모터

마이크로프로세서

HRI 연구실

김동한

Human-Robot Interaction Laboratory

KYUNG HEE UNIVERSITY

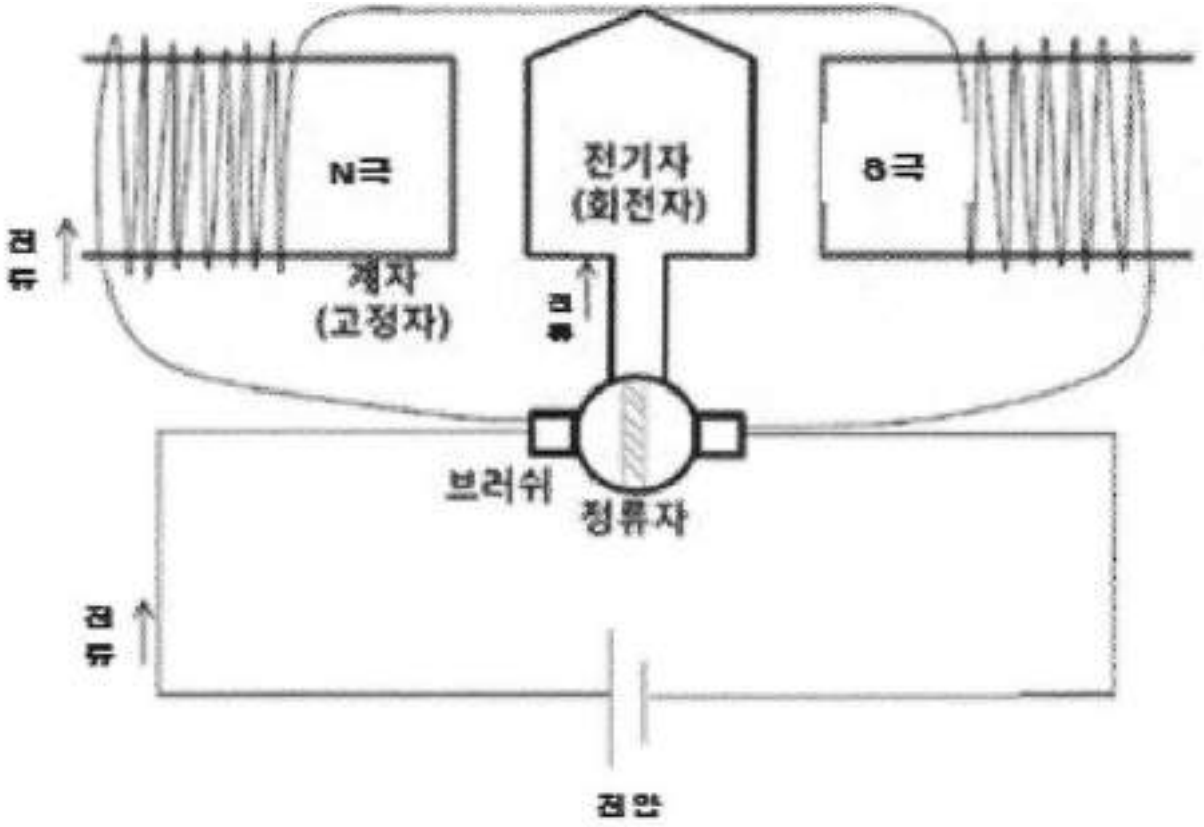
7.1 DC Motor 이론 및 실습

7.1.1 DC motor의 개요 및 내부 구성도

- 직류 전류를 전원으로 이용하여 돌아가는 모터
- 정류자편, 브러쉬, 전기자 및 계자 코일로 구성
- 전압을 인가하면 회전자 도선 속에 전류가 흐름

  => 주변 자석 (고정자) 과 상호작용하여 플레밍의 왼손 법칙에 의해 힘이 생겨 회전자가 계속 돌아감

- 대부분의 가전기기 (세탁기, 선풍기, 믹서 등) , 도구 및 장난감에서 핵심 부품으로 사용



7.1 DC Motor 이론 및 실습

7.1.2 Transistor를 이용한 DC motor 제어 회로

- Transistor를 이용하여 DC motor를 제어하는 방법

① Vce 전압이 낮은 경우

Collector와 emitter 사이에 전압이 낮을 경우의 회로, 효율이 가장 좋은 경우의 회로

② 역기전력 방지를 위한 제어회로

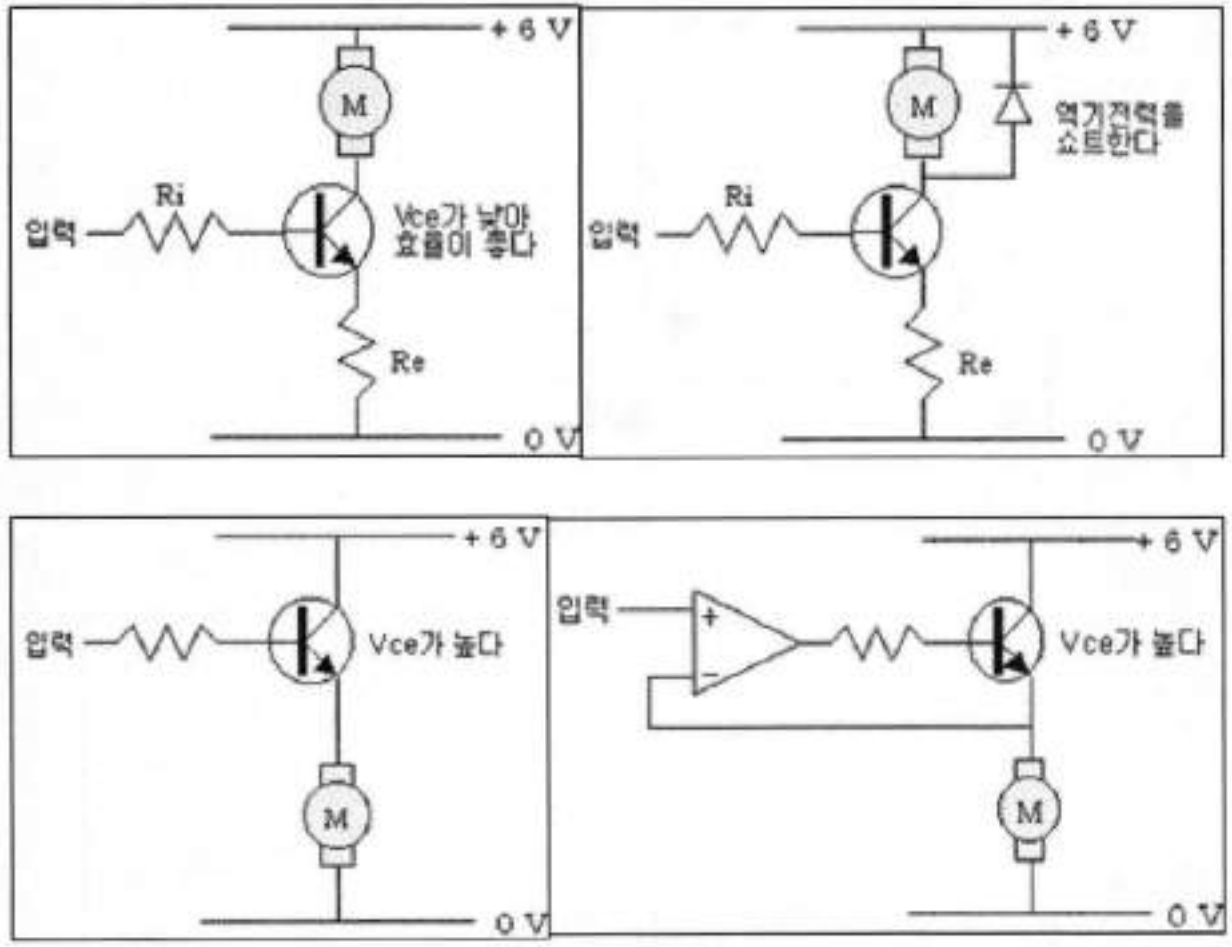
저전력의 DC motor를 구동하면서 역기전력이 발생 할 우려가 있을 경우에 다이오드를 이용하여 역기전력을 방지하는 회로를 구성

③ Vce 전압이 높은 경우

Transistor의 출력 전압을 높게 형상하는 경우, 실제 고전력 모터를 사용할 때 구성

④ 증폭기를 이용한 경우

Vce 전압을 높게 형성하고 emitter 전압과 입력 전압을 비교하여 증폭 형태로 base에 입력되는 전압을 조절하는 회로



KYUNG HEE UNIVERSITY

7.1 DC Motor 이론 및 실습

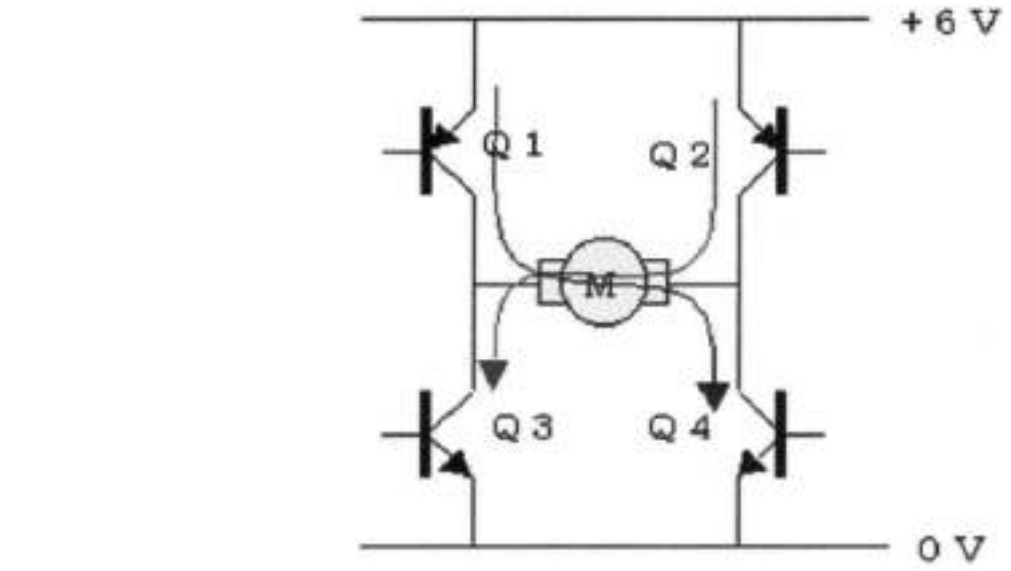
7.1.3 Transistor를 이용한 DC motor 제어용 H - 회로 구성하기

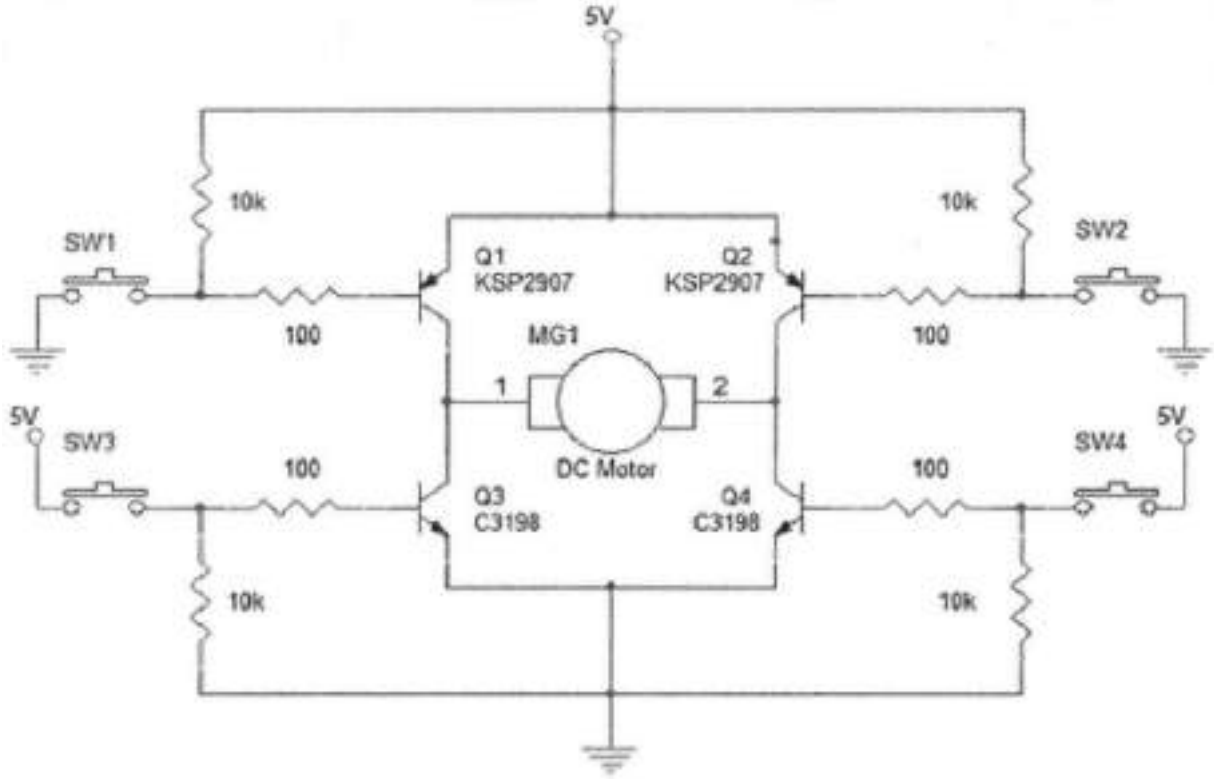
- NPN 형과 PNP 형 transistor 2개 씩 사용하여 모터의 정회전과 역회전을 하기 위한 H - 회로를 구성

