



# 목 차

<b>1장 ATmega128 시작 하기</b>	<b>1</b>
1. 학습목표	1
2. ATmega128	1
3. PAMS-VER V9.1 아두이노 & MCU 실습 장비	2
4. 세부 구성	4
5. 개발 환경 구축(CodeVision 설치)	14
<b>2장 LED 제어 실습</b>	<b>29</b>
1. 학습목표	29
2. 기초 이론	29
3. LED 구성	32
4. LED 제어	34
<b>3장 스위치 제어 실습</b>	<b>44</b>
1. 학습 목표	44
2. 기초 이론	44
3. 스위치 구성	46
4. 스위치 제어	48
<b>4장 Segment 제어 실습</b>	<b>55</b>
1. 학습 목표	55
2. 기초 이론	55
3. Segment 구성	59
4. Segment 제어	61
<b>5장 Character LCD 제어 실습</b>	<b>69</b>
1. 학습 목표	69
2. 기초 이론	69
3. Character LCD 구성	78
4. Character LCD 제어	80
<b>6장 KEY-Matrix 제어 실습</b>	<b>87</b>
1. 학습 목표	87
2. 기초 이론	87
3. Key-Matrix 구성	88

4. Key-Matrix 제어 .....	91
<b>7장 Servo Motor 제어 실습 .....</b>	<b>100</b>
1. 학습 목표 .....	100
2. 기초 이론 .....	100
3. Servo Motor 구성 .....	102
4. Servo Motor 제어 .....	104
<b>8장 Step Motor 제어 실습 .....</b>	<b>111</b>
1. 학습 목표 .....	111
2. 기초 이론 .....	111
3. Step Motor 구성 .....	114
4. Setp Motor 제어 .....	116
<b>9장 DC Motor 제어 실습 .....</b>	<b>125</b>
1. 학습 목표 .....	125
2. 기초 이론 .....	125
3. DC Motor 구성 .....	129
4. DC Motor 제어 .....	131
<b>10장 CDS 제어 실습 .....</b>	<b>140</b>
1. 학습 목표 .....	140
2. 기초 이론 .....	140
3. CDS 구성 .....	142
4. CDS 제어 .....	144
<b>11장 Interrupt 제어 실습 .....</b>	<b>153</b>
1. 학습 목표 .....	153
2. 기초 이론 .....	153
3. Interrupt 구성 .....	154
4. Interrupt 제어 .....	156
<b>12장 시리얼 통신 실습 .....</b>	<b>164</b>
1. 학습 목표 .....	164
2. 기초 이론 .....	164
3. 시리얼 통신 구성 .....	165
4. 시리얼 통신 제어 .....	167

## ■ 품 질 보 증 ■

제품에 대한 품질보증을 하고 있습니다.

이의 품질보증기간은 취득자가 매입일로부터 2년간을 무상보수 기간으로 하고, 요청시 수시 실시함을 원칙으로 정하고 있으며, 이 기간 중에 장비의 이상이 있을 때에는 직접 문의하여 주시기 바랍니다.

### 품질보증에 대한 유의사항

이 제품은 적절한 품질보증을 받기 위하여 다음 사항을 지켜줄 것을 바랍니다.

1. 장비를 사용하기 전에 취급설명서에 포함된 내용에 따라 충분한 장비 취급 및 사용법을 알고 있어야 한다.
2. 열이 많은 장소나 습기가 많은 장소에 보관하지 말아야 한다.
3. 전원을 연결하기 전에 장비의 입력전압을 체크하고 또한 장비에 누전이 되지 않도록 하여야 한다.
4. 본 장비는 특히 기계적으로 정밀성을 요구하므로 물리적인 큰 충격이 가지 않도록 한다.
5. 다음과 같은 경우에는 보증책임을 지지 않는다.
  - (1) 실수 또는 부주의로 인한 파손이나 성능을 개조한 경우
  - (2) 보증기간이 지난 경우(단, 보증기간 이후의 고장수리는 교체부품의 비용만 지불하는 실비 보수를 지속적으로 실시한다.)

