 TIM6 사용

# Timer

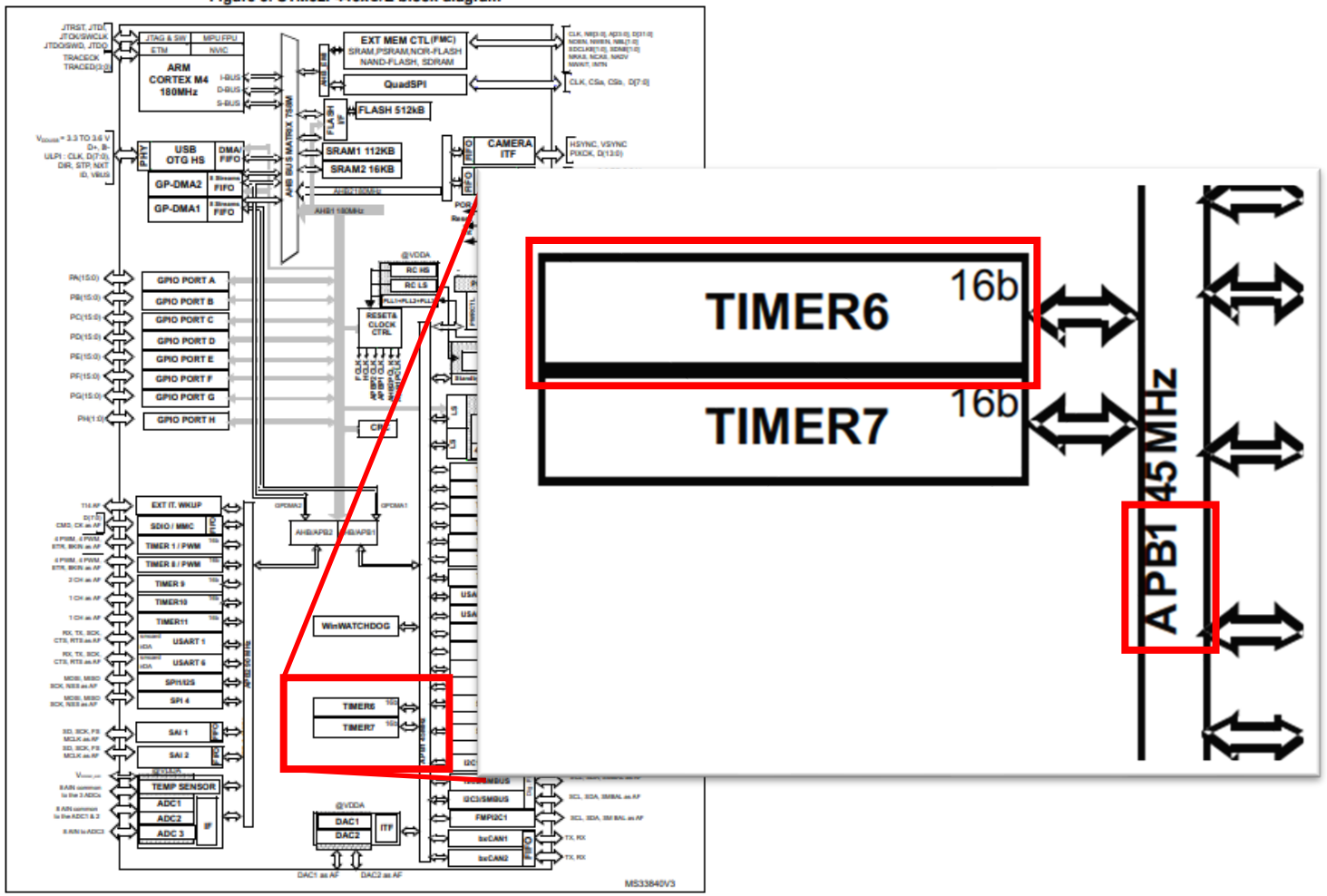


Source ÷ Prescaler ÷ Counter = 원하는 Hz

# Timer



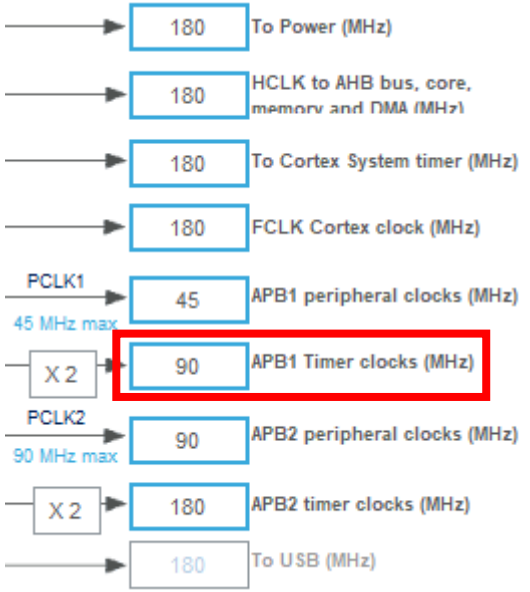
Figure 3. STM32F446xC/E block diagram







TIM6은 APB1 사용



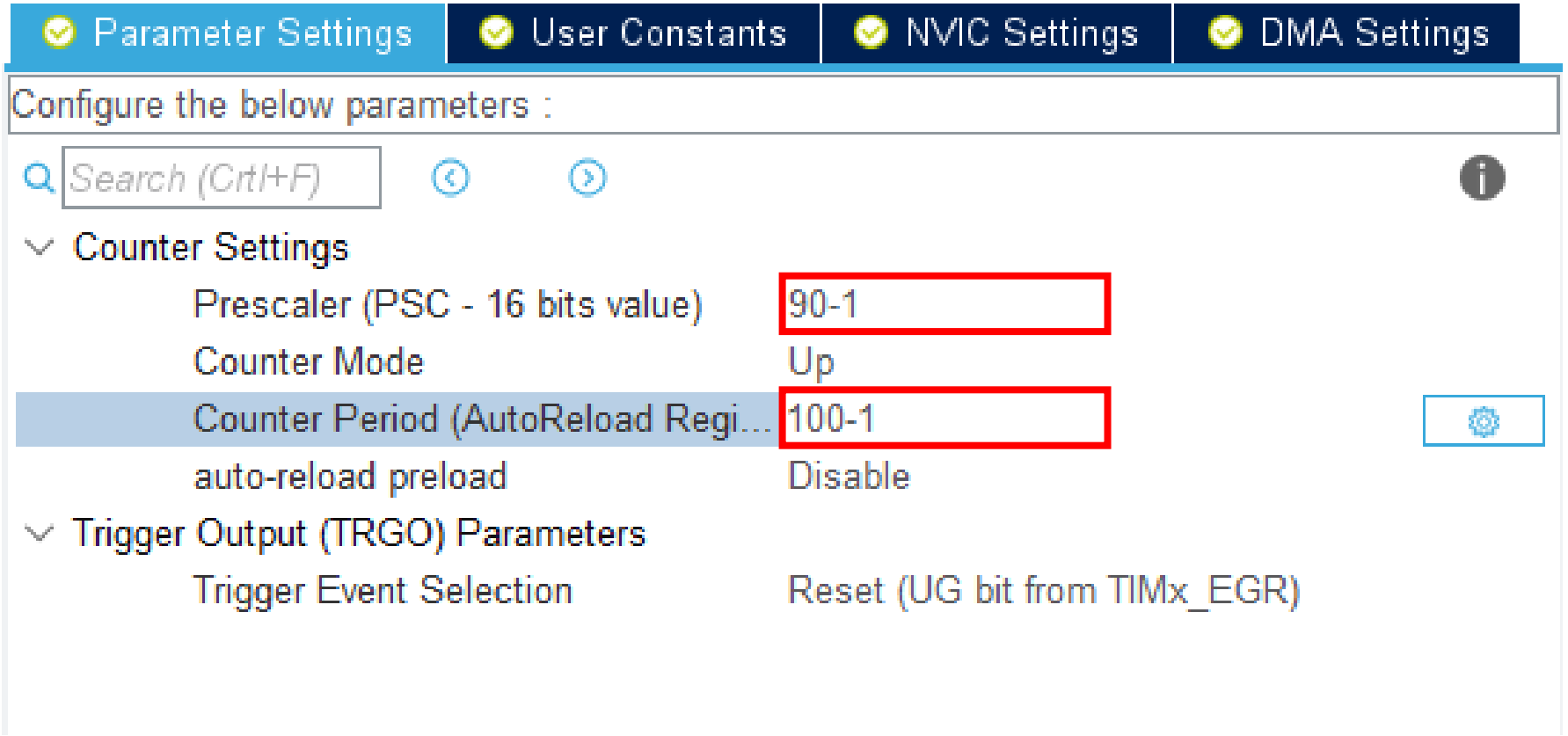

APB1는 90MHz

# Timer

90MHz(90000000Hz)를 10000Hz로 만드려면?

