AVR - HW7

임베디드스쿨1기

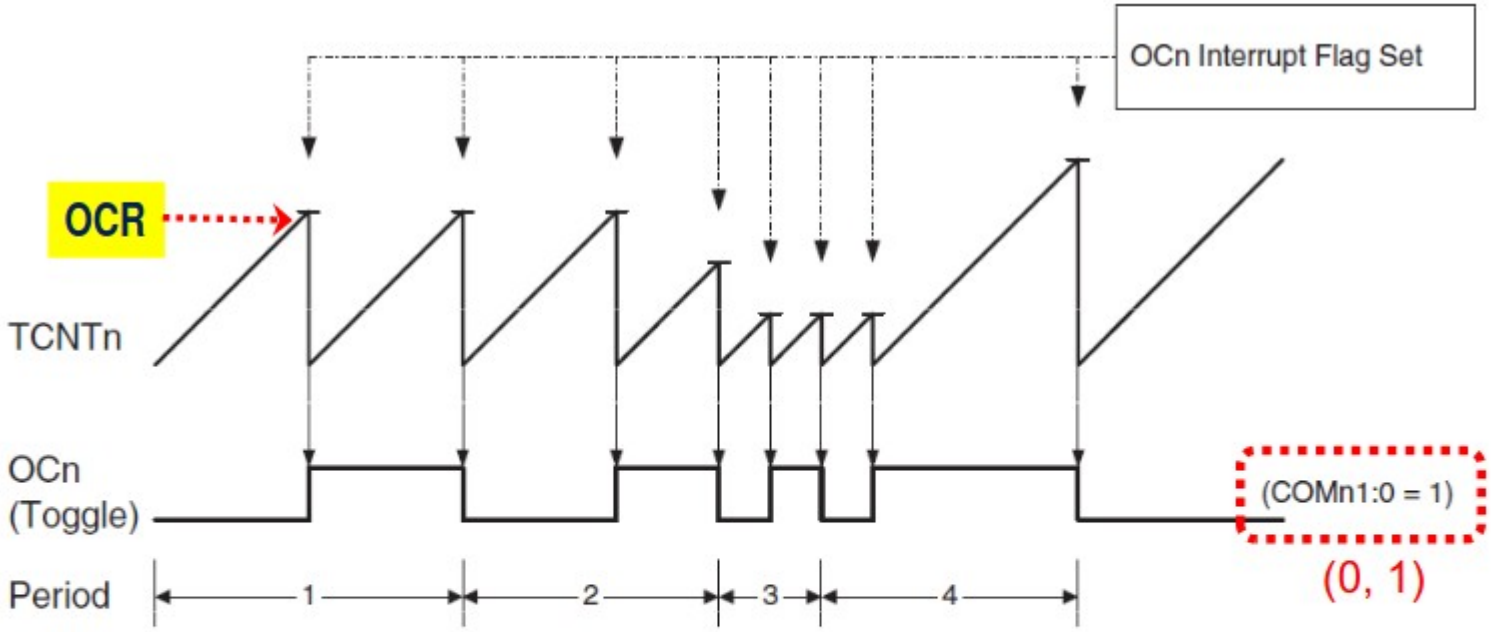
Lv1과정

2020. 10. 30

박하늘

1. [Review] T/C 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

1. [Review] T/C CTC Mode waveform Generation

```c
#define F_CPU 16000000UL

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

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

volatile unsigned int counter = 0;
void CTC_timer_Init(void);

SIGNAL(TIMER0_COMPA_vect)
{
}
int main(void)
{
    CTC_timer_Init();
    while(1)
    {
    }
}

void CTC_timer_Init(void)
{
    cbi(SREG, 7);

    sbi(TCCR0A, WGM01);
    sbi(TCCR0A, COM0A0);
    TCCR0B = (1 << CS02) |(1 <<CS00);
    TIMSK0 = (1 << OCIE0A);

    OCR0A = 255;
    DDRD = 0xff;
    PORTD =0xff;
    sbi(SREG,7);
}
```