- H - 회로

  Transistor의 Q1의 Q4가 작동하면 모터는 정회전하고, transistor의 Q2와 Q3가 작동하면 모터는 역회전하는 제어 방식으로 H자 형태로 구성

- 단순히 모터를 정회전과 역회전을 제어하는 경우 사용 (모터의 RPM을 제어하는 회로 X)

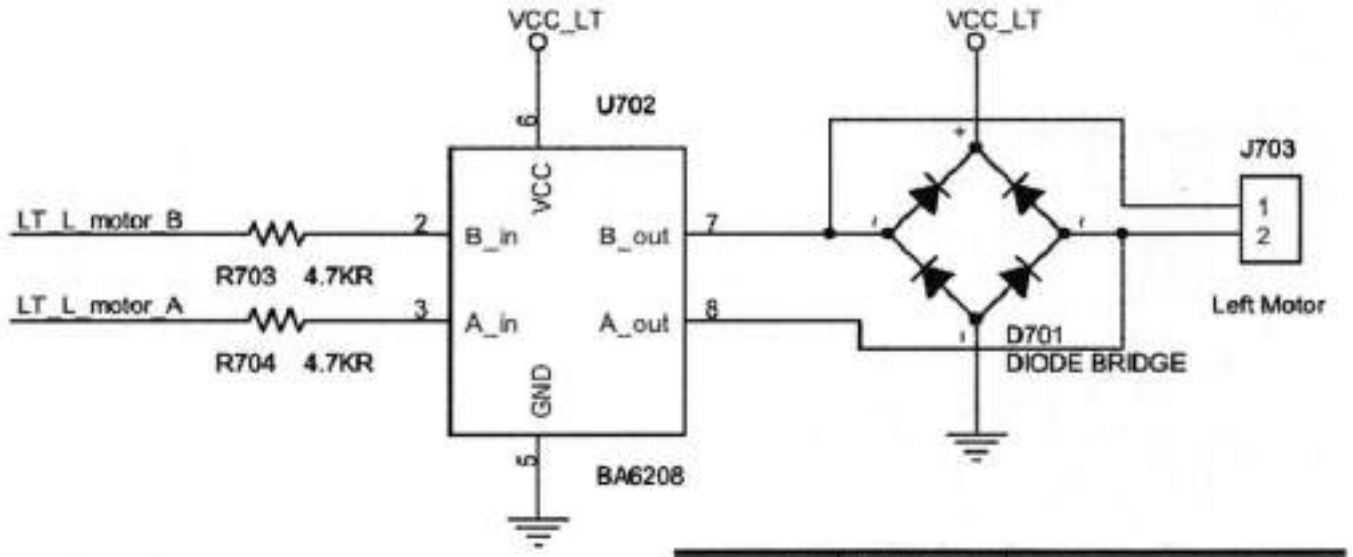


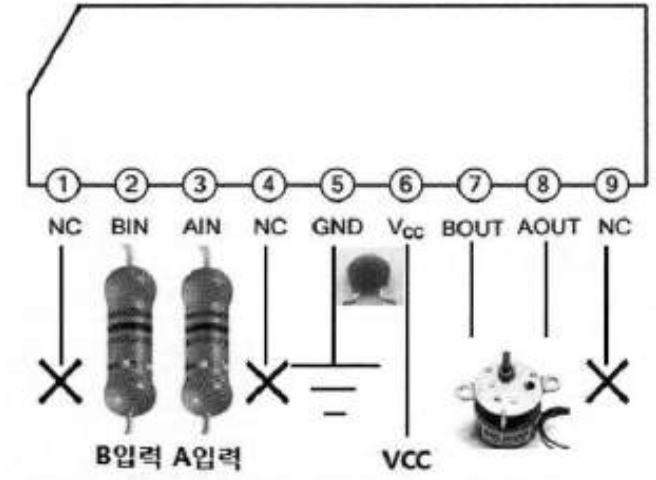


7.1 DC Motor 이론 및 실습

7.1.4 BA6208 DC motor 전용 IC를 이용한 모터 제어 회로

- BA6208
  - H - 회로의 원리가 내장된 DC motor 제어 전용 IC
  - 9핀 (1, 4, 9번은 사용하지 않는 포트)
  - 입력은 2, 3번으로 신호를 인가하면 7, 8번에 모터를 연결하여 H - 회로 방식으로 제어
  - DC motor를 구동하기 위한 전원
    - ① 5번 (GND, 0V), 6번 (Vcc, +5V 이상) 에 공급
    - ② 정전압을 인가하기 위해 사이에 세라믹 콘덴서 추가





B입력	A입력	모터
0	1	정회전
1	0	역회전
1	1	정지
0	0	전원 공급 중지

0	PWM	정회전 속도
1	PWM	역회전 속도

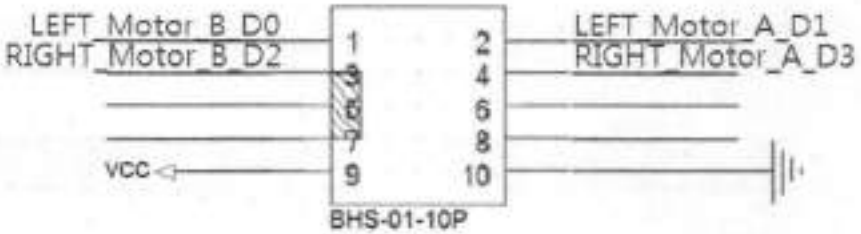
KYUNG HEE UNIVERSITY

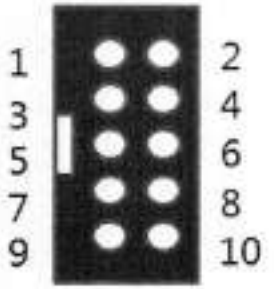
7.1 DC Motor 이론 및 실습

실습 3. BA6208 모터 전용 IC를 이용한 DC motor 정 / 역회전 제어 실험

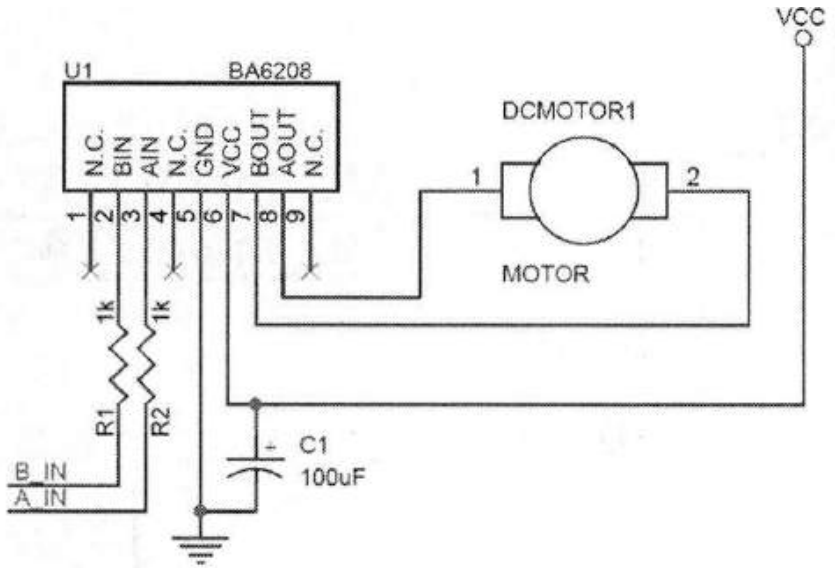
- H - 회로 1개가 들어 있는 모터 전용 IC BA6208을 이용하여 정 / 역회전을 제어 할 수 있는 실습을 수행
- 실험 장비 : 전원공급장치 (DC 0 ~ 30V), 멀티테스터기, Bread Board
- 실험 재료 : DC motor, BA6208 모터 전용 IC, 저항 (100 Ohm), 점퍼선

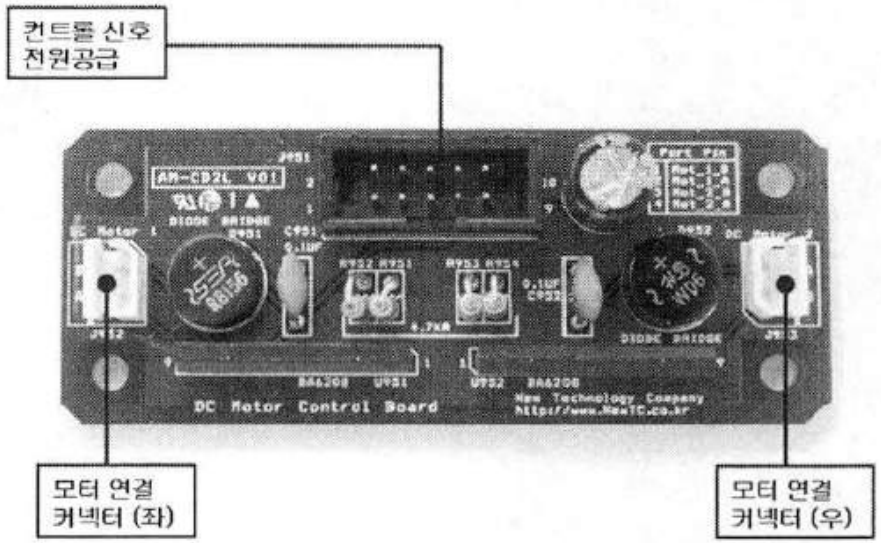
TOP VIEW
(AM-CD2L)





LEFT_Motor_B_D0	1	2	LEFT_Motor_A_D1
RIGHT_Motor_B_D2	3	4	RIGHT_Motor_A_D3
	5	6	
	7	8	
VCC(DC5V)	9	10	GND





7.1 DC Motor 이론 및 실습

실습 3. BA6208 모터 전용 IC를 이용한 DC motor 정 / 역회전 제어 실험

- BA6208 IC의 제어 신호에 따른 출력 신호 결과

■ **INPUT TRUTH TABLE**

3pin (Ain)	2pin (Bin)	8pin (Aout)	7pin (Bout)
H	L	H	L
L	H	L	H
H	H	L	L
L	L	OPEN	OPEN

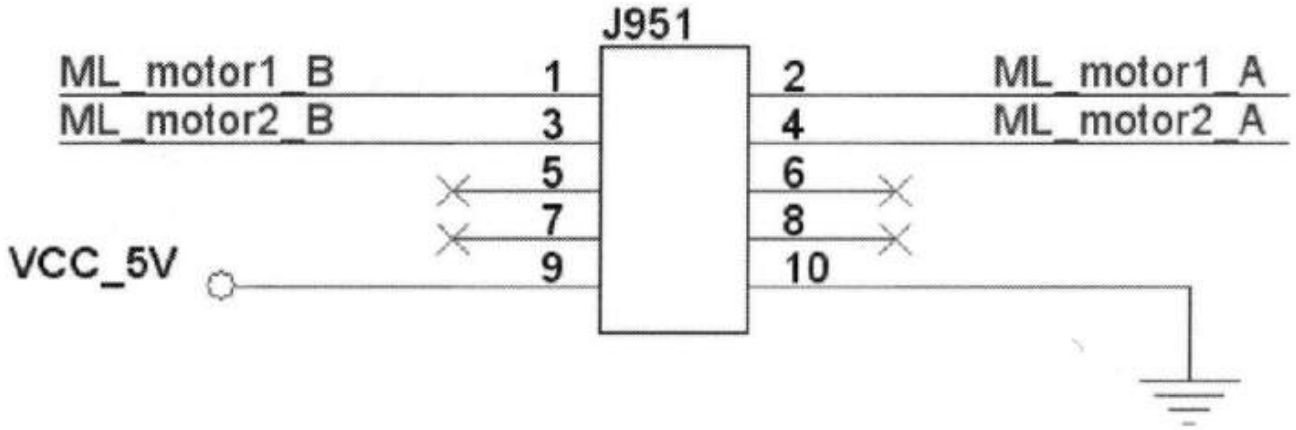
PORTC를 이용하여 모터 2개 제어 가능

PORTC.0(ML_motor1_B) PORTC.1(ML_motor1_A)

