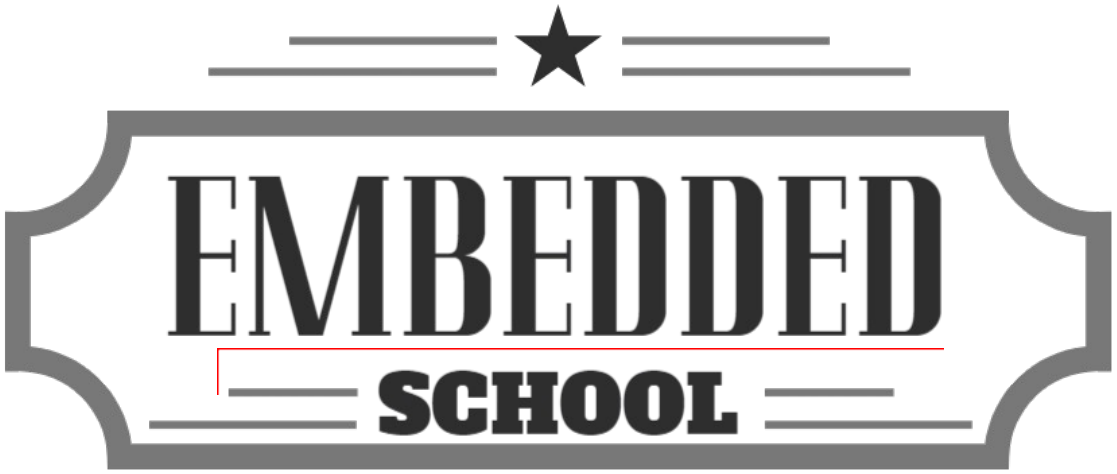


AVR - HW6

임베디드스쿨1기

Lv1과정

2020. 10. 23

박하늘

1. [Review] Timer/Counter External Interrupt

1) 외부 신호(T0)만 사용하여 카운팅 하는 예제

```c
#define F_CPU 16000000UL

#include <avr/io.h>
#include <avr/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <stdio.h>

#define sbi(PORTX, BitX) (PORTX |= (1 << BitX))
#define cbi(PORTX, BitX) (PORTX &= ~(1 << BitX))

void EXT_falling_timer_Init(void);

int main(void)
{
    EXT_falling_timer_Init();
    while (1)

    {
        if(TCNT0 > 5)
        {
            PORTB = 0xff;
        }
    }
}

void EXT_falling_timer_Init(void)
{

    cbi(SREG, 7);
    TCCR0A = 0; //카운터 초기화
    TCCR0B = 0; //카운터 초기화
    PORTB = 0x00;
    PORTD = 0x10;
    DDRB = 0xff;
    DDRD = 0x10;
    TCCR0B = (1 << CS02) | (1 << CS01); //EXTERNAL FALLING
    sbi(SREG,7);
}
```

1. 버튼을 누르면 TCNT0 카운트 올라간다.
2. 스위치(4번): PD4 를 입력으로 사용
   출력 (13번): PB를 모두 출력으로 사용
3. 레지스터 설정
   - falling edge로 설정

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	**WGM02**	**CS02**	**CS01**	**CS00**	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

1. [Review] Timer/Counter Overflow Code

2) 내부 클럭 그대로 사용하여 Timer overflow감지

```c
#include <avr/io.h>
#include <avr/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <stdio.h>

#define sbi(PORTX, BitX) (PORTX |= (1 << BitX))
#define cbi(PORTX, BitX) (PORTX &= ~(1 << BitX))

unsigned char counter = 0;

void overflow_timer_init(void);

SIGNAL(TIMER0_OVF_vect)
{
    if(counter == 125)
    {
        PORTB = 0xff;
        counter = 0;
    }
    else{
        PORTB = 0x00;
        counter++;
    }
}
int main(void)
{
    overflow_timer_init();

    while(1)
    {
    }
}

void overflow_timer_init(void)
{
    cbi(SREG, 7);
    TCCR0A = 0; //카운터 초기화
    TCCR0B = 0; //카운터 초기화
    PORTB = 0x00;
    DDRB = 0xff;

    TCCR0B = (1 << CS02) | (1 << CS00); //EXTERNAL FALLING
    TCNT0 = 131; //1/clkt = 64us, 64us *125 = 8ms
    sbi(TIMSK0,TOIE0);
    sbi(SREG,7);
}
```

