6강. PWM with Timer/Counter

박 원 엽
010.5451.0113

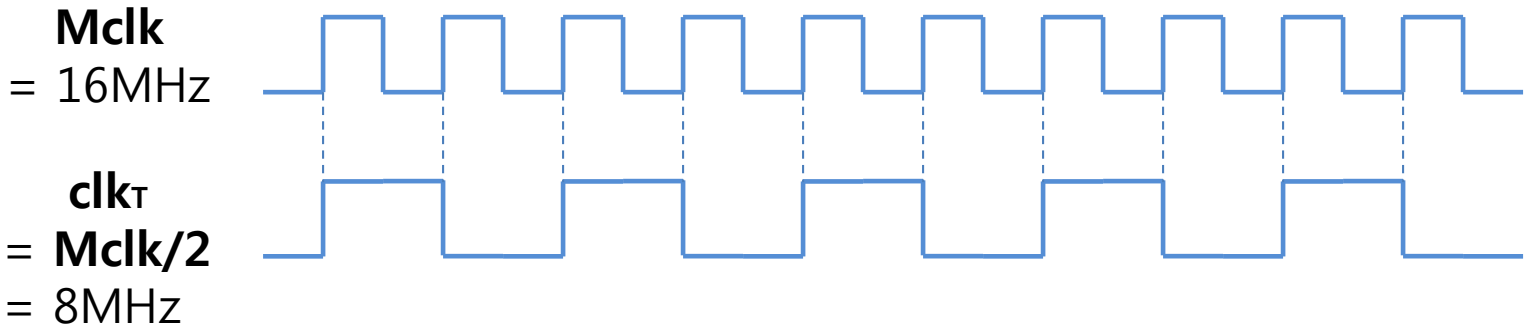
목 차

- **Timer/Counter란?**

  - 시간을 측정하기 위한 Peripheral 중 하나로써, 말 그대로 시간을 측정하는데 쓰이거나 PWM(Pulse Width Modulation)신호를 만드는데 사용된다.

- **Timer/Counter의 동작 원리?**

  - 시간 측정은 클럭(clkT)의 개수를 카운팅함으로써 가능해진다.
  - 예를 들어, clkT 가 100Hz의 주기로 펄스가 발생한다면 clkT 를 100번 카운팅하면 1초가 지난것이다.
  - 여기서 clkT 를 Timer/Counter의 clock source라고 하는데, 이 clkT 는 마스터 클럭(Mclk)인 16MHz와 관계가 있다.
  - 타이머/카운터를 사용하기 위해선 clock source를 입력해주어야 하는데 마스터 클럭(Mclk)인 16MHz를 얼마만큼의 비율로 나누어서 사용할지를 설정해야 한다.
  - 예를 들어, Mclk의 1/2로 clkT 를 설정한다고 하면 clkT 는 8MHz의 클럭이 된다. 이때 2분주(Prescale) 라는 표현을 쓴다.



- **Timer/Counter의 동작 원리?**

  - clock source를 설정해주면 클럭(clkT) 속도에 동기화 되어 타이머가 동작하기 시작하는데, 이때 카운팅 된 숫자가 TCNTn 레지스터에 자동으로 저장된다.

  - 즉, 타이머의 모든 설정을 완료하고 타이머를 동작시키면, 그 후에는 TCNTn 레지스터가 클럭(clkT) 속도에 맞춰 1씩 증가하는 것이다!

  - Atmega128에는 총 4개의 타이머가 존재하는데, Timer/Counter0, 2는 8bit이고, Timer/Counter1, 3은 16bit이다.
  여기서 8bit와 16bit가 의미하는 것은 최대 카운팅할 수 있는 숫자이다.

  - 즉, TCNT0과 TCNT2는 0~255까지의 값을 가질수 있고, TCNT1과 TCNT3은 0~65535까지의 값을 가질수 있다.

- **Timer/Counter 동작시키기 (8bit)**

  - Atmega128의 Timer/Counter의 동작 모드는 4가지가 있다.

  1. Normal Mode (데이터시트 98페이지)
  타이머/카운터의 가장 기본적인 모드이다.(위에서 설명한 내용) 클럭 인가시 카운터 레지스터(TCNTn)가 1씩 증가(또는 감소)하는 동작 모드이고, Up카운트 모드, Down카운트 모드가 있다.
  최대로 카운트 할 수 있는 숫자를 넘어가면 오버플로우(Overflow)가 발생하면서 인터럽트를 요청할 수 있다. 그 후 TCNTn는 다시 0부터 1씩 증가 한다.
  (Up카운트 모드 시. Down모드는 그 반대)

인터럽트(Interrupt)란?

