EMBEDDED SCHOOL

AVR – HW7

임베디드스쿨1기

Lv1과정

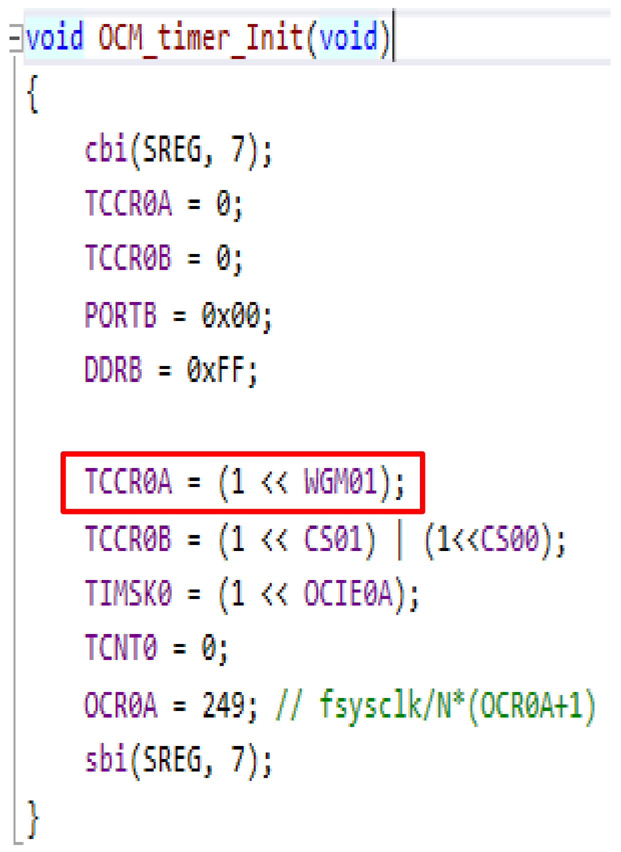
2020. 10. 26

손표훈

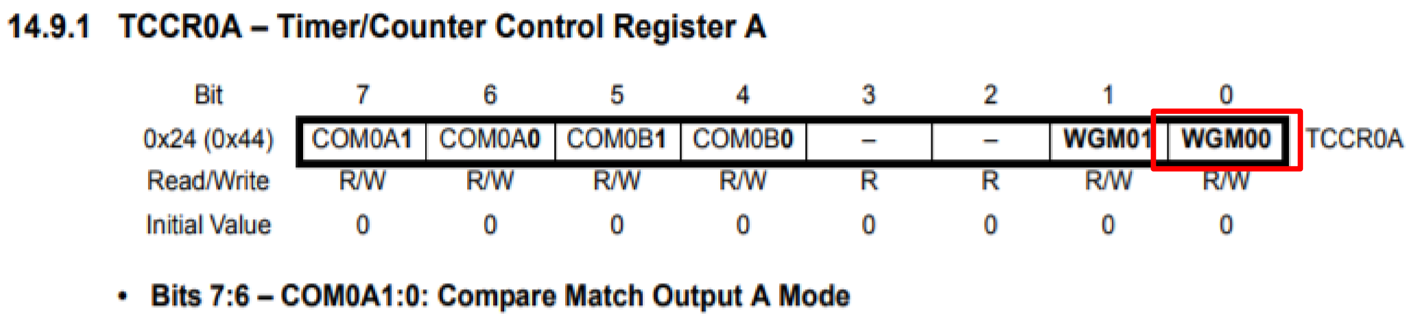
1. 복습

(1) CTC MODE

- Register Setting



```
void OCM_timer_Init(void)
{
    cbi(SREG, 7);
    TCCR0A = 0;
    TCCR0B = 0;
    PORTB = 0x00;
    DDRB = 0xFF;

    TCCR0A = (1 << WGM01);
    TCCR0B = (1 << CS01) | (1<<CS00);
    TIMSK0 = (1 << OCIE0A);
    TCNT0 = 0;
    OCR0A = 249; // fsysclk/N*(OCR0A+1)
    sbi(SREG, 7);
}
```



14.9.1 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	

- **Bits 7:6 – COM0A1:0: Compare Match Output A Mode**

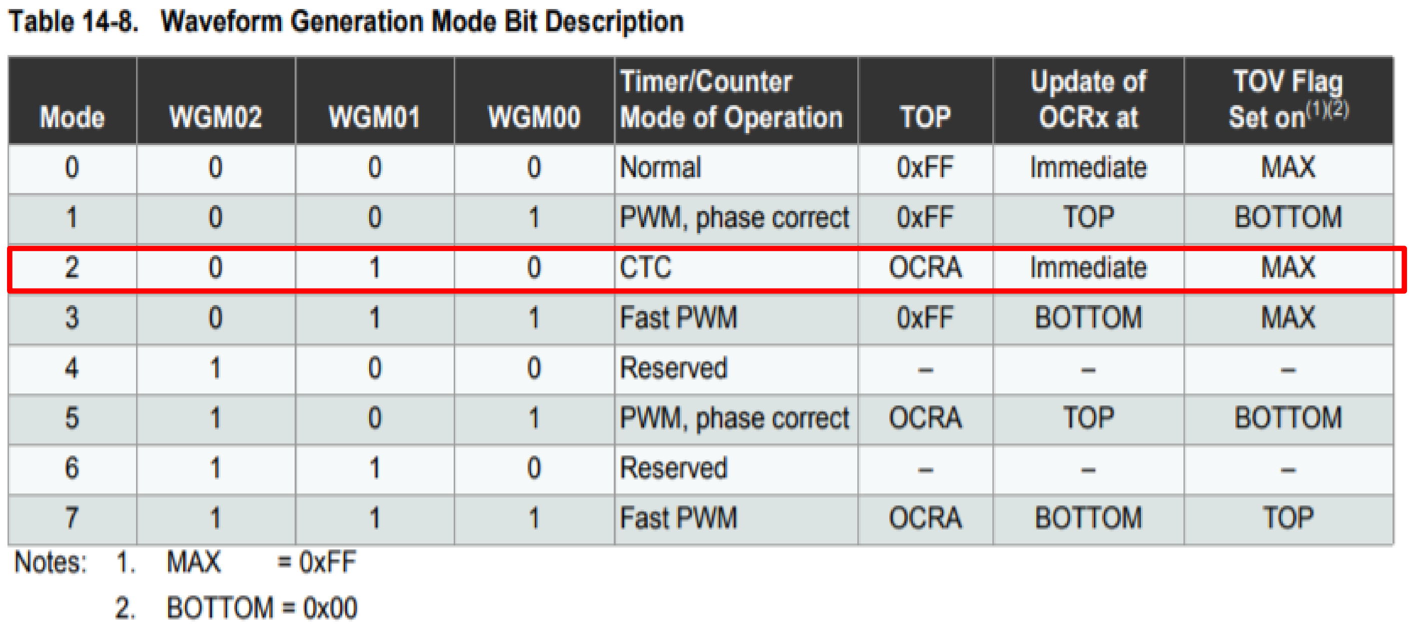


Table 14-8. Waveform Generation Mode Bit Description

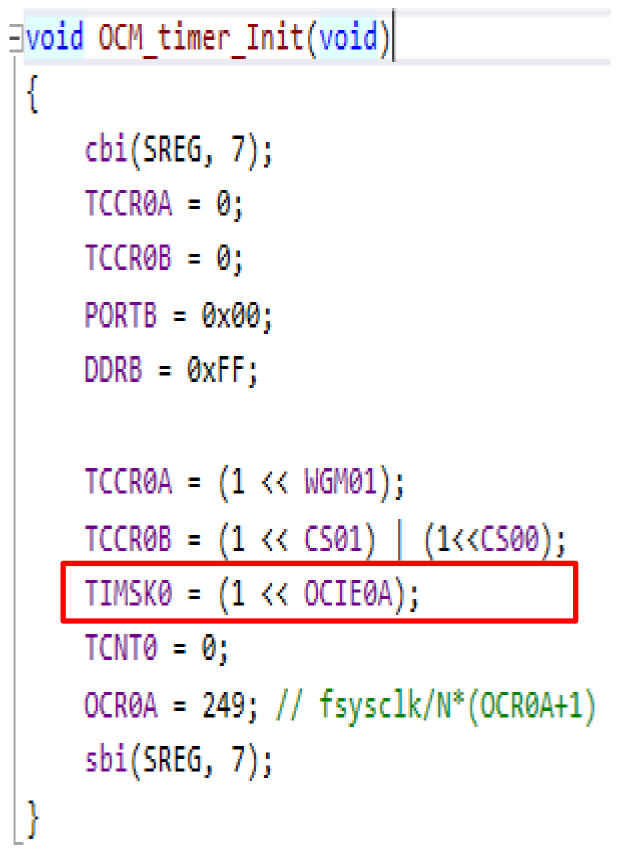
Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[(1)(2)]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

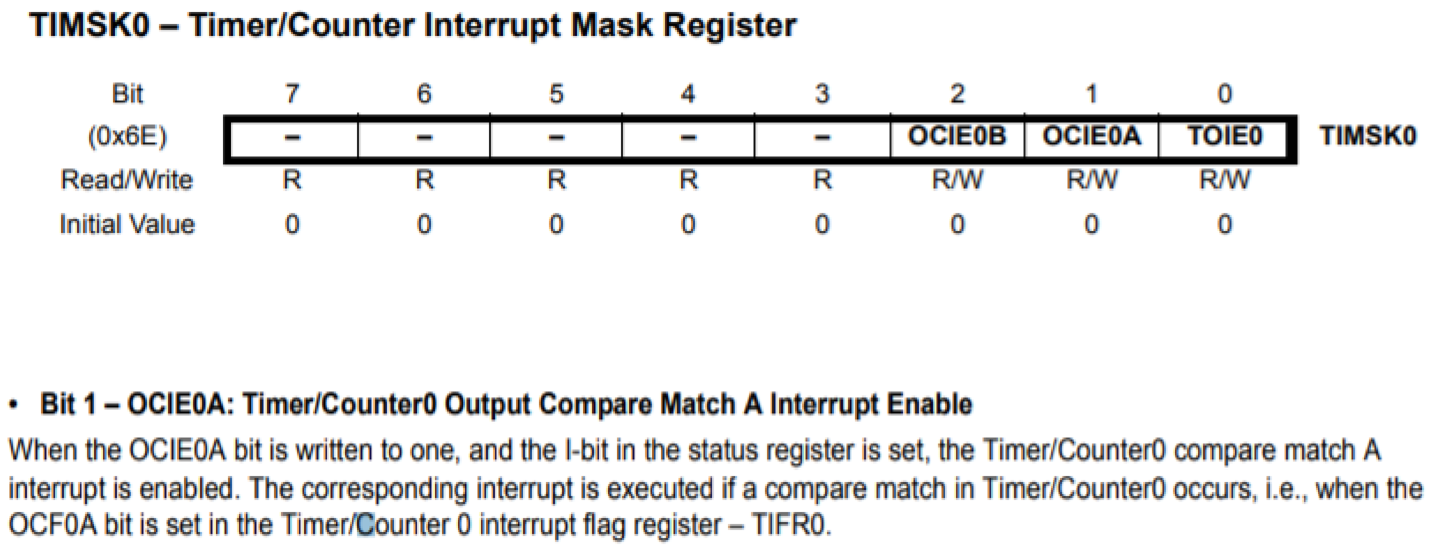
1. 복습

(1) CTC MODE

- Register Setting



```
void OCM_timer_Init(void)
{
    cbi(SREG, 7);
    TCCR0A = 0;
    TCCR0B = 0;
    PORTB = 0x00;
    DDRB = 0xFF;

    TCCR0A = (1 << WGM01);
    TCCR0B = (1 << CS01) | (1<<CS00);
    TIMSK0 = (1 << OCIE0A);
    TCNT0 = 0;
    OCR0A = 249; // fsysclk/N*(OCR0A+1)
    sbi(SREG, 7);
}
```



TIMSK0 – Timer/Counter Interrupt Mask Register

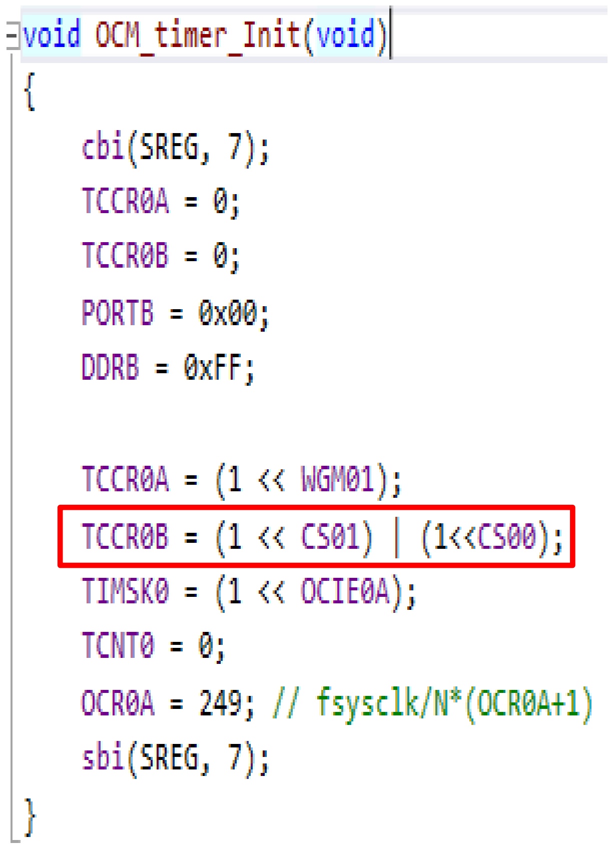
Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 1 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 compare match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 interrupt flag register – TIFR0.

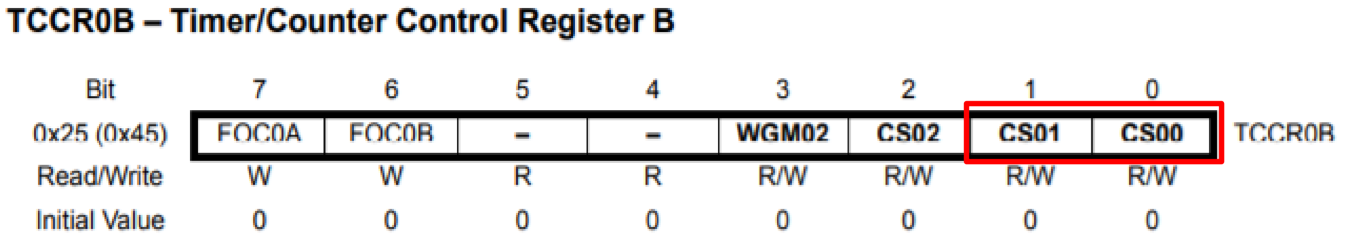
1. 복습

(1) CTC MODE
- Register Setting
- System CLK = 16MHz



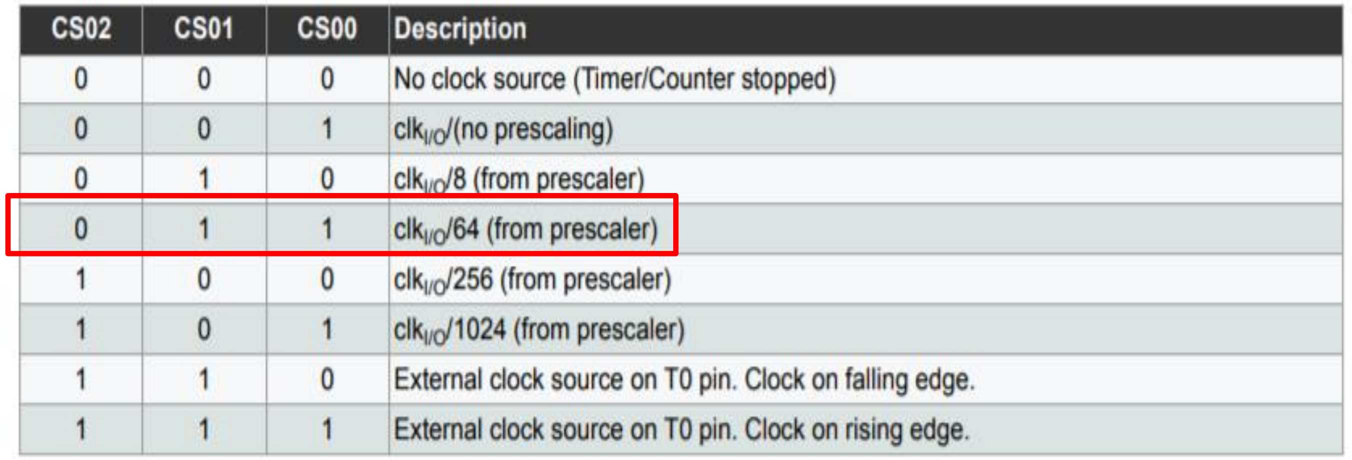
```
void OCM_timer_Init(void)
{
    cbi(SREG, 7);
    TCCR0A = 0;
    TCCR0B = 0;
    PORTB = 0x00;
    DDRB = 0xFF;

    TCCR0A = (1 << WGM01);
    TCCR0B = (1 << CS01) | (1<<CS00);
    TIMSK0 = (1 << OCIE0A);
    TCNT0 = 0;
    OCR0A = 249; // fsysclk/N*(OCR0A+1)
    sbi(SREG, 7);
}
```

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	

Table 14-9. Clock Select Bit Description



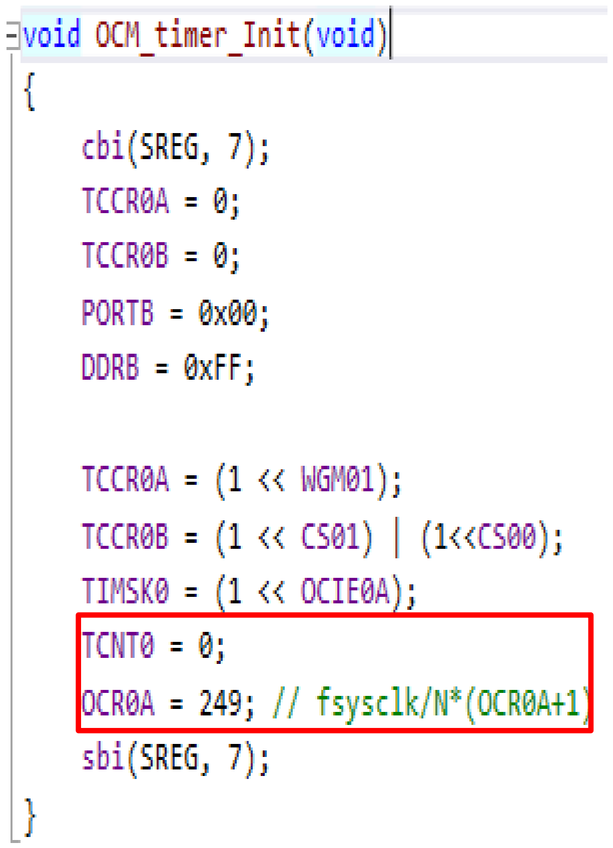
CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock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

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

- 프리스케일 값을 64분주, 타이머/카운터 모듈에 16MHz/64의 클럭을 공급

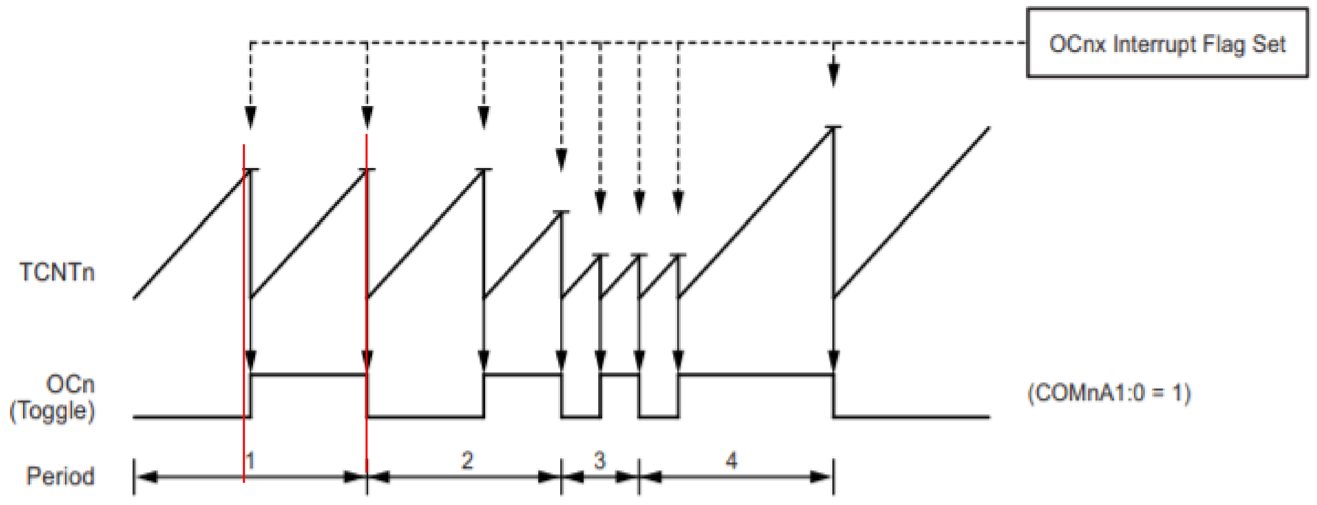
1. 복습

(1) CTC MODE
- Register Setting(OC핀 출력 주파수설정)
- System CLK = 16MHz



```
void OCM_timer_Init(void)
{
    cbi(SREG, 7);
    TCCR0A = 0;
    TCCR0B = 0;
    PORTB = 0x00;
    DDRB = 0xFF;

    TCCR0A = (1 << WGM01);
    TCCR0B = (1 << CS01) | (1<<CS00);
    TIMSK0 = (1 << OCIE0A);
    TCNT0 = 0;
    OCR0A = 249; // fsysclk/N*(OCR0A+1)
    sbi(SREG, 7);
}
```

