피지컬 컴퓨팅

Lec. 7. Pulse Width Modulation (PWM)

Heenam Yoon

Department of
Human-Centered Artificial Intelligence

E-mail) h-yoon@smu.ac.kr
Room) 0112

Pulse Width Modulation (PWM)

배경

- GPIO를 output으로 사용하는 예시

  C 포트 0~7번 핀에 8개의 LED를 연결하여, LED를 켜고 끄겠다

```
while(1)
  {
    GPIO_Write(GPIOC, 0x0000);
    Delay(5000000);
    GPIO_Write(GPIOC, 0x00FF);
    Delay(5000000);
  }
```

LED 8개를 끈다

LED 8개를 켠다 = 전원을 공급한다

바이너리 (켠다/끈다)로 제어하고 있음
전원의 "크기"를 제어 할 수는 없음
= LED의 밝기를 조절할 수는 없음

진짜 못하는가? 하고 싶은데..

Pulse Width Modulation (PWM)

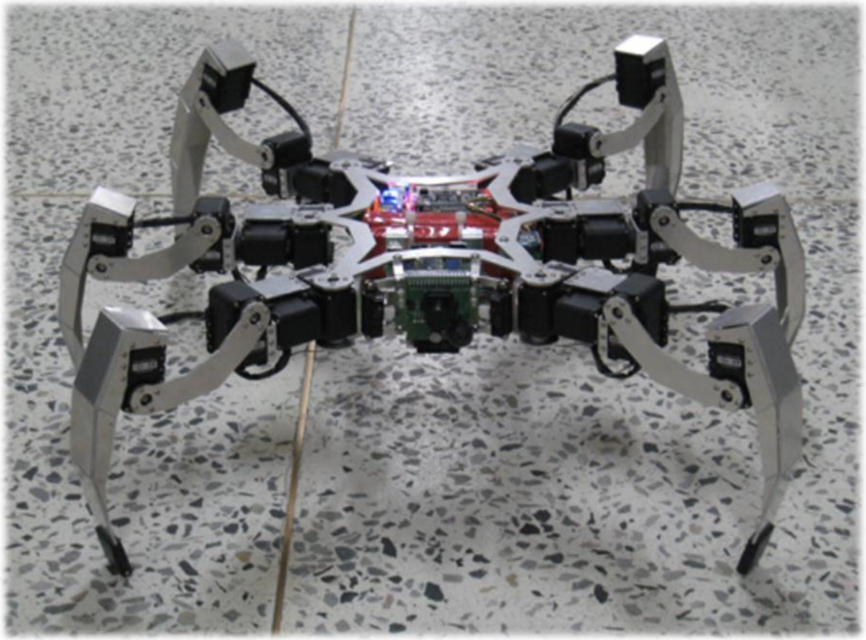
배경

한국지능로봇경진대회, https://www.youtube.com/watch?v=ZMGBnxAJWUc

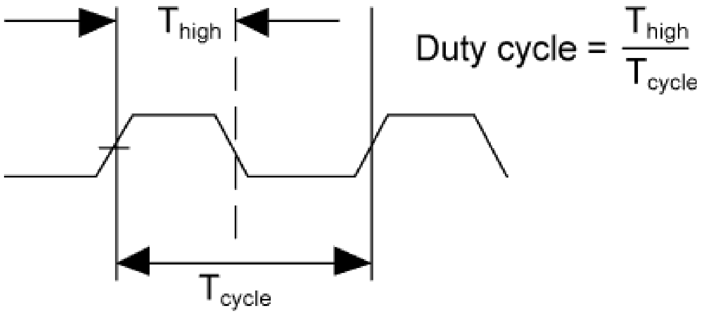
Pulse Width Modulation (PWM)

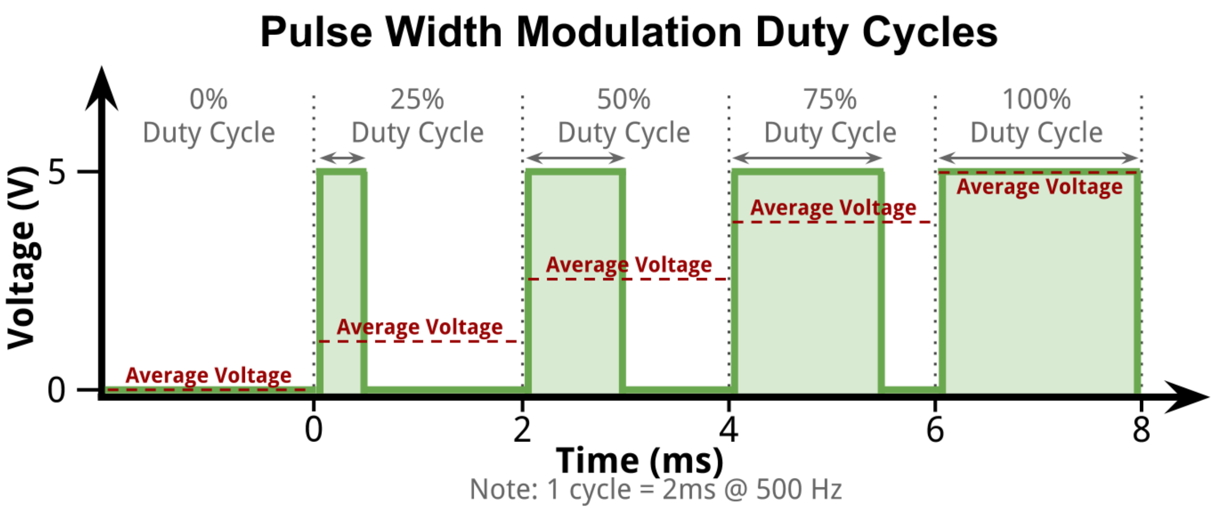
배경

- 움직임 제어는 각 모터를 제어하는 것이고, 모터가 특정 각도를 유지하도록 하는 것
- 보통 0 ~ 180도로 움직이는데, 특정 각도만큼 어떻게 움직이게 할 것인가?



Pulse Width Modulation (PWM)





Pulse Width Modulation (PWM)

- **TIMx_ARR : Period**
  Auto-reload register
- **TIMx_CCR : Duty**
  Capture/compare register

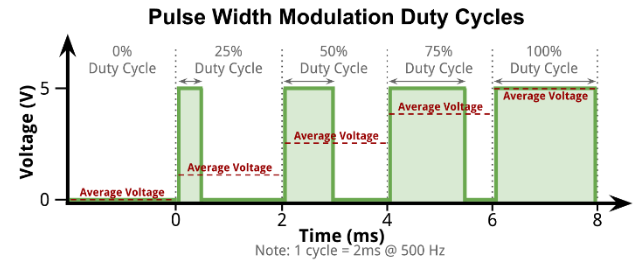
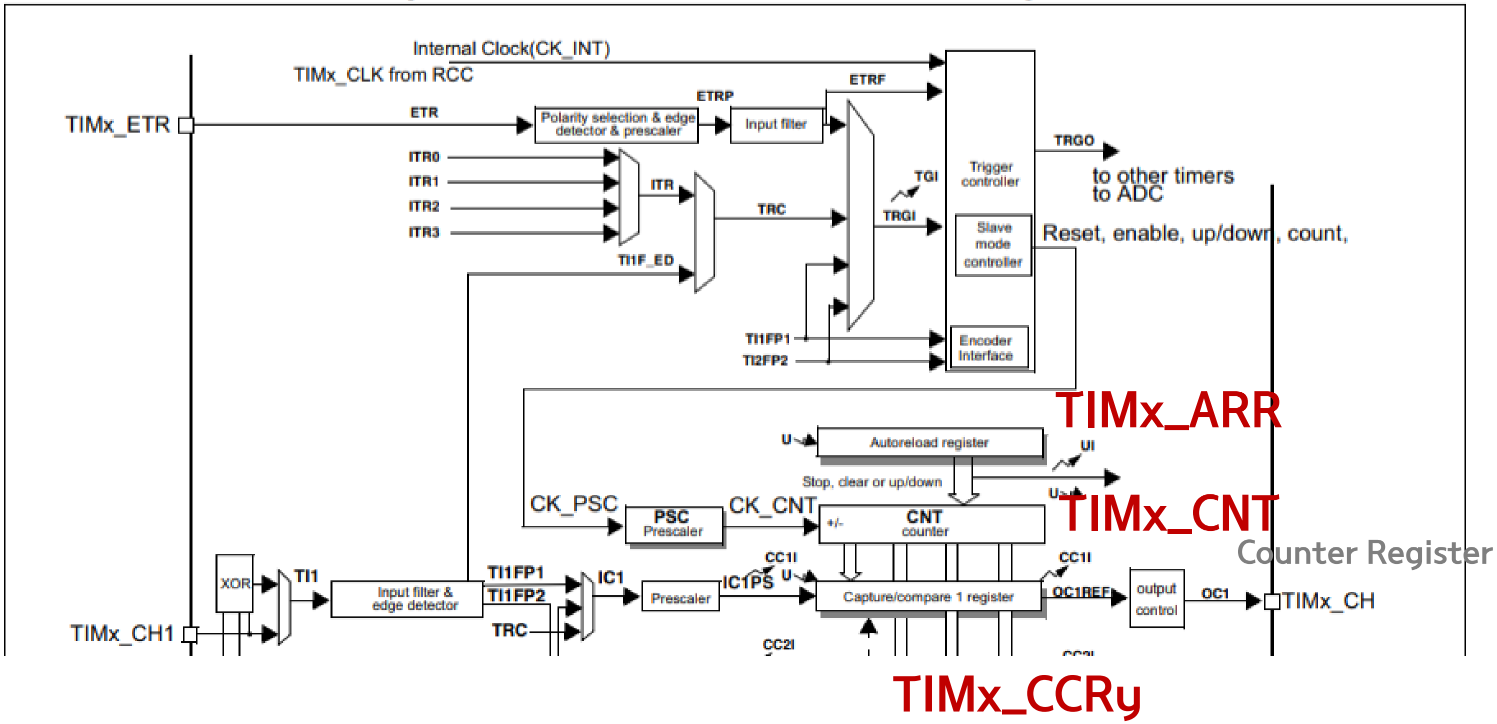
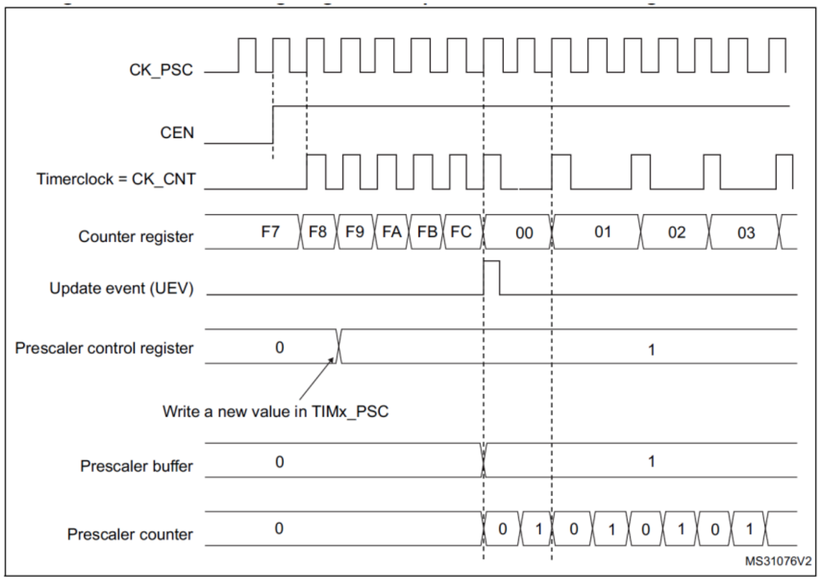


