

11 장 : 모터 제어를 통한 미니 선풍기 만들기

ATmega128 마이크로컨트롤러를 이용한 임베디드시스템 구현



JCnet
제이씨넷

신 상 석

목차

1. 모터(Motor)
2. ATmega128의 PWM
3. JKIT-128-1에서의 DC 모터 연결 설계
4. 실습 MR-1 : 모터 움직이기
5. 실습 MR-2 : 모터 속도 조절하기
6. 실습 MR-3 : 모터 제어를 통한 미니선풍기 만들기

모터(Motor)

□ 모터 (Motor)

- 전류가 흐르는 도체가 자기장 속에서 받는 힘을 이용하여 전기에너지를 역학적에너지로 바꾸는 장치. 일명 전동기.

□ 공급전원에 따른 모터의 종류

- AC 모터
- DC 모터



모터(Motor)

□ 기능에 따른 모터의 종류

- 서보(Servo) 모터 : 기어를 내부에 장착한 기어드 모터 중 하나로 로봇이나 기계장치 등 단거리의 힘있는 동작을 위하여 만들어진 모터
- 스텝핑(Stepping)모터 : 회전수와 정지 위치 등을 정확하게 할 수 있게 한 모터
- 스펀들(Spindle) 모터 : 고속의 회전이 필요한 드릴, 하드디스크 등에 사용되는 모터



모터(Motor)

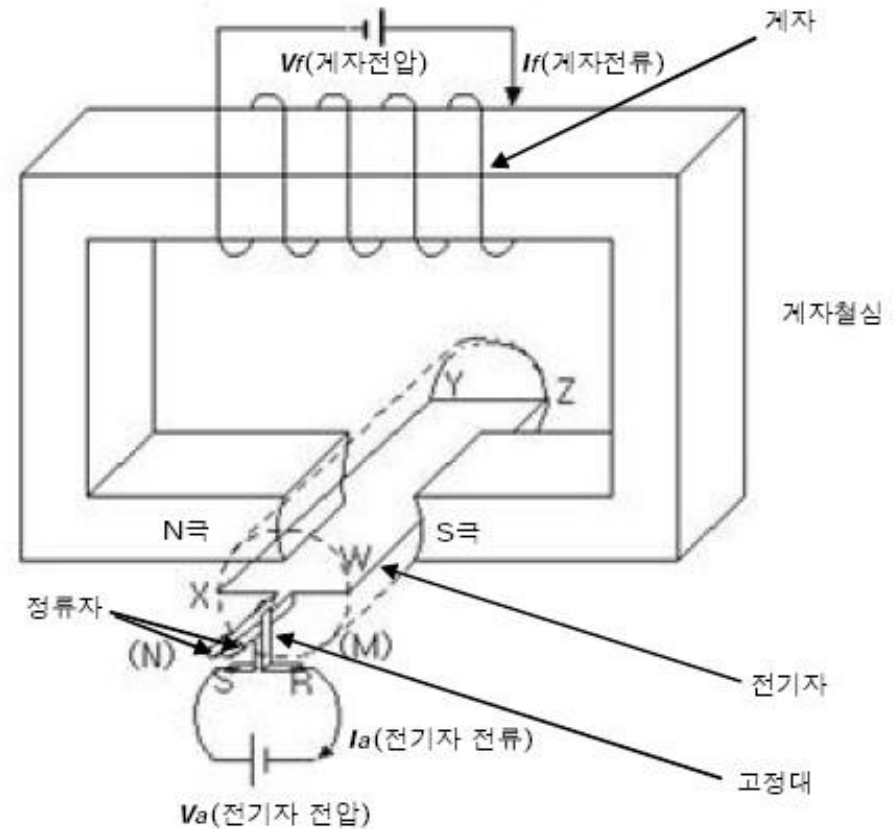
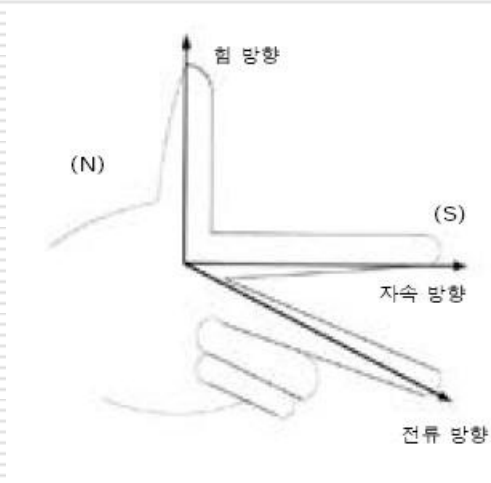
□ DC 모터