90000000 / 90 / 100 =10000

10000Hz이면
1000(ms) / 10000(Hz) = 0.1(ms)
0.1ms 주기

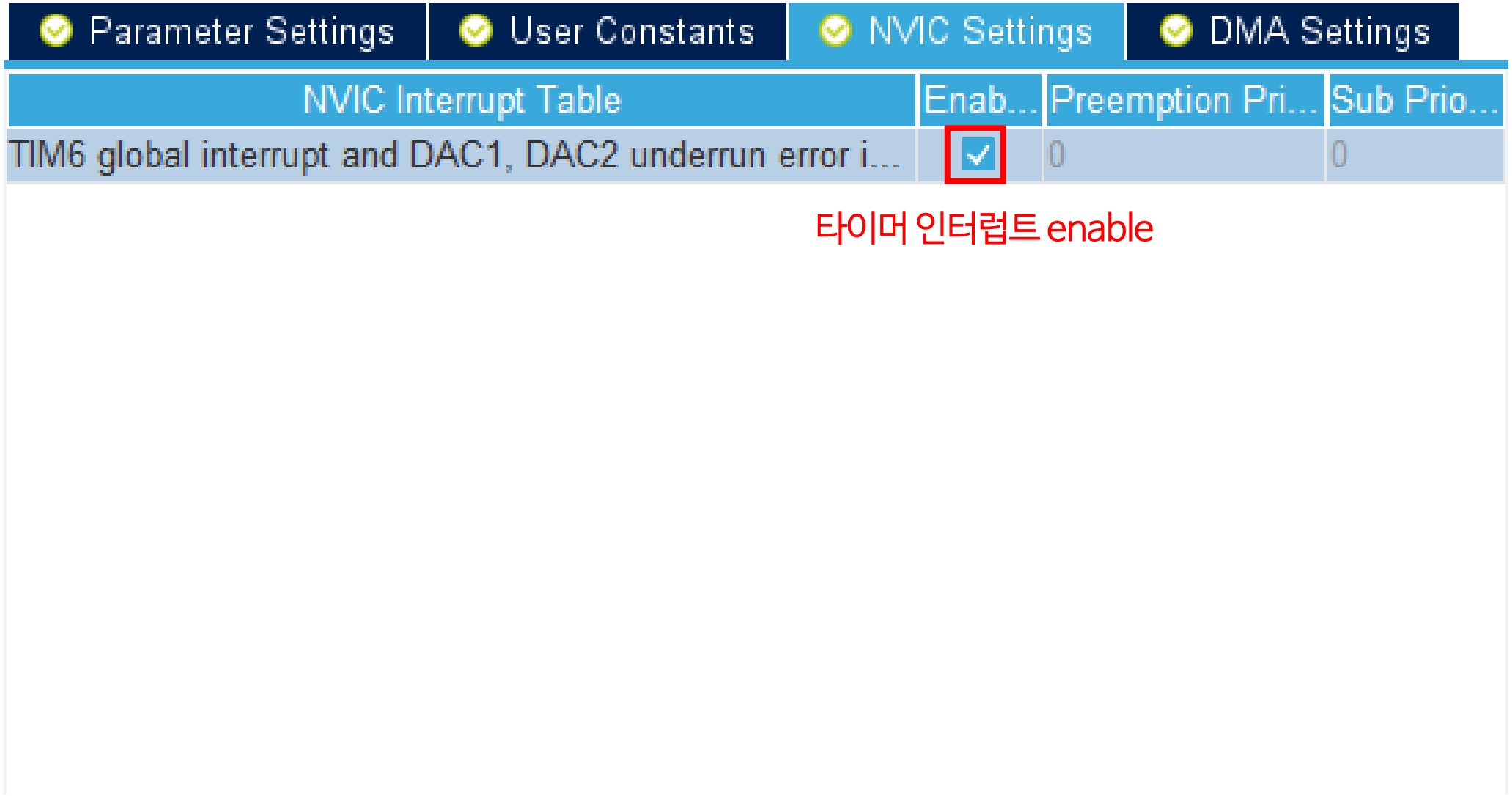



$$1tick = \frac{(prescaler + 1)}{clock}\ (s)$$

$$period = 1tick * (count + 1)\ (s)$$

# Timer




Parameter Settings
User Constants
NVIC Settings
DMA Settings
NVIC Interrupt Table
Enab...
Preemption Pri...
Sub Prio...
TIM6 global interrupt and DAC1, DAC2 underrun error i...
0
0
타이머 인터럽트 enable

# Timer

ROBIT ROBOT SPORT GAME TEAM

## Timer6 시작

```
LL_TIM_EnableIT_UPDATE(TIM6);
LL_TIM_EnableCounter(TIM6);
```

## Timer6 주기로 실행할 코드

```
void TIM6_DAC_IRQHandler(void)
{
  /* USER CODE BEGIN TIM6_DAC_IRQn 0 */
    if(LL_TIM_IsActiveFlag_UPDATE(TIM6))
    {
        cnt++;
```