Figure 87. General-purpose timer block diagram

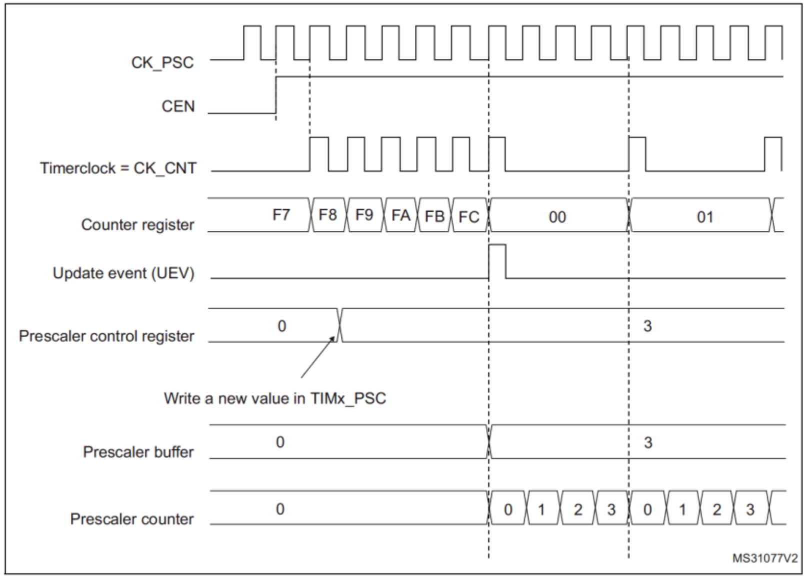


Review: Timer

- Prescaler: Counter register가 얼마에 한번씩 숫자가 증가할지를 결정
- ARR: Counter register가 최대 얼마까지 증가할 수 있는지를 결정
- Timer 인터럽트는 counter register가 지정 숫자 값까지 도달하면 발생



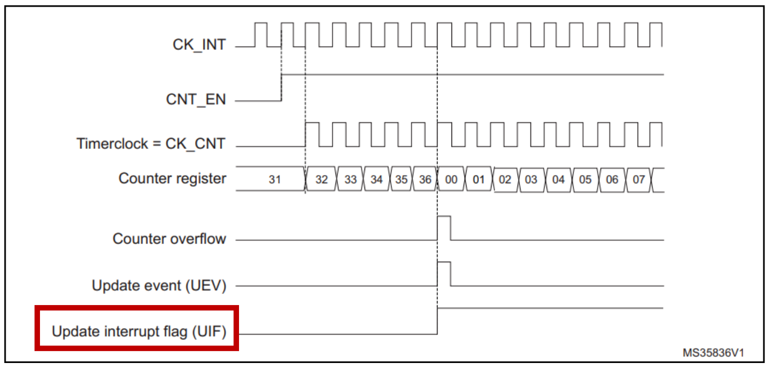
Counter timing diagram with prescaler division change from 1 to 2
ARR = FC



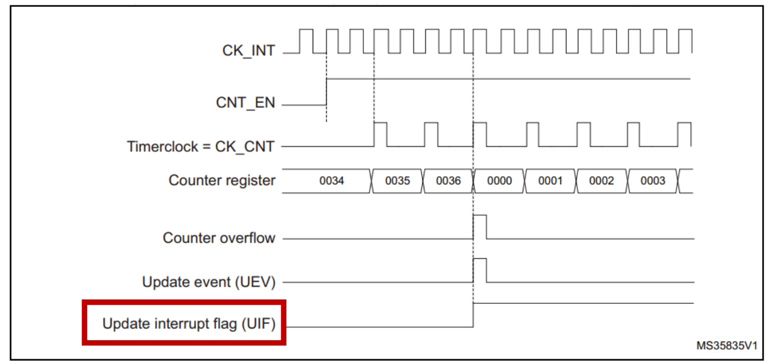
Counter timing diagram with prescaler division change from 1 to 4
ARR = FC

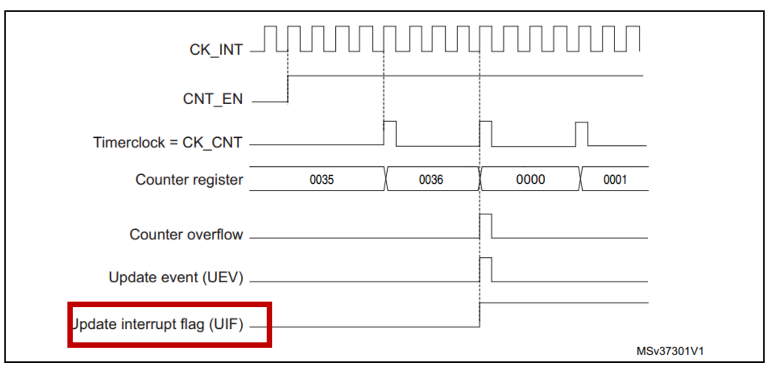
Review: Timer

- ARR = 36
- Up counting



Prescaler값이 작은 경우





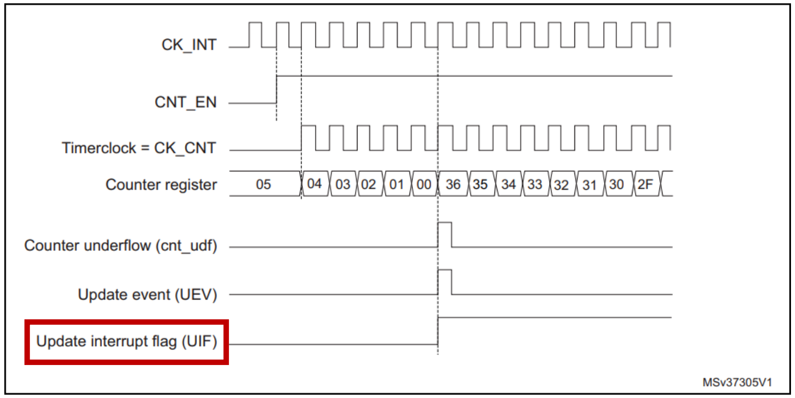
Prescaler값이 큰 경우

예.