Figure 14-5. CTC Mode, Timing Diagram

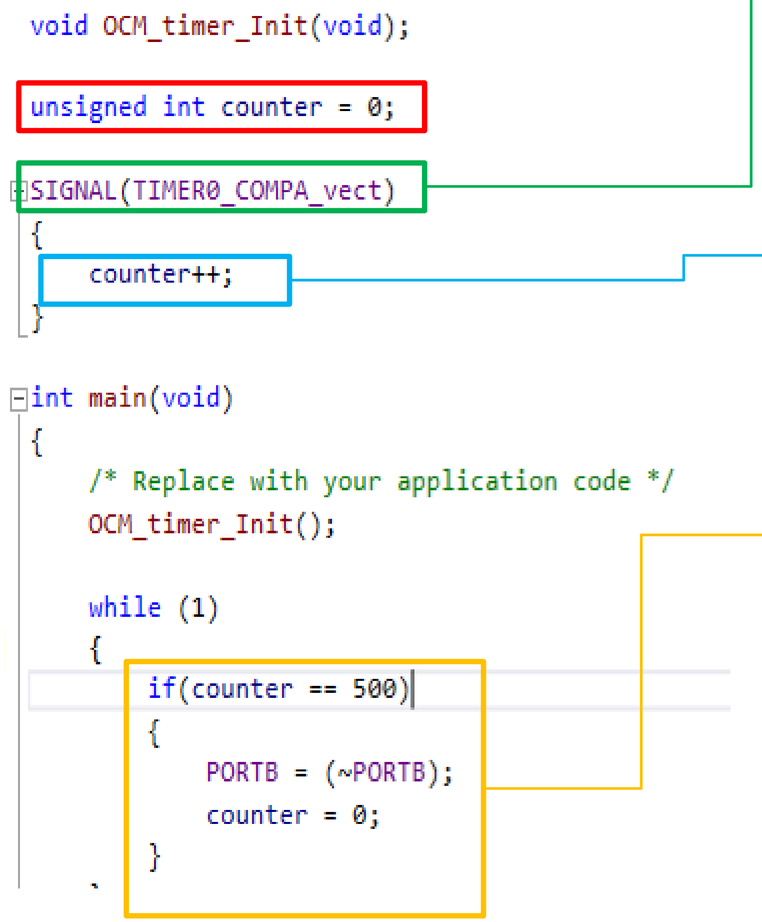


$$f_{OCnx} = \frac{f_{clk\_I/O}}{2 \times N \times (1 + OCRnx)}$$

- 비교 인터럽트 주파수 : FclkI/O / N*(1+OCRnX)
- 출력 파형 주파수 : 비교인터럽트 주파수/2(왜? 위 파형에서 보이는것 처럼 인터럽트가 2번 발생해 Ocn핀의 한주기 파형이 생성된다)
- 설정된 주파수 : 16MHz/2*64*(1+249) = 500Hz(2ms)

1. 복습

(1) CTC MODE(타이머 기능만 이용 PWM 사용x)
- CTC모드 테스트



```
void OCM_timer_Init(void);

unsigned int counter = 0;

SIGNAL(TIMER0_COMPA_vect)
{
    counter++;
}

int main(void)
{
    /* Replace with your application code */
    OCM_timer_Init();

    while (1)
    {
        if(counter == 500)
        {
            PORTB = (~PORTB);
            counter = 0;
        }
```

1. Timer0_COMPA 인터럽트는 매 1ms마다 발생

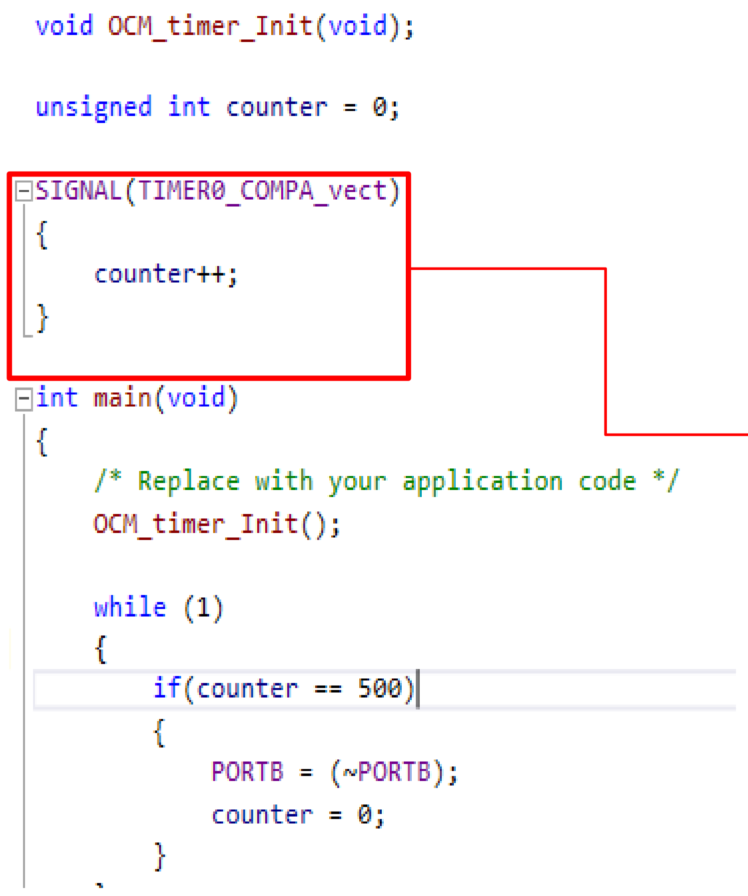
2. Counter변수를 매 1ms마다 1씩 증가

3. Counter변수가 500이 되면 출력핀을 Toggle(현 상태 반전)
   Counter 변수를 초기화

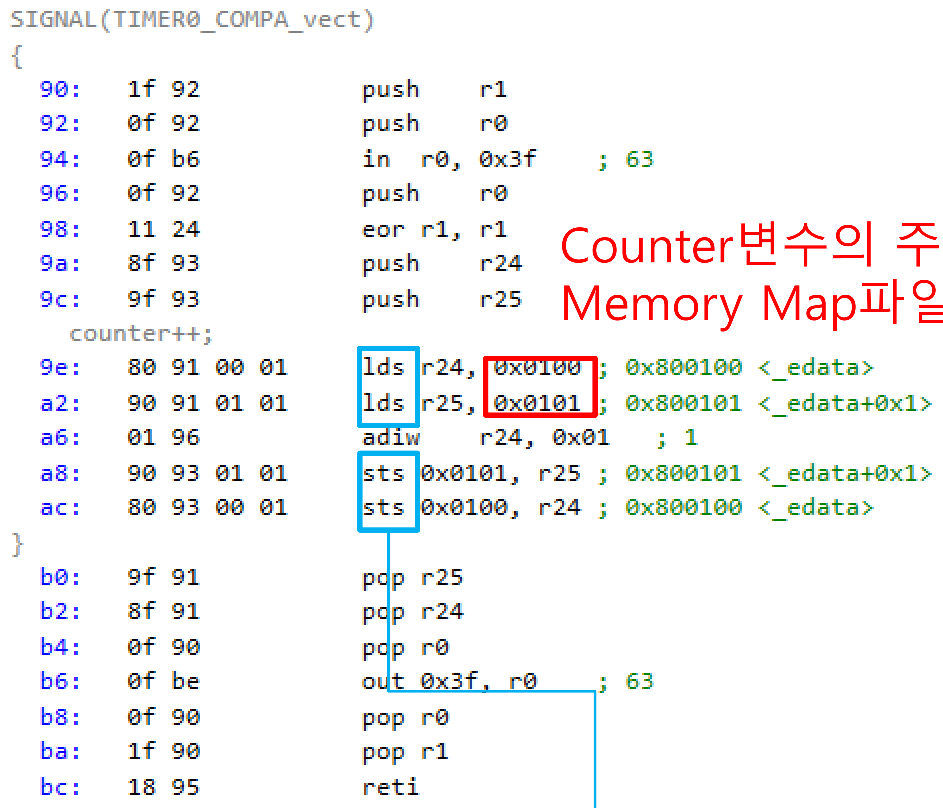
- 테스트 의도 : 1ms 인터럽트를 이용하여 500ms의 파형을 생성!
- 위 코드는 의도한 동작을 수행하지 않음..
- Counter변수 선언 중요!

1. 복습

(2) Volatile Keyword-질문
Volatile선언 안했을 때 Assem



```c
void OCM_timer_Init(void);

unsigned int counter = 0;

SIGNAL(TIMER0_COMPA_vect)
{
    counter++;
}
int main(void)
{
    /* Replace with your application code */
    OCM_timer_Init();

    while (1)
    {
        if(counter == 500)
        {
            PORTB = (~PORTB);
            counter = 0;
        }
```



```
SIGNAL(TIMER0_COMPA_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
  9c:   9f 93          push    r25
    counter++;
  9e:   80 91 00 01    lds r24, 0x0100 ; 0x800100 <_edata>
  a2:   90 91 01 01    lds r25, 0x0101 ; 0x800101 <_edata+0x1>
  a6:   01 96          adiw    r24, 0x01   ; 1
  a8:   90 93 01 01    sts 0x0101, r25 ; 0x800101 <_edata+0x1>
  ac:   80 93 00 01    sts 0x0100, r24 ; 0x800100 <_edata>
}
  b0:   9f 91          pop r25
  b2:   8f 91          pop r24
  b4:   0f 90          pop r0
  b6:   0f be          out 0x3f, r0    ; 63
  b8:   0f 90          pop r0
  ba:   1f 90          pop r1
  bc:   18 95          reti
```

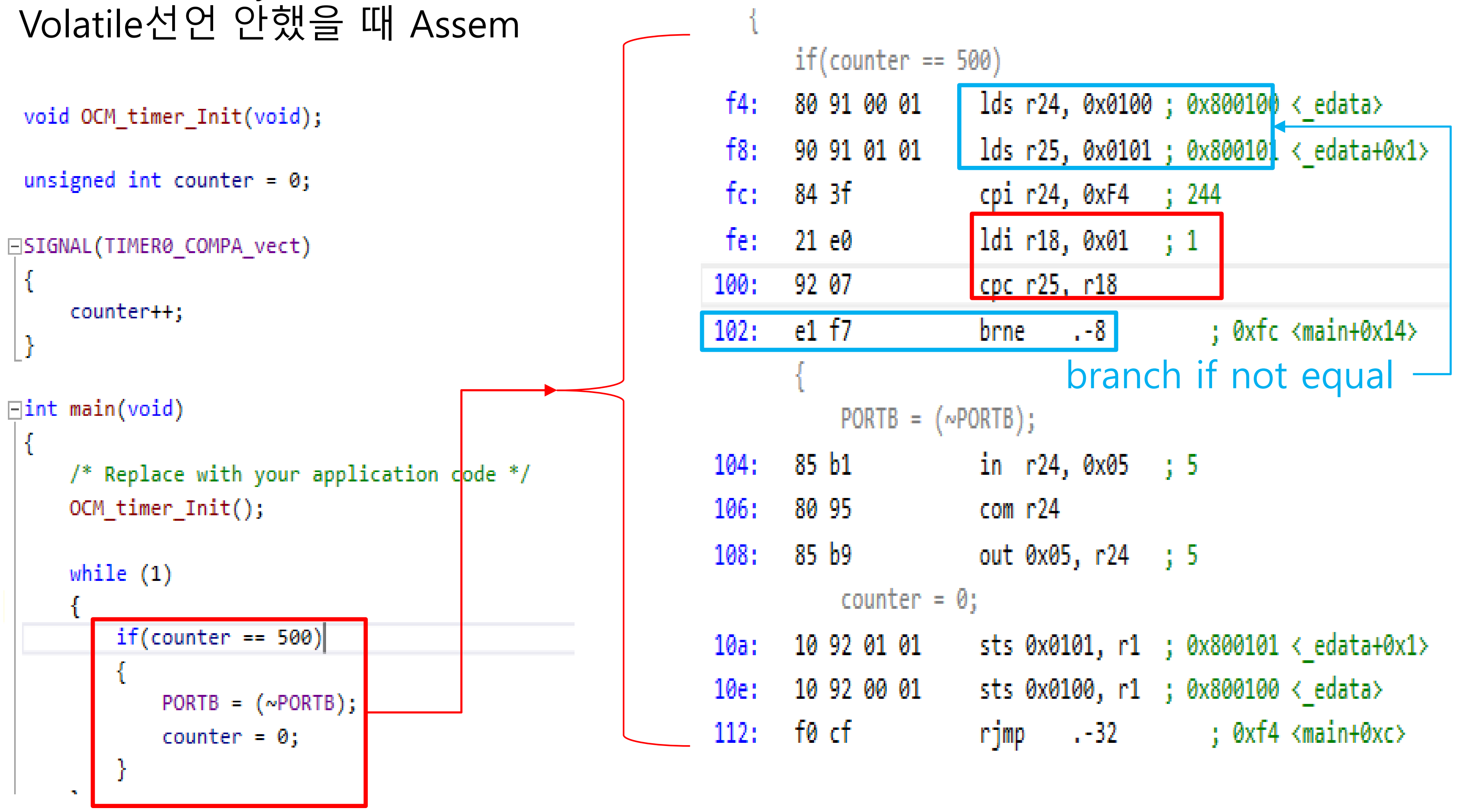
Counter변수의 주소값
Memory Map파일에서 확인

메모리에서 읽고/쓰기 명령어

포기하면 얻는 건 아무것도 없다.

1. 복습

(2) Volatile Keyword-질문
Volatile선언 안했을 때 Assem



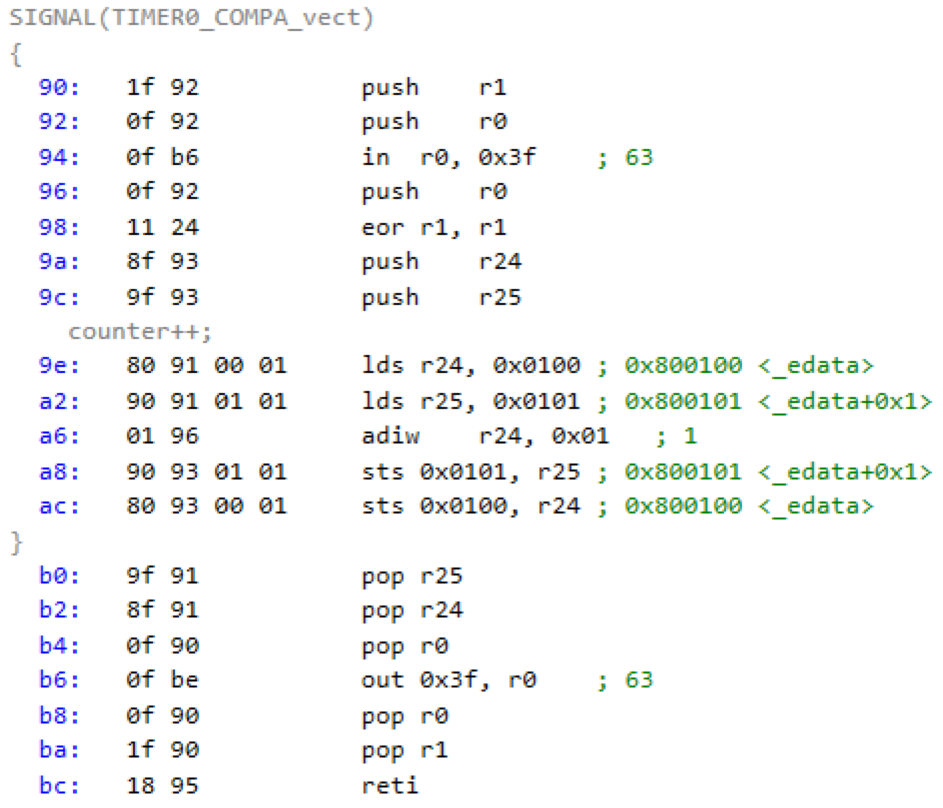
```
void OCM_timer_Init(void);

unsigned int counter = 0;

SIGNAL(TIMER0_COMPA_vect)
{
    counter++;
}

int main(void)
{
    /* Replace with your application code */
    OCM_timer_Init();

    while (1)
    {
        if(counter == 500)
        {
            PORTB = (~PORTB);
            counter = 0;
        }
```

```
    {
        if(counter == 500)
  f4:   80 91 00 01     lds r24, 0x0100 ; 0x800100 <_edata>
  f8:   90 91 01 01     lds r25, 0x0101 ; 0x800101 <_edata+0x1>
  fc:   84 3f           cpi r24, 0xF4   ; 244
  fe:   21 e0           ldi r18, 0x01   ; 1
 100:   92 07           cpc r25, r18
 102:   e1 f7           brne    .-8         ; 0xfc <main+0x14>
        {
            PORTB = (~PORTB);
 104:   85 b1           in  r24, 0x05   ; 5
 106:   80 95           com r24
 108:   85 b9           out 0x05, r24   ; 5
            counter = 0;
 10a:   10 92 01 01     sts 0x0101, r1  ; 0x800101 <_edata+0x1>
 10e:   10 92 00 01     sts 0x0100, r1  ; 0x800100 <_edata>
 112:   f0 cf           rjmp    .-32        ; 0xf4 <main+0xc>
```

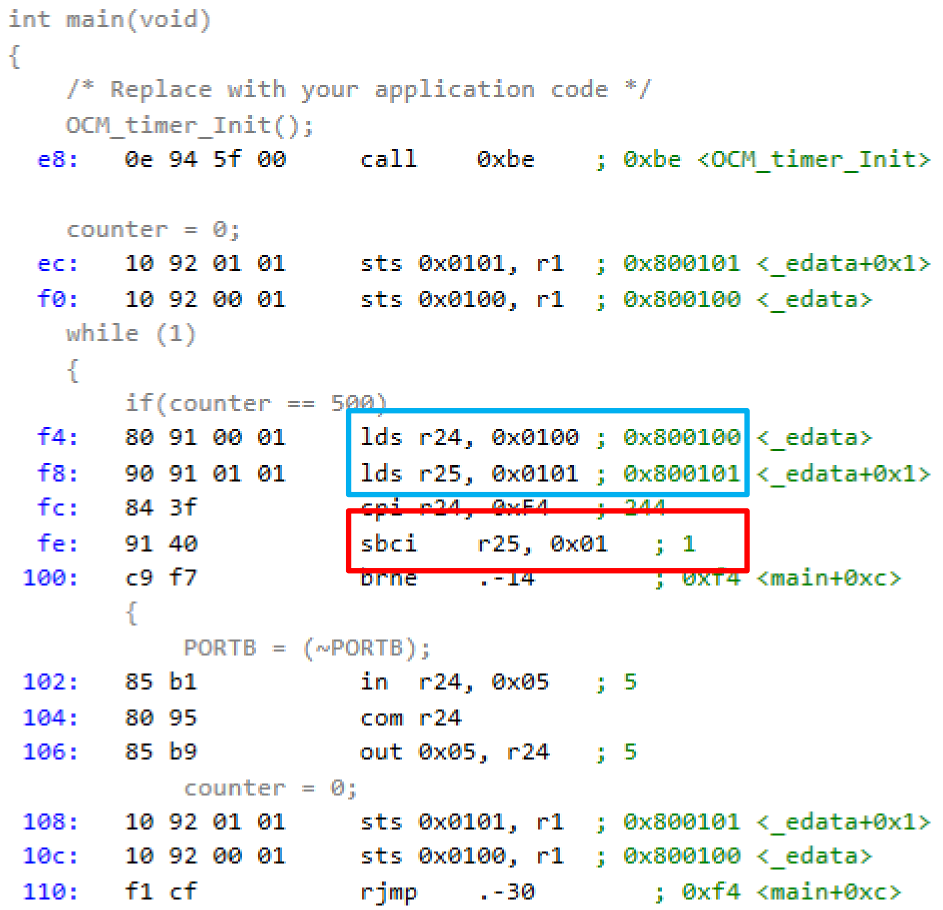
- 테스트 의도 : 1ms 인터럽트를 이용하여 500ms의 파형을 생성!
- 위 코드는 의도한 동작을 수행하지 않음..
- Counter변수 선언 중요!

1. 복습

(2) Volatile Keyword
Volatile선언 했을 때 Assem



```
SIGNAL(TIMER0_COMPA_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
  9c:   9f 93           push    r25
    counter++;
  9e:   80 91 00 01     lds r24, 0x0100 ; 0x800100 <_edata>
  a2:   90 91 01 01     lds r25, 0x0101 ; 0x800101 <_edata+0x1>
  a6:   01 96           adiw    r24, 0x01   ; 1
  a8:   90 93 01 01     sts 0x0101, r25 ; 0x800101 <_edata+0x1>
  ac:   80 93 00 01     sts 0x0100, r24 ; 0x800100 <_edata>
}
  b0:   9f 91           pop r25
  b2:   8f 91           pop r24
  b4:   0f 90           pop r0
  b6:   0f be           out 0x3f, r0    ; 63
  b8:   0f 90           pop r0
  ba:   1f 90           pop r1
  bc:   18 95           reti
```



```
int main(void)
{
    /* Replace with your application code */
    OCM_timer_Init();
  e8:   0e 94 5f 00     call    0xbe    ; 0xbe <OCM_timer_Init>

    counter = 0;
  ec:   10 92 01 01     sts 0x0101, r1  ; 0x800101 <_edata+0x1>
  f0:   10 92 00 01     sts 0x0100, r1  ; 0x800100 <_edata>
    while (1)
    {
        if(counter == 500)
  f4:   80 91 00 01     lds r24, 0x0100 ; 0x800100 <_edata>
  f8:   90 91 01 01     lds r25, 0x0101 ; 0x800101 <_edata+0x1>
  fc:   84 3f           cpi r24, 0xF4   ; 244
  fe:   91 40           sbci    r25, 0x01   ; 1
 100:   c9 f7           brne    .-14        ; 0xf4 <main+0xc>
        {
            PORTB = (~PORTB);
 102:   85 b1           in  r24, 0x05   ; 5
 104:   80 95           com r24
 106:   85 b9           out 0x05, r24   ; 5
            counter = 0;
 108:   10 92 01 01     sts 0x0101, r1  ; 0x800101 <_edata+0x1>
 10c:   10 92 00 01     sts 0x0100, r1  ; 0x800100 <_edata>
 110:   f1 cf           rjmp    .-30        ; 0xf4 <main+0xc>
```

Volatile 안쓸때와 다른점은 비교문에서 명령어가 sbci 한줄로 줄었음…
sbci는 결과 값을 r25에 저장하는 명령어임..