## SECTION

## 01

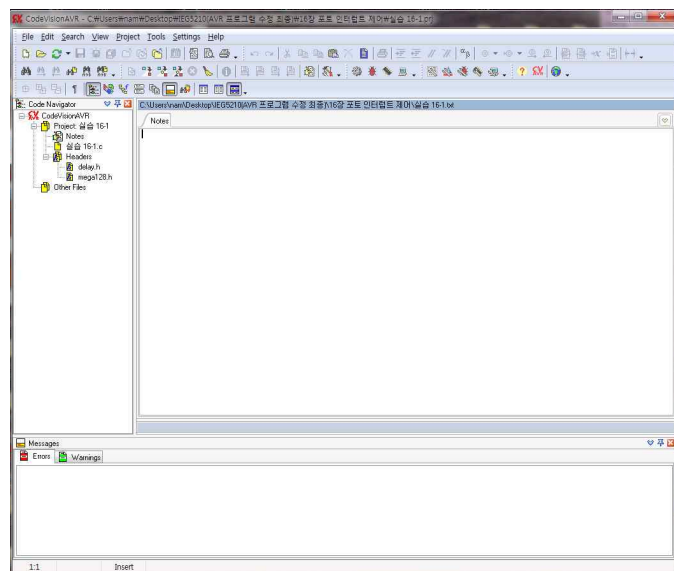
## ATmega128 시작 하기

## 1-1. 학습 목표

- ATmega128가 무엇인지 알고 장비의 구성요소를 이해 한다.
- ATmega128 실습을 위하여 ATmega128 개발 환경을 구축
- 장비 제어를 위하여 결선법 프로그램 작성법을 이해하도록 한다.

## 1-2. ATmega128

ATmega128는 주로 Atmel사의 AVR을 사용하는 마이크로컨트롤러 보드 이다. Atmega128은 저렴하며 가장 많이 사용하는 마이크로 컨트롤러중에 하나이다. ATmega128을 사용하기 위해서는 Codevision , AVR Studio 라는 컴파일러 플랫폼을 이용하여 프로그램을 작성하고 프로그램을 ATmega128 보드에 프로그램을 올리는 과정을 하게 된다. 그중에서 Codevision을 이용하여 장비를 컨트롤 하고 프로그램을 작성하는 과정을 다루도록 한다.



## 1-3. PAMS - AEB V9.1 아두이노 & MCU 실습 장비



### 장비의 특징

- ☐ 위 장비는 아두이노와 MCU를 동작 실험을 할수 있게 구성이 되어 있어 아두이노, AVR Stdio, Codevision 등과 같이 AVR 에 제어를 쉽게 할수있게 구성이 되어 있다.
- ☐ 본 Multi One-Chip CPU 장비는 One-Chip CPU 모듈 (아두이노 ATmega 2560 / 328 / ATmega128)을 교체하면서 실험 할 수 있도록 설계되어 손쉽게 여러 가지 One-Chip CPU를 교체하여 사용 할 수 있다.
- ☐ Multi One-Chip CPU 장비는 전자, 전기, 메카트로닉스 제어장비로 설계되어있다.
- ☐ 다양한 Peripheral System의 제공으로 I/O 포트, DC MOTOR PWM 양방향, STEPPING MOTOR의 속도 및 각도, LCD, FND, RELAY 및 외부 장비는 4Ø 단자와 케이블을 이용하여 제어 실험실습이 가능하도록 설계 되어 있다.
- ☐ Peripheral Part는 One-Chip CPU (아두이노 / AVR )를 MCU 와 외부 기기 등을 4Ø잭으로 직접 연결하여 One-Chip CPU 에서 제어 실습이 가능함.
- ☐ 3개의 확장포트를 이용하여 센서 및 응용 모듈의 장착이 가능하여야 한다.
- ☐ 전기/ 전자/ 통신/ 자동제어/ 자동차/ 로봇 등 다양한 산업 분야에 사용되는 광센서, 온도센서, 자기센서, 근접센서 등 31종을 MCU & DAQ 응용 실습장비 에 적용하여 프로그램 제어 및 센서 Data 측정 실습이 가능 하다.
- ☐ 센서의 기본회로 원리, 응용회로설계를 학습하고 센서 적용 방법을 습득할 수 있어야 한다.

- ☐ 센서모듈의 회로특성 Part를 Test Point 단자에서 오실로스코프로 측정이 가능하여야 한다.
- ☐ 센서 Data는 H/L Output (MCU용), Analog Output (DAQ용) 2가지 출력이 나오도록 설계되어야 한다.
- ☐ 산업현장에 사용되는 요소를 접목하여 현장 실무자 엔지니어 인력양성에 적합하여야 한다.
- ☐ 실험용 소스코드와 회로도를 포함한 각종 자료들이 포함되어 있어서 실험시 발생하는 문제점들을 지원한다.

