TI DSP, MCU, Xilinx Zynq FPGA
프로그래밍 전문가 과정

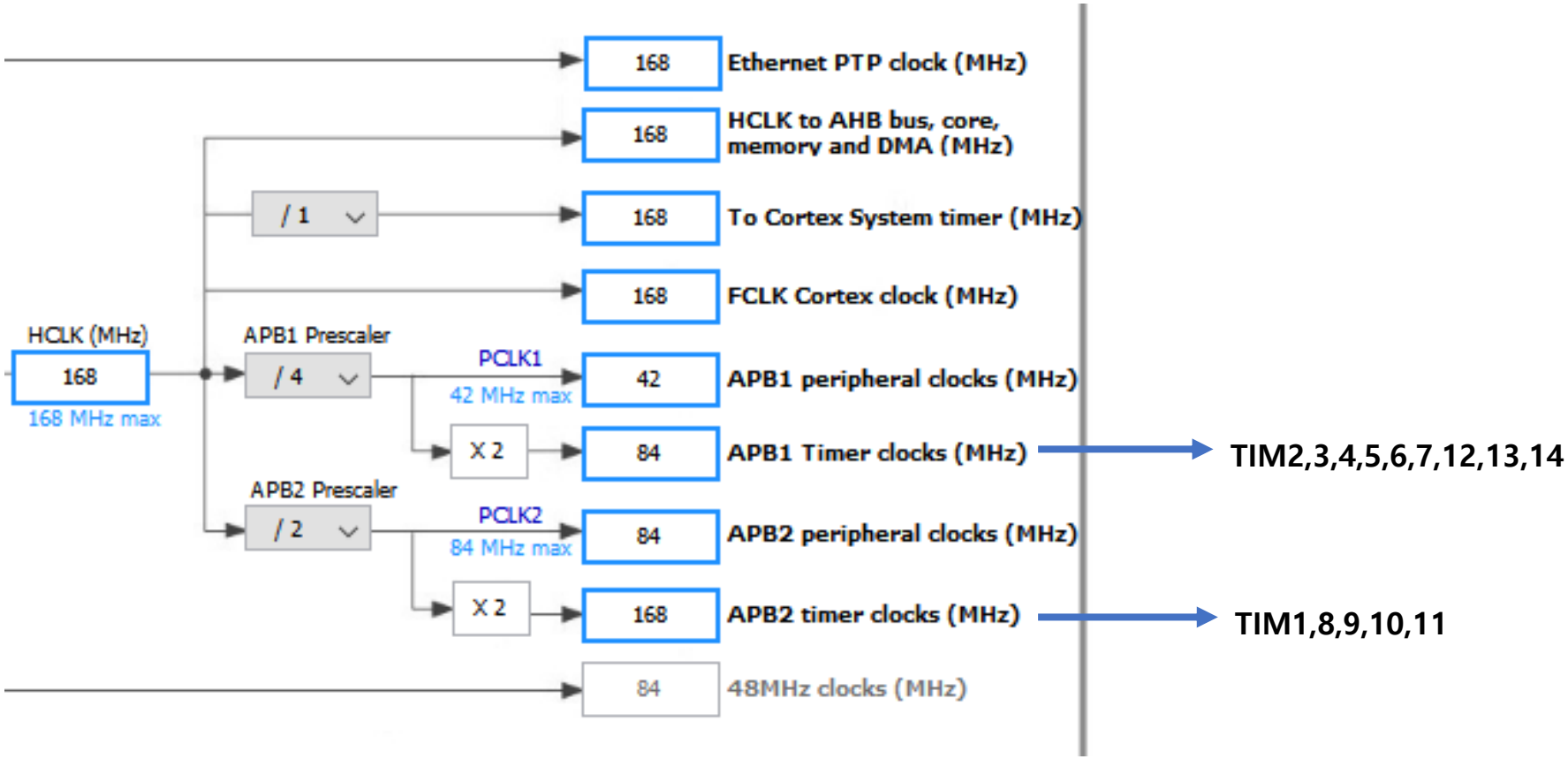
TIMER & PWM control with STM32F407

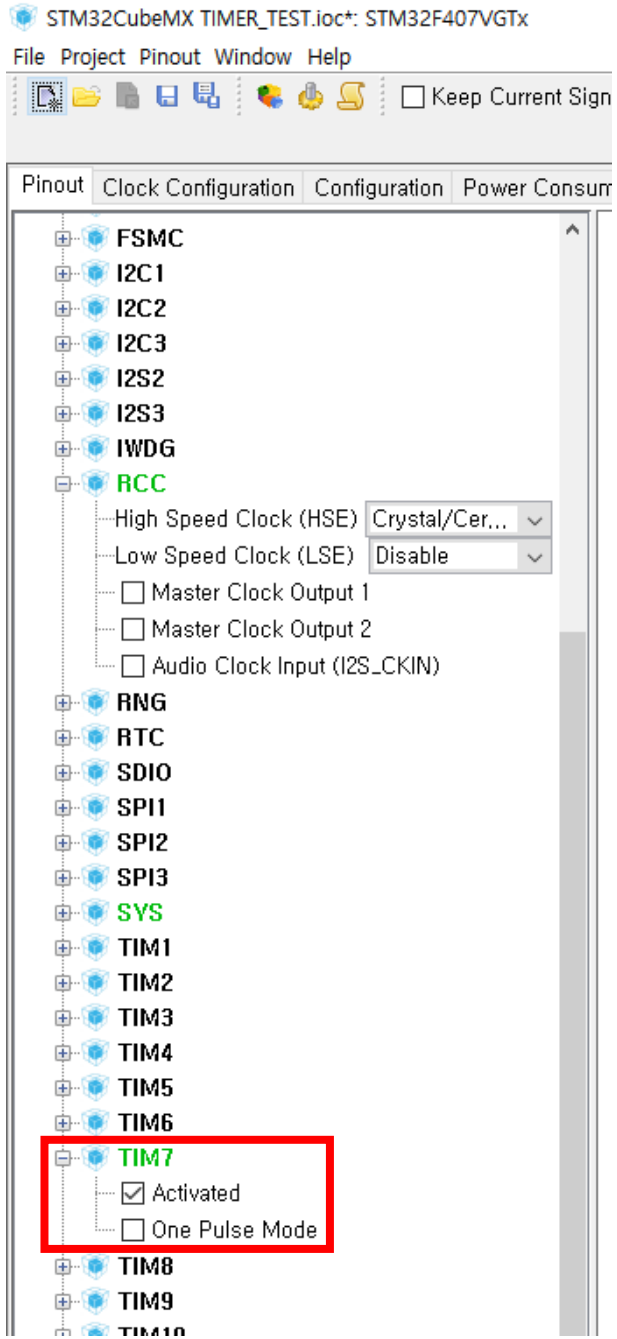
강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – 안상재
sangjae2015@naver.com