1. sbi(TCCR0A, WGM01); //TOP: OCRA

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	

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

2. sbi(TCCR0A, COM0A0); //OCRA와 비교하여 토글

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

3. TCCR0B = (1 << CS02)|(1 << CS00) // 1024로 분주
16MHz/ 1024 = 15624 hz
1/15624 = 0.000064 sec
0.000064 * 256 = 0.0164 sec
Duty: 16.4 msec, 주기: 32.8msec

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] T/C CTC Mode waveform Generation

```c
#define F_CPU 16000000UL

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

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

volatile unsigned int counter = 0;
void CTC_timer_Init(void);

SIGNAL(TIMER0_COMPA_vect)
{
}
int main(void)
{
    CTC_timer_Init();
    while(1)
    {
    }
}

void CTC_timer_Init(void)
{
    cbi(SREG, 7);

    sbi(TCCR0A, WGM01);
    sbi(TCCR0A, COM0A0);
    TCCR0B = (1 << CS02) |(1 <<CS00);
    TIMSK0 = (1 << OCIE0A);

    OCR0A = 255;
    DDRD = 0xff;
    PORTD =0xff;
    sbi(SREG,7);
}
```

4. TIMSK0 = (1<<OCIE0A); //CIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable

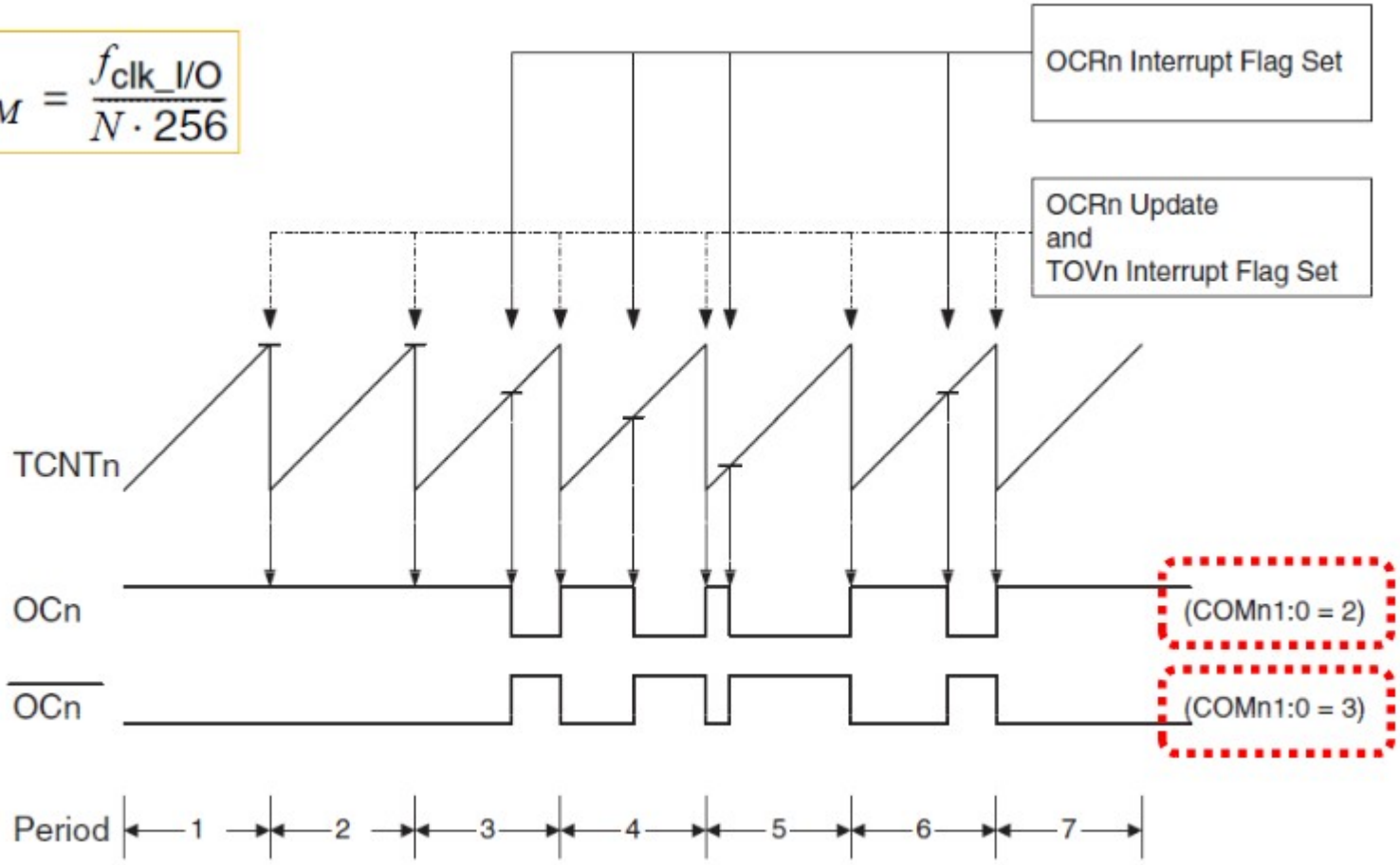
5. OCR0A = 255; // 255로 Top값을 정한다.
DDRD = 0xff; //모두 출력으로 사용하겠다(LED사용)
PORTD = 0xff; //5V 출력 사용