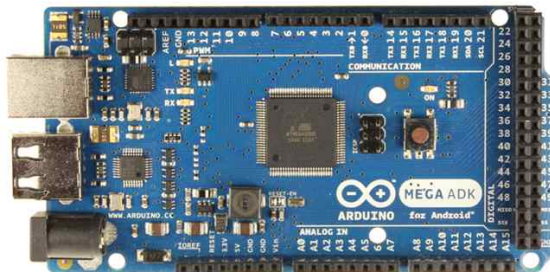
## 1-4. 세부 구성

### ☐ ATmega 128 Module



- Microcontroller : ATmega128
- Operating Voltage : 5V
- System Clock : 16MHz
- Flash Memory : 128 KB
- EEPROM : 4 KB
- SRAM : 4 KB

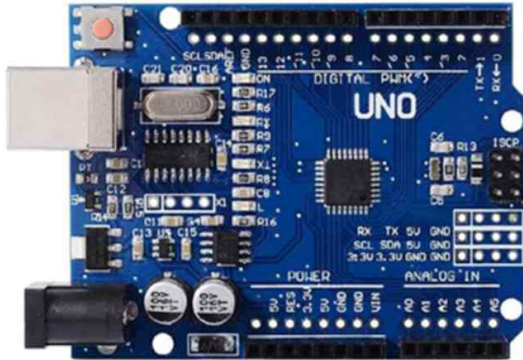
### ☐ Arduino 2560 Module



- Microcontroller : ATmega2560
- Operating Voltage : 5V
- Input Voltage (recommended) : 7~12V
- Input Voltage (limits) : 6~20V
- Digital I/O Pins : 54
- Analog Input Pins : 16
- Flash Memory : 256 KB of which 8 KB used by bootloader
- EEPROM : 4 KB
- SRAM : 8 KB
- Clock Speed : 16 MHz
- Base Module
- \* Arduino CPU Connection Port : 8P × 5EA, 10P × 1EA, 36P × 1EA



□ Arduino 328 MCU MODULE



- ATmega 328 모듈
- 클럭 : 16MHz 이상
- 디지털 입출력 Pin : 12개 이상
- 아날로그 Port Pin : 6개 이상
- PWM CH : 6이상
- 플래시메모리 : 32KBytes 이상
- SRAM : 2KBytes 이상
- EEPROM : 1KBytes 이상
- USB 케이블

□ ADC 구성(온도, 가변 전압, CDS)



- 가변 저항 × 1EA
- CDS × 1EA
- LM35 × 1EA

□ 입력 스위치



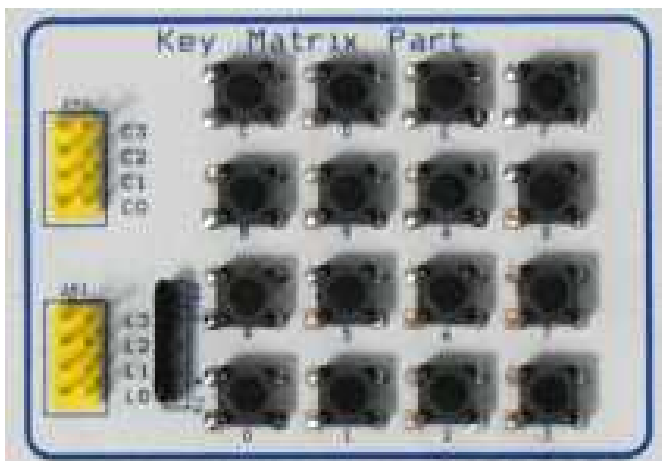
- Toggle S.W × 4EA
- Tack S.W × 4EA
- S.W Output Pin : 8P