- DC 전원을 사용하며 영구자석의 자기장과 그 자기장 속에 놓여 있는 도체에 흐르는 전류에 의해 발생한 전자력 간의 반발력으로 회전하는 모터
- 회전체로 흐르는 DC 전원의 방향이 회전에 의하여 끊어졌다가 반대방향으로 흐르도록 기계식 브러시(brush)로 설계되어 있는데, 이로 인하여 접점에서의 전기 불꽃, 회전 소음, 수명 단축의 현상이 나타남
- 기동 토크가 큼
- 인가 전압에 회전 특성이 선형적으로 비례
- 입력 전류에 출력 토크가 선형적으로 비례
- 출력 효율이 양호하며 가격이 저렴

% BLDC(BrushLess DC) 모터 : 반도체를 이용하여 DC를 AC 변환하여 사용하므로써 브러시를 사용하지 않아도 되는 DC 모터

모터(Motor)

- DC 모터의 원리
 - 플레밍의 왼손 법칙



모터(Motor)

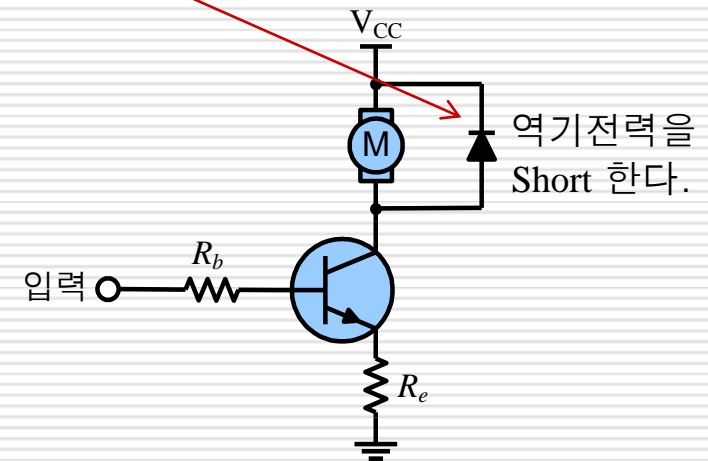
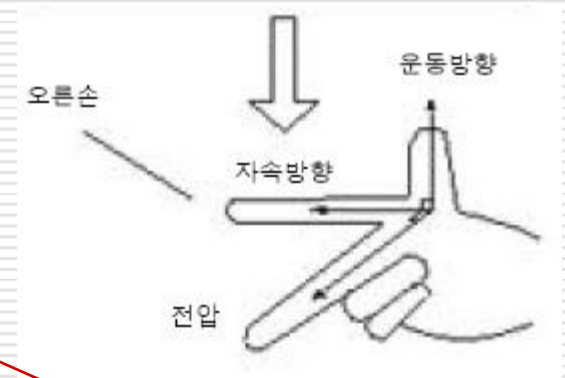
□ 트랜지스터를 이용한 모터 제어

- 모터는 부하가 크고, 대용량 전류가 흐르므로, DC모터인 경우에도 일반 GPIO 신호를 이용하여 직접 제어할 수는 없음
- 작은 용량의 모터는 트랜지스터를 이용하여 구동하고, 큰 용량의 모터는 파워트랜지스터나, 릴레이 등을 이용하여 구동
- 트랜지스터를 사용하는 경우는 트랜지스터의 Collector 쪽에 모터를 접속하는 콜렉터 방식이 완전히 포화된 on 상태로 구동할 수 있어 드라이브 성능이 좋고, 전압 손실도 줄일 수 있어 일반적으로 많이 사용됨

모터(Motor)

□ 트랜지스터를 이용한 모터 제어

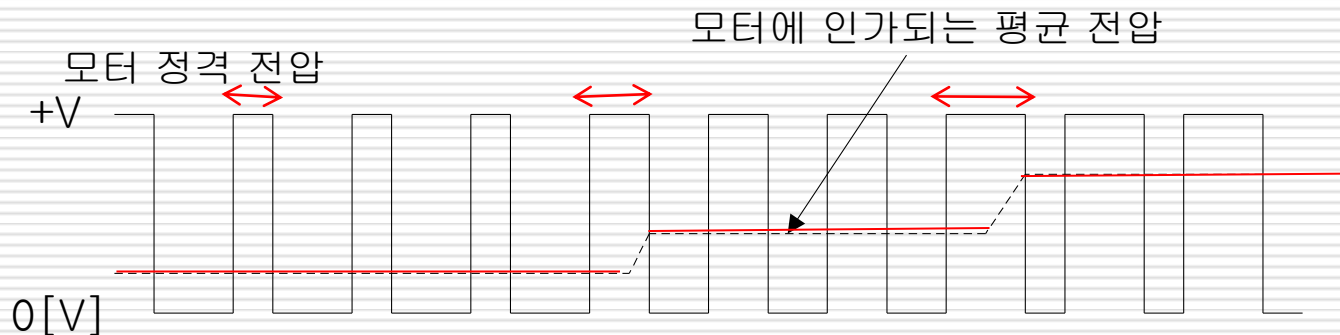
- 플레밍의 오른손 법칙에 따른 역기전력에 대한 보호 필요하므로 **관성 다이오드(Flywheel Diode)**를 **역방향으로 장착**
- 관성 다이오드(flywheel diode)는 역방향의 기전력만 단락시키고, 통상적인 전압에 대해서는 높은 저항이 되어 전류가 흐르지 않도록 하는 역할 수행(없으면 트랜지스터가 손상될 우려가 있음)



