

3. Timer

3. Timer

- Timer 개요

- 타이머(Timer)란 MCU의 매우 중요한 기능으로, 타이머 클럭에 따라 카운터를 세는 기능을 한다.
- 해당 타이머의 기능을 응용해 많은 추가 기능들을 만들어 낼 수 있다.



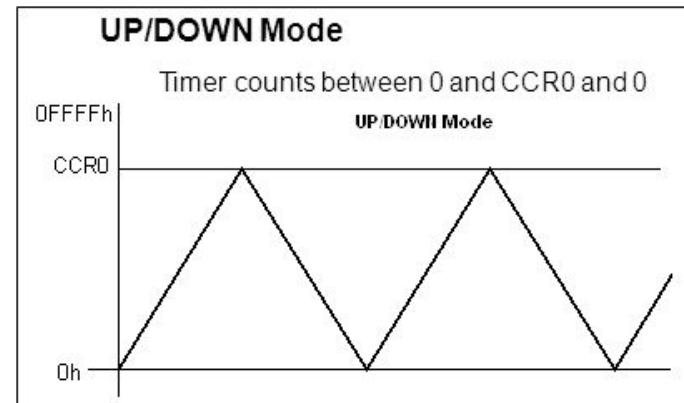
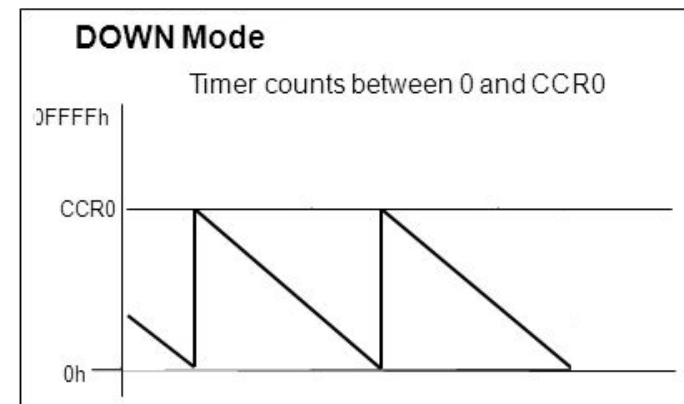
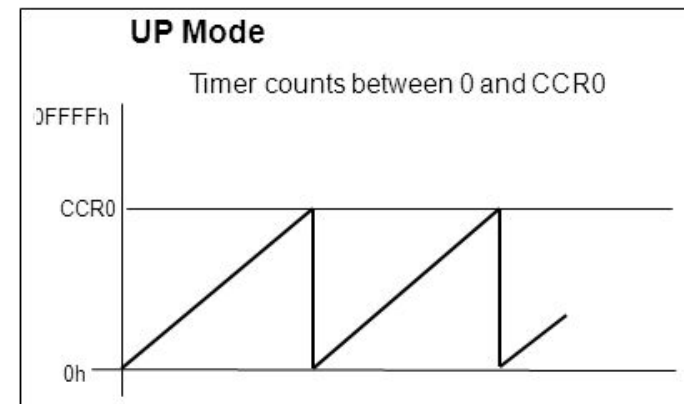
3. Timer

- Timer 개요

Q. 타이머는 카운터를 어떻게 세는가?

A.

- 타이머는 크게 3가지 방식으로 카운터를 센다.
- **Up Mode** : 매 타이머 클럭마다 카운터를 1씩 증가시킨다. 그러다 **지정된 값** 또는 최대값을 넘으면 0으로 초기화 한다.
- **Down Mode** : 매 타이머 클럭마다 카운터를 1씩 감소시킨다. 그러다 카운터가 0이 되면 **미리 지정된 값**으로 카운터를 초기화 한다.
- **Up/Down Mode** : Up Mode와 Down Mode의 결합. STM32에서는 Center Align Mode라고 부른다.



3. Timer

- Timer 개요

Q. 타이머는 카운터를 얼마나 셀 수 있는가? (카운터의 최대값은?)

A.

- 모든 타이머는 항상 몇 bit timer라고 명시되어 있다.
- 예를 들어 **16bit timer**는 카운터가 16bit 이므로, $0 \sim 2^{16}-1(65535)$ 까지 셀 수 있다.
- 만약 **32bit timer**라면 카운터가 32bit 이므로, $0 \sim 2^{32}-1(4,294,967,295)$ 까지 셀 수 있다.

3. Timer

- Timer 개요

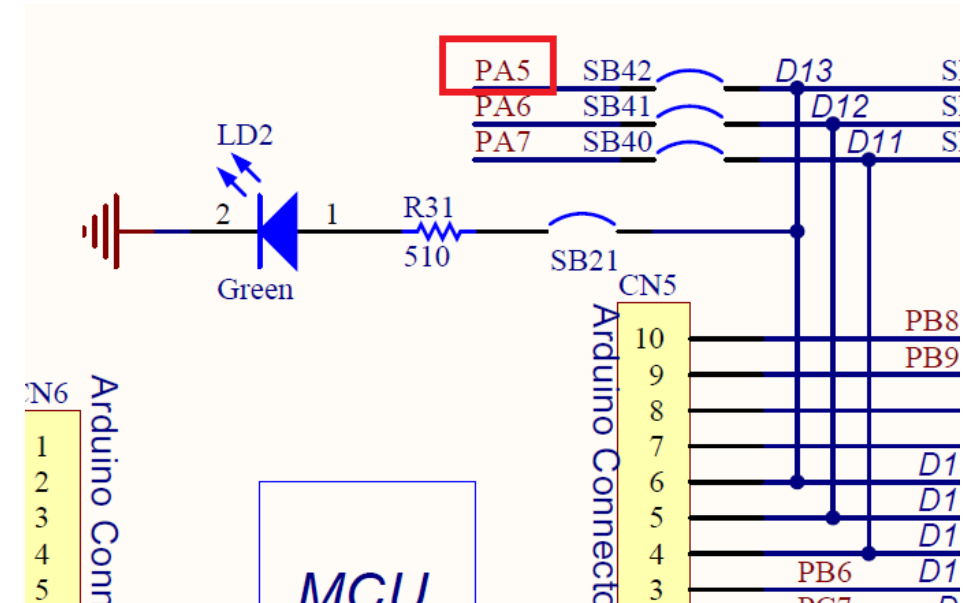
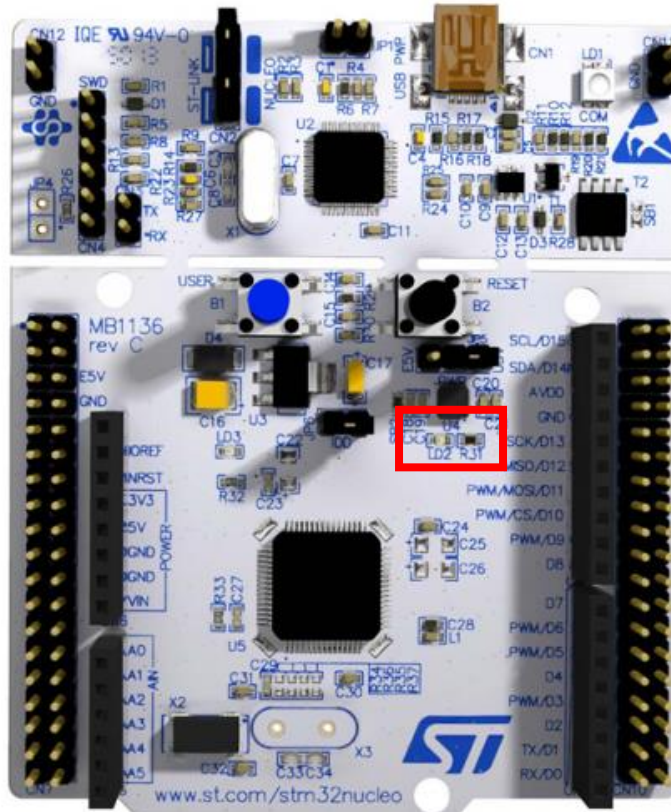
Q. 타이머는 카운터를 얼마나 셀 수 있는가?