□ LED



- LED × 8EA (RED × 4EA, Green × 4EA)
- LED Output Pin : 8P

□ KEY- Matrix



- Tack S.W × 16EA
- Control Port : 4P X 2EA

☐ Segment



- FND Control Part : FND × 4EA
- FND Control Driver : TR × 4EA

☐ LCD 16x2



- 16 X 2 Line Character LCD
- Control Pin : 4P, 8P

☐ Relay



- Realy (DC 24V)
- Operation LED × 2EA

☐ Bluetooth



- Module : HC-06
- Input Voltage : 3.6V ~ 6V
- Effective distance : 10m

☐ USB, RS232C Serial



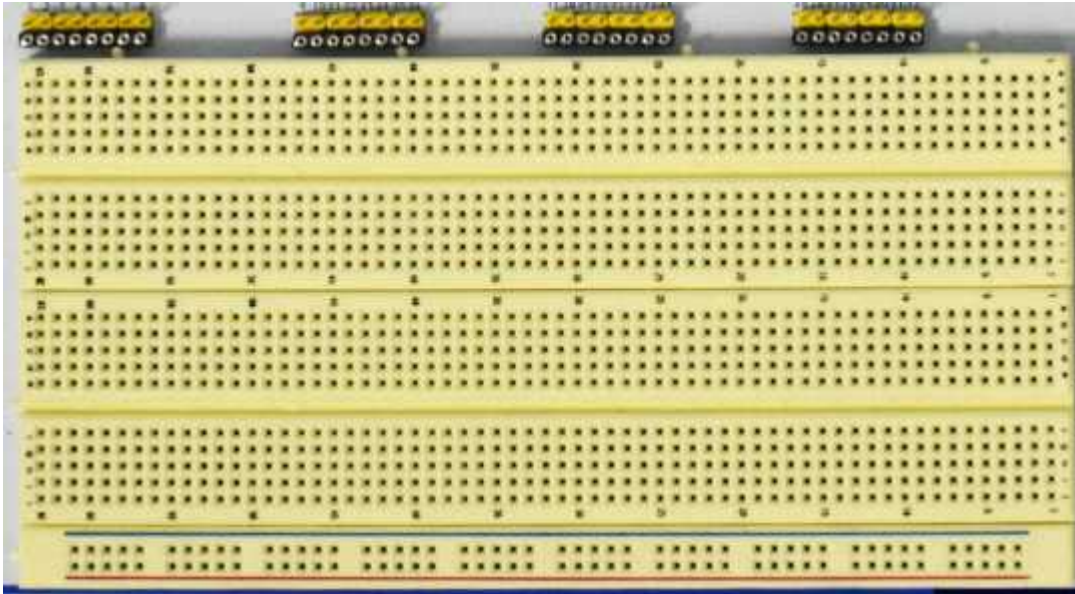
- USB Serial Port
- RS232C Serial Port

☐ EXT Power