ATmega128의 PWM

□ PWM(Pulse Width Modulation)이란?

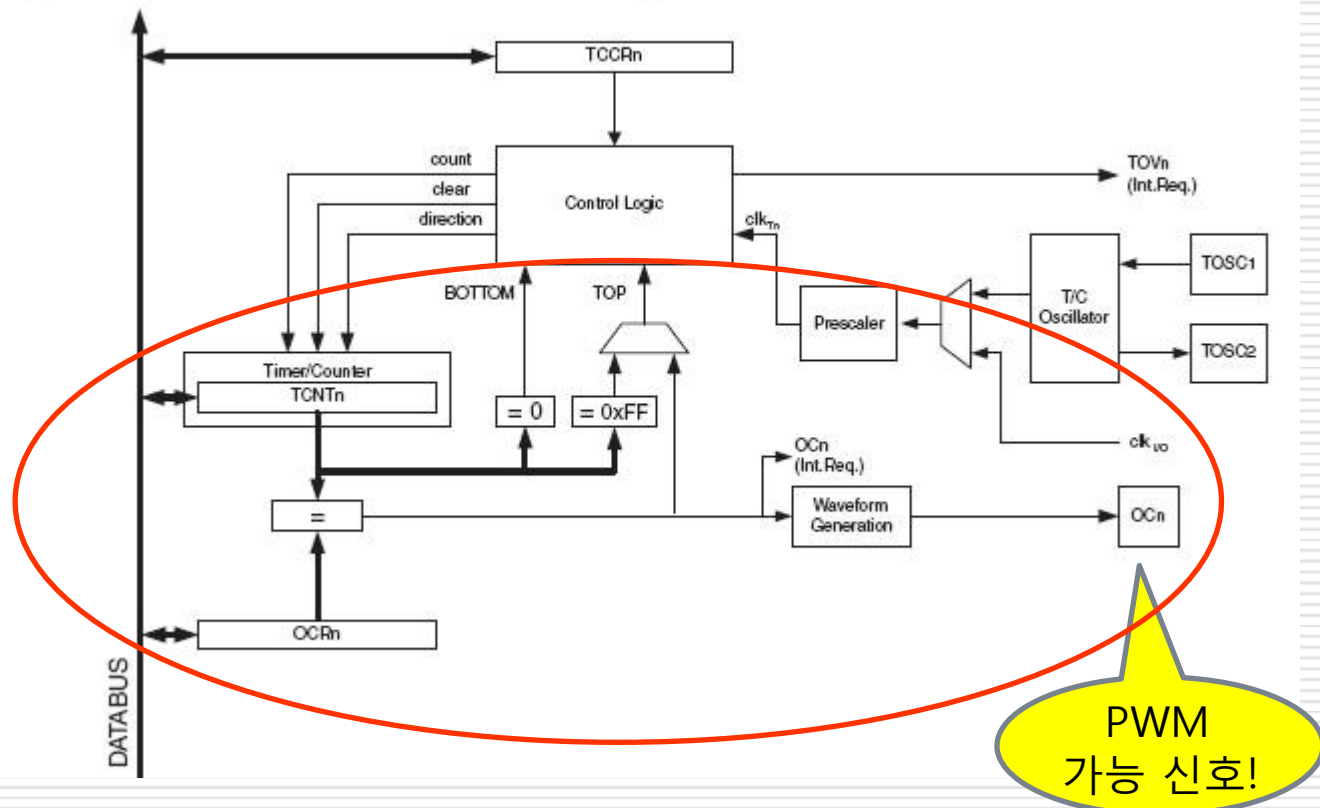
- 펄스(Pulse)는 짧은 시간동안 레벨이 변하는 신호를 의미
- 펄스폭 변조(PWM : Pulse Width Modulation)는 펄스의 'ON' 또는 'OFF'의 길이(시간)을 조절하는 방식을 의미
- 펄스폭을 조절하면 DC 모터의 경우 모터에 인가되는 평균 전압을 바꿀 수 있어 전압의 크기를 바꾸는 효과를 얻을 수 있으므로 모터 제어가 가능