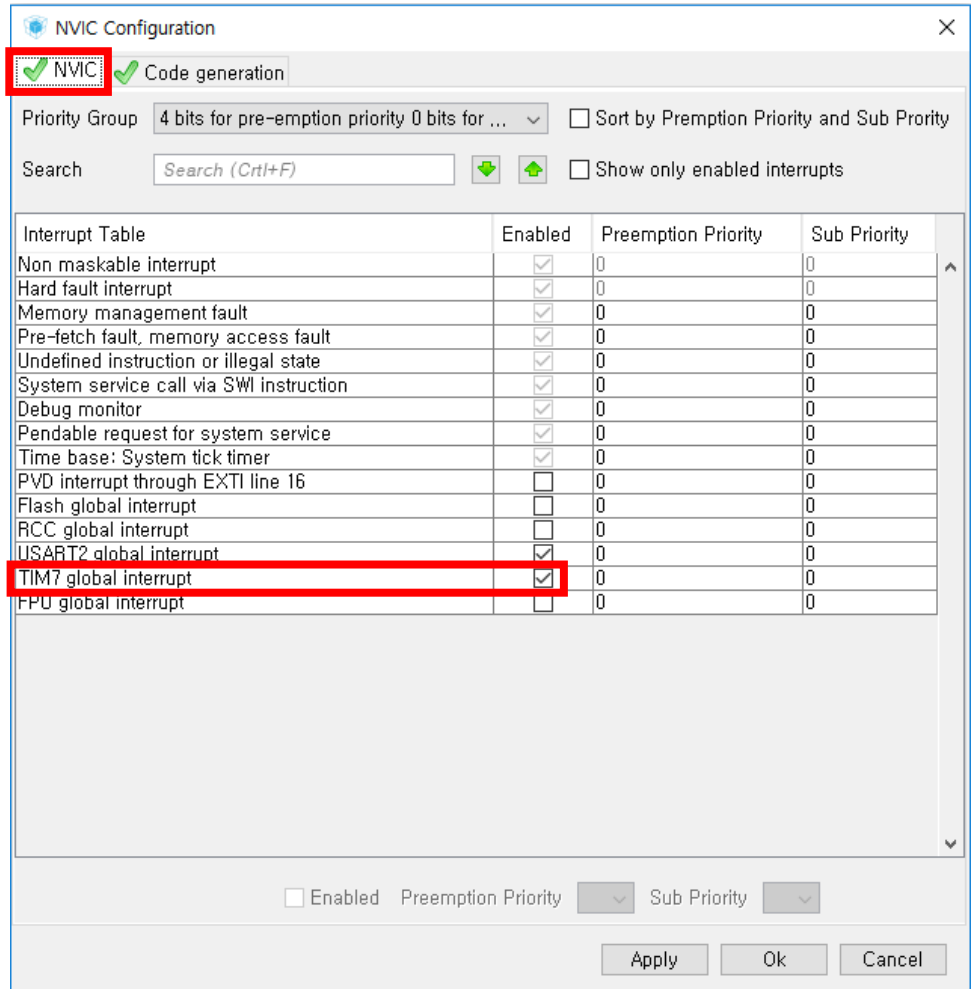
1. TIMER

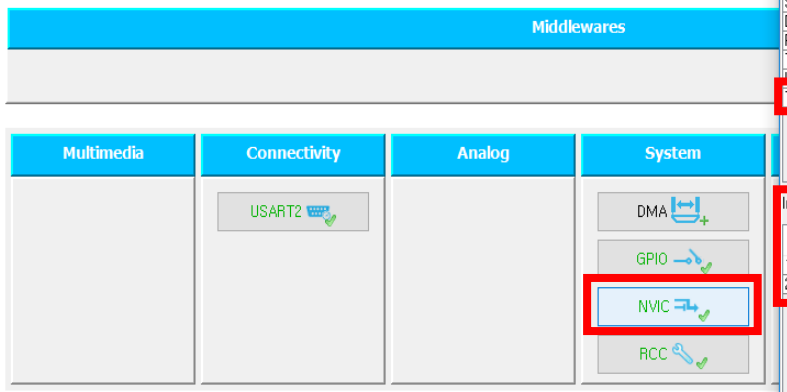
- Clock Tree

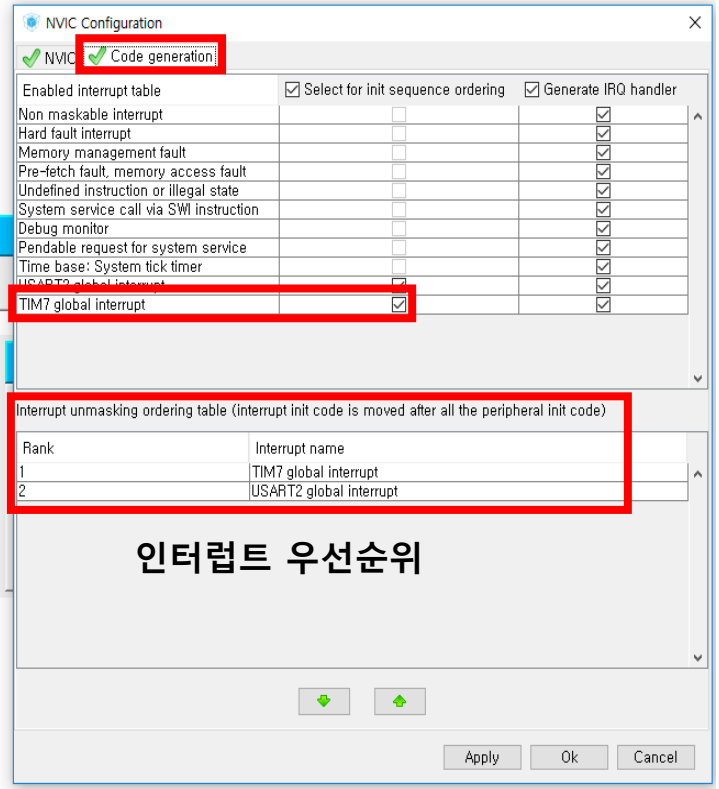



STM32CubeMX TIMER_TEST.ioc*: STM32F407VGTx
File Project Pinout Window Help
Keep Current Sign
Pinout
Clock Configuration
Configuration
Power Consum
FSMC
I2C1
I2C2
I2C3
I2S2
I2S3
IWDG
RCC
High Speed Clock (HSE)
Crystal/Cer...
Low Speed Clock (LSE)
Disable
Master Clock Output 1
Master Clock Output 2
Audio Clock Input (I2S_CKIN)
RNG
RTC
SDIO
SPI1
SPI2
SPI3
SYS
TIM1
TIM2
TIM3
TIM4
TIM5
TIM6
TIM7
Activated
One Pulse Mode
TIM8
TIM9

- CubeMX 설정


NVIC Configuration
NVIC
Code generation
Priority Group
4 bits for pre-emption priority 0 bits for ...
Sort by Premption Priority and Sub Prority
Search
Search (Crtl+F)
Show only enabled interrupts
Interrupt Table
Enabled
Preemption Priority
Sub Priority
Non maskable interrupt
Hard fault interrupt
Memory management fault
Pre-fetch fault, memory access fault
Undefined instruction or illegal state
System service call via SWI instruction