- +5V, +1V, +24V, -12V, GND

□ 브레드 보드

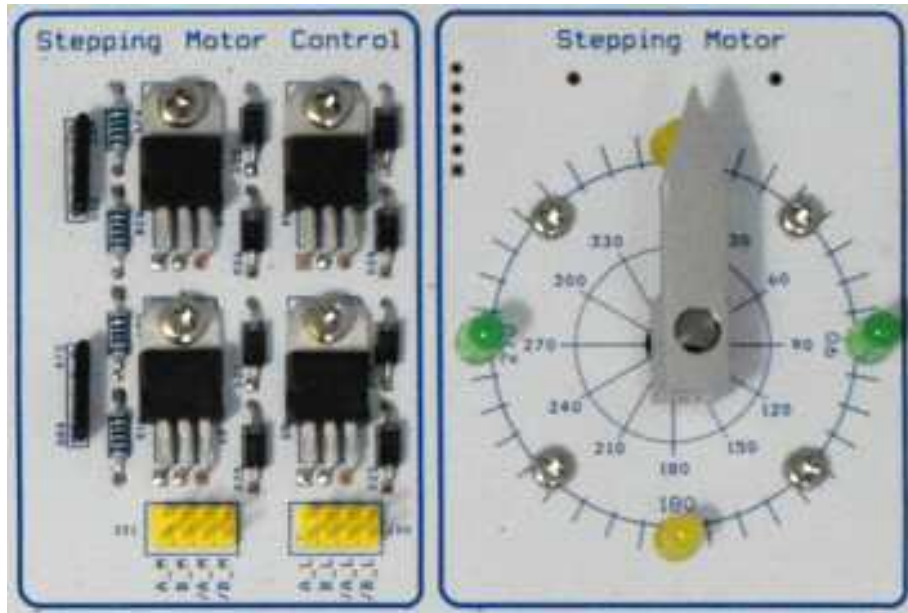


- Bus × 2EA, Power × 1EA

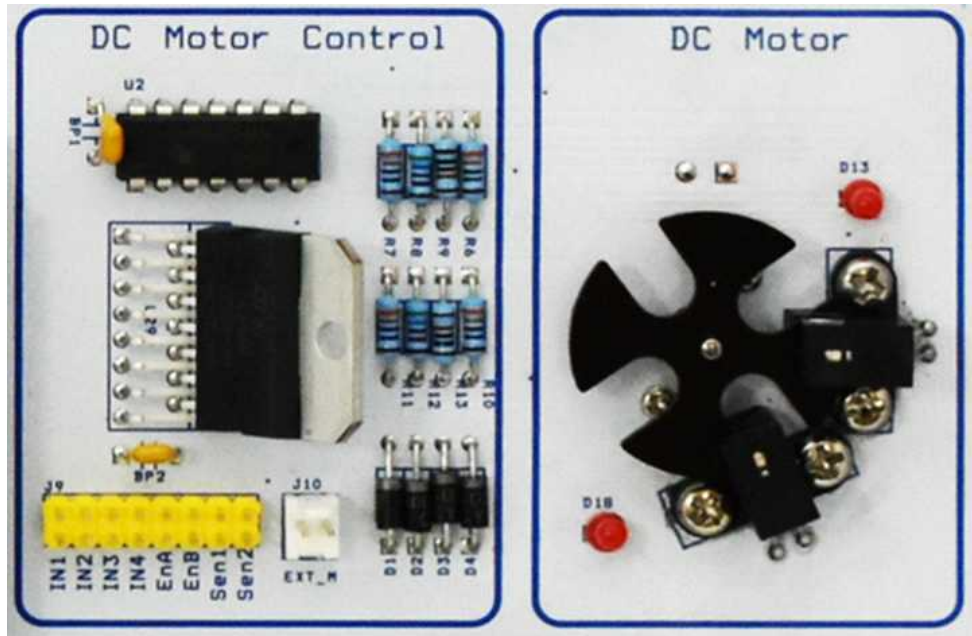
□ Interrupt



- Tack S.W × 2EA
- Output Port : 2P X 1EA

☐ Step Motor

- Stepping Motor × 1EA(42각, 1.8°)
- Phase Control LED × 4EA
- Drive TR × 4EA
- Motor Control Port : 4P X 2EA
- 각도 제어 및 속도 제어

☐ DC Motor


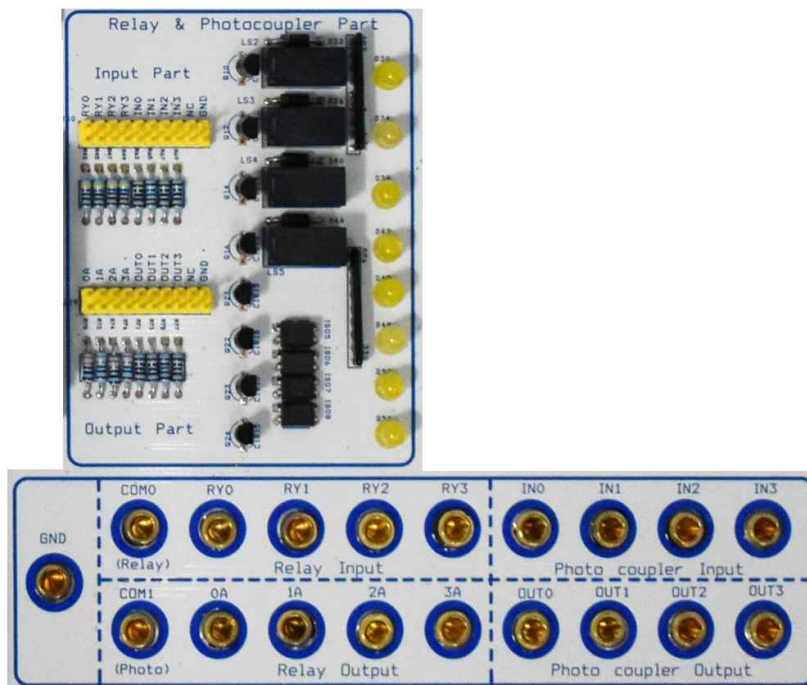
- DC Motor × 1EA (Drive IC × 1EA, Photo Interrupt × 2EA)
- Photo Interrupt 감지 LED : 2EA
- DC 5V/6,000 RPM
- Motor Control Port : 6P X 1EA