1. 복습

(2) Volatile Keyword-질문

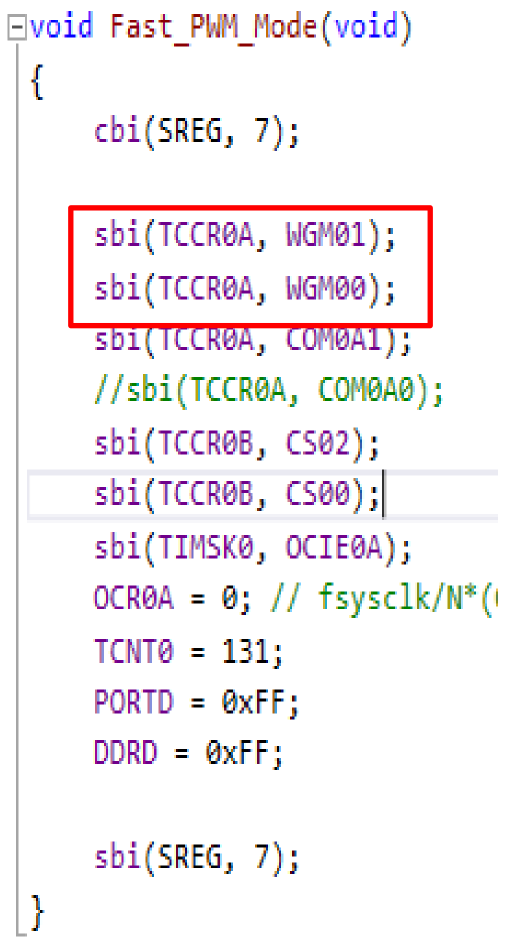
- ISR에서 Counter가 저장된 메모리에서 값을 읽고/쓰기가 발생..
volatile Keyword가 CPU의 캐시(자주 사용된 변수의 경우 레지스터에 넣고 동작속도를 올리는 기능?)
금지하여 원칙적으로 메모리에서 읽고/쓰기가 되게끔 하는 기능으로 예상했지만…

- ISR과 main문에서도 Volatile 쓸때와 안쓸때 모두 메모리에 접근하는 명령어는 있음
if문에서 비교가 끝나고 다시 while문으로 돌아올 때 count변수가 저장된 data space에서
count를 load하는 명령어가 있음..

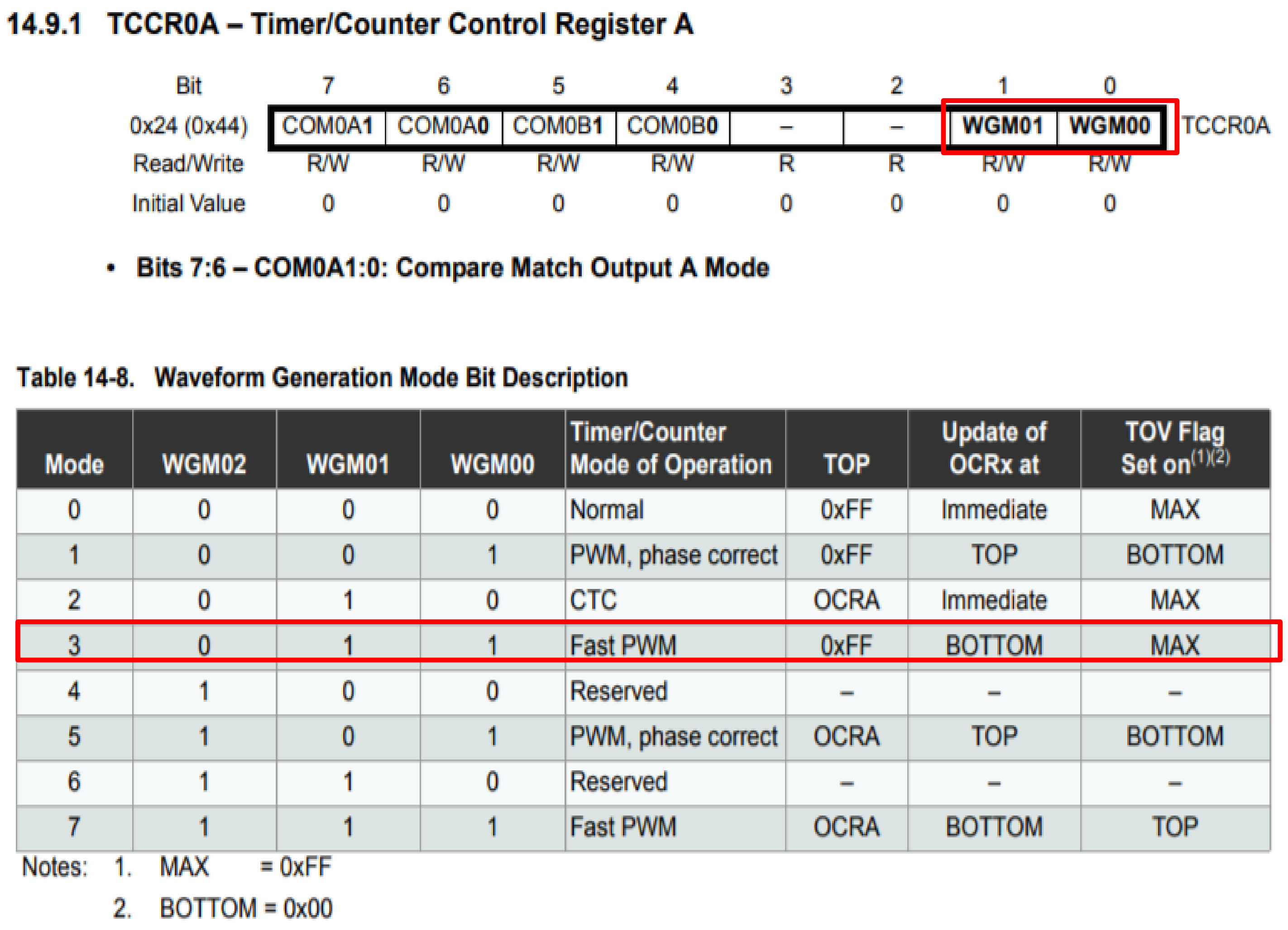
- 결론: 정확히 어떻게 volatil을 쓸 때와 안 쓸 때 동작이 차이가 나는지 명확하지 않습니다..

1. 복습

(2) FAST PWM MODE(높은 주파수, 낮은 분해능)
- Register Setting



```
void Fast_PWM_Mode(void)
{
    cbi(SREG, 7);

    sbi(TCCR0A, WGM01);
    sbi(TCCR0A, WGM00);
    sbi(TCCR0A, COM0A1);
    //sbi(TCCR0A, COM0A0);
    sbi(TCCR0B, CS02);
    sbi(TCCR0B, CS00);
    sbi(TIMSK0, OCIE0A);
    OCR0A = 0; // fsysclk/N*(
    TCNT0 = 131;
    PORTD = 0xFF;
    DDRD = 0xFF;

    sbi(SREG, 7);
}
```



14.9.1 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	

- **Bits 7:6 – COM0A1:0: Compare Match Output A Mode**

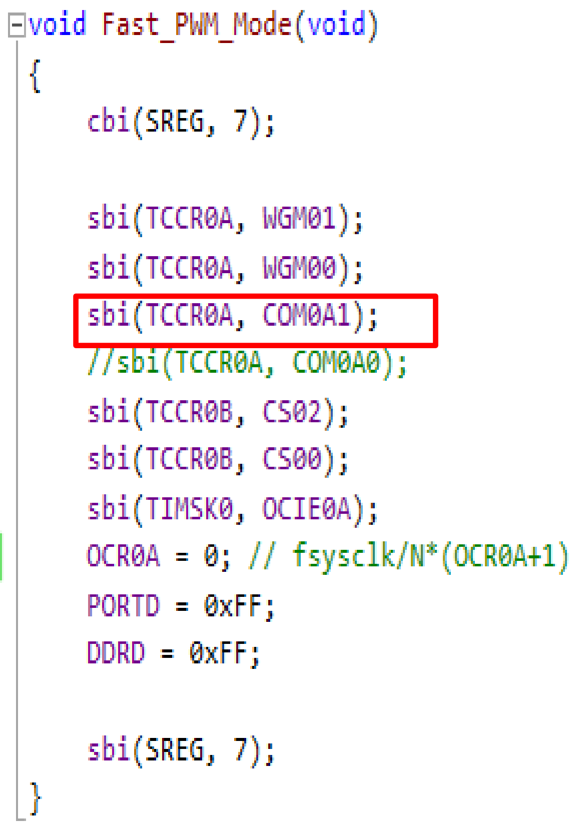
Table 14-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[(1)(2)]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

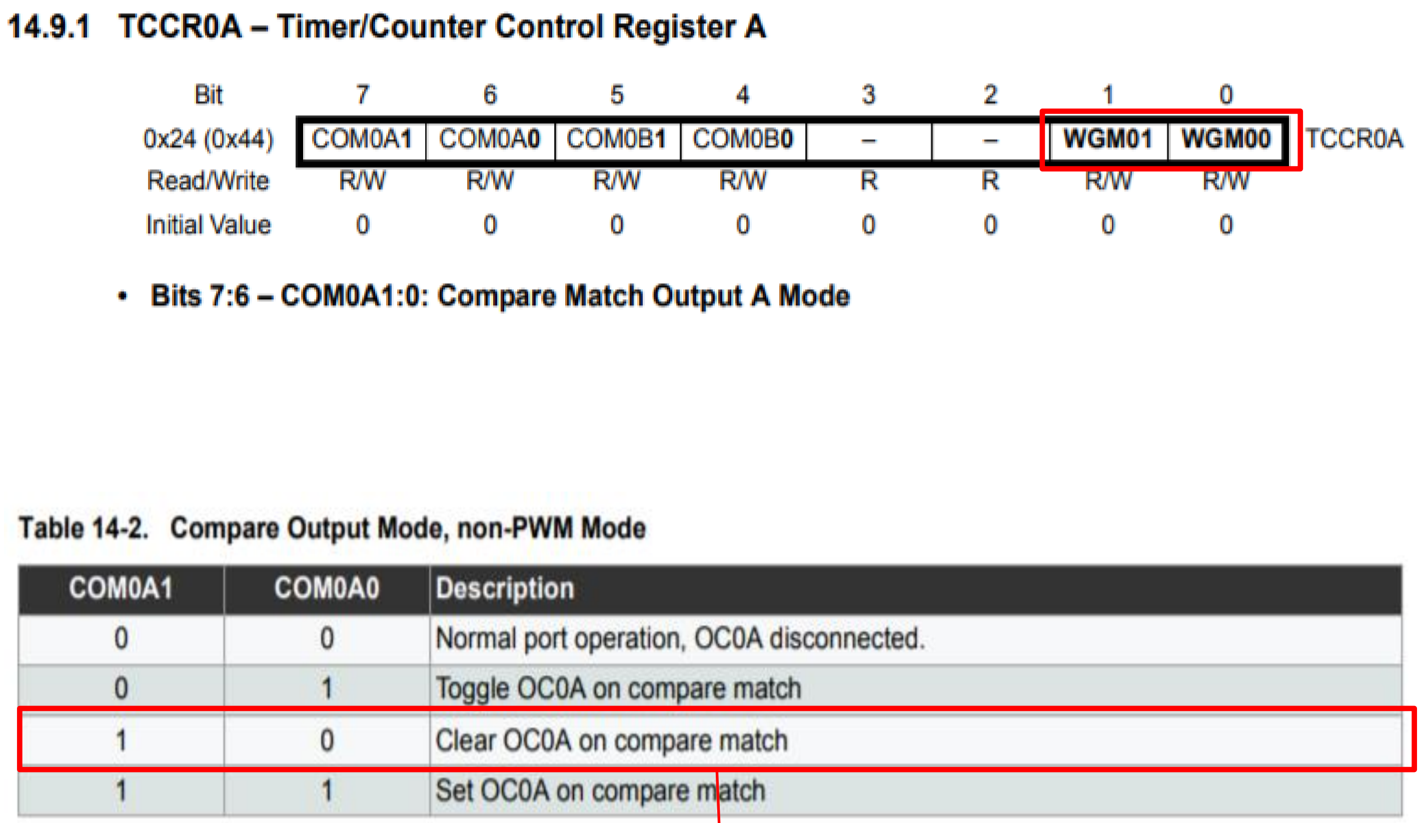
Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

1. 복습

(2) FAST PWM MODE(높은 주파수, 낮은 분해능)
- Register Setting



```
void Fast_PWM_Mode(void)
{
    cbi(SREG, 7);

    sbi(TCCR0A, WGM01);
    sbi(TCCR0A, WGM00);
    sbi(TCCR0A, COM0A1);
    //sbi(TCCR0A, COM0A0);
    sbi(TCCR0B, CS02);
    sbi(TCCR0B, CS00);
    sbi(TIMSK0, OCIE0A);
    OCR0A = 0; // fsysclk/N*(OCR0A+1)
    PORTD = 0xFF;
    DDRD = 0xFF;

    sbi(SREG, 7);
}
```



14.9.1 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	

- **Bits 7:6 – COM0A1:0: Compare Match Output A Mode**

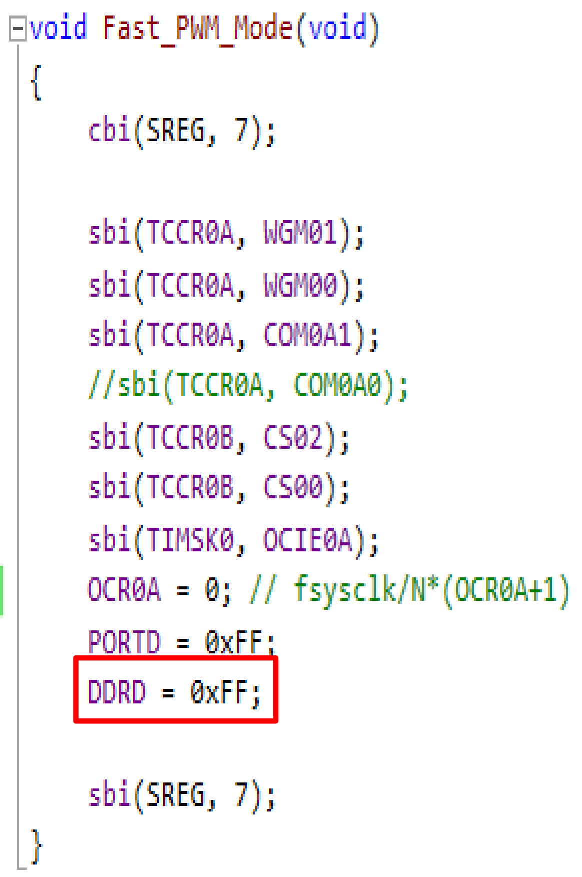
Table 14-2. Compare Output Mode, non-PWM Mode

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

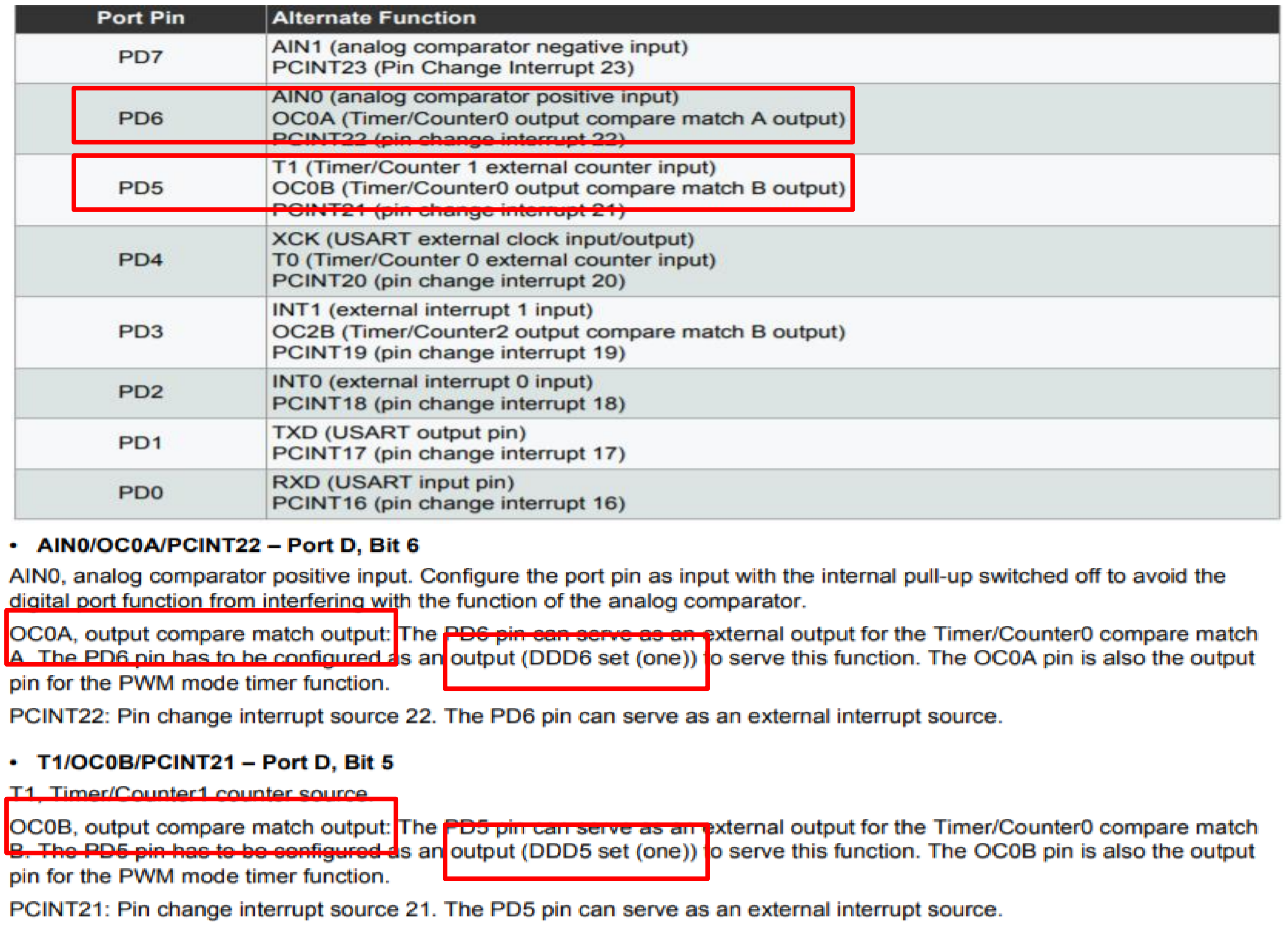
TCNT값과 OCR값이 일치하면
OC핀을 "L"로 출력

1. 복습

(2) FAST PWM MODE(높은 주파수, 낮은 분해능)
- Register Setting
- 출력핀 설정



```
void Fast_PWM_Mode(void)
{
    cbi(SREG, 7);

    sbi(TCCR0A, WGM01);
    sbi(TCCR0A, WGM00);
    sbi(TCCR0A, COM0A1);
    //sbi(TCCR0A, COM0A0);
    sbi(TCCR0B, CS02);
    sbi(TCCR0B, CS00);
    sbi(TIMSK0, OCIE0A);
    OCR0A = 0; // fsysclk/N*(OCR0A+1)
    PORTD = 0xFF;
    DDRD = 0xFF;

    sbi(SREG, 7);
}
```



Port Pin	Alternate Function
PD7	AIN1 (analog comparator negative input) PCINT23 (Pin Change Interrupt 23)
PD6	AIN0 (analog comparator positive input) OC0A (Timer/Counter0 output compare match A output) PCINT22 (pin change interrupt 22)
PD5	T1 (Timer/Counter 1 external counter input) OC0B (Timer/Counter0 output compare match B output) PCINT21 (pin change interrupt 21)
PD4	XCK (USART external clock input/output) T0 (Timer/Counter 0 external counter input) PCINT20 (pin change interrupt 20)
PD3	INT1 (external interrupt 1 input) OC2B (Timer/Counter2 output compare match B output) PCINT19 (pin change interrupt 19)
PD2	INT0 (external interrupt 0 input) PCINT18 (pin change interrupt 18)
PD1	TXD (USART output pin) PCINT17 (pin change interrupt 17)
PD0	RXD (USART input pin) PCINT16 (pin change interrupt 16)

- **AIN0/OC0A/PCINT22 – Port D, Bit 6**

AIN0, analog comparator positive input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the analog comparator.

OC0A, output compare match output: The PD6 pin can serve as an external output for the Timer/Counter0 compare match A. The PD6 pin has to be configured as an output (DDD6 set (one)) to serve this function. The OC0A pin is also the output pin for the PWM mode timer function.

PCINT22: Pin change interrupt source 22. The PD6 pin can serve as an external interrupt source.

- **T1/OC0B/PCINT21 – Port D, Bit 5**

T1, Timer/Counter1 counter source.

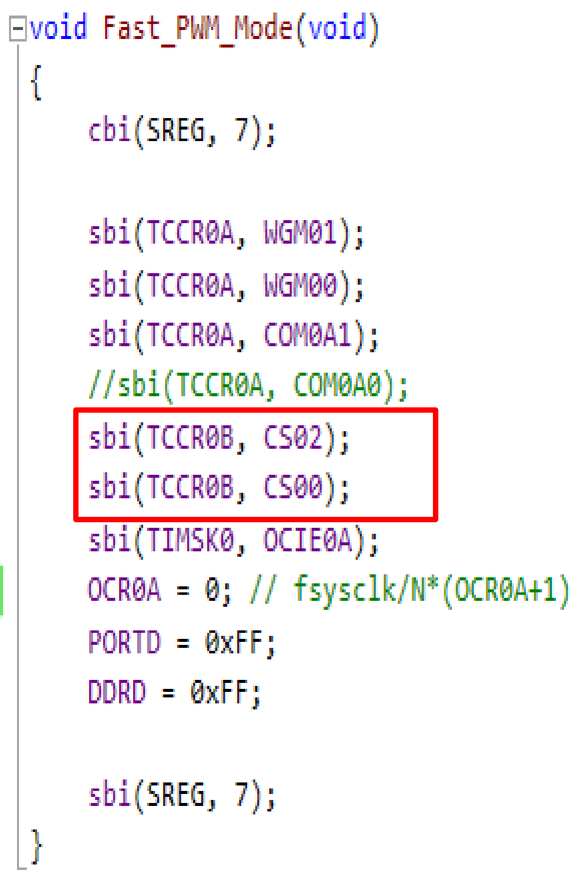
OC0B, output compare match output: The PD5 pin can serve as an external output for the Timer/Counter0 compare match B. The PD5 pin has to be configured as an output (DDD5 set (one)) to serve this function. The OC0B pin is also the output pin for the PWM mode timer function.