Debug monitor
Pendable request for system service
Time base: System tick timer
PVD interrupt through EXTI line 16
Flash global interrupt
RCC global interrupt
USART2 global interrupt
TIM7 global interrupt
FPU global interrupt
Enabled
Preemption Priority
Sub Priority
Apply
Ok
Cancel


Middlewares
Multimedia
Connectivity
Analog
System
USART2
DMA
GPIO
NVIC
RCC

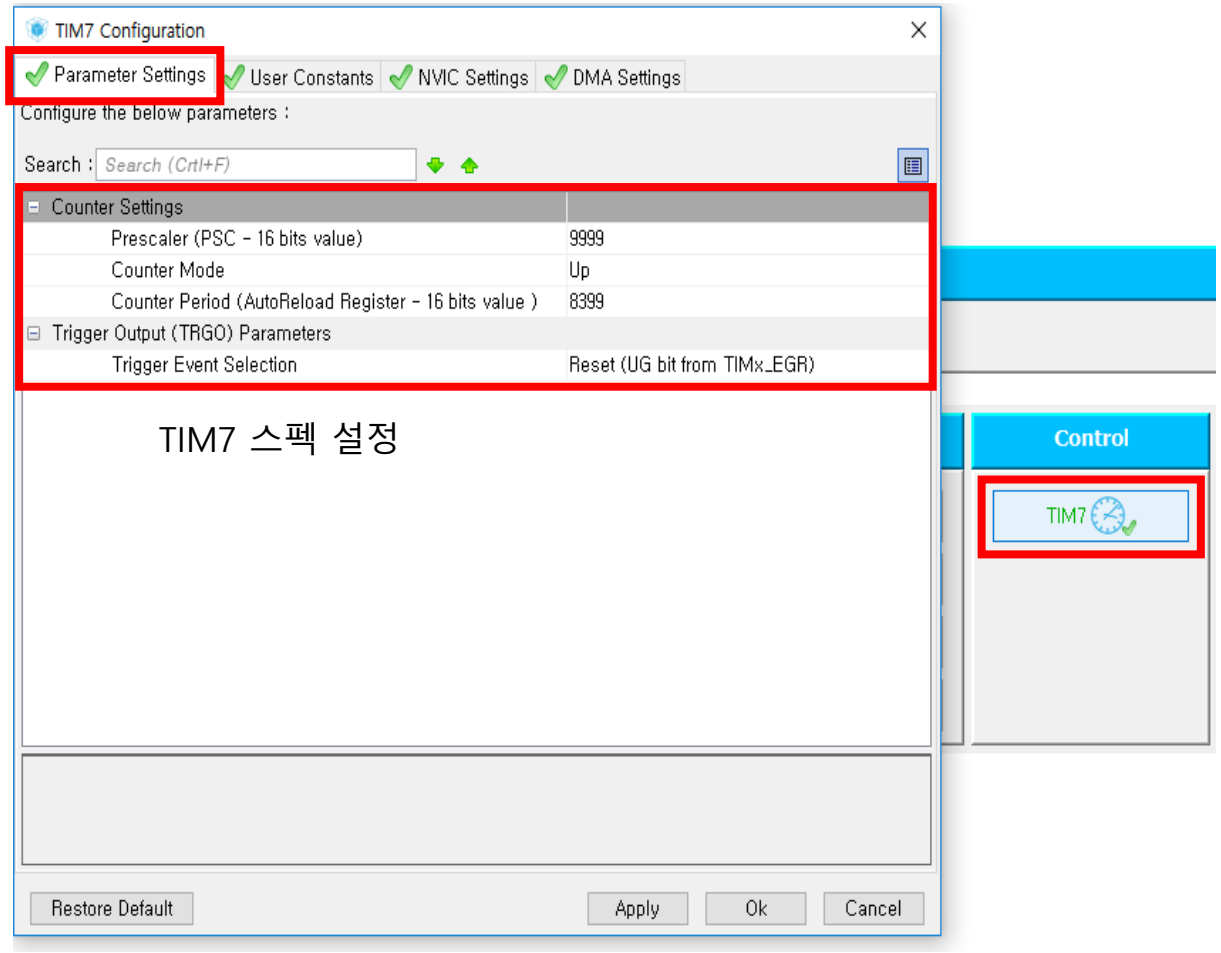

NVIC Configuration
NVIC
Code generation
Enabled interrupt table
Select for init sequence ordering
Generate IRQ handler
Non maskable interrupt
Hard fault interrupt
Memory management fault
Pre-fetch fault, memory access fault
Undefined instruction or illegal state
System service call via SWI instruction
Debug monitor
Pendable request for system service
Time base: System tick timer
TIM7 global interrupt
Interrupt unmasking ordering table (interrupt init code is moved after all the peripheral init code)
Rank
Interrupt name
1
TIM7 global interrupt
2
USART2 global interrupt
인터럽트 우선순위
Apply
Ok
Cancel

