2023년 IoT기반 스마트 솔루션 개발자 양성과정

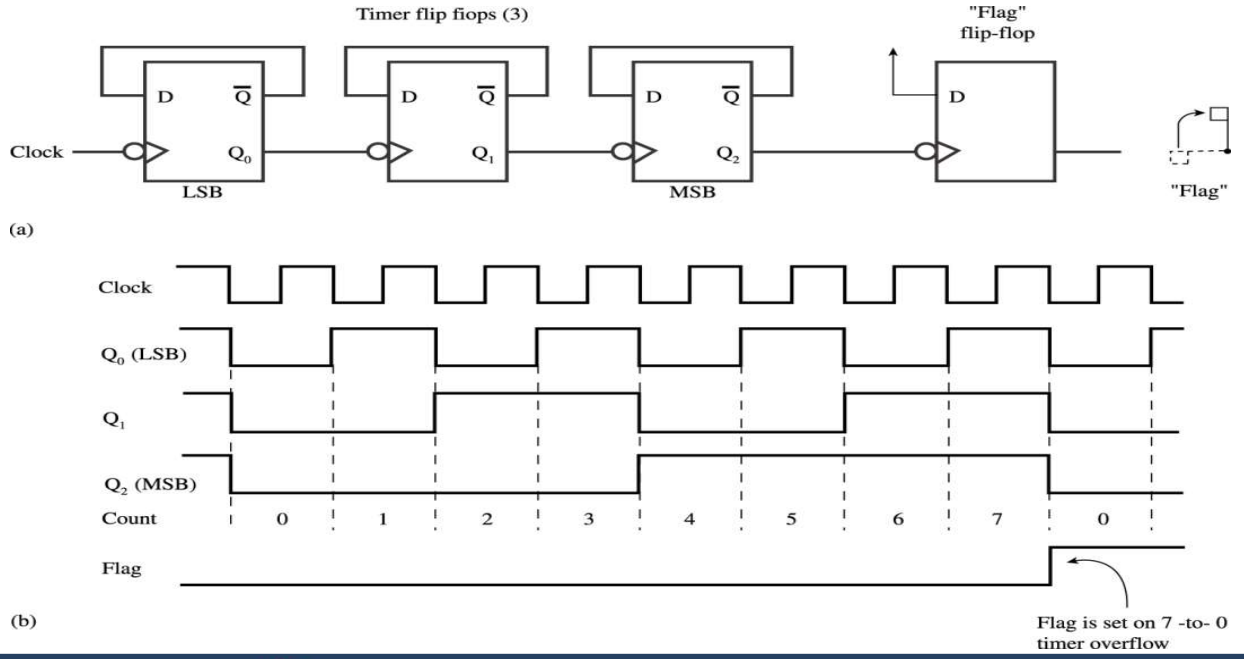
Embedded Application

14-Timer/Counter

담당 교수 : 윤 종 이
010-9577-1696
ojo1696@naver.com
https://cafe.naver.com/yoons2023

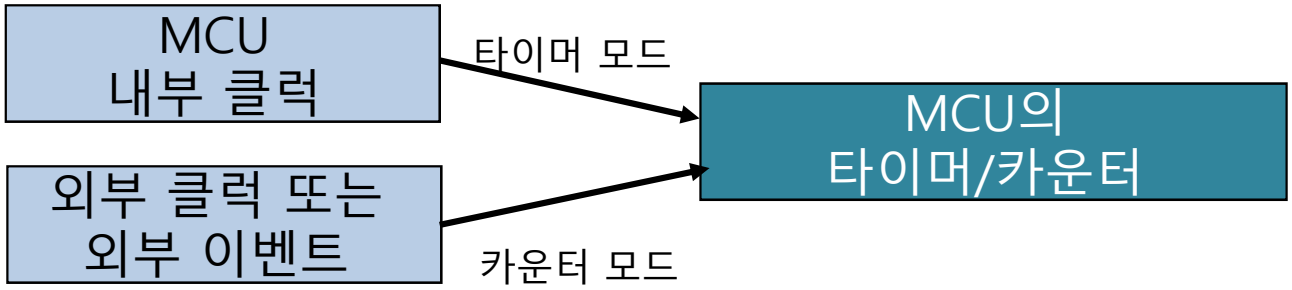
타이머/카운터

- 클럭이 입력되면 들어오는 클럭을 계수하는 기능
- 타이머/카운터의 각 단계는 입력 클럭을 2분주하여 동작하는 하나의 D 플립플롭 (D-type negative edge-triggered)으로 되어 있으며, D 플립플롭의 출력은 다음 단의 D 입력에 연결되어 있음.
- 오버플로우 : 3-비트 타이머/카운터의 경우 카운터의 상태가 111에서 000으로 변화할 때 발생함.