프로그램 수행 시, C언어의 문법을 따라 절차지향(위에서부터 순서대로 한 문장씩 처리)적으로 처리되는데, 어떤 요인에 의해 인터럽트 요청이 발생되면 현재 수행하는 코드를 중단하고, 인터럽트 서비스 루틴(ISR) 함수를 수행하게 된다.

자세한 건 외부 인터럽트에서 다시 다룬다.

- **Timer/Counter 동작시키기 (8bit)**

2. Clear Timer on Compare Match (CTC) Mode
CTC모드는 Normal모드와는 약간 다르게 동작한다.

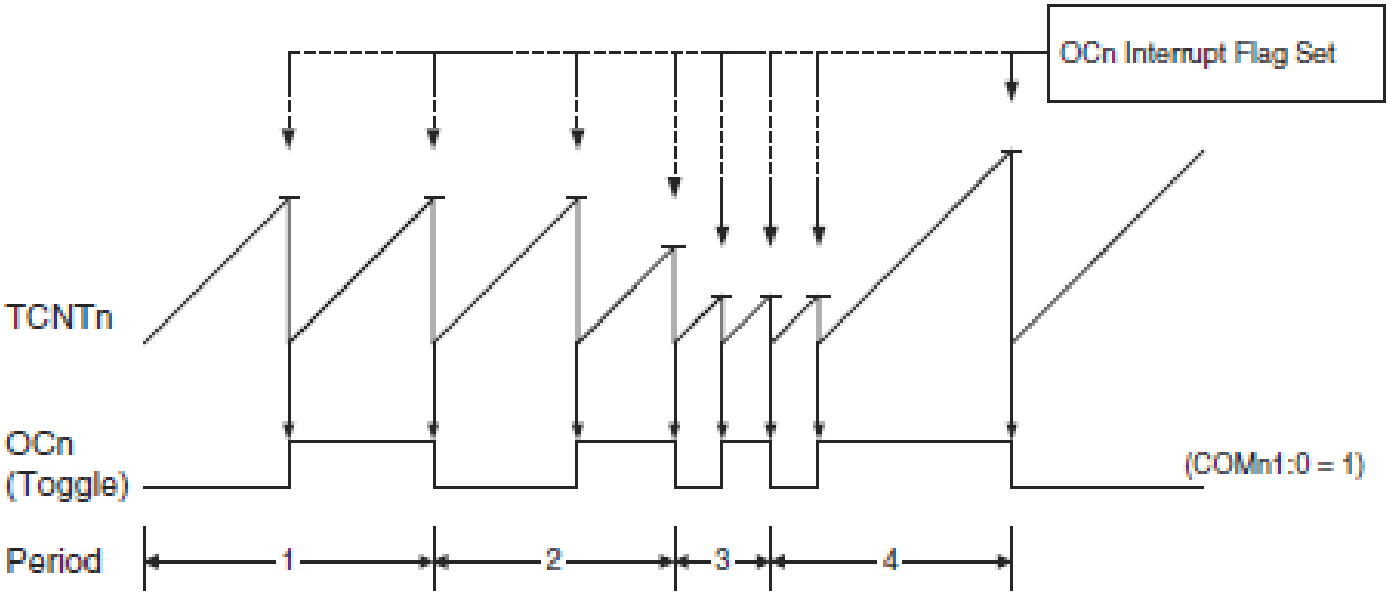
Normal모드는 TCNTn이 0부터 255까지 증가하고 256이 될 때 Overflow Interrupt가 요청된다.(업 카운트, 8bit 타이머일 때). 그 후, TCNTn은 0이 되고 다시 1씩 증가한다.

CTC모드는 TCNTn이 OCRn 레지스터의 값까지만 증가하고, TCNTn과 OCRn이 같아지면 Compare Match Interrupt가 요청된다. 그 후, TCNTn은 0부터 다시 증가.
또한, 비교매치때 Ocn핀을 토글시켜 출력할 수 있다.

