DC MOTOR

- DC Motor의 개요 및 원리
- DC Motor의 속도 제어 방법
- DC Motor 정회전, 역회전 시키기
- DC Motor 정지 시키기
- PWM을 이용하여 DC 모터 속도 제어하기 1
- PWM을 이용하여 DC 모터 속도 제어하기 2
- 인터럽트로 DC모터 동작 및 가변저항 값에 따라 속도 제어하기

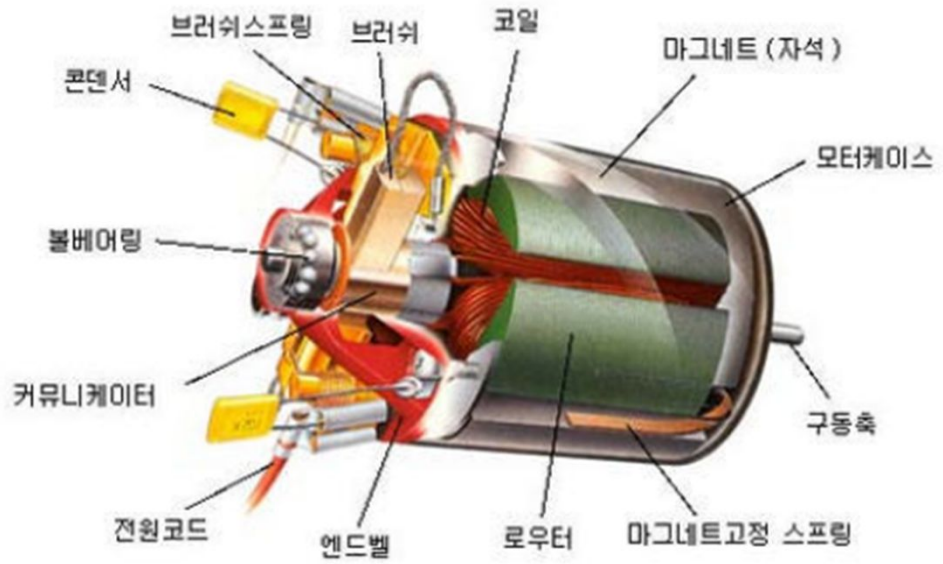
엣지아이랩

DC 모터 개요 및 원리

- DC 모터 개요
  - DC 모터란?
    - 고정자로 영구자석을 사용하고, 회전자(전기자)로 코일을 사용하여 구성한 것으로, 전기자에 흐르는 전류의 방향을 전환함으로써 자력의 반발, 흡인력으로 회전력을 생성시키는 모터
  - 특징
    - 기동 토크가 큼
    - 인가 전압에 대하여 회전 특성이 직선적으로 비례
    - 입력 전류에 대하여 출력 토크가 직선적으로 비례하며, 출력 효율이 양호
    - 저렴한 가격
    - 제어회로가 단순하고 제어하기 쉬움
  - 구조상 브러시(brush)와 정류자(commutator)에 의한 기계식 접점을 가지고 있어, 전기 불꽃(spark), 회전 소음, 수명 단축 등의 단점

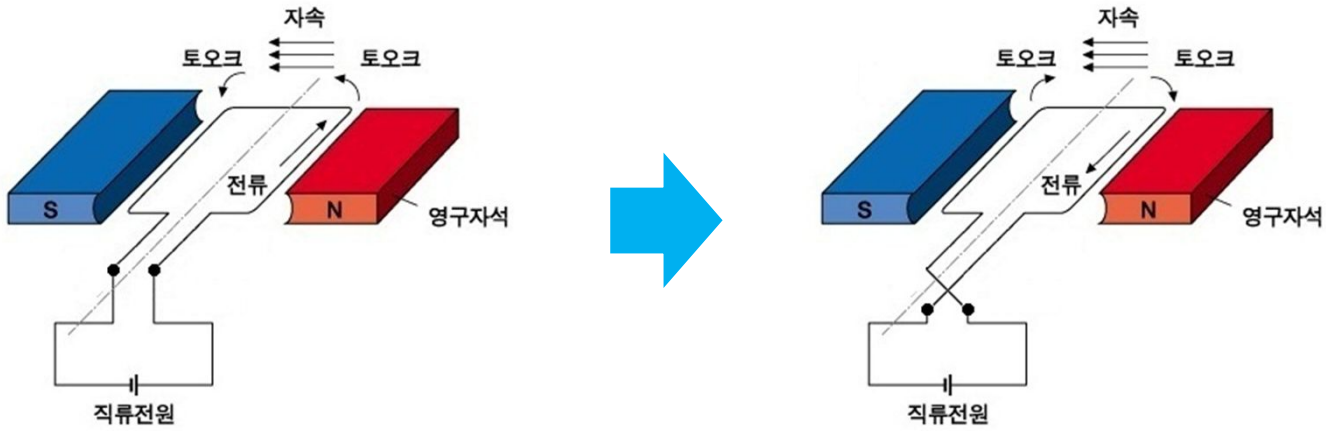
DC 모터 개요 및 원리

- DC 모터의 구조
  - DC모터의 구조는 크게 전기자(armature, 로터), 영구자석(permanent magnet, 계자용 마그넷), 브러시(brush), 베어링 모터 케이스 등으로 이루어져 있음



DC 모터 개요 및 원리

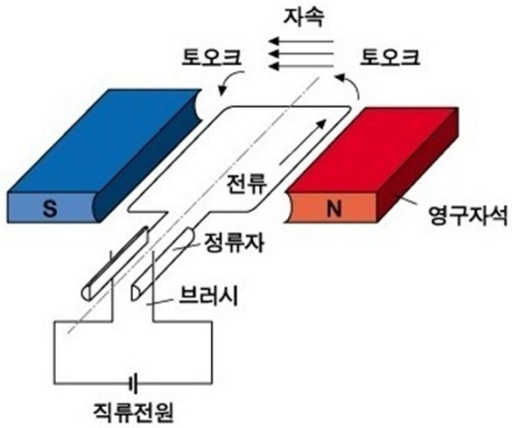
- DC 모터 구동 원리
  - 플래밍의 왼손법칙에 의해 전류가 흐르는 도선은 시계 반대 방향으로 회전

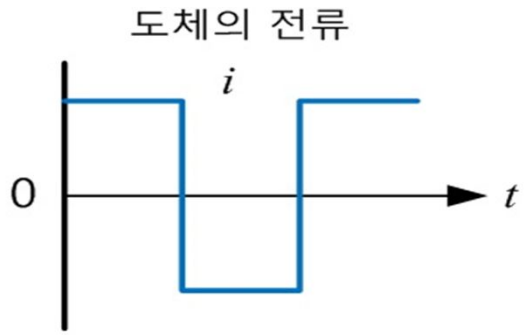


  - 그러나 도선이 회전하여 왼쪽 그림과 같이 위치하게 되면 시계 방향으로 회전하려는 힘 때문에 도선은 초기 위치로 돌아가게 됨

DC 모터 개요 및 원리

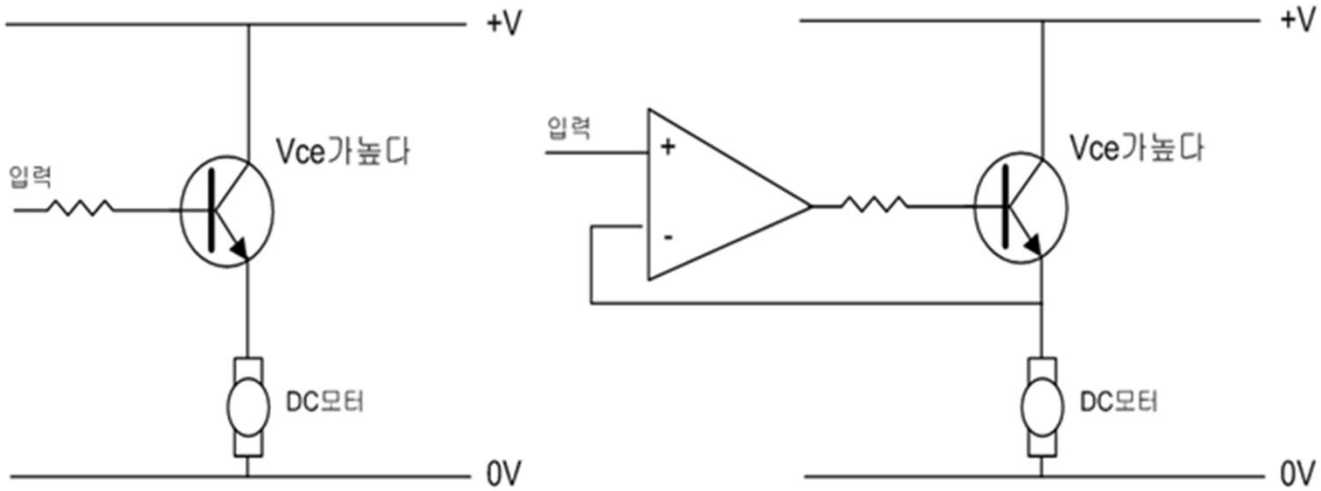
- DC 모터 구동 원리
  - 도선에 전류를 흘려 주기 위한 브러시(Brush)와 정류자(Commutator)가 추가된 그림을 보면, 플래밍의 왼손 법칙에 의해 도선은 시계 반대 방향으로 회전하게 되지만, 도선이 회전하더라도 정류자와 브러시에 의해 도선에 흐르는 전류는 뒤바뀌지 않게 되어 한쪽방향으로 연속해서 회전할 수 있게 됨