(1) M1 정/역회전 가능

(2) PORTC.2(ML_motor2_B)
PORTC.3(ML_motor2_A)
M2 정/역회전 가능

- ATmega128에 PORTC에 연결된 포트의 계략도



#1	#3	#5	#7	E9
왼쪽모터 B상	오른쪽 모터 B상			전원 5V

#2	#4	#6	#8	#10
왼쪽 모터 A상	오른쪽 모터 A상			전원 GND

7.1 DC Motor 이론 및 실습

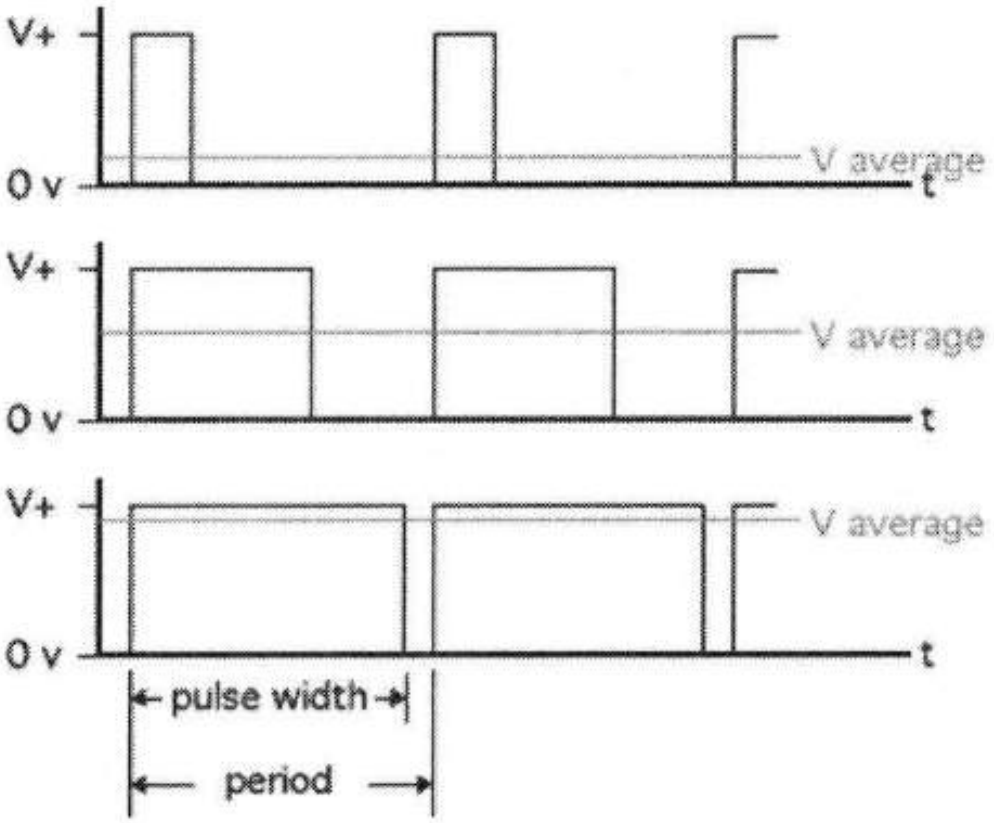
실습 3. BA6208 모터 전용 IC를 이용한 DC motor 정 / 역회전 제어 실험

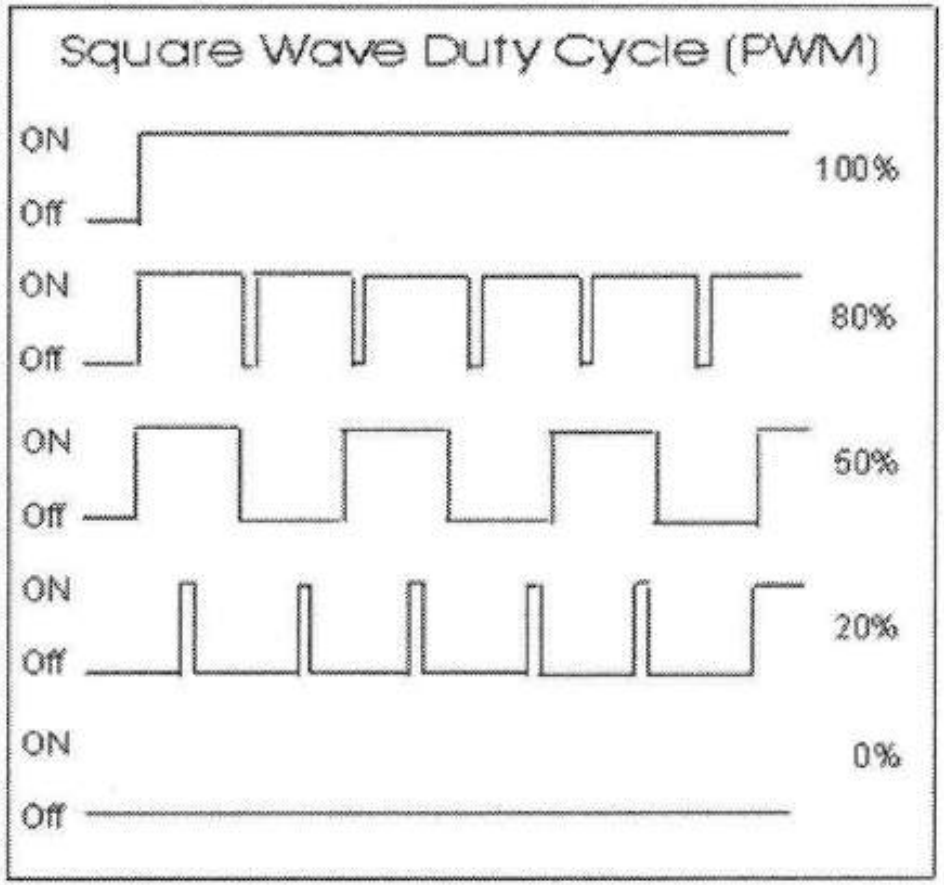
- BA6208을 이용한 DC motor 제어 프로그램 결과

```
#include <mega128.h>
#include <delay.h>

#define SW1 PORTC.0
#define SW2 PORTC.1
#define SW3 PORTC.2
#define SW4 PORTC.3

void main(void)
{
    PORTC = 0b00000011
    DDRC = 0b00001111;

    while(1)
    {
        PORTC = 0b00000001;
        delay_ms(1000);

        PORTC = 0b00000000;
        delay_ms(1000);

        PORTC = 0b00000010;
        delay_ms(1000);

        PORTC = 0b00000000;
        delay_ms(1000);
    }
}
```

7.1 DC Motor 이론 및 실습

[과제 8] BA6208 모터 전용 IC를 이용한 DC motor PWM 제어 실험





- PWM 제어 신호 : 펄스 주기를 나누고 그 구간 안에서 '1'과 '0'의 비율에 의하여 RPM을 조절할 수 있는 제어 신호
- '1'의 비율을 10%, 20%, 50%, 80%, 100%로 키우면 RPM은 점점 증가 (평균 전압 증가)

7.1 DC Motor 이론 및 실습

[과제 8] BA6208 모터 전용 IC를 이용한 DC motor PWM 제어 실험

- BA6208 모터 전용 IC를 이용한 DC motor PWM 제어 프로그램

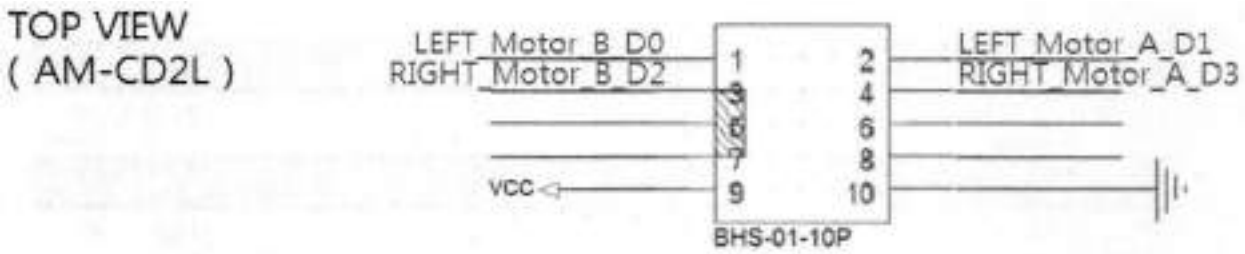
```
#include <mega128.h>
#include <delay.h>

void main(void)
{
    PORTC=0b00000000;
    DDRC=0b00000011;

    while(1)
    {
        PORTC=0b00000010;
        delay_ms(80);

        PORTC=0b00000000;
        delay_ms(20);
    }
}
```

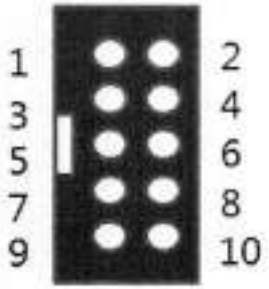
```
#include <mega128.h>
#include <delay.h>

void main(void)
{
    PORTC=0b00000000;
    DDRC=0b00000011;

    while(1)
    {
        PORTC=0b00000010;
        delay_ms(20);

        PORTC=0b00000000;
        delay_ms(80);
    }
}
```

7.1 DC Motor 이론 및 실습

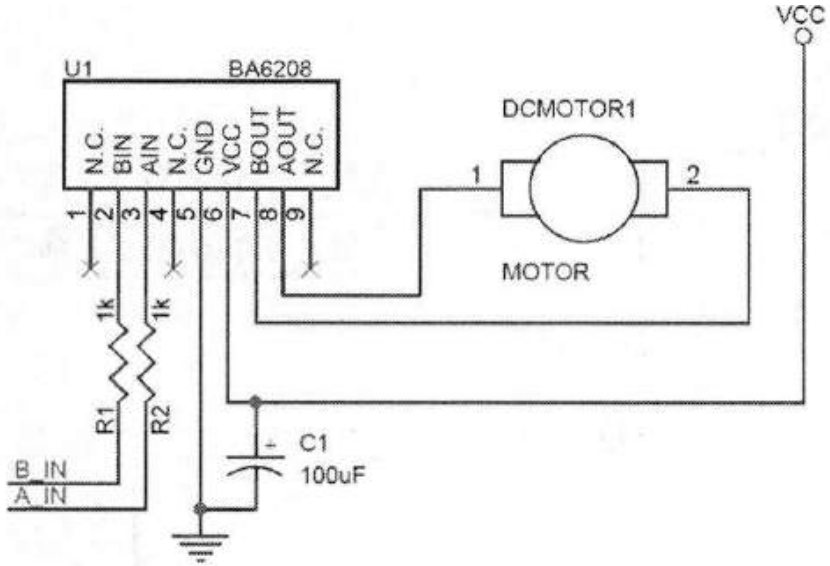
실습 4. BA6208 모터 전용 IC를 이용한 DC motor PWM 속도 제어 실험