- **Timer/Counter 동작시키기 (8bit)**

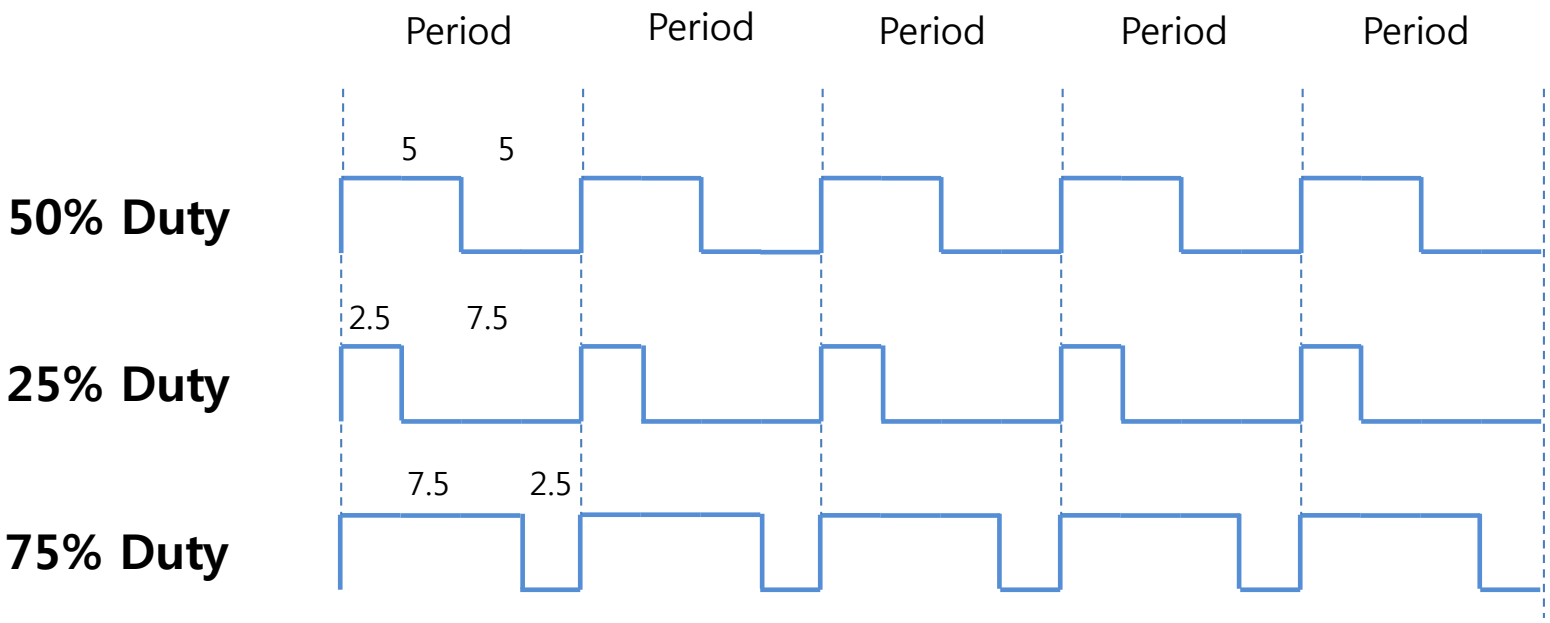
2. Clear Timer on Compare Match (CTC) Mode

Figure 38. CTC Mode, Timing Diagram



- **Timer/Counter 동작시키기 (8bit)**

- PWM이란?
PWM(Pulse Width Modulation)이란, 펄스폭 변조로 만들어지는 신호로서, 한 주기동안 H와 L의 비율(Duty rate)을 변화시켜 변조하는 방식이다.



- **Timer/Counter 동작시키기 (8bit)**

3. Fast PWM Mode

Timer/Counter를 이용하여 PWM신호를 출력할 수 있는 2가지 모드가 존재한다.