6. SIGNAL(TIMER0_COMPA_vect) //main문에 함수를 집어넣지 않아도 HW차원에서 함수를 호출한다.
즉, 타이머 오버플로우 되었을때 인터럽트 실행

1. [Review] T/C 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

1. [Review] T/C Fast PWM Mode

```c
#define F_CPU 16000000UL

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

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

volatile unsigned int counter = 0;
void Fast_PWM_Mode(void);

SIGNAL(TIMER0_COMPA_vect)
{
    OCR0A += 1;

    if(OCR0A == 255)
    {
        OCR0A = 0;
    }
}
int main(void)
{
    Fast_PWM_Mode();
    while(1)
    {
    }
}
void Fast_PWM_Mode(void)
{
    cbi(SREG, 7);

    sbi(TCCR0A, WGM00);
    sbi(TCCR0A, WGM01);
    sbi(TCCR0A, COM0A1);
    TCCR0B = (1 << CS02) |(1 <<CS00);
    TIMSK0 = (1 << OCIE0A);

    OCR0A = 0;
    DDRD = 0xff;
    PORTD =0x00;
    sbi(SREG,7);
}
```

1. sbi(TTCR0A, WGM00)
sbi(TTCR0A, WGM01); // Fast PWM Mode

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

2. sbi(TCCR0A, COM0A1); //OC0A의 BOTTOM일때 1, OC0A와 매치되면 0

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal port operation, OC0A disconnected. WGM02 = 1: Toggle OC0A on compare match.
1	0	Clear OC0A on compare match, set OC0A at BOTTOM, (non-inverting mode).
1	1	Set OC0A on compare match, clear OC0A at BOTTOM, (inverting mode).