Table 6. Timer feature comparison

| Timer type | Timer | Counter resolution | Counter type | Prescaler factor | DMA request generation | Capture/compare channels | Complementary output | Max interface clock (MHz) | Max timer clock (MHz) ⁽¹⁾ |
|------------------|--------------|--------------------|-------------------|---------------------------------|------------------------|--------------------------|----------------------|---------------------------|--------------------------------------|
| Advanced-control | TIM1, TIM8 | 16-bit | Up, Down, Up/down | Any integer between 1 and 65536 | Yes | 4 | Yes | 90 | 180 |
| General purpose | TIM2, TIM5 | 32-bit | Up, Down, Up/down | Any integer between 1 and 65536 | Yes | 4 | No | 45 | 90/180 |
| | TIM3, TIM4 | 16-bit | Up, Down, Up/down | Any integer between 1 and 65536 | Yes | 4 | No | 45 | 90/180 |
| | TIM9 | 16-bit | Up | Any integer between 1 and 65536 | No | 2 | No | 90 | 180 |
| | TIM10, TIM11 | 16-bit | Up | Any integer between 1 and 65536 | No | 1 | No | 90 | 180 |
| | TIM12 | 16-bit | Up | Any integer between 1 and 65536 | No | 2 | No | 45 | 90/180 |
| | TIM13, TIM14 | 16-bit | Up | Any integer between 1 and 65536 | No | 1 | No | 45 | 90/180 |
| Basic | TIM6, TIM7 | 16-bit | Up | Any integer between 1 and 65536 | Yes | 0 | No | 45 | 90/180 |

- **Timer 실습 (0.5초 만들기)**

- 누클레오 보드 중앙에 있는 LD2 Led는 내부적으로 칩의 PA5 핀에 연결되어 있다.
- Timer를 이용해 정확히 0.5초 마다 LED를 깜빡여보자(Toggle)
- 현재 STM32의 클럭 속도는 **180MHz** 이다.

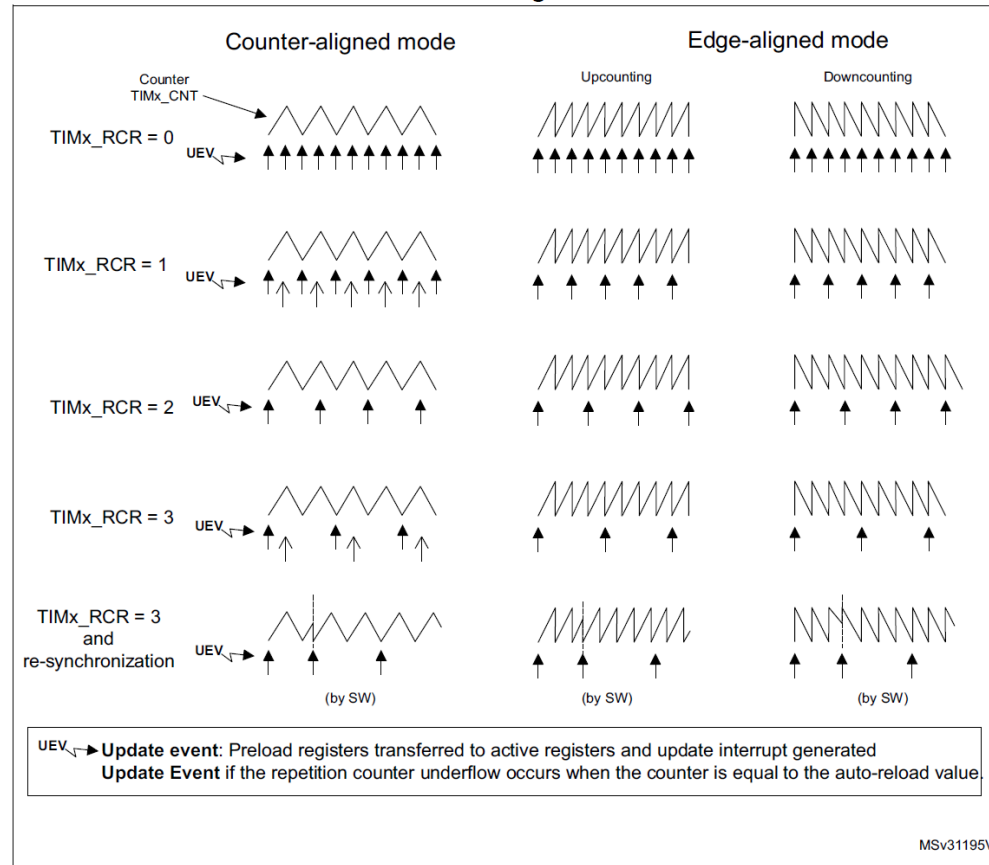


3. Timer

• Timer 실습 (0.5초 만들기)

- Timer의 카운터 값이 최대값(혹은 미리 지정된 값), 또는 0이 되어 값이 초기화 될 때를 **Update Event**라고 부른다.
- Update Event가 발생 할 때, **인터럽트**를 발생시킬 수 있다. 이를 이용해 보자.

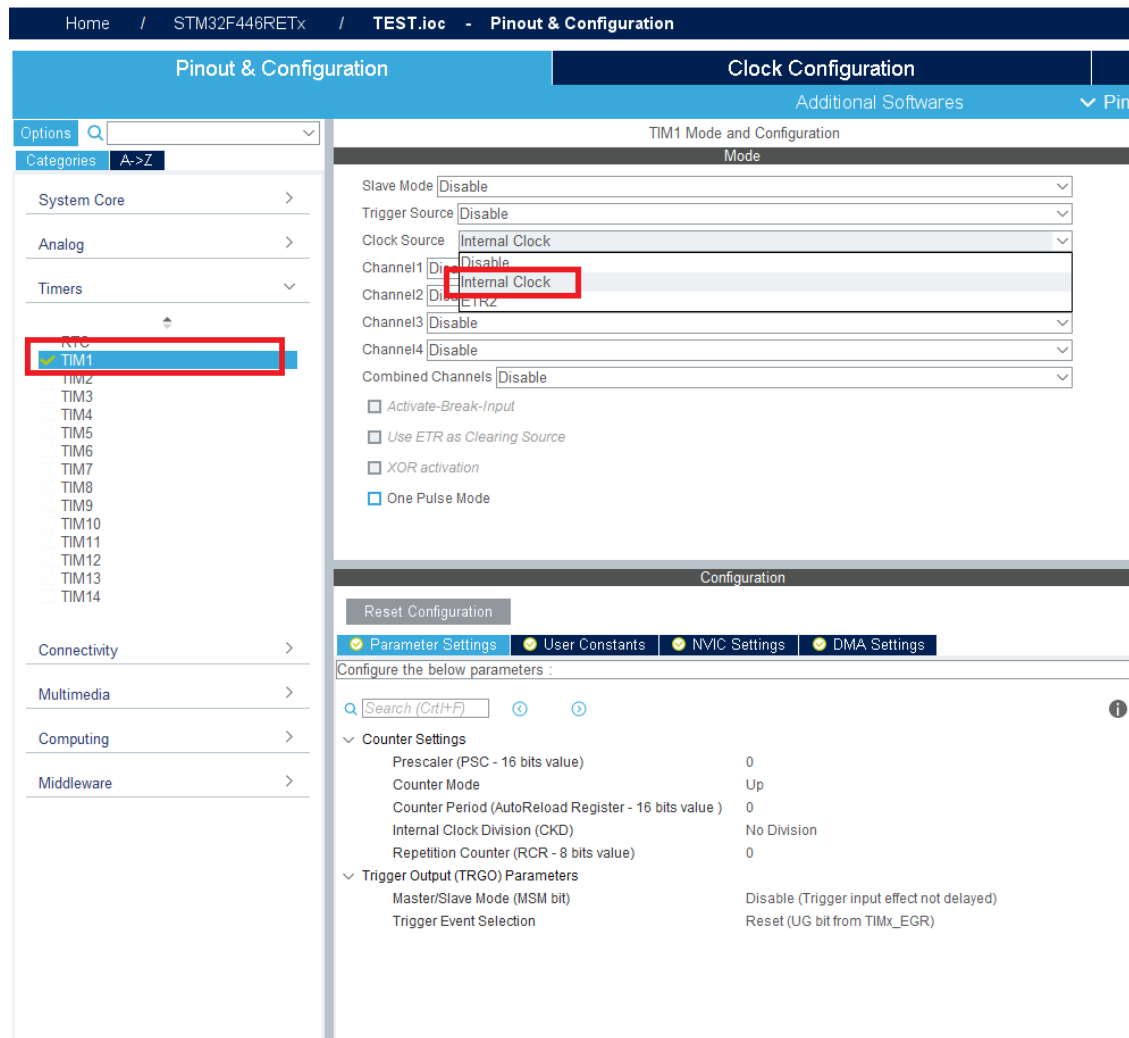
Figure 130. Update rate examples depending on mode and TIMx_RCR register settings



3. Timer

- Timer 실습 (0.5초 만들기)

- CubeMX에서 TIM1을 클릭 후, Clock Source에서 Internal Clock을 선택한다.