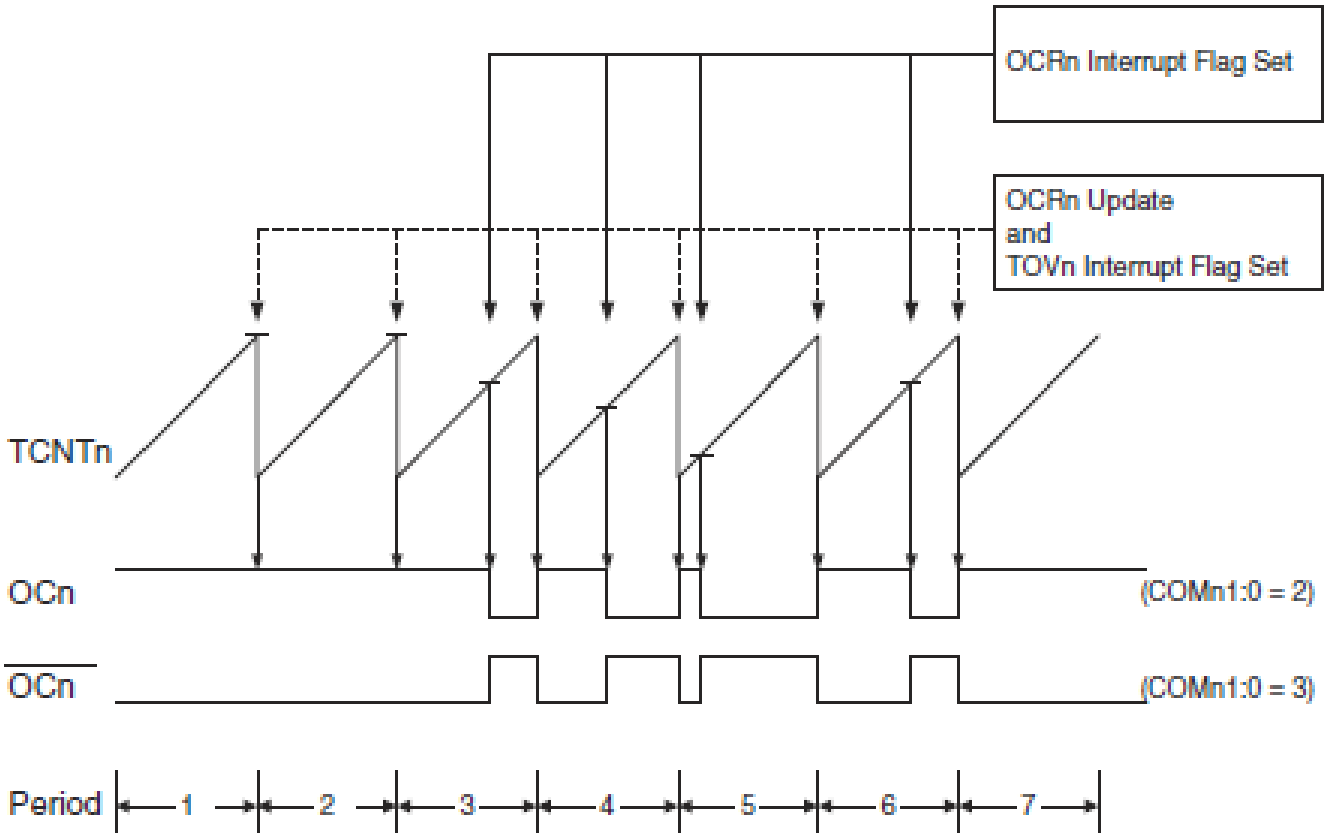
Fast PWM 모드를 시작하면 OCn핀이 H(+5V)출력으로 시작된다. TCNTn이 증가하다가 OCRn과 같아지면 Ocn핀이 L(0V)로 떨어지고 TCNTn이 Overflow가 일어나서 0이 되면 다시 Ocn핀에서 H가 출력된다 (COMn1:0 = 2일때). 이후 반복.

즉, PWM의 한 주기는 TCNTn이 0~255(8bit일때)까지 증가하는데 걸리는 시간이 되며, Duty는 0~OCRn까지 증가하는데 걸리는 시간과 OCRn~255까지 걸리는 시간의 비율로 계산될 수 있다.

- **Timer/Counter 동작시키기 (8bit)**

3. Fast PWM Mode

Figure 39. Fast PWM Mode, Timing Diagram



- **Timer/Counter 동작시키기 (8bit)**

4. Phase correct PWM Mode

Phase correct PWM 모드는 TCNTn이 0부터 255까지 증가한 후, 다시 255부터 0까지 감소한다(반복).