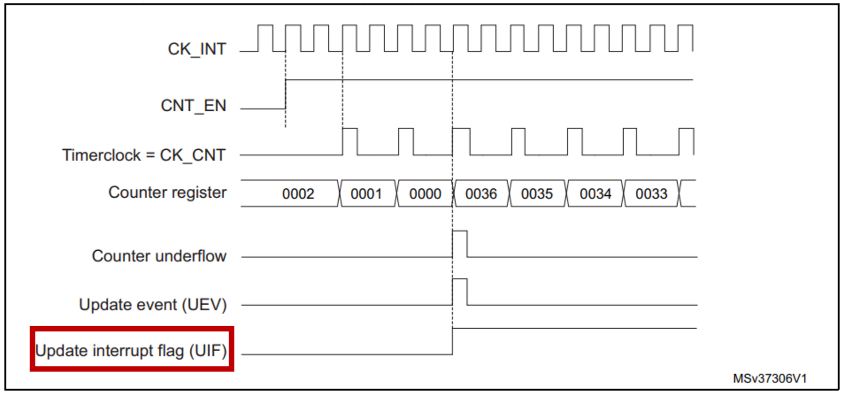
$$\frac{SYSTEM\ CLOCK}{Prescaler\text{값} \times Period\text{값}} = Freq. \Rightarrow \frac{26.88MHz}{26880 \times 1000} = 1Hz$$

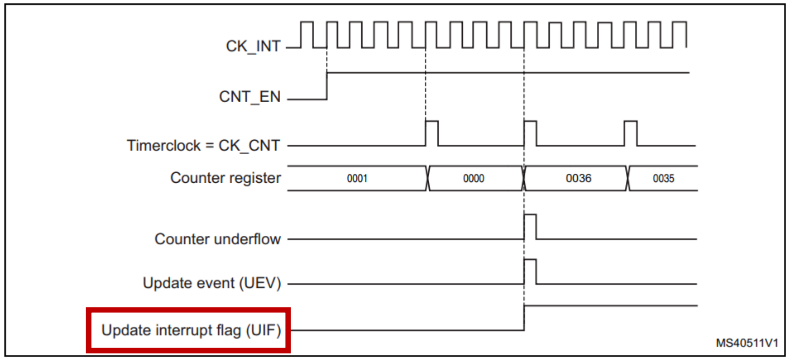
Review: Timer

- ARR = 36
- Down counting



Prescaler값이 작은 경우





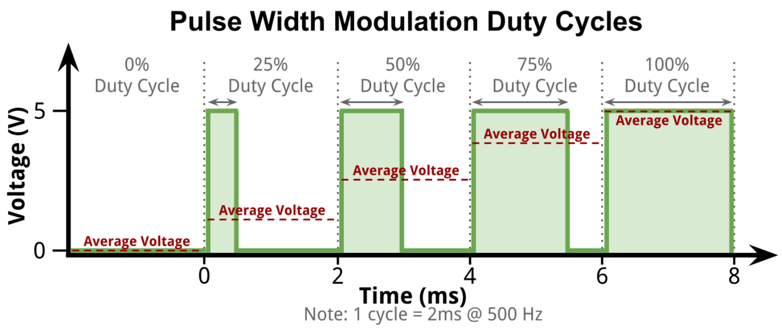
Prescaler값이 큰 경우

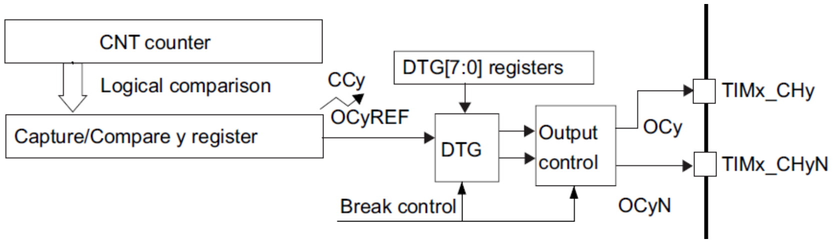
예.

$$\frac{SYSTEM\ CLOCK}{Prescaler\text{값} \times Period\text{값}} = Freq. \Rightarrow \frac{26.88MHz}{26880 \times 1000} = 1Hz$$

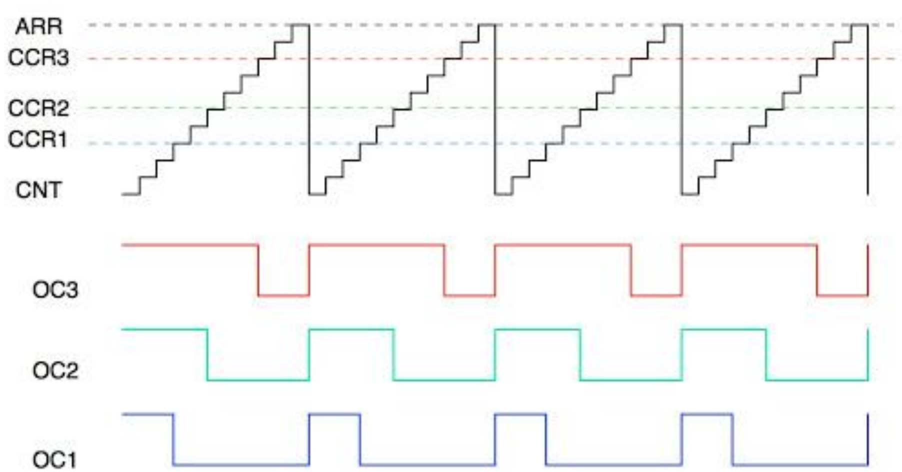
Pulse Width Modulation (PWM)

- TIMx_ARR : Period
- TIMx_CCR : Duty





Three PWM signals from the Output Compare Channels of a general purpose timer



Pulse Width Modulation (PWM)

- Timer CH
- Alternate function mapping

Table 9. Alternate function mapping

Port		AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09	AF10	AF11	AF12	AF13	AF14	AF15
		SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/I2S1SPI2/ I2S2/SPI3/ I2S3	SPI2/I2S2/ SPI3/ I2S3/SPI4/ I2S4/SPI5/ I2S5	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3	OTG1_FS		SDIO			
Port A	PA0	-	TIM2_CH1/ TIM2_ETR	TIM5_CH1	-	-	-	-	USART2_ CTS	-	-	-	-	-	-	-	EVENT OUT
	PA1	-	TIM2_CH2	TIM5_CH2	-	-	SPI4_MOSI /I2S4_SD	-	USART2_ RTS	-	-	-	-	-	-	-	EVENT OUT
	PA2	-	TIM2_CH3	TIM5_CH3	TIM9_CH1	-	I2S2_CKIN	-	USART2_ TX	-	-	-	-	-	-	-	EVENT OUT
	PA3	-	TIM2_CH4	TIM5_CH4	TIM9_CH2	-	I2S2_MCK	-	USART2_ RX	-	-	-	-	-	-	-	EVENT OUT
	PA4	-	-	-	-	-	SPI1_NSS/I2S1_WS	SPI3_NSS/I2S3_WS	USART2_ CK	-	-	-	-	-	-	-	EVENT OUT
	PA5	-	TIM2_CH1/ TIM2_ETR	-	-	-	SPI1_SCK/I2S1_CK	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA6	-	TIM1_BKIN	TIM3_CH1	-	-	SPI1_MISO	I2S2_MCK	-	-	-	-	-	SDIO_ CMD	-	-	EVENT OUT
	PA7	-	TIM1_CH1N	TIM3_CH2	-	-	SPI1_MOSI /I2S1_SD	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA8	MCO_1	TIM1_CH1	-	-	I2C3_ SCL	-	-	USART1_ CK	-	-	USB_FS_ SOF	-	SDIO_ D1	-	-	EVENT OUT
	PA9	-	TIM1_CH2	-	-	I2C3_ SMBA	-	-	USART1_ TX	-	-	USB_FS_ VBUS	-	SDIO_ D2	-	-	EVENT OUT
	PA10	-	TIM1_CH3	-	-	-	-	SPI5_MOSI/I2S5_SD	USART1_ RX	-	-	USB_FS_ ID	-	-	-	-	EVENT OUT
	PA11	-	TIM1_CH4	-	-	-	-	SPI4_MISO	USART1_ CTS	USART6_ TX	-	USB_FS_ DM	-	-	-	-	EVENT OUT
	PA12	-	TIM1_ETR	-	-	-	-	SPI5_MISO	USART1_ RTS	USART6_ RX	-	USB_FS_ DP	-	-	-	-	EVENT OUT
	PA13	JTMS- SWDIO	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA14	JTCK- SWCLK	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PA15	JTDI	TIM2_CH1/ TIM2_ETR	-	-	-	SPI1_NSS/I2S1_WS	SPI3_NSS/I2S3_WS	USART1_ TX	-	-	-	-	-	-	-	EVENT OUT

Pulse Width Modulation (PWM)

- Timer CH
- Alternate function mapping

Table 9. Alternate function mapping (continued)