1. 8bit register 이므로 256까지 표현 가능

분주: 16M/1024 = 15625
1/15625 = 64us
64us * 125 = 8msec

2. TCNT0 = 131 ; //131에서 시작, 131~256(125)

3. 8msec마다 Interrupt동작

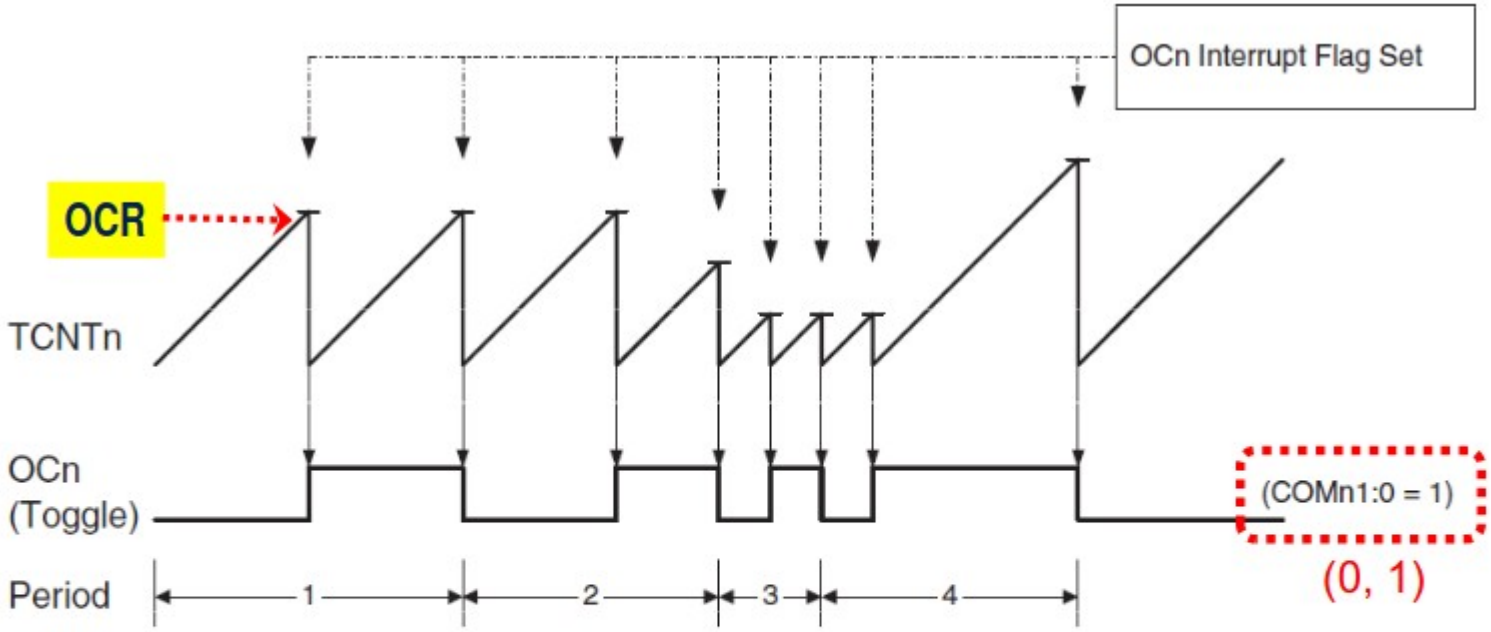
8*125 = 1000msec //1초마다 인터럽트 걸림, LED주기 2초