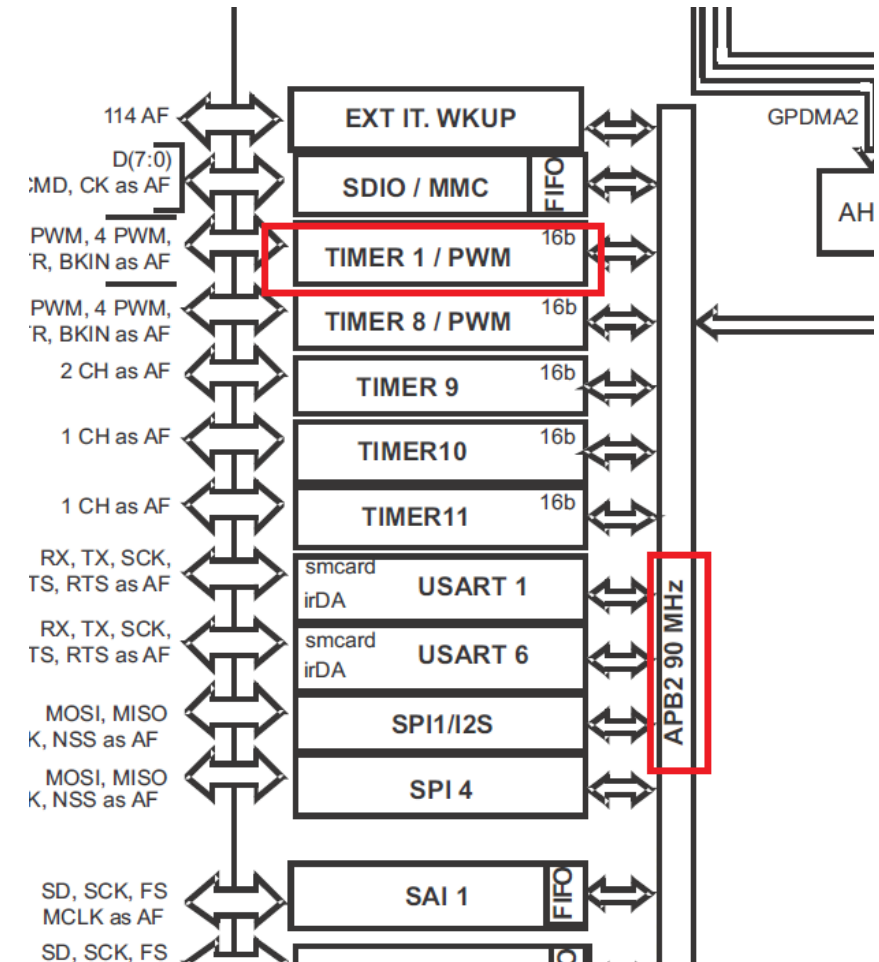
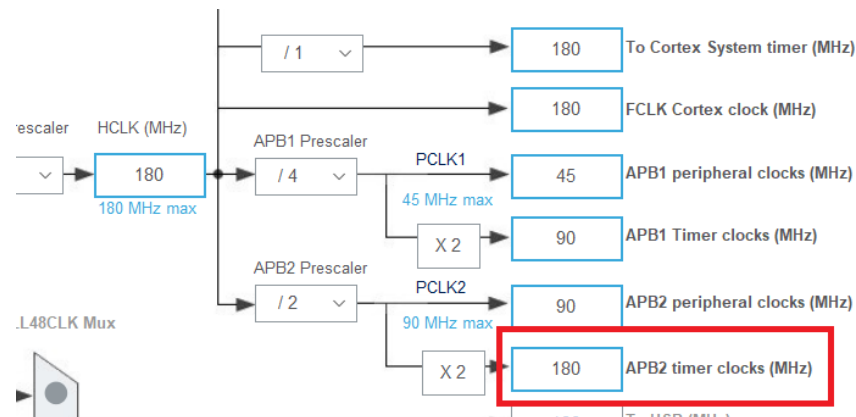
3. Timer

- Timer 실습 (0.5초 만들기)

Q. TIM1 Peripheral은 몇 Clock으로 동작하는가?

A.

- 데이터 시트에 따르면 **Timer 1은 APB2 Clock으로 동작한다.**
- Clock configuration에서 현재 APB2 timer clock이 180MHz로 설정되어 있으므로, **Timer 1은 180MHz로 동작한다.**



3. Timer

- Timer 실습 (0.5초 만들기)

Q. 180MHz TIM1의 Up Mode로 0.5초를 세려면 카운터를 몇까지 세야 하는가?
(TIM1은 16bit 타이머이다)

A.

$$\frac{0.5}{1/180000000} - 1 = 89,999,999 > 65535$$

Q. 카운터 최대값 보다 더 큰 시간을 쟈 수 있는가?

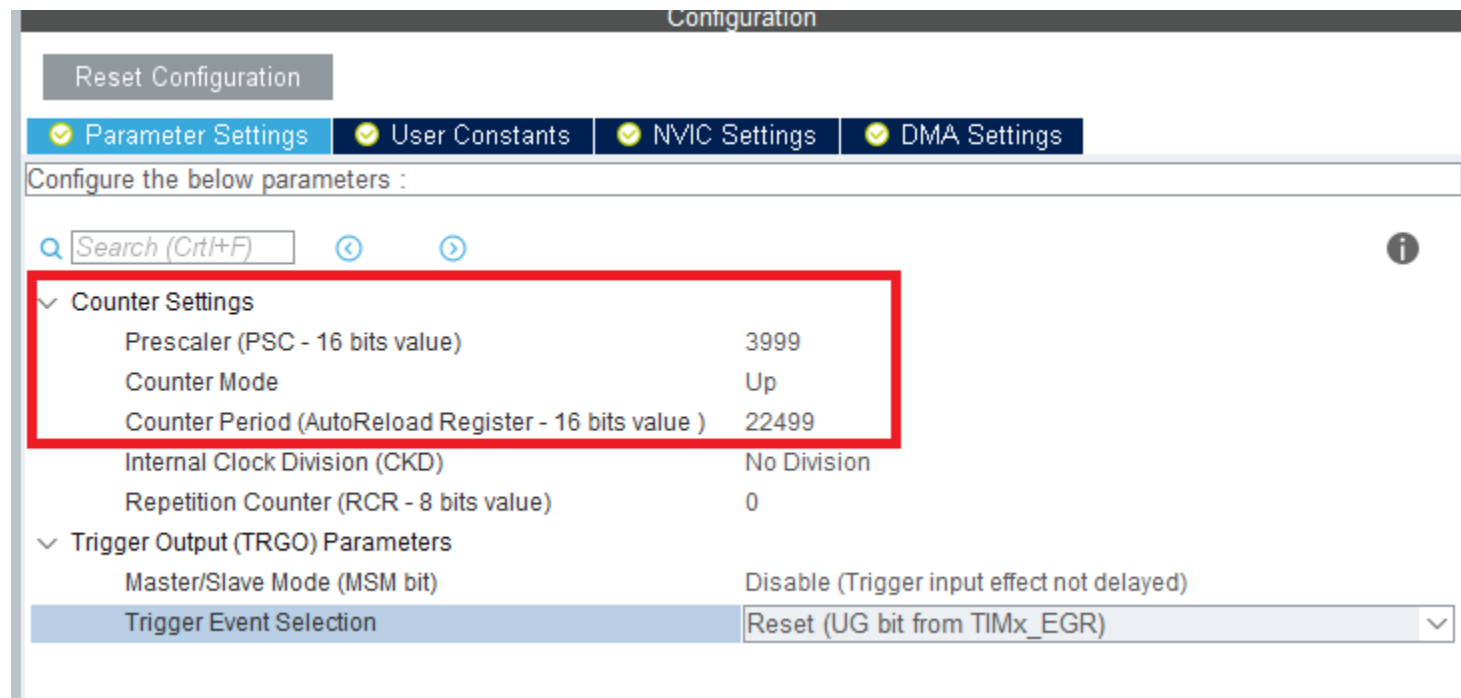
A. Prescaler를 이용하면 가능하다!

- 타이머는 Prescaler값 만큼 클럭이 지난 후에 카운터를 1 증가시킨다.
- 예를 들어 Prescaler를 2로 지정해 놓으면 3클럭 뒤에 카운터를 1 증가시킨다.