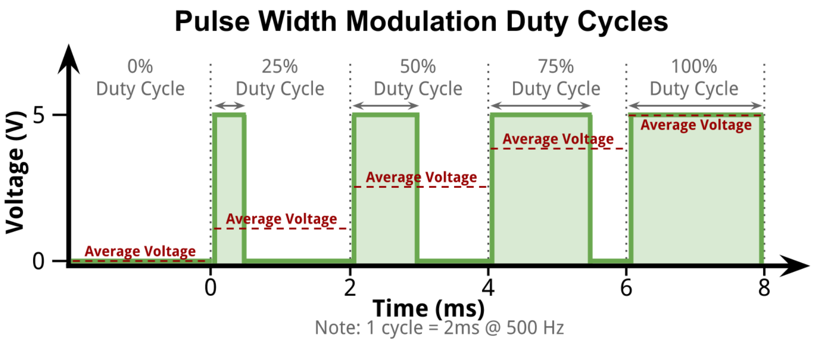
Port	AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09	AF10	AF11	AF12	AF13	AF14	AF15
	SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/I2S1SPI2/ I2S2/SPI3/ I2S3	SPI2/I2S2/ SPI3/ I2S3/SPI4/ I2S4/SPI5/ I2S5	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3	OTG1_FS		SDIO			
Port B PB0	-	TIM1_CH2N	TIM3_CH3	-	-	-	SPI5_SCK /I2S5_CK		-	-	-	-	-	-	-	EVENT OUT
PB1	-	TIM1_CH3N	TIM3_CH4	-	-	-	SPI5_NSS /I2S5_WS		-	-	-	-	-	-	-	EVENT OUT
PB2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
PB3	JTDO-SWO	TIM2_CH2	-	-	-	SPI1_SCK/I2S1_CK	SPI3_SCK /I2S3_CK	USART1_RX	-	I2C2_SDA	-	-	-	-	-	EVENT OUT
PB4	JTRST		TIM3_CH1	-	-	SPI1_MISO	SPI3_MISO	I2S3ext_SD	-	I2C3_SDA			SDIO_D0	-	-	EVENT OUT
PB5	-	-	TIM3_CH2	-	I2C1_SMBA	SPI1_MOSI /I2S1_SD	SPI3_MOSI/ I2S3_SD		-	-	-	-	SDIO_D3	-	-	EVENT OUT
PB6	-	-	TIM4_CH1	-	I2C1_SCL	-	-	USART1_TX	-	-	-	-		-	-	EVENT OUT
PB7	-	-	TIM4_CH2	-	I2C1_SDA	-	-	USART1_RX	-	-	-	-	SDIO_D0	-	-	EVENT OUT
PB8	-	-	TIM4_CH3	TIM10_CH1	I2C1_SCL	-	SPI5_MOSI/ I2S5_SD	-	-	I2C3_SDA	-	-	SDIO_D4	-	-	EVENT OUT
PB9	-	-	TIM4_CH4	TIM11_CH1	I2C1_SDA	SPI2_NSS/I2S2_WS	-	-	-	I2C2_SDA	-	-	SDIO_D5	-	-	EVENT OUT
PB10	-	TIM2_CH3	-	-	I2C2_SCL	SPI2_SCK/I2S2_CK	I2S3_MCK	-	-	-	-	-	SDIO_D7	-	-	EVENT OUT
PB11	-	TIM2_CH4	-	-	I2C2_SDA	I2S2_CKIN	-	-	-	-	-	-	-	-	-	EVENT OUT
PB12	-	TIM1_BKIN	-	-	I2C2_SMBA	SPI2_NSS/I2S2_WS	SPI4_NSS /I2S4_WS	SPI3_SCK /I2S3_CK	-	-	-	-	-	-	-	EVENT OUT
PB13	-	TIM1_CH1N	-	-	-	SPI2_SCK/I2S2_CK	SPI4_SCK/ I2S4_CK	-	-	-	-	-	-	-	-	EVENT OUT
PB14	-	TIM1_CH2N	-	-	-	SPI2_MISO	I2S2ext_SD	-	-	-	-	-	SDIO_D6	-	-	EVENT OUT
PB15	RTC_50Hz	TIM1_CH3N	-	-	-	SPI2_MOSI /I2S2_SD	-	-	-	-	-	-	SDIO_CK	-	-	EVENT OUT

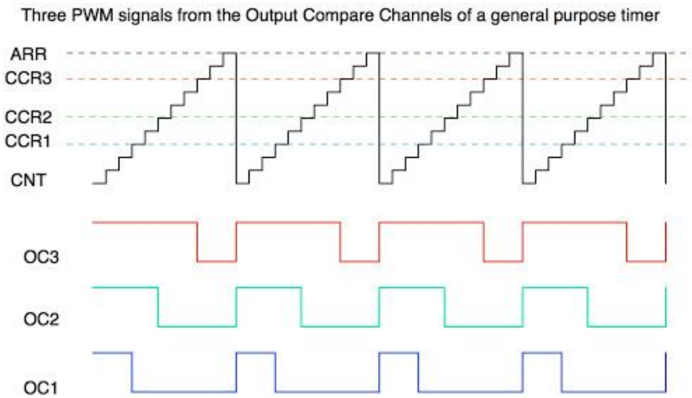
예.
PB6 ~ 9를 AF로 쓰면,
PWM 신호를 4개 만들 수 있음

Pulse Width Modulation (PWM)

LED case

- 전원 공급 / 미공급의 주기가 빠르면, 우리 눈에는 보이지 않을 것
- PWM을 통해 공급되는 average voltage를 조절할 수 있고, LED의 밝기는 바뀔 것

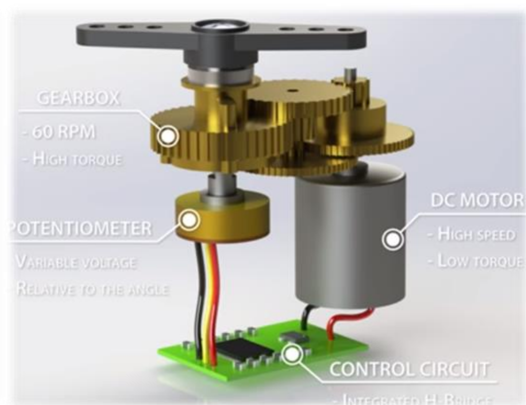




Pulse Width Modulation (PWM)

RC 서보모터

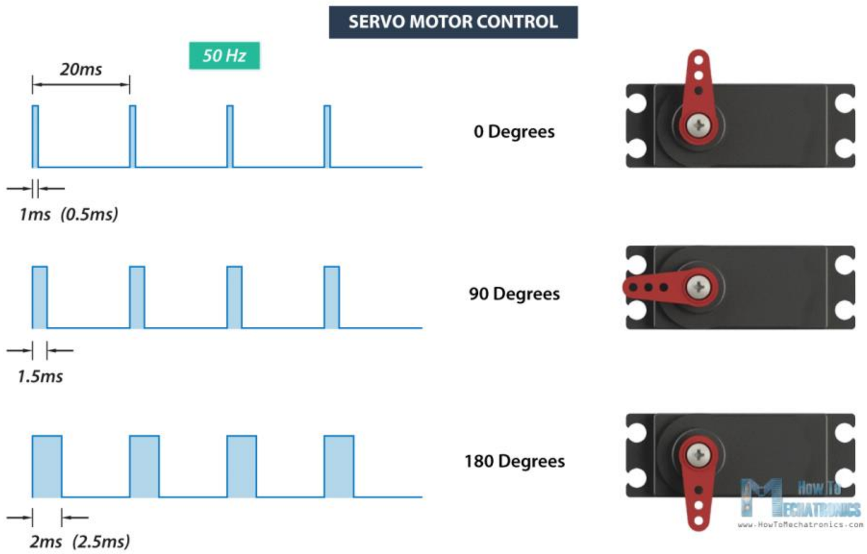
- 명령에 따라 정확한 위치와 속도를 맞출 수 있는 모터
- 서보 시스템 = 모터 + 기어박스 + 제어장치
- 소형, 탈부착 용이, 고정밀
- 로봇의 관절에 많이 사용됨



Pulse Width Modulation (PWM)

RC 서보모터

- 서보 모터는 일반적으로 약 0~180도 범위의 회전각을 가짐
- 서보 모터는 PWM (펄스폭변조) 방식으로 제어
- 일반적으로 주기는 20ms
- 펄스 폭이 1ms 이면 각은 0도
- 펄스 폭이 1.5ms 이면 각은 90도
- 펄스 폭이 2ms 면 각은 180도



실습 1: RGB-LED 제어

RGB-LED

- R, G, B 핀과 GND 핀으로 구성(LED 뒷면에 핀 배치 적혀 있음)
- 예를 들어, R에 전원이 공급되면 빨간색이 켜지고, R, G, B에 전원이 공급되면 모두 켜짐 (= 흰색으로 보임)

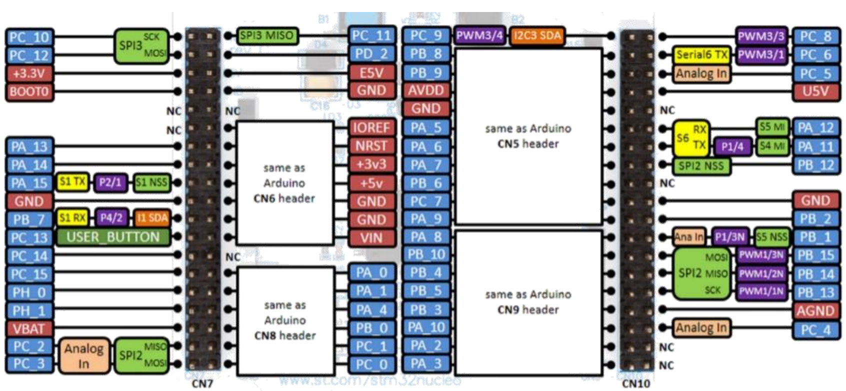
실습 1: RGB-LED 제어

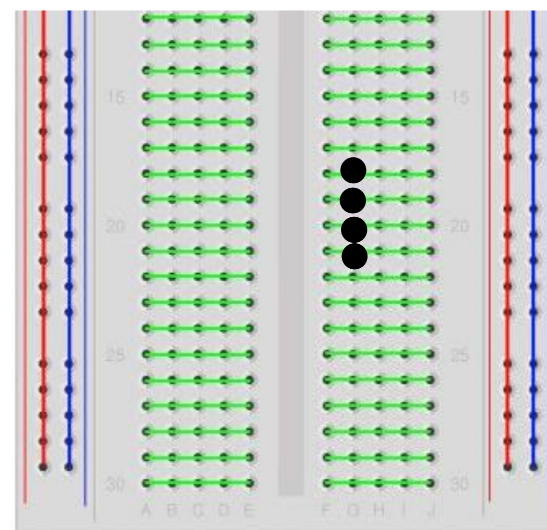
HW / FW 구성

Table 9. Alternate function mapping (continued)

