AVR – HW6

임베디드스쿨1기

Lv1과정

2020. 10. 23

강경수

1. TIMER/COUNTER REVIEW

▣ TIMER/COUNTER CODE REVIEW

2020.10.08 강경수

```
void overflow_timer_init(void)
{
    cbi(SREG,7);
    TCCR0A=0; //카운터 초기화
    TCCR0B=0;

    PORTB=0x00;
    DDRB=0xff;

    TCCR0B = (1<<CS02) |(1<<CS00);
    TCNT0 = 131;

    sbi(TIMSK0,TOIE0);
    sbi(SREG,7);
}
```

OC0A, OC0B핀에 파형을 출력하지 않겠다는 설정.
(즉 일반 GPIO로 사용한다.)

만약 사용 할 거면 DDR레지스터 및 COM0A1, COMA0레지스터 설정해야 함.

PORTB PULL-UP

분주비 : 1024 → 16,000,000/1024 = 15,625Hz
TCNT0 → 타이머 발생시마다 쌓여 나가는 값.
초기값 설정

TIMER/COUNTER 인터럽트 활성화.
전역 INTERRUPT 활성화.

1. TIMER/COUNTER REVIEW

TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 14-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on compare match
1	0	Clear OC0A on compare match
1	1	Set OC0A on compare match

TCCR0B – Timer/Counter Control Register B

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	

1. TIMER/COUNTER REVIEW

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.

TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

1. TIMER/COUNTER REVIEW

```
void EXT_falling_timer_Init(void)
{
    cbi(SREG,7);
    TCCR0A=0;
    TCCR0B=0;

    PORTB=0x00;
    PORTD=0x10;

    DDRB=0xff;
    DDRD=0x00;

    TCCR0B = (1<<CS02) |(1<<CS01);
    sbi(SREG,7);
}
```

T0pin의 변화를 감지 할 수 있다.

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. TIMER COUNTER와 PWM 관계

▣ TIMER/COUNTER 와 PWM의 관계

0. TIMER/COUNTER와 PWM은 별도로 생각하는게 좋을듯. 단지 PWM신호를 만들어내기 위하여 TIMER/COUNTER DATA Register를 활용할 뿐이다.

1. TIMER/COUNTER를 이용하여 PWM신호를 만들어내는 방법

(1) ISR루틴에서 GPIO를 제어하여 만들어 내는방법

```
ISR(TIMER0_COMPA_vect)
{
    PORTB |= 0xFF;
}
```

ISR루틴에서 GPIO를 제어하여 만들어 낼 수 있다.

이러한 방법은 CPU의 리소스를 잡아먹는다.

(2) MCU STATUS REGISTER에서 직접 설정하여 PWM신호 만들어내는 방법

Table 14-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on compare match
1	0	Clear OC0A on compare match
1	1	Set OC0A on compare match

ATmega328p 의 OC0A, OC0B 핀은 별도의 GPIO제어 없이 (PORT제어 필요 없음, DDR 입출력만 SETTING) H/W에서 알아서 COMPARE MATCH가 발생함에 따라서 PWM 신호를 출력해 준다.