PCINT21: Pin change interrupt source 21. The PD5 pin can serve as an external interrupt source.

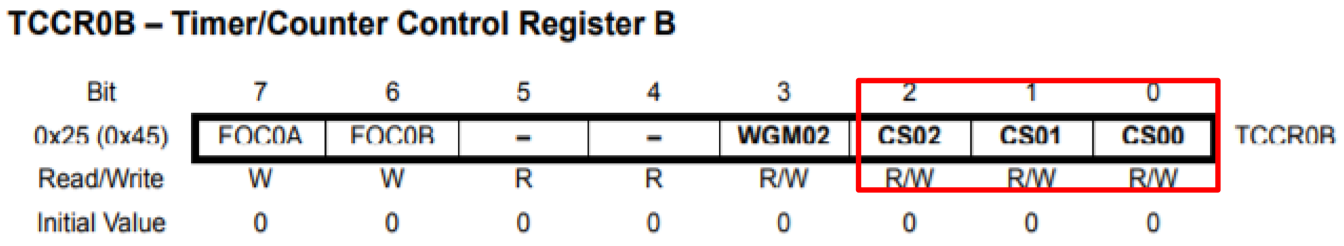
1. 복습

(2) FAST PWM MODE(높은 주파수, 낮은 분해능)

- Register Setting
- SysCLK = 16MHz



```
void Fast_PWM_Mode(void)
{
    cbi(SREG, 7);

    sbi(TCCR0A, WGM01);
    sbi(TCCR0A, WGM00);
    sbi(TCCR0A, COM0A1);
    //sbi(TCCR0A, COM0A0);
    sbi(TCCR0B, CS02);
    sbi(TCCR0B, CS00);
    sbi(TIMSK0, OCIE0A);
    OCR0A = 0; // fsysclk/N*(OCR0A+1)
    PORTD = 0xFF;
    DDRD = 0xFF;

    sbi(SREG, 7);
}
```



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	

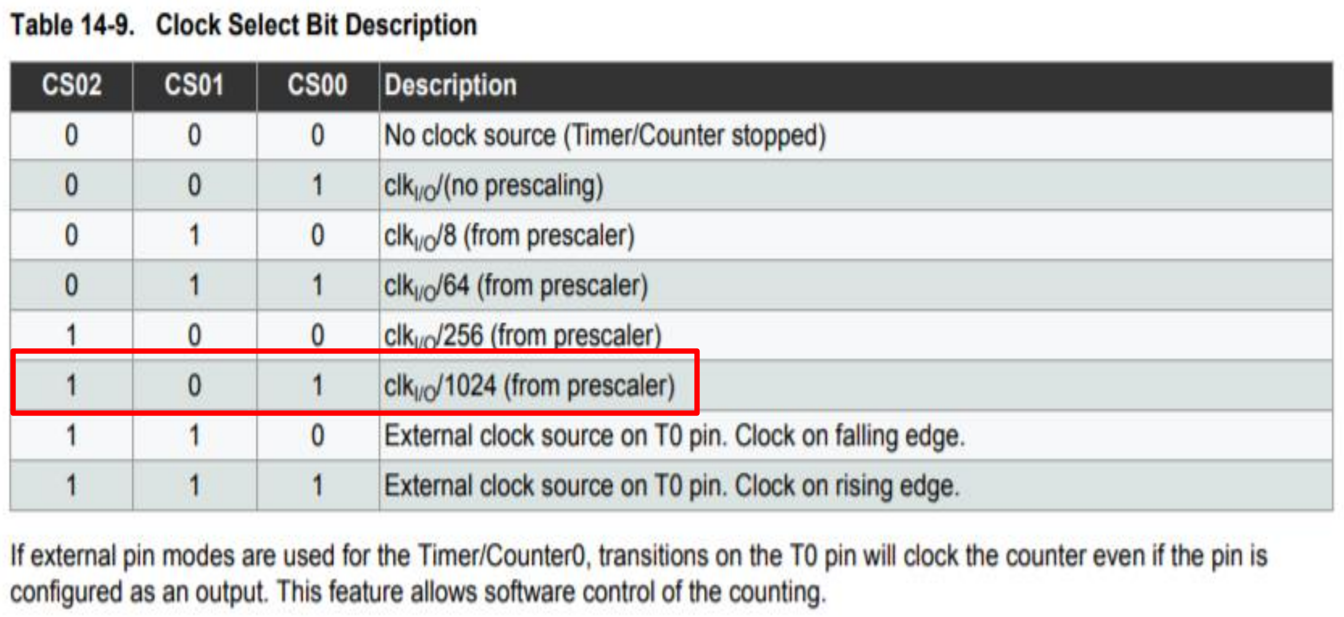


Table 14-9. Clock Select Bit Description

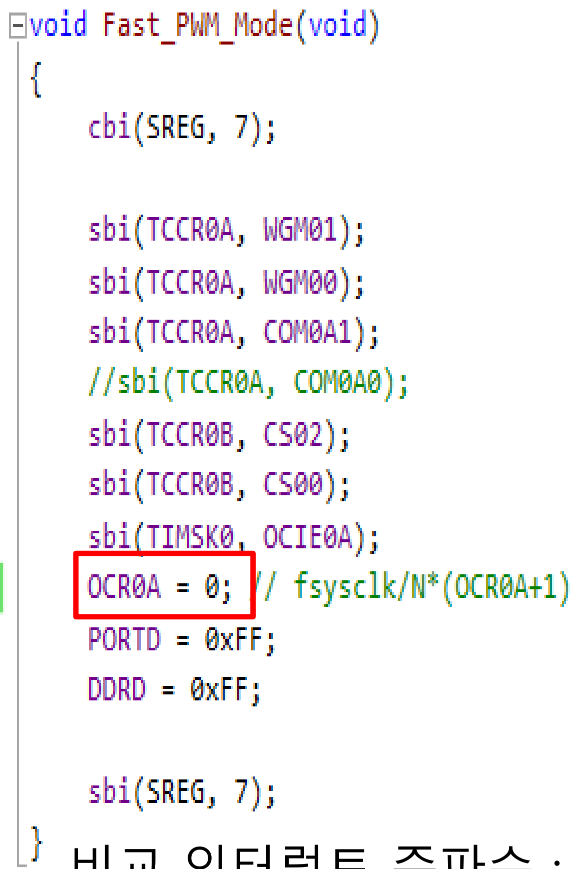
CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock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

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

1. 복습

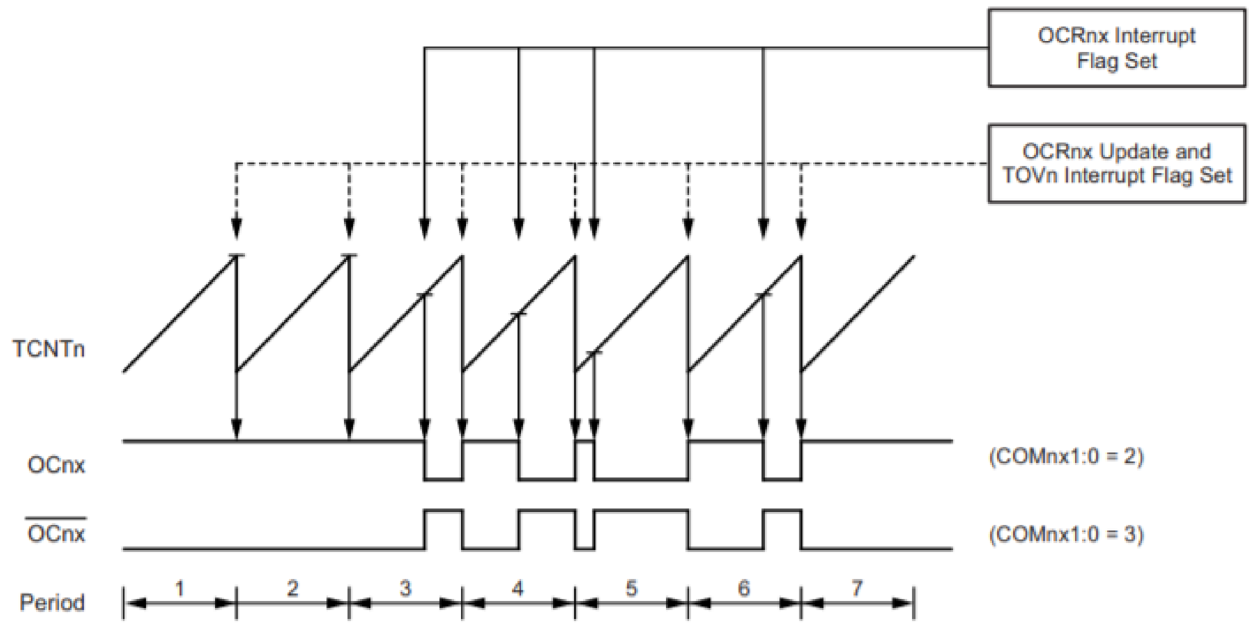
(2) FAST PWM MODE(높은 주파수, 낮은 분해능)

- Register Setting
- SysCLK = 16MHz



```
void Fast_PWM_Mode(void)
{
    cbi(SREG, 7);

    sbi(TCCR0A, WGM01);
    sbi(TCCR0A, WGM00);
    sbi(TCCR0A, COM0A1);
    //sbi(TCCR0A, COM0A0);
    sbi(TCCR0B, CS02);
    sbi(TCCR0B, CS00);
    sbi(TIMSK0, OCIE0A);
    OCR0A = 0; // fsysclk/N*(OCR0A+1)
    PORTD = 0xFF;
    DDRD = 0xFF;

    sbi(SREG, 7);
}
```

Figure 14-6. Fast PWM Mode, Timing Diagram



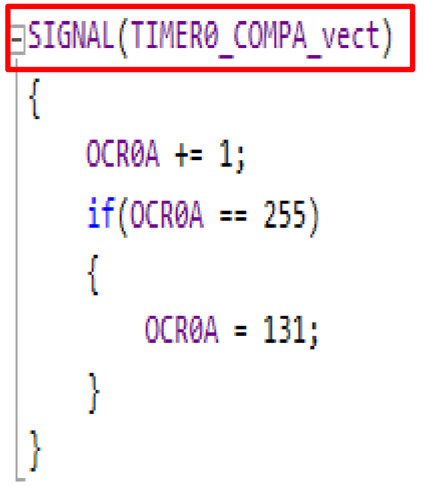
$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \times 256}$$

- 비교 인터럽트 주파수 : FclkI/O / N * 256
- 출력 파형 주파수는 인터럽트 주파수와 같다
- 설정된 주파수 : 16MHz / 1024 * 256 = 279Hz(약 3.6ms)

1. 복습

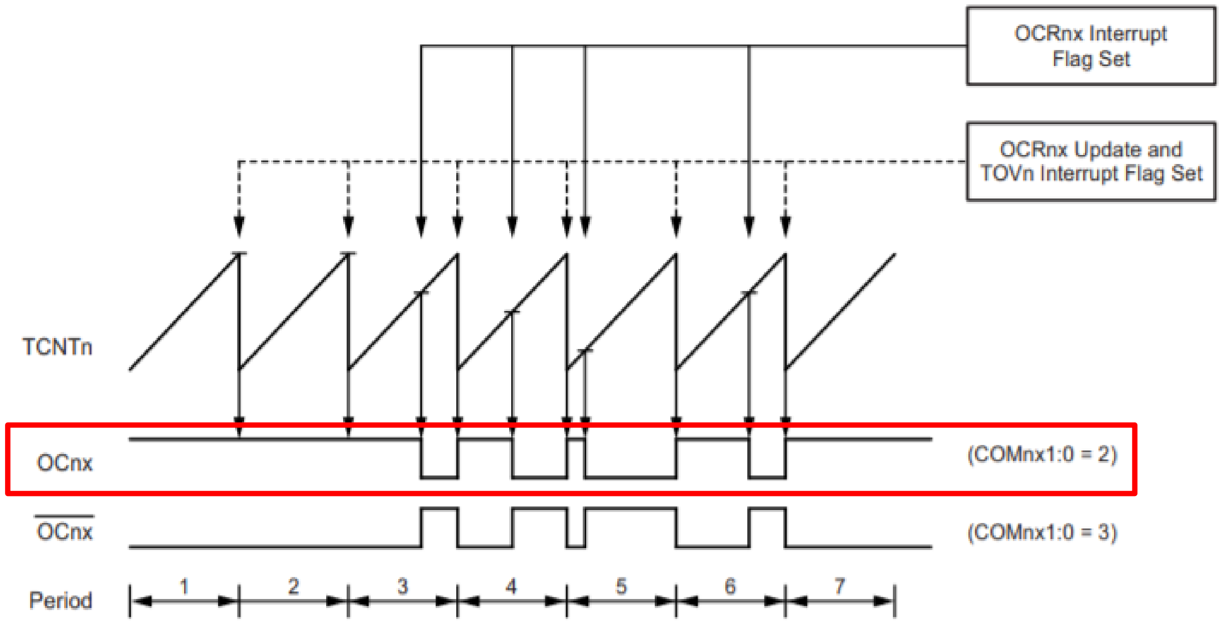
(2) FAST PWM MODE(높은 주파수, 낮은 분해능)

- FAST PWM MODE 테스트



```
SIGNAL(TIMER0_COMPA_vect)
{
    OCR0A += 1;
    if(OCR0A == 255)
    {
        OCR0A = 131;
    }
}
```

Figure 14-6. Fast PWM Mode, Timing Diagram



- 매 3.6ms마다 인터럽트 발생
  OCR값을 1씩 증가시킴

- 8bit 분해능으로 0~255값을 가지고

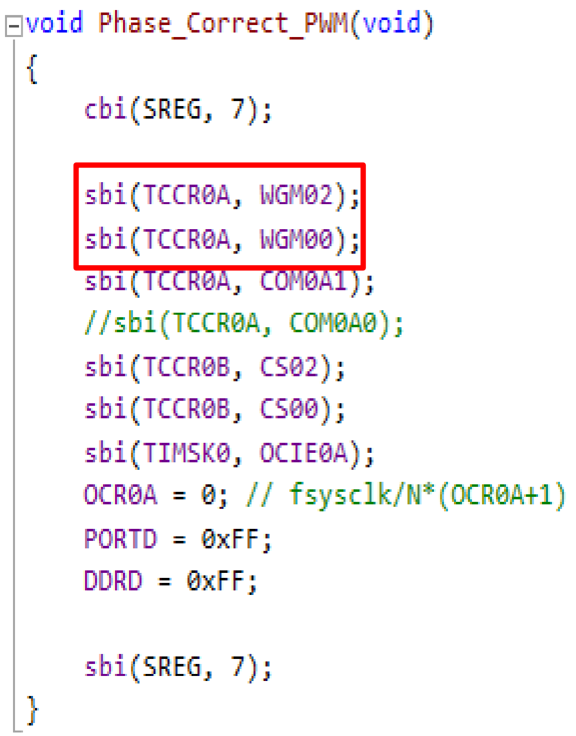
- 255일때 최대 듀티 100%가 됨

- OCR값 1씩 증가할 때마다 듀티비는
  (1/256) * 100 = 약 4%씩 증가

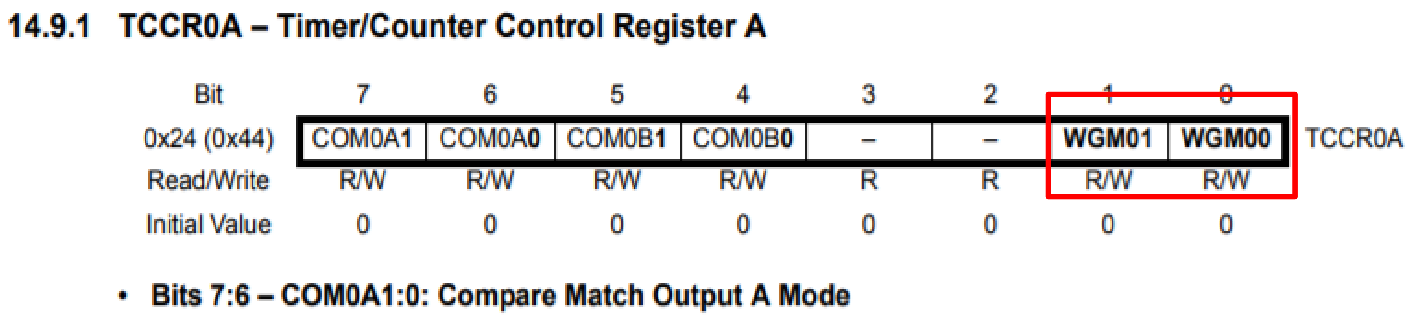
1. 복습

(3) Phase Correct PWM MODE(높은 주파수, 낮은 분해능)

- Register Setting(Fast PWM과 동일하나 MODE 설정부분만 다름..)



```
void Phase_Correct_PWM(void)
{
    cbi(SREG, 7);

    sbi(TCCR0A, WGM02);
    sbi(TCCR0A, WGM00);
    sbi(TCCR0A, COM0A1);
    //sbi(TCCR0A, COM0A0);
    sbi(TCCR0B, CS02);
    sbi(TCCR0B, CS00);
    sbi(TIMSK0, OCIE0A);
    OCR0A = 0; // fsysclk/N*(OCR0A+1)
    PORTD = 0xFF;
    DDRD = 0xFF;

    sbi(SREG, 7);
}
```



14.9.1 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	

- **Bits 7:6 – COM0A1:0: Compare Match Output A Mode**

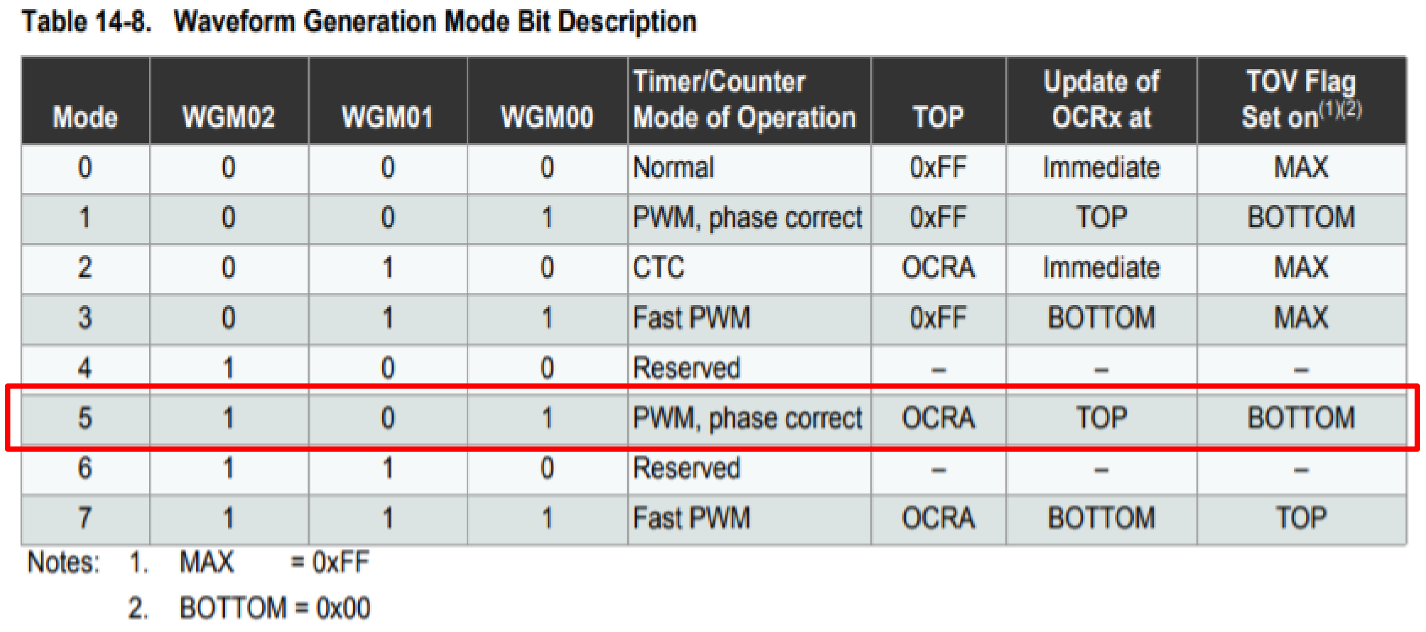


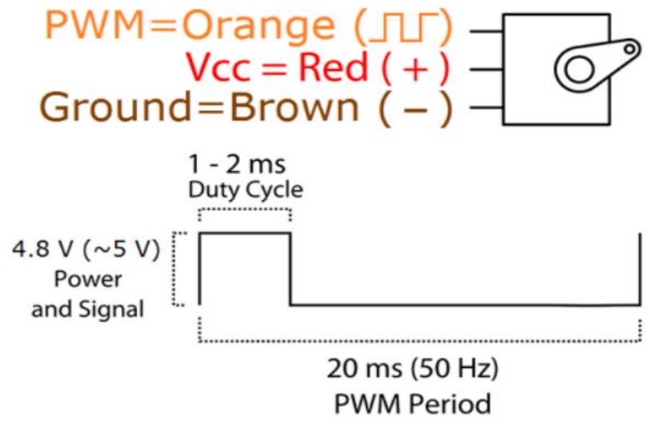
Table 14-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[(1)(2)]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

1. 복습

(4) Servo Motor(SG90) 구동



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.

- 위 서보모터는 0~180도까지 회전이 가능하다.