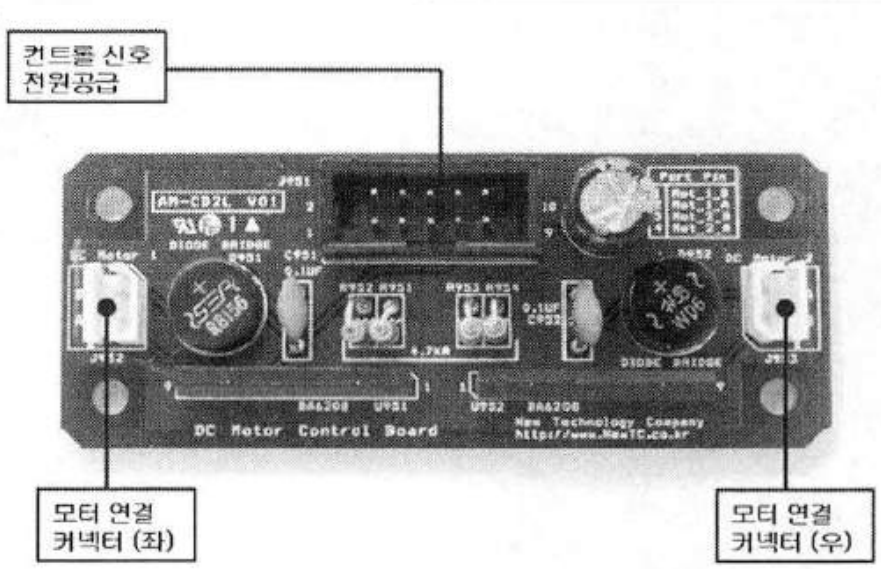
- H - 회로 1개가 들어 있는 모터 전용 IC BA6208을 이용하여 정 / 역회전을 제어 할 수 있는 실습을 수행
- 실험 장비 : 전원공급장치 (DC 0 ~ 30V), 멀티테스터기, Bread Board
- 실험 재료 : DC motor, BA6208 모터 전용 IC, 저항 (100 Ohm), 점퍼선





LEFT_Motor_B_D0	1	2	LEFT_Motor_A_D1
RIGHT_Motor_B_D2	3	4	RIGHT_Motor_A_D3
	5	6	
	7	8	
VCC(DC5V)	9	10	GND





7.1 DC Motor 이론 및 실습

실습 4. BA6208 모터 전용 IC를 이용한 DC motor PWM 속도 제어 실험

- BA6208 IC의 제어 신호에 따른 출력 신호 결과

PORTC를 이용하여 모터 2개 제어 가능

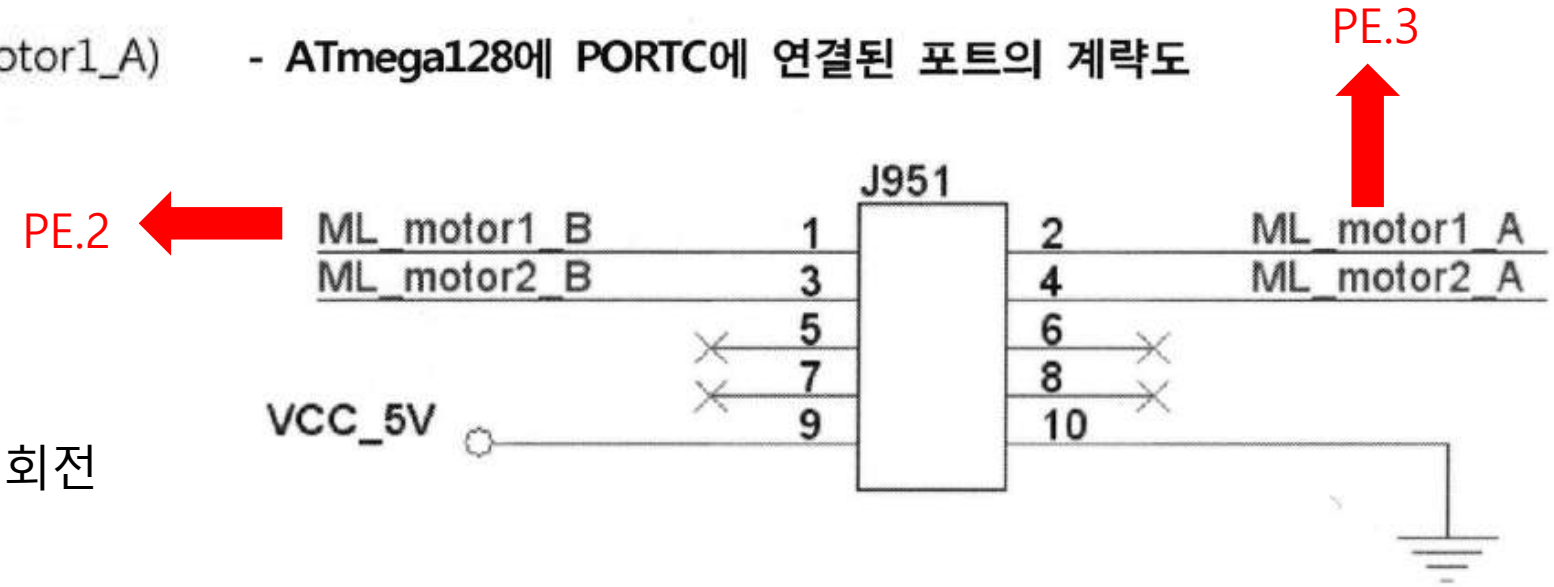
PORTC.0(ML_motor1_B) PORTC.1(ML_motor1_A)
(1) M1 정/역회전 가능
(2) PORTC.2(ML_motor2_B)
PORTC.3(ML_motor2_A)
M2 정/역회전 가능

PE.2와 PE.3에 신호를 입력하면 모터가 회전

■ **INPUT TRUTH TABLE**

3pin (Ain)	2pin (Bin)	8pin (Aout)	7pin (Bout)
H	L	H	L
L	H	L	H
H	H	L	L
L	L	OPEN	OPEN

- ATmega128에 PORTC에 연결된 포트의 계략도



#1	#3	#5	#7	E9
왼쪽모터 B상	오른쪽 모터 B상			전원 5V

#2	#4	#6	#8	#10
왼쪽 모터 A상	오른쪽 모터 A상			전원 GND

7.1 DC Motor 이론 및 실습

실습 4. BA6208 모터 전용 IC를 이용한 DC motor PWM 속도 제어 실험

```
#include <mega128.h>
#include <delay.h>
void setDCMotor(unsigned int dir, unsigned int speed)
{
    if(dir == 0)
    {
    PORTE.2 = 0;
    OCR3AH = (speed>>8)&0xff;
    OCR3AL = speed&0xff;
    }
    else
    {
    PORTE.2 = 1;
    OCR3AH = ((1023-speed)>>8)&0xff;
    OCR3AL = (1023-speed)&0xff;
    }
}
```

7.1 DC Motor 이론 및 실습

실습 4. BA6208 모터 전용 IC를 이용한 DC motor PWM 속도 제어 실험

```
void main(void)
{
    DDRE = 0x0c;
    PORTE = 0x00;

    TCCR1A=(1<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) |
    (0<<COM1C1) | (0<<COM1C0) | (1<<WGM11) | (1<<WGM10);
    TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (1<
    <CS12) | (0<<CS11) | (0<<CS10);
    TCNT1H=0x00;
    TCNT1L=0x00;
    ICR1H=0x00;
    ICR1L=0x00;
    OCR1AH=0x00;
    OCR1AL=0x00;
    OCR1BH=0x00;
    OCR1BL=0x00;
    OCR1CH=0x00;
    OCR1CL=0x00;
```

7.1 DC Motor 이론 및 실습

실습 4. BA6208 모터 전용 IC를 이용한 DC motor PWM 속도 제어 실험

```
    while (1)
    {
        setDCMotor(0, 500);
        delay_ms(1000);
        setDCMotor(0, 0);
        delay_ms(1000);
        setDCMotor(1, 500);
        delay_ms(1000);
    }
}
```

7.1 DC Motor 이론 및 실습

DC모터를 구동할 때 주의해야 할 점!

- 모터가 회전하고 있을 때, 모터 샤프트를 손으로 잡는다거나 회전하지 못하게 방해하는 행위를 하면 안됨. => 소모전류가 증가하여 주변 회로 및 모터 드라이버나 모터가 손상될 수 있음.
- 모터가 정지하였을 때, 모터를 강제로 돌리는 일을 하면 모터의 성능저하가 발생할 수 있음. 또한 순간적으로 강한 전류가 발생하여 회로에 안좋은 영향을 줄 수 있음.
- 저속에서 큰 전류를 소모함. 따라서 저속으로 오래 구동하는 것은 좋지 않음. => 저속으로 오래 구동시켜야 한다면 기어를 사용하는 것이 바람직 함.
- 모터 구동시 강한 전류가 필요하므로, 반드시 모터 구동 IC또는 모터 구동 회로를 구성하여 모터를 구동시킴. 일반적인 I/O 핀에 바로 연결시켜 구동하면 MCU가 손상될 수 있음!

7.1 DC Motor 이론 및 실습

- DC모터를 구동할 때 주의해야 할 점!

  - DC모터는 극성이 바뀌면 회전방향이 바뀜.
  - DC모터를 급격하게 회전시키면 순간적으로 많은 전류를 소모하며, 충분한 전류를 공급해주지 못할 경우 전류 부족으로 인한 리셋과 같은 현상이 발생할 수 있음.
  - DC모터의 회전 방향을 급격히 바꾸면(또는 갑자기 회전속도를 줄이거나 정지시키면) 역기전력이 발생하여 주변 회로에 안 좋은 영향을 주며, 심한 경우 회로에 손상이 발생.

7.1 DC Motor 이론 및 실습

- DC모터를 구동시키기 위한 방법

- 가장 흔한 방법으로는 시중에 나와있는 모터 드라이버를 이용한다. 즉, MCU와 모터를 바로 연결하지 않고, 중간에 모터 드라이버를 연결하여 MCU⇔모터 드라이버⇔모터 방식으로 모터를 제어한다.

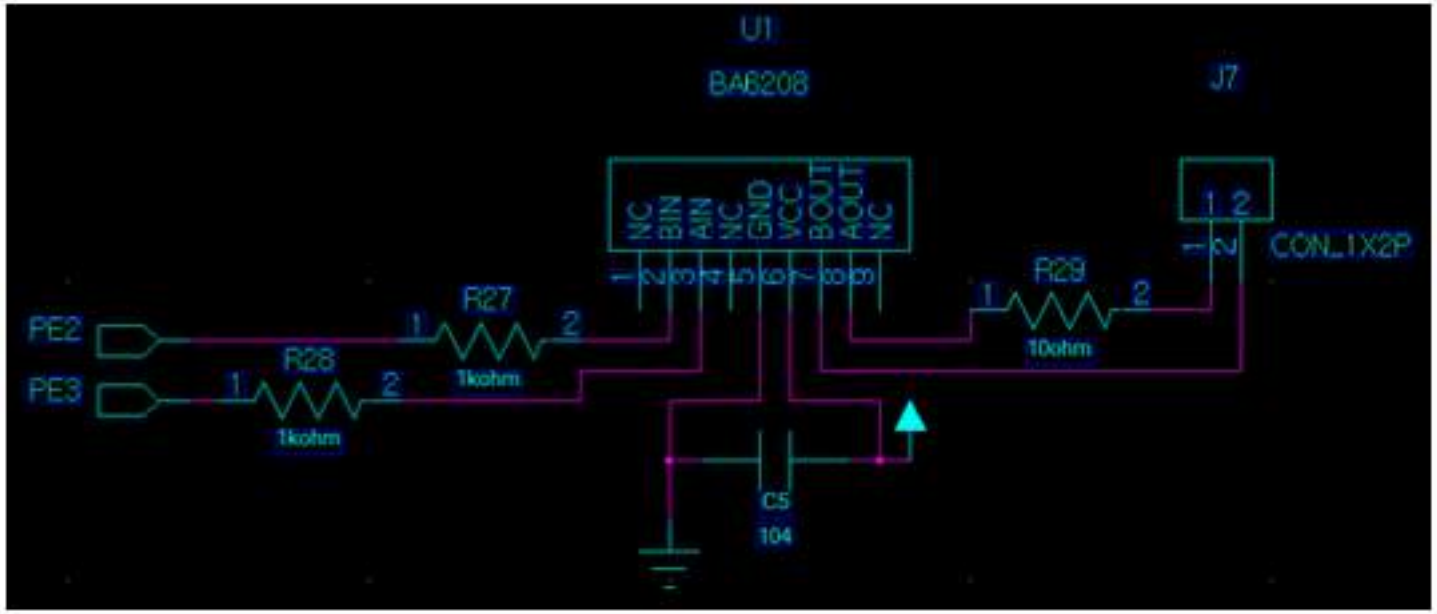
- 모터 드라이버가 하는 역할은 크게

1. 모터에 직접적인 전원 공급
2. MCU의 Logical High, Low를 입력받아 모터에 신호 출력
3. 모터의 회전 방향 제어

라고 볼 수 있다.