# UART

# UART

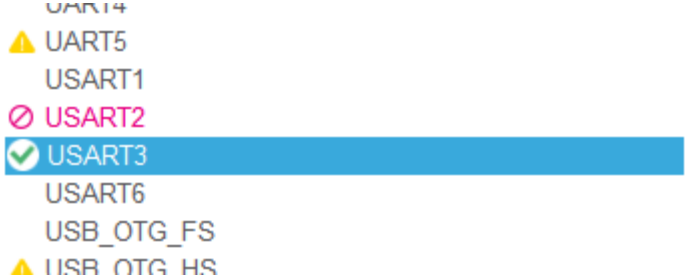
HAL : STM에서 제공하는 하드웨어 추상화 라이브러리, 무겁다

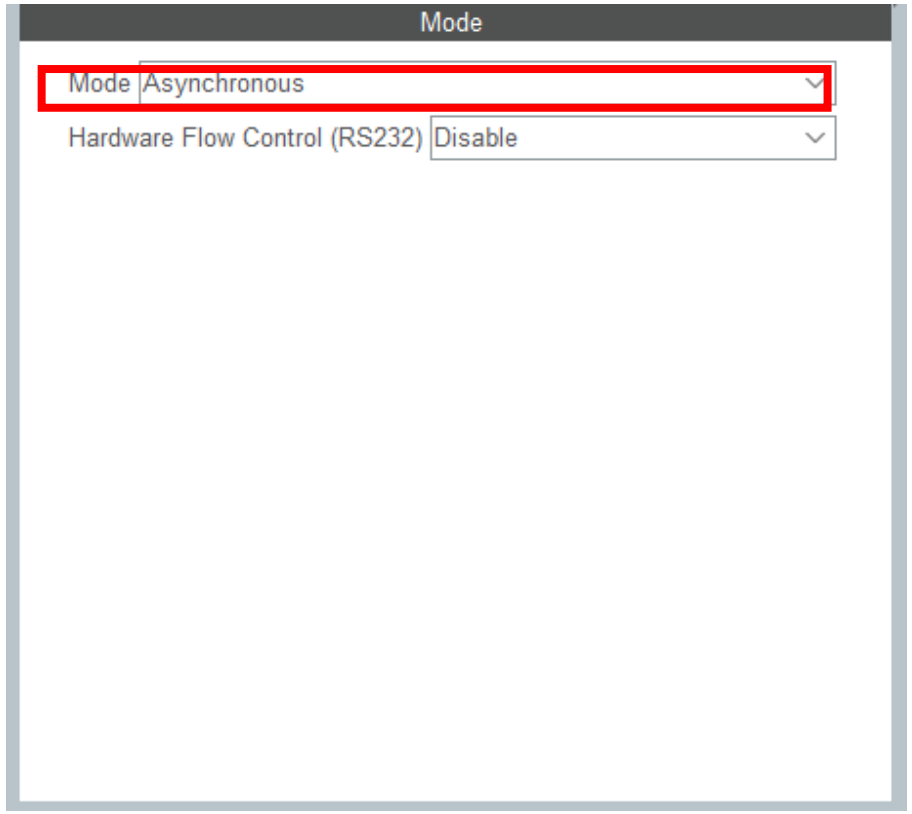
---

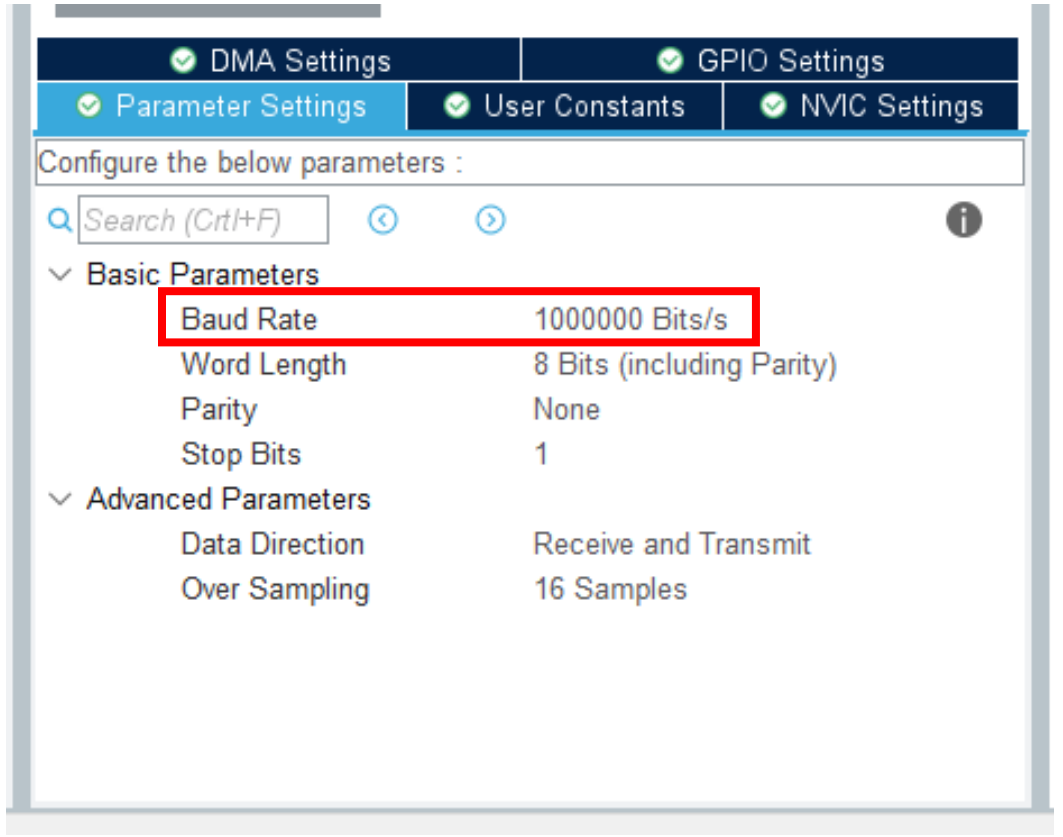
## LL + DMA + Circular queue + IDLE interrupt