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

2-1. Timer/Interrupt - CTC Mode

■ Clear Timer on Compare Match

- BOTTOM 에서 설정된 OCR(MAX가 아님) 값과 같아지면 0으로 클리어되며 인터럽트 발생



✓ OCn 핀 출력신호의 주파수

$$f_{OCn} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

N : Prescaler 분주비

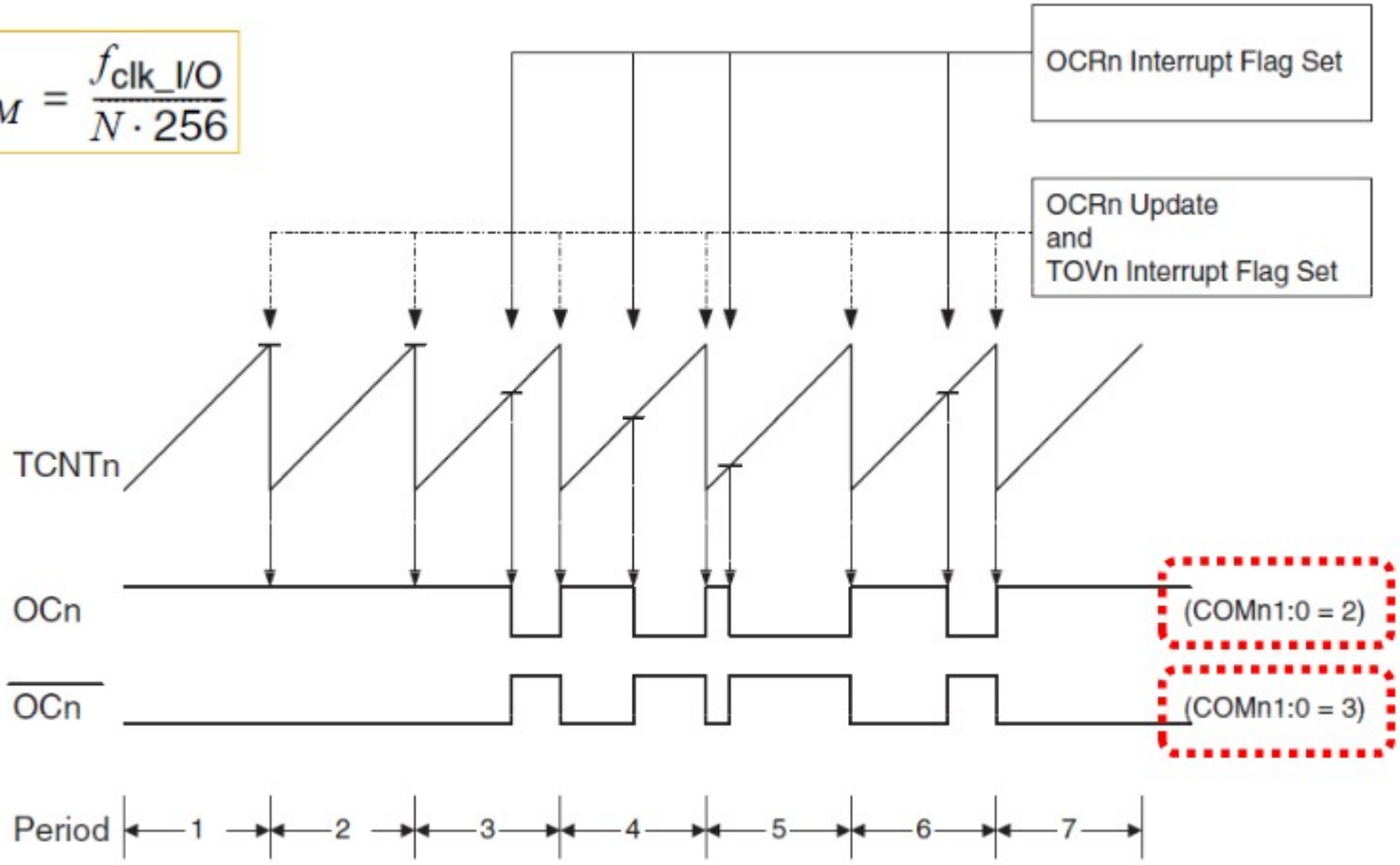
2: Toggle

→ TCNT의 counting개수를 조절하면 주기가 바뀜. 즉, 주기를 이용해서 duty 제어

2-2. Timer/Interrupt - fast PWM Mode

- Fast PWM Mode
  - BOTTOM → MAX로 단순 증가하다 OCR값과 비교해 같아지면 OC=0으로 클리어
  - COM 비트 설정에 따라 반전 비반전 출력 토글 가능