타이머/카운터 모드

- 타이머 모드 : MCU의 내부 클럭을 입력으로 사용하여 이를 분주해서 클럭 소스로 사용하는 경우
- 카운터 모드 : MCU의 외부에서 입력되는 클럭 신호를 받아서 이를 클럭 소스로 사용하는 경우



타이머/카운터의 기능

타이머/카운터	타이머/카운터 0	타이머/카운터 1	타이머/카운터 2	타이머/카운터 3
비트 수	8비트	16비트	8비트	16비트
카운터 입력	TOSC1	T1	T2	T3
관련 레지스터	TCCR0 TCNT0 OCR0 ASSR SFIOR TIMSK TIFR	TCNT1,TCCR1A, TCCR1B,TCCR1C, TCNT1H,TCNT1L, OCR1AH,OCR1AL, OCR1BH,OCR1BL, OCR1CH,OCR1CL ICR1H, ICR1L SFIOR TIMSK, ETIMSK, TIFR,ETIFR	TCCR2 TCNT2 OCR2 SFIOR TIMSK TIFR	TCNT3,TCCR3A, TCCR3B,TCCR3C, TCNT3H, TCNT3L, OCR3AH,OCR3AL, OCR3BH,OCR3BL, OCR3CH,OCR3CL ICR3H, ICR3L SFIOR TIMSK, ETIMSK, TIFR,ETIFR

8비트 타이머/카운터0

- 타이머/카운터0 은 PWM 출력을 가지고 있는 8비트 타이머/카운터
- 프리스케일러를 통하여 내부/외부 클럭을 입력으로 사용하여 동작하는 타이머/카운터 기능을 수행

Timer/Counter 0 Register

타이머/카운터0 레지스터	설 명
TCCR0	Timer/Counter 0 제어 레지스터
TCNT0	Timer/Counter 0 레지스터
OCR0	출력 비교 레지스터
ASSR	비동기 상태 레지스터
SFIOR	특수 기능 I/O 레지스터
TIMSK	타이머/카운터 인터럽트 마스크 레지스터
TIFR	타이머/카운터 인터럽트 플래그 레지스터

TCCR0 Register

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	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	

- 비트7: FOC0(강제 출력 비교,Force Output Compare)
  - PWM 모드가 아닌 경우에만 유효, 이를 1로 설정하면 강제로 즉시 OC0 단자에 출력 비교가 일치된 것과 같은 출력을 내보내는 기능을 함
- 비트 6,3 : WGM01~WGM00(파형 발생 모드, Waveform Generation Mode)
  - 카운터의 계수 순서와 카운터의 최대값, 파형 발생의 형태 등의 카운터의 동작 모드를 설정
  - 카운터/타이머0는 일반 모드, CTC모드, PWM모드, 고속PWM모드가 있음

TCCR0 : Clock Select

타이머/카운터0에서의 CS02~CS00비트에 의한 클럭 선택

CS02	CS01	CS00	클럭 소스의 기능
0	0	0	클럭 소스 차단(타이머/카운터 기능이 정지)
0	0	1	ClK TOS
0	1	0	ClK TOS/8
0	1	1	ClK TOS/32
1	0	0	ClK TOS/64
1	0	1	ClK TOS/128
1	1	0	ClK TOS/256
1	1	1	ClK TOS/1024

TCCR0 : Timer/Counter Mode

WGM01~00비트에 의한 파형 발생모드의 설정

모드	WGM01 (CTC)	WGM00 (PWM)	동작모드	최대값	OCR0레지스터의 업데이트 시기	TOV0 플래그의 세트 시점
0	0	0	일반	0xFF	설정 즉시	MAX
1	0	1	PWM Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCRn	설정 즉시	MAX
3	1	1	고속 PWM	0xFF	TOP	MAX

TCNT0 Register

- TCNT0(Timer/Counter Register 0) 레지스터 :
  타이머/카운터0의 8비트 카운터 값을 저장하는 레지스터

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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	

OCR0 Register

- OCR0(Timer/Counter Output Compare Register 0)레지스터 : TCNT0값과 비교하여 OC0단자에 출력 신호를 발생하기 위해 8비트의 값을 저장하는 레지스터

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
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	

TIMSK Register

- TIMSK(timer/Counter Interrupt Mask Register)레지스터

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
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	

- 비트1: TOIE0(타이머/카운터 오버플로우 인터럽트 허가 비트)
- 비트0 : OCIE0 (타이머/카운터0 출력 비교 인터럽트 허가 비트)

TIFR Register

- TIFR(Timer/Counter Interrupt Flag Register) 레지스터

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
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	