: HAL 대신 레지스터에 직접 접근하는 LL을 사용하고, DMA. 원형큐, IDLE 인터럽트를 사용한 효율적인 방식

# UART




Baud Rate 설정 : 115200

# UART



## UART(Universal Asynchronous Receiver Transmit)

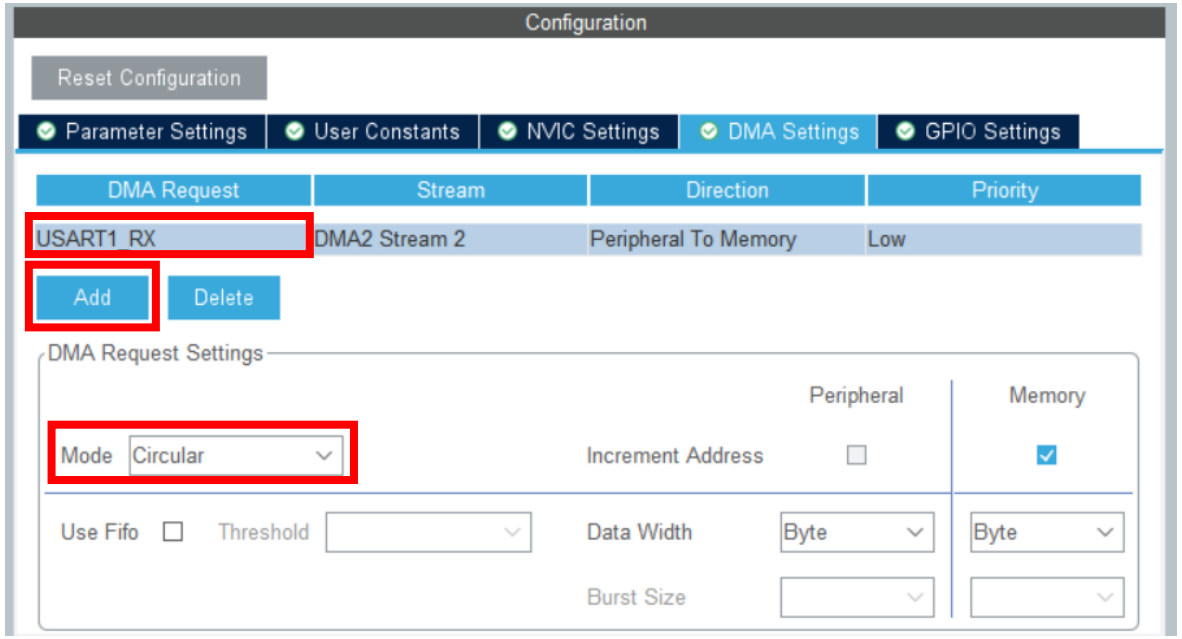
비동기 방식만 지원

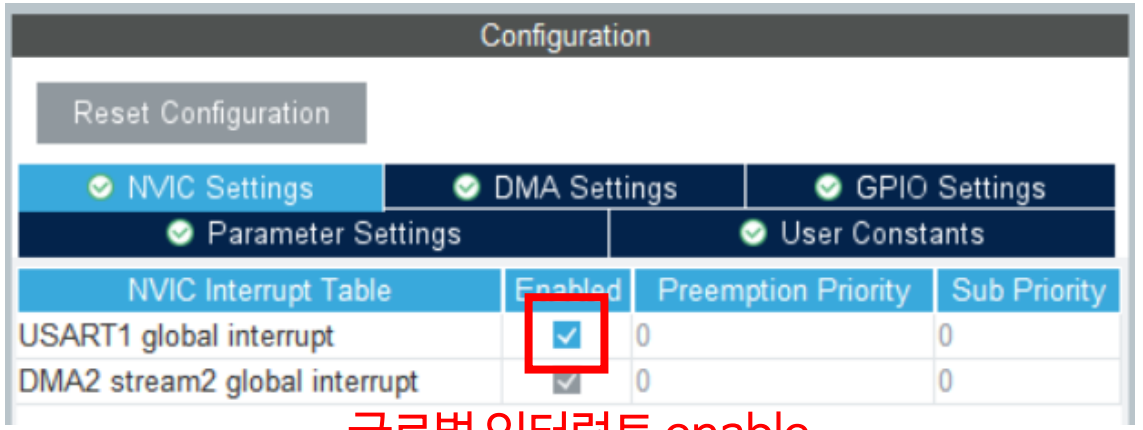
---

## USART(Universal Synchronous Asynchronous Receiver Transmit)

동기방식, 비동기방식 모두 지원

# UART

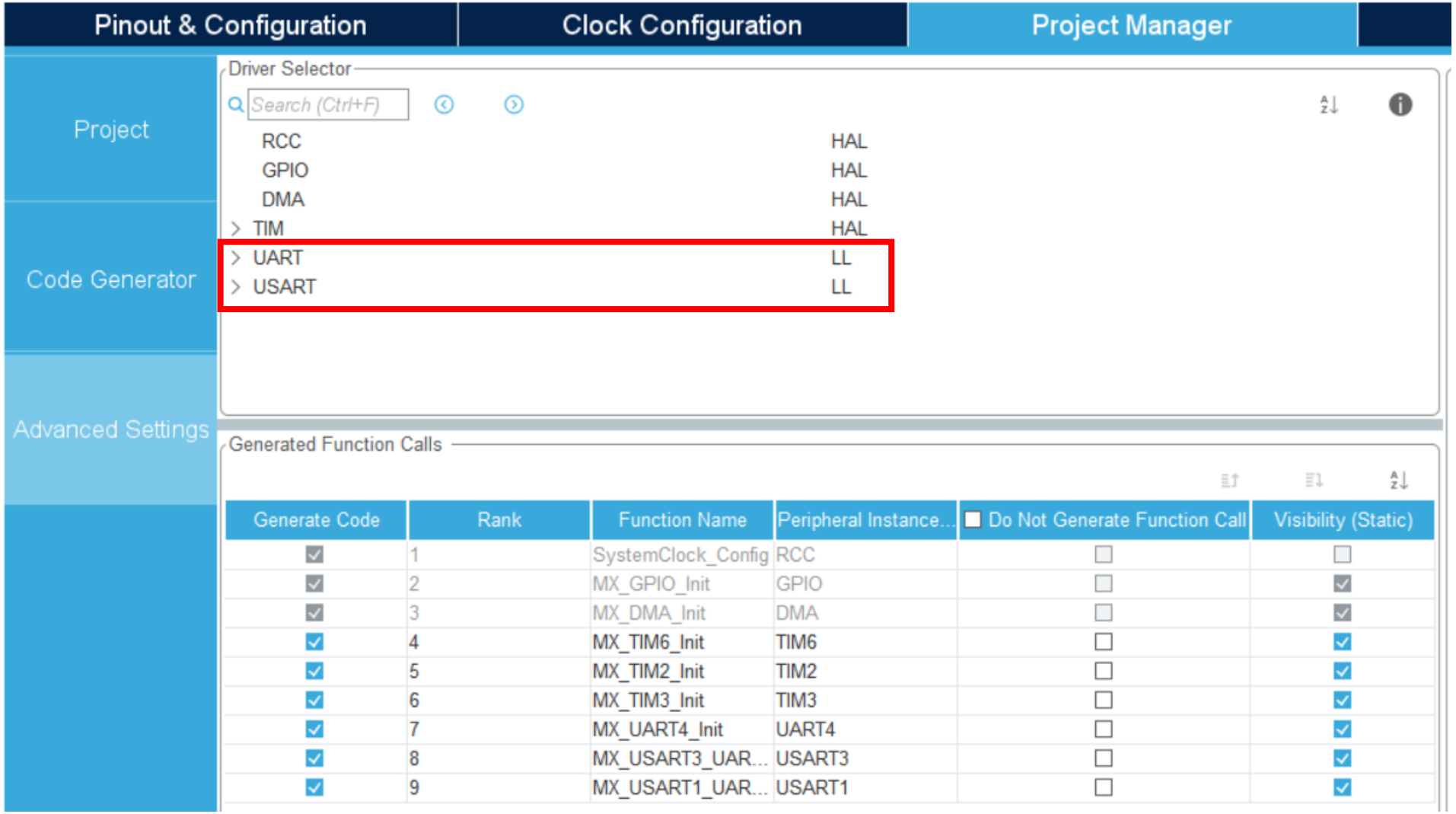



글로벌 인터럽트 enable

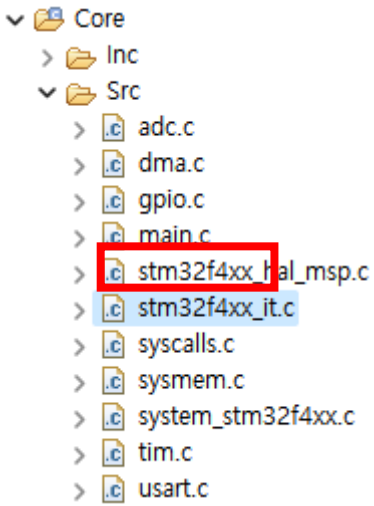
* Stream칸에 DMAx Stream x 메인문 코드에 사용
(DMA1 Stream 1)

# UART


Pinout & Configuration
Clock Configuration
Project Manager
Project
Code Generator
Advanced Settings
Driver Selector
Search (Ctrl+F)
RCC HAL
GPIO HAL
DMA HAL
TIM HAL
UART LL
USART LL
Generated Function Calls
Generate Code
Rank
Function Name
Peripheral Instance...
Do Not Generate Function Call
Visibility (Static)
1 SystemClock_Config RCC
2 MX_GPIO_Init GPIO
3 MX_DMA_Init DMA
4 MX_TIM6_Init TIM6
5 MX_TIM2_Init TIM2
6 MX_TIM3_Init TIM3
7 MX_UART4_Init UART4
8 MX_USART3_UAR... USART3
9 MX_USART1_UAR... USART1

# UART

## RX




```
uint8_t rxBuffer[32] = {0, };
```

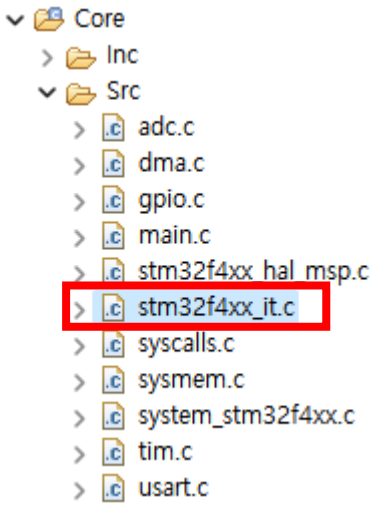
```
LL_DMA_ConfigAddresses(DMA1,
        LL_DMA_STREAM_1,
        LL_USART_DMA_GetRegAddr(USART3),
        rxBuffer,
        LL_DMA_DIRECTION_PERIPH_TO_MEMORY);
LL_DMA_SetDataLength(DMA1, LL_DMA_STREAM_1, sizeof(rxBuffer));
LL_DMA_EnableStream(DMA1, LL_DMA_STREAM_1);
LL_USART_EnableIT_IDLE(USART3);
LL_USART_EnableDMAReq_RX(USART3);
```