ATmega128의 PWM

□ ATmega128 8비트 타이머/카운터 블록다이어그램

Figure 14-1. 8-bit Timer/Counter Block Diagram



ATmega128의 PWM

□ TCCR2(Timer/Counter Control Register 2)

- 타이머/카운터 제어 레지스터 2
- 동작 모드, 프리스케일러 등 타이머/카운터의 전반적인 동작 형태를 결정

7	6	5	4	3	2	1	0
FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20

0 1 1 1 1 0 1 1 = 0x7B

- 비트 7 : FOC2 (Force Output Compare) = 0 (PWM시 의미없음)
- 비트 3,6 : WGM (Waveform Generation Mode)=11 (Fast PWM)
- 비트 5,4 : COM (Compare Output Mode) = 11 (매치시 OC2=1)
- 비트 2, 1, 0 : CSn (Clock Select) = 011 (64분주)

ATmega128의 PWM

□ TCCR2(Timer/Counter Control Register 2)

- WGM (Waveform Generation Mode) : 동작 모드 결정

WGM21	WGM20	설명
0	0	Normal
0	1	Phase Correct PWM
1	0	CTC(Clear Timer on Compare Match Mode)
1	1	Fast PWM

- Fast PWM 모드 : TCNT2 의 값과 OCR2(Output Compare Register 2) 값을 비교하여 값이 같으면 인터럽트가 발생하고 또한 Overflow가 발생해도 인터럽트가 발생함. 또한 인터럽트 발생시 OC2 신호는 반전되므로 일정한 PWM 파형을 만들 수 있음

ATmega128의 PWM

- TCCR2(Timer/Counter Control Register 2)
 - COM(Compare Output Mode)
 - OC2핀의 동작을 조정
 - COM21/COM20에 따른 OC2 핀의 동작

(Fast-PWM 모드 시)

COM21	COM20	내용
0	0	Normal 포트 동작, OC2 연결하지 않음
0	1	Reserved
1	0	Clear OC2 on Compare Match, Set OCn2at TOP
1	1	Set OC2 on Compare Match, Clear OC2 at TOP

ATmega128의 PWM

□ TCCR2(Timer/Counter Control Register 2)

- CS22-20 (Clock Select) : 클럭 및 프리스케일러 선택

CS22	CS21	CS20	설명
0	0	0	클럭 입력 차단
0	0	1	No Prescaler, 1분주
0	1	0	8분주
0	1	1	64분주
1	0	0	256분주
1	0	1	1024분주
1	1	0	T2 외부클럭 하강에지
1	1	1	T2 외부클럭 상승에지

ATmega128의 PWM

□ TCNT2(Timer/Counter Register 2)

- 타이머/카운터 레지스터2
- 타이머/카운터2의 8비트 카운터 값을 저장하고 있는 레지스터 (Read/Write 가능)

7	6	5	4	3	2	1	0
TCNT7	TCNT6	TCNT5	TCNT4	TCNT3	TCNT2	TCNT1	TCNT0

ATmega128의 PWM

□ TIMSK(Timer Interrupt Mask)

- 타이머 인터럽트 마스크 레지스터
- 타이머/카운터0, 타이머/카운터1, 타이머/카운터2가 발생하는 인터럽트를 개별적으로 enable하는 레지스터

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

1 0 0 0 0 0 0 0 = 0x80

- OCIE2(Output Compare Interrupt Enable 2) : 타이머/카운터2의 출력비교 인터럽트 인에이블

ATmega128의 PWM

□ OCR2(Output Compare Register 2)

- 타이머/카운터 2 출력 비교 레지스터
- TCNT2 과 비교하기 위한 값을 저장하고 OC2 단자에 출력신호 발생

7	6	5	4	3	2	1	0
OCR7	OCR6	OCR5	OCR4	OCR3	OCR2	OCR1	OCR0

1 0 0 0 0 0 0 0 = 0x80
= 128
(저속)