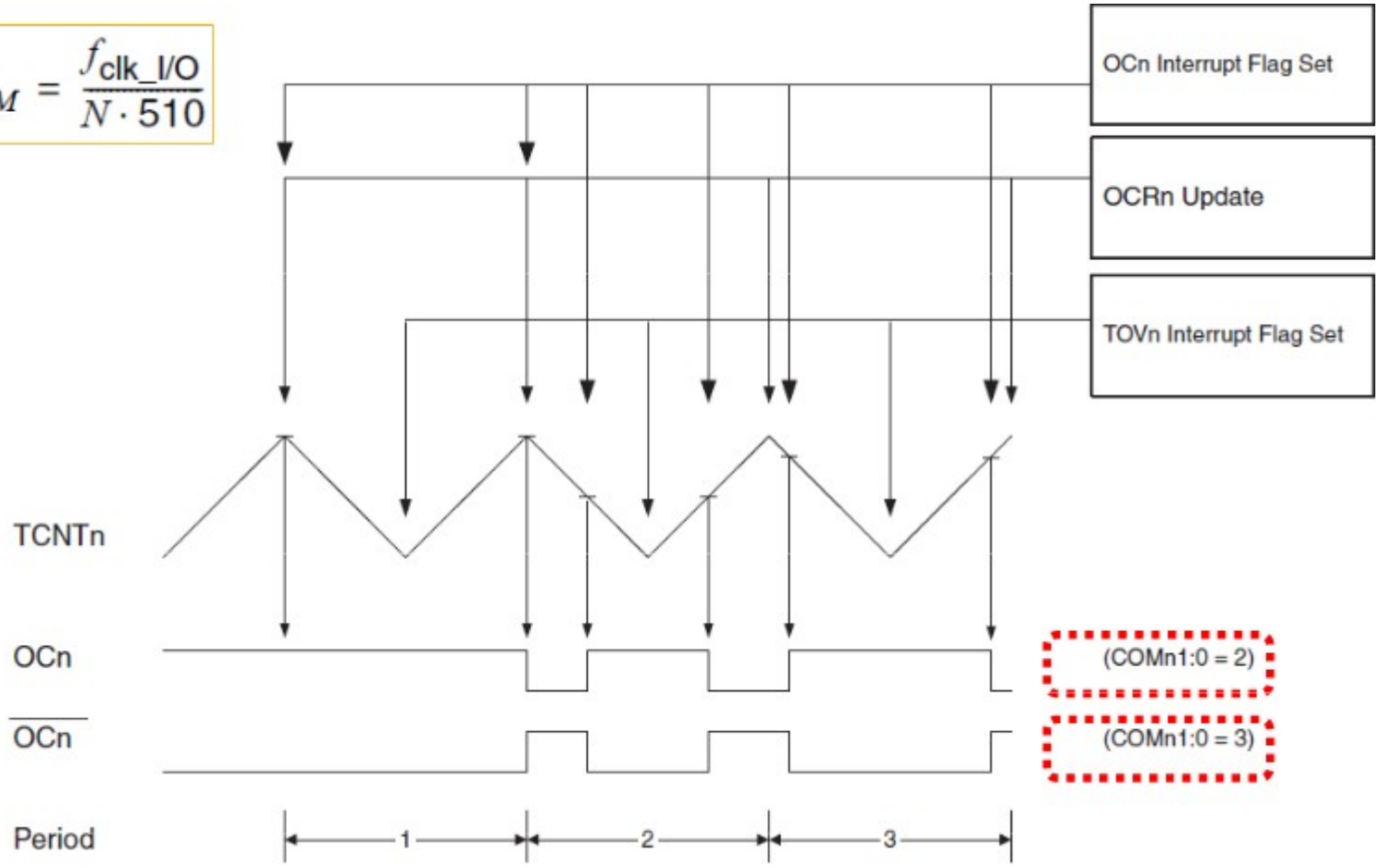
3.
4. DDRD = 0xff; //DDRD를 5V출력으로 할 것이다.
   PORTD = 0x00; //없어도 됨
5. HW 인터럽트 발생시, OCRA ++(counter)하여, 255 넘으면 OCRA 0 (timer)으로 됨

1. [Review] T/C Phase Correct PWM Mode

- PC PWM Mode

  : Dual Slope Operation이라서 Fast PWM 보다 동작 주파수가 작다.
  하지만, 대칭적인 특징이 있어서 모터 제어 응용에 많이 사용된다.