	Port	AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09	AF10	AF11	AF12	AF13	AF14	AF15
		SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/I2S1SPI2/ I2S2/SPI3/ I2S3	SPI2/I2S2/ SPI3/ I2S3/SPI4/ I2S4/SPI5/ I2S5	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3	OTG1_FS		SDIO			
Port B	PB0	-	TIM1_CH2N	TIM3_CH3	-	-	-	SPI5_SCK /I2S5_CK		-	-	-	-	-	-	-	EVENT OUT
	PB1	-	TIM1_CH3N	TIM3_CH4	-	-	-	SPI5_NSS /I2S5_WS		-	-	-	-	-	-	-	EVENT OUT
	PB2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVENT OUT
	PB3	JTDO-SWO	TIM2_CH2	-	-	-	SPI1_SCK/I2S1_CK	SPI3_SCK /I2S3_CK	USART1_RX	-	I2C2_SDA	-	-	-	-	-	EVENT OUT
	PB4	JTRST		TIM3_CH1	-	-	SPI1_MISO	SPI3_MISO	I2S3ext_SD	-	I2C3_SDA			SDIO_D0	-	-	EVENT OUT
	PB5	-	-	TIM3_CH2	-	I2C1_SMBA	SPI1_MOSI /I2S1_SD	SPI3_MOSI/ I2S3_SD		-	-	-	-	SDIO_D3	-	-	EVENT OUT
	PB6	-	-	TIM4_CH1	-	I2C1_SCL	-	-	USART1_TX	-	-	-	-		-	-	EVENT OUT
	PB7	-	-	TIM4_CH2	-	I2C1_SDA	-	-	USART1_RX	-	-	-	-	SDIO_D0	-	-	EVENT OUT
	PB8	-	-	TIM4_CH3	TIM10_CH1	I2C1_SCL	-	SPI5_MOSI/ I2S5_SD	-	-	I2C3_SDA	-	-	SDIO_D4	-	-	EVENT OUT
	PB9	-	-	TIM4_CH4	TIM11_CH1	I2C1_SDA	SPI2_NSS/I2S2_WS	-	-	-	I2C2_SDA	-	-	SDIO_D5	-	-	EVENT OUT
	PB10	-	TIM2_CH3	-	-	I2C2_SCL	SPI2_SCK/I2S2_CK	I2S3_MCK	-	-	-	-	-	SDIO_D7	-	-	EVENT OUT
	PB11	-	TIM2_CH4	-	-	I2C2_SDA	I2S2_CKIN	-	-	-	-	-	-	-	-	-	EVENT OUT
	PB12	-	TIM1_BKIN	-	-	I2C2_SMBA	SPI2_NSS/I2S2_WS	SPI4_NSS /I2S4_WS	SPI3_SCK /I2S3_CK	-	-	-	-	-	-	-	EVENT OUT
	PB13	-	TIM1_CH1N	-	-	-	SPI2_SCK/I2S2_CK	SPI4_SCK/ I2S4_CK	-	-	-	-	-	-	-	-	EVENT OUT
	PB14	-	TIM1_CH2N	-	-	-	SPI2_MISO	I2S2ext_SD	-	-	-	-	-	SDIO_D6	-	-	EVENT OUT
	PB15	RTC_50Hz	TIM1_CH3N	-	-	-	SPI2_MOSI /I2S2_SD	-	-	-	-	-	-	SDIO_CK	-	-	EVENT OUT

- HW 구성
  - PB6~8까지 각각 R, G, B 단자와 연결
  - RGB-LED 모듈의 GND 연결
- FW 구성
  - 프로젝트 파일 추가
  - 폴더 : STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Libraries₩STM32F4xx_StdPeriph_Driver₩src
  - 파일 : misc.c, stm32f4xx_tim.c, stm32f4xx_syscfg.c





실습 1: RGB-LED 제어

SW 구성

- User LED 사용
- RGB-LED 사용
- Timer 2 사용
- Timer 4 - PWM 사용

- 다음 페이지에 코드가 구현되어 있습니다

먼저 코드를 살펴보시고, 이 코드를 실행하면 어떻게 동작할지 예상해봅시다!

실습 1: RGB-LED 제어

SW 구성

```
#include “stm32f4xx.h”

int intensity = 10;

void LED_init(void)
{
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

실습 1: RGB-LED 제어

SW 구성

```
void TIM4_Configuration(int period)
{
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);
  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_TimeBaseStructure.TIM_Prescaler = 1 - 1;
  TIM_TimeBaseStructure.TIM_Period = period - 1;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
  TIM_Cmd(TIM4, ENABLE);
}
```

실습 1: RGB-LED 제어

SW 구성

```
void PWM_TIM4_Configuration(void)
{
  TIM_OCInitTypeDef TIM_OCInitStructure;
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OCInitStructure.TIM_Pulse = 0;
  TIM_OC1Init(TIM4, &TIM_OCInitStructure);
  TIM_OC2Init(TIM4, &TIM_OCInitStructure);
  TIM_OC3Init(TIM4, &TIM_OCInitStructure);
}
```

(참고) OC: output compare

실습 1: RGB-LED 제어

SW 구성

```
void RGB_LED_init(void)
{
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOB, &GPIO_InitStructure);
  GPIO_PinAFConfig(GPIOB, GPIO_PinSource6, GPIO_AF_TIM4);
  GPIO_PinAFConfig(GPIOB, GPIO_PinSource7, GPIO_AF_TIM4);
  GPIO_PinAFConfig(GPIOB, GPIO_PinSource8, GPIO_AF_TIM4);
}
```

실습 1: RGB-LED 제어

SW 구성

```
void TIM2_Configuration(int interval_ms)
{
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_TimeBaseStructure.TIM_Prescaler = 26880 - 1;
  TIM_TimeBaseStructure.TIM_Period = interval_ms - 1;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
  TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
  TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
  TIM_Cmd(TIM2, ENABLE);

  NVIC_InitTypeDef NVIC_InitStructure;
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
  NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
```

실습 1: RGB-LED 제어

SW 구성

```
void TIM2_IRQHandler(void)
{
 if(TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
 {
  if (intensity > 25600)
  {
   intensity = 10;
  }
  TIM4->CCR1 = intensity;
  TIM4->CCR2 = intensity;
  TIM4->CCR3 = intensity;
  intensity *= 2;
  GPIO_ToggleBits(GPIOA, GPIO_Pin_5);
  // clear interrupt flag
  TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
 }
}
```

(참고) CCR: capture/compare register

실습 1: RGB-LED 제어

SW 구성

```
int main()
{
  LED_init();
  RGB_LED_init();
  intensity = 10;
  TIM4_Configuration(25600);
  PWM_TIM4_Configuration();
  TIM2_Configuration(500);
  while(1)
  {
  }
  return 0;
}
```

실습 1: RGB-LED 제어

SW 해석

- Timer 4에서 PWM 3개를 생성하여 RGB-LED에 공급
- PWM의 duty는 Timer 2의 인터럽트가 걸렸을 때 (0.5초에 한번) 업데이트 됨
  - CCR1/2/3가 10, 20, 40, 80, 160, …, 증가
  - CCR1/2/3 가 R, G, B에 전원을 공급하는 소스임
  - Timer 2의 인터럽트가 걸리면, User LED가 on/off

- 결과
  - 0.5초에 한번 씩 User LED가 on/off
  - R, G, B LED가 모두 꺼져 있다가 서서히 밝아짐

실습 1: RGB-LED 제어

SW 해설

- **User LED 초기화: port A, 5번핀 output으로 사용**

```
#include “stm32f4xx.h”

int intensity = 10;

void LED_init(void)
{
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

실습 1: RGB-LED 제어

SW 구성

- **Timer 4의 기본 설정**

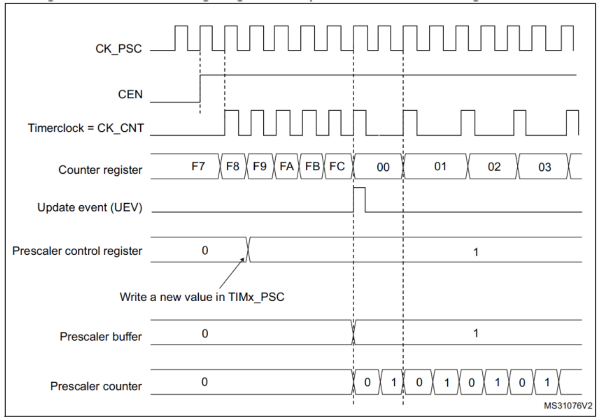
25600

```
void TIM4_Configuration(int period)
{
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);
  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_TimeBaseStructure.TIM_Prescaler = 1 - 1;
  TIM_TimeBaseStructure.TIM_Period = period - 1;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
  TIM_Cmd(TIM4, ENABLE);
}
```

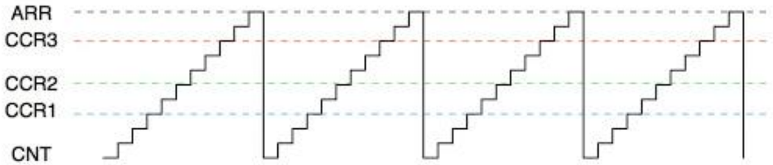
Prescaler와 ARR 설정

$$\frac{SYSTEM\ CLOCK}{Prescaler값 \times Period값}$$

ARR



Three PWM signals from the Output Compare Channels of a general purpose timer



실습 1: RGB-LED 제어

SW 구성

(참고) OC: output compare

- **Timer 4의 PWM 설정**

```
void PWM_TIM4_Configuration(void)
{
 TIM_OCInitTypeDef TIM_OCInitStructure;
 TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
 TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
 TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;

 TIM_OCInitStructure.TIM_Pulse = 0;

 TIM_OC1Init(TIM4, &TIM_OCInitStructure);
 TIM_OC2Init(TIM4, &TIM_OCInitStructure);
 TIM_OC3Init(TIM4, &TIM_OCInitStructure);
}
```