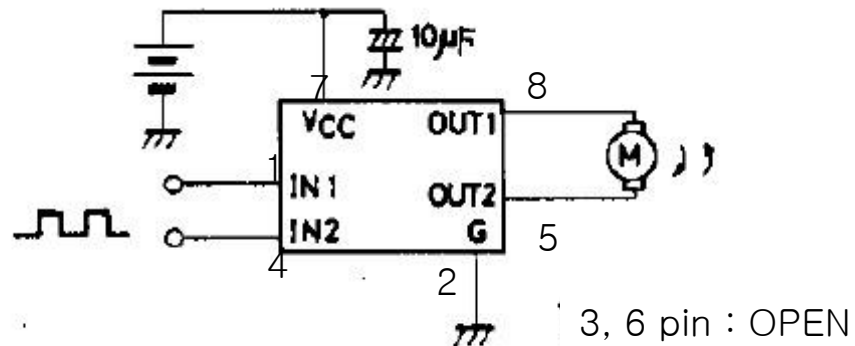
JKIT-128-1에서의 모터 연결 설계

- JKIT-128-1에서의 모터 연결 설계 개념
 - 제어하기 편리한 DC 모터를 사용
 - 모터는 전류 소모가 적고, 소형이고, 가격이 저렴한 3V용 DC모터 WRE-260A 사용
 - 모터 회전 방향을 제어하기 위하여는 2포트를 할당하여야 하며, PWM이 가능한 출력 포트이어야 하므로 PB(PB6, PB7)에 할당
 - 모터는 전류를 많이 소모하므로 트랜지스터 구동회로를 연결하여 사용하며, 과전류로부터 보호하기 위하여 모터 양단에는 Flyweel Diode도 역방향으로 삽입
 - 회로를 단순하게 하기 위하여 위 조건을 만족하는 모터구동 IC LB1630을 사용하여 구현