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

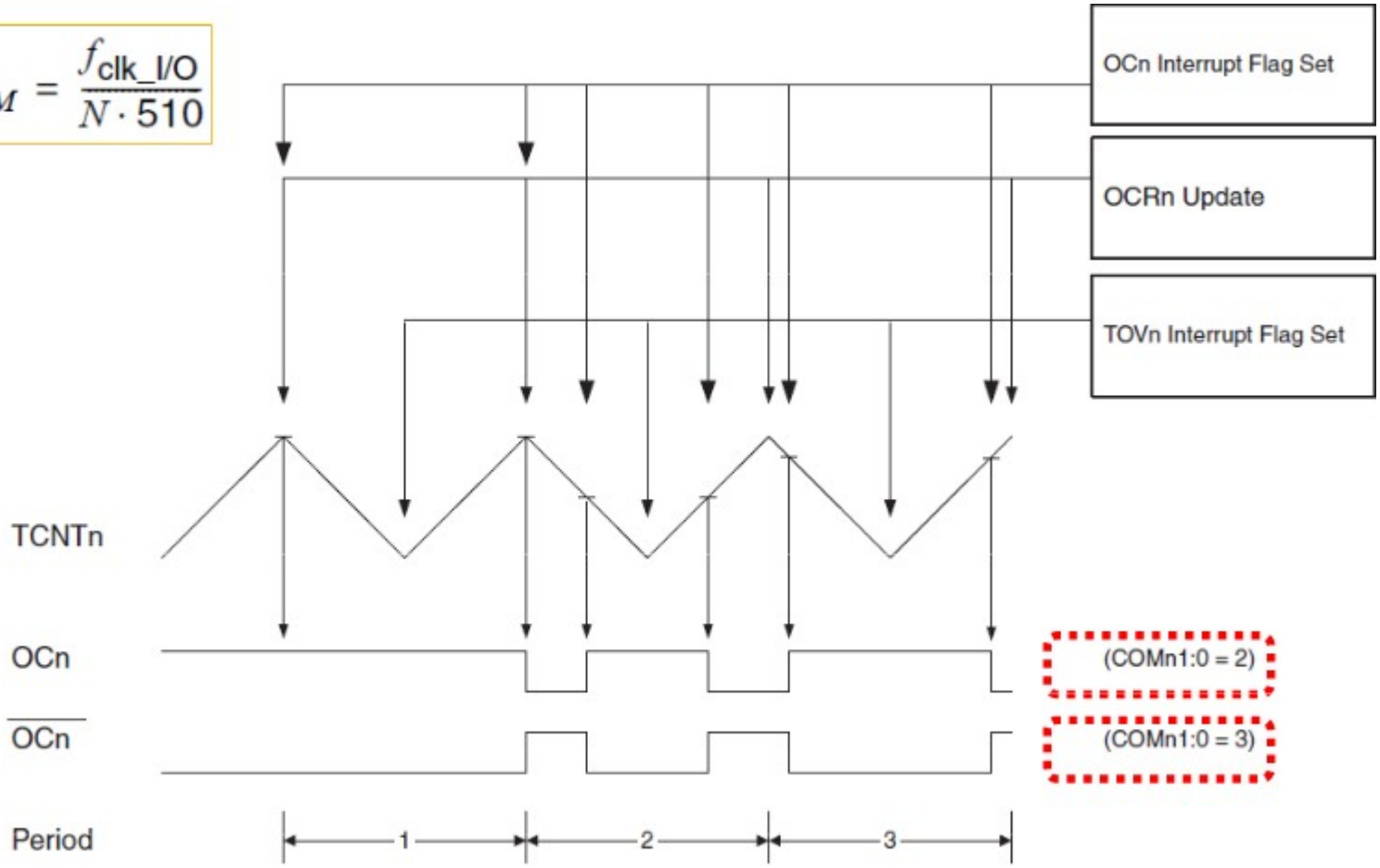


→ 주기가 일정하면서 duty를 제어할때 사용

2-3. Timer/Interrupt - Phase Correct-PWM Mode

- PC PWM Mode

$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$



→ 원신호의 주기는 일정하지만, 출력되는 주기는 제어됨 (Phase 제어)