3. Timer

- Timer 실습 (0.5초 만들기)

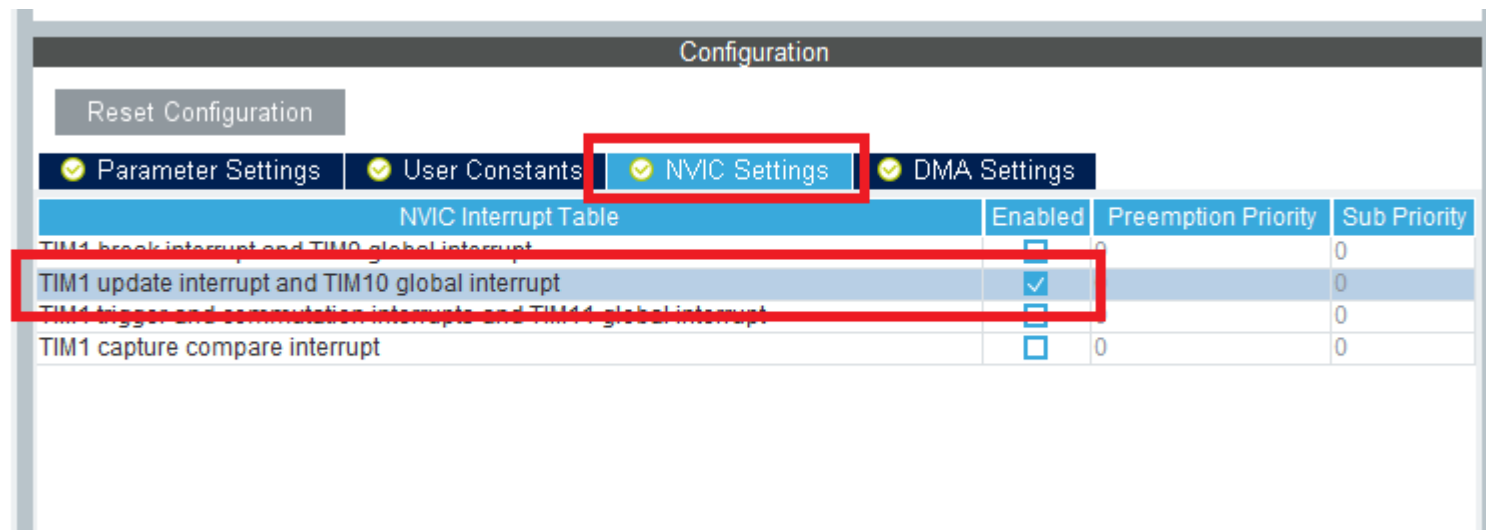
- Timer mode : **Up Mode**
- Prescaler : **3999**
- Counter Max Value : **22499**



3. Timer

- Timer 실습 (0.5초 만들기)

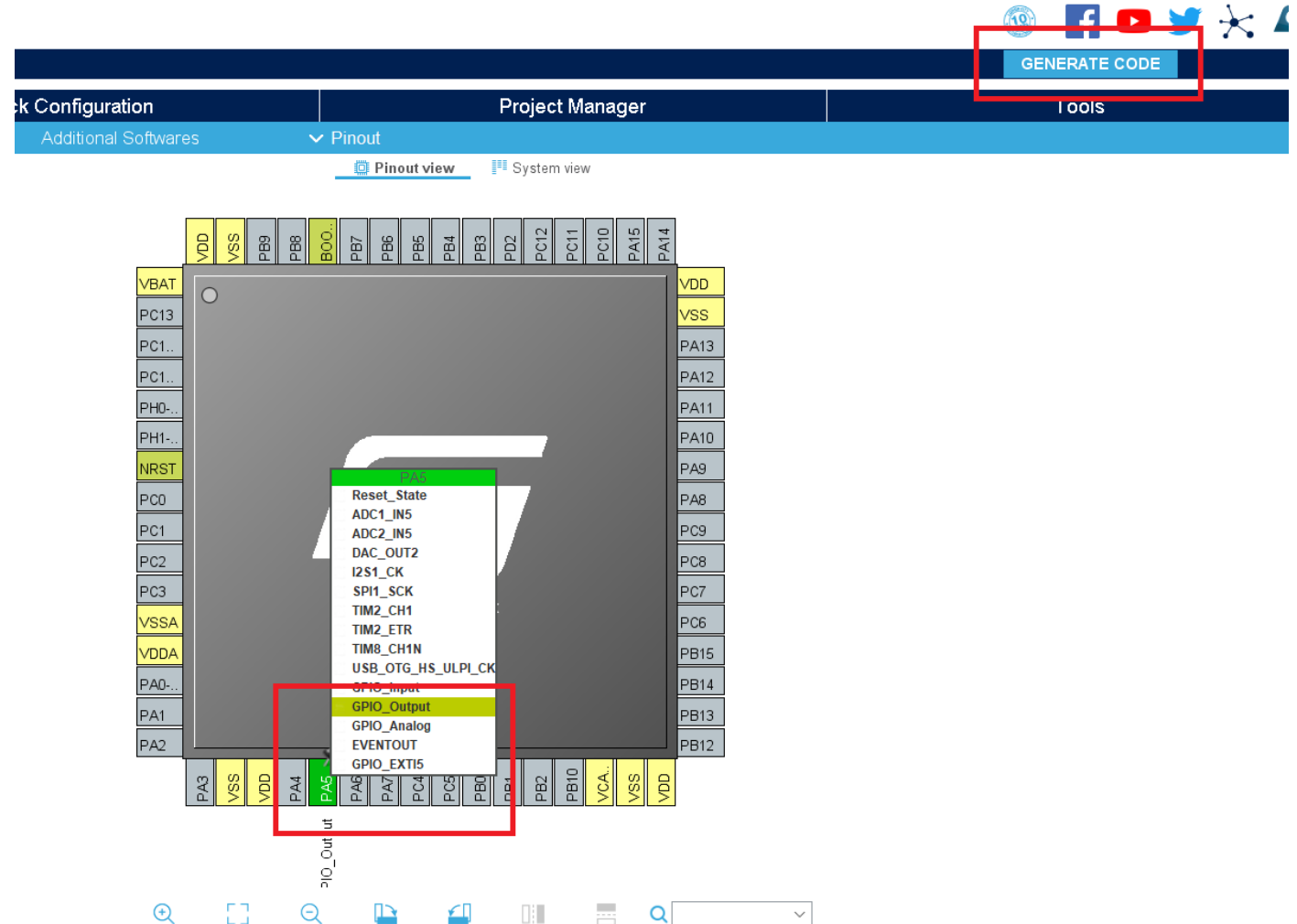
- Update Event마다 인터럽트를 발생시키기 위해 ,NVIC Settings에서 **TIM1 update interrupt**를 **Enable** 한다.



3. Timer

- Timer 실습 (0.5초 만들기)

- LED를 켜기 위해서 PA5에 GPIO_Output을 지정해 주고 "GENERATE CODE"를 통해 코드를 재생성한다.



3. Timer

- Timer 실습 (0.5초 만들기)

- **HAL_TIM_Base_Start_IT**(Timer Instance)
- 해당 타이머를 시작하고, 인터럽트를 활성화 시킨다.
- Main.c의 USER CODE BEGIN 2 밑에 해당 코드를 입력한다.

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_TIM1_Init();
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim1);
/* USER CODE END 2 */
```

3. Timer

- Timer 실습 (0.5초 만들기)

- Stm32f4xx_it.c 파일을 보면 TIM1 인터럽트에 해당하는 함수를 찾을 수 있다.
- 매 0.5초마다 해당 함수가 실행된다.
- HAL_TIM_IRQHandler를 오른쪽 클릭하여 함수가 정의된 곳으로 가보자.

```
EST
Files
TEST - TEST
├── Application
│   ├── EWARM
│   └── User
│       ├── main.c
│       ├── stm32f4xx_hal_m...
│       └── stm32f4xx_it.c
├── Drivers
│   ├── CMSIS
│   └── STM32F4xx_HAL_...
├── Output
│   ├── TEST.map
│   └── TEST.out
└── ...

/*****
 * STM32F4xx Peripheral Interrupt Handlers
 * Add here the Interrupt Handlers for the used peripherals.
 * For the available peripheral interrupt handler names,
 * please refer to the startup file (startup_stm32f4xx.s).
 *****/

/**
 * @brief This function handles TIM1 update interrupt and TIM10
 */
void TIM1_UP_TIM10_IRQHandler(void)
{
    /* USER CODE BEGIN TIM1_UP_TIM10_IRQn 0 */

    /* USER CODE END TIM1_UP_TIM10_IRQn 0 */
    HAL_TIM_IRQHandler(&htim1);
    /* USER CODE BEGIN TIM1_UP_TIM10_IRQn 1 */

    /* USER CODE END TIM1_UP_TIM10_IRQn 1 */
}

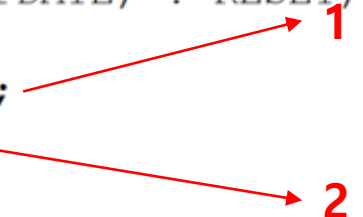
/* USER CODE BEGIN 1 */
```

3. Timer

- Timer 실습 (0.5초 만들기)

- HAL_TIM_URQHandler 함수를 살펴보면, 우리가 원하는 Update event에 대한 부분이 있다.
- 앞에서 봤던 인터럽트와 같이 먼저 flag를 클리어 한 후, 콜백 함수로 들어간다.
- HAL_TIM_PeriodElapsedCallback 함수를 오른쪽 클릭하여 함수가 정의된 부분으로 들어가면 역시 이 함수도 `__weak`로 정의되어 있음을 볼 수 있다.