PWM모드 설정
PWM1 mode: Tpulse = '1'
PWM2 mode: Tpulse = '0

PWM의 출력 여부 결정

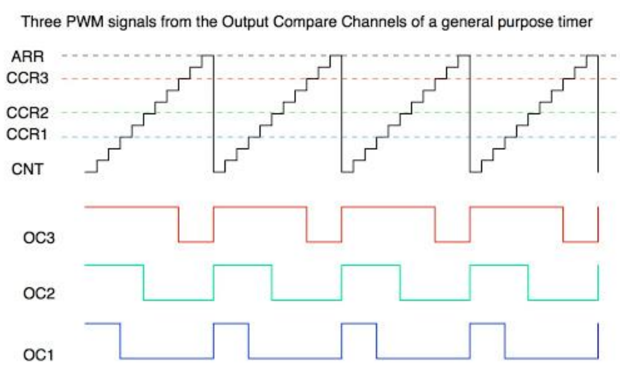
PWM출력 시작값 정함 (1 or 0)

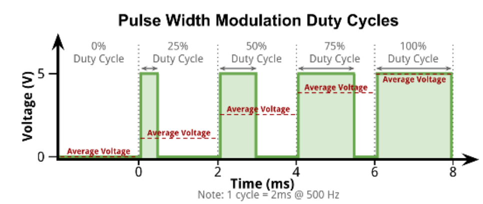
PWM의 duty비를 설정(CCRx)

$$Duty\ Ratio = \frac{High\ 상태시간}{주기} = \frac{TIM\_Pulse(CCRx)}{TIM\_Period(TIMx\_ARR)},$$

단 PWM1 모드의 경우

Output compare channel 1~3까지 초기화





실습 1: RGB-LED 제어

SW 구성

- **RGB-LED에 PWM 연동**

```
void RGB_LED_init(void)
{
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
  GPIO_InitTypeDef GPIO_InitStructure;

  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOB, &GPIO_InitStructure);

  GPIO_PinAFConfig(GPIOB, GPIO_PinSource6, GPIO_AF_TIM4);
  GPIO_PinAFConfig(GPIOB, GPIO_PinSource7, GPIO_AF_TIM4);
  GPIO_PinAFConfig(GPIOB, GPIO_PinSource8, GPIO_AF_TIM4);
}
```

AF로 Timer4의 channel1~3까지 사용

실습 1: RGB-LED 제어

SW 구성

- **Timer 2를 Timer 인터럽트로 사용 (지난 수업 자료 참고)**

```
void TIM2_Configuration(int interval_ms)
{
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_TimeBaseStructure.TIM_Prescaler = 26880 - 1;
  TIM_TimeBaseStructure.TIM_Period = interval_ms - 1;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
  TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
  TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
  TIM_Cmd(TIM2, ENABLE);

  NVIC_InitTypeDef NVIC_InitStructure;
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
  NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
```

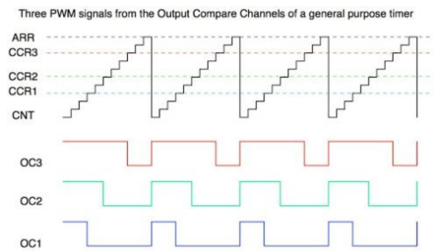
실습 1: RGB-LED 제어

SW 구성

- **Timer 2를 Timer 인터럽트로 사용 (지난 수업 자료 참고)**

```
void TIM2_IRQHandler(void)
{
 if(TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
 {
  if (intensity > 25600)
  {
   intensity = 10;
  }
  TIM4->CCR1 = intensity;
  TIM4->CCR2 = intensity;
  TIM4->CCR3 = intensity;
  intensity *= 2;
  GPIO_ToggleBits(GPIOA, GPIO_Pin_5);
  // clear interrupt flag
  TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
 }
}
```

타이머2의 인터럽트가 걸리면,



Timer 4 PWM의 CCR을 변경

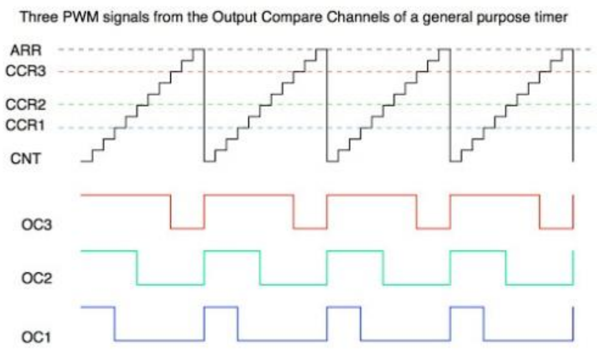
intensity 변수 조절

User LED on/off (인터럽트가 잘 걸리는지 확인하는 목적)

(참고) CCR: capture/compare register

실습 1: RGB-LED 제어

SW 구성



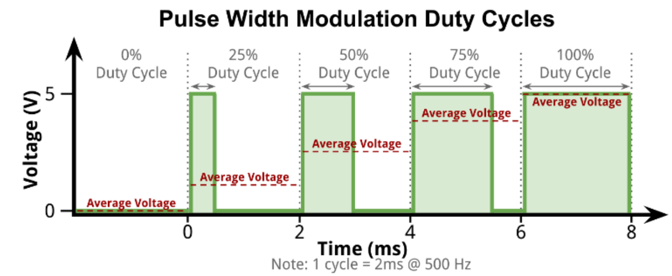

```
int main()
{
  LED_init();
  RGB_LED_init();
  intensity = 10;
  TIM4_Configuration(25600);
  PWM_TIM4_Configuration();
  TIM2_Configuration(500);
  while(1)
  {
  }
  return 0;
}
```

User LED 초기화

RGB LED 초기화

Timer 4 PWM 설정

Timer 2 설정

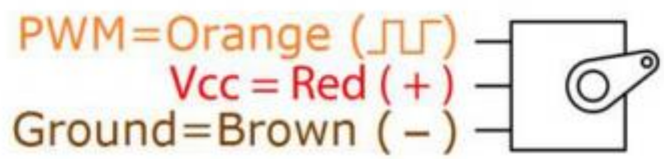


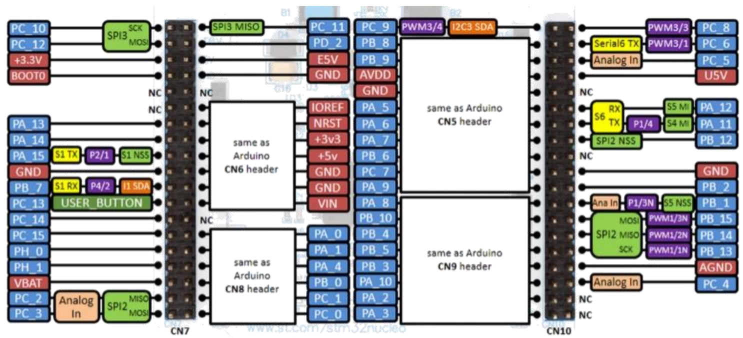
- CCR1/2/3 를 intensity변수로 동일하게 조절하므로, R, G, B의 밝기가 동일하게 변경
- Intensity가 25600이 되면, ARR과 CCR아 같아지는 것이므로 OC1/2/3출력은 주기동안 모두 1 (5V)을 출력 (가장 밝은 밝기)
- 참고로, 이 실습에서는 한 주기에 대해서는 크게 고려하지 않음
  $25600 / 26.88 * 10^6 \approx 1ms$ (1 cycle)
  충분히 빠른 시간으로 우리 눈에는 감지할 수 없음

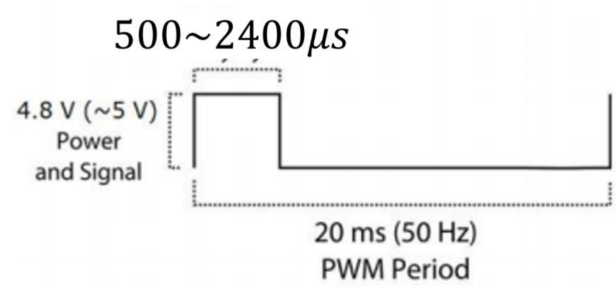
실습 2: PWM을 이용하여 RC servo motor 구동하기

HW / FW구성

- HW 구성: SG90(servo motor) 전원 연결
  - Vcc : 5V
  - PWM : PB9







- FW 구성: 프로젝트 파일 추가
  - 폴더 : STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Libraries ₩STM32F4xx_StdPeriph_Driver₩src
  - 파일 : misc.c, stm32f4xx_tim.c, stm32f4xx_syscfg.c

실습 2: PWM을 이용하여 RC servo motor 구동하기

HW / FW구성

- FW 구성
  - TIM4 CH4를 이용하여 PWM신호를 생성
  - 주기 : 20msec (50Hz)
  - pulse width : 500 ~ 2400usec (0°~180°)
  - 10°각도 만큼 제어가 가능하도록 설계
  - TIM2를 이용하여 1초마다 0°↔180°를 반복

실습 2: PWM을 이용하여 RC servo motor 구동하기

HW / FW구성

- 상황 1
  - 주기 : 20msec (50Hz)
  - 시스템 클럭은 26.88 MHz

$$\frac{\boldsymbol{SYSTEM\ CLOCK}}{\boldsymbol{Prescaler}\text{값} \times \boldsymbol{Period}\text{값}}$$