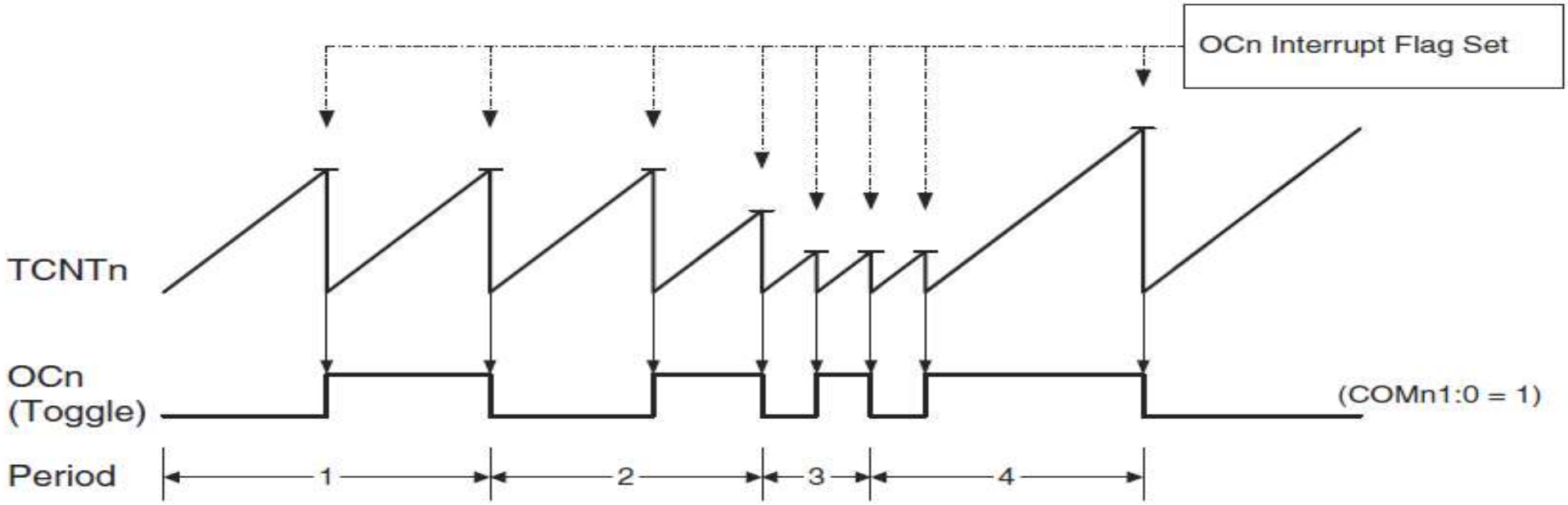
- 비트 1 : TOV0(타이머/카운터 0 오버플로우 플래그)
- 비트 0 : OCF0 (출력 비교 플래그 0)

Normal Mode

- TCNT0은 항상 상향 카운터로만 동작
- 타이머/카운터0의 값이 0xFF에서 0x00으로 바뀌는 순간에 TIFR레지스터의 TOV0비트가 1로 세트 되면서 오버플로우 인터럽트가 발생하며 인터럽트 서비스 루틴을 수행 자동으로 TOV0비트는 0으로 리셋