7.1 DC Motor 이론 및 실습

다음과 같이 회로를 구성



위와 같이 BA6208의 2, 3번 핀을 각각 1k옴의 저항을 거쳐 PE.2, PE.3에 연결.
BA6208의 8, 9번 핀 중 하나를 10옴의 저항을 거쳐 모터에 연결.
위와 같이 연결하고 PE.2와 PE.3에 신호를 입력하면 모터가 회전한다!

7.1 DC Motor 이론 및 실습

다음과 같은 코드를 작성 (**GPIO로 구동시키기**)

```
1  #include <mega128.h>
2  #include <delay.h>
3
4  main()
5  {
6      /**   GPIO 입출력 설정   **/
7      DDRE=0x0c; //PE.2와 PE.3에 BA6208 연결. 출력설정.
8      PORTE=0x00; //E포트 출력 초기화
9      /*************************/
10
11     while(1)
12     {
13         PORTE=0x08; //PE.2에 Low, PE.3에 High => 정방향 회전.
14         delay_ms(1000);
15         PORTE=0x00; //PE.2에 Low, PE.3에 Low => 모터 Open.
16         delay_ms(1000);
17         PORTE=0x04; //PE.2에 High, PE.3에 Low => 역방향 회전.
18         delay_ms(1000);
19     }
20 }
```

BA6208에 신호를 출력해주기 위해 다음과 같이 설정.

7.1 DC Motor 이론 및 실습

다음과 같은 코드를 작성 (**PWM으로 구동시키기**)

모터 구동 함수 정의

```
33  void setDCMotor(unsigned int dir, unsigned int speed)
34  {
35      if(dir == 0) //정방향 회전
36      {
37          PORTE.2 = 0;
38          OCR3AH = (speed>>8)&0xff;
39          OCR3AL = speed&0xff;
40      }
41      else //역방향 회전
42      {
43          PORTE.2 = 1;
44          OCR3AH = ((1023-speed)>>8)&0xff;
45          OCR3AL = (1023-speed)&0xff;
46      }
47
48      //PORTE.2 = dir;
49      //OCR3AH = dir==0?(speed>>8)&0xff:((1023-speed)>>8)&0xff;
50      //OCR3AL = dir==0?speed&0xff:(1023-speed)&0xff;
51  }
```

역방향 회전 시, 1023-speed를 해주는 이유에 대해 생각해보자!

7.1 DC Motor 이론 및 실습

다음과 같은 코드를 작성 (**PWM으로 구동시키기**)

```
main()
{
    /**   GPIO 입출력 설정  **/
    DDRE=0x0c; //PE.2와 PE.3에 BA6208 연결. 출력설정
    PORTE=0x00; //E포트 출력 초기화
    /*************************/

    // Timer/Counter 3 initialization
    // Clock source: System Clock
    // Clock value: 62.500 kHz
    // Mode: Ph. correct PWM top=03FFh
    // OC3A output: Non-Inv.
    TCCR3A=0x83;
    TCCR3B=0x04;
    TCCR3C=0x00;

    while(1)
    {
        setDCMotor(0, 500);
        delay_ms(1000);
        setDCMotor(0, 0);
        delay_ms(1000);
        setDCMotor(1, 500);
        delay_ms(1000);
    }
}
```

7.1 DC Motor 이론 및 실습

다음과 같은 코드를 작성 (**PWM으로 구동시키기**)

모터 구동 함수 정의

```
33  void setDCMotor(unsigned int dir, unsigned int speed)
34  {
35      if(dir == 0) //정방향 회전
36      {
37          PORTE.2 = 0;
38          OCR3AH = (speed>>8)&0xff;
39          OCR3AL = speed&0xff;
40      }
41      else //역방향 회전
42      {
43          PORTE.2 = 1;
44          OCR3AH = ((1023-speed)>>8)&0xff;
45          OCR3AL = (1023-speed)&0xff;
46      }
47
48      //PORTE.2 = dir;
49      //OCR3AH = dir==0?(speed>>8)&0xff:((1023-speed)>>8)&0xff;
50      //OCR3AL = dir==0?speed&0xff:(1023-speed)&0xff;
51  }
```

이 두 가지 방식의 차이점을 생각해보자!

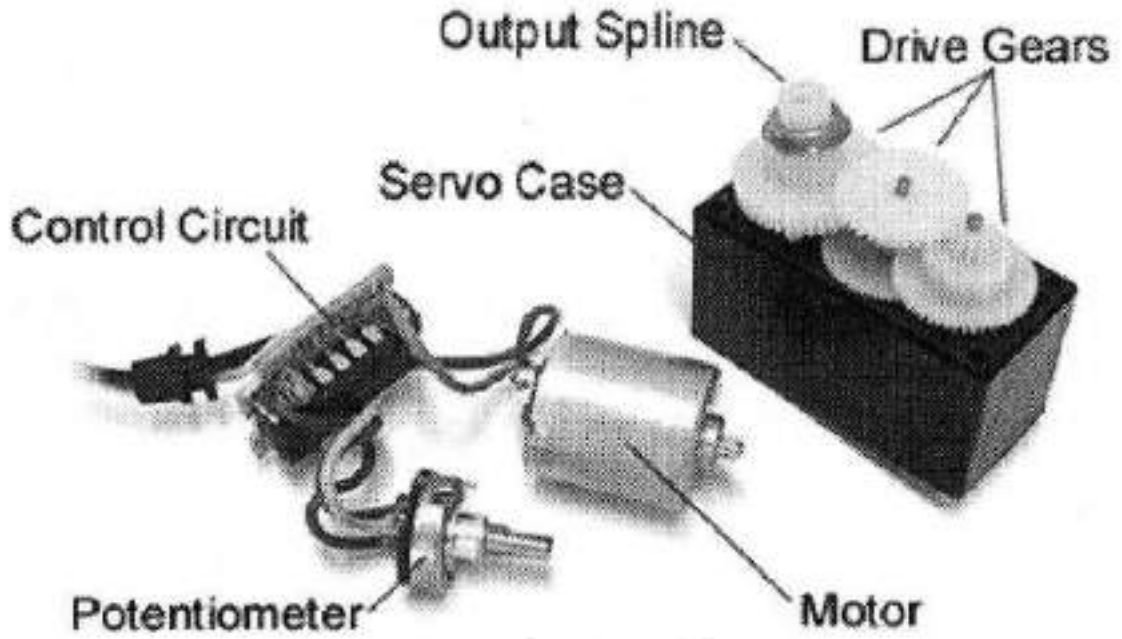
7.2 RC Servo Motor 이론 및 실습

7.2.1 RC Servo motor의 개요

- Servo Motor
  - 추종한다 / 따른다는 의미로서 명령을 따르는 모터 라는 뜻
  - 모터와 제어 구동 보드 (제어 회로와 알고리즘) 를 포함
  - 정확한 위치와 속도 제어 가능
  - CCTV 카메라, 캠코더, 프린터 등에 사용

7.2 RC Servo Motor 이론 및 실습

7.2.2 RC Servo motor의 내부 구성도



- 서보 구동부에서 위치 명령을 입력 받아 서보 모터를 제어
- 엔코더를 이용한 위치 검출을 통해 피드백 제어 시스템을 구성하여 구동 장치부의 위치 제어를 함

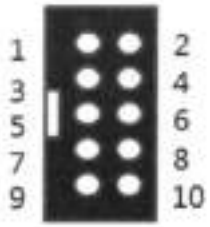
7.2 RC Servo Motor 이론 및 실습

7.2.3 RC Servo motor의 내부 구조도

- 위의 그림과 같이 DC 모터와 제어회로, 알고리즘으로 구성
- 제어 신호를 펄스로 인가 => 펄스에 따라 특성 각도를 유지 (피드백 제어 기능)

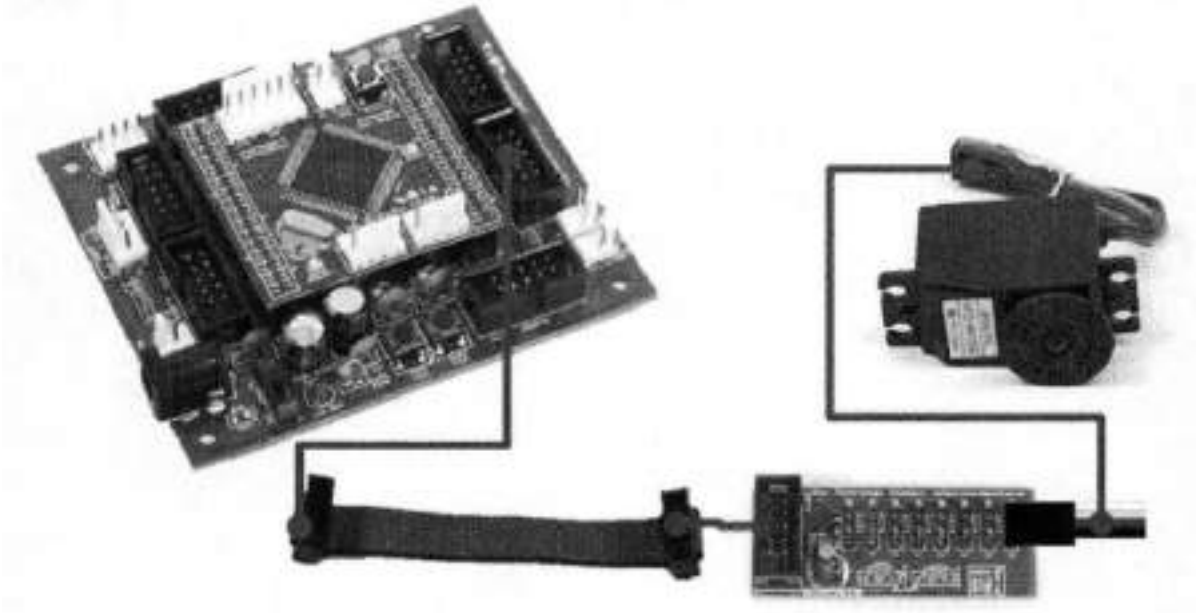
7.2 RC Servo Motor 이론 및 실습

7.2.4 RC Servo motor의 하드웨어 구성도



	1	2	
	3	4	
	5	6	
	7	8	
VCC(DC5V)	9	10	GND

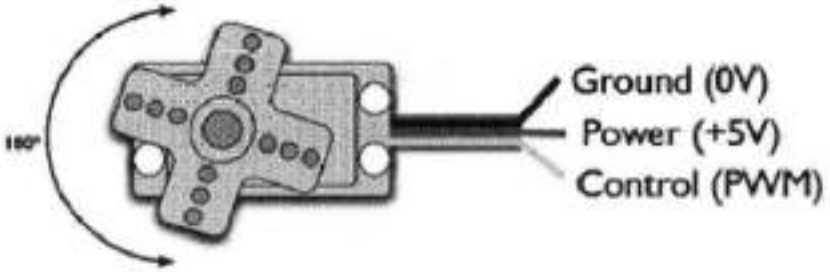
- RC 서보 모터의 선
  - 검은색 : 그라운드
  - 적색 : 전원 (4.8V ~ 6V)
  - 노란색 : 제어신호 선 (각도 제어)



- 현재 드라이버 포트는 8개의 RC 서보 모터를 제어 할 수 있도록 구성

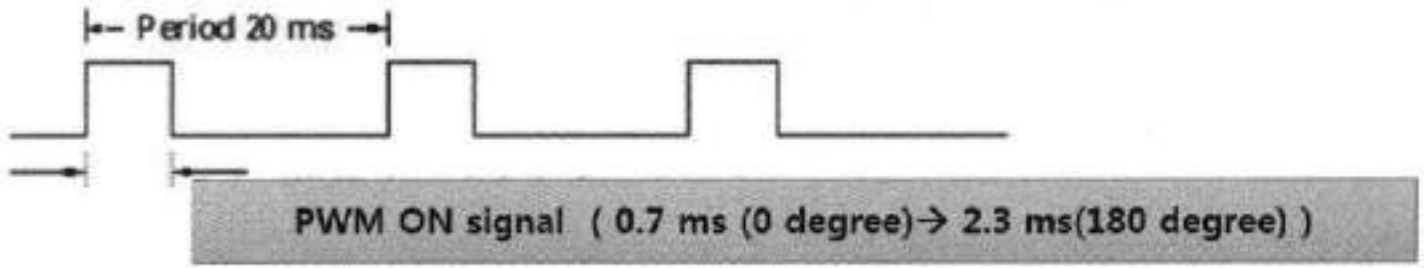
Port Connection 8bit Data

1st	3rd	5th	7th	9th
Motor 1st	Motor 3rd	Motor 5th	Motor 8th	Power 5V
2nd	**4th**	**6th**	**8th**	**10th**
Motor 2nd	Motor 4th	Motor 6th	Motor 7th	Power GND