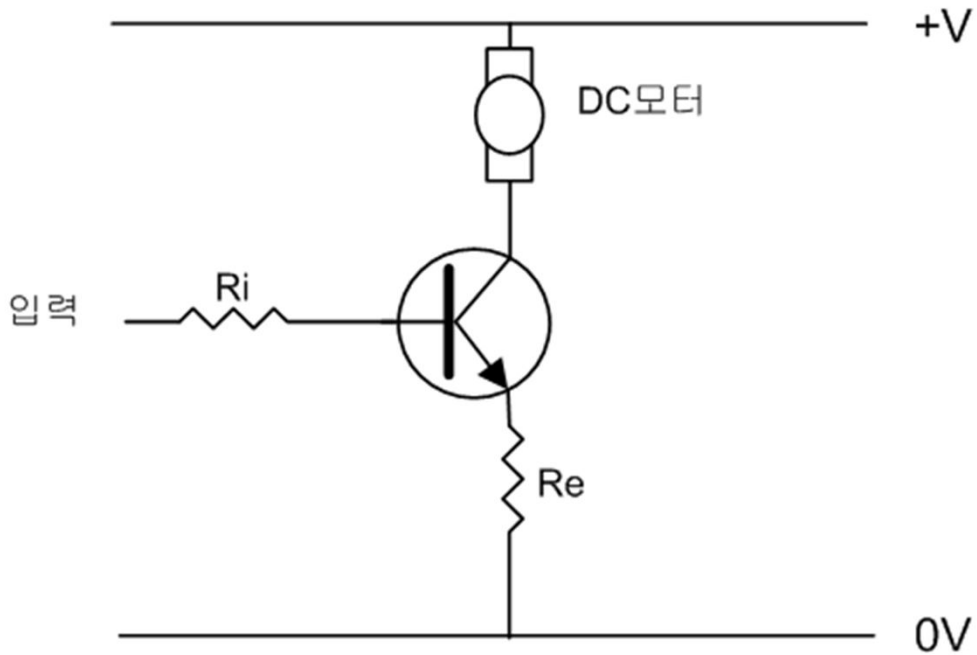
DC 모터 속도 제어 방법

- 트랜지스터 구동(이미터 부하)
  - 아래 [그림]의 회로에 의해 트랜지스터를 On/Off함으로써 모터를 On/Off
  - 이 회로는 트랜지스터가 완전히 포화 되는 On 상태로는 할 수 없고, Vce가 크기 때문에 전압 손실이 커지게 됨. 동작으로서는 자동적으로 부귀환이 동작하기 때문에 동작은 안정적
  - 이 때문에 간단한 속도 제어를 하기 위해 OP 앰프를 추가한 회로가 사용
    - 이 경우, 트랜지스터에서의 전력손실이 그대로 열로 되기 때문에 트랜지스터의 열 대책 고려



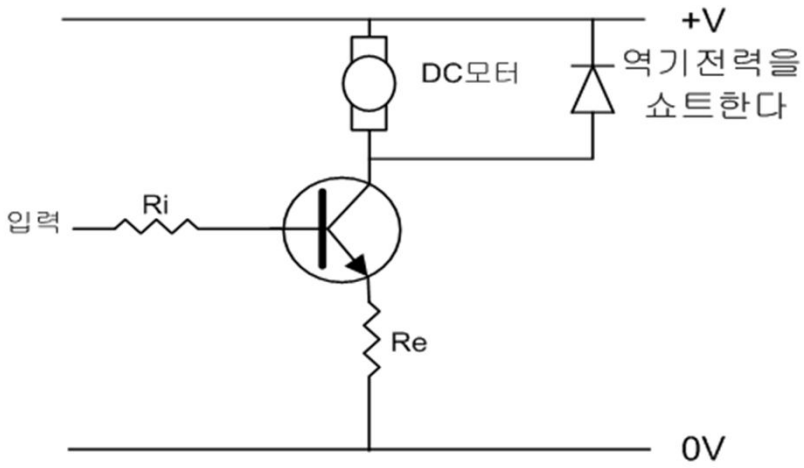
DC 모터 속도 제어 방법

- 트랜지스터 구동(컬렉터 부하)
  - 모터를 트랜지스터 컬렉터의 부하로 이용한 것으로, 트랜지스터가 완전히 포화된 On 상태로 구동할 수 있기 때문에 드라이브 능력이 크고 전압 손실도 적게 할 수 있음
    - 일반적으로는 이 회로가 많이 사용



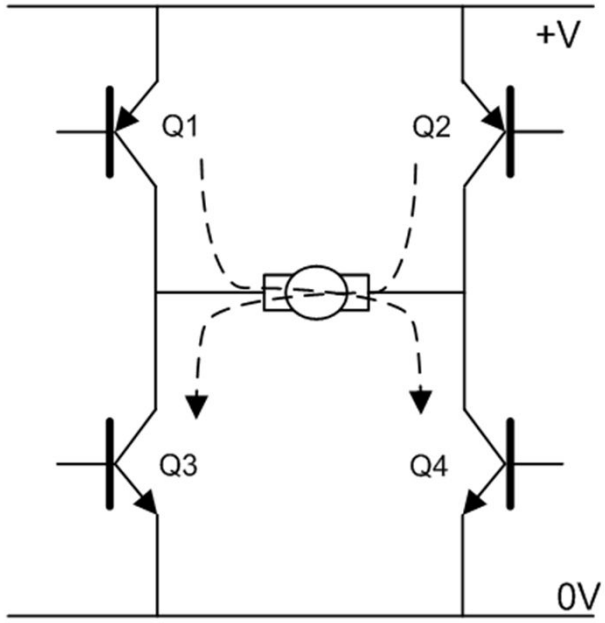
DC 모터 속도 제어 방법

- 역기전력의 처리
  - 모터가 회전하고 있는 동안에는 모터의 코일에 에너지가 축적되고, 트랜지스터가 Off로 되면 그 에너지를 방출하려고 하기 때문에, 모터 코일의 양단에는 플러스, 마이너스가 역방향의 기전력이 발생
    - 이 전압은 매우 크기 때문에 그대로는 트랜지스터가 파괴되어 버리는 경우도 있음
  - 모터 양단에 다이오드를 달아서 코일을 쇼트시켜 남아 있는 에너지를 순간적으로 전류로서 흘려 버리는 식으로 해서, 역기전력을 억제하도록 함



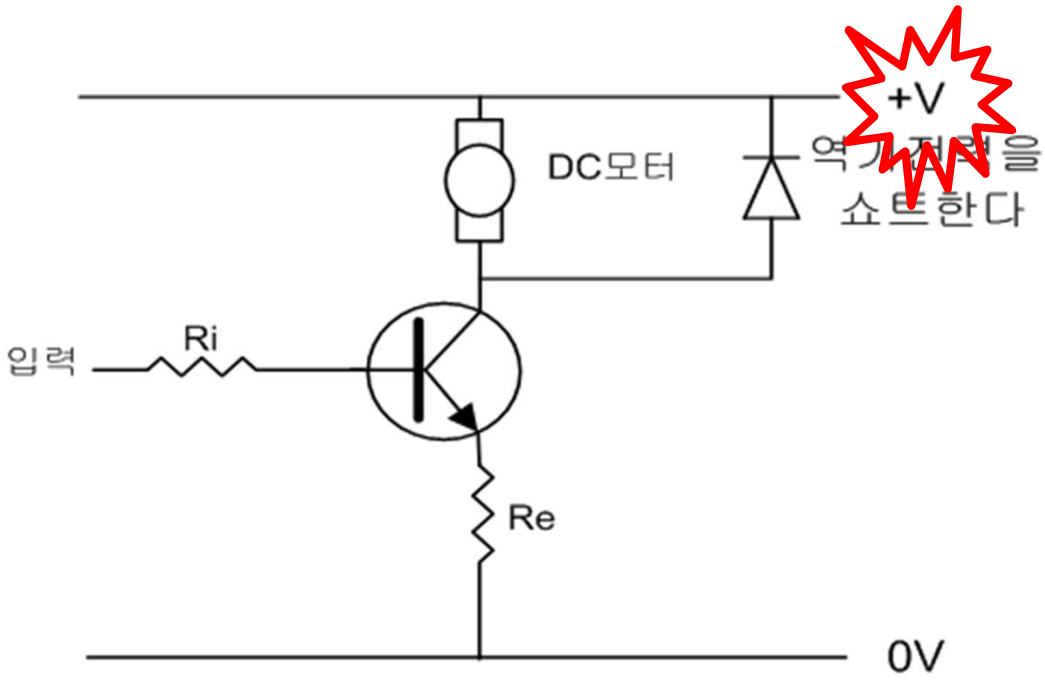
DC 모터 속도 제어 방법

- 회전 방향을 바꾸는 H 브리지 제어회로
  - 단일전원으로 모터에 가하는 전압의 방향을 바꿀 수 있는 회로로 고안된 것이 "H 브리지 회로"
    - 모터는 정회전
      - Q1과 Q4의 트랜지스터만 동시에 On
    - 모터는 역회전
      - Q2와 Q3만 트랜지스터만 동시에 On