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



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

1. [Review] T/C Phase Correct PWM Mode

```c
#define F_CPU 16000000UL

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

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

volatile unsigned int counter = 0;
void Phase_Correct_PWM_Init(void);

SIGNAL(TIMER0_COMPA_vect)
{
    OCR0A += 1;
    if(OCR0A == 255)
    {
        OCR0A = 0;
    }
}
int main(void)
{
    Phase_Correct_PWM_Init();
    while(1)
    {
    }
}
void Phase_Correct_PWM_Init(void)
{
    cbi(SREG, 7);

    sbi(TCCR0A, WGM02);
    sbi(TCCR0A, WGM00);
    sbi(TCCR0A, COM0A1);
    TCCR0B = (1 << CS02) |(1 <<CS00);
    TIMSK0 = (1 << OCIE0A);

    OCR0A = 0;
    DDRD = 0xff;
    //PORTD =0x00;
    sbi(SREG,7);
}
```

1. sbi(TTCR0A, WGM02)
sbi(TTCR0A, WGM00); // Phase Correct PWM Mode

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

2. sbi(TCCR0A, COM0A1); //OC0A와 비교할때 upcounting시 0, downcounting시 1

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal port operation, OC0A disconnected. WGM02 = 1: Toggle OC0A on compare match.
1	0	Clear OC0A on compare match when up-counting. Set OC0A on compare match when down-counting.
1	1	Set OC0A on compare match when up-counting. Clear OC0A on compare match when down-counting.

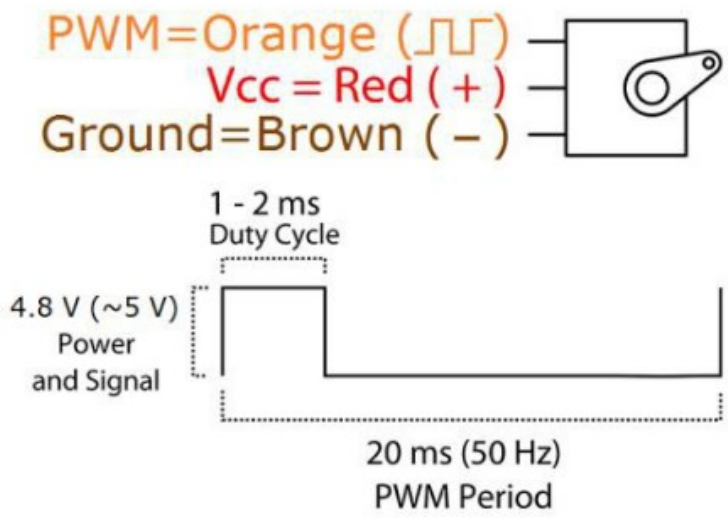
4. DDRD = 0xff; //DDRD를 5V출력으로 할 것이다.

5. HW 인터럽트 발생시, OCRA ++(counter)하여, 255 넘으면 OCRA 0 (timer)으로 됨

1. [Review] T/C CTC Mode waveform Generation

```c
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/delay.h>
#include <avr/interrupt.h>

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

volatile unsigned int counter = 0;
void SG_Fast_PWM_MODE(void);

int main(void)
{
    SG_Fast_PWM_MODE();
    while(1)
    {
        OCR1A = 2000;
        _delay_ms(1000);
        OCR1A = 3000;
        _delay_ms(1000);
        OCR1A = 4000;
        _delay_ms(1000);
        OCR1A = 3000;
        _delay_ms(1000);
    }
}
void SG_Fast_PWM_MODE(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 << PORTB1);

    sbi(SREG,7);
}
```

1. Servo Motor Datasheet



2.