- 서보모터를 구동하기 위해 주기가 20ms이고

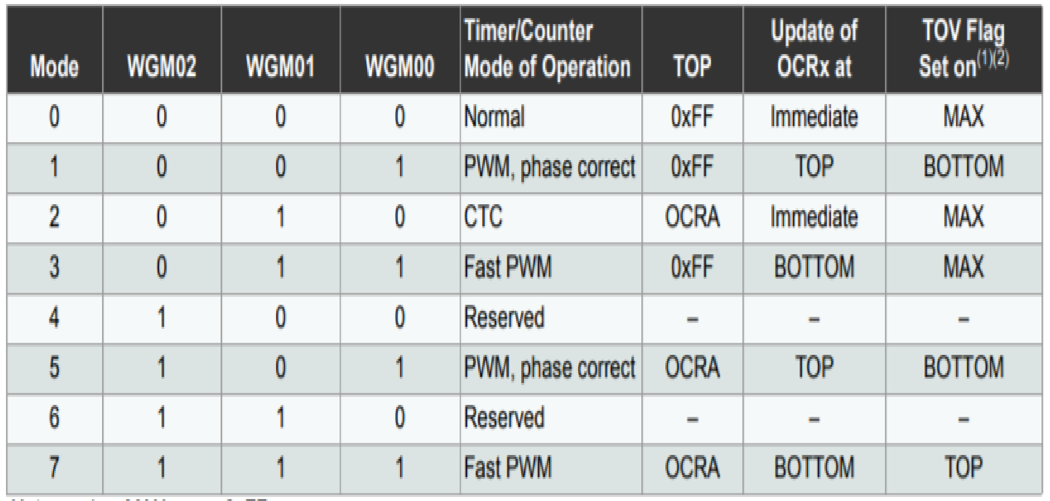
- 0도 : 1.5ms, 90도 : ~2ms, -90 : ~1ms에 해당하는 듀티사이클을 만들어줘야 한다.

1. 복습

(4) Servo Motor(SG90) 구동

- 8bit Timer/Counter로 정상구동이 안된다..(MCU 외부 클럭이 느리면 가능..)
- 8bit Timer/Counter의 PWM모드들 중 PWM 최대 주기는 Phase Correct PWM의 7.2ms이다.
- 분주비 최대는 1024이고, 8bit 분해능은 256으로 16MHz / 1024*256 =3.6ms이고
- Phase Correct PWM은 Fast PWM의 두배 주기를 가지므로 7.2ms가 최대이다..

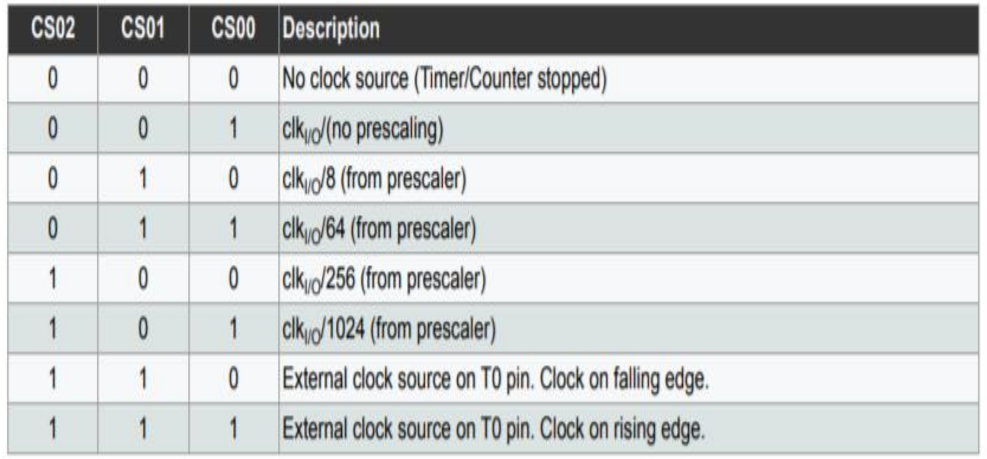
Table 14-8. Waveform Generation Mode Bit Description



Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1][2]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

Table 14-9. Clock Select Bit Description



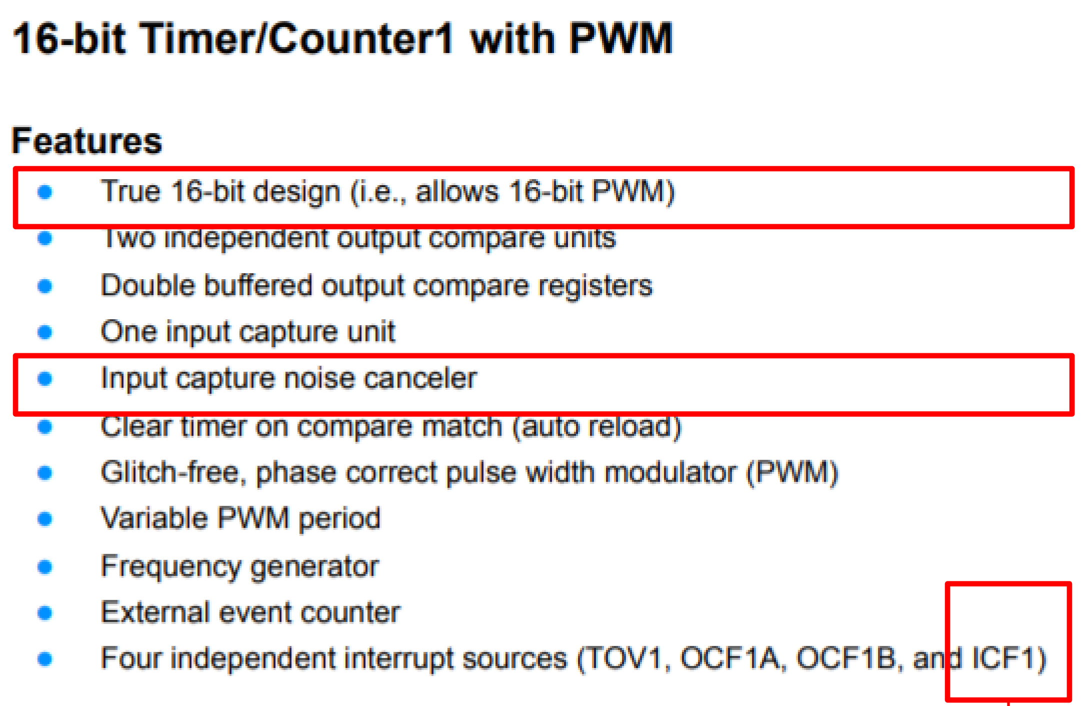
CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock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

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

- 해결 방법은 분해능이 더 높은 Timer/Counter모듈을 쓰거나
- 외부클럭을 50Hz의 주파수를 생성할 만큼 느린것으로 설정하면 됨.

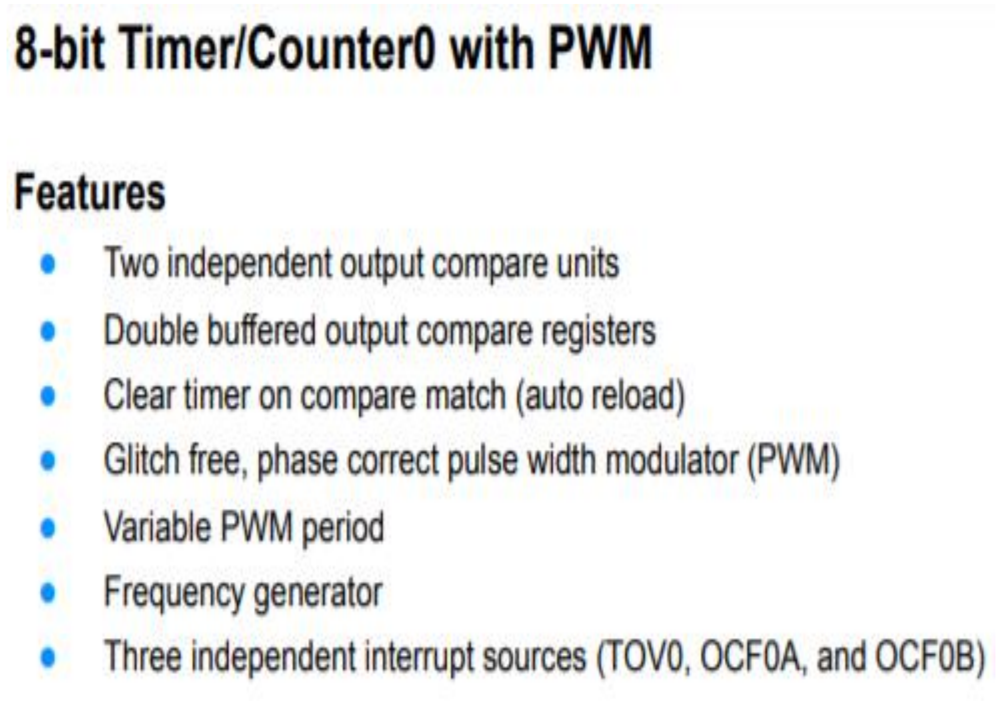
1. 복습

(4) Servo Motor(SG90) 구동
- 16bit Timer/counter이용(Timer/Counter1)



16-bit Timer/Counter1 with PWM

Features
- True 16-bit design (i.e., allows 16-bit PWM)
- Two independent output compare units
- Double buffered output compare registers
- One input capture unit
- Input capture noise canceler
- Clear timer on compare match (auto reload)
- Glitch-free, phase correct pulse width modulator (PWM)
- Variable PWM period
- Frequency generator
- External event counter
- Four independent interrupt sources (TOV1, OCF1A, OCF1B, and ICF1)



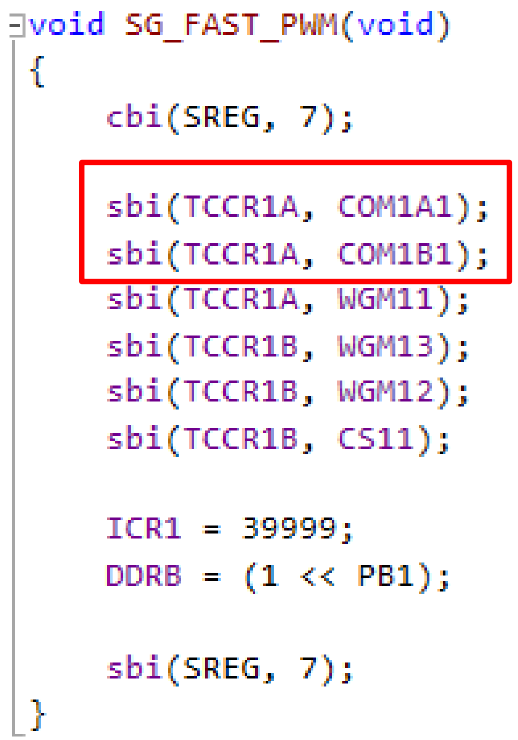
8-bit Timer/Counter0 with PWM

Features
- Two independent output compare units
- Double buffered output compare registers
- Clear timer on compare match (auto reload)
- Glitch free, phase correct pulse width modulator (PWM)
- Variable PWM period
- Frequency generator
- Three independent interrupt sources (TOV0, OCF0A, and OCF0B)

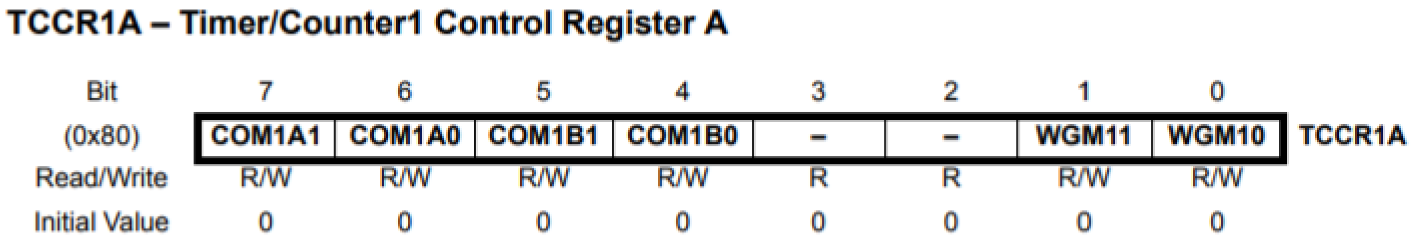
ICF라는 인터럽트 소스가 생김
ICF는 Input Capture Flag로
특정핀으로 출력된 파형을 감지하거나 외부 신호를 감지할 수 있는 기능이 추가됨

1. 복습

(4) Servo Motor(SG90) 구동
- 16bit Timer/counter이용(Timer/Counter1)
- Register Setting (16MHz)



```
void SG_FAST_PWM(void)
{
    cbi(SREG, 7);

    sbi(TCCR1A, COM1A1);
    sbi(TCCR1A, COM1B1);
    sbi(TCCR1A, WGM11);
    sbi(TCCR1B, WGM13);
    sbi(TCCR1B, WGM12);
    sbi(TCCR1B, CS11);

    ICR1 = 39999;
    DDRB = (1 << PB1);

    sbi(SREG, 7);
}
```



TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
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	

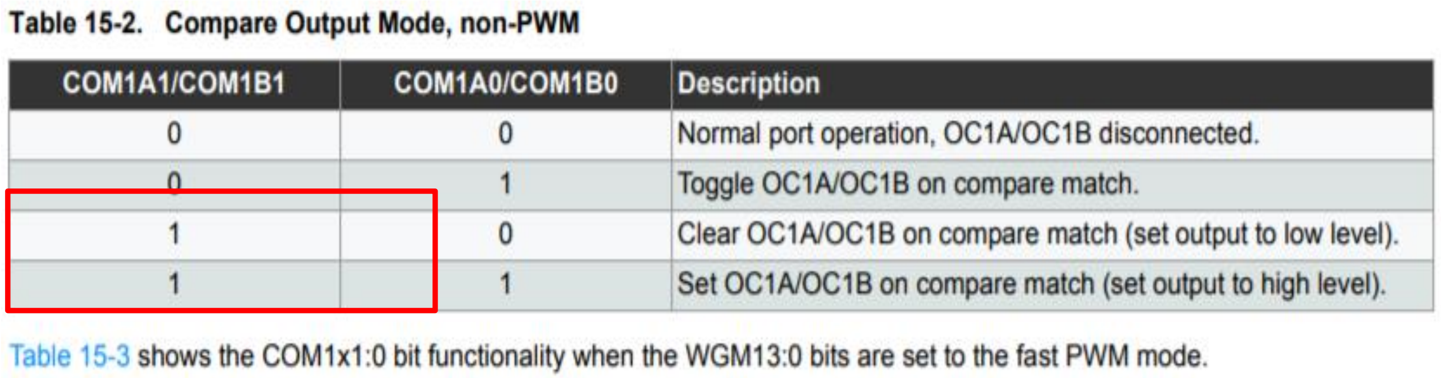


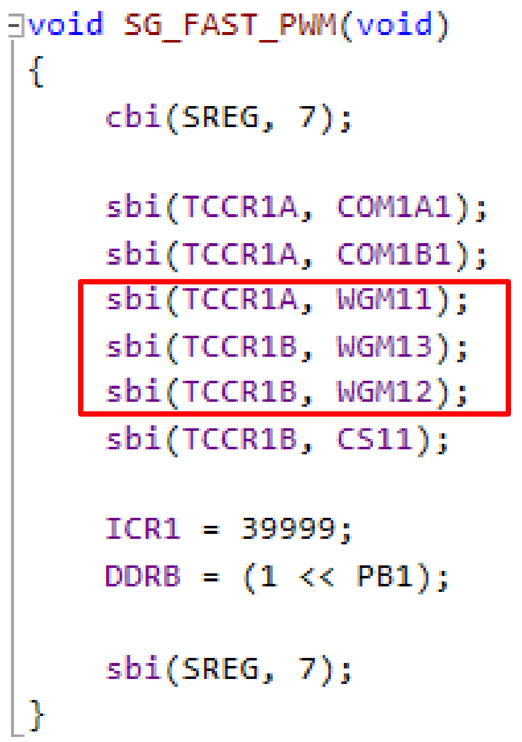
Table 15-2. Compare Output Mode, non-PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on compare match.
1	0	Clear OC1A/OC1B on compare match (set output to low level).
1	1	Set OC1A/OC1B on compare match (set output to high level).

Table 15-3 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the fast PWM mode.

1. 복습

(4) Servo Motor(SG90) 구동
- 16bit Timer/counter이용(Timer/Counter1)
- Register Setting (16MHz)



```
void SG_FAST_PWM(void)
{
    cbi(SREG, 7);

    sbi(TCCR1A, COM1A1);
    sbi(TCCR1A, COM1B1);
    sbi(TCCR1A, WGM11);
    sbi(TCCR1B, WGM13);
    sbi(TCCR1B, WGM12);
    sbi(TCCR1B, CS11);

    ICR1 = 39999;
    DDRB = (1 << PB1);

    sbi(SREG, 7);
}
```

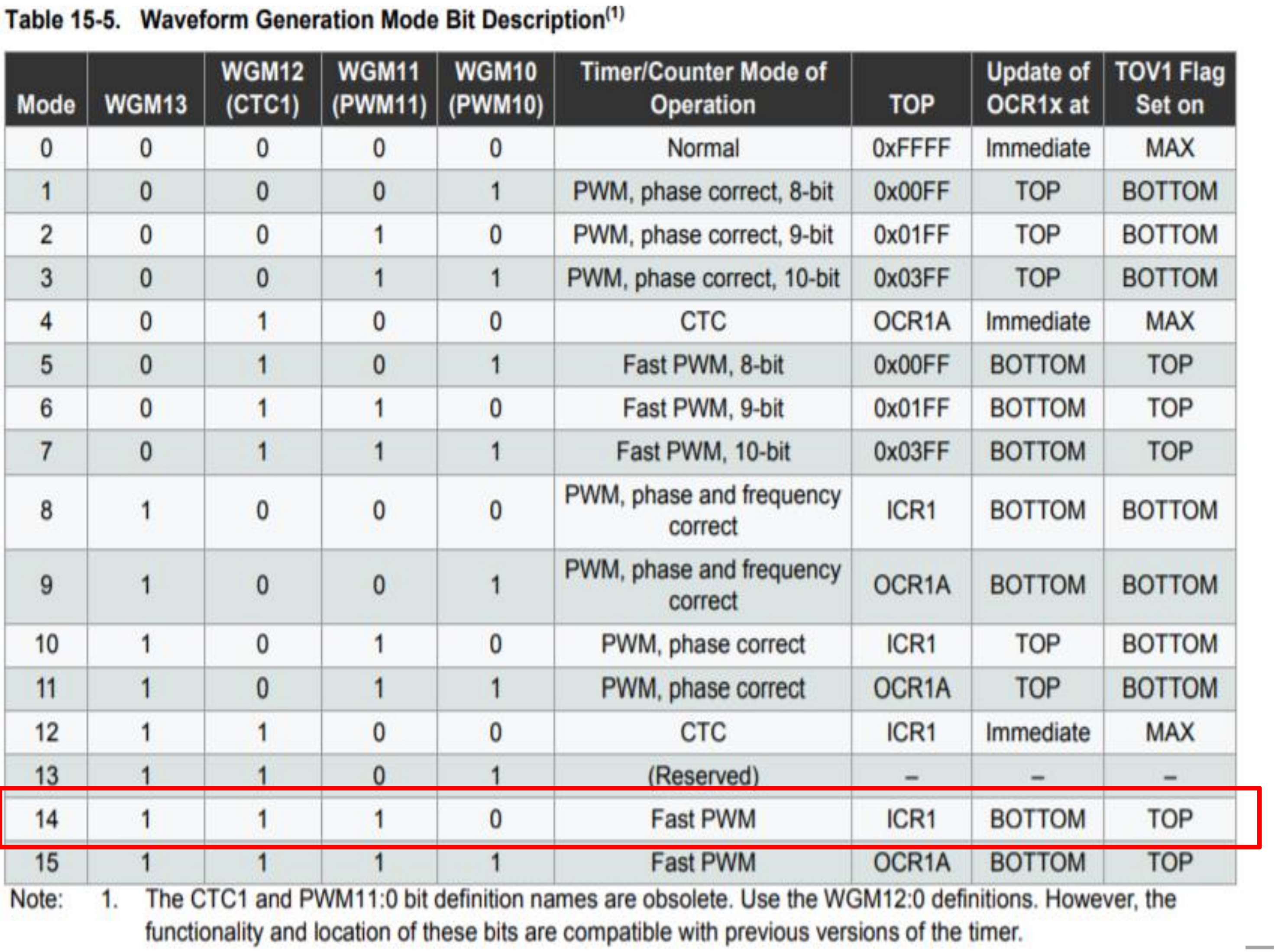


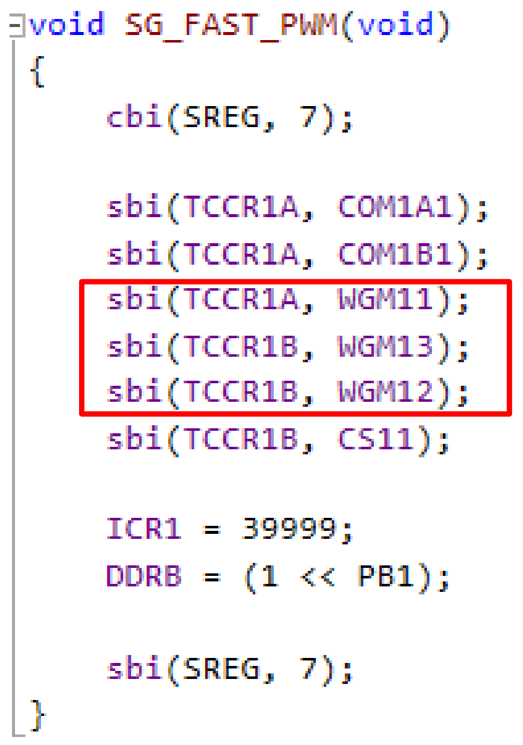
Table 15-5. Waveform Generation Mode Bit Description[1]

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, phase correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, phase correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, phase correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, phase and frequency correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, phase and frequency correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, phase correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, phase correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

1. 복습

(4) Servo Motor(SG90) 구동
- 16bit Timer/counter이용(Timer/Counter1)
- Register Setting (16MHz)



```
void SG_FAST_PWM(void)
{
    cbi(SREG, 7);

    sbi(TCCR1A, COM1A1);
    sbi(TCCR1A, COM1B1);
    sbi(TCCR1A, WGM11);
    sbi(TCCR1B, WGM13);
    sbi(TCCR1B, WGM12);
    sbi(TCCR1B, CS11);

    ICR1 = 39999;
    DDRB = (1 << PB1);

    sbi(SREG, 7);
}
```