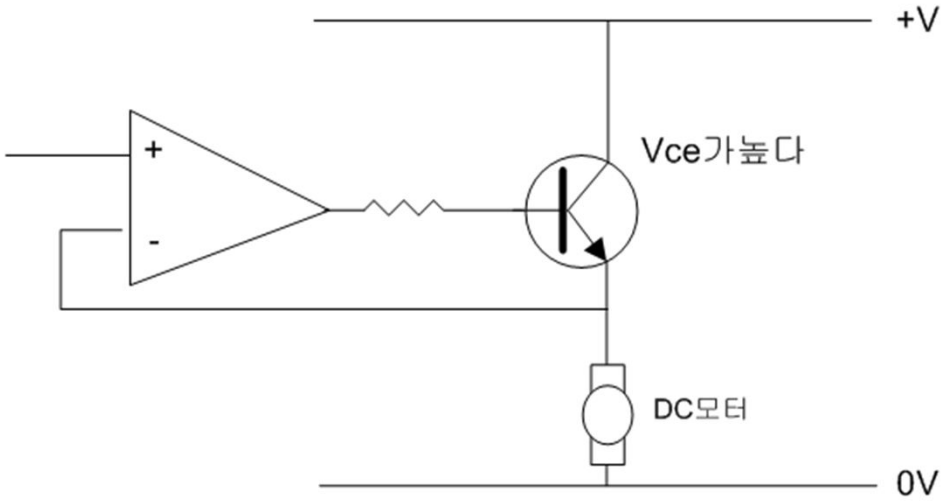
DC 모터 속도 제어 방법

- DC 모터의 가변속 제어법
  - DC 모터의 속도를 연속적으로 바꾸려는 경우에는 기본적으로는 DC 모터에 가하는 전압을 바꾸면 속도는 변함
  - 단순히 모터의 코일에 흐르는 전류와 속도가 정비례하기 때문에, 모터의 구동 전압 +V를 변화시키면 속도도 변함



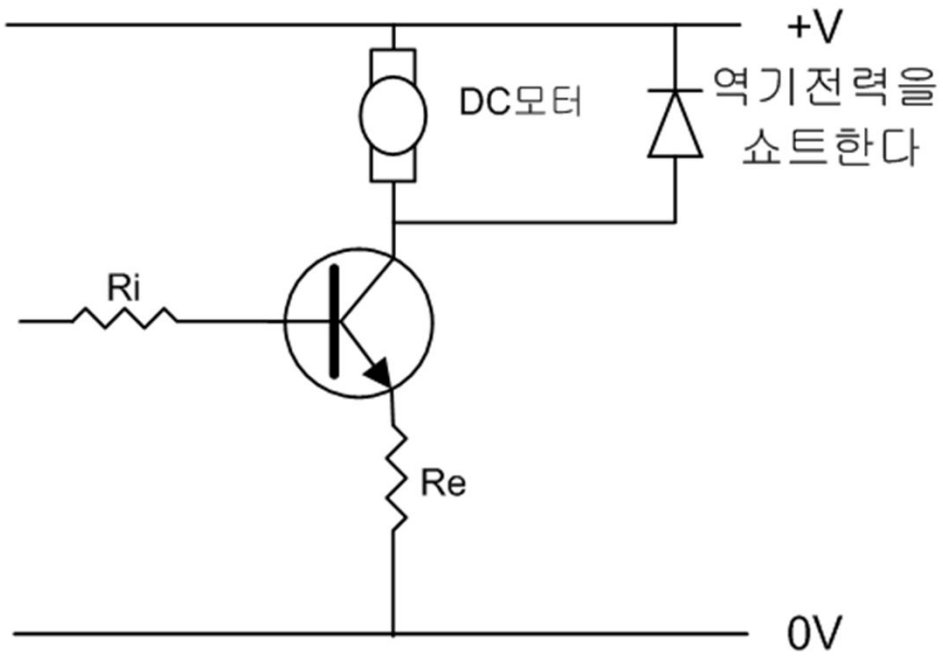
DC 모터 속도 제어 방법

- 구동 전압을 변화시키는 방법으로는 아날로그 방식과 펄스폭 변조방식이 있음
  - 아날로그 방식의 가변속 제어
    - 직접 구동 전압을 변화시키는 것으로, 기본회로는 다음과 같음
      - 트랜지스터로 전압 dropper를 구성하고, 컬렉터 이미터간의 드롭 전압을 바꿈으로써 모터에 가해지는 구동 전압을 가변



DC 모터 속도 제어 방법

- 펄스폭 변조(PWM : Pulse Width Modulation)
  - PWM 방식는 구동전압을 바꾸고 있는 것과 같은 효과를 내고 있지만, 그 방법이 펄스폭에 따르고 있으므로 펄스폭 변조(PWM : Pulse Width Modulation)라 부름
    - 펄스의 duty비(On 시간과 Off 시간의 비)를 바꿈으로써 실현

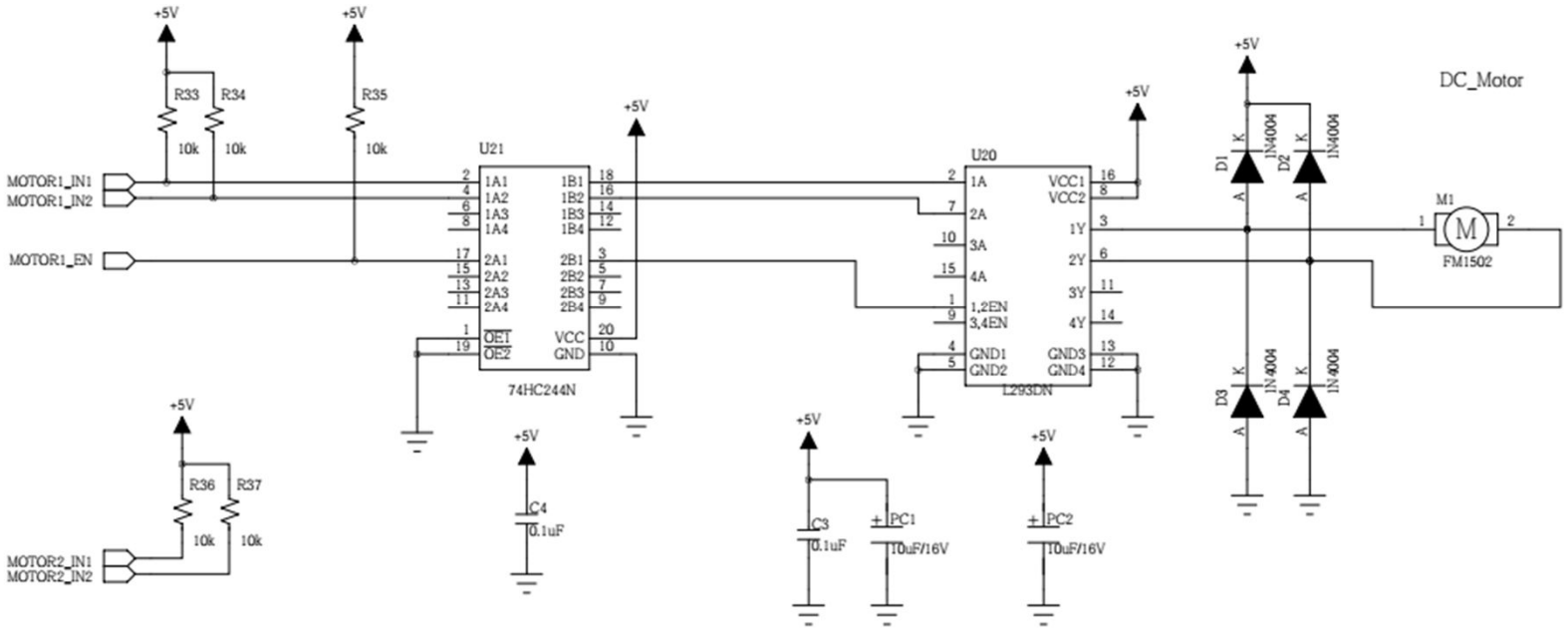


DC 모터 사양과 제어 회로

- 실습에 사용하는 모델은 FM1502
  - 정격 전압 : DC 5V
  - 정격 토크 : 3.54g·cm
  - 정격 회전수 : 8,670rpm
  - 정격 전류 : 90mA 이하
  - 무부하 회전수 : 10,860rpm
  - 무부하 전류 : 20mA 이하

DC 모터 사양과 제어 회로

- DC 모터 회로도
  - L293DN – DC 모터 드라이버 IC, 2 채널
  - 방향제어 - MOTOR1_IN1, MOTOR1_IN2,
    MOTOR2_IN1(사용안함), MOTOR2_IN2(사용안함)
  - 속도제어 - MOTOR1_EN, MOTOR2_EN(사용안함)



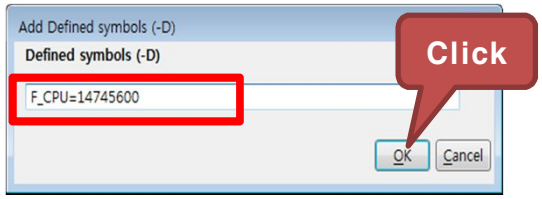
실습 1 : DC 모터 정회전, 역회전 시키기

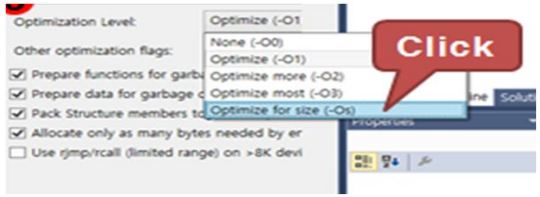
- 실습 개요
  - DC Motor를 정회전 시켰다가 다시 역회전 시키기
  - H-Bridge 회로가 내장되어 있는 L293DN 칩을 사용