1초 만들기

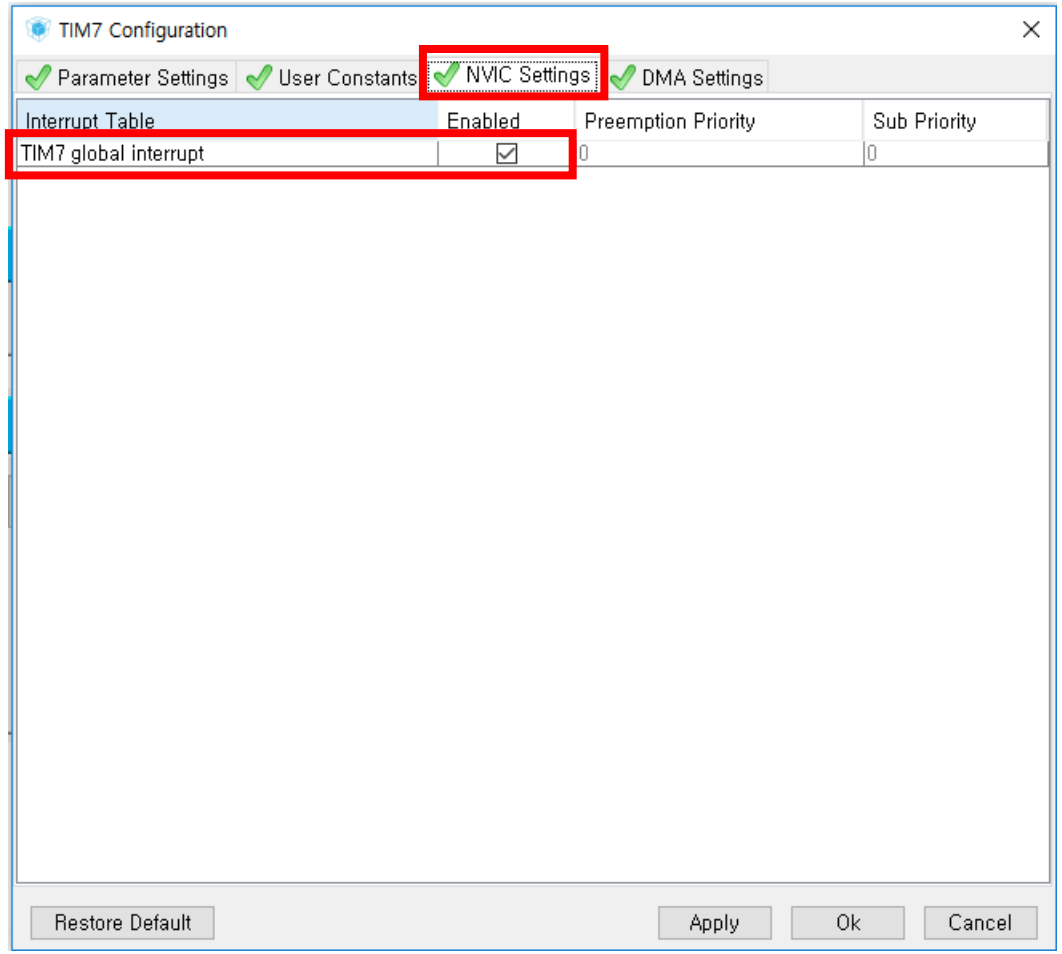
=> TIM7 은 84MHZ 이므로, prescaler 10000로 나누면 8400HZ가 됨.

=> Counter Period 8400 에서 분주된 주파수 (8400HZ) 를 나누면 1초가 됨!

* Prescaler 와 Counter Period는 원하는 값에서 1을 빼주어야함!



TIM7 스펙 설정



- 소스 코드

```
75 int main(void)
76 {
77   /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
78   HAL_Init();
79   /* Configure the system clock */
80   SystemClock_Config();
81
82   /* USER CODE BEGIN SysInit */
83
84   /* USER CODE END SysInit */
85
86   /* Initialize all configured peripherals */
87   MX_GPIO_Init();
88   MX_USART2_UART_Init();
89   MX_TIM7_Init();
90
91   /* Initialize interrupts */
92   MX_NVIC_Init();
93   /* USER CODE BEGIN 2 */
94   HAL_TIM_Base_Start_IT(&htim7);
95
96   /* USER CODE END 2 */
97
98   /* Infinite loop */
99   /* USER CODE BEGIN WHILE */
100  while (1)
101  {
102
103  }
104  /* USER CODE END 3 */
105 }
```

TIM7 초기 설정

인터럽트 초기 설정

TIM7 인터럽트 시작

```c
void MX_TIM7_Init(void)
{
  TIM_MasterConfigTypeDef sMasterConfig;

  htim7.Instance = TIM7;
  htim7.Init.Prescaler = 9999;
  htim7.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim7.Init.Period = 8399;
  if (HAL_TIM_Base_Init(&htim7) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim7, &sMasterConfig) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
}
```

```c
static void MX_NVIC_Init(void)
{
  /* TIM7_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(TIM7_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(TIM7_IRQn);
  /* USART2_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(USART2_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(USART2_IRQn);
}
```

```c
HAL_StatusTypeDef HAL_TIM_Base_Start_IT(TIM_HandleTypeDef *htim)
{
  /* Check the parameters */
  assert_param(IS_TIM_INSTANCE(htim->Instance));

  /* Enable the TIM Update interrupt */
  __HAL_TIM_ENABLE_IT(htim, TIM_IT_UPDATE);

  /* Enable the Peripheral */
  __HAL_TIM_ENABLE(htim);

  /* Return function status */
  return HAL_OK;
}
```