```
    }  
}  
/* TIM Update event */  
if( __HAL_TIM_GET_FLAG(htim, TIM_FLAG_UPDATE) != RESET)  
{  
    if( __HAL_TIM_GET_IT_SOURCE(htim, TIM_IT_UPDATE) !=RESET)  
    {  
        __HAL_TIM_CLEAR_IT(htim, TIM_IT_UPDATE);  
        HAL_TIM_PeriodElapsedCallback(htim);  
    }  
}
```



3. Timer

- Timer 실습 (0.5초 만들기)

- 해당 함수 명을 복사하여, main.c 적당한 곳에 함수를 정의하고, 아래와 같이 써준다.
- 이제 프로그램을 업로드 하고 실행하면, 보드위의 녹색 LED가 0.5초 주기로 깜빡이는 것을 볼 수 있다.

```
/* Private user code -----*/  
/* USER CODE BEGIN 0 */  
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){  
    if(htim == &htim1){  
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);  
    }  
}  
/* USER CODE END 0 */  
  
/**  
 * @brief The application entry point.  
 * @retval int  
 */  
int main(void)
```

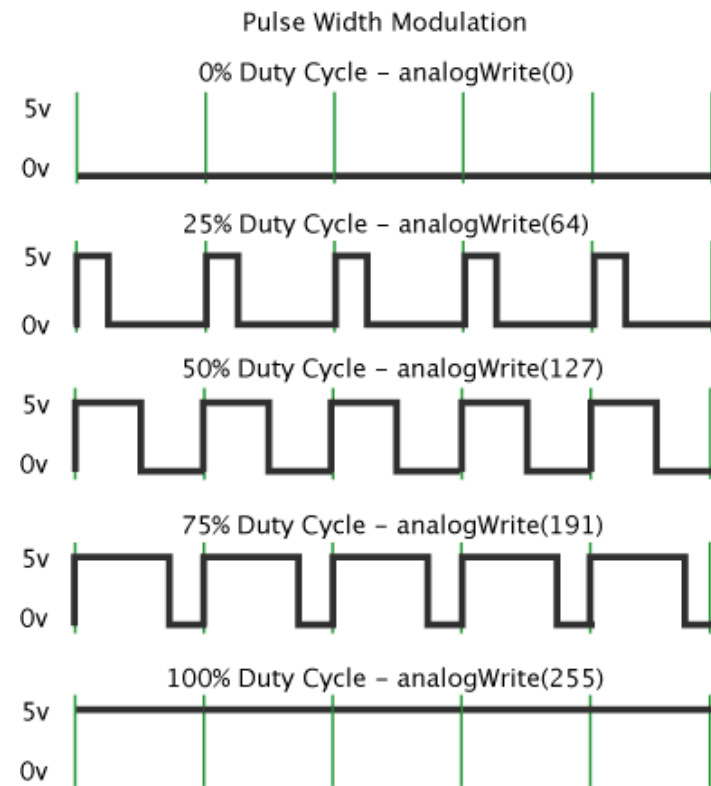
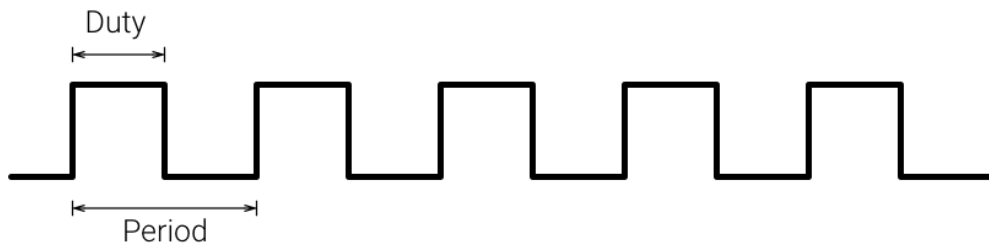
4. PWM

(Pulse Width Modulation)

4. PWM

- PWM 개요

- **PWM**이란 Pulse Width Modulation의 준말로, 한 주기 동안 High인 시간과 Low인 시간을 조절하여 파형을 생성해 내는 방법이다.
- 아두이노의 **analogWrite()** 함수가 이러한 PWM을 생성하는 기능을 한다.



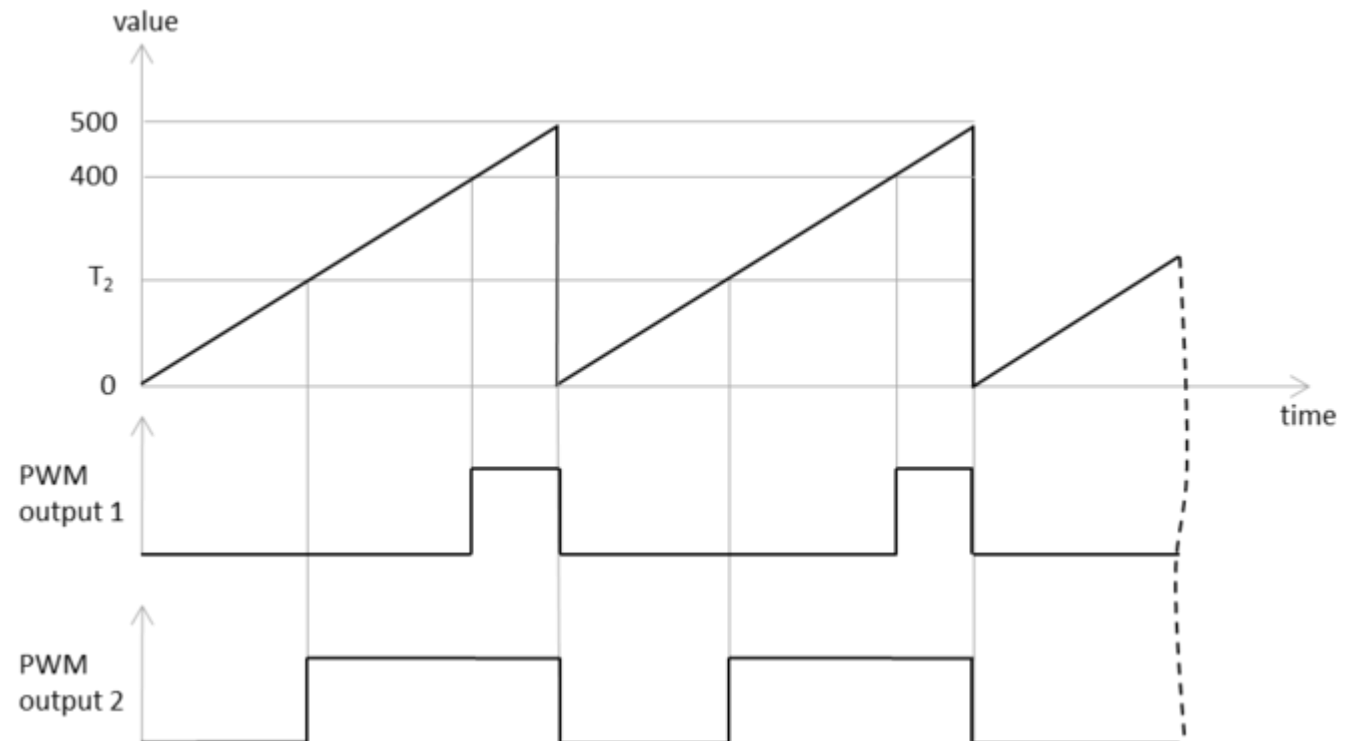
4. PWM

- PWM 개요

Q. PWM은 어떤 원리로 만들어내는가?

A.

- 타이머 카운터와의 비교를 통해 생성해 낼 수 있다.
- Polarity 세팅을 통해 PWM의 출력을 반전 시킬 수도 있다.



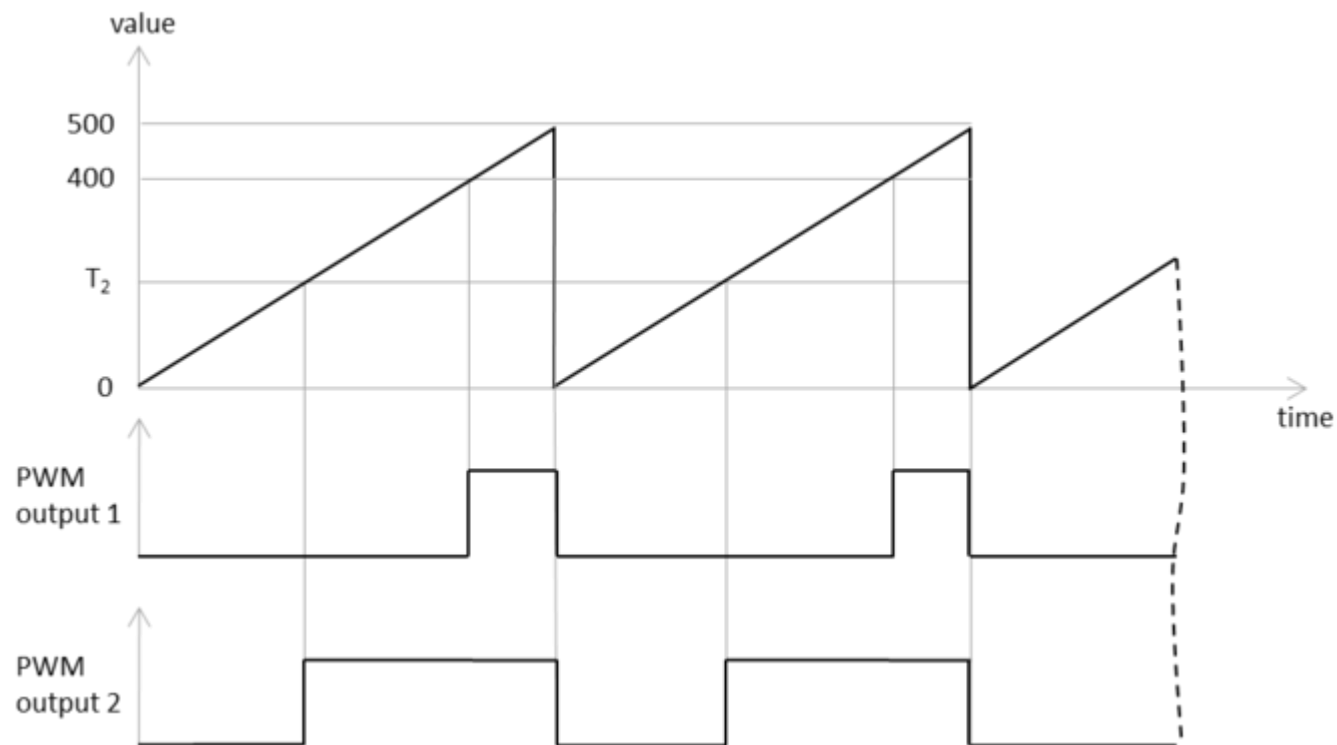
4. PWM

- PWM 개요

Q. PWM의 주기(Period)와 Duty는 어떻게 결정하는가?

A.

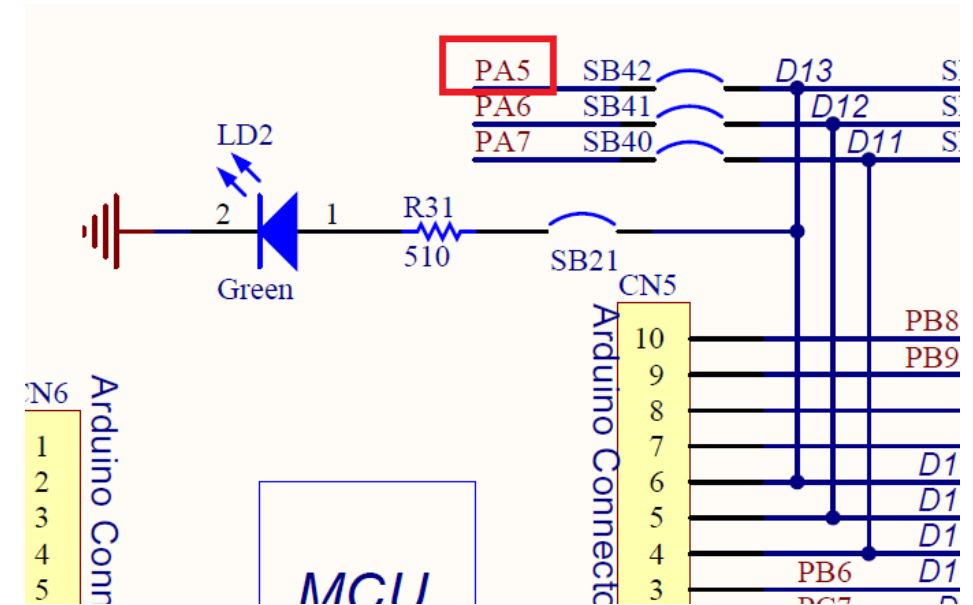