3. 모드별 DUTY 및 주기 계산

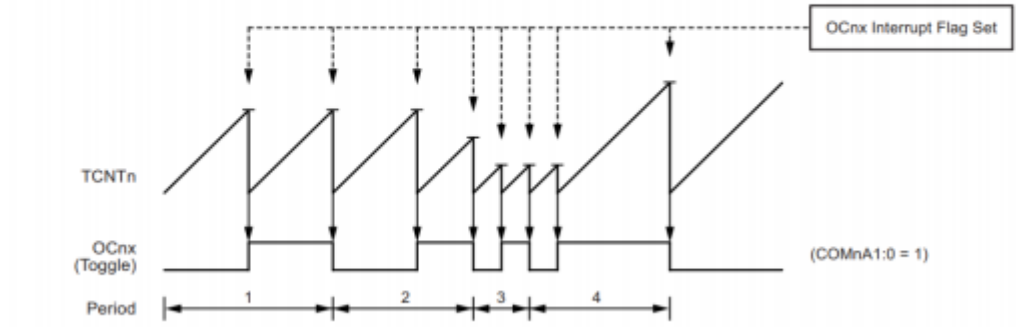
▣ 8Bit TIMER/COUNTER PWM DUTY

1. Normal Mode
- TCNTn 값이 overflow 발생시 interrupt발생. 이 Normal Mode를 이용하여
- 항상 up(증가)하는 방향으로 TCNTn 레지스터 변화
- Normal Mode에서는 PWM OCnx H/W 출력도 지원 안한다.
- 별로 쓸일 없는 TIMER/COUNTER DATA SHEET에서도 추천안함.

The output compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much of the CPU time.

2. CTC모드

Figure 17-5. CTC Mode, Timing Diagram

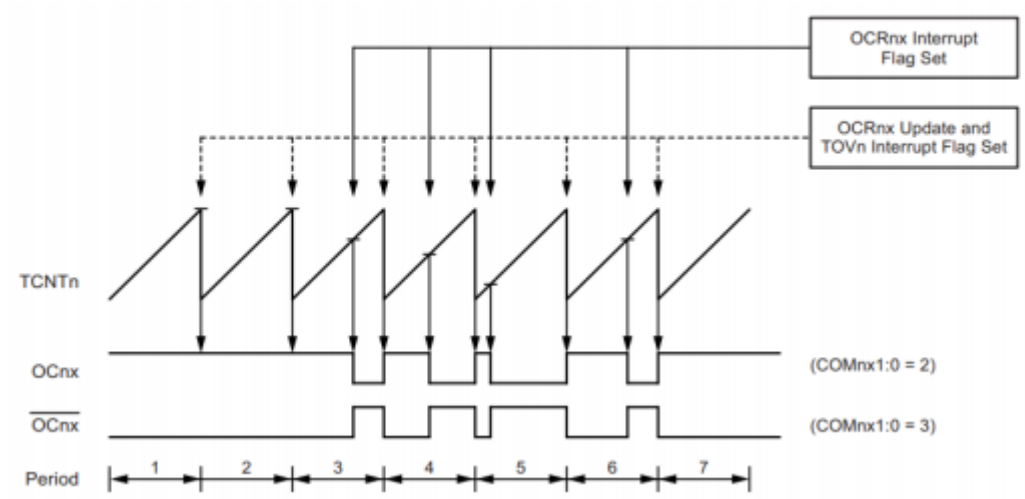


주기 : (1/CLK)XOCRnX2

DUTY : 항상50%

3. 모드별 DUTY 및 주기 계산

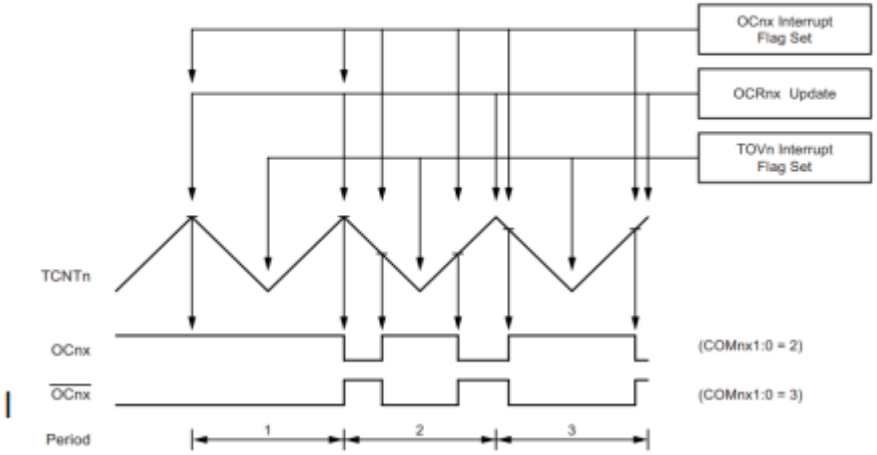
3. FAST PWM MODE



주기 : (1/CLK) * 256 (변경불가)

DUTY : OCRn / TCNTn * 100(%)

4. Phase Correct PWM MODE



주기 및 DUTY 계산하는법 모르겠음..