3. PWM Duty Printf Code

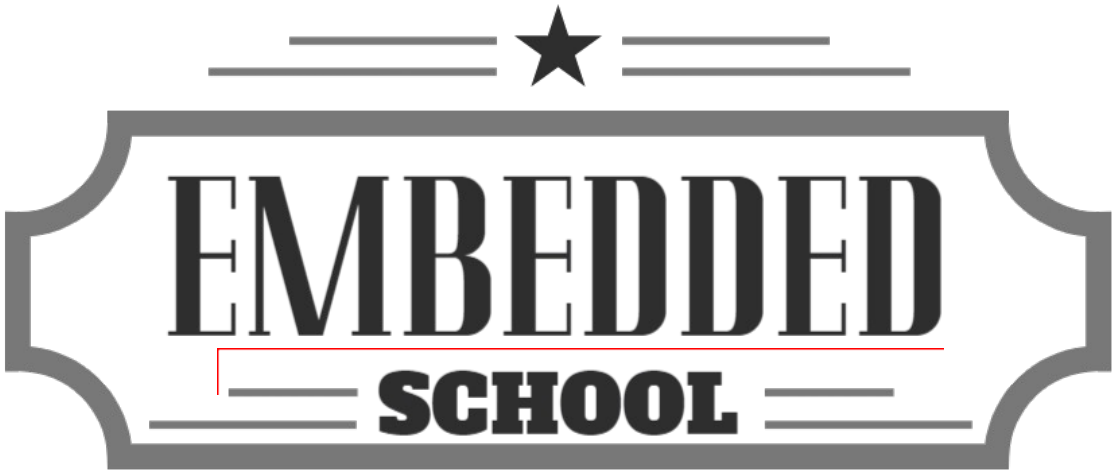
```c
#define F_CPU 16000000UL

#include <avr/io.h>
#include <avr/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <stdio.h>

#define sbi(PORTX, BitX) (PORTX |= (1 << BitX))
#define cbi(PORTX, BitX) (PORTX &= ~(1 << BitX))

unsigned char counter = 0;

void overflow_timer_init(void);

#define UART_BUFLEN 10
static int USART_TX_vect(char, FILE*);
static int usartTxChar(char, FILE*);
void UART_INIT(void)
{
    sbi(UCSR0A, U2X0);      //U2X0 = 1 -> Baudrate 9600 = 207

    UBRR0H = 0x00;
    UBRR0L = 207;

    UCSR0C |=0X06;

    sbi(UCSR0B , RXEN0);
    sbi(UCSR0B, TXEN0);
}
unsigned char UART_tranmit(unsigned char data)
{
    while(!(UCSR0A & (1<<UDRE0)));
    UDR0 = data;
}
void UART_string_transmit(char *string)     //*string 문자열의 시작주소
{
    while(*string != '\0')                  //문자열 맨 마지막 문자("\0") 확인
    {
        UART_tranmit( *string);
        string++;
    }
}
void UART_PRINT(char *name, long val )
{

    char debug_buffer[UART_BUFLEN] = {'\0'};    //배열 초기화
    UART_string_transmit(name);
    UART_string_transmit(" = ");
    itoa((val),debug_buffer, UART_BUFLEN);      //itoa함수: int데이터를 문자열로 변환시켜주는 함수
    UART_string_transmit(debug_buffer);
    UART_string_transmit("\n");
}
```

```c
SIGNAL(TIMER0_OVF_vect)
{
    if(counter == 125)
    {

        PORTB = 0xff;
        counter = 0;
    }
    else{
        PORTB = 0x00;
        counter++;

    }
int main(void)
{
    char str;
    char a[3] = {'\0',};//unsigned char이
    UART_INIT();        //UART초기화
    overflow_timer_init();

    while(1)                                ;
    {
        str = counter;
        //printf("ocr: %d\r\n",counter);
        itoa(str, a, 10);
        UART_string_transmit(a);
        UART_string_transmit("\n");
        _delay_ms(100);

    }
    return 0;
}

void overflow_timer_init(void)
{
    cbi(SREG, 7);
    TCCR0A = 0; //카운터 초기화
    TCCR0B = 0; //카운터 초기화
    PORTB = 0x00;
    DDRB = 0xff;

    TCCR0B = (1 << CS02) | (1 << CS00); //EXTERNAL FALLING
    TCNT0 = 131; //1/clkt = 64us, 64us *125 = 8ms
    sbi(TIMSK0,TOIE0);
    sbi(SREG,7);
}
```

3. PWM Duty Printf Code 결과

Receive

```
51
57
63
69
76
82
88
94
100
106
113
119
125
5
12
18
```



감사합니다.