- 인터럽트 핸들러 부분

```c
void TIM7_IRQHandler(void)
{
  /* USER CODE BEGIN TIM7_IRQn 0 */

  /* USER CODE END TIM7_IRQn 0 */
  HAL_TIM_IRQHandler(&htim7);
  /* USER CODE BEGIN TIM7_IRQn 1 */

  /* USER CODE END TIM7_IRQn 1 */
}
```

```c
void HAL_TIM_IRQHandler(TIM_HandleTypeDef *htim)
{
  /* Capture compare 1 event */
  if(__HAL_TIM_GET_FLAG(htim, TIM_FLAG_CC1) != RESET)
  {
    if(__HAL_TIM_GET_IT_SOURCE(htim, TIM_IT_CC1) !=RESET)
    {
      {
        __HAL_TIM_CLEAR_IT(htim, TIM_IT_CC1);
        htim->Channel = HAL_TIM_ACTIVE_CHANNEL_1;

        /* Input capture event */
        if((htim->Instance->CCMR1 & TIM_CCMR1_CC1S) != 0x00U)
        {
          HAL_TIM_IC_CaptureCallback(htim);
        }
        /* Output compare event */
        else
        {
          HAL_TIM_OC_DelayElapsedCallback(htim);
          HAL_TIM_PWM_PulseFinishedCallback(htim);
        }
        htim->Channel = HAL_TIM_ACTIVE_CHANNEL_CLEARED;
      }
    }
  }
  /* Capture compare 2 event */
  if(__HAL_TIM_GET_FLAG(htim, TIM_FLAG_CC2) != RESET)
  {
    if(__HAL_TIM_GET_IT_SOURCE(htim, TIM_IT_CC2) !=RESET)
  /* 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);
    }
  }
  /* TIM Break input event */
  if(__HAL_TIM_GET_FLAG(htim, TIM_FLAG_BREAK) != RESET)
  {
    if(__HAL_TIM_GET_IT_SOURCE(htim, TIM_IT_BREAK) !=RESET)
    {
      __HAL_TIM_CLEAR_IT(htim, TIM_IT_BREAK);
      HAL_TIMEx_BreakCallback(htim);
    }
  }
```

```
__weak void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(htim);
  /* NOTE : This function Should not be modified, when the callback is needed,
            the __HAL_TIM_PeriodElapsedCallback could be implemented in the user file
   */
}
```

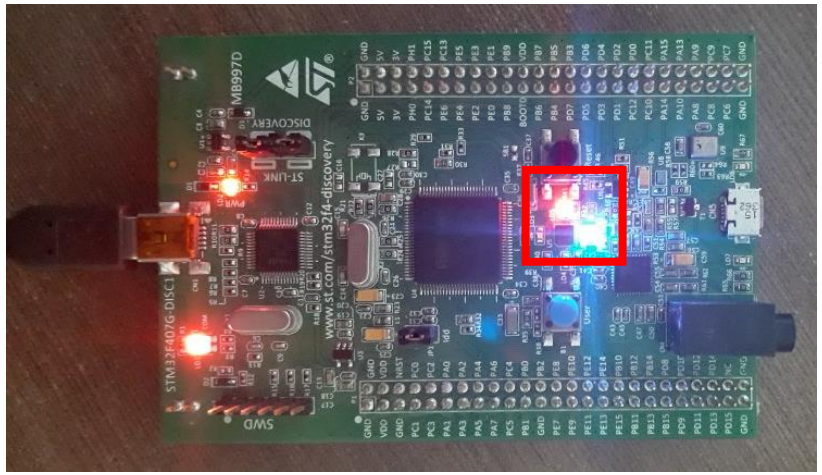
__weak 이 앞에 붙으면 원하는 대로 커스텀 하라는 뜻

main 문 아래에서 다시 정의 해줌!

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance == TIM7)
    {
        if(k%4 == 0)
            HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);
        else if(k%4 == 1)
            HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_13);
        else if(k%4 == 2)
            HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_14);
        else if(k%4 == 3)
            HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15);