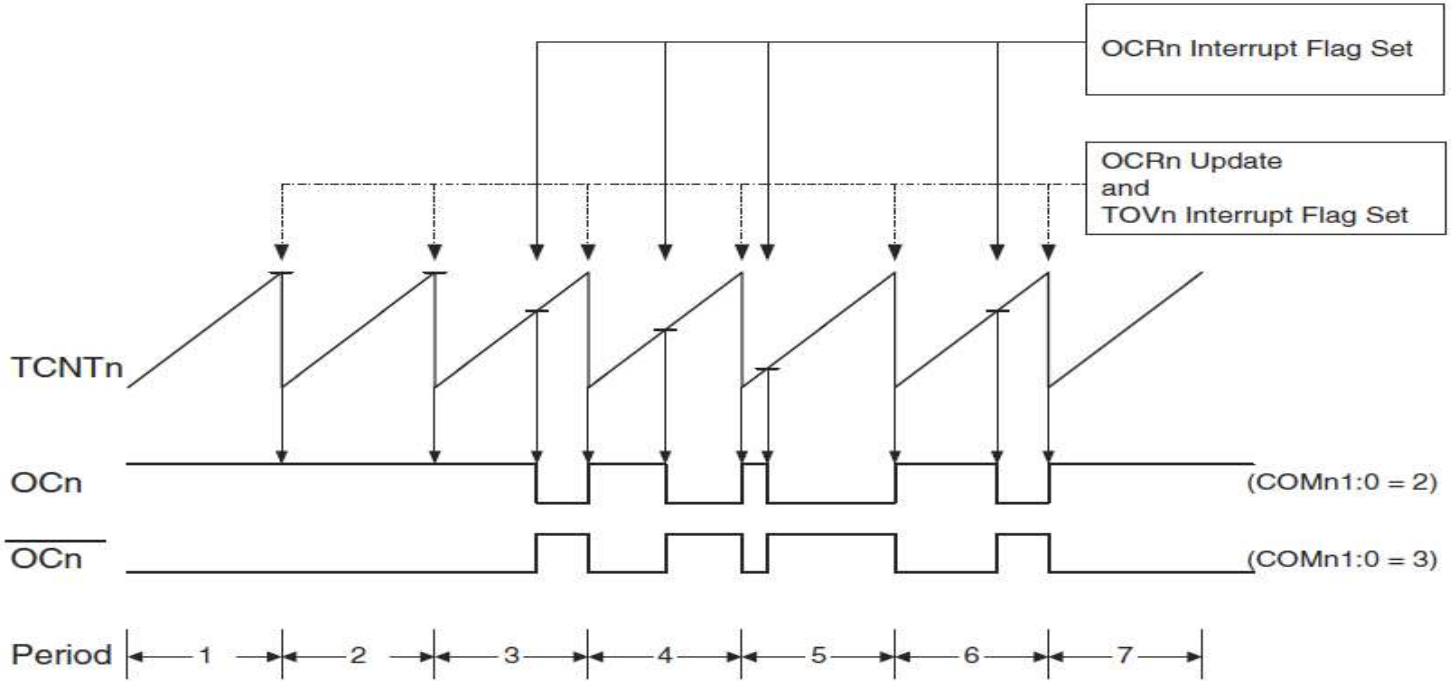
CTC Mode

- Clear Timer on Compare Match (CTC) Mode
- TCNT0의 값이 클럭의 입력에 따라서 증가하여 출력 비교 레지스터OCR0의 값과 같아지면 그 다음 클럭 사이클에서 0으로 클리어 됨



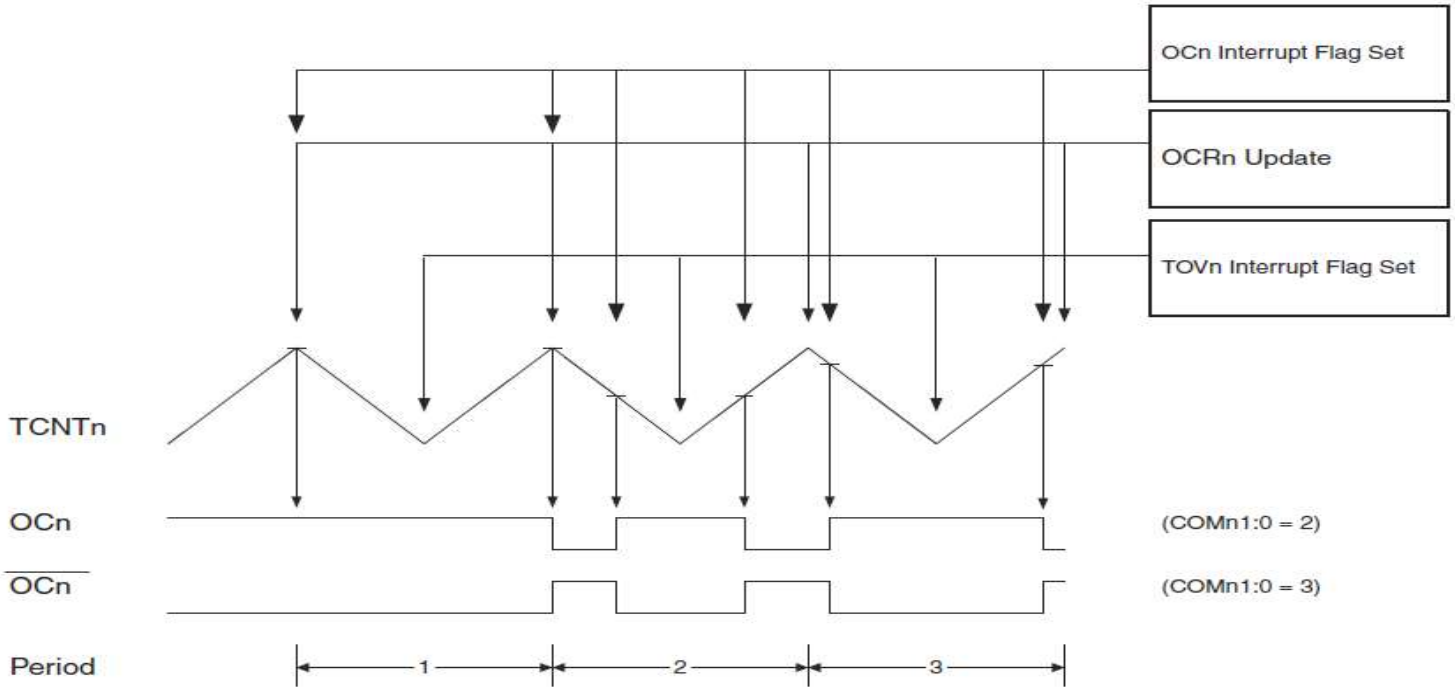
Fast PWM Mode

- TCNT0 레지스터의 값이 항상 bottom에서 시작하여 max까지 증가하는 방향으로만 반복하여 수행됨. : 단방향 경사 동작(single-slope operation)
- COM01~COM00 비트의 설정에 따라 비반전 비교 출력 모드와 반전 비교 출력 모드로 동작