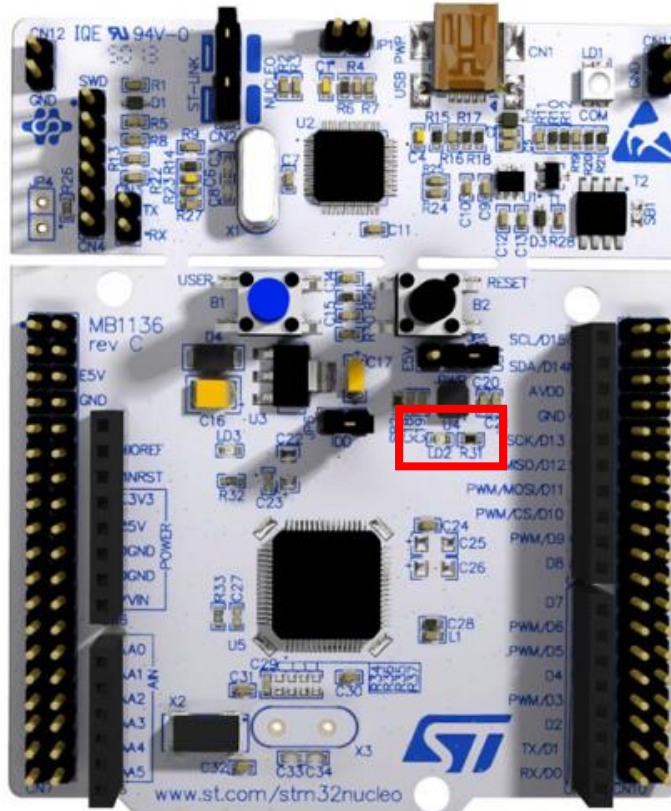
- PWM의 주기 = 타이머의 주기이다.
- PWM의 Duty는 타이머 카운터(CNT)의 비교값(CCR : Counter Compare Register)을 조절해서 바꿀 수 있다.



4. PWM

- PWM 실습(LED 밝기 제어)

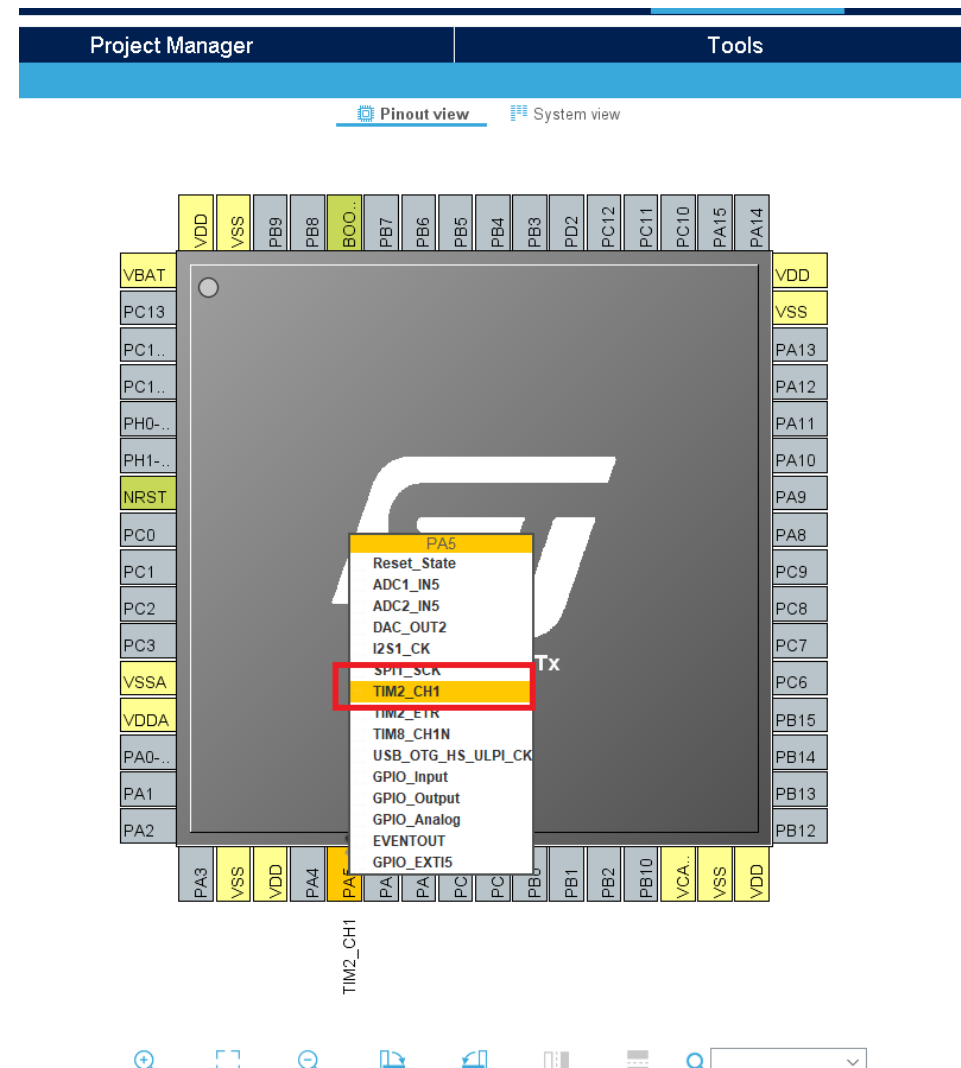
- 누클레오 보드 중앙에 있는 LD2 Led는 내부적으로 칩의 PA5 핀에 연결되어 있다.
- 1kHz의 PWM을 PA5로 출력하고, Duty를 조절하여 밝기를 조절해 보자.
- 현재 STM32의 클럭 속도는 **180MHz** 이다.



4. PWM

- PWM 실습(LED 밝기 제어)

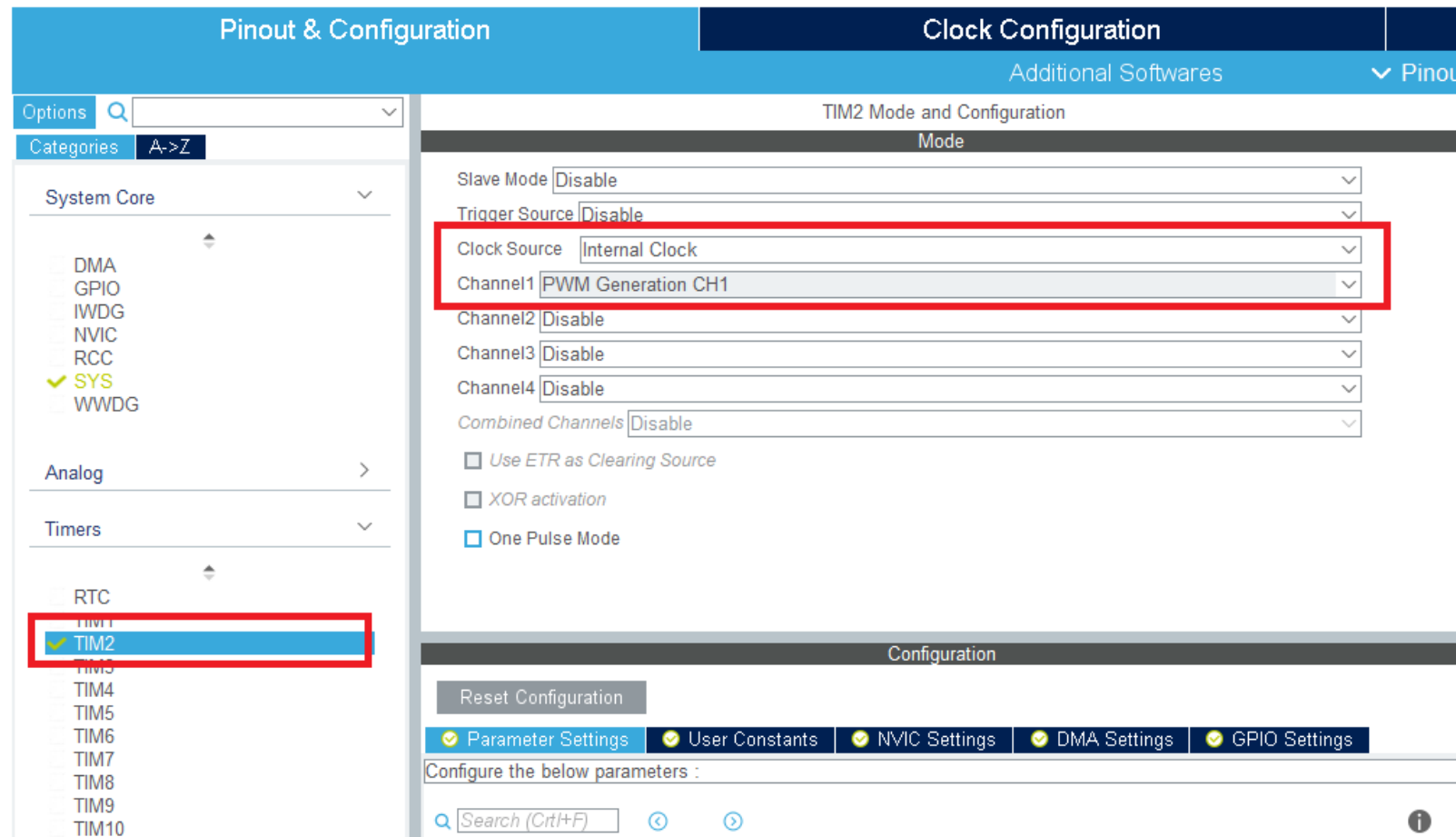
- CubeMX에 PA5를 TIM2_CH1로 설정한다.
- 이를 통해 PA5는 TIM2 Peripheral과 연결될 수 있음을 알 수 있다.



4. PWM

- PWM 실습(LED 밝기 제어)

- CubeMX에서 TIM2를 클릭하고, Internal Clock선택 및 Channel1을 PWM Generation으로 설정한다.



Q. TIM2로 1kHz를 만들기 위해서는 Prescaler와 Counter Period를 얼마로 해야 하는가?

4. PWM

- PWM 실습(LED 밝기 제어)

Q. TIM2로 1kHz를 만들기 위해서는 Prescaler와 Counter Period를 얼마로 해야 하는가?

[참고 1]

Table 6. Timer feature comparison

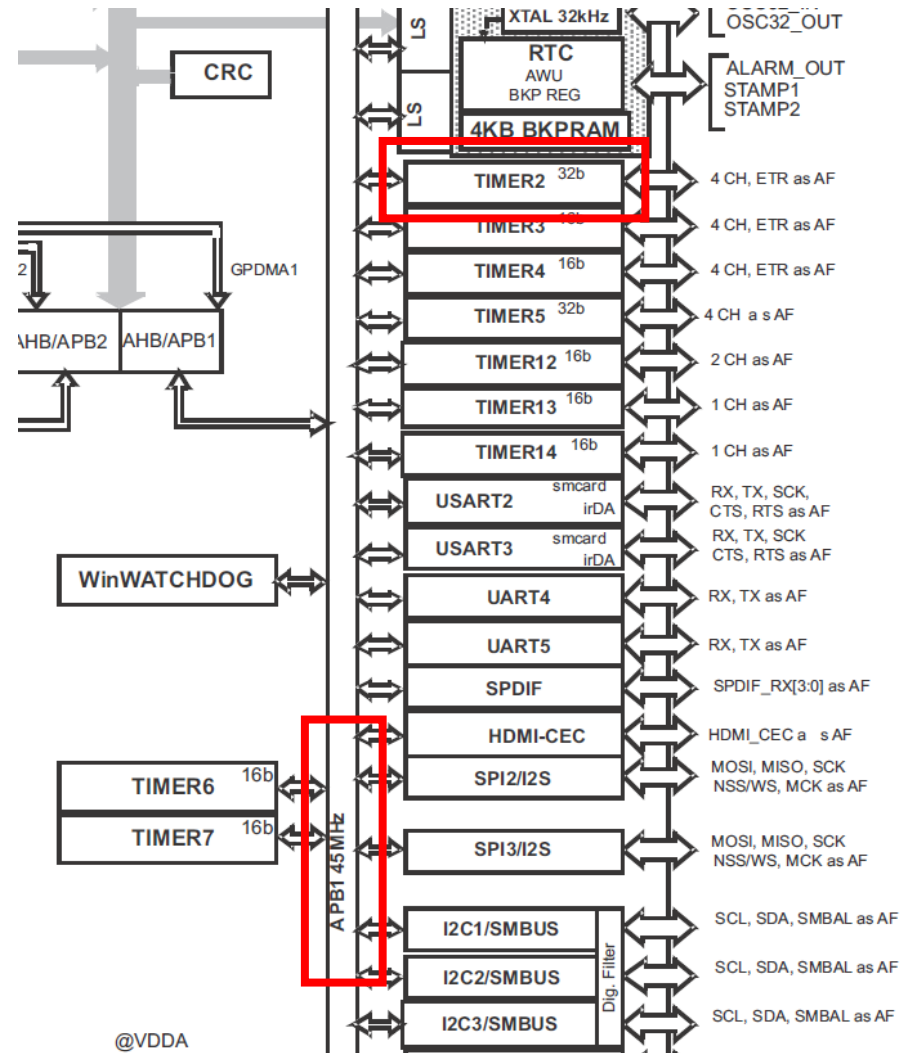
| Timer type | Timer | Counter resolution | Counter type | Prescaler factor | DMA request generation | Capture/compare channels | Complementary output | Max interface clock (MHz) | Max timer clock (MHz) ⁽¹⁾ |
|------------------|--------------|--------------------|-------------------|---------------------------------|------------------------|--------------------------|----------------------|---------------------------|--------------------------------------|