7.2 RC Servo Motor 이론 및 실습

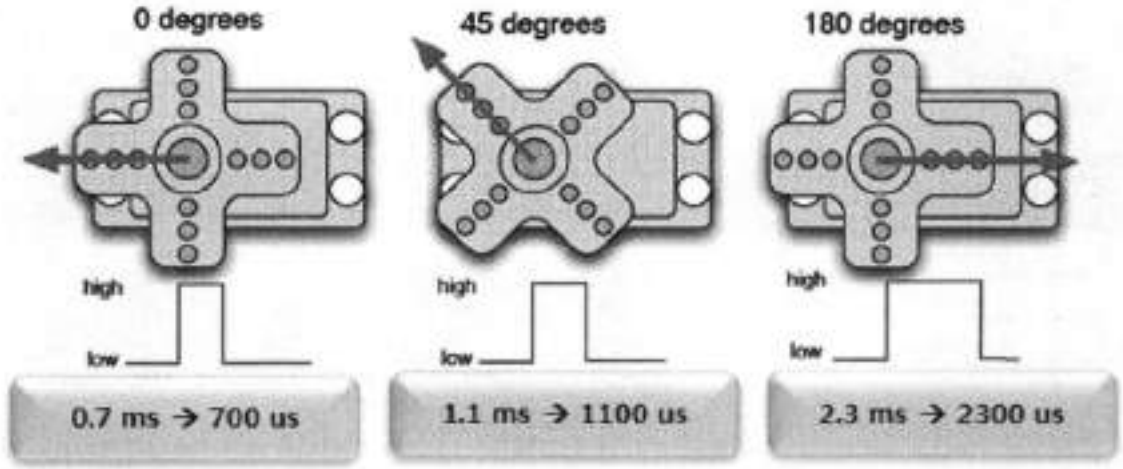
7.2.5 RC Servo motor의 제어 입력 신호에 따른 각도 변화



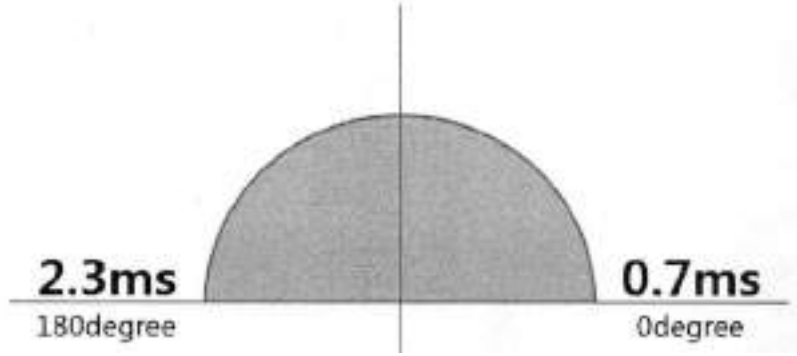
- 제어 입력 신호의 주기 : 20ms (0.02초)
- 펄스 주기와 내부 PWM 제어 방식으로 ON 시간 조절하여 각도를 제어
  - ON 시간 : 0.7ms - 0도, 2.3ms - 180도
- 각도 변화에 따른 펄스 계산 방법
  - 0도에서 180도 까지의 ON 시간 간격 계산
    2.3ms - 0.7ms = 1.6ms
  - 45도 계산
    180 : 1.6 = 45 : x
    x = (1.6 * 45) / 180 = 1.1ms

7.2 RC Servo Motor 이론 및 실습

7.2.6 RC Servo motor의 구동 원리



- RC 서보 모터는 180도의 각도를 제어 할 수 있음
- 실제 0도는 0.7ms, 180도는 2.3ms로 펄스의 ON 시간을 조절하여 각도 제어
- 주기는 20ms (0.02초)를 주기로 펄스를 발생
- 중앙에 위치한 상태에서,
- 2ms의 펄스가 인가되면 시계방향으로 90도,
- 1ms의 펄스가 인가되면 반 시계 방향으로 90도 움직임



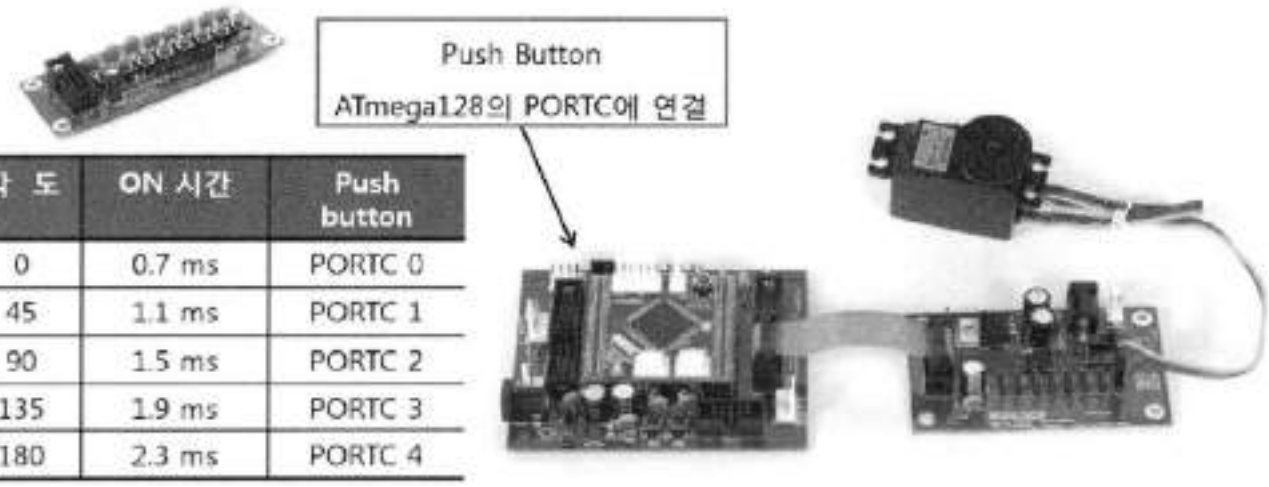
각도	ON 시간	OFF 시간	전체주기시간
0	0.7ms	19.3ms	20ms 0도에서 20-0.7 = 19.3ms
45	1.1ms		
90	1.5ms		
135	1.9ms		
180	2.3ms		

7.2 RC Servo Motor 이론 및 실습

실습 6. Delay 함수를 이용한 RC servo motor 각도 제어 하기

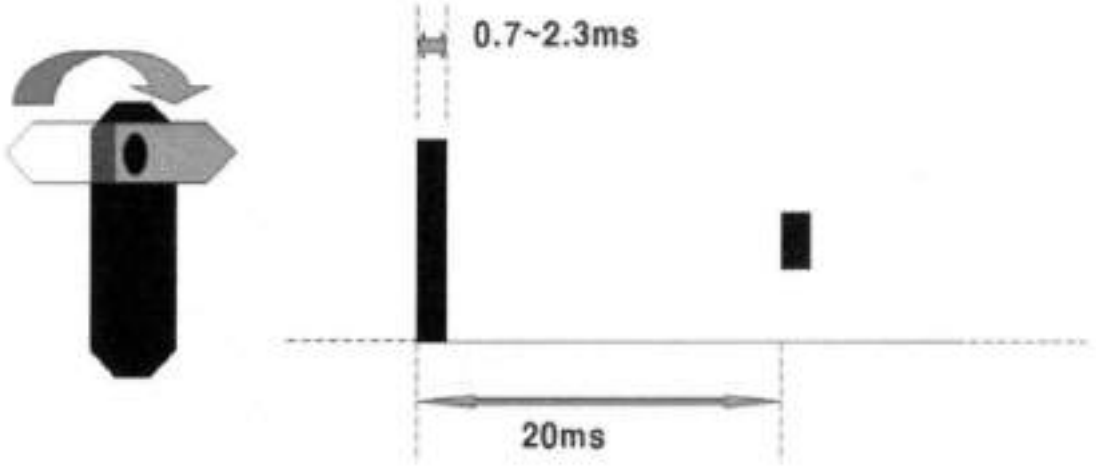
1) 기초 구동 실험을 위한 ATmega128과 Servo motor의 하드웨어 연결

① Servo motor를 PORTB의 PORTB.4(OC0)와 연결하여 SW에 의해 45도씩 회전하는 프로그램 작성
② (5V - 신호 - GND) 로 구성된 서보 모터의 연결을 오른 쪽 (위) 그림과 같이 구성
③ PORTC에 Push Button을 연결하여 입력 신호로 사용
④ SE-A410은 PWM 신호를 사용하여 구동, PWM 제어 타이밍도는 오른 쪽 (아래) 그림과 같음
⑤ 0도에 위치할 때에는 PWM의 듀티가 0.7ms가 나타나도록 신호 인가
⑥ 180도에 위치할 때에는 PWM 듀티가 2.3ms가 나타나도록 신호 인가
⑦ PWM 펄스 주기 : 20ms



각 도	ON 시간	Push button
0	0.7 ms	PORTC 0
45	1.1 ms	PORTC 1
90	1.5 ms	PORTC 2
135	1.9 ms	PORTC 3
180	2.3 ms	PORTC 4

(위)



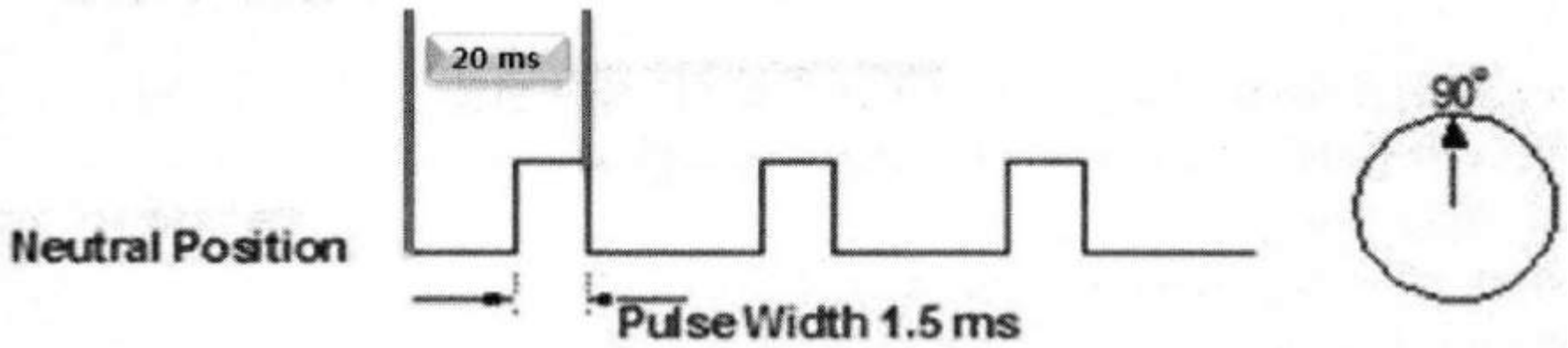
서보 모터 제어 타이밍도.