Phase Correct PWM Mode

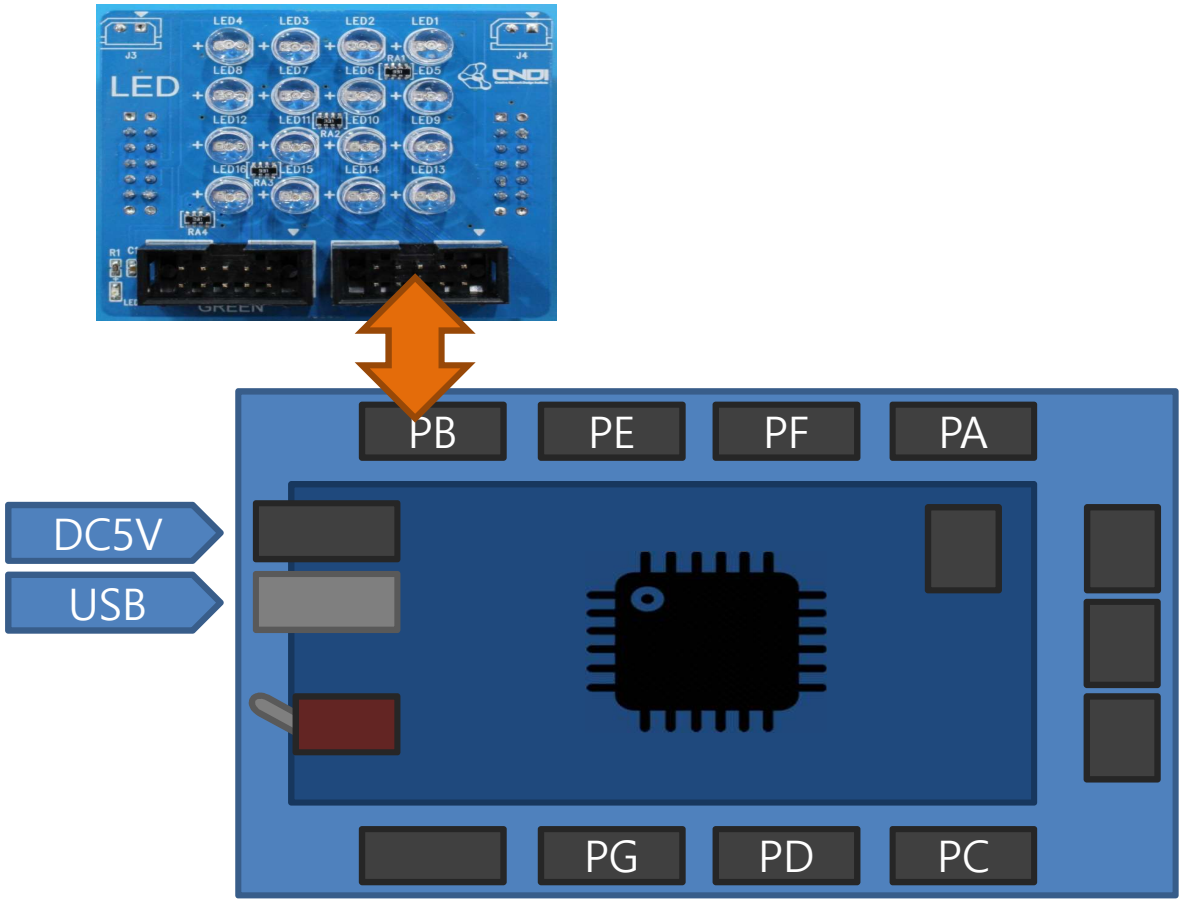
- 높은 분해능의 PWM 출력 파형을 발생하는데 유용하게 사용됨.
- TCNT0가 상향 카운터로서 bootom(0x00)에서 max(0xFF)까지 증가하였다가 다시 하향 카운터로서 max서 bootom으로 감소를 하는 동작을 반복하여 수행.(양방향 동작(dual-slope operation))



Ex-1 : LED 회전

- 10mSec로 회전하는 LED를 구현해보자

Ex-1 : Wiring


LED
LED4
LED3
LED2
LED1
LED8
LED7
LED6
LED5
LED12
LED11
LED10
LED9
LED16
LED15
LED14
LED13
RA1
RA2
RA3
RA4
J3
J4
GREEN
PB
PE
PF
PA
DC5V
USB
PG
PD
PC

Ex-1 : Define/sub

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define LED_RED PORTB
unsigned char LED[8]={0x08,0x04,0x02,0x01,0x10,0x20,0x40,0x80};
unsigned char Count=0;

ISR(TIMER0_OVF_vect) {
    LED_RED=LED[Count];
    if (++Count>7) Count=0;
    TCNT0=0xff-144;
}

void Timer0_Init(){
    TCCR0=0x07;                    //Clk / 1024
    TIMSK=1<<TOIE0;                //Overflow Interrupt Enable
    TIFR=1<<TOV0;                  //Set Overflow Interrupt Flag
    TCNT0=0xff-144;                //((1/14745600)*1024)*144=0.01sec
}
```

Ex-1 : main