JKIT-128-1에서의 모터 연결 설계

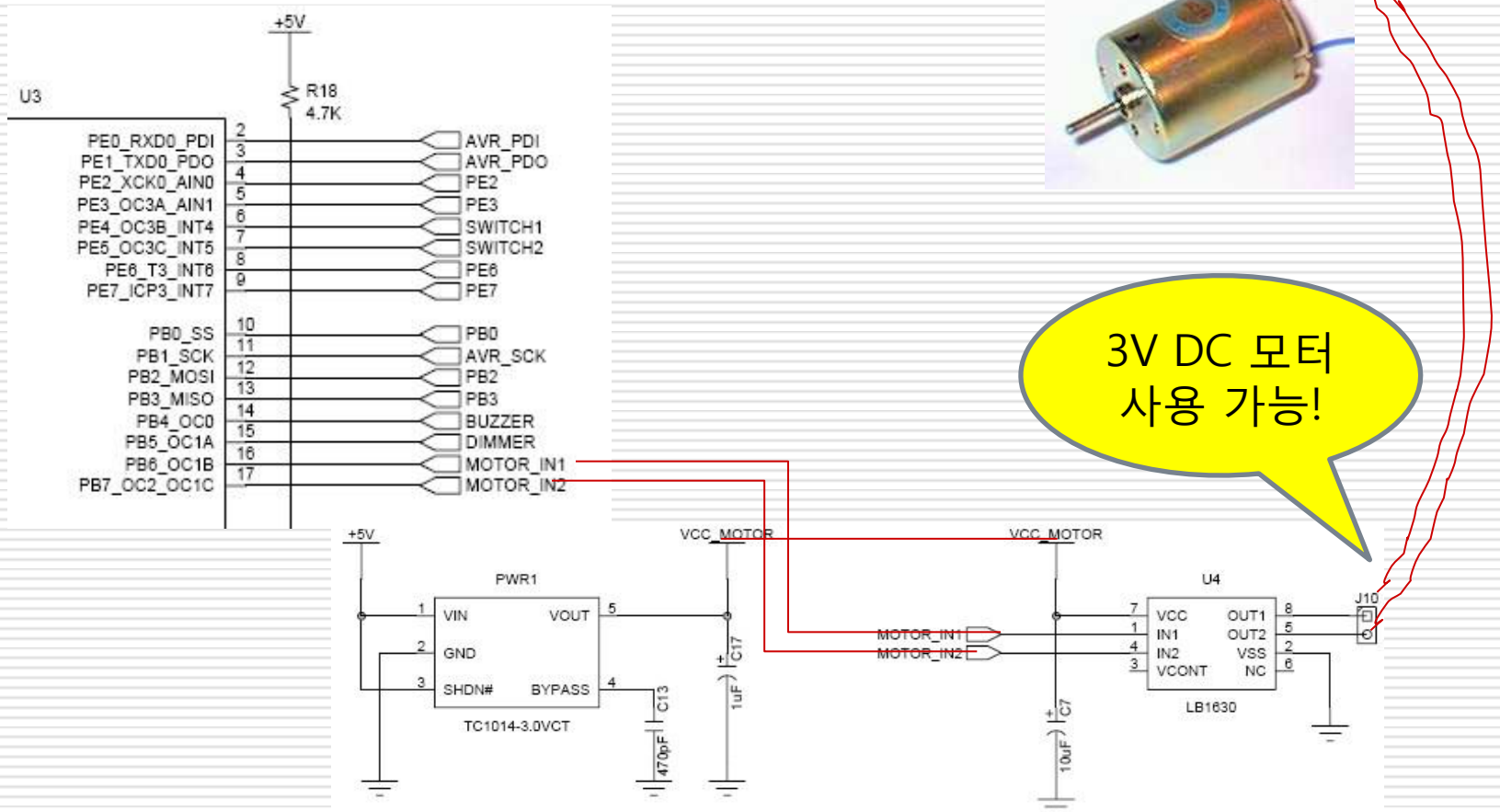
□ LB1630 : 저전압용 DC 모터 드라이버 IC

- 2개 TTL 입력을 트랜지스터를 거친 모터용 출력으로 전환
- IN1, IN2의 값에 따라 양방향 구동 가능
 - 10 : 정방향, 01 : 역방향, 00, 11 : 정지
- 역기전력(스파크) 보호 회로(Flywheel Diode) 내장
- 출력 전압 : 2.5V ~ 6.0V
- 최대 소모 전력 : 785mW
- 8핀 DIP 타입 IC