수신한 packet 받을 배열

배열 크기

main.c에 작성

# UART




```c
void USART3_IRQHandler(void)
{
  /* USER CODE BEGIN USART3_IRQn 0 */
    if(LL_USART_IsActiveFlag_IDLE(USART3))
    {
        //int len = RX_BUFFER_SIZE - LL_DMA_GetDataLength(DMA1, LL_DMA_STREAM_1);

        //connect = readPacket(&serial, PCrxBuffer, len);

        LL_DMA_DisableStream(DMA1, LL_DMA_STREAM_1);

        LL_DMA_ClearFlag_DME1(DMA1);
        LL_DMA_ClearFlag_FE1(DMA1);
        LL_DMA_ClearFlag_HT1(DMA1);
        LL_DMA_ClearFlag_TC1(DMA1);
        LL_DMA_ClearFlag_TE1(DMA1);

        LL_DMA_EnableStream(DMA1, LL_DMA_STREAM_1);

        LL_USART_ClearFlag_IDLE(USART3);
    }

  /* USER CODE END USART3_IRQn 0 */
  /* USER CODE BEGIN USART3_IRQn 1 */

  /* USER CODE END USART3_IRQn 1 */
}
```

각자 RX 코드 작성

# UART

## TX

```
void writePacket(SerialLine *serialLine, uint8_t packet[], int length)
{
    for(int i=0; i<length; i++)
    {
        while(!LL_USART_IsActiveFlag_TXE(USART3));
        LL_USART_TransmitData8(USART3, packet[i]);
    }
}
```

전송할 데이터 길이

전송할 데이터 패킷

# 과제

# 과제1

1. 타이머 인터럽트를 이용하여
PD2 LED TOGGLE (1초 주기 - 0.5초동안 켜지고 0.5초동안 꺼짐)

# 과제2

UART

U2D2와 통신하여 시리얼 프로그램과 값 주고받음 보이기
(RX, TX모두 구현)

# 과제3

UART 통신(2인 1조)
A STM : MASTER
B STM : SLAVE

- 보고서 PPT 10장이상
- Ex) uart개념,
- 동기,비동기방식 차이점
- 코드 설명
- 등등

MASTER
- 전체적인 통신 주기 설정
- PSD 값 TX
- 거리값 cm RX

SLAVE
- PSD값 RX
- PSD값 cm 변환
- 거리값 cm TX
(정확한 값 RX 후 바로 TX해야함,
특정 주기로하는건X)

- 통신 시작을 알리는 packet 필요
- 데이터가 정확하게 들어왔는지 확인하는 error check 필요

# DYNAMIXEL PACKET

ROBIT ROBOT SPORT GAME TEAM

| Header 1 | Header 2 | Header 3 | Reserved | Packet ID | Length 1 | Length 2 | Instruction | Param | Param | Param | CRC 1 | CRC 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xFF | 0xFF | 0xFD | 0x00 | Packet ID | Len_L | Len_H | Instruction | Param 1 | ··· | Param N | CRC_L | CRC_H |

## 2. 5. Instruction

Packet의 용도를 정의하는 필드

| 값 | 명령 | 설명 |
|---|---|---|
| 0x01 | Ping | Packet ID와 동일한 ID를 가지는 장치에 Packet이 도달했는지 여부 확인을 위한 Instruction |
| 0x02 | Read | 장치로부터 데이터를 읽어오기 위한 Instruction |
| 0x03 | Write | 장치에 데이터를 쓰기 위한 Instruction |
| 0x04 | Reg Write | Instruction Packet을 대기 상태로 등록하는 Instruction, Action 명령에 의해 실행됨 |
| 0x05 | Action | Reg Write로 대기 등록한 Packet을 실행하는 Instruction |

# ENCODER

# Encoder

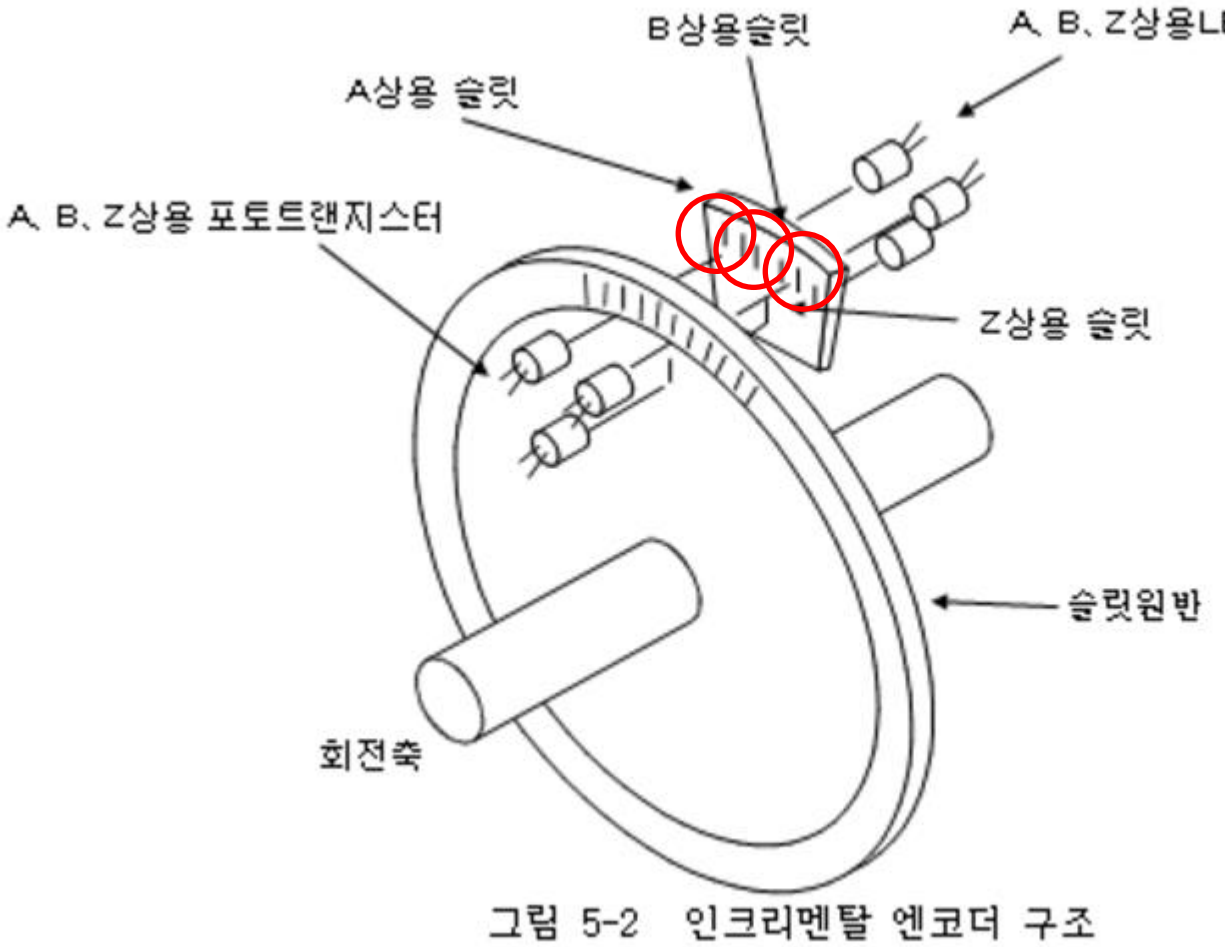



그림 5-2 인크리멘탈 엔코더 구조

- 모터의 위치, 회전속도 알기 위한 센서
- A상, B상, Z상 존재
- Photo interrupter 사용

- Photo interrupter

– 발광 다이오드와 Photo Transistor를 조합 한 것으로서 발광 다이오드의 빛을 포토 트랜지스터가 수광하면 전류가 흐른다.

- Photo Transistor

– 빛을 쬐면 전류가 흐르는 Transistor

# Encoder

ROBIT ROBOT SPORT GAME TEAM

## Absolute Encoder

모터의 위치에 따른 절대적인 값을 얻는데 사용.
전원을 켜서 엔코더 값을 읽었을 때 각도에 따른 값이 나온다.

Ex) 모터의 위치가 90도일때, 어떠한 상황에서도 엔코더 값은 125가 나온다.