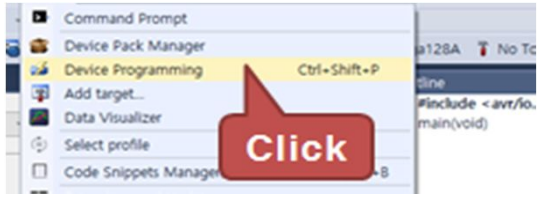
- 실습 목표
  - ATmega128A의 GPIO 포트에 대한 프로그램 방법 습득
  - L293DN을 사용한 DC Motor 제어의 원리 이해

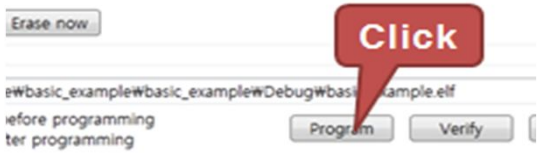
실습 1 : DC 모터 정회전, 역회전 시키기

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : 11_DCMOTOR_01, Location : D:₩AVR_Example
    - Device Selection : ATmega128A
  - 프로젝트 설정
    - Project 탭에서 "... Properties..." 선택
    - Toolchain -> AVR/GNU C Compiler에서
    - Symbols -> F_CPU=14745600 추가
    - Optimization -> Optimize for size (-OS) 선택
    - 저장 (Ctrl+S)
  - 소스코드 작성
  - 프로젝트 빌드
    - Build 탭에서 "Build Solution" 클릭
  - 프로그래밍
    - Tool 탭에서 "Device Programming" 클릭
    - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
    - 인식 완료되면, Memories 탭 선택, "Program" 클릭









실습 1 : DC 모터 정회전, 역회전 시키기

- DC 모터 제어 방법
  - Forward(정회전)는 시계방향 회전이며 Reverse(역회전)은 반시계방향 회전
  - Fast Motor Stop은 모터 양단에 같은 전압을 입력하여 급정지 시키는 것이며, Free Running Motor Stop은 모터 출력 enable 신호를 'disable' 시켜서 모터의 회전에 토크를 가하지 않아 서서히 정지하게되는 것을 뜻함

입 력		기 능
MOTOR1_EN = H	MOTOR1_IN1 = H MOTOR1_IN2 = L	Forward
	MOTOR1_IN1 = L MOTOR1_IN2 = H	Reverse
	MOTOR1_IN1 = MOTOR1_IN2	Fast Motor Stop
MOTOR1_EN = L	MOTOR1_IN1 = X MOTOR1_IN2 = X	Free Running Motor Stop

실습 1 : DC 모터 정회전, 역회전 시키기

edgeILAB

- DC 모터 방향
  - 정회전
    - MOTOR1_IN1(PD4) :'1'(High)
    - MOTOR1_IN2(PD5) :'0'(Low)
    - MOTOR1_EN(PB5) :'1'(High)

```
PORTD |= 0x10;
PORTD &= 0xDF;
PORTB |= 0x20;
```

  - 역회전
    - MOTOR1_IN1(PD4) :'0'(Low)
      MOTOR1_IN2(PD5) :'1'(High)
      MOTOR1_EN(PB5) :'1'(High)

```
PORTD &= 0xEF;
PORTD |= 0x20;
PORTB |= 0x20;
```

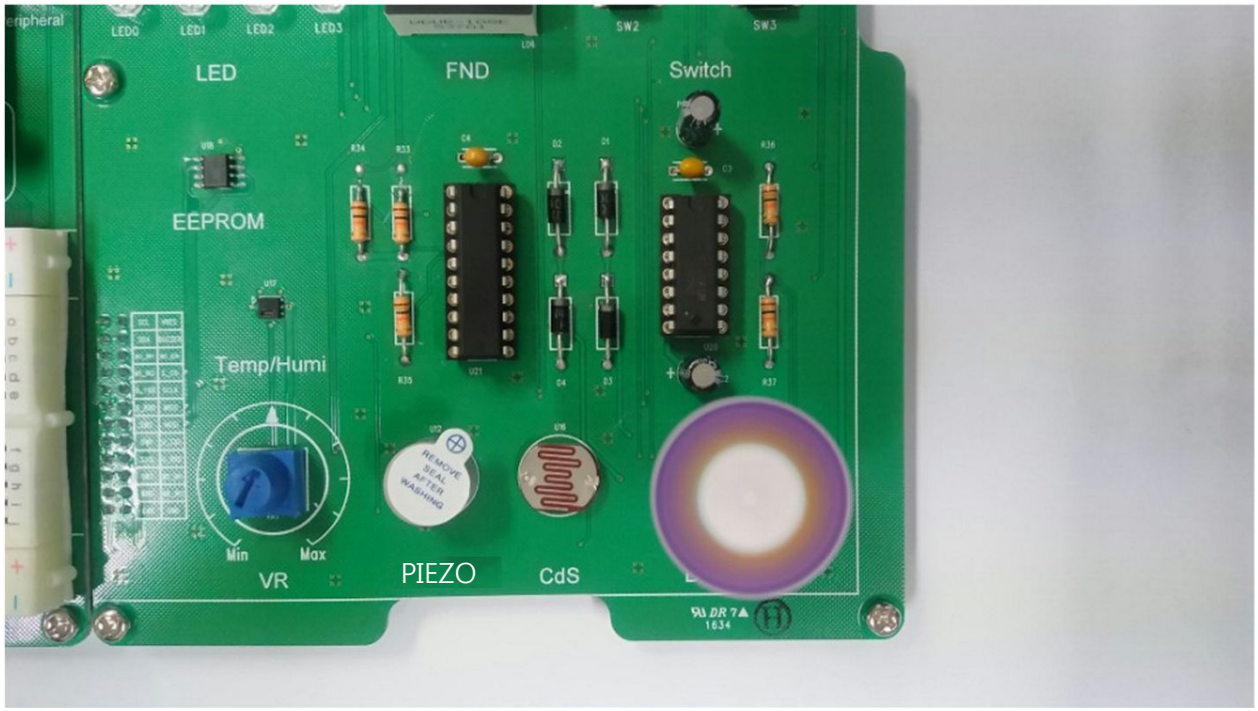
실습 1 : DC 모터 정회전, 역회전 시키기

- 구동 프로그램
  - main.c 코드 작성

```
#include <avr/io.h>      // AVR 입출력에 대한 헤더 파일
#include <util/delay.h> // delay 함수사용을 위한 헤더파일
#define M1_Forword 0x10
#define M1_Reverse 0x20
#define M1_Enable 0x20
int main(void) {
  DDRD = 0x30;  // MOTOR1_IN1, MOTOR1_IN2를 출력 포트로 설정
  DDRB = 0x20;  // MOTOR1_EN 를 출력 포트로 설정
  while (1)  {
    PORTD = M1_Forword; // DC Motor 정회전
    PORTB = M1_Enable;
    _delay_ms(1000);
    PORTD = M1_Reverse; // DC Motor 역회전
    PORTB = M1_Enable;
    _delay_ms(1000);
  }
}
```

실습 1 : DC 모터 정회전, 역회전 시키기

- 실행 결과
  - 모터가 반복적으로 정회전, 역회전함



실습 2 : DC 모터 정지 시키기

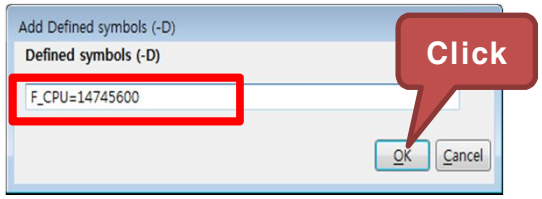
edgeILAB

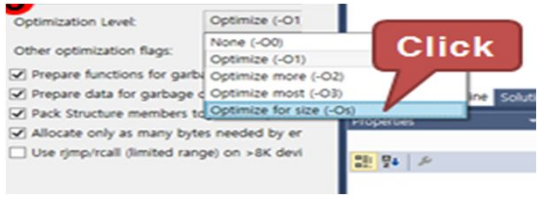
- 실습 개요
  - DC 모터의 정지 방법인 Fast Motor Stop과 Free Running Motor Stop의 차이점을 알아보는 실습
  - 스위치 입력을 통해 SW0' 키가 눌리면 Fast Motor Stop을, 'SW1' 키가 눌리면 Free Running Motor Stop을, 'SW2' 키가 눌리면 다시 모터가 정회전시키기

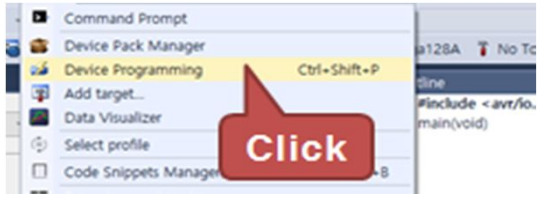
- 실습 목표
  - ATmega128A의 GPIO 포트에 대한 프로그램 방법 습득
  - L293DN을 사용한 DC Motor 제어의 원리 이해
  - Fast Motor Stop과 Free Running Motor Stop의 차이점 이해