Up 카운트시 OCRn과 같아지면 Ocn핀은 L가 출력되고,

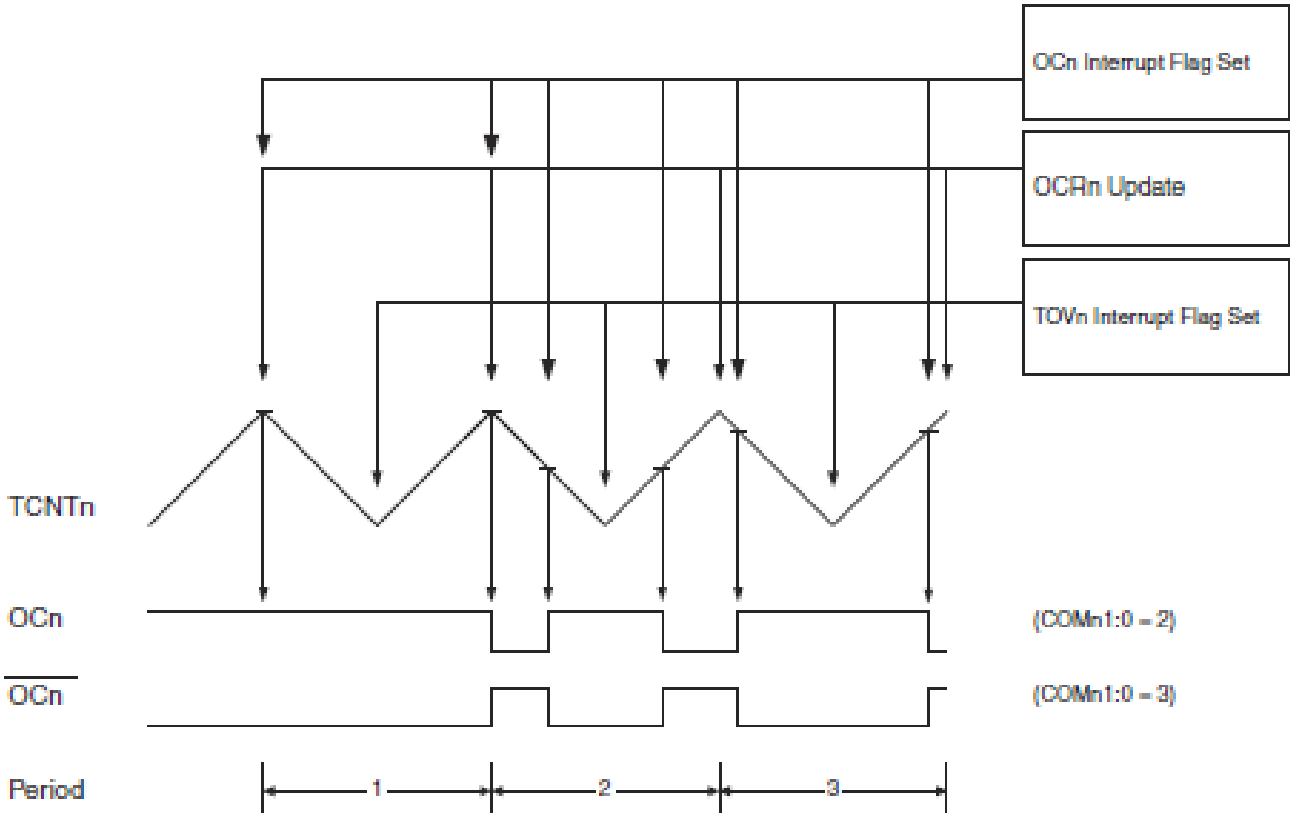
Down 카운트시 OCRn과 같아지면 Ocn핀은 H가 출력된다.

(COMn1:0 = 2일때)

- **Timer/Counter 동작시키기 (8bit)**

4. Phase correct PWM Mode

Figure 40. Phase Correct PWM Mode, Timing Diagram



6강 PWM with Timer/Counter

PWM 모드는 Fast PWM 모드, Phase Correct PWM 모드 이렇게 두 가지가 존재한다.

이 둘의 큰 차이는

1. Fast PWM 모드 – 고주파의 PWM을 생성할 수 있다.
2. Phase Correct PWM 모드 – 저주파의 PWM을 생성할 수 있다.

여기서 분해능(Resolution)의 의미는 **PWM의 한 주기안에 PWM Duty 비율을 얼마만큼 정밀하게 조절 가능한 것인가**를 의미한다.
(앞으로 분해능(Resolution)이란 말을 여러 번 더 듣게 될 텐데 그때마다 다른 의미로 해석될 수있다.)