JKIT-128-1에서의 모터 연결 설계

□ JKIT-128-1에서의 모터 연결 설계



실습 MR-1 : 모터 움직이기

□ 실습 내용

1. 모터 움직이기

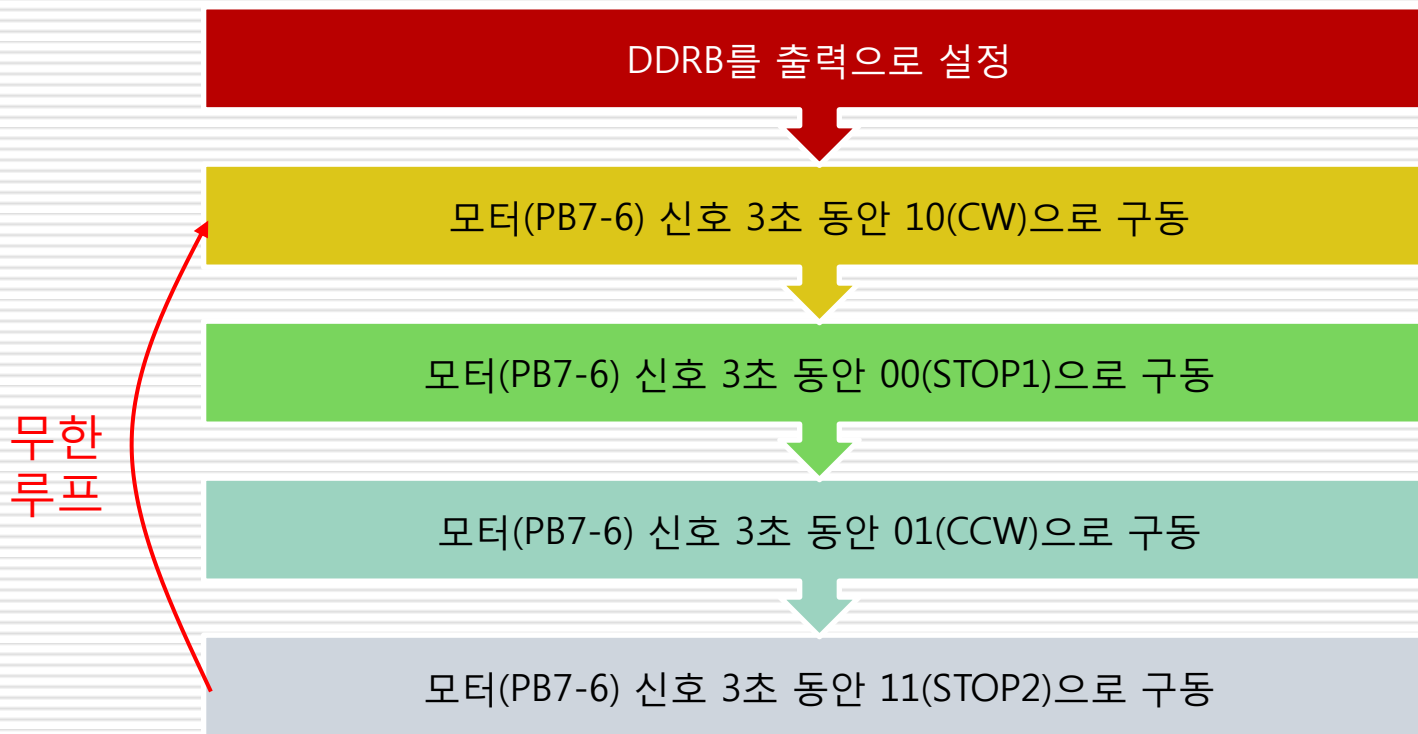
시계 방향 3초 → 정지 3초 → 반시계방향 6초 → 정지 3초
반복

실습 MR-1 : 모터 움직이기

- 구동프로그램 설계 : 모터 움직이기 (moter_1_1.c)
 - 모터에 연결되는 2개의 GPIO 신호의 값을 다르게(10 또는 01)하면 모터가 기동하고, 신호의 값을 같게(00, 또는 11)하면 모터가 정지하므로 이를 이용함
 - 모터에 할당된 신호는 PB7, PB6 신호임
 - PB7, PB6의 조합을 적당한 시간만큼의 delay를 주고 10 → 00 → 01 → 11의 값을 갖도록 반복하는 프로그램 작성
 - DC 모터는 관성이 있으므로 아주 짧은 시간 내에 기동, 정지를 수행하면 동작하지 않을 수도 있으므로 주의!

실습 MR-1 : 모터 움직이기

□ 구동프로그램 설계 : 모터 움직이기 (moter_1_1.c)



실습 MR-1 : 모터 움직이기

□ 구동프로그램 설계 : 모터 움직이기 (moter_1_1.c)

```
#include <avr/io.h>
#define F_CPU 16000000UL
#define MOTOR_CW      0x80
#define MOTOR_CCW      0x40
#define MOTOR_STOP1    0x00
#define MOTOR_STOP2    0x11
#include <util/delay.h>

void delay_sec(int sec)
{
    int i;
    for(i=0; i<1000*sec; i++)
        _delay_ms(1);
}
```

```
int main(void)
{
    DDRB = 0xff;
    while(1)
    {
        PORTB = MOTOR_CW;
        delay_sec(3);
        PORTB = MOTOR_STOP1;
        delay_sec(3);
        PORTB = MOTOR_CCW;
        delay_sec(6);
        PORTB = MOTOR_STOP2;
        delay_sec(3);
    }
}
```


실습 MR-2 : 모터 속도 조절하기

□ 실습 내용

1. 모터를 저속, 중속, 고속의 3단계로 나누어 10초씩 기동하되, 이의 구현을 PWM을 이용하여 프로그램하기
2. 모터를 아래의 테이블 형태로 임의 구동하기 (각자 해보기)

순서	방향	강약	시간
1	시계	약	3초
2	시계	강	5초
3	정지		1초
4	반시계	강	6초
5	정지		4초
6	시계	약	9초
7	반시계	중	7초

실습 MR-2 : 모터 속도 조절하기

- 구동프로그램 설계 : 모터 속도 조절 (moter_2_1.c)
 - 모터에 연결되는 2개의 GPIO 신호 중 1개의 신호값(PB6)을 0으로 고정하고 다른 1개의 신호(PB7/OC2)를 PWM 방식을 이용하여 신호 생성(8비트 Timer/Counter2 이용)
 - Fast PWM 모드 사용
 - 고속의 경우는 PWM 방식을 duty cycle 90% 이상으로 하고, 중속은 duty cycle 70% 정도, 저속은 duty cycle 50% 정도로 처리할 수 있도록 OCR2의 값을 설정
 - OCR2 세팅 값
 - 고속 : $255 * (1-0.9) = 26$
 - 중속 : $255 * (1-0.7) = 77$
 - 저속 : $255 * (1-0.5) = 128$
 - 출력 비교 인터럽트 서비스 루틴에서는 count 값 1 증가
 - 메인 루틴에서는 count 값으로 시간 경과 확인

실습 MR-2 : 모터 속도 조절하기

□ 구동프로그램 설계 : 모터 속도 조절 (moter_2_1.c)

■ main() 프로그램

- TCCR2 : Fast PWM 모드, Compare Output Mode = Match시 ON, 64분주
- TIMSK : OCIE2 세트
- 저속으로 5초간 모터 기동 (count 값이 1000이면 1초 경과)
- 중속으로 5초간 모터 기동
- 고속으로 5초간 모터 기동
- TCCR2 : Nomal 모드, Normal Mode(No OC2), 64분주
- 참고 : TCNT2가 1 루프를 수행하는데 걸리는 시간은 $1/16000000 * 64 * 256 \text{ (sec)} = 1024 \text{ (us)} = 1 \text{ (ms)}$ 이므로 약 1000번 정도 반복 수행하면 1초 경과함 (count 값 체크)