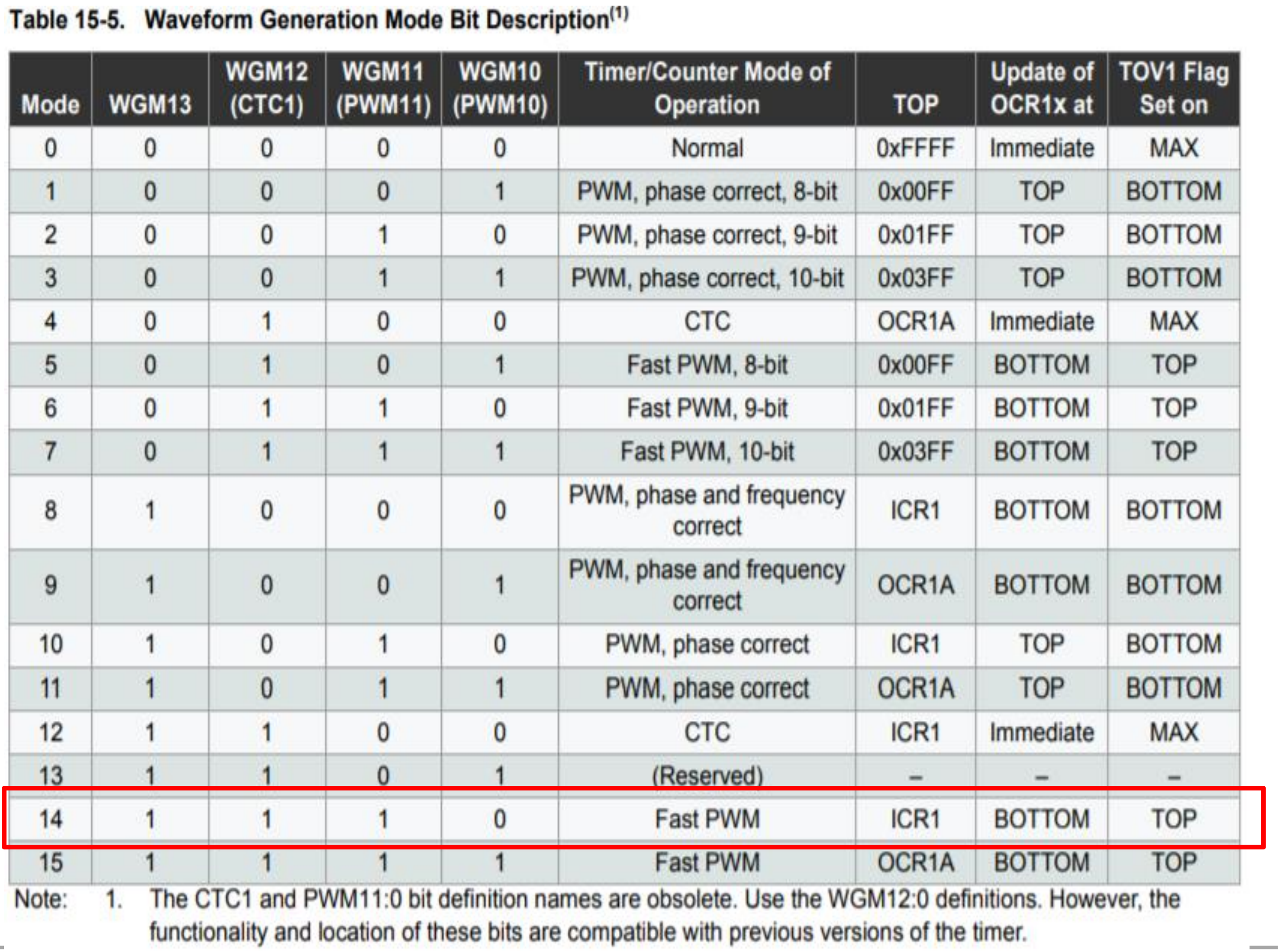


Table 15-5. Waveform Generation Mode Bit Description[1]

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, phase correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, phase correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, phase correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, phase and frequency correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, phase and frequency correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, phase correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, phase correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

1. 복습

(4) Servo Motor(SG90) 구동
- 16bit Timer/counter이용(Timer/Counter1)

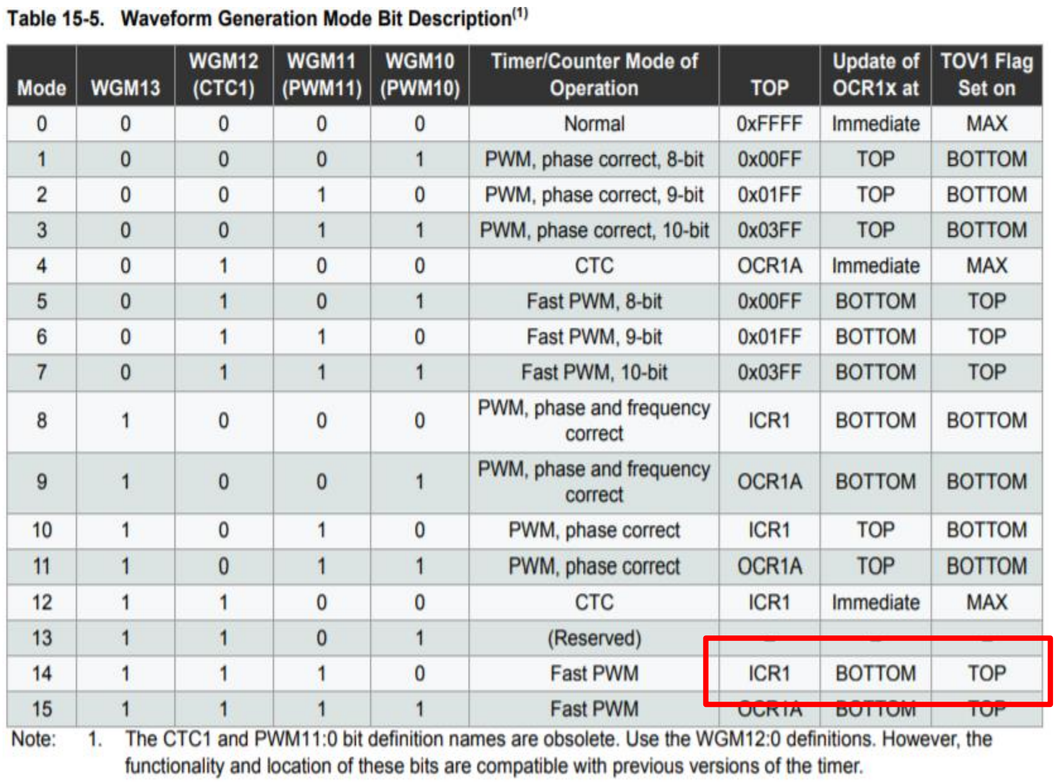
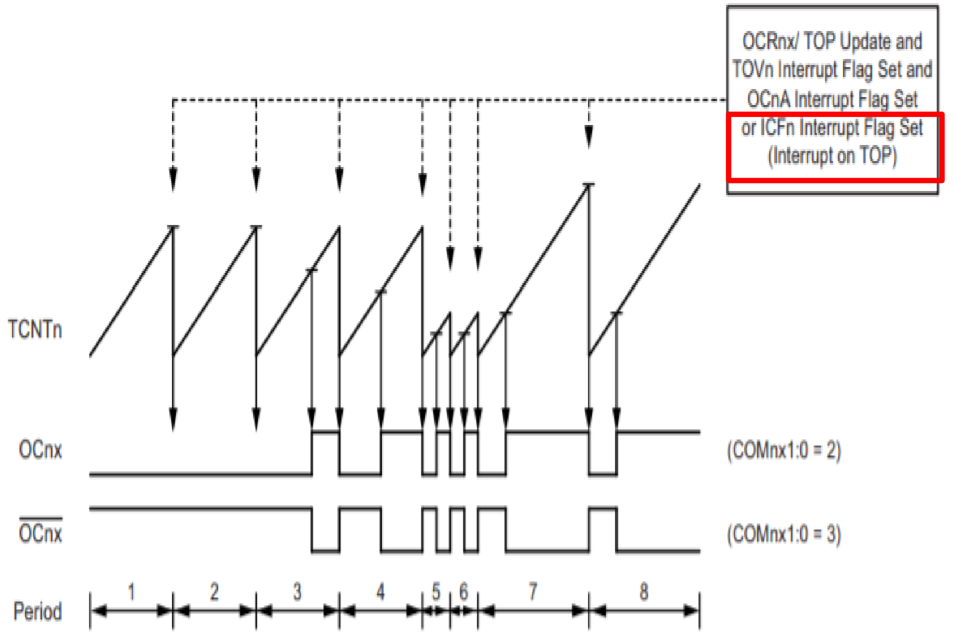


Table 15-5. Waveform Generation Mode Bit Description[1]

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, phase correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, phase correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, phase correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, phase and frequency correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, phase and frequency correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, phase correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, phase correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

Figure 15-7. Fast PWM Mode, Timing Diagram

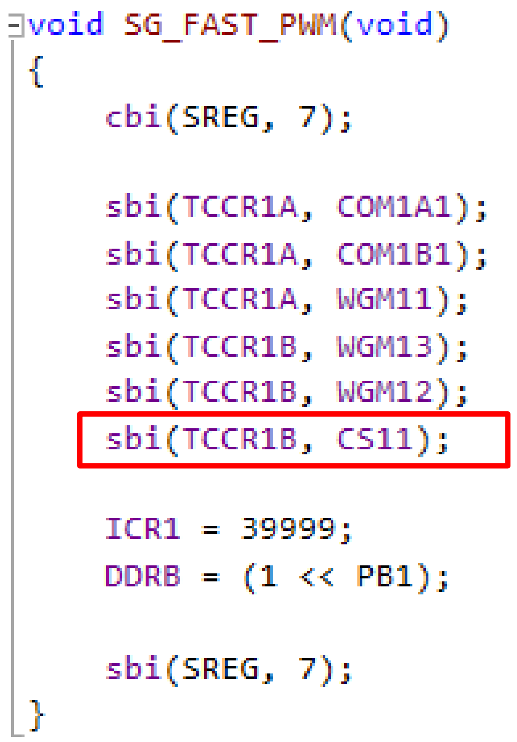


OCRnx/ TOP Update and TOVn Interrupt Flag Set and OCnA Interrupt Flag Set or ICFn Interrupt Flag Set (Interrupt on TOP)
TCNTn
OCnx
(COMnx1:0 = 2)
OCnx
(COMnx1:0 = 3)
Period
1
2
3
4
5
6
7
8

- 16bit의 분해능을 이용하여 기준 주파수 생성을 위해 TOP값을 ICR1값으로 설정하면 원하는 주파수를 설정할 수 있다.

1. 복습

(4) Servo Motor(SG90) 구동
- 16bit Timer/counter이용(Timer/Counter1)
- 기준 주파수 설정



```
void SG_FAST_PWM(void)
{
    cbi(SREG, 7);

    sbi(TCCR1A, COM1A1);
    sbi(TCCR1A, COM1B1);
    sbi(TCCR1A, WGM11);
    sbi(TCCR1B, WGM13);
    sbi(TCCR1B, WGM12);
    sbi(TCCR1B, CS11);

    ICR1 = 39999;
    DDRB = (1 << PB1);

    sbi(SREG, 7);
}
```

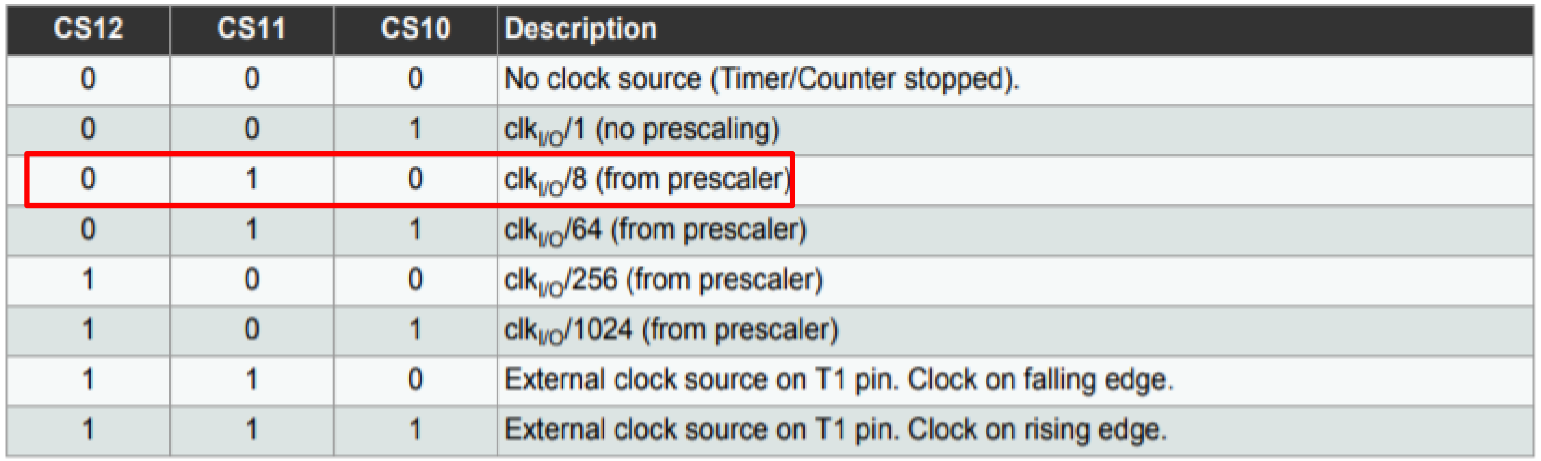


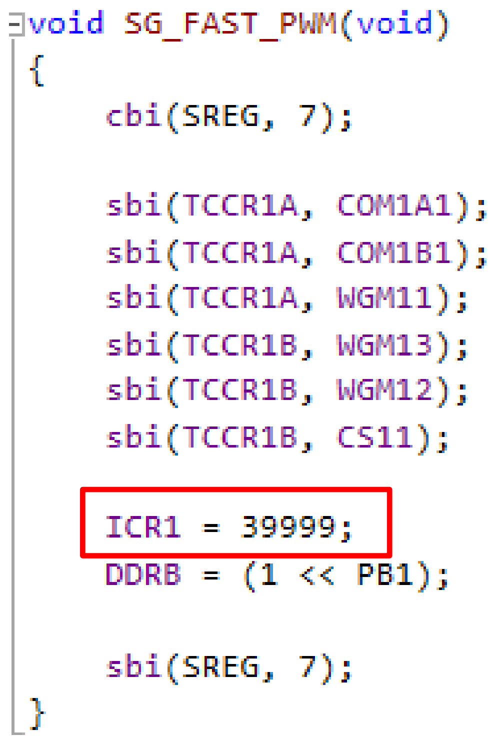
Table 15-6. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk$_{I/O}$/1 (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 T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

1. 복습

(4) Servo Motor(SG90) 구동

- 16bit Timer/counter이용(Timer/Counter1)
- 기준 주파수 설정



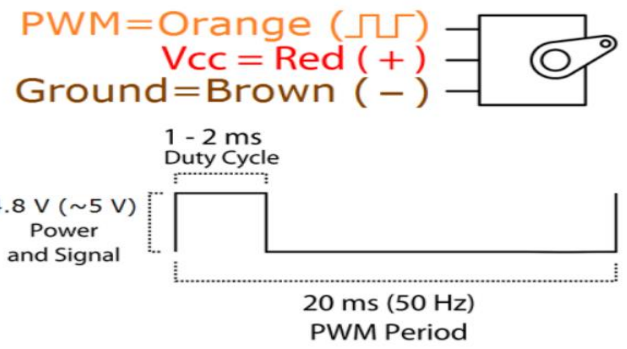
```c
void SG_FAST_PWM(void)
{
    cbi(SREG, 7);

    sbi(TCCR1A, COM1A1);
    sbi(TCCR1A, COM1B1);
    sbi(TCCR1A, WGM11);
    sbi(TCCR1B, WGM13);
    sbi(TCCR1B, WGM12);
    sbi(TCCR1B, CS11);

    ICR1 = 39999;
    DDRB = (1 << PB1);

    sbi(SREG, 7);
}
```

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \times (1 + TOP)}$$

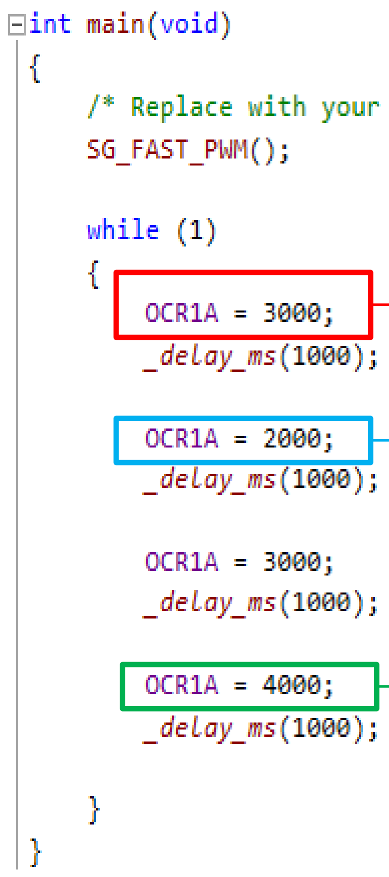
- 16MHz / 8 * (1+39999) = 50Hz(=20ms)
- 39999 = duty 100%

1. 복습

(4) Servo Motor(SG90) 구동
- 16bit Timer/counter이용(Timer/Counter1)
- 듀티 가변



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.



```
int main(void)
{
    /* Replace with your application code */
    SG_FAST_PWM();

    while (1)
    {
        OCR1A = 3000;
        _delay_ms(1000);

        OCR1A = 2000;
        _delay_ms(1000);

        OCR1A = 3000;
        _delay_ms(1000);

        OCR1A = 4000;
        _delay_ms(1000);

    }
}
```

- 3000 / 40000 = 0.075
- 40000 : 20ms = 3000 : 1.5ms
- 0도

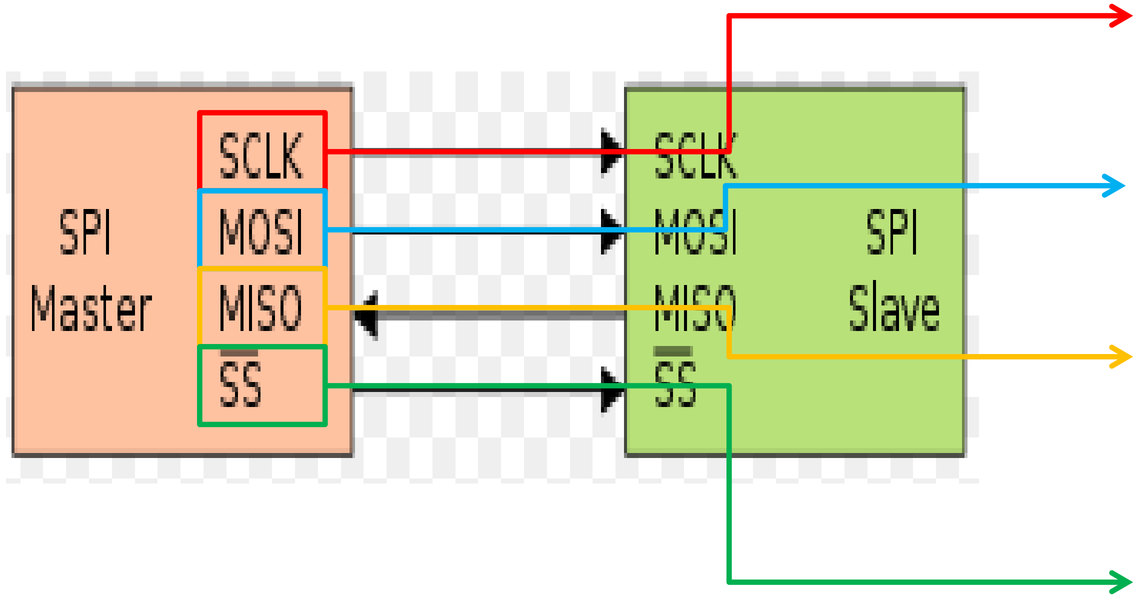
- 2000 / 40000 = 0.05
- 40000 : 20ms = 2000 : 1ms
- -90도

- 4000 / 40000 = 0.1
- 40000 : 20ms = 4000 : 2ms
- +90도

3. SPI 통신

(1) SPI 통신이란?