매니퓰레이터의 전원을 켰을 때 각 관절의 위치 read

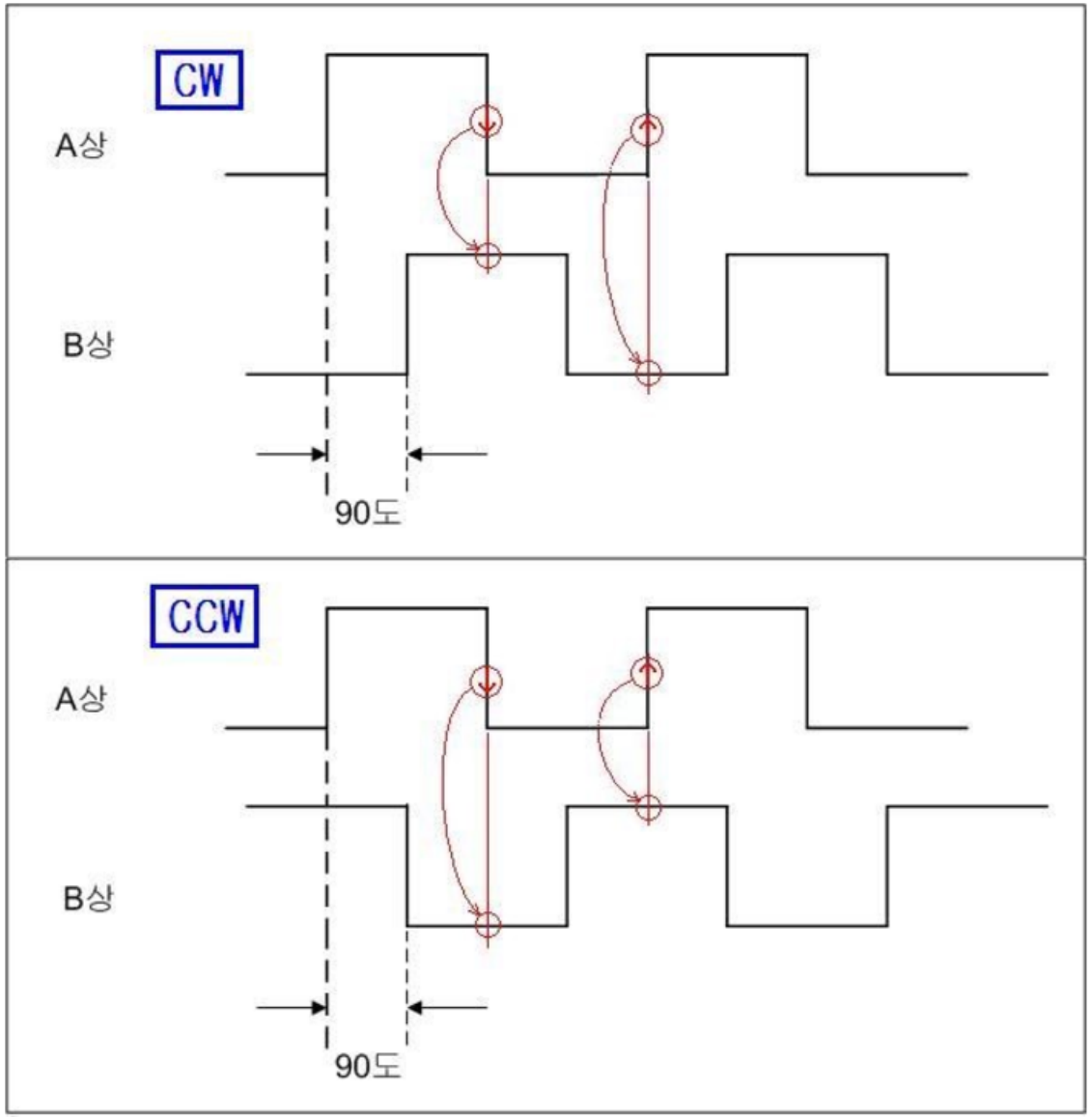
## Incremental Encoder

모터의 위치에 따른 상대적인 값을 얻는데 사용.
전원을 켜서 엔코더 값을 읽었을 때 각도에 상관없이 0부터 시작한다.

Ex) 모터의 위치가 90도일때, 엔코더 값은 때에 따라 다를 수 있지만, 모터가 1rad/s의 일정한 속도로 회전할 때 항상 1000씩 증가한다.

모바일 로봇의 바퀴 회전속도를 일정하게 유지하기

# Encoder




- 2체배

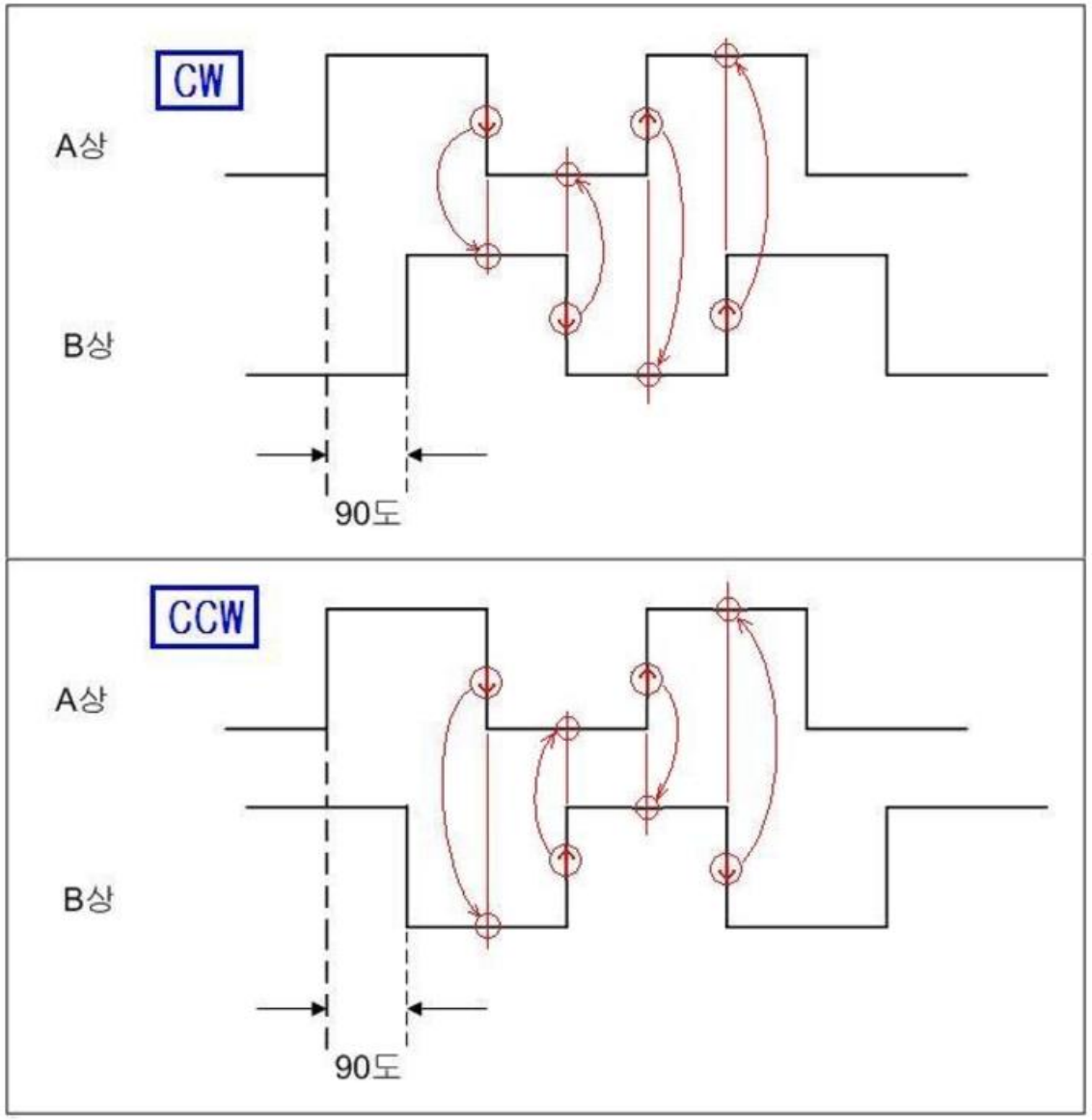
1. Cw(Clock Wise)