| Advanced-control | TIM1, TIM8 | 16-bit | Up, Down, Up/down | Any integer between 1 and 65536 | Yes | 4 | Yes | 90 | 180 |
| General purpose | TIM2, TIM5 | 32-bit | Up, Down, Up/down | Any integer between 1 and 65536 | Yes | 4 | No | 45 | 90/180 |
| | TIM3, TIM4 | 16-bit | Up, Down, Up/down | Any integer between 1 and 65536 | Yes | 4 | No | 45 | 90/180 |
| | TIM9 | 16-bit | Up | Any integer between 1 and 65536 | No | 2 | No | 90 | 180 |
| | TIM10, TIM11 | 16-bit | Up | Any integer between 1 and 65536 | No | 1 | No | 90 | 180 |
| | TIM12 | 16-bit | Up | Any integer between 1 and 65536 | No | 2 | No | 45 | 90/180 |
| | TIM13, TIM14 | 16-bit | Up | Any integer between 1 and 65536 | No | 1 | No | 45 | 90/180 |
| Basic | TIM6, TIM7 | 16-bit | Up | Any integer between 1 and 65536 | Yes | 0 | No | 45 | 90/180 |

4. PWM

- PWM 실습(LED 밝기 제어)

Q. TIM2로 1kHz를 만들기 위해서는 Prescaler와 Counter Period를 얼마로 해야 하는가?

[참고 2]

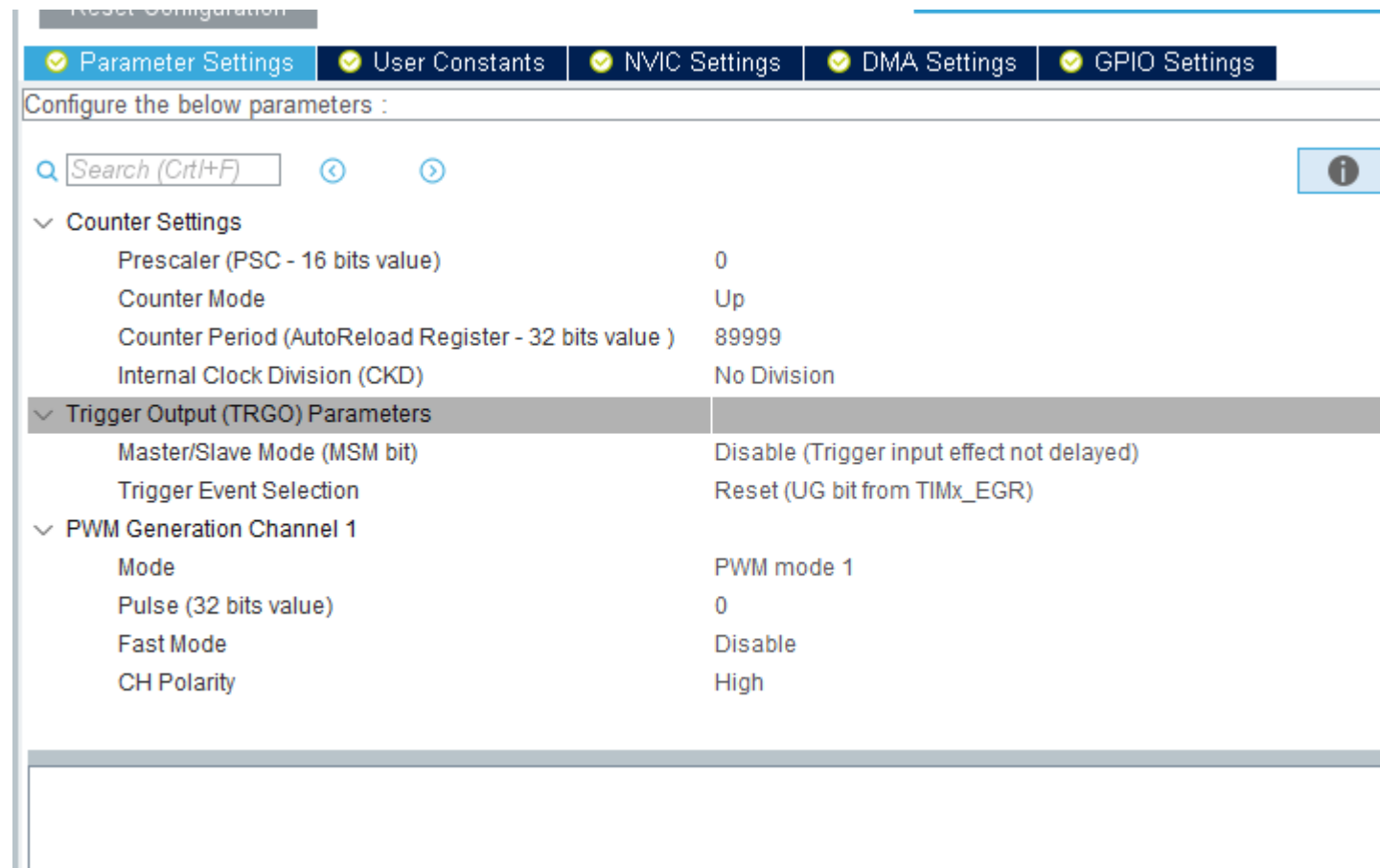


참고1_stm32f446RE.pdf (16page)

4. PWM

- PWM 실습(LED 밝기 제어)

- TIM2 설정을 다음과 같이 하고 코드를 생성한다.



Reset Configuration

✓ Parameter Settings ✓ User Constants ✓ NVIC Settings ✓ DMA Settings ✓ GPIO Settings

Configure the below parameters :

Search (Ctrl+F) ◀ ▶ ⓘ

▼ Counter Settings

| | |
|---|-------------|
| Prescaler (PSC - 16 bits value) | 0 |
| Counter Mode | Up |
| Counter Period (AutoReload Register - 32 bits value) | 89999 |