- Serial Peripheral Interface로 직렬통신이다.
- 마스터에 의해 CLK이 발생하고, CLK에 동기되어 데이터가 전송된다(동기 통신)
- Full-Duplex형식 : 송신과 수신이 동시에 가능함
- 1(Master) : N(Slave) 형태의 구조를 가짐
- 하드웨어 구성은 아래와 같다



통신 CLK Data는 CLK에 맞춰 전송된다.
설정에 따라 'H'에 동기 또는 'L'에 동기하여
전송되도록 설정가능

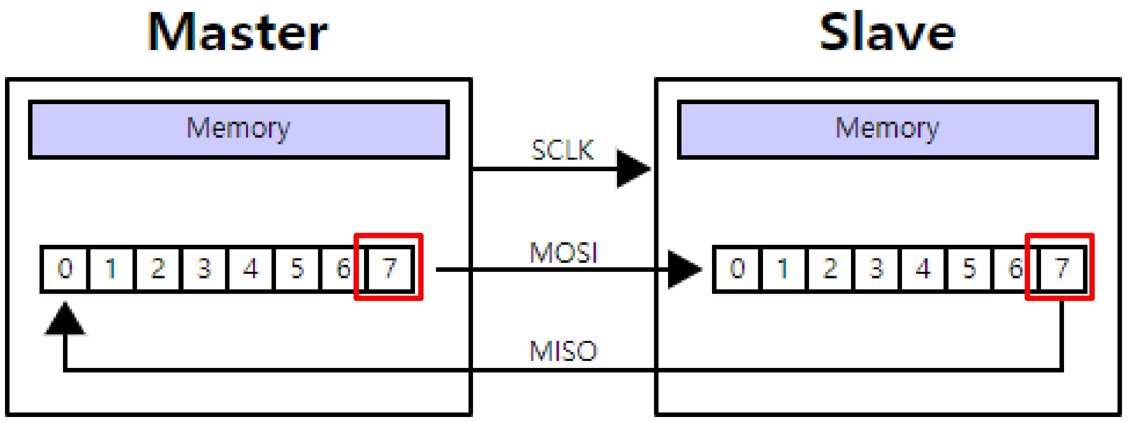
Master Out Slave In으로
Master -> Slave로 Data 전송을 위한 핀

Slave Out Master In으로
Slave -> Master로 Data 전송을 위한 핀

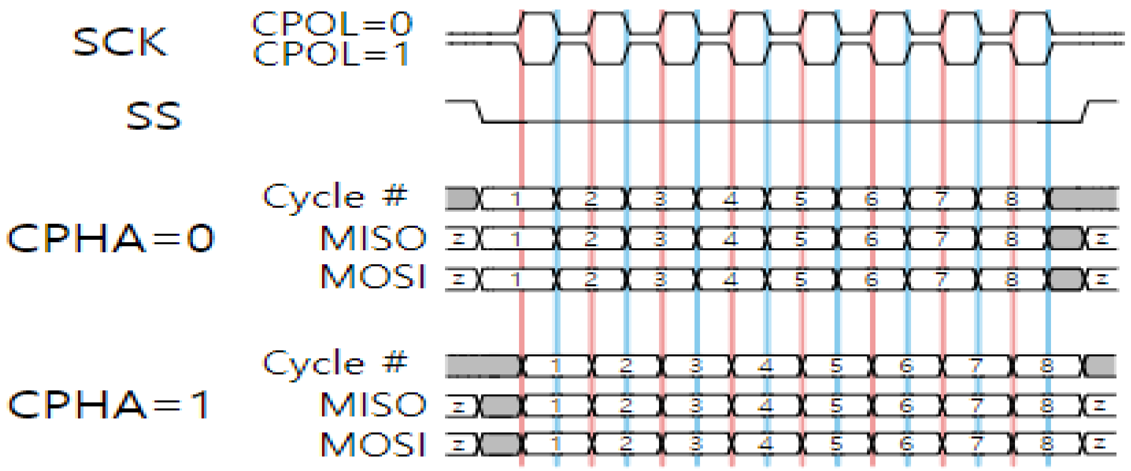
Slave Selct or CS(Chip Select)
1 : N통신시 Slave 선택을 위한 핀
Slave에 통신 시작을 알림

3. SPI 통신

(1) SPI 통신이란?
- Data전송
  : MSB부터 전송되어 수신측은 Shift Register로 원래 Data Bit순서를 만든다



MSB



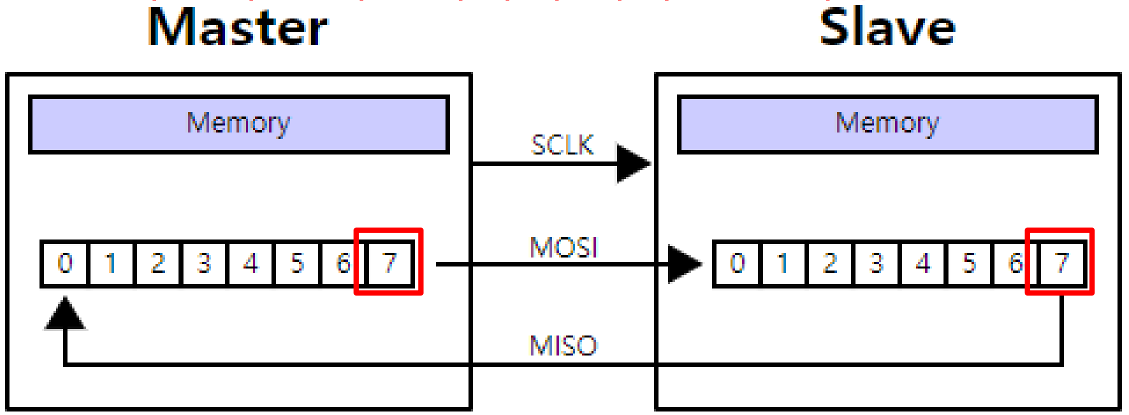
3. SPI 통신

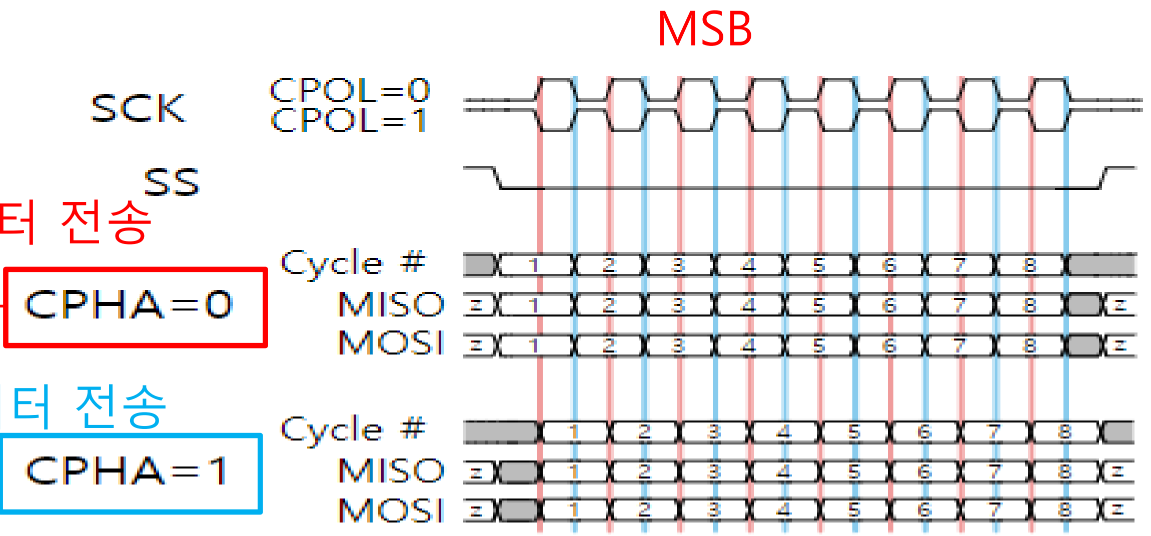
(1) SPI 통신이란?

- Data전송

  : MSB부터 전송되어 수신측은 Shift Register로 원래 Data Bit순서를 만든다.

  : MSB, LSB는 지원되는 하드웨어에 따라 설정가능한것도 있음!



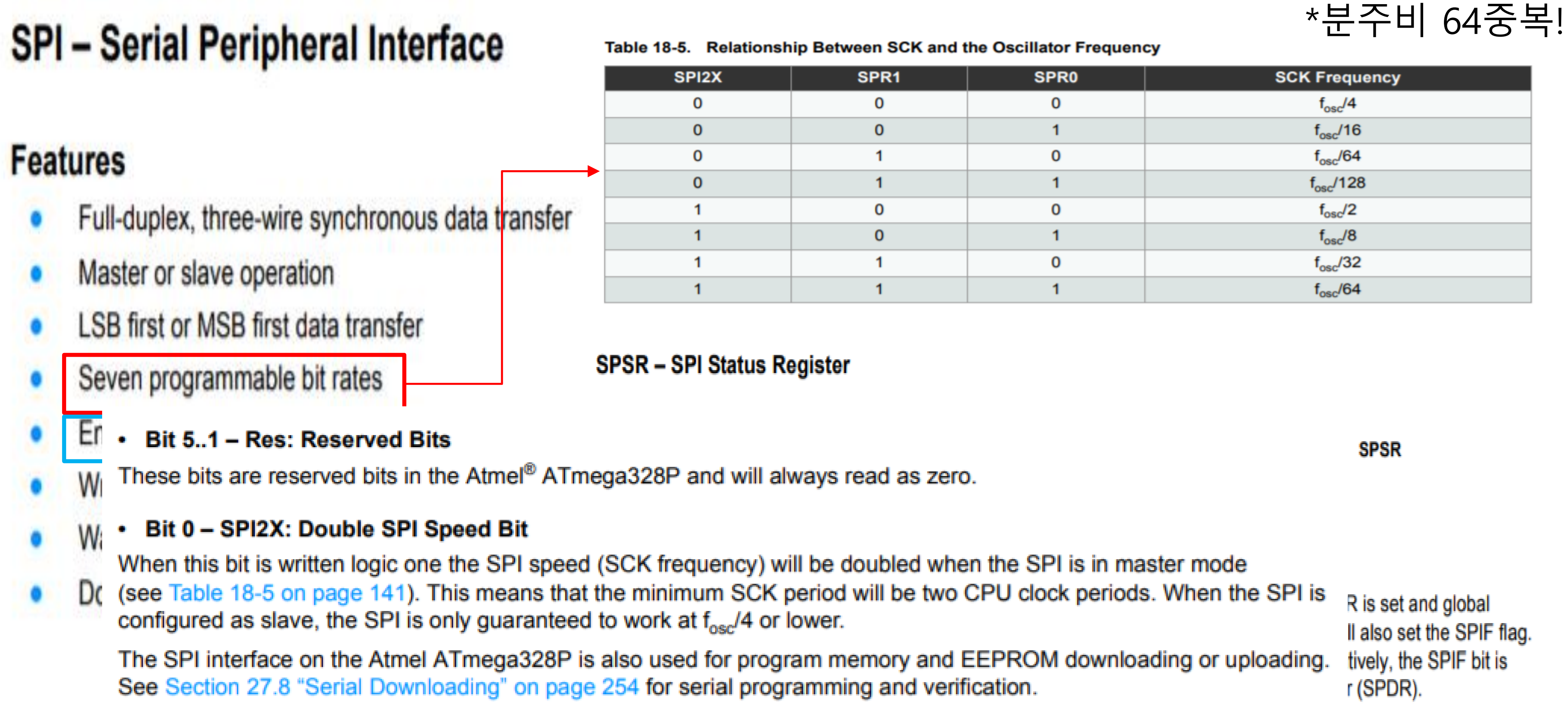


CLK의 "L"에 동기하여 데이터 전송

CLK의 "H"에 동기하여 데이터 전송

3. SPI 통신

(2) ATmega328의 SPI 모듈



*분주비 64중복!

SPI – Serial Peripheral Interface

Features

- Full-duplex, three-wire synchronous data transfer
- Master or slave operation
- LSB first or MSB first data transfer
- Seven programmable bit rates
- En
- Wi
- Wa
- Do

Table 18-5. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

SPSR – SPI Status Register

SPSR

- **Bit 5..1 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATmega328P and will always read as zero.

- **Bit 0 – SPI2X: Double SPI Speed Bit**

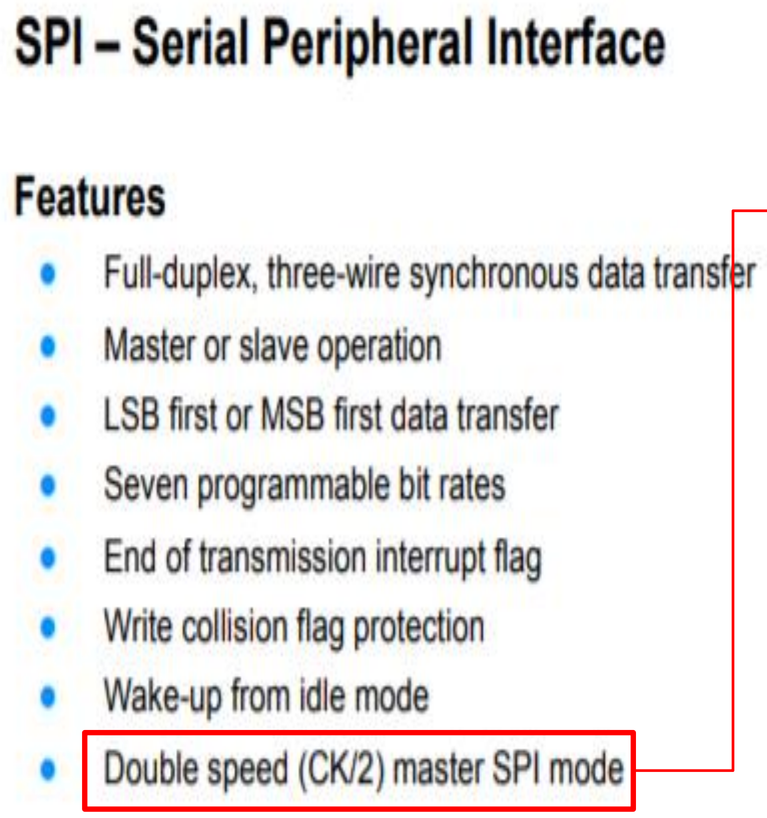
When this bit is written logic one the SPI speed (SCK frequency) will be doubled when the SPI is in master mode (see Table 18-5 on page 141). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as slave, the SPI is only guaranteed to work at $f_{osc}/4$ or lower.

The SPI interface on the Atmel ATmega328P is also used for program memory and EEPROM downloading or uploading. See Section 27.8 "Serial Downloading" on page 254 for serial programming and verification.

R is set and global
ll also set the SPIF flag.
tively, the SPIF bit is
r (SPDR).

3. SPI 통신

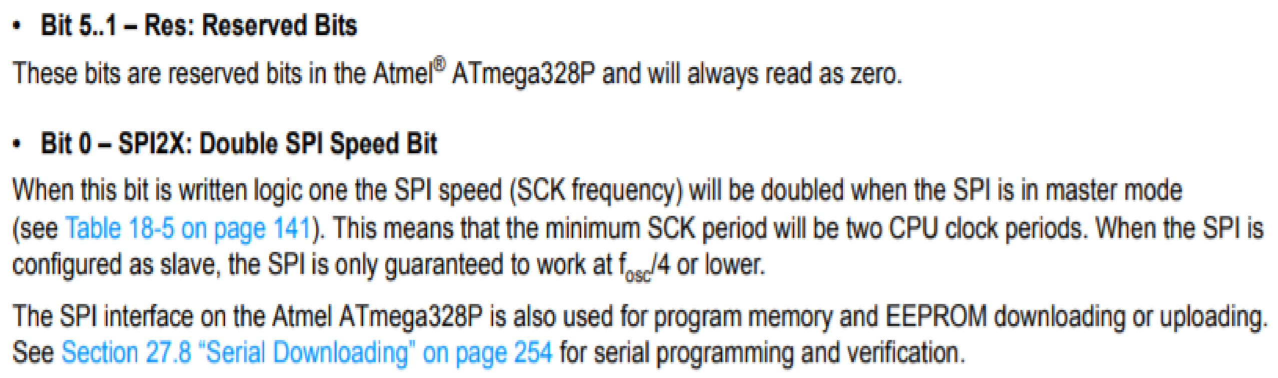
(2) ATmega328의 SPI 모듈



SPI – Serial Peripheral Interface

Features

- Full-duplex, three-wire synchronous data transfer
- Master or slave operation
- LSB first or MSB first data transfer
- Seven programmable bit rates
- End of transmission interrupt flag
- Write collision flag protection
- Wake-up from idle mode
- Double speed (CK/2) master SPI mode



- **Bit 5..1 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATmega328P and will always read as zero.

- **Bit 0 – SPI2X: Double SPI Speed Bit**

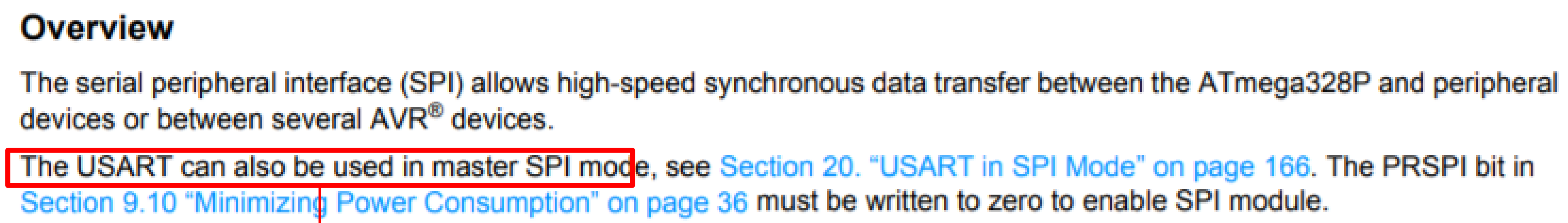
When this bit is written logic one the SPI speed (SCK frequency) will be doubled when the SPI is in master mode (see Table 18-5 on page 141). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as slave, the SPI is only guaranteed to work at $f_{osc}/4$ or lower.

The SPI interface on the Atmel ATmega328P is also used for program memory and EEPROM downloading or uploading. See Section 27.8 "Serial Downloading" on page 254 for serial programming and verification.

Master모드시 설정된 분주비에 의해 정해진 SCK의 주파수를 두배로 설정해준다

3. SPI 통신

(2) ATmega328의 SPI 모듈



Overview

The serial peripheral interface (SPI) allows high-speed synchronous data transfer between the ATmega328P and peripheral devices or between several AVR® devices.

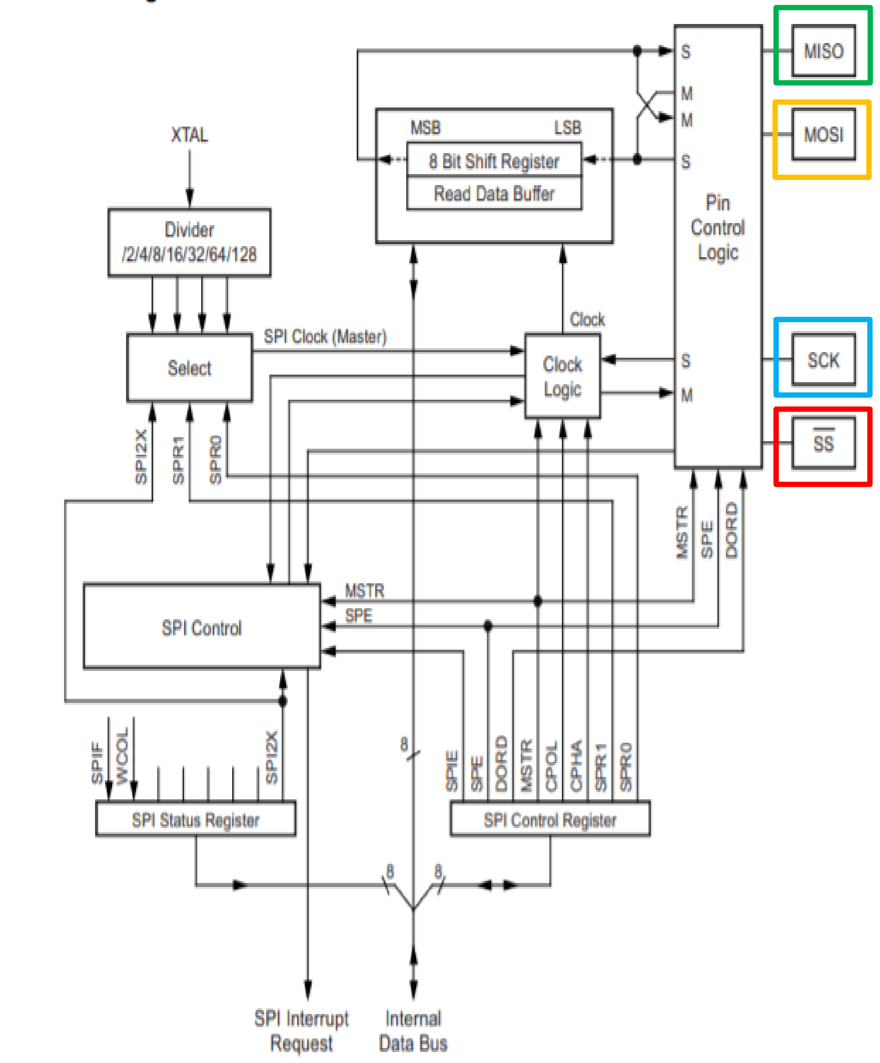
The USART can also be used in master SPI mode, see Section 20. "USART in SPI Mode" on page 166. The PRSPI bit in Section 9.10 "Minimizing Power Consumption" on page 36 must be written to zero to enable SPI module.

SPI의 Master모드를 통해 범용 동기 직렬통신을 사용할 수 있다.
(USART도 동기 직렬이고 SPI도 동기 직렬통신이니까..)
SCK, MOSI, SIMO핀을 CLK, TX,RX로 연결하면 가능..