- A상의 Rising Edge와 Falling Edge에서 Counting
- A상의 Rising Edge : B상 Low
- A상의 Falling Edge : B상 High

2. CCW(Counter Clock Wise)

- A상의 Rising Edge와 Falling Edge에서 Counting
- A상의 Rising Edge : B상 High
- A상의 Falling Edge : B상 Low

# Encoder




- 4체배

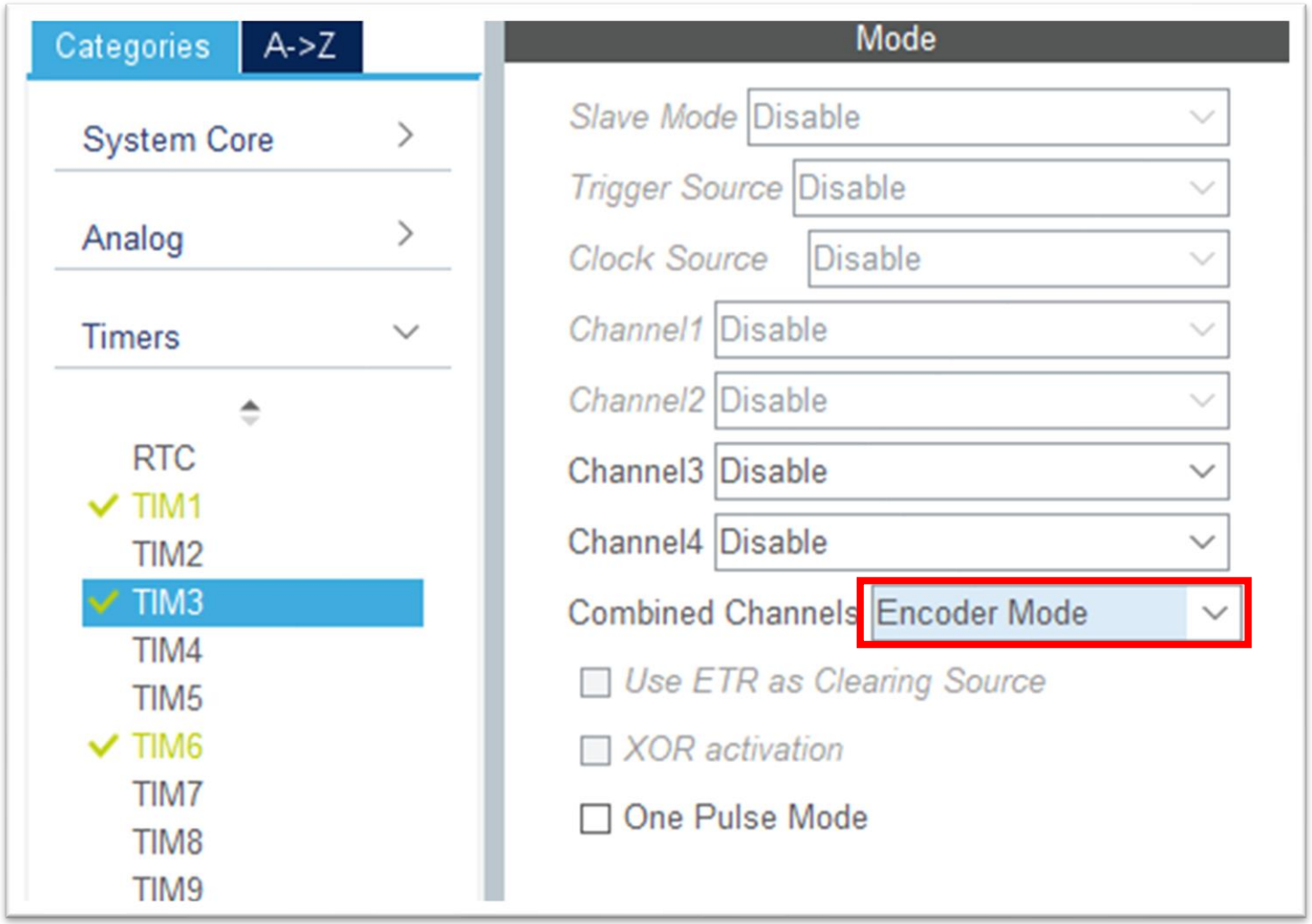
1. Cw(Clock Wise)

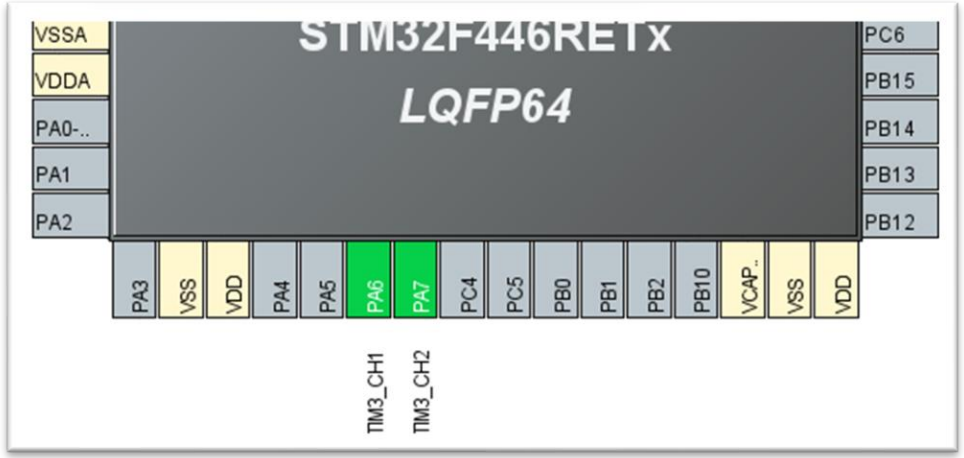
- A상과 B상의 Both Edge에서 Counting
- A상의 Rising Edge : B상 Low
- A상의 Falling Edge : B상 High
- B상의 Rising Edge : A상 High
- B상의 Falling Edge : A상 Low