        k++;
    }
}
```

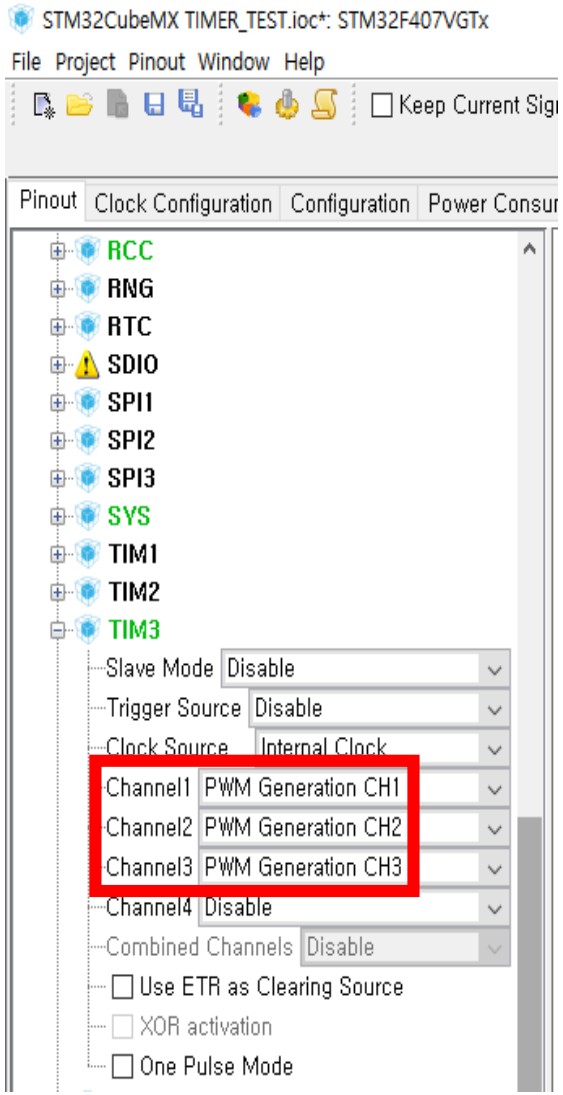
- 결과 사진

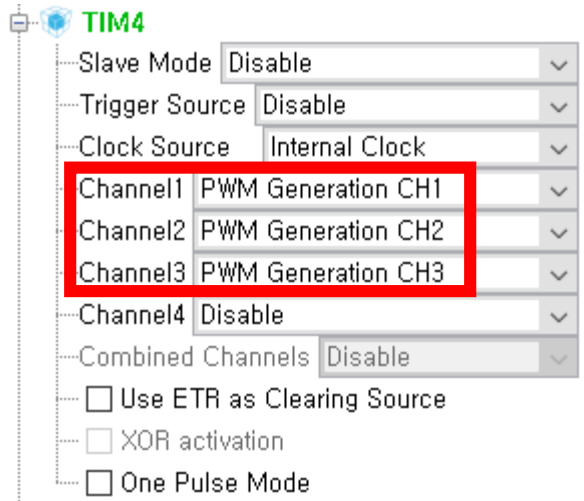


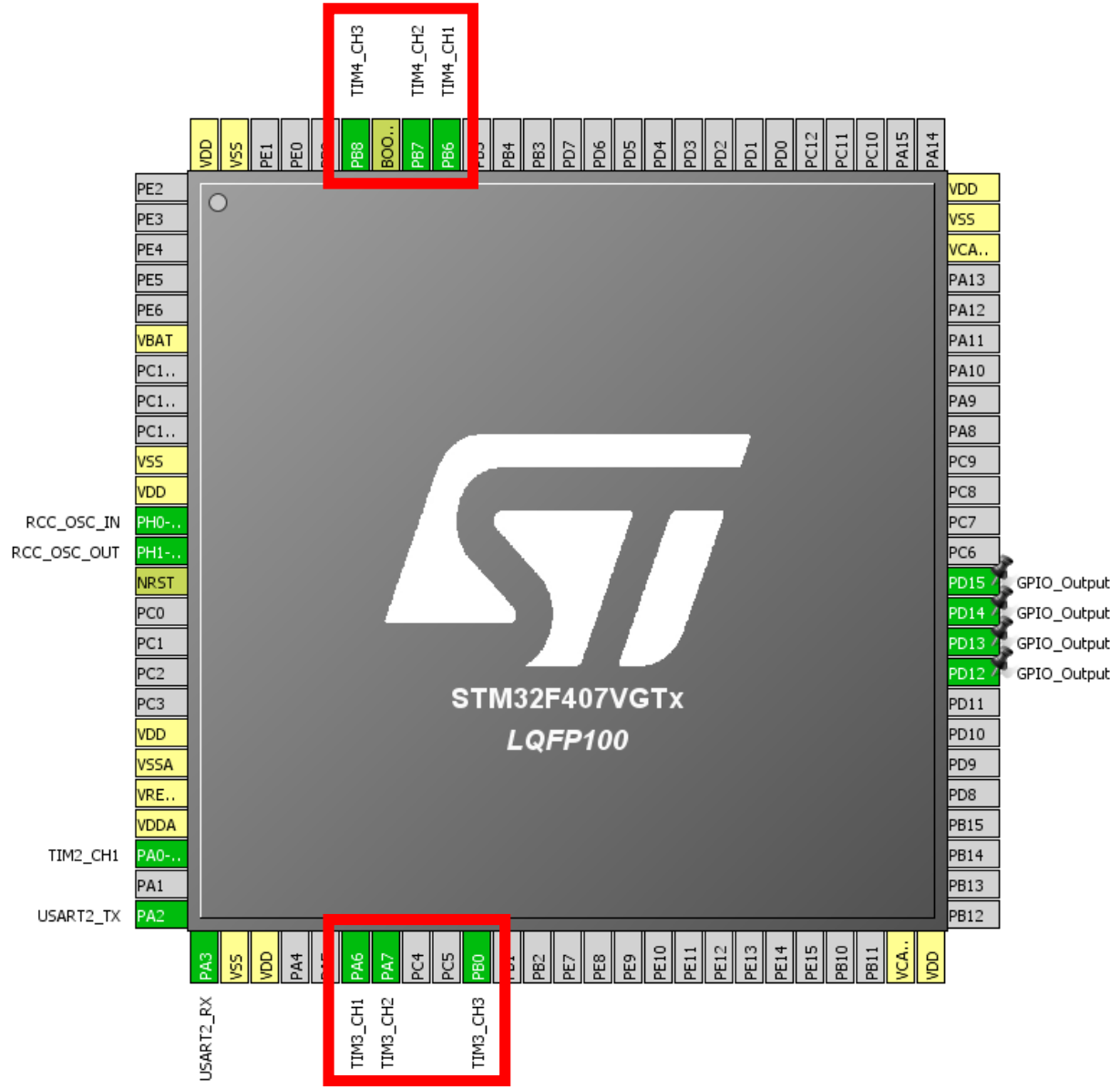
LED가 순차적으로 켜졌다 꺼짐

1. PWM

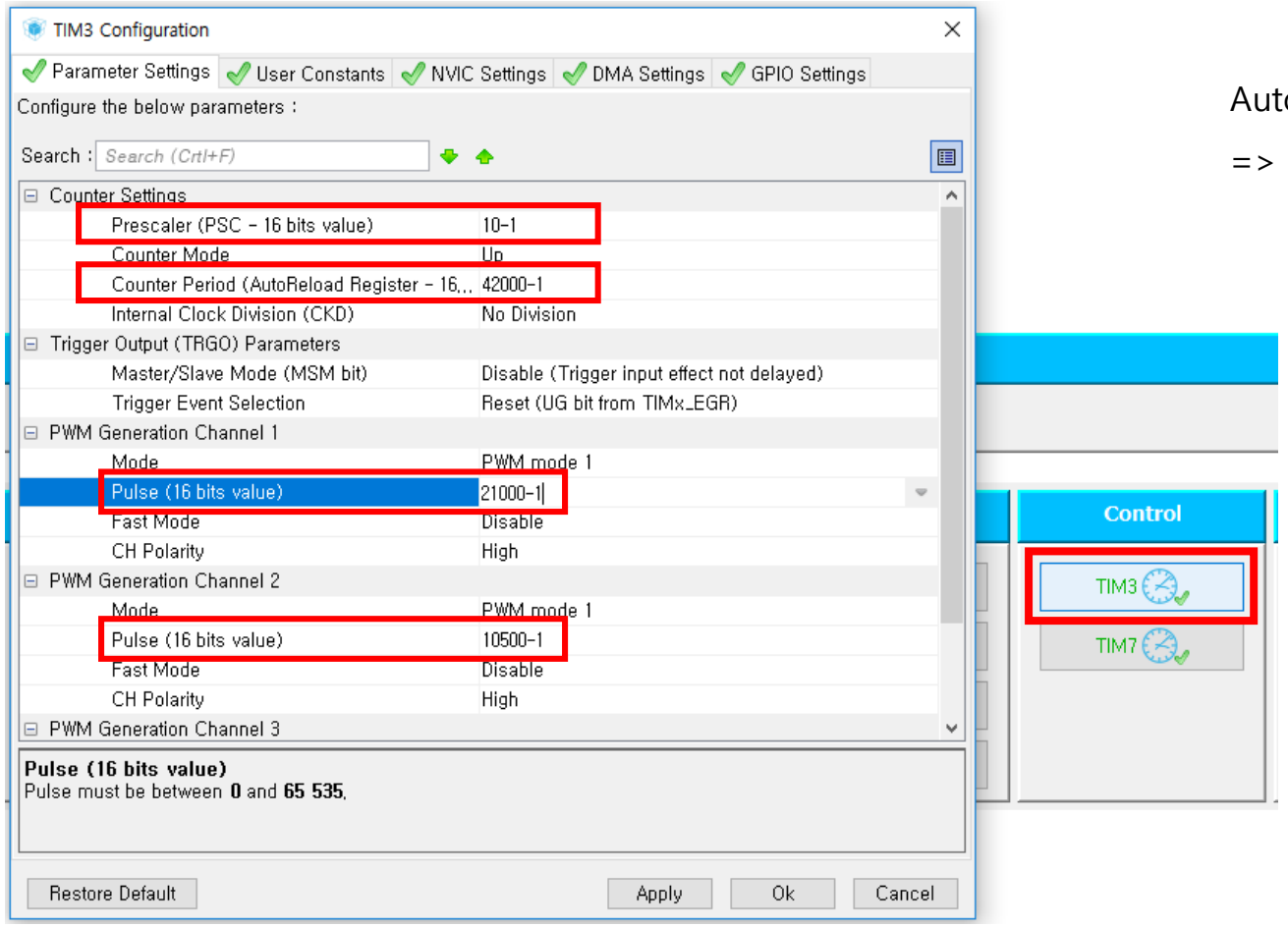
- CubeMX 설정

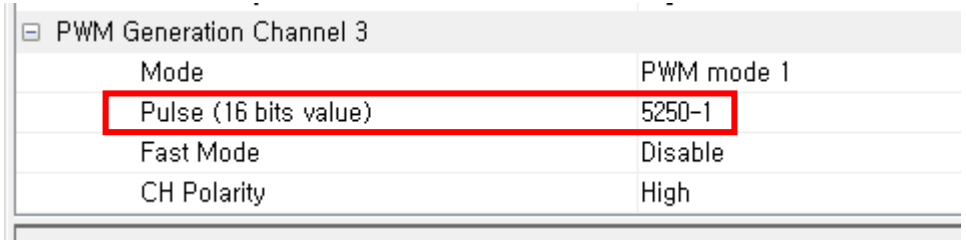






- TIMER3





AutoReload Register는 한 주기 동안의 카운터 갯수

=> AutoReload Register / 분주된 주파수 = 타이머의 주기