3. SPI 통신

(2) ATmega328의 SPI 모듈
- 블록도

SPI Block Diagram[1]



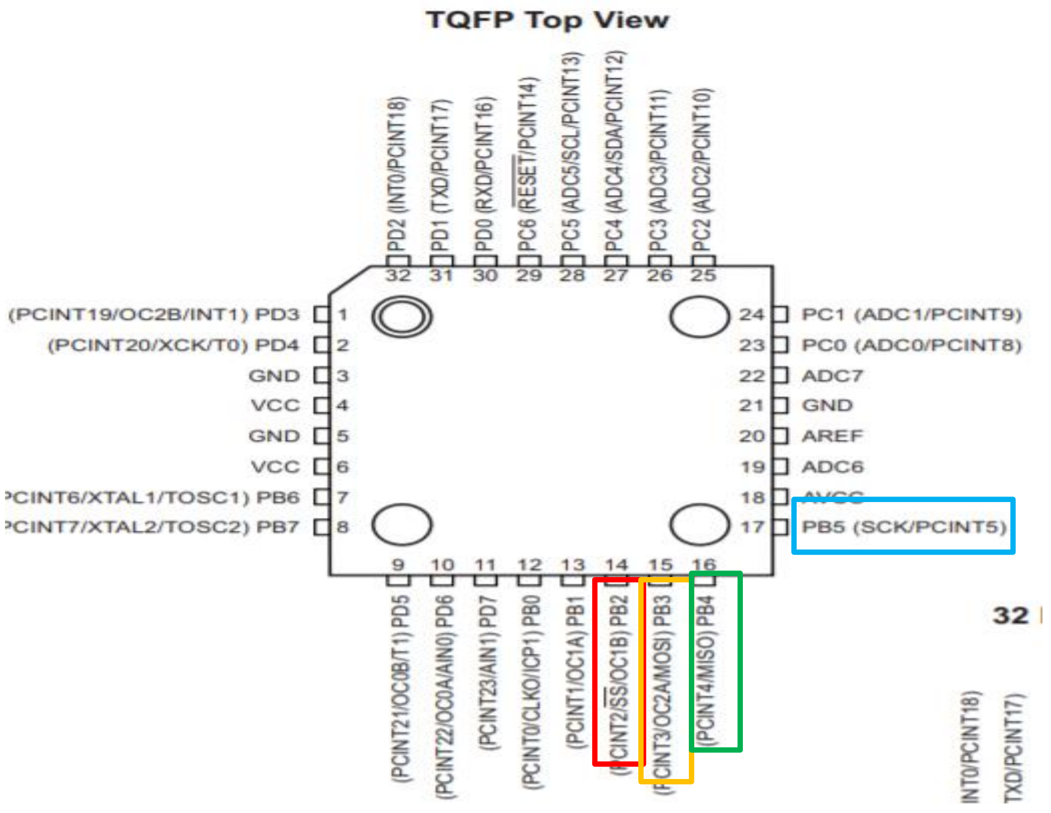


Table 18-1. SPI Pin Overrides[1]

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User defined	Input
MISO	Input	User defined
SCK	User defined	Input
$\overline{SS}$	User defined	Input

Note: 1. See Section 13.3.1 "Alternate Functions of Port B" on page 65 for a detailed description of how to define the

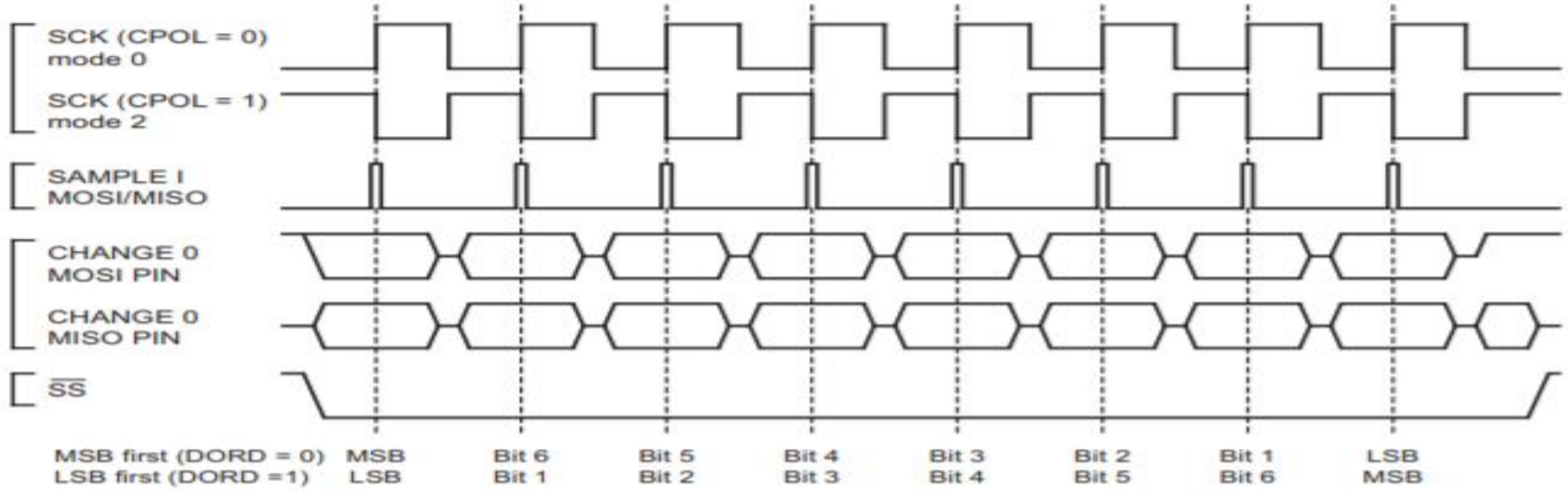
3. SPI 통신

(2) ATmega328의 SPI 모듈
- 데이터 송/수신 동작
- CPHA(CLK PHASE = 0)

Table 18-2. SPI Modes

SPI Mode	Conditions	Leading Edge	Trailing eDge
0	CPOL=0, CPHA=0	Sample (rising)	Setup (falling)
1	CPOL=0, CPHA=1	Setup (rising)	Sample (falling)
2	CPOL=1, CPHA=0	Sample (falling)	Setup (rising)
3	CPOL=1, CPHA=1	Setup (falling)	Sample (rising)

Figure 18-3. SPI Transfer Format with CPHA=0



SCK (CPOL = 0) mode 0
SCK (CPOL = 1) mode 2
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD =1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

- 위에서 보이는것처럼 CPOL 설정에 따라 SCK "H", "L"에 따라 데이터 송/수신이 이뤄진다.
- CPHA = 0는 CPOL = 0일 때 SCK "H"에서 데이터 송/수신, CPOL = 1일 때 "L"에서 데이터 송수신이 된다.

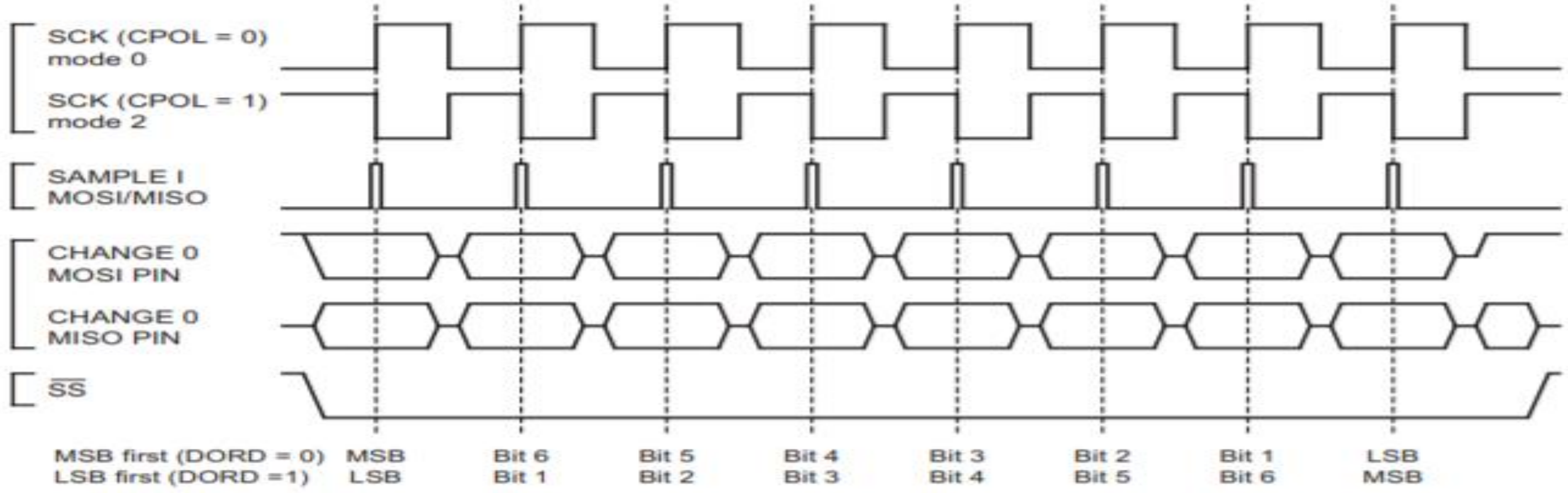
3. SPI 통신

(2) ATmega328의 SPI 모듈
- 데이터 송/수신 동작
- CPHA(CLK PHASE = 1)

Table 18-2. SPI Modes

SPI Mode	Conditions	Leading Edge	Trailing eDge
0	CPOL=0, CPHA=0	Sample (rising)	Setup (falling)
1	CPOL=0, CPHA=1	Setup (rising)	Sample (falling)
2	CPOL=1, CPHA=0	Sample (falling)	Setup (rising)
3	CPOL=1, CPHA=1	Setup (falling)	Sample (rising)

Figure 18-3. SPI Transfer Format with CPHA=0

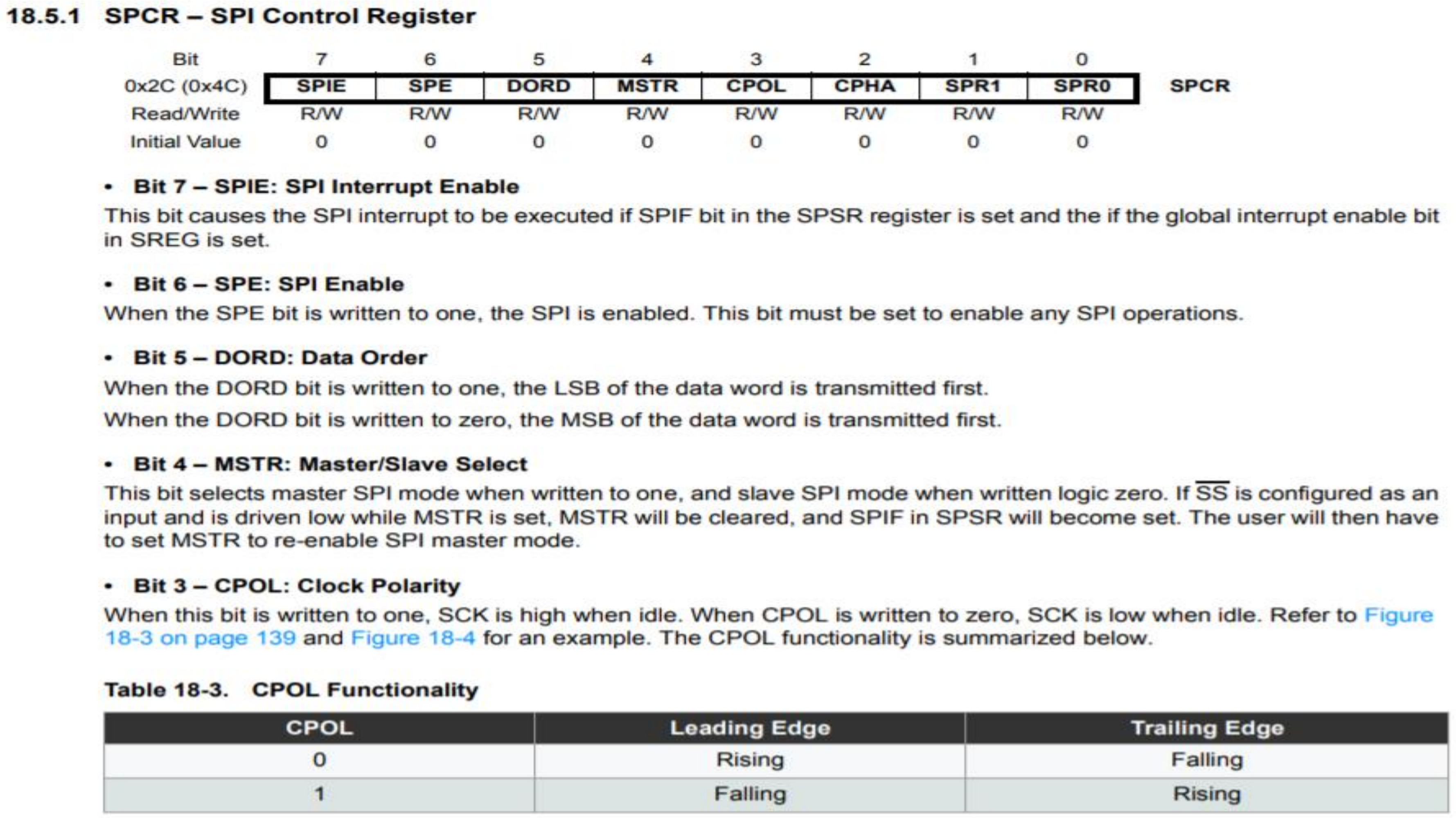


SCK (CPOL = 0) mode 0
SCK (CPOL = 1) mode 2
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD =1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

- 위에서 보이는것처럼 CPOL 설정에 따라 SCK "H", "L"에 따라 데이터 송/수신이 이뤄진다.
- CPHA = 0는 CPOL = 0일 때 SCK "H"에서 데이터 송/수신, CPOL = 1일 때 "L"에서 데이터 송수신이 된다.(CPHA = 1이면 반대로 동작..)

3. SPI 통신

(2) ATmega328의 SPI 모듈
- 관련 레지스터 : Control Register
- SCK주파수, 인터럽트, Master/Slave모드 설정, SCK Phase, Pole설정이 가능하다..



18.5.1 SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
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	

- **Bit 7 – SPIE: SPI Interrupt Enable**

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR register is set and the if the global interrupt enable bit in SREG is set.

- **Bit 6 – SPE: SPI Enable**

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

- **Bit 5 – DORD: Data Order**

When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

- **Bit 4 – MSTR: Master/Slave Select**

This bit selects master SPI mode when written to one, and slave SPI mode when written logic zero. If $\overline{\text{SS}}$ is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI master mode.

- **Bit 3 – CPOL: Clock Polarity**

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 18-3 on page 139 and Figure 18-4 for an example. The CPOL functionality is summarized below.

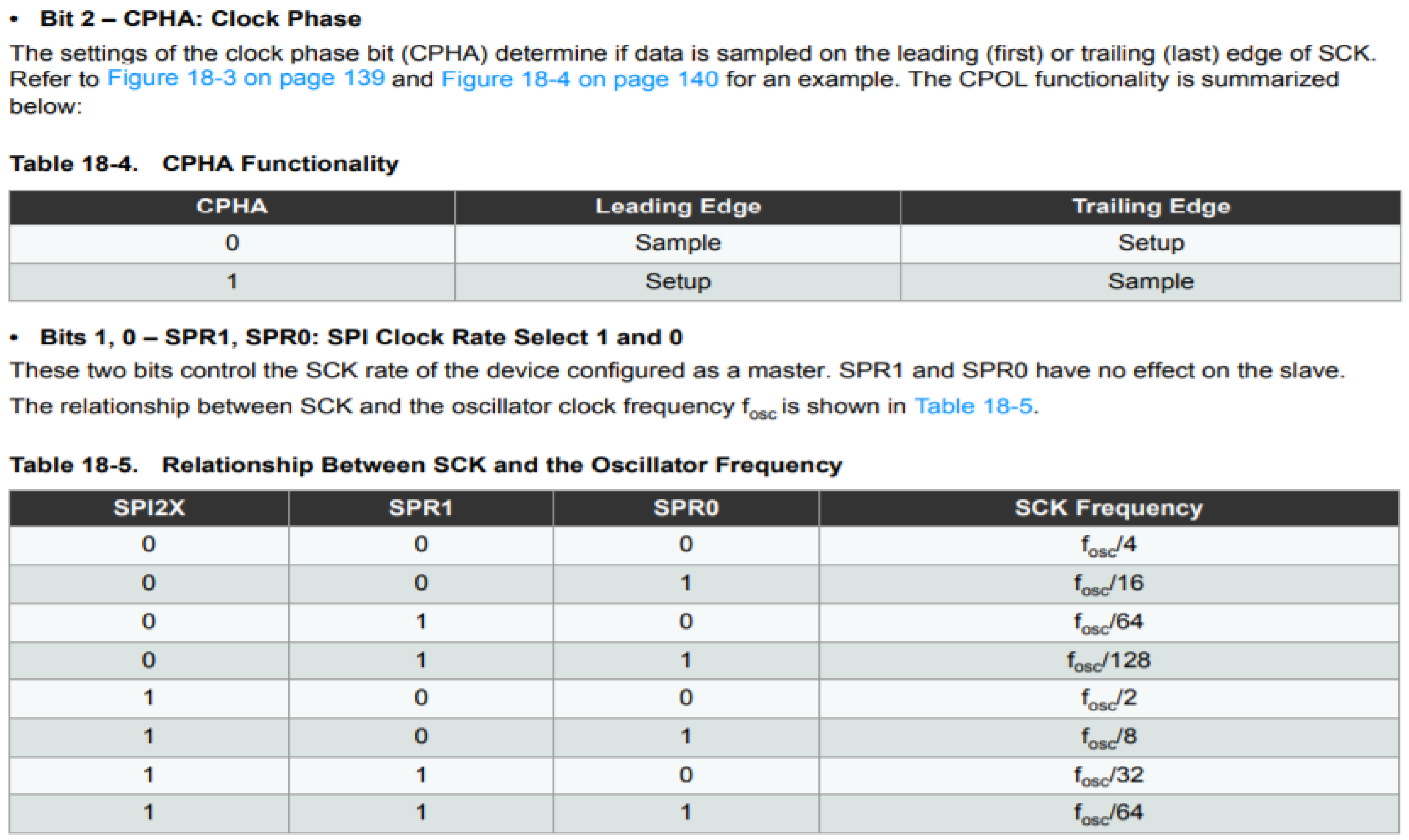
Table 18-3. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

포기하면 얻는 건 아무것도 없다.

3. SPI 통신

(2) ATmega328의 SPI 모듈
- 관련 레지스터 : Control Register



- **Bit 2 – CPHA: Clock Phase**

The settings of the clock phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 18-3 on page 139 and Figure 18-4 on page 140 for an example. The CPOL functionality is summarized below:

Table 18-4. CPHA Functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

- **Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a master. SPR1 and SPR0 have no effect on the slave.

The relationship between SCK and the oscillator clock frequency f_{osc} is shown in Table 18-5.

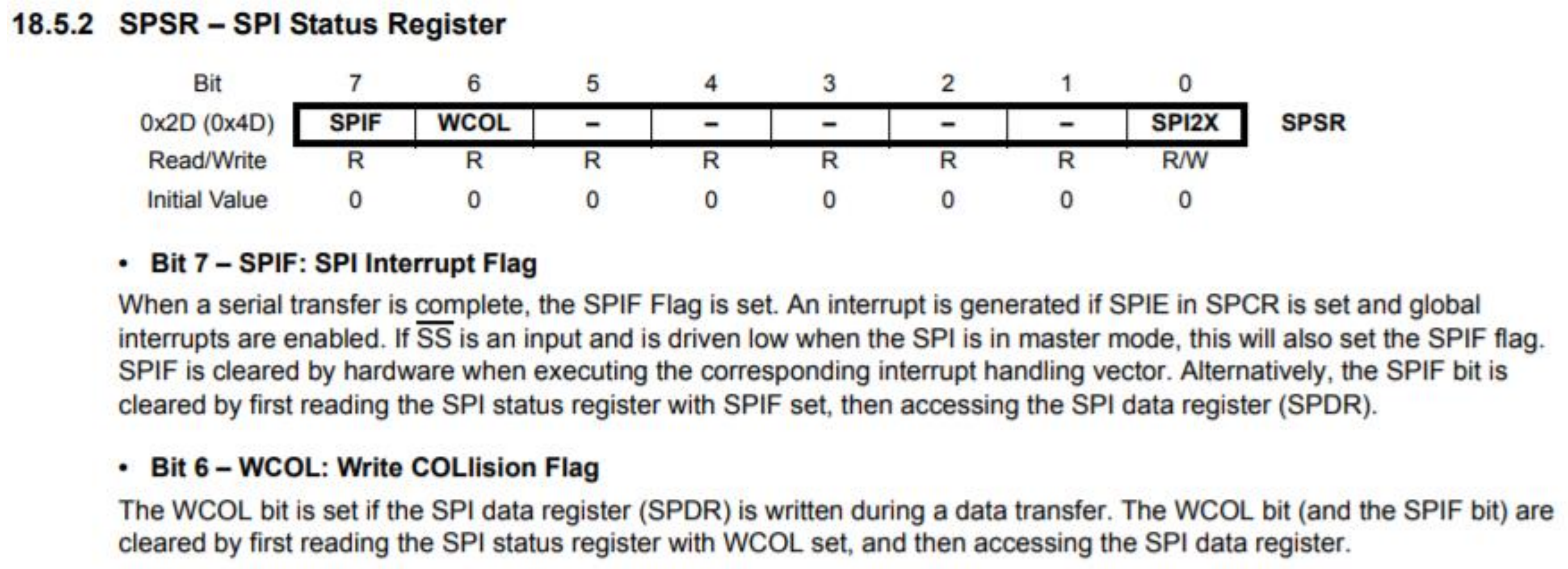
Table 18-5. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

3. SPI 통신

(2) ATmega328의 SPI 모듈

- 관련 레지스터 : Status Register



18.5.2 SPSR – SPI Status Register

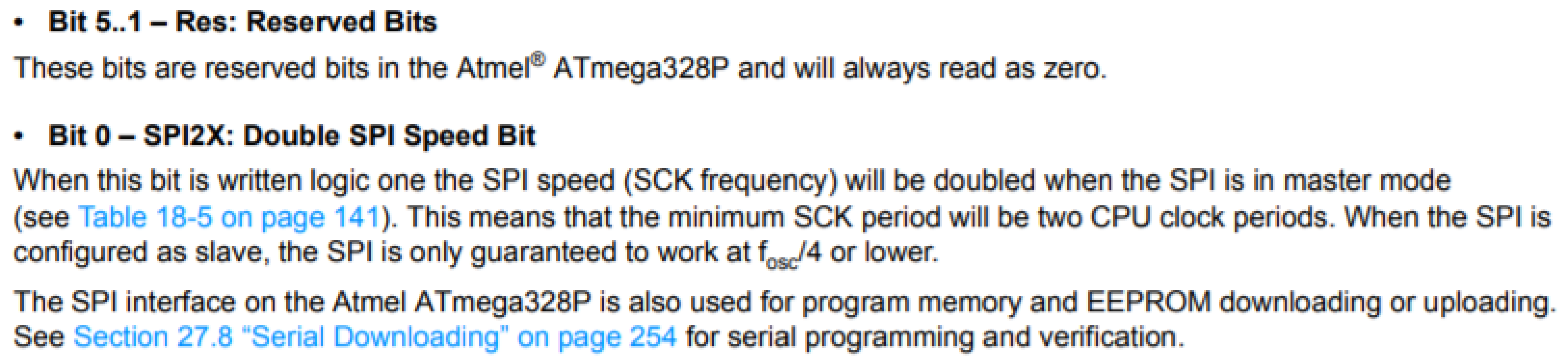
Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF: SPI Interrupt Flag**

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in master mode, this will also set the SPIF flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI status register with SPIF set, then accessing the SPI data register (SPDR).

- **Bit 6 – WCOL: Write COLlision Flag**

The WCOL bit is set if the SPI data register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI status register with WCOL set, and then accessing the SPI data register.



- **Bit 5..1 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATmega328P and will always read as zero.

- **Bit 0 – SPI2X: Double SPI Speed Bit**

When this bit is written logic one the SPI speed (SCK frequency) will be doubled when the SPI is in master mode (see Table 18-5 on page 141). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as slave, the SPI is only guaranteed to work at $f_{osc}/4$ or lower.

The SPI interface on the Atmel ATmega328P is also used for program memory and EEPROM downloading or uploading. See Section 27.8 "Serial Downloading" on page 254 for serial programming and verification.