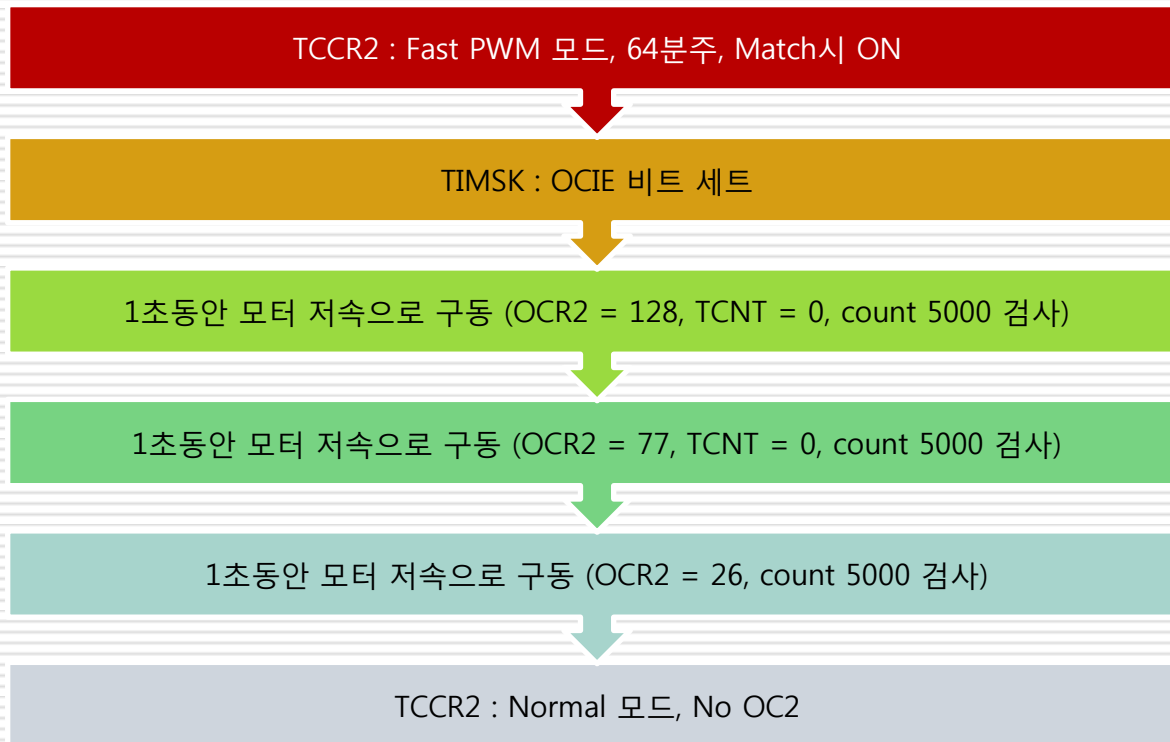
■ OC2 인터럽트 서비스 프로그램

- count 값을 1 증가

실습 MR-2 : 모터 속도 조절하기

□ 구동프로그램 설계 : 모터 속도 조절 (moter_2_1.c)

■ main() 프로그램



실습 MR-2 : 모터 속도 조절하기

- 구동프로그램 설계 : 모터 속도 조절 (moter_2_1.c)
 - OCR2 Match 인터럽트 서비스 프로그램

```
count++
```

실습 MR-2 : 모터 속도 조절하기

□ 구동프로그램 코딩 : 모터 속도 조절 (moter_2_1.c)

```
#include <avr/io.h>
#include <avr/interrupt.h>
volatile int count=0;
SIGNAL(SIG_OUTPUT_COMPARE2)
{
    count++;
}
void motor(int sec, unsigned char type)
{
    OCR2 = type;
    TCNT2 = 0;
    PORTA = type;
    while(count != 1000*sec)      ;
    count = 0;
}
```

```
#define MAX    26
#define MID    77
#define MIN    128
int main(void)
{
    DDRA = 0xff;
    DDRB = 0xff;
    PORTB = 0x00;
    TCCR2 = 0x7B;
    TIMSK = 0x80;
    sei();
    motor(5, MIN);
    motor(5, MID);
    motor(5, MAX);
    TCCR2 = 0x03;
}
```

실습 MR-3 : 모터 제어로 미니 선풍기 만들기

□ 실습 내용

1. SW1을 한 번 누를 때마다 모터가 저속 → 중속 → 고속 상태로 회전하며(계속 누르면 처음부터 다시 반복), SW2를 누르면 정지하는 미니 선풍기를 프로그램하기 (각자 해보기)

2. 힌트

- state에 따라 motor() 함수를 실행
- SW1가 한 번 눌러질 때마다 state를 저속 → 중속 → 고속으로 변화시킴
- SW2는 state를 정지로 변화시킴

묻고 답하기

Q & A