```
void CPU_Setup(){
    DDRB=0xff;
}

int main(void){
    CPU_Setup();
    Timer0_Init();
    sei();

    while (1) {

    }
}
```

Ex-2 : LED 회전

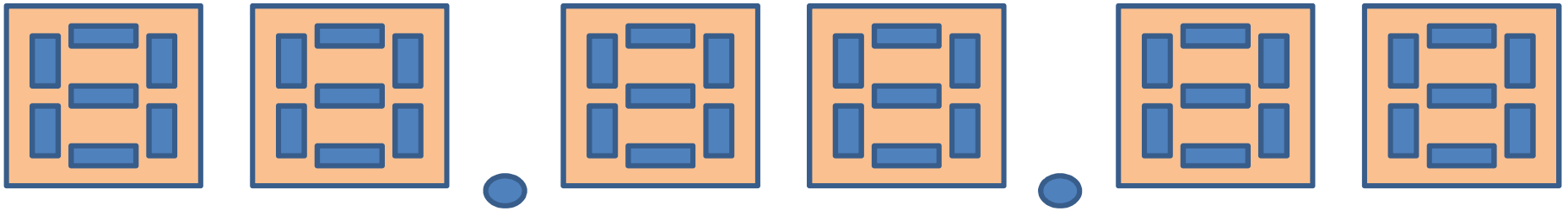
- 1 Sec로 회전하는 LED를 구현해보자

Ex-2 : Define

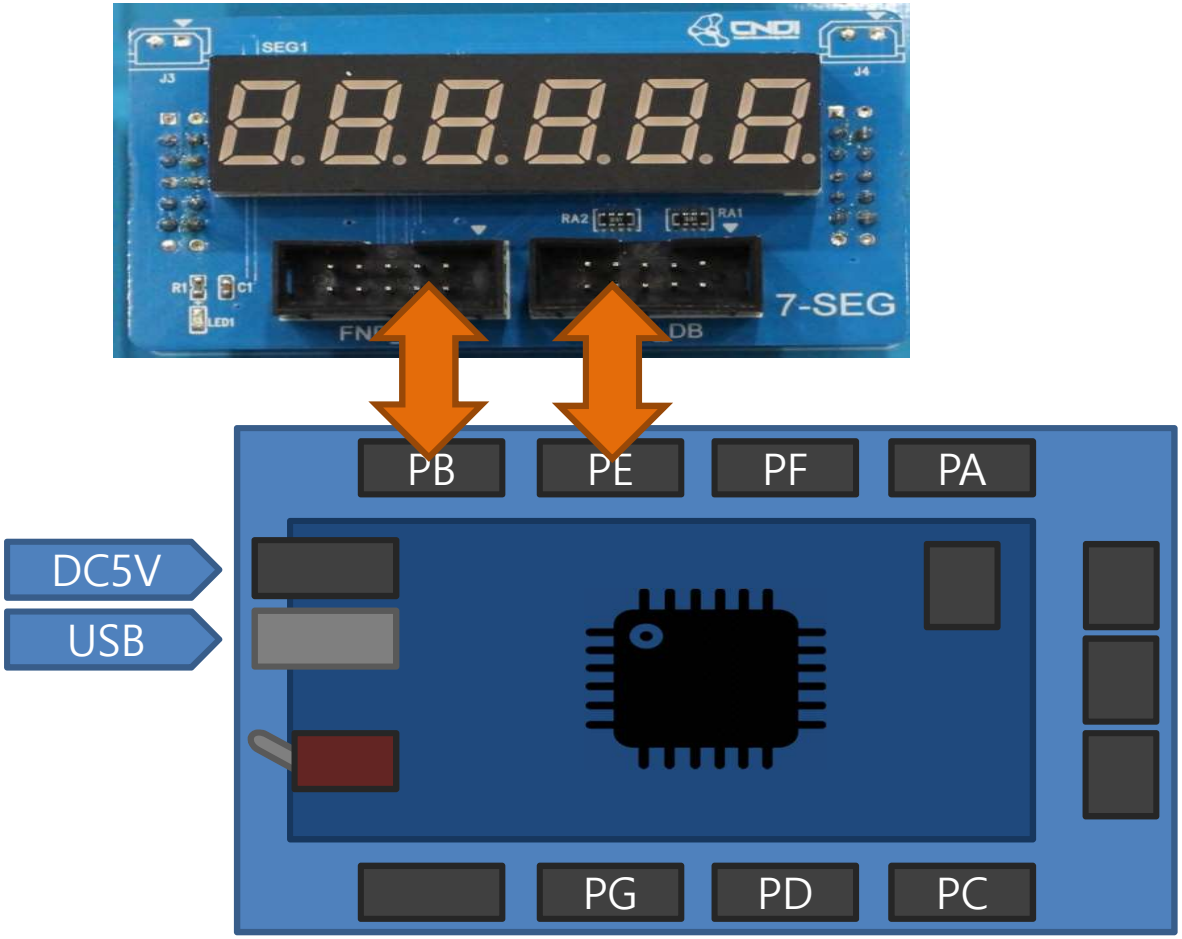
```
#include <avr/io.h>
#include <avr/interrupt.h>