| Internal Clock Division (CKD) | No Division |

▼ Trigger Output (TRGO) Parameters

| | |
|-----------------------------|--|
| Master/Slave Mode (MSM bit) | Disable (Trigger input effect not delayed) |
| Trigger Event Selection | Reset (UG bit from TIMx_EGR) |

▼ PWM Generation Channel 1

| | |
|-----------------------|------------|
| Mode | PWM mode 1 |
| Pulse (32 bits value) | 0 |
| Fast Mode | Disable |
| CH Polarity | High |

4. PWM

- PWM 실습(LED 밝기 제어)

- **HAL_TIM_PWM_Start**(Timer Instance, Timer channel)
- 특정 타이머의 특정 채널의 PWM을 시작한다.
- Main.c의 USER CODE BEGIN 2 밑에 해당 코드를 입력한다.

```
/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_TIM2_Init();
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start(&htim2);
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
```

4. PWM

- PWM 실습(LED 밝기 제어)

- Main.c 파일 상단에 듀티 값을 저장할 uint32형 변수를 하나 선언한다.

```
/* USER CODE END PFP */  
  
/* Private user code -----  
/* USER CODE BEGIN 0 */  
uint32_t tim2_ccr1;  
/* USER CODE END 0 */
```

- While문에 아래와 같은 코드를 작성한 후, 프로그램을 업로드 한다.

```
/* Infinite loop */  
/* USER CODE BEGIN WHILE */  
while (1)  
{  
    tim2_ccr1 += 1000;  
    htim2.Instance->CCR1 = tim2_ccr1;  
    HAL_Delay(10);  
  
    if(tim2_ccr1 > 90000) tim2_ccr1 = 0;  
  
    /* USER CODE END WHILE */  
  
    /* USER CODE BEGIN 3 */
```