2. CCW(Counter Clock Wise)

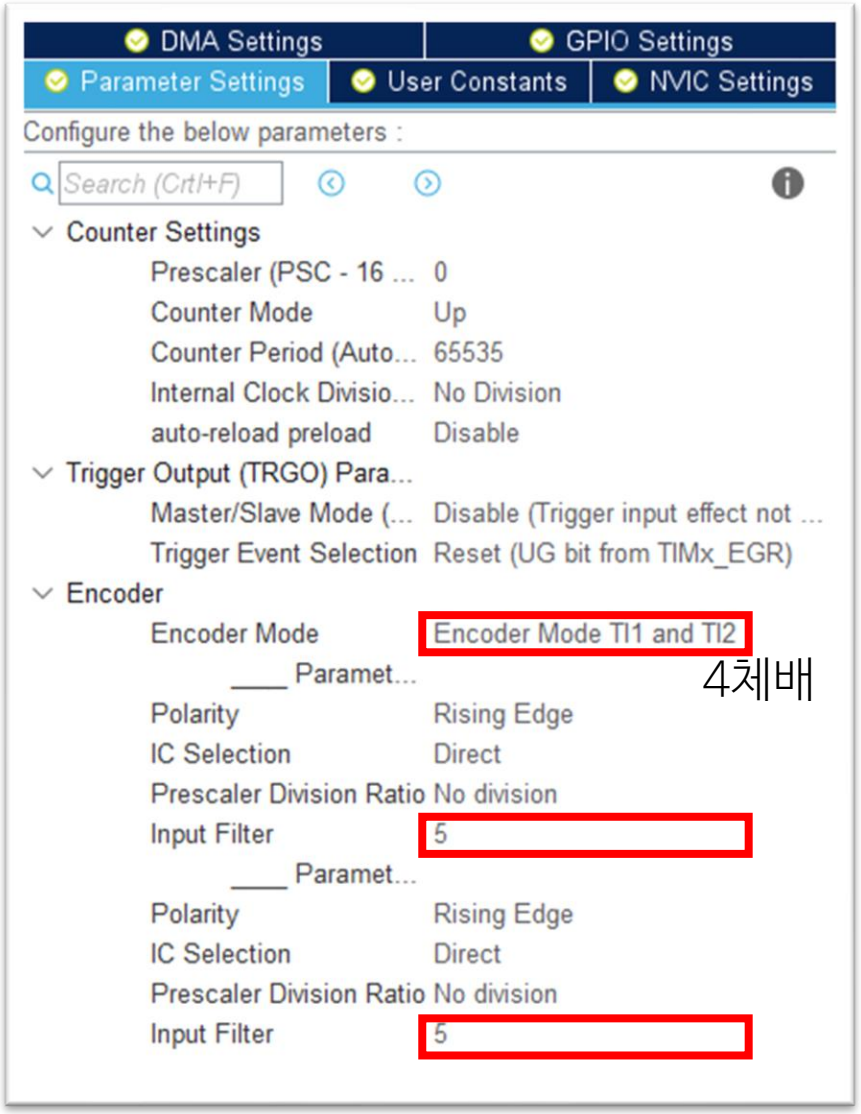
- A상의 Rising Edge와 Falling Edge에서 Counting
- A상의 Rising Edge : B상 High
- A상의 Falling Edge : B상 Low
- B상의 Rising Edge : A상 Low
- B상의 Falling Edge : A상 High

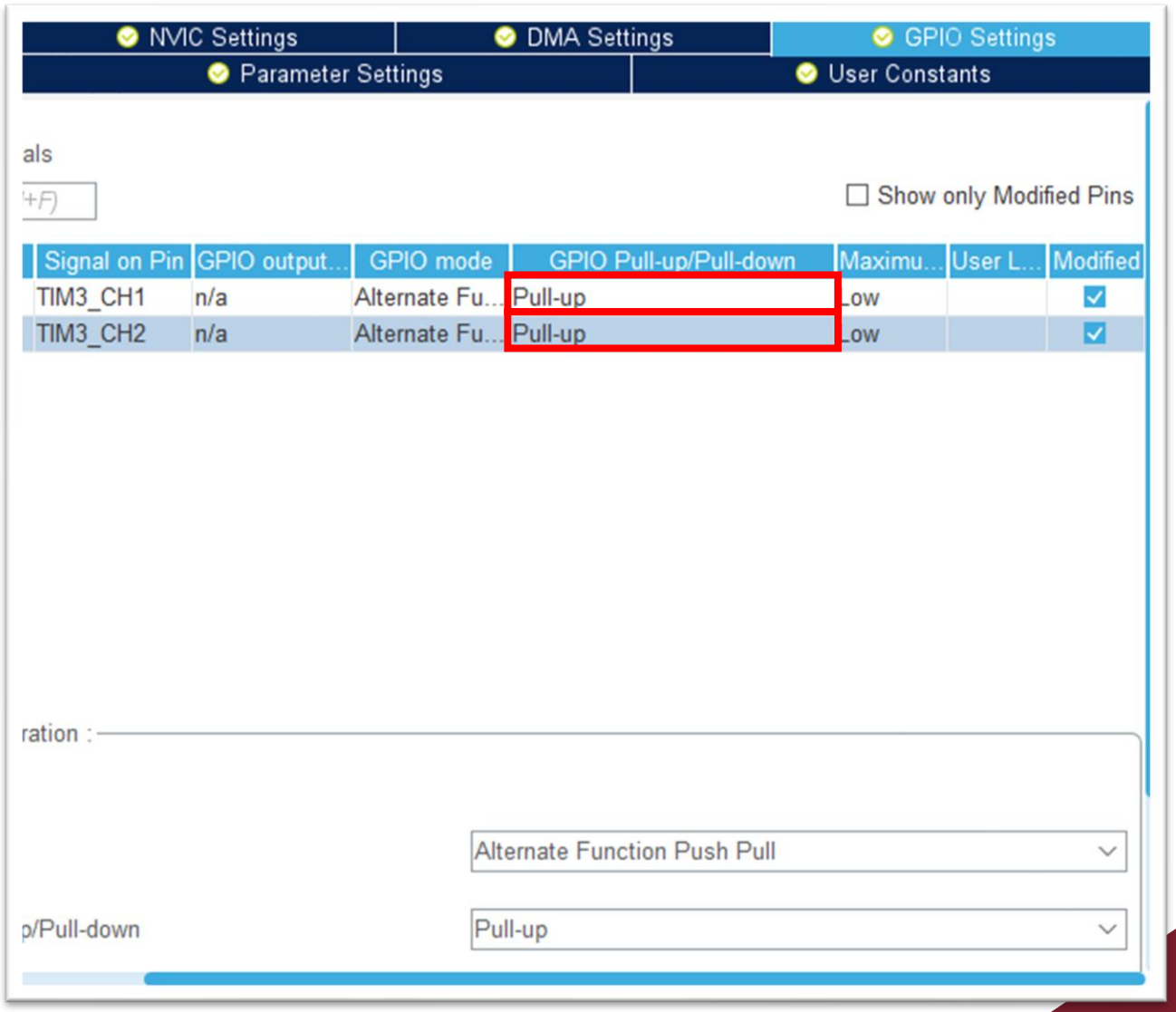
# Encoder


Categories
A->Z
System Core
Analog
Timers
RTC
TIM1
TIM2
TIM3
TIM4
TIM5
TIM6
TIM7
TIM8
TIM9
Mode
Slave Mode Disable
Trigger Source Disable
Clock Source Disable
Channel1 Disable
Channel2 Disable
Channel3 Disable
Channel4 Disable
Combined Channels Encoder Mode
Use ETR as Clearing Source
XOR activation
One Pulse Mode


STM32F446RETx
LQFP64
VSSA
VDDA
PA0-..
PA1
PA2
PC6
PB15
PB14
PB13
PB12
PA3
VSS
VDD
PA4
PA5
PA6
PA7
PC4
PC5
PB0
PB1
PB2
PB10
VCAP..
VSS
VDD
TIM3_CH1
TIM3_CH2

# Encoder


DMA Settings
GPIO Settings
Parameter Settings
User Constants
NVIC Settings
Configure the below parameters :
Search (Crtl+F)
Counter Settings
Prescaler (PSC - 16 ... 0
Counter Mode Up
Counter Period (Auto... 65535
Internal Clock Divisio... No Division
auto-reload preload Disable
Trigger Output (TRGO) Para...
Master/Slave Mode (... Disable (Trigger input effect not ...
Trigger Event Selection Reset (UG bit from TIMx_EGR)
Encoder
Encoder Mode Encoder Mode TI1 and TI2
____ Paramet...
Polarity Rising Edge
IC Selection Direct
Prescaler Division Ratio No division
Input Filter 5
____ Paramet...
Polarity Rising Edge
IC Selection Direct
Prescaler Division Ratio No division
Input Filter 5
4체배


NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
als
Show only Modified Pins
Signal on Pin
GPIO output...
GPIO mode
GPIO Pull-up/Pull-down
Maximu...
User L...
Modified
TIM3_CH1 n/a Alternate Fu... Pull-up ow
TIM3_CH2 n/a Alternate Fu... Pull-up ow
ration :
Alternate Function Push Pull
p/Pull-down
Pull-up

# Encoder

ROBIT ROBOT SPORT GAME TEAM

엔코더 값 받기 위한 TIMER 시작

```
//timer3 - encoder
HAL_TIM_Encoder_Start(&htim3, TIM_CHANNEL_ALL);
```

엔코더 값 받아오기

```
encoder_value = TIM3->CNT;
```

# Thank you