PWM1

=> 21000은 42000의 절반이므로 Duty비는 50%

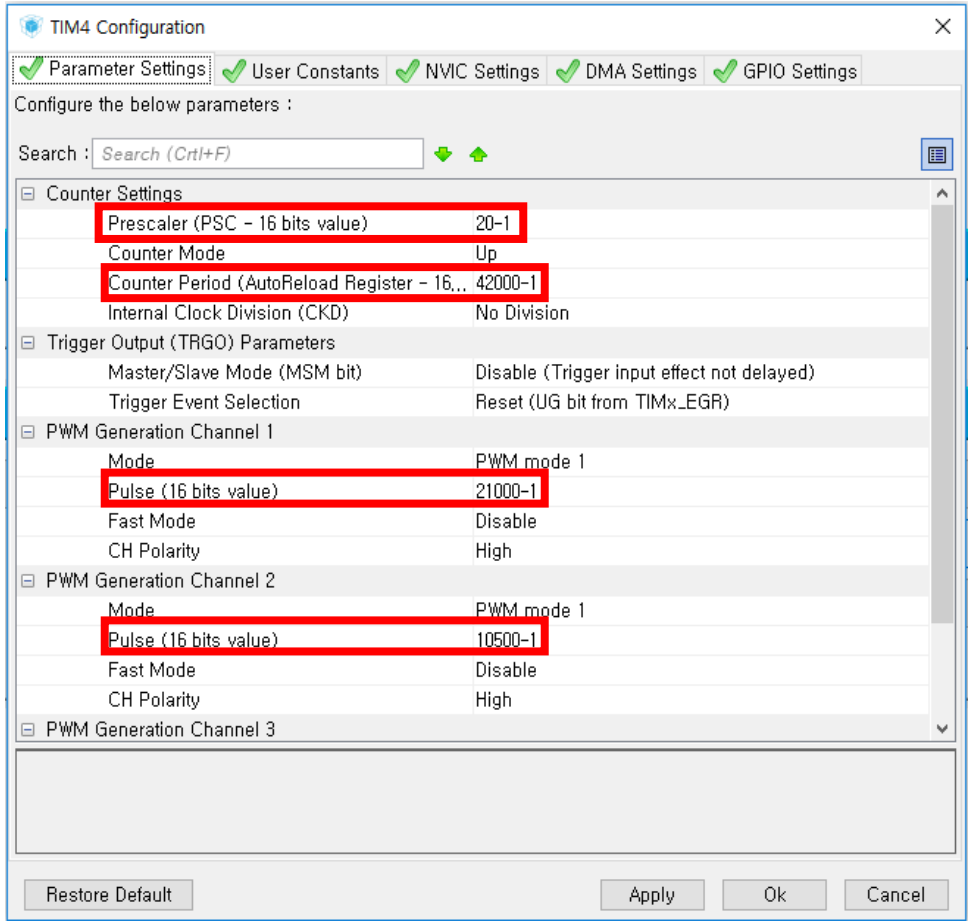
PWM2

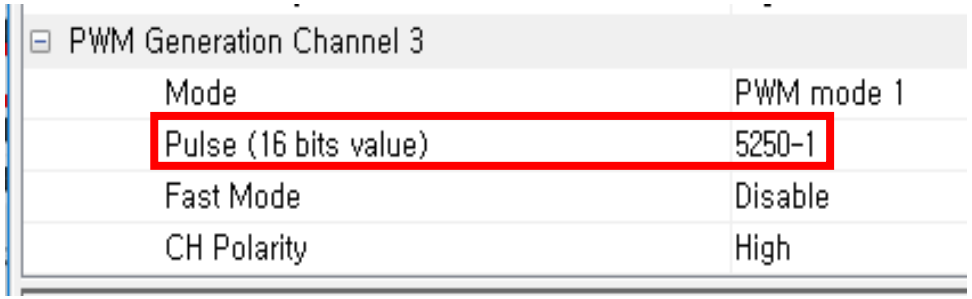
=> 10500은 42000의 ¼이므로 Duty비는 25%

PWM3

=> 5250은 42000의 1/8 이므로 Duty비는 12.5%

- TIMER4





- 분주된 주파수 : 84000000 / 20 = 4200000
- 주기 : 42000 / 4200000 = 10ms
- PWM1 Duty비 : 50%
- PWM2 Duty비 : 25%
- PWM3 Duty비 : 12.5%

- 소스 코드