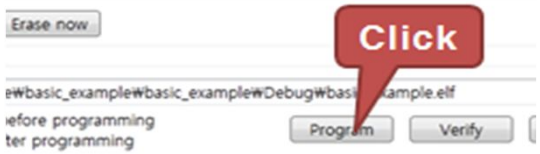
실습 2 : DC 모터 정지 시키기

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : 11_DCMOTOR_Example_02, Location : D:₩AVR_Example
    - Device Selection : ATmega128A
  - 프로젝트 설정
    - Project 탭에서 "... Properties..." 선택
    - Toolchain -> AVR/GNU C Compiler에서
    - Symbols -> F_CPU=14745600 추가
    - Optimization -> Optimize for size (-OS) 선택
    - 저장 (Ctrl+S)
  - 소스코드 작성
  - 프로젝트 빌드
    - Build 탭에서 "Build Solution" 클릭
  - 프로그래밍
    - Tool 탭에서 "Device Programming" 클릭
    - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
    - 인식 완료되면, Memories 탭 선택, "Program" 클릭









실습 2 : DC 모터 정지 시키기

- 구동 프로그램
  - main.c 코드 작성

```
#include <avr/io.h>      // AVR 입출력에 대한 헤더 파일
#include <util/delay.h> // delay 함수사용을 위한 헤더파일
#define M1_Forword 0x10
#define M1_Reverse 0x20
#define M1_Stop    0x30
#define M1_Enable  0x20

int main(void) {
  unsigned char key = 0;
  DDRD = 0x30;    // MOTOR1_IN1, MOTOR1_IN2를 출력 포트로 설정
  DDRB = 0x20;    // MOTOR1_EN 를 출력 포트로 설정
  DDRE = 0x00;    // 스위치 입력핀인 포트 E를 입력 포트로 설정

  // DC Motor 정회전
  PORTD = M1_Forword;
  PORTB = M1_Enable;
```

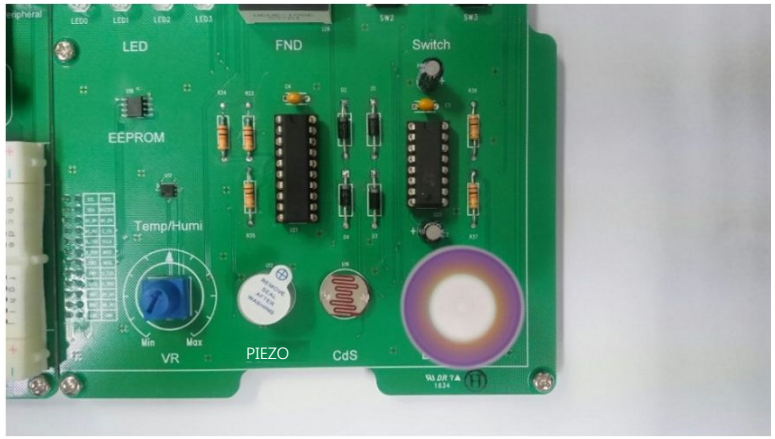
실습 2 : DC 모터 정지 시키기

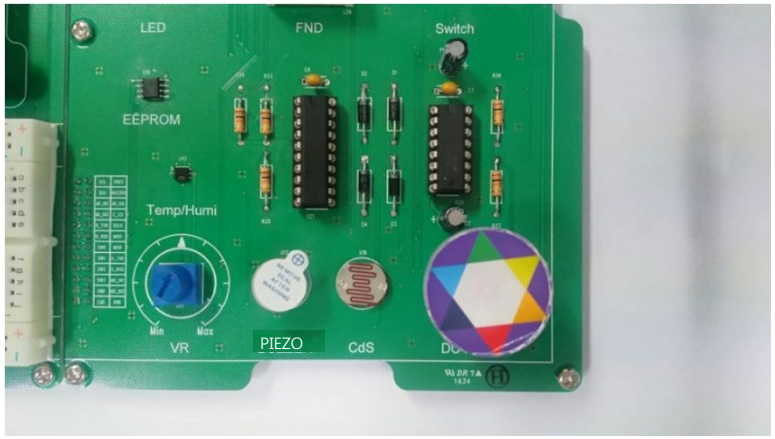
- 구동 프로그램
  - main.c 코드 작성

```
int main(void) {
  //...
  while (1) {
    key = PINE & 0xF0;  // E포트의 상위 4비트만 읽음
    if(key == 0x10) {   // '1' 키가 눌렸을때
      PORTD |= M1_Stop; // Motor1 정지, Fast Motor Stop
    }
    else if(key == 0x20) { //'2' 키가 눌렸을때
      PORTB &= ~M1_Enable; // Motor1 정지, Free Running Motor Stop
    }
    else if(key == 0x40) { // '3' 키가 눌렸을때
      PORTD = M1_Forword;  // Motor1 정회전
      PORTB = M1_Enable;
    }
  }
}
```

실습 2 : DC 모터 정지 시키기

- 실행 결과
  - 동작 버튼인 'SW2'를 눌러 모터를 회전 확인
  - 정지 버튼인 'SW0'과 'SW1'을 각각 눌러 정지 확인
  - 모터가 빠르게 정지할 때와 천천히 정지하는 것을 확인
  - 모터를 바로 정지하는 Fast Motor stop은 강제로 정지시키는 것이므로 순간 전류가 많이 필요하므로 유의해서 사용





실습 3 : PWM을 이용하여 DC 모터 속도 제어하기 1

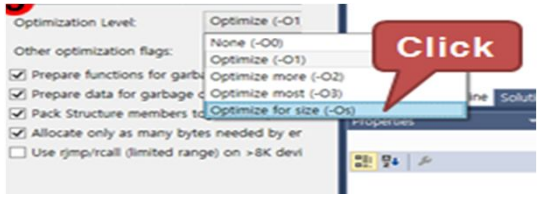
edgeILAB

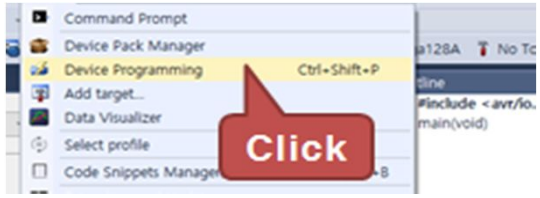
- 실습 개요
  - 타이머1의 PWM 기능을 이용하여 DC 모터 제어
  - Fast PWM, 모드 14를 이용하여 18.432KHz(70% 듀티비)의 PWM신호를 발생시켜 DC 모터 회전

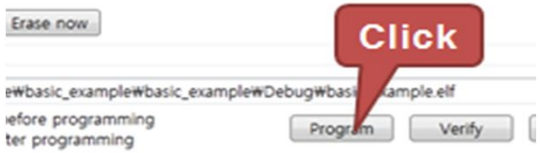
- 실습 목표
  - ATmega128A의 PWM 핀에 대한 프로그램 방법 습득
  - L293DN을 사용한 DC Motor 제어의 원리 이해
  - 타이머1의 PWM 기능 동작원리 이해

실습 3 : PWM을 이용하여 DC 모터 속도 제어하기 1

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : 11_DCPWM_Example_01, Location : D:₩AVR_Example
    - Device Selection : ATmega128A
  - 프로젝트 설정
    - Project 탭에서 "... Properties..." 선택
    - Toolchain -> AVR/GNU C Compiler에서
    - Symbols -> F_CPU=14745600 추가
    - Optimization -> Optimize for size (-OS) 선택
    - 저장 (Ctrl+S)
  - 소스코드 작성
  - 프로젝트 빌드
    - Build 탭에서 "Build Solution" 클릭
  - 프로그래밍
    - Tool 탭에서 "Device Programming" 클릭
    - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
    - 인식 완료되면, Memories 탭 선택, "Program" 클릭







실습 3 : PWM을 이용하여 DC 모터 속도 제어하기 1

edgeILAB

- 구동 프로그램 사전 지식
  - Fast PWM, 모드 14의 PWM

```
TCCR1A = 0x82;
TCCR1B = 0x19;
```

  - 18.432KHz의 주파수 → 14745600/18432 = 800

```
ICR1 = 800;          // 14745600/18432 => 800
```

  - 듀티비 70% → 800x70/100 = 560

```
OCR1A = 560;
```

실습 3 : PWM을 이용하여 DC 모터 속도 제어하기 1

edgeILAB

- 구동 프로그램
  - main.c 코드 작성

```
#include <avr/io.h>      // AVR 입출력에 대한 헤더 파일
#include <util/delay.h> // delay 함수사용을 위한 헤더파일
#define M1_Forword 0x10
#define M1_Reverse 0x20
#define M1_Enable  0x20
int main(void) {
  DDRD = 0x30;  // MOTOR1_IN1, MOTOR1_IN2를 출력 포트로 설정
  DDRB = 0x20;  // MOTOR1_EN 를 출력 포트로 설정

  TCCR1A=0x82;  // COMnA1=1,COMnA0=0,COMnB1=0,COMnB0=0,
                // COMnC1=0,COMnC0=0,WGMn1=1,WGMn0=0
  TCCR1B=0x19;  // ICNCn=0,ICESn=0,Blink=0,WGMn3=1
                // WGMn2=1,CSn2=0,CSn1=0,CSn0=1
  TCCR1C=0x00;
  ICR1=800;     // 14745600/800 => 18.432KHz
  OCR1A=560;    // 듀티비 70%
```