PWM 모드를 선택할 때, 원하는 주기와 분해능에 따라 선택해주면 된다.

본 강의에서는 16bit 타이머1의 Phase Correct PWM 모드를 예로 들어 설명하겠다.
(ATmega128에는 총 4개의 타이머가 존재하고, 각각의 타이머는 모두 기본모드, CTC 모드, Fast PWM모드, Phase Correct PWM모드로 동작 가능하다.)

데이터시트 133 페이지를 참고.

6강 PWM with Timer/Counter

- 데이터시트 133페이지 부터

Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter1 Control Register C – TCCR1C

Bit	7	6	5	4	3	2	1	0	
	FOC1A	FOC1B	FOC1C	–	–	–	–	–	TCCR1C
Read/Write	W	W	W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

6강 PWM with Timer/Counter

- 데이터시트 133페이지 부터

Timer/Counter1 – TCNT1H and TCNT1L

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register 1 A – OCR1AH and OCR1AL

Bit	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	OCR1A[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register 1 B – OCR1BH and OCR1BL

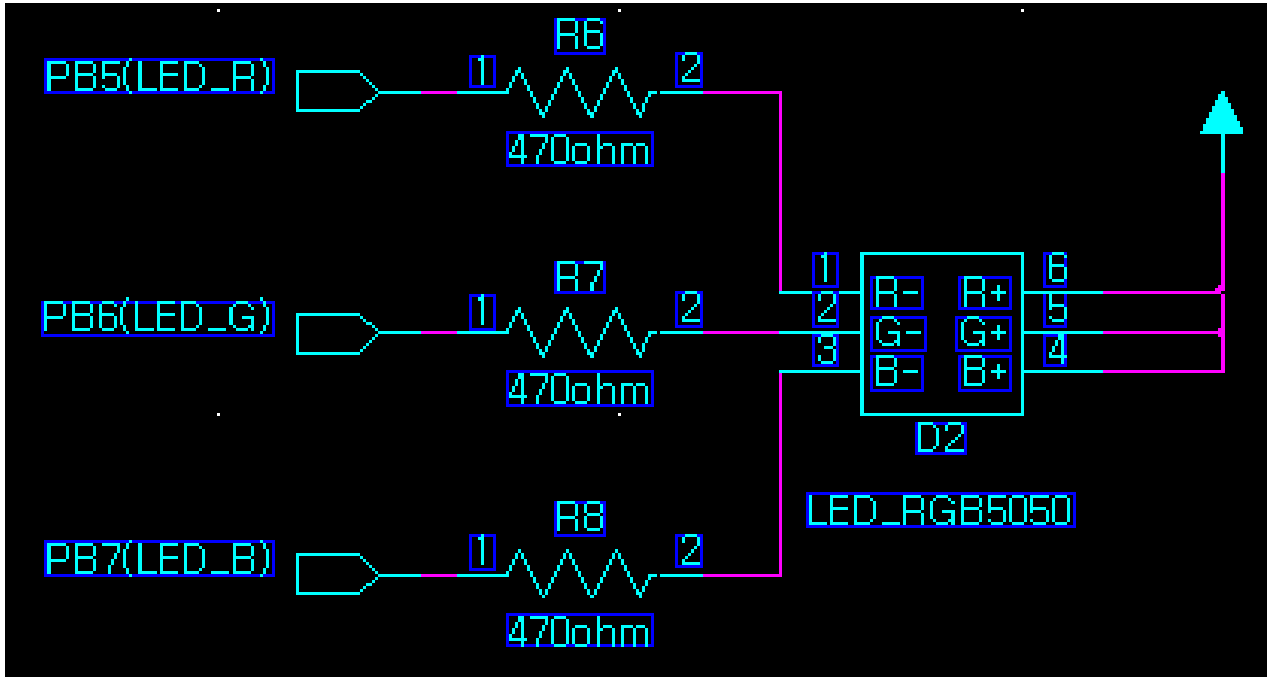
Bit	7	6	5	4	3	2	1	0	
	OCR1B[15:8]								OCR1BH
	OCR1B[7:0]								OCR1BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register 1 C – OCR1CH and OCR1CL

Bit	7	6	5	4	3	2	1	0	
	OCR1C[15:8]								OCR1CH
	OCR1C[7:0]								OCR1CL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

6강 PWM with Timer/Counter

다음과 같이 회로를 구성



RGB LED를 연결하여 회로를 구성해보자. 위의 회로에서 사용한 LED는 6핀 RGB LED이다.