(아래)

7.2 RC Servo Motor 이론 및 실습

실습 6. Delay 함수를 이용한 RC servo motor 각도 제어 하기

2) 90도의 각도를 유지하기 위한 프로그램 구성



- 1주기에서 OFF 시간 계산 = 20ms - 1.5ms = 18.5ms

```
while(1)
{
    PORTB=0xFF;
    delay_us(1500);

    PORTB=0x00;
    delay_us(18500);
}
```

7.2 RC Servo Motor 이론 및 실습

실습 6. Delay 함수를 이용한 RC servo motor 각도 제어 하기

3) 기초 구동 실험

```
#include <mega128.h>
#include <delay.h>

void main(void)
{
    DDRB=0x10;
    PORTB=0x00;

    DDRC=0x00;
    PORTC=0x00;

    while(1)
    {
        //0도
        if(PINC.0 == 1)
        {
            PORTB=0xff;
            delay_us(700);
            PORTB=0x00;
            delay_us(19300);
        }

        //45도
        else if(PINC.1 == 1)
        {
            PORTB=0xff;
            delay_us(1100);
            PORTB=0x00;
            delay_us(18900);
        }

        //90도
        else if(PINC.2 == 1)
        {
            PORTB=0xff;
            delay_us(1500);
            PORTB=0x00;
            delay_us(18500);
        }
        //135도
        else if(PINC.3 == 1)
        {
            PORTB=0xff;
            delay_us(1900);
            PORTB=0x00;
            delay_us(18100);
        }

        //180도
        else if(PINC.4 == 1)
        {
            PORTB=0xff;
            delay_us(2300);
            PORTB=0x00;
            delay_us(17700);
        }
    }
}
```

7.2 RC Servo Motor 이론 및 실습

실습 7. Timer0와 가변 저항을 이용한 Servo motor 구동제어 실습

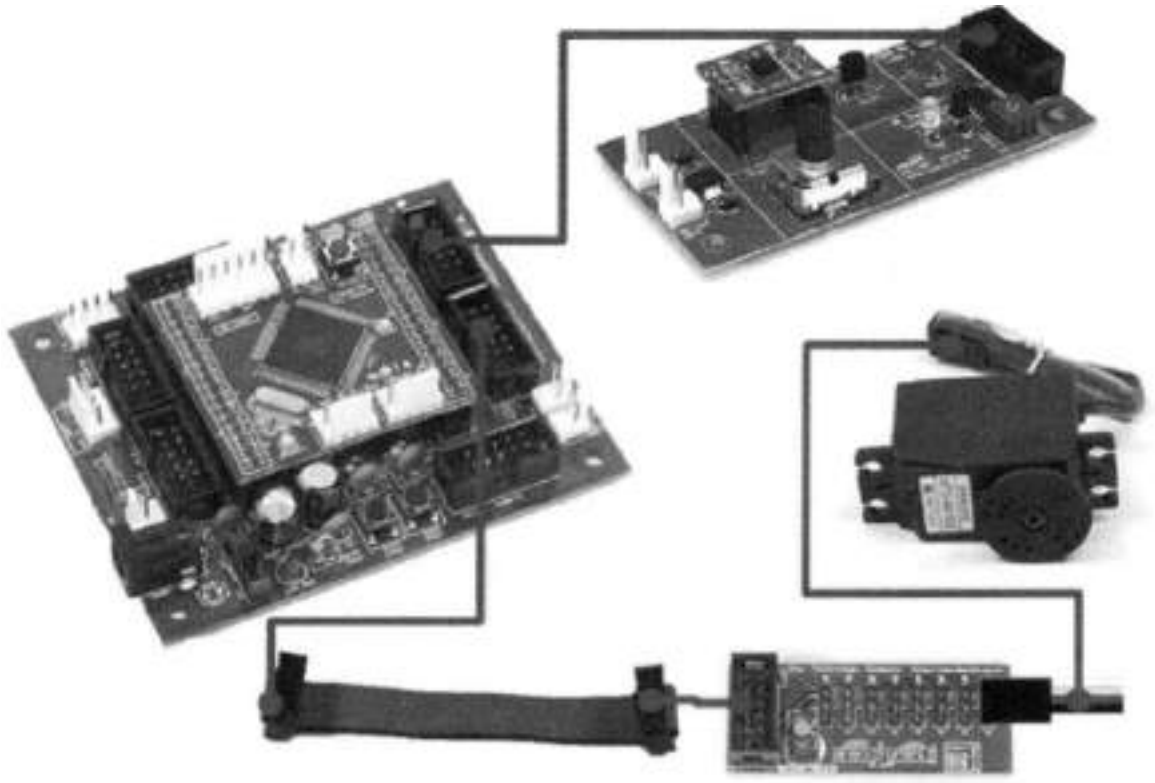
1) Timer()와 가변 저항을 이용한 Servo Motor 구동 하드웨어 연결

① PWM input port를 PORTB의 OC0(PORTB.4)에 연결

② 5V, GND 각각 연결

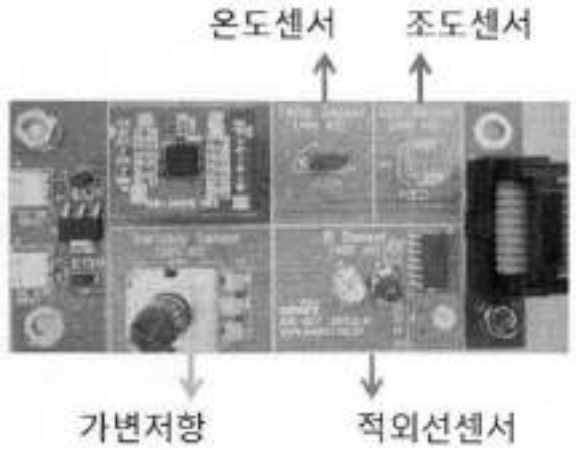
③ PORTF에 센서 모듈 (가변 저항) 을 연결

④ 가변 저항을 돌리면 Servo Motor가 함께 연동되어 구동



7.2 RC Servo Motor 이론 및 실습

실습 7. Timer0와 가변 저항을 이용한 Servo motor 구동제어 실습



2) 센서 모듈 (가변 저항)의 ADC 설정

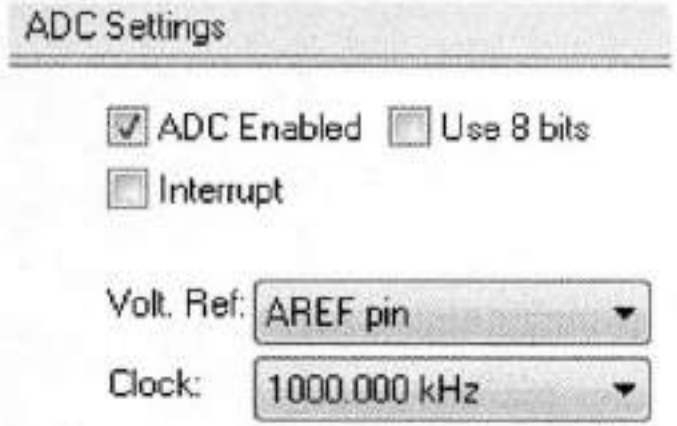
① 0 ~ 180 회전 PWM 파형 + Duty Pulse Width 0.7 ~ 2.3ms
② Timer0 Overflow Interrupt
③ 20ms PWM 0.1ms 단위로 제어 : 0.7 ~ 2.3ms (16 step)
④ 20ms = 20,000us => 10us 단위로 제어

700us ~ 2300 us (160 step)

⑤ 주파수 분주비 1을 사용할 경우

1 clk 당 소요 시간 1/16000000 = 0.0625
0.0625 * 16 = 1us : Timer가 16번 count에 1us 소요
160번 카운터 10us 소요
255 - 160 + 1 = 96 (0x60) 부터 160번 카운트하면 159번째 카운터 값은 255, 160번째 Overflow interrupt 발생

1번핀	3번핀	5번핀	7번핀	9번핀
ADC A0-X out	ADC A2-Y out	ADC A4-Z out	IR거리 (ADC-A6)	VCC
2번핀	**4번핀**	**6번핀**	**8번핀**	**10번핀**
가변저항 (ADC A1) PORTF.1	온도 (ADC A3)	CDS조도 (ADC-A5)	N.C	GND



센서 모듈의 가변 저항 ADC 설정
(AVCC pin 사용, PORTF.1 Enable)

7.2 RC Servo Motor 이론 및 실습

실습 7. Timer0와 가변 저항을 이용한 Servo motor 구동제어 실습

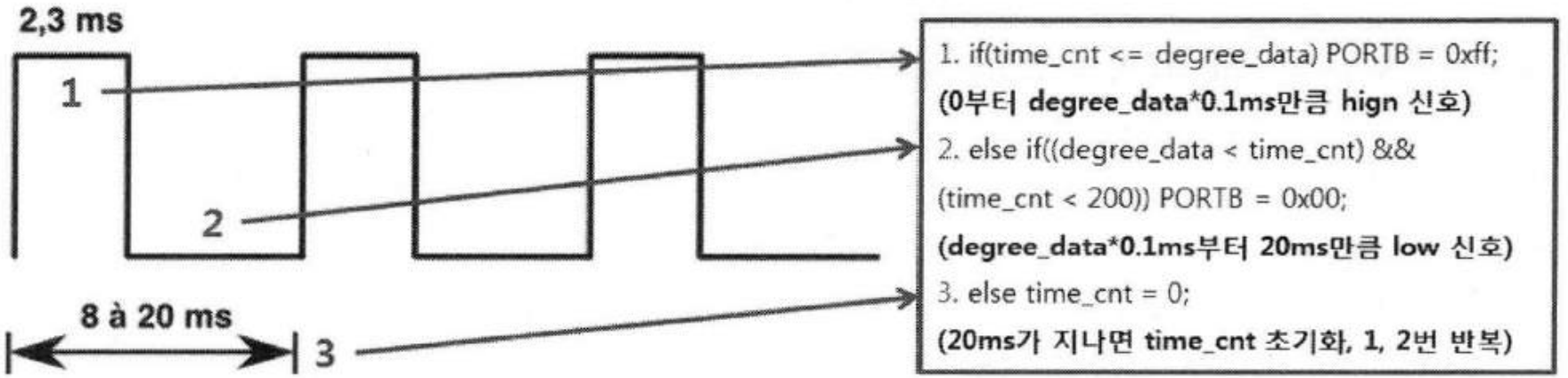
3) Timer0 (64분주) 로 0.1ms를 만든 후 펄스 설정

```
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
    TCNT0=0xE7;
    time_cnt++;
    if(time_cnt <= degree_data) PORTB=0xff;
    else if((degree_data < time_cnt) && (time_cnt <200)) PORTB=0x00;
    else time_cnt=0;
}
```

Timer0 interrupt 설정

TCNT0 - 0xE7; => 64분주에서 Overflow가 0.1ms 간격으로 발생하기 위한 Timer0의 주기 시작 값

time_cnt++; => 0.1ms 간격으로 증가



7.2 RC Servo Motor 이론 및 실습

실습 7. Timer0와 가변 저항을 이용한 Servo motor 구동제어 실습

4) 프로그램 작성

```
#include <mega128.h>
#include <delay.h>

#define ADC_VREF_TYPE ((0<<REFS1) | (1<<REFS0) | (0<<ADLAR))

int time_cnt, adc_data;
float degree_data;

interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
    TCNT0=0xE7;
    time_cnt++;
    if(time_cnt <= degree_data) PORTB=0xff;
    else if((degree_data < time_cnt) && (time_cnt <200)) PORTB=0x00;
    else time_cnt=0;
}

unsigned int read_adc(unsigned char adc_input)
{
    ADMUX = adc_input | ADC_VREF_TYPE;
    delay_us(10);
    ADCSRA |=(1<<ADSC);
    while((ADCSRA & (1<<ADIF)) == 0);
    ADCSRA |= (1<<ADIF);
    return ADCW;
}
```

7.2 RC Servo Motor 이론 및 실습

실습 7. Timer0와 가변 저항을 이용한 Servo motor 구동제어 실습

4) 프로그램 작성

```
void main(void)
{
    DDRB=0x10;
    PORTB=0x00;

    DDRF=0x00;
    PORTF=0x00;

    ASSR=0x00;
    TCCR0=0x04;
    TCNT0=0xE7;

    TIMSK=0x01;

    ADMUX=ADC_VREF_TYPE;
    ADCSRA=0x84;
    SFIOR=(0<<ACME);

    #asm("sei")

    while(1)
    {
        adc_data = read_adc(1);
        degree_data = (float)adc_data / 1023 * 16+7;
    }
}
```

adc_data 값의 범위 : 0 ~ 1023

-> degree_data 시간의 범위 : 7 ~ 23(0.7ms ~ 2.3ms)

degree_data = 16 / 1023 * adc_data + x ->

각각의 값을 대입해보면 x = 7

degree_data = adc_data / 1023 * 16 + 7 식이 등장한다.