4. Context Switching in AVR

▣ AVR CORE에서 Context Switching

- Scheduling 개념이 포함되지 않기때문에 엄밀한 의미에서 Context Switching은 일어나지 않는다.

```
SIGNAL(TIMER0_OVF_vect)
{
  90:   1f 92           push    r1
  92:   0f 92           push    r0
  94:   0f b6           in  r0, 0x3f    ; 63
  96:   0f 92           push    r0
  98:   11 24           eor r1, r1
  9a:   8f 93           push    r24
    if(counter==125)
  9c:   80 91 00 01     lds r24, 0x0100 ; 0x800100 <_edata>
  a0:   8d 37           cpi r24, 0x7D   ; 125
  a2:   29 f4           brne    .+10        ; 0xae <__vector_16+0x1e>
    {
        PORTB= 0xff;
  a4:   8f ef           ldi r24, 0xFF   ; 255
  a6:   85 b9           out 0x05, r24   ; 5
        counter = 0;
  a8:   10 92 00 01     sts 0x0100, r1  ; 0x800100 <_edata>
  ac:   06 c0           rjmp    .+12        ; 0xba <__vector_16+0x2a>
    }
    else{
```

lss파일(assmbly) 을 까보게 되면 r1,r0와 같은 레지스터를 push하는것위 확인 가능하다. 즉 연산과 관련된 레지스터를 저장하는 것을 확인 할 수 있다.

제어 레지스터를 push하는게 아니라 연산과 관련된것만 한다. (여태가지 PORTB가 인터럽트 수행 이후 원래 상태로 돌아오지 않아 헛갈렸음)

4. Context Switching in AVR

```
        PORTB = 0x00;
  ae:   15 b8           out 0x05, r1    ; 5
        counter++;
  b0:   80 91 00 01     lds r24, 0x0100 ; 0x800100 <_edata>
  b4:   8f 5f           subi    r24, 0xFF   ; 255
  b6:   80 93 00 01     sts 0x0100, r24 ; 0x800100 <_edata>
    }
}
  ba:   8f 91           pop r24
  bc:   0f 90           pop r0
  be:   0f be           out 0x3f, r0    ; 63
  c0:   0f 90           pop r0
  c2:   1f 90           pop r1
  c4:   18 95           reti
```

5. Task in AVR

```
typedef enum status{t_ready, t_busy, t_wait, t_quit} status;

typedef struct TaskControlBlock     //defines and controls a task for this s
{
    uint8_t task_id;
    enum status state;
    uint8_t priority;
    unsigned int stack_pointer_begin;
    unsigned int stack_pointer_end;
    void (*fnctpt)(void);
} tcb;
tcb task[maxTask];
```

AVR CORE로 구현한 스케쥴링 비스무리한 예시..

task control block 구조체를 만들고
함수포인터로 각각 실행할 함수를 할당하여

timer interrupt ISR 수행할때마다
각각의 함수(Task)를 조금씩 실행한다.

```
ISR(TIMER0_COMPA_vect)
{
    uint8_t q=0;
    cli();
    //getting the termination pointer of individual stacks
    task[currentTaskId].stack_pointer_end=StackPointer;
    for(q=0; q<maxTask; q++)
    {
        if(task[q].state==t_ready)
        {
            //setting up task parameters
            task[q].state=t_busy;
            currentTaskId=task[q].task_id;      //sets task_id for saving cc
            StackPointer=task[q].stack_pointer_begin;
            sei();
            task[q].fnctpt();       //execute task if not priorly executed
        }
        R=q;
    }
```