위와 같이 LED를 각각 PB5(OC1A), PB6(OC1B), PB7(OC1C)에 연결한다.
위와 같이 연결하면 Low신호를 출력할때 LED가 점등된다.

6강 PWM with Timer/Counter

다음과 같은 코드를 작성

```
main()
{
    /**   GPIO 입출력 설정   **/
    DDRB = 0xf0; //PB4에 LED 연결, PB5에 LED_R, PB6에 LED_G, PB7에 LED_B 연결
    PORTB = 0x00; //B포트 출력 초기화
    /************************/

    // Timer/Counter 1 initialization
    // Clock value: 15.625 kHz
    // Mode: Ph. correct PWM top=03FFh
    // OC1A output: Non-Inv.
    // OC1B output: Non-Inv.
    // OC1C output: Non-Inv.
    TCCR1A = 0xAB;
    TCCR1B = 0x05;
    TCCR1C = 0x00;
    TCNT1H = 0x00;
    TCNT1L = 0x00;
    OCR1AH = 0x00;
    OCR1AL = 0x00;
    OCR1BH = 0x00;
    OCR1BL = 0x00;
    OCR1CH = 0x00;
    OCR1CL = 0x00;

    while(1)
    {
        OCR1AH = 0x00;
        OCR1AL = 0xff; //Duty 25%. 최대 0x3ff(1023)

        OCR1BH = 0x01;
        OCR1BL = 0xff; //Duty 50%. 최대 0x3ff(1023)

        OCR1CH = 0x02;
        OCR1CL = 0xff; //Duty 75%. 최대 0x3ff(1023)
    }
}
```

레지스터 설정의 의미를
한번 생각해보자.

이와 같이 설정하면
Timer/Counter1를
Phase Correct PWM모드로
사용할 수 있게 된다!

값을 쓸 때는 H를 먼저 쓰고
L를 나중에 써야 한다!

6강 PWM with Timer/Counter

다음과 같은 코드를 작성

```
main()
{
      /**   GPIO 입출력 설정  **/
      DDRB = 0xf0; //PB4에 LED 연결, PB5에 LED_R, PB6에 LED_G, PB7에 LED_B 연결
      PORTB = 0x00; //B포트 출력 초기화
      /************************/

      // Timer/Counter 1 initialization
      // Clock value: 15.625 kHz
      // Mode: Ph. correct PWM top=03FFh
      // OC1A output: Non-Inv.
      // OC1B output: Non-Inv.
      // OC1C output: Non-Inv.
      TCCR1A = 0xAB;
      TCCR1B = 0x05;
      TCCR1C = 0x00;
      TCNT1H = 0x00;
      TCNT1L = 0x00;
      OCR1AH = 0x00;
      OCR1AL = 0x00;
      OCR1BH = 0x00;
      OCR1BL = 0x00;
      OCR1CH = 0x00;
      OCR1CL = 0x00;

      while(1)
      {
            OCR1AH = 0x00;
            OCR1AL = 0xff; //Duty 25%. 최대 0x3ff(1023)

            OCR1BH = 0x01;
            OCR1BL = 0xff; //Duty 50%. 최대 0x3ff(1023)

            OCR1CH = 0x02;
            OCR1CL = 0xff; //Duty 75%. 최대 0x3ff(1023)
      }
}
```

다음과 같이 레지스터를 변경해 주면,
PB5에 25% PWM
PB6에 50% PWM
PB7에 75% PWM
이 생성된다.