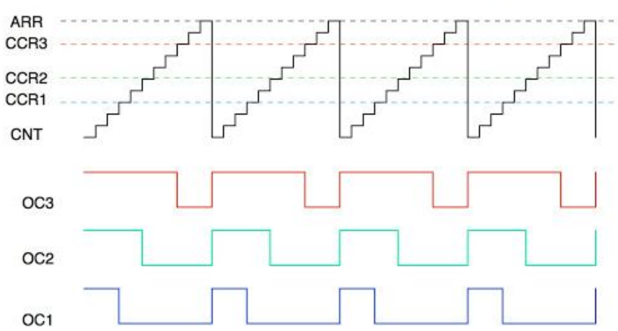
Prescaler 값 ARR 값

CNT는 perscaler 값까지 증가 후 1씩 증가됨
Perscaler 0 이면, 시스템 한 클럭에 1씩 증가할 것임

그리고 CNT가 ARR까지 도달하면, 한 사이클이 끝남

그러므로, 시스템 클럭 속도 / (Prescaler 값 * ARR 값)가
한 사이클을 결정할 것인데, 시스템 클럭 속도를 알고 있고,
한 사이클이 50Hz가 되야한다는 사실을 알고 있는 상황임

Three PWM signals from the Output Compare Channels of a general purpose timer



$$\frac{\mathbf{26.88 \times 10^6}}{\boldsymbol{Prescaler}\text{값} \times \boldsymbol{Period}\text{값}} = \mathbf{50}\boldsymbol{Hz}$$

참고로, Hz는 주파수의 단위이고, 초당 진동횟수로 정의됨
50Hz라고 하면, 초당 50번 진동한다는 의미이고,
이를 시간으로 환산하면 1/50 = 0.02s = 20ms가 됨

실습 2: PWM을 이용하여 RC servo motor 구동하기

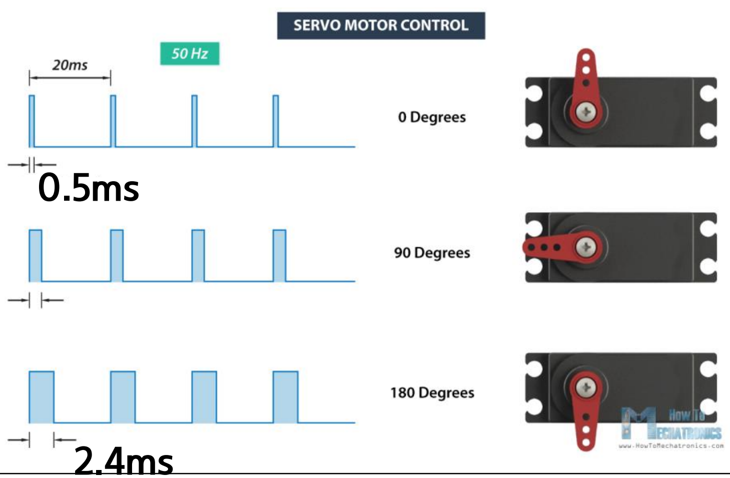
HW / FW구성

- 상황 2
  - 주기 : 20msec (50Hz)
  - 시스템 클럭은 26.88 MHz

$$\frac{26.88 \times 10^6}{Prescaler\text{값} \times Period\text{값}} = 50Hz$$

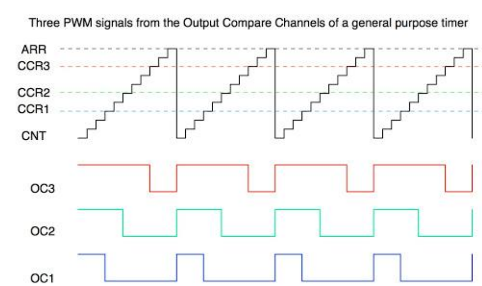
  - pulse width : 500 ~ 2400usec (0°~180°)
  - 10°각도 만큼 제어가 가능하도록 설계

여기서 문제가 있는데, 결정해야 할 변수가 2개임
Prescaler와 Period



이 단서들을 가지고,
Prescaler와 Period를 "잘" 결정해야 함!

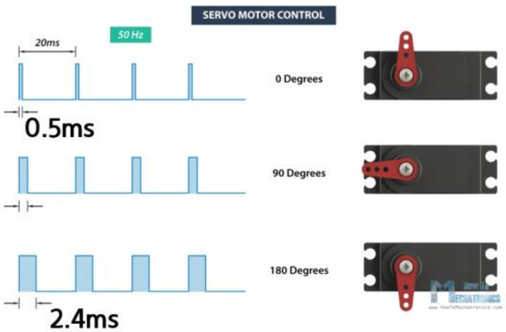
왜냐하면, perscaler 값과 CCR값은 연결이 되기때문!



실습 2: PWM을 이용하여 RC servo motor 구동하기

HW / FW구성

- 상황 2
  - pulse width : 500 ~ 2400usec (0°~180°)
  - 10°각도 만큼 제어가 가능하도록 설계



degree	Pulse width
0°	0.5ms
10°	
20°	
30°	
40°	
…	…
180°	2.4ms

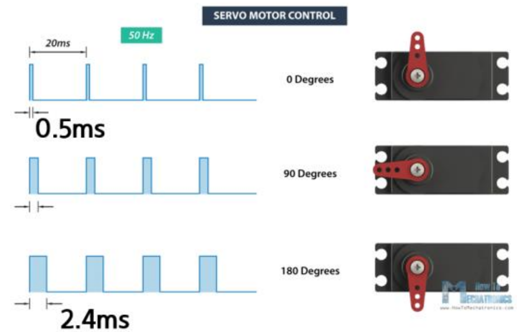
degree	Pulse width
0°	0.5ms
10°	0.6056ms
20°	0.7112ms
30°	0.8168ms
40°	0.9224ms
…	…
180°	2.4008ms

0.1056ms

실습 2: PWM을 이용하여 RC servo motor 구동하기

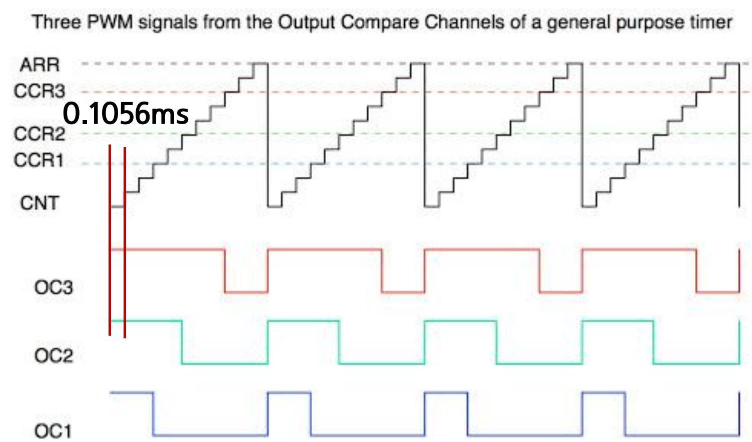
HW / FW구성

- Prescaler & period 계산
  - pulse width : 500 ~ 2400usec (0°~180°)
  - 10°각도 만큼 제어가 가능하도록 설계



degree	Pulse width
0°	0.5ms
10°	0.6056ms
20°	0.7112ms
30°	0.8168ms
40°	0.9224ms
…	…
180°	2.4008ms

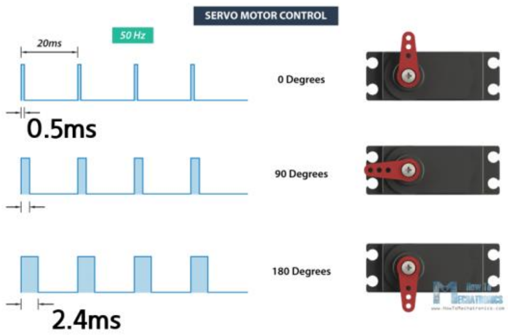
0.1056ms



CNT가 1 증가하는 시간이 0.1056ms 보다 작거나 같아야 10°를 인식할 수 있음을 의미!
이 0.1056ms를 결정할 수 있는 변수는 prescaler!!

실습 2: PWM을 이용하여 RC servo motor 구동하기

HW / FW구성

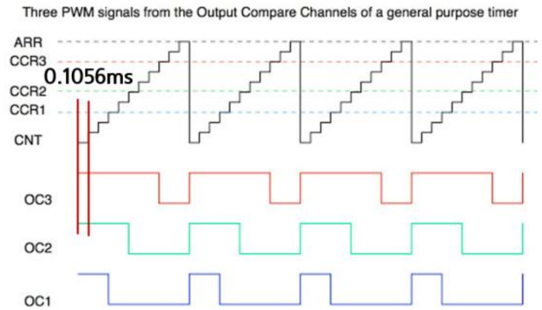


- Prescaler & period 계산
  - pulse width : 500 ~ 2400usec (0°~180°)
  - 10°각도 만큼 제어가 가능하도록 설계

$$\frac{2400 - 500\mu s}{18} = \frac{1900\mu s}{18} = 105.556\,\mu s = 0.1056\,ms$$