#define LED_RED PORTB
unsigned char LED[8]={0x08,0x04,0x02,0x01,0x10,0x20,0x40,0x80};
unsigned char Count=0;
unsigned char mSec=0;

ISR(TIMER0_OVF_vect) {
   if (++mSec>=100){
        mSec=0;
        LED_RED=LED[Count];
        if (++Count>7) Count=0;
   }
   TCNT0=0xff-144;
}
```

Ex-3 : Timer

- 다음 동작의 시간을 표시하는 타이머를 만들어 보자



Ex-3 : Wiring



Ex-3 : Define

```
#define F_CPU 14745600UL
#define FND_SEL PORTB
#define FND_DB PORTE
#define dTime 3
#define tConst 144                    //((1/14745600)*1024)*144=0.01sec

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

unsigned char FND[17]={0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x27, 0x7f, 0x6f, 0x77, 0x7c, 0x58,
0x5e, 0x40, 0x49, 0x40};
unsigned char DGT[6]={0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf};
unsigned char NUM[6]={0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

unsigned char Min=0;
unsigned char Sec=0;
unsigned char mSec=0;
```

Ex-3 : Timer

```
ISR(TIMER0_OVF_vect) {
    if (++mSec>99){
        mSec=0;
        if (++Sec>59){
            Sec=0;
            if (++Min>59) Min=0;
        }
     }
    TCNT0=0xff-tConst;
}

void Timer0_Init(){
    TCCR0=0x07;                 //Clk / 1024
    TIMSK=1<<TOIE0;             //Overflow Interrupt Enable
    TIFR=1<<TOV0;               //Set Overflow Interrupt Flag
    TCNT0=0xff-tConst;
}
```

Ex-3 : Display

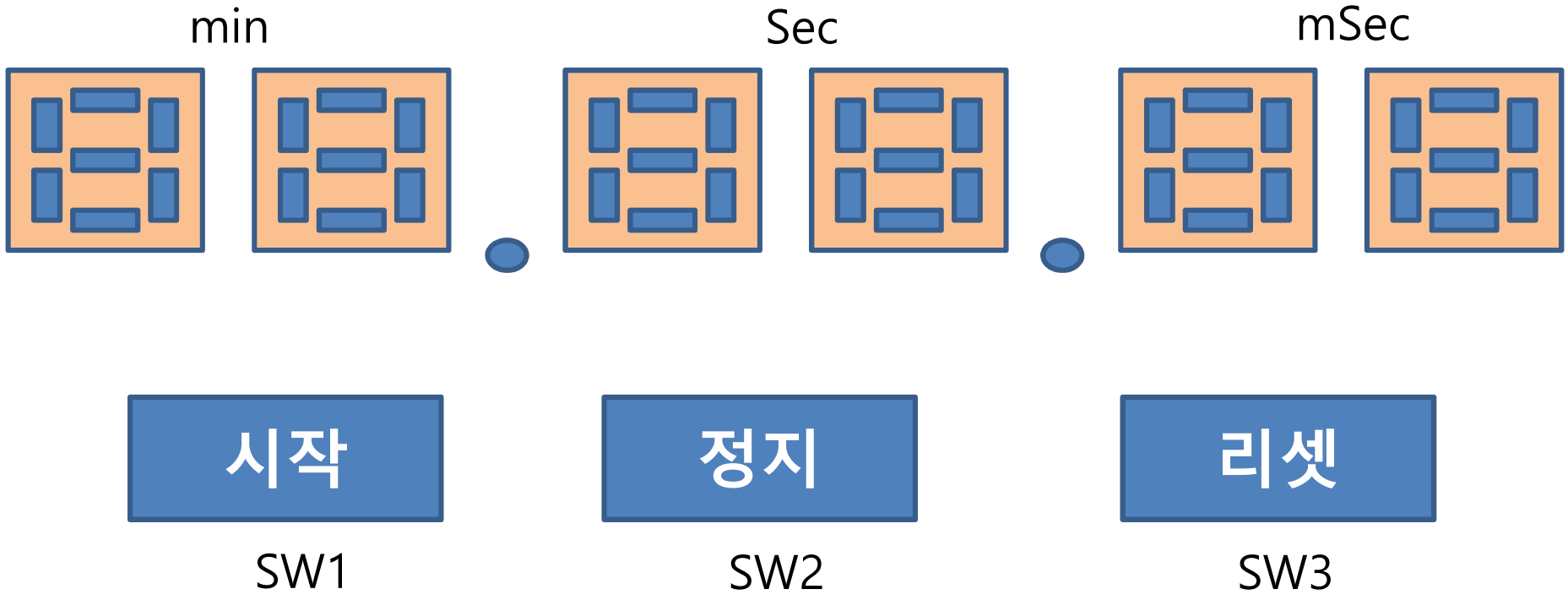
```
void Hex2Dec(){
    unsigned char tmp=Min;
    NUM[5]=tmp/10;
    NUM[4]=tmp%10;