```c
int main(void)
{
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_TIM3_Init();
  MX_TIM7_Init();
  MX_TIM4_Init();

  /* Initialize interrupts */
  MX_NVIC_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT(&htim7);

  HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);
  HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3);

  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2);
  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3);

  /* USER CODE END 2 */

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

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

TIMER 초기화

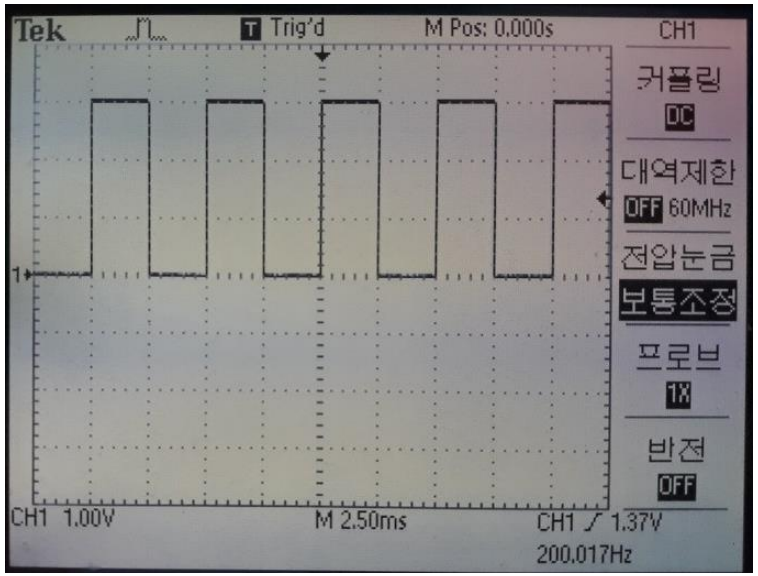
PWM 시작

- 결과 화면

TIMER3

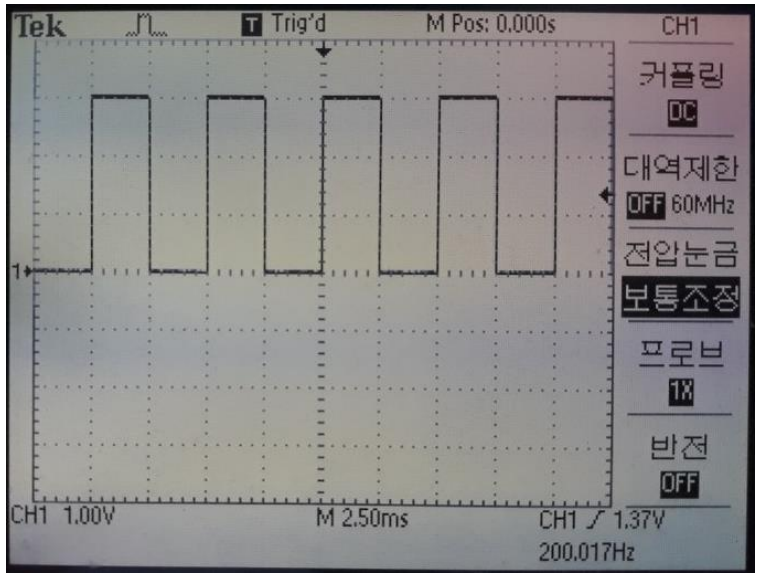
PWM1

- 주기 5ms, Duty 50%



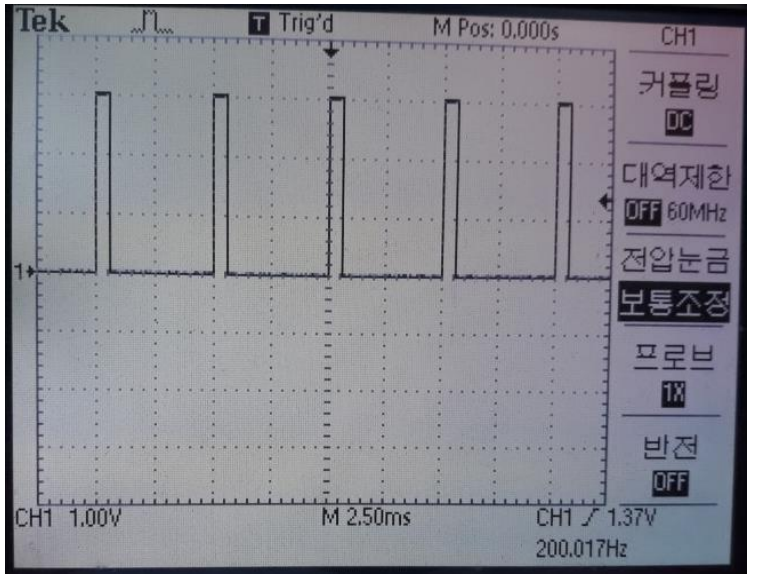
PWM2

- 주기 5ms, Duty 25%



PWM3

- 주기 5ms, Duty 12.5%



* 실시간 PWM Duty비 변경

- 소스 코드

```
  uint16_t ccr = 0;
  /* USER CODE END 2 */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
      __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, ccr); // 매크로 함수
      ccr += 1000;

      if(ccr > TIM4->ARR)
          ccr = 0;

      HAL_Delay(50);
  }
  /* USER CODE END 3 */
}
```

Duty비 조정(같은 코드) ⟷ `TIM4->CCR1 = ccr;`

Duty비 변화