IC

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

1. [Review] 16bit T/C PWM Servo Motor

```c
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/delay.h>
#include <avr/interrupt.h>

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

volatile unsigned int counter = 0;
void SG_Fast_PWM_MODE(void);

int main(void)
{
    SG_Fast_PWM_MODE();
    while(1)
    {
        OCR1A = 2000;
        _delay_ms(1000);
        OCR1A = 3000;
        _delay_ms(1000);
        OCR1A = 4000;
        _delay_ms(1000);
        OCR1A = 3000;
        _delay_ms(1000);
    }
}
void SG_Fast_PWM_MODE(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 << PORTB1);

    sbi(SREG,7);
}
```

3. sbi(TCCR1A, COM1A1);
sbi(TCCR1A, COM1B1); // non-inverting mode 사용

Table 15-3. Compare Output Mode, Fast PWM(1)

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 14 or 15: Toggle OC1A on compare match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at BOTTOM (inverting mode)

4. sbi(TCCR1B, CS11); //16Mhz/8 = 2Mhz

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)

6. ICR1로 TCTNT0의 끝을 조절하고,
OCR로 제어하면 duty비 제어 가능

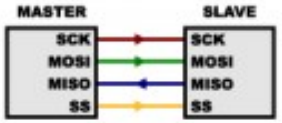
2. [Preview] SPI – Serial Peripheral Interrupt

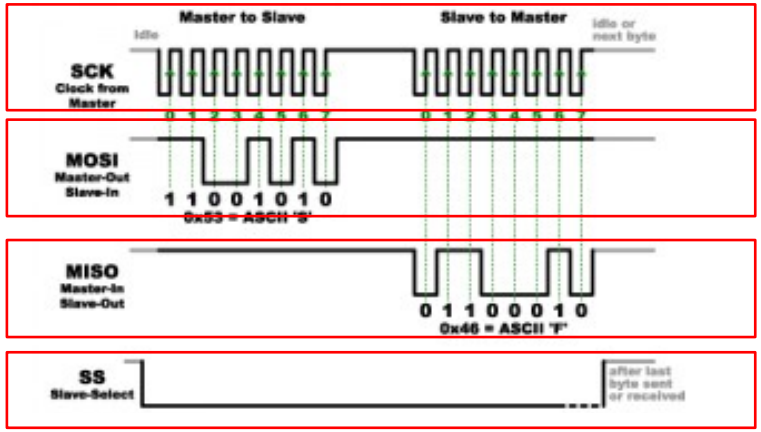
1) SPI란?

: MOSI, MISO, SCK, /SS의 4개 통신선을 이용하는 (전이중) 고속 동기식 직렬통신 방식이다.
이는 UART 통신 규격에 비하여 빠른 속도와 멀티 통신이 지원되며, I2C (반이중) 통신 규격에 비하여 빠르며 간단한 제어가 가능하, 1:n의 통신도 가능하다는 장점이 있다.

2) SPI 특징?

- Full-duplex, three-wire synchronous data transfer
- Master or slave operation → 항상 Master와 Slave사이에서 직렬로 데이터를 송수신, 클럭은 항상 Master가 발생
- 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 → Wake-Up : 슬립모드 해제 기능
- Double speed (CK/2) master SPI mode