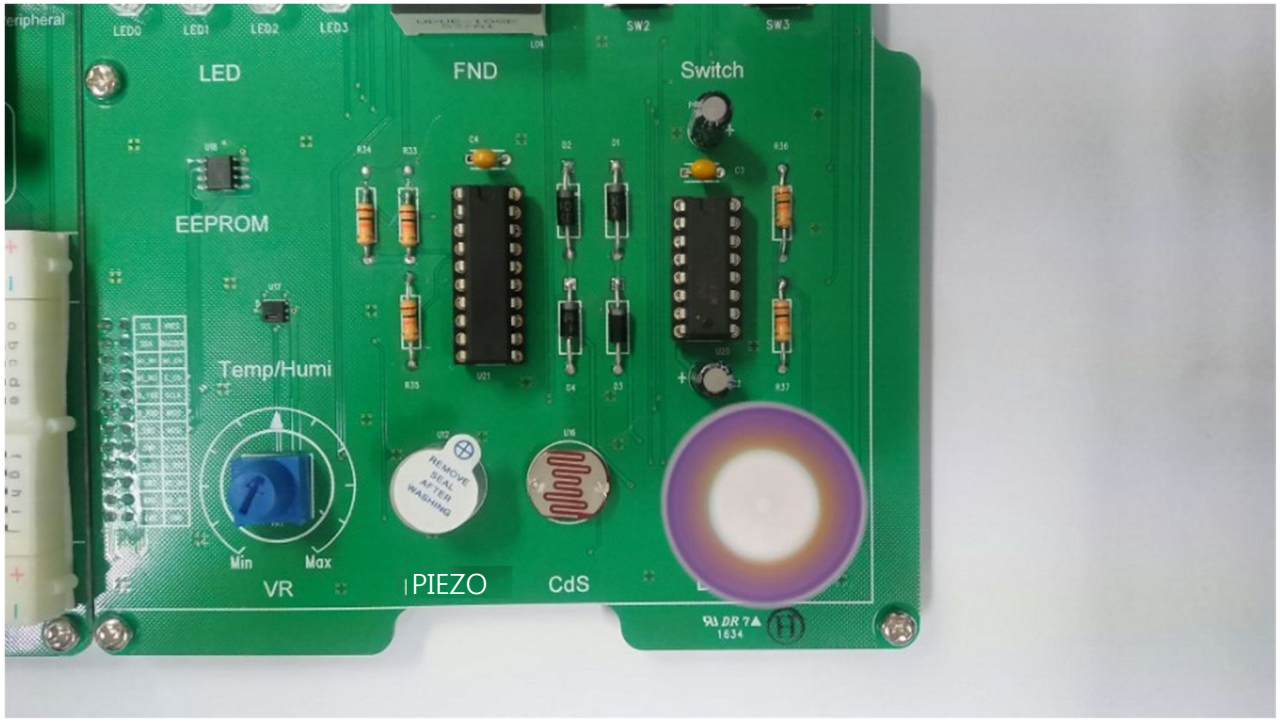
실습 3 : PWM을 이용하여 DC 모터 속도 제어하기 1

- 구동 프로그램
  - main.c 코드 작성

```
int main(void) {
  //...
  while (1)
  {
    // DC Motor 정회전
    PORTD = M1_Forword;
    _delay_ms(1000);
    // DC Motor 역회전
    PORTD = M1_Reverse;
    _delay_ms(1000);
  }
}
```

실습 3 : PWM을 이용하여 DC 모터 속도 제어하기 1

- 실행 결과
  - 모터가 반복해서 정회전, 역회전함
  - 실습1과 비교해서 회전속도가 느림



실습 4 : PWM을 이용하여 DC 모터 속도 제어하기 2

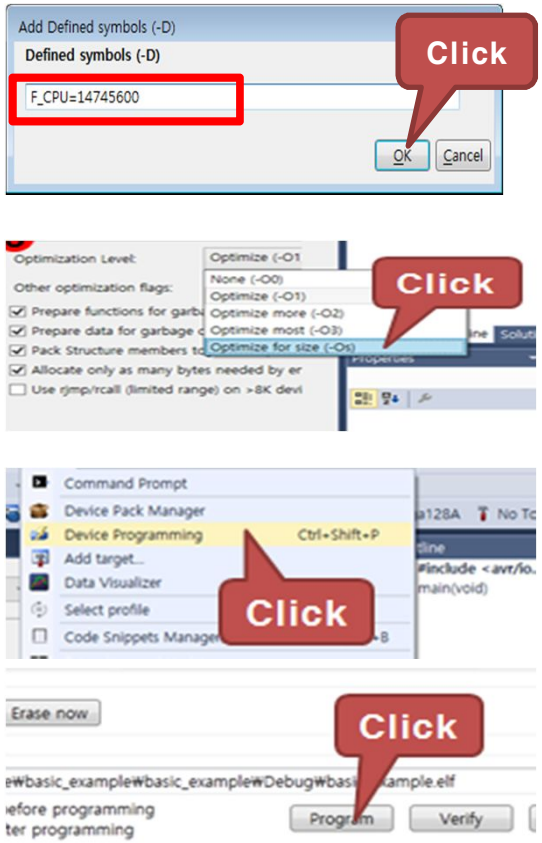
edgeILAB

- 실습 개요
  - 타이머1의 PWM 기능을 이용하여 DC 모터 제어
  - Fast PWM, 모드 14를 이용하여 18.432KHz(70% 듀티비)의 PWM신호를 발생시켜 DC 모터 회전
  - 시간에 따라 PWM 듀티비를 증가 및 감소시켜 DC모터 속도의 변화를 관찰
  - 변화하는 듀티비는 Text LCD에 출력

- 실습 목표
  - ATmega128A의 PWM 핀에 대한 프로그램 방법 습득
  - L293DN을 사용한 DC Motor 제어의 원리 이해
  - 타이머1의 PWM 기능 동작원리 이해
  - Text LCD의 사용법 이해

실습 4 : PWM을 이용하여 DC 모터 속도 제어하기 2

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : 11_DCPWM_Example_02, Location : D:₩AVR_Example
    - Device Selection : ATmega128A
  - 프로젝트 설정
    - Project 탭에서 "... Properties..." 선택
    - Toolchain -> AVR/GNU C Compiler에서
    - Symbols -> F_CPU=14745600 추가
    - Optimization -> Optimize for size (-OS) 선택
    - 저장 (Ctrl+S)
  - 소스코드 작성
  - 프로젝트 빌드
    - Build 탭에서 "Build Solution" 클릭
  - 프로그래밍
    - Tool 탭에서 "Device Programming" 클릭
    - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
    - 인식 완료되면, Memories 탭 선택, "Program" 클릭



실습 4 : PWM을 이용하여 DC 모터 속도 제어하기 2

- 구동 프로그램 사전 지식
  - Fast PWM, 모드 14의 PWM

```
TCCR1A = 0x82;
TCCR1B = 0x19;
```

  - 18.432KHz의 주파수 → 14745600/18432 = 800

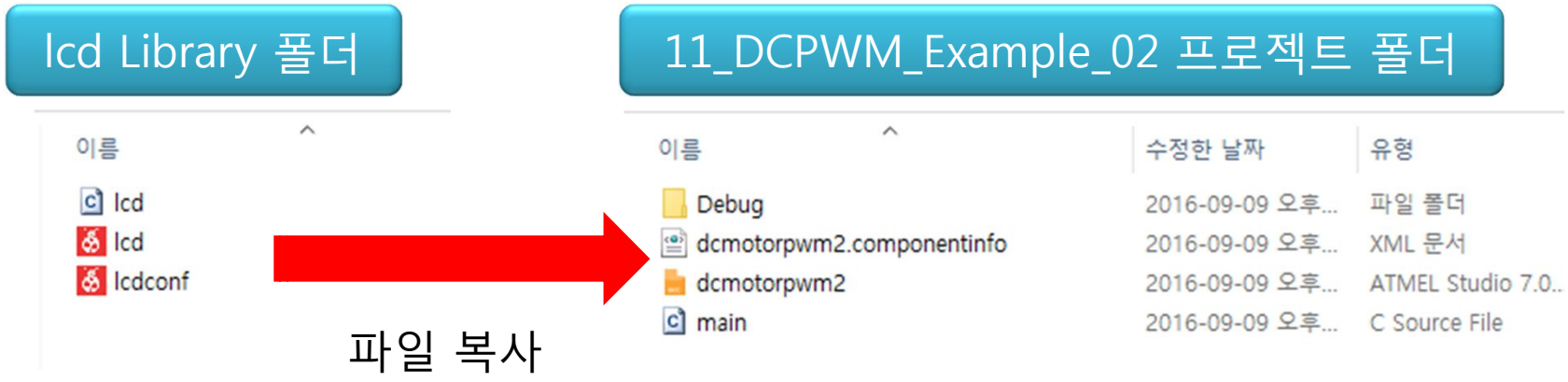
```
ICR1 = 800;      // 14745600/18432 => 800
```