  - 주기 (s) = 1/주파수 (Hz), 주파수 (Hz) = 1/주기 (s)

$$\frac{1}{0.1056 \times 10^{-3}\,s} = 9473.7Hz$$



  - 시스템 클럭이 26.88MHz

$$\frac{26.88 \times 10^{6}}{X} = 9473.7Hz$$

$$X = 2837.3 \approx 2837 = prescaler$$

실습 2: PWM을 이용하여 RC servo motor 구동하기

HW / FW구성

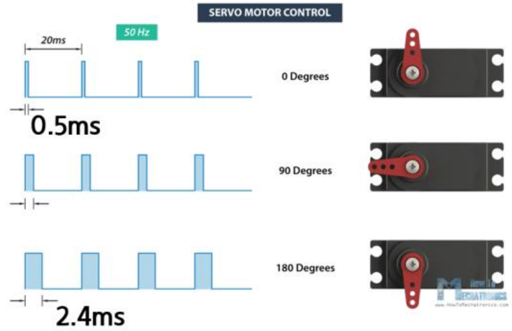
- Prescaler & period 계산

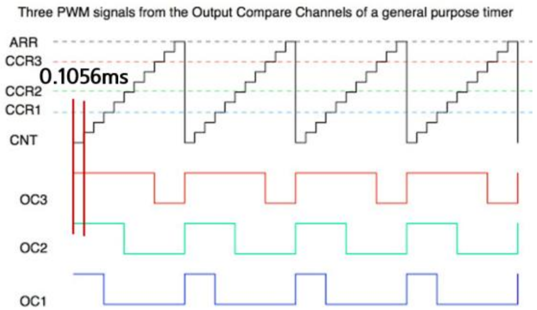
$$X = 2837.3 \approx 2837 = prescaler$$

$$\frac{26.88 \times 10^6}{Prescaler값 \times Period값} = 50Hz$$

$$\frac{26.88 \times 10^6}{2837 \times Y} = 50$$

$$Y = 189.49 \approx 189 = period$$





HW / FW구성

- CCR4 계산
  - pulse width : 500 ~ 2400usec (0°~180°)
  - 0°도 일 때,

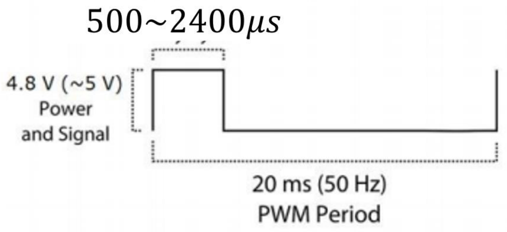
$$\frac{26.88\times 10^6}{2837\ \times\ CCR}(Hz)=\frac{1}{0.5\times 10^{-3}(s)}(Hz)$$

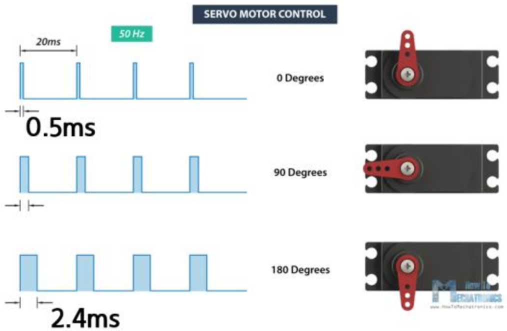
$$CCR=4.73\approx 5$$

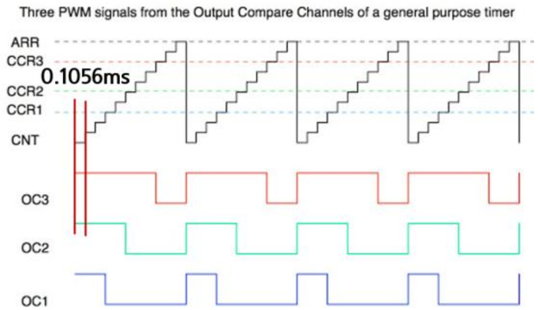
  - 180°도 일 때,

$$\frac{26.88\times 10^6}{2837\ \times\ CCR}(Hz)=\frac{1}{2.4\times 10^{-3}(s)}(Hz)$$

$$CCR=22.74\approx 23$$





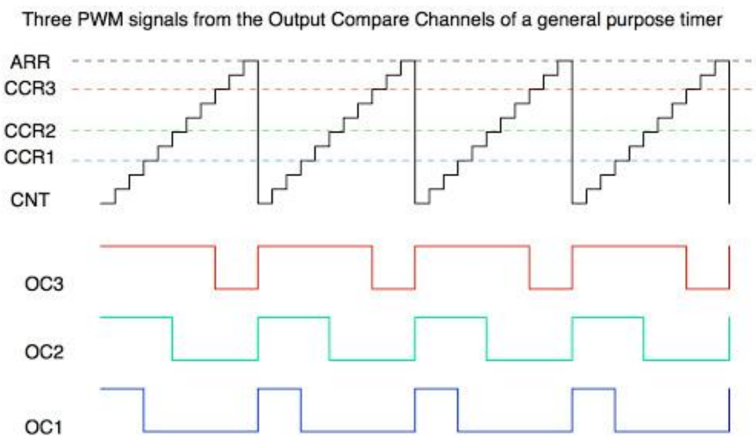


실습 2: PWM을 이용하여 RC servo motor 구동하기

HW / FW구성

- 정리
  - Prescaler: 2837
  - Period: 189

degree	CCR
0°	5
10°	6
20°	7
30°	8
40°	9
…	…
180°	23



실습 2: PWM을 이용하여 RC servo motor 구동하기

SW 구성

```
#include "stm32f4xx.h"

int degree = 5;
int intensity = 10;

void TIM4_Configuration(void)
{
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);
  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_TimeBaseStructure.TIM_Prescaler = 2837 - 1;
  TIM_TimeBaseStructure.TIM_Period = 189 - 1;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
  TIM_Cmd(TIM4, ENABLE);
}
```

실습 2: PWM을 이용하여 RC servo motor 구동하기

SW 구성

```
void PWM_TIM4_Configuration(void)
{
  TIM_OCInitTypeDef TIM_OCInitStructure;
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OCInitStructure.TIM_Pulse = 0;
  TIM_OC1Init(TIM4, &TIM_OCInitStructure);
  TIM_OC2Init(TIM4, &TIM_OCInitStructure);
  TIM_OC3Init(TIM4, &TIM_OCInitStructure);
  TIM_OC4Init(TIM4, &TIM_OCInitStructure);
}
```

실습 2: PWM을 이용하여 RC servo motor 구동하기

SW 구성

```
void PWM_TIM4_Configuration(void)
{
 TIM_OCInitTypeDef TIM_OCInitStructure;
 TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
 TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
 TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
 TIM_OCInitStructure.TIM_Pulse = 0;
 TIM_OC1Init(TIM4, &TIM_OCInitStructure);
 TIM_OC2Init(TIM4, &TIM_OCInitStructure);
 TIM_OC3Init(TIM4, &TIM_OCInitStructure);
 TIM_OC4Init(TIM4, &TIM_OCInitStructure);
}
```

실습 2: PWM을 이용하여 RC servo motor 구동하기

SW 구성

```
void RGB_LED_init(void)
{
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8 | GPIO_Pin_9;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOB, &GPIO_InitStructure);
  GPIO_PinAFConfig(GPIOB, GPIO_PinSource6, GPIO_AF_TIM4);
  GPIO_PinAFConfig(GPIOB, GPIO_PinSource7, GPIO_AF_TIM4);
  GPIO_PinAFConfig(GPIOB, GPIO_PinSource8, GPIO_AF_TIM4);
  GPIO_PinAFConfig(GPIOB, GPIO_PinSource9, GPIO_AF_TIM4);
}
```

실습 2: PWM을 이용하여 RC servo motor 구동하기

SW 구성

```
void TIM2_Configuration(int interval_ms)
{
 RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
 TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
 NVIC_InitTypeDef NVIC_InitStructure;
 TIM_TimeBaseStructure.TIM_Prescaler = 26880 - 1;
 TIM_TimeBaseStructure.TIM_Period = interval_ms - 1;
 TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
 TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
 TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
 TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
 TIM_Cmd(TIM2, ENABLE);
 NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
 NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
 NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
 NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
 NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
 NVIC_Init(&NVIC_InitStructure);
}
```

실습 2: PWM을 이용하여 RC servo motor 구동하기

SW 구성

```
void TIM2_IRQHandler(void)
{
  if(TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
  {
     TIM4->CCR1 = intensity;
     TIM4->CCR2 = intensity;
     TIM4->CCR3 = intensity;
     TIM4->CCR4 = degree;
     degree++;
     if (degree > 23)
     {
       degree = 5;
     }
     if (intensity > 25600)
     {
      intensity = 10;
     }
    intensity *= 2;
    GPIO_ToggleBits(GPIOA, GPIO_Pin_5);
    // clear interrupt flag
    TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
  }
}
```

실습 2: PWM을 이용하여 RC servo motor 구동하기

SW 구성

```
void LED_init(void)
{
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

실습 2: PWM을 이용하여 RC servo motor 구동하기

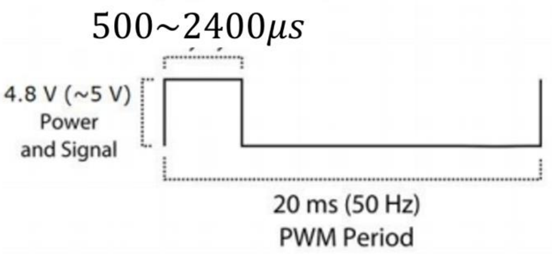
SW 구성

```
int main()
{
  LED_init();
  RGB_LED_init();
  degree = 5;
  intensity = 10;
  TIM4_Configuration();
  PWM_TIM4_Configuration();
  TIM2_Configuration(1000);
  while(1)
  {
  }
  return 0;
}
```

과제

- Timer 4의 PWM (CCR 1 / 2 /3)을 이용하여,
  - 노랑색, 하늘색, 자주색을 돌아가면서 출력
  - 밝기 조절
- Timer 4의 PWM (CCR 4)을 이용하여,
  - RC 서보모터를 1°씩 제어
- TIM2를 이용하여 1초마다
  - 색깔 변경
  - 밝기 조절
  - 0°↔180°를 반복

Prescaler
Period
CCR 계산



Thank you.