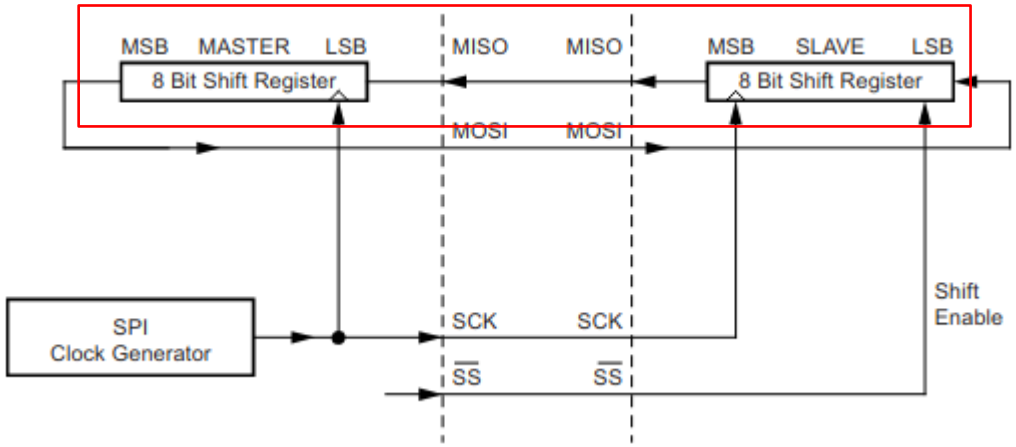




MOSI(Master Out Slave IN): 마스터 → 슬레이브 데이터 전송선
MISO(Master In Slave Out): 슬레이브 → 마스터 데이터 전송선
(SPI 송신이 곧 수신임. → 송수신의 타이밍을 결정짓는 것은 Master장치)
-SCLK, CLK(Clock): 동기화 신호를 위한 Clock
-SS, CS(Slave Select, Chip Select): Master가 여러개의 Slave중 어디로 보낼지 결정하는 신호선

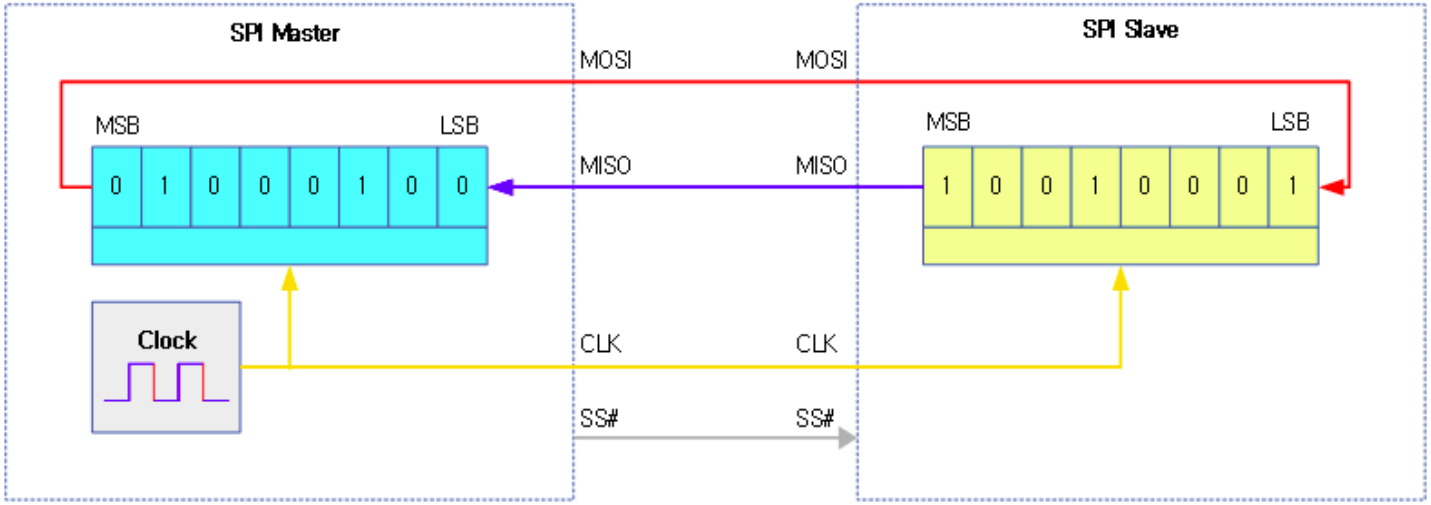
2. [Preview] SPI – Serial Peripheral Interrupt

3) SPI Pin Description?



Shift Register에 의해서 데이터가 이동.
마스터가 어떤값을 슬레이브로 보낼때, 1 클럭의 신호마다 1비트의 데이터가 이동함.

- 기본적으로 MSB부터 전송되는데 특정 컨트롤러는 LSB부터 전송을 수행시키는 방법도 지원한다.



감사합니다.