    tmp=Sec;
    NUM[3]=tmp/10;
    NUM[2]=tmp%10;

    tmp=mSec;
    NUM[1]=tmp/10;
    NUM[0]=tmp%10;
}

void FND_Display(){
    for (unsigned char k=0; k<6; k++) {
        FND_SEL=DGT[k];
        FND_DB=FND[ NUM[k] ];
        _delay_ms(dTime );
    }
}
```

Ex-3 : main

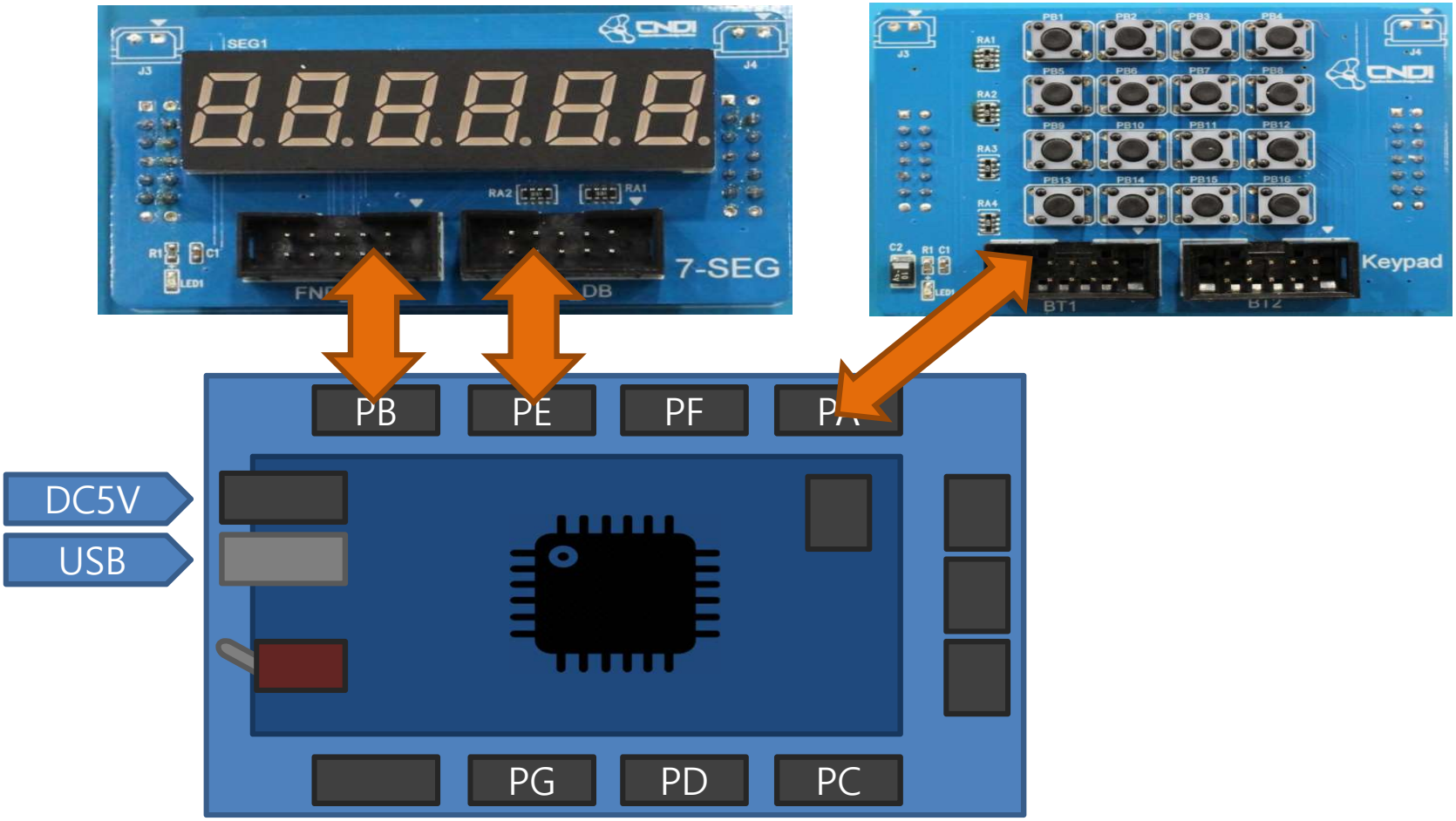
```
void CPU_Setup( ) {
    DDRB=0xff;
    DDRE=0xff;
    DDRF=0xF0;
}

int main(void) {
    CPU_Setup();
    Timer0_Init();
    sei();

    while (1)   {
        Hex2Dec();
        FND_Display();
    }
}
```

Ex-4 : Stop Watch

- 다음 동작의 시간을 표시하는 Stop Watch를 만들어 보자



Ex-4 : Wiring



Ex-4 : Program