  - 듀티비 70% → 800x70/100 = 560

```
OCR1A = 560;
```

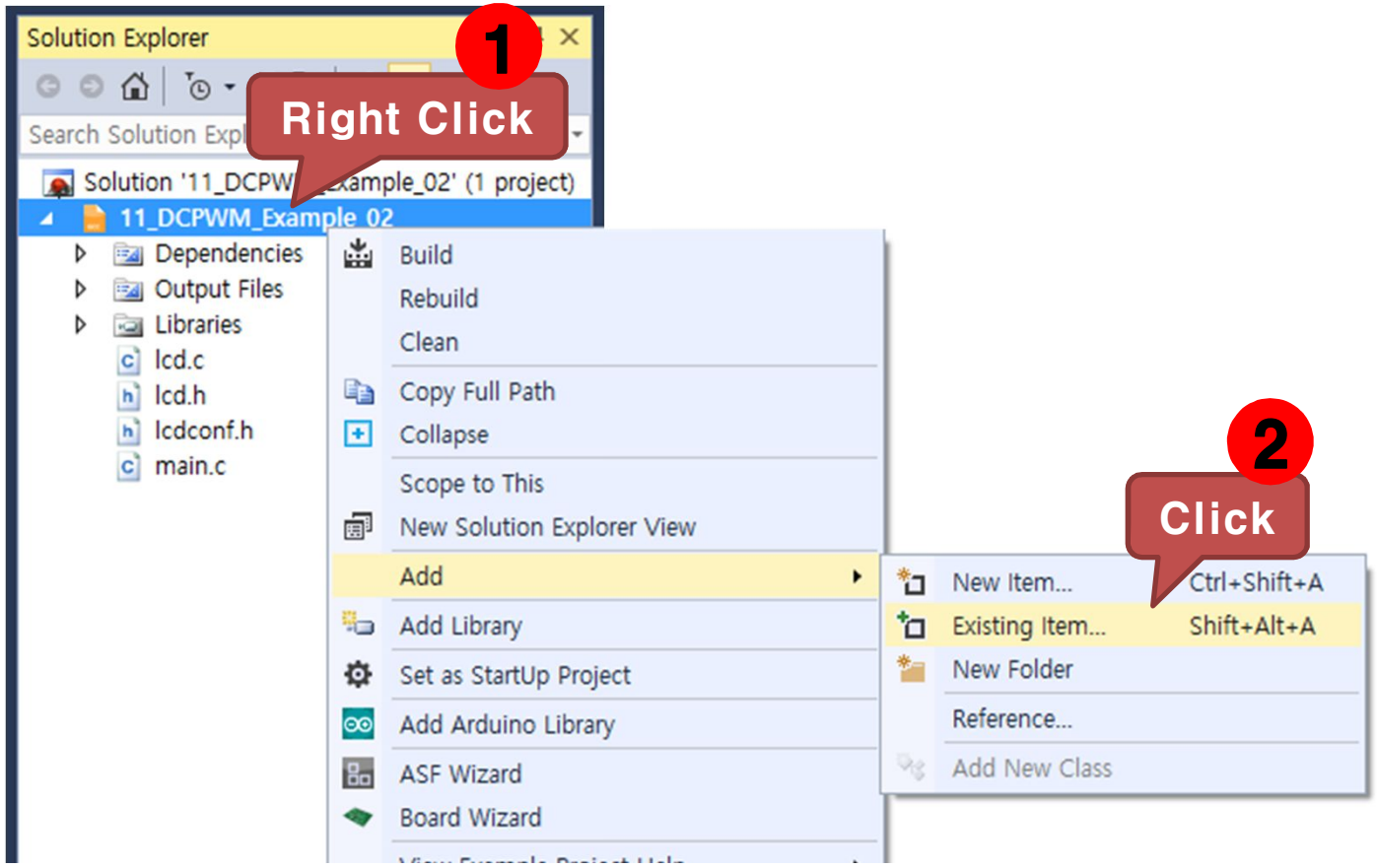
실습 4 : PWM을 이용하여 DC 모터 속도 제어하기 2

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
  - 라이브러리 함수파일을 프로젝트내로 복사
    - lcd.c, lcd.h, lcdconf.h 파일은 "AVR_Example₩library₩lcd" 폴더에 존재
    - 라이브러리 파일은 Atmel Studio에서 새 프로젝트를 생성한 후 프로젝트가 생성된 폴더안에 있는 프로젝트 폴더에 복사



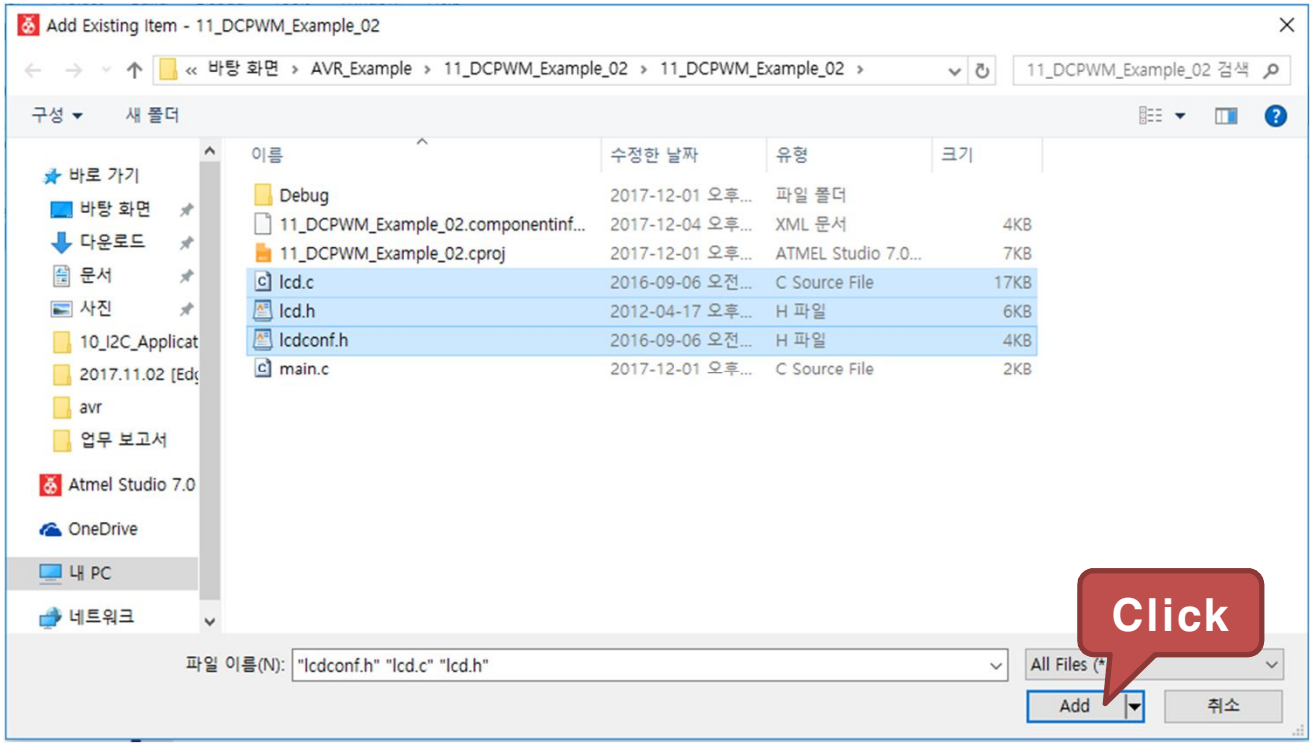
실습 4 : PWM을 이용하여 DC 모터 속도 제어하기 2

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
  - Atmel Studio 상에서 프로젝트의 솔루션 탐색기에서 라이브러리 파일들을 추가



실습 4 : PWM을 이용하여 DC 모터 속도 제어하기 2

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
  - lcd.c, lcd.h, lcdconf.h 파일을 각각 추가
  - 또는 한번에 세 파일을 동시에 추가
    - Ctrl 키를 누르고 파일 세개를 순서대로 클릭하여 세파일을 동시에 선택하고 한번에 추가



실습 4 : PWM을 이용하여 DC 모터 속도 제어하기 2

- 구동 프로그램
  - main.c 코드 작성

```
#include <avr/io.h>      // AVR 입출력에 대한 헤더 파일
#include <util/delay.h> // delay 함수사용을 위한 헤더파일
#include “lcd.h”         // Text LCD를 사용하기 위한 헤더 파일

#define M1_Forword 0x10
#define M1_Reverse 0x20
#define M1_Enable  0x20

int main(void) {
  unsigned char pwmduty = 0, cnt_dir = 0;
  DDRD = 0x30;  // MOTOR1_IN1, MOTOR1_IN2를 출력 포트로 설정
  DDRB = 0x20;  // MOTOR1_EN 를 출력 포트로 설정
```

실습 4 : PWM을 이용하여 DC 모터 속도 제어하기 2

- 구동 프로그램
  - main.c 코드 작성

```
int main(void) {
  //...
  TCCR1A=0x82;  // COMnA1=1,COMnA0=0,COMnB1=0,COMnB0=0
                // COMnC1=0,COMnC0=0,WGMn1=1,WGMn0=0
  TCCR1B=0x19;  // ICNCn=0,ICESn=0,Blink=0,WGMn3=1
                // WGMn2=1,CSn2=0,CSn1=0,CSn0=1
  TCCR1C=0x00;
  ICR1=800;     // 14745600/800 => 18.432KHz
  OCR1A=560;    // 듀티비 70%
  PORTD = M1_Forword;  // DC Motor 정회전

  lcdInit();             // Text LCD를 초기화
  lcdGotoXY(0,0);        // 커서위치를 첫번째 줄, 첫번째 칸으로 이동
  lcdPrintData(" Duty: ",7);    // " Duty: " 출력
```