(degree_data의 값은 실수 자료형 float 사용)

7.2 RC Servo Motor 이론 및 실습

[과제 9] Timer1과 가변 저항을 이용한 RC Servo motor의 PWM 제어 (Code Wizard 사용)

1) Pulse Width Modulation

- Timer/Counter0 ~ 3에서 사용
- 할 수 있고, Timer/Counter Control Register의 설정으로 PWM을 Enable 할 수 있음
- 출력은 Ocnx 핀으로 가능
- 현 과제에서는 Timer1을 이용한 16bit Fast PWM을 이용

	Timer/Counter0	Timer/Counter1	Timer/Counter2	Timer/Counter3
동작모드	Normal CTC Fast PWM Phase Correct PWM	Normal CTC Fast PWM Phase Correct PWM Phase and Frequency Correct PWM	Normal CTC Fast PWM Phase Correct PWM	Normal CTC Fast PWM Phase Correct PWM Phase and Frequency Correct PWM
출력신호	OC0	OC1A OC1B OC1C	OC2	OC3A OC3B OC3C

아날로그핀을 PORTB.5로 바꿔주어야 함

7.2 RC Servo Motor 이론 및 실습

[과제 9] Timer1과 가변 저항을 이용한 RC Servo motor의 PWM 제어 (Code Wizard 사용)

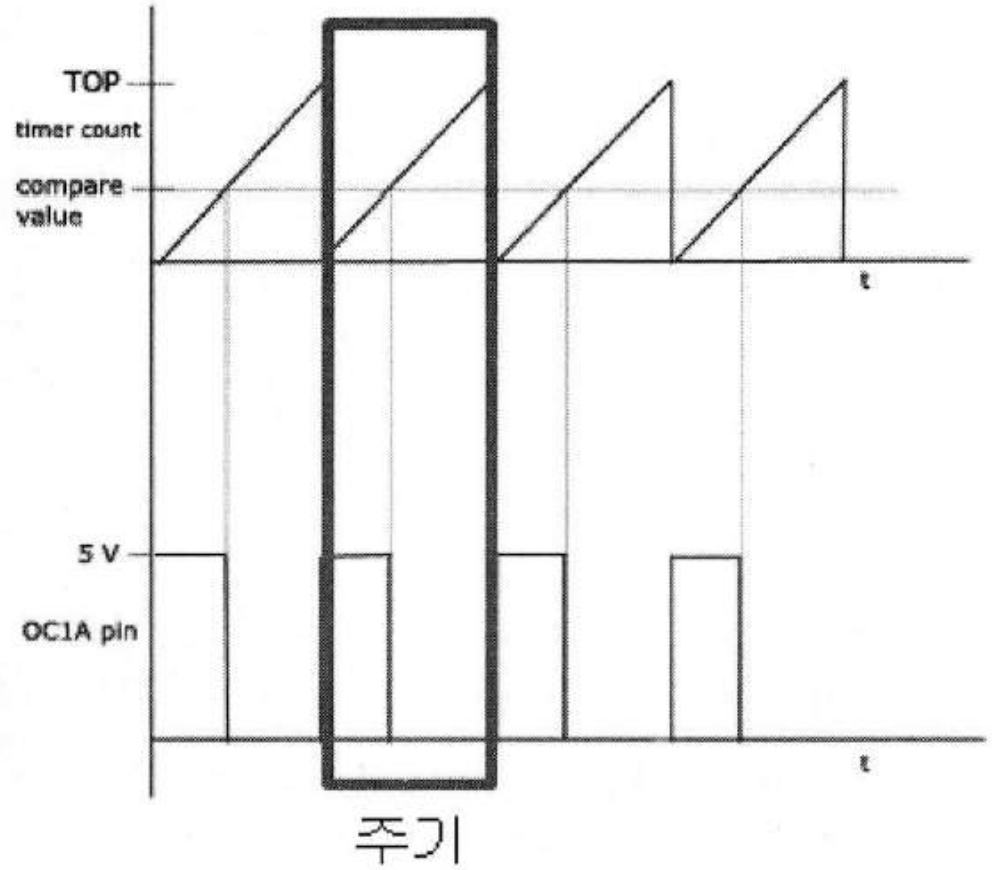
2) ATmega128의 Fast PWM 출력 원리

- TOP 값 : TCNT의 최대값 / PWM 주기를 결정 (ICRn 값)
- OCRn 값 : TCNT와 같아질 때 신호 반전 / PWM 듀티비를 결정
- 출력 원리

① TCNT 증가 : High 출력 / OCR 값과 같아지면 Low 출력

② TCNT 최대값 도달 (TOP 값) : TCNT = 0 (Overflow)
⇒ High 출력

③ 1번과 2번 반복



7.2 RC Servo Motor 이론 및 실습

[과제 9] Timer1과 가변 저항을 이용한 RC Servo motor의 PWM 제어 (Code Wizard 사용)

3) Servo Motor의 PWM 제어 (Fast PWM)

ICR1(TOP 값)은 다음과 같다. (20ms를 만들기 위해 16bit 사용)

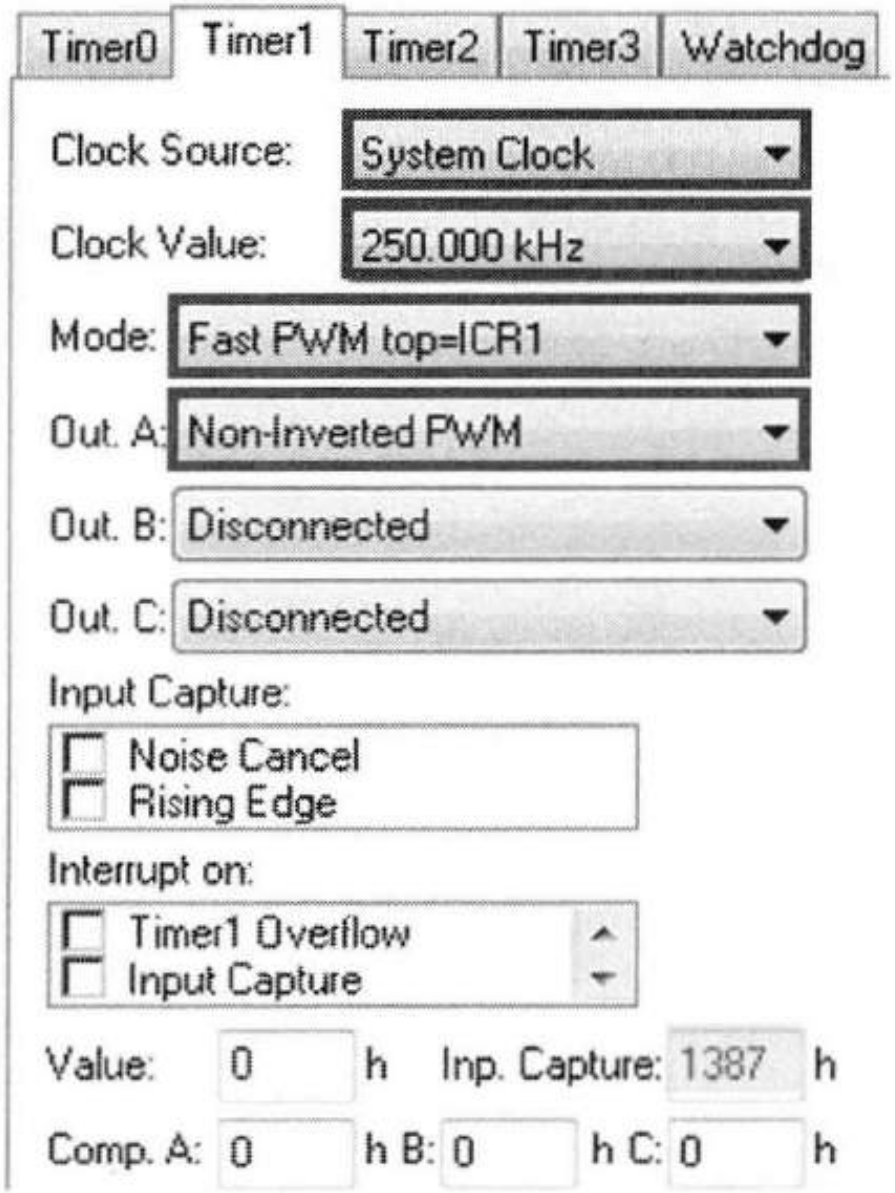
TOP 값 = Prescaler Clock(250kHz) / 원하는 주파수 – 1

원하는 주기 = 20ms -> 원하는 주파수 = 1 / 20ms = 0.05kHz

TOP값 = (250kHz / 0.05kHz) – 1 = 5000 – 1 = 4999 = 0x1387

```
TCCR1A = (1<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) |
         (0<<COM1C1) | (0<<COM1C0) | (1<<WGM11) | (1<<WGM10)

TCCR1B = (0<<ICNC1) | (0<<ICES1) | (1<<WGM13) | (1<<WGM12) |
         (0<<CS12) | (0<<CS11) | (1<<CS10);

ICR1H = 0x13;
ICR1L = 0x87;
```

Timer Counter Control Register1(TCCR1)의 레지스터 설정



7.2 RC Servo Motor 이론 및 실습

[과제 9] Timer1과 가변 저항을 이용한 RC Servo motor의 PWM 제어 (Code Wizard 사용)

4) 프로그램 작성

```
#include <io.h>

#include <delay.h>

// Declare your global variables here

// Voltage Reference: AREF pin
#define ADC_VREF_TYPE ((0<<REFS1) | (1<<REFS0) | (0<<ADLAR))

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
    ADMUX=adc_input | ADC_VREF_TYPE;
    // Delay needed for the stabilization of the ADC input voltage
    delay_us(10);
    // Start the AD conversion
    ADCSRA|=(1<<ADSC);
    // Wait for the AD conversion to complete
    while ((ADCSRA & (1<<ADIF))==0);
    ADCSRA|=(1<<ADIF);
    return ADCW;
}
```

7.2 RC Servo Motor 이론 및 실습

[과제 9] Timer1과 가변 저항을 이용한 RC Servo motor의 PWM 제어 (Code Wizard 사용)

4) 프로그램 작성

```
void main(void)
{
    // Declare your local variables here

    // Input/Output Ports initialization
    // Port B initialization
    // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
    DDRB=(0<<DDB7) | (0<<DDB6) | (1<<DDB5) | (0<<DDB4) |
         (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (0<<DDB0); /* 0x20*/
    // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
    PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) |
          (0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0); /*0x00*/

    // Port F initialization
    // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
    DDRF=(0<<DDF7) | (0<<DDF6) | (0<<DDF5) | (0<<DDF4) |
         (0<<DDF3) | (0<<DDF2) | (0<<DDF1) | (0<<DDF0); /*0x00*/
    // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
    PORTF=(0<<PORTF7) | (0<<PORTF6) | (0<<PORTF5) | (0<<PORTF4) |
          (0<<PORTF3) | (0<<PORTF2) | (0<<PORTF1) | (0<<PORTF0); /*0x00*/
    TCCR1A=(1<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) |
           (0<<COM1C1) | (0<<COM1C0) | (1<<WGM11) | (1<<WGM10); /*0x83*/
    TCCR1B=(0<<ICNC1) | (0<<ICES1) | (1<<WGM13) | (1<<WGM12) |
           (0<<CS12) | (0<<CS11) | (1<<CS10) /*0x19*/;
```

```
    TCNT1H=0x00;
    TCNT1L=0x00;
    ICR1H=0x13;
    ICR1L=0x87;
    // Timer(s)/Counter(s) Interrupt(s) initialization
    TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) |
          (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0); /*0x00*/
    ETIMSK=(0<<TICIE3) | (0<<OCIE3A) | (0<<OCIE3B) |
           (0<<TOIE3) | (0<<OCIE3C) | (0<<OCIE1C); /*0x00*/
    // ADC Voltage Reference: AREF pin
    ADMUX=ADC_VREF_TYPE;
    ADCSRA=(1<<ADEN) | (0<<ADSC) | (0<<ADFR) | (0<<ADIF) |
           (0<<ADIE) | (1<<ADPS2) | (0<<ADPS1) | (0<<ADPS0); /*0x84*/
    SFIOR=(0<<ACME);
    while (1)
    {
        adc_data = read_adc(1);
        degree_data = (unsigned int)((float)adc_data/1023 * 400 + 175);
        OCR1AH = degree_data >> 8;
        OCR1AL = degree_data & 0x00FF;
    }
}
```

adc_data 값의 범위 : 0 ~ 1023 -> OCR1 값 범위 : 175(0.7ms) ~ 575(2.3ms)

degree_data = 400 / 1023 * adc_data + x -> 각각의 값을 대입해보면 x = 175

degree_data = adc_data / 1023 * 400 + 175 식이 등장한다. (degree_data의 값은 연산자 사용이 가능한 unsigned int 범위값을 사용)