6강 PWM with Timer/Counter

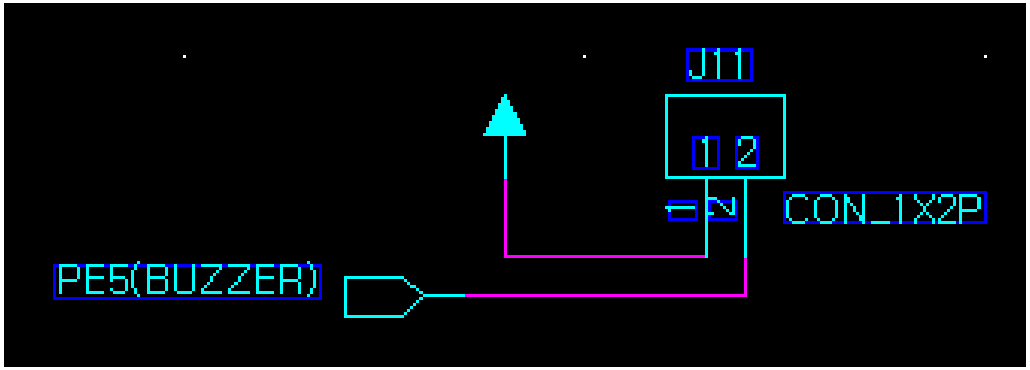
문1) RGB LED를 각각 부드럽게 색 변경을 하는 프로그램을 작성

문2) 랜덤함수를 사용해서 밝기를 랜덤하게 변경하는 프로그램 작성

문3) 1번 스위치를 입력 받아 LED 색 선택, 2번 스위치를 입력 받아 밝기 제어

6강 PWM with Timer/Counter

다음과 같이 회로를 구성



위와 같이 피에조 Buzzer를 PE5(OC3C) 에 연결한다.
(부저는 방향성이 없다.)

6강 PWM with Timer/Counter


```
unsigned int tcnt3;
main()
{
      /**   GPIO 입출력 설정  **/
      DDRB = 0xf0; //PB4에 LED 연결, PB5에 LED_R, PB6에 LED_G, PB7에 LED_B 연결
      PORTB = 0x00; //B포트 출력 초기화

      DDRE=0x20; //PE5에 피에조 Buzzer 연결
      PORTE=0x00;
      /************************/

      // Timer/Counter 3 initialization
      // Clock source: System Clock
      // Clock value: 65.5 kHz
      // Mode: Normal top=FFFFh
      TCCR3A=0x00;
      TCCR3B=0x04;
      TCNT3H=0x00;
      TCNT3L=0x00;
      OCR3AH=0x00;
      OCR3AL=0x00;
      OCR3BH=0x00;
      OCR3BL=0x00;
      OCR3CH=0x00;
      OCR3CL=0x00;
      ETIMSK=0x04;  SREG=0x80;
      while(1)
      {

      }
}

interrupt [TIM3_OVF] void tim3_ovf(void)
{
      PORTE ^= 0x20; //Toggle
      TCNT3H = tcnt3>>8;
      TCNT3L = tcnt3;
}
```

레지스터 설정의 의미를
한번 생각해보자.

이와 같이 설정하면
Timer/Counter3를
일반모드로
사용할 수 있게 된다!

또한 Timer/Counter3의
오버플로우와 전역인터럽트를
활성화 시킨다!!

6강 PWM with Timer/Counter


```
unsigned int tcnt3;
main()
{
      /**   GPIO 입출력 설정   **/
      DDRB = 0xf0; //PB4에 LED 연결, PB5에 LED_R, PB6에 LED_G, PB7에 LED_B 연결
      PORTB = 0x00; //B포트 출력 초기화

      DDRE=0x20; //PE5에 피에조 Buzzer 연결
      PORTE=0x00;
      /************************/

      // Timer/Counter 3 initialization
      // Clock source: System Clock
      // Clock value: 65.5 kHz
      // Mode: Normal top=FFFFh
      TCCR3A=0x00;
      TCCR3B=0x04;
      TCNT3H=0x00;
      TCNT3L=0x00;
      OCR3AH=0x00;
      OCR3AL=0x00;
      OCR3BH=0x00;
      OCR3BL=0x00;
      OCR3CH=0x00;
      OCR3CL=0x00;
      ETIMSK=0x04;  SREG=0x80;
      while(1)
      {

      }
}

interrupt [TIM3_OVF] void tim3_ovf(void)
{
      PORTE ^= 0x20; //Toggle
      TCNT3H = tcnt3>>8;
      TCNT3L = tcnt3;
}
```

다음 코드의 의미를
생각해보자!

6강 PWM with Timer/Counter

문1) 주파수를 가변하여 부저의 소리를 조절하는 프로그램 작성

문2) 100Hz~10kHz까지 100Hz 단위로 가변하는 프로그램을 작성

문3) 피아노 음계 도~시까지의 음계를 각각 주파수로 저장 후 재생