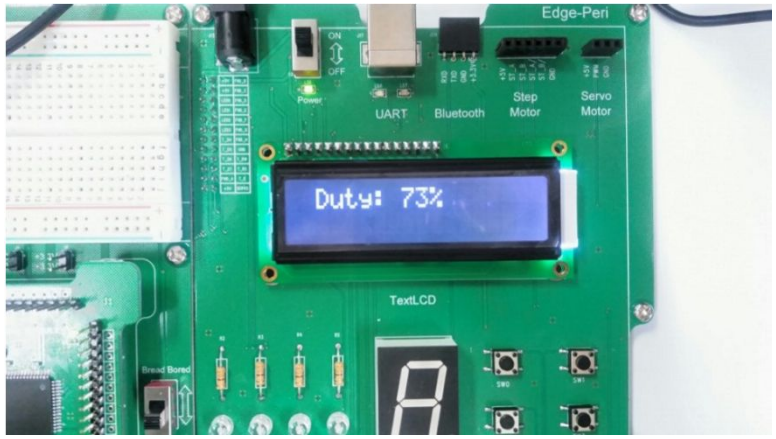
실습 4 : PWM을 이용하여 DC 모터 속도 제어하기 2

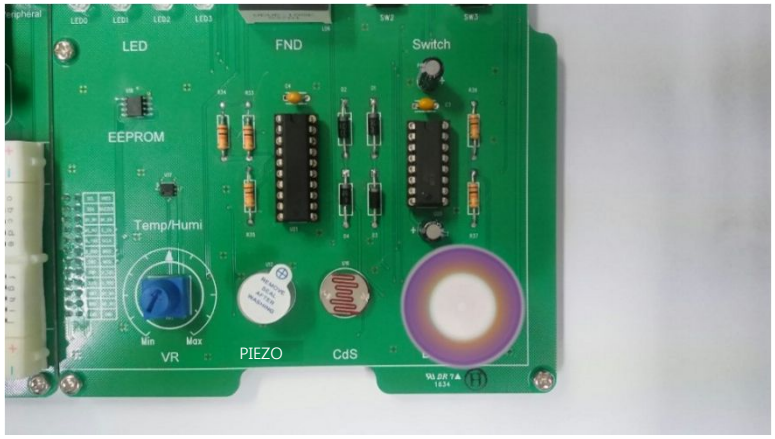
- 구동 프로그램
  - main.c 코드 작성

```
while (1) {
    if(cnt_dir) {
      pwmduty--;                              // pwmduty가 1씩 감소
      if(pwmduty == 0) cnt_dir = 0;           // 0이 되면 증가
    }else {
      pwmduty++;                              // pwmduty가 1씩 증가
      if(pwmduty == 100) cnt_dir = 1;         // 100이 되면 감소
    }
    OCR1A = 8*pwmduty;                        // 듀티비가 1%씩 증가
    lcdGotoXY(7,0);
    lcdDataWrite((pwmduty/10)%10 + '0'); // 10의 자리 출력
    lcdDataWrite((pwmduty)%10 + '0');    // 1의 자리 출력
    lcdDataWrite('%');                        // %출력
      _delay_ms(1000);
  }
}
```

실습 4 : PWM을 이용하여 DC 모터 속도 제어하기 2

- 실행 결과
  - 특정 듀티비 이상이면 모터가 회전하며, 이하이면 모터 정지





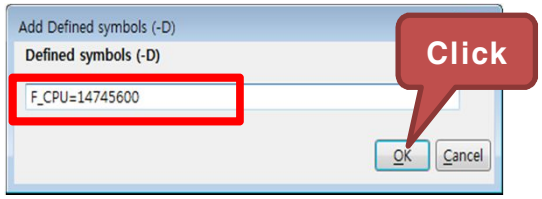
응용 : 인터럽트로 DC모터 동작 및 가변저항 값에 따라 속도 제어하기

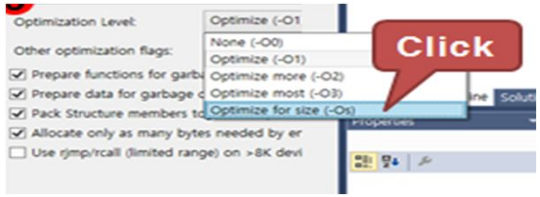
- 실습 개요
  - 타이머1의 PWM 기능을 이용하여 DC 모터 제어
  - 가변저항 값에 따라 PWM 듀티비를 증가 및 감소시켜 DC모터 속도의 변화를 관찰
  - 외부 인터럽트를 이용하여 스위치 입력에 따라 DC모터 동작 및 정지

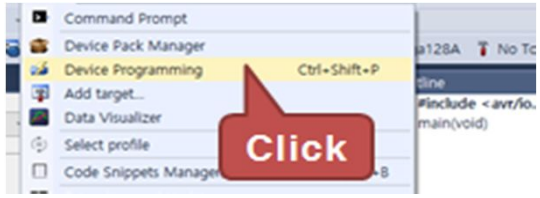
- 실습 목표
  - ATmega128A의 PWM 핀에 대한 프로그램 방법 습득
  - L293DN을 사용한 DC Motor 제어의 원리 이해
  - 타이머1의 PWM 기능 동작원리 이해

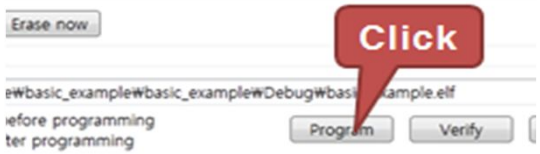
응용 : 인터럽트로 DC모터 동작 및 가변저항 값에 따라 속도 제어하기

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : 11_DCMOTOR_Application, Location : D:₩AVR_Example
    - Device Selection : ATmega128A
  - 프로젝트 설정
    - Project 탭에서 "... Properties..." 선택
    - Toolchain -> AVR/GNU C Compiler에서
    - Symbols -> F_CPU=14745600 추가
    - Optimization -> Optimize for size (-OS) 선택
    - 저장 (Ctrl+S)
  - 소스코드 작성
  - 프로젝트 빌드
    - Build 탭에서 "Build Solution" 클릭
  - 프로그래밍
    - Tool 탭에서 "Device Programming" 클릭
    - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
    - 인식 완료되면, Memories 탭 선택, "Program" 클릭









응용 : 인터럽트로 DC모터 동작 및 가변저항 값에 따라 속도 제어하기

- 구동 프로그램
  - Main.c 코드 작성

```
#include <avr/io.h>                 //AVR 입출력에 대한 헤더 파일
#include <avr/interrupt.h>          // AVR 인터럽트에 대한 헤더파일
#include <util/delay.h>             //delay 함수사용을 위한 헤더파일

#define M1_Forword 0x10
#define M1_Reverse 0x20
#define M1_Enable 0x20

volatile unsigned char DC_flag = 0;

int main(void)
{
        unsigned int AdData = 0;
```

응용 : 인터럽트로 DC모터 동작 및 가변저항 값에 따라 속도 제어하기

- 구동 프로그램
  - Main.c 코드 작성

```
int main(void) {
  //...
   DDRD = 0x30; // MOTOR1_IN1, MOTOR1_IN2를 출력 포트로 설정 한다.
   DDRB = 0x20; // MOTOR1_EN 를 출력 포트로 설정 한다.
   DDRE = 0x00; // PF1을 입력 포트로 설정한다.

   TCCR1A=0x83; //  COMnA1=1  COMnA0=0  COMnB1=0  COMnB0=0  COMnC1=0
COMnC0=0 WGMn1=1 WGMn0=1
   TCCR1B=0x01; //  ICNCn=0 ICESn=0 Blink=0 WGMn3=0 WGMn2=0 CSn2=0
CSn1=0 CSn0=1
   TCCR1C=0x00;  // PWM Phase Correct, 10-bit, Mode 3

   TCNT1=0;     // 주기 1474560/2/1024 = 720hz
   OCR1A=0;     // Off

   PORTD = M1_Forword;  // DC Motor 정회전
```

응용 : 인터럽트로 DC모터 동작 및 가변저항 값에 따라 속도 제어하기

- 구동 프로그램
  - Main.c 코드 작성

```
int main(void) {
  //...
   EICRB = 0x0F;  // 인터럽트 4, 5를 상승엣지에서 동작하도록 설정한다.
   EIMSK = 0x30;  // 인터럽트 4, 5를 허용
   EIFR = 0x30;   // 인터럽트 4, 5 플래그를 클리어

   ADMUX = 0x41;  //single mode, 1번 채널, 3.3V 외부 기준전압(AREF)
   ADCSRA = 0x87; // 10000111 //ADC 허가, 128분주

   sei();

   while (1)
   {
        if(DC_flag)
        {
```

응용 : 인터럽트로 DC모터 동작 및 가변저항 값에 따라 속도 제어하기

- 구동 프로그램
  - Main.c 코드 작성

```
int main(void) {
  //...
            // ADC0을 통한 ADC값 읽어오기
            ADCSRA |= 0x40; //ADSC AD 개시(Start)
            while((ADCSRA & 0x10) == 0x00); //ADIF AD 다 될 때까지
기다림.
            AdData = ADC;   //전압이 디지털로 변환된 값 읽어오기.

            OCR1A = AdData; // 가변 저항에 따라 모터 속도 제어

            _delay_ms(100);
        }
    }
}
```

응용 : 인터럽트로 DC모터 동작 및 가변저항 값에 따라 속도 제어하기

- 실행 결과
  - SW0을 누르면 모터가 정지
  - SW1을 누르고 가변저항을 돌리면 모터의 회전속도가 변함을 관찰