☐ Servo Motor


- RC SERVO Motor × 1EA
- RC SERVO Control Port : 2P × 1EA



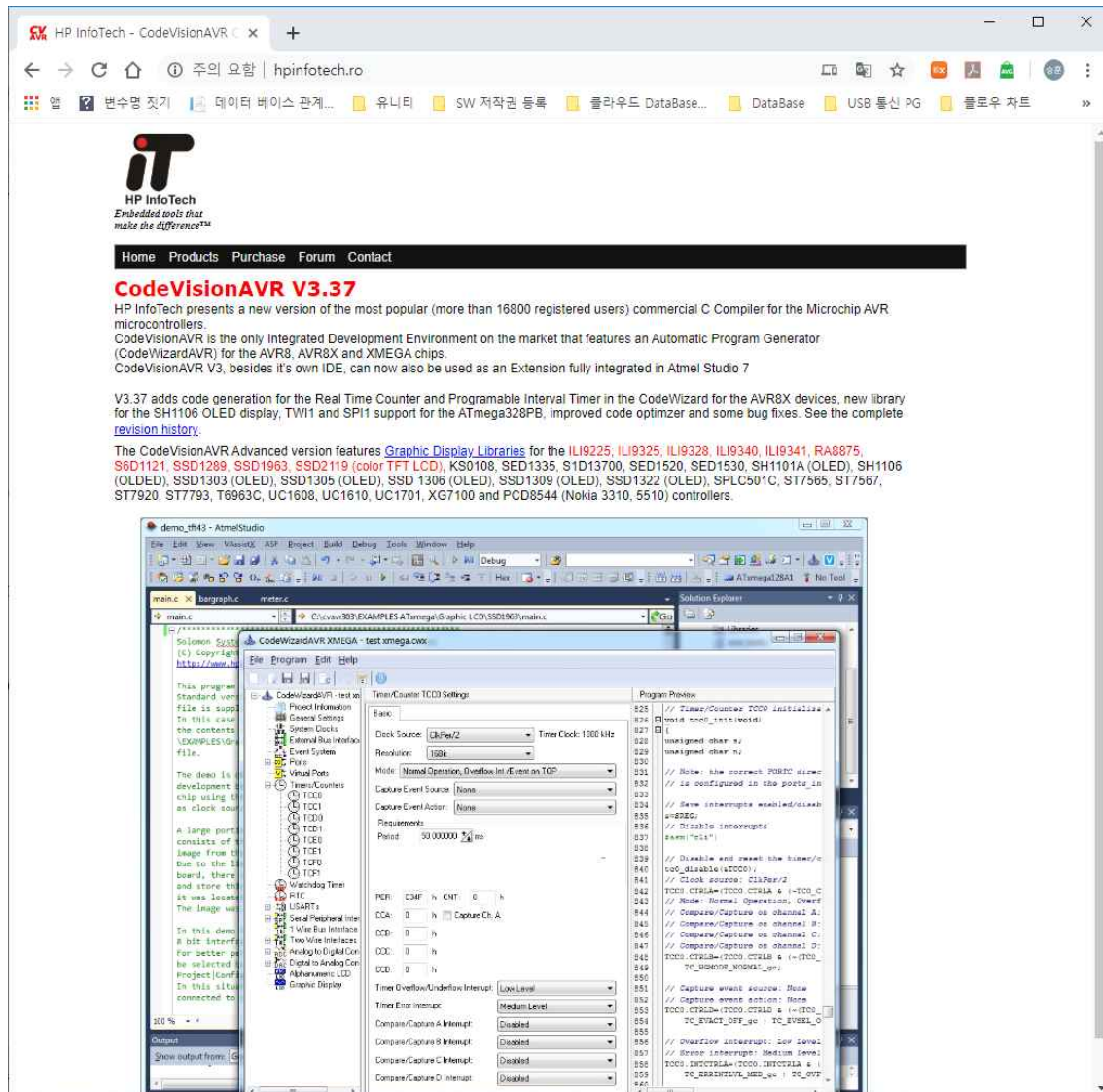
□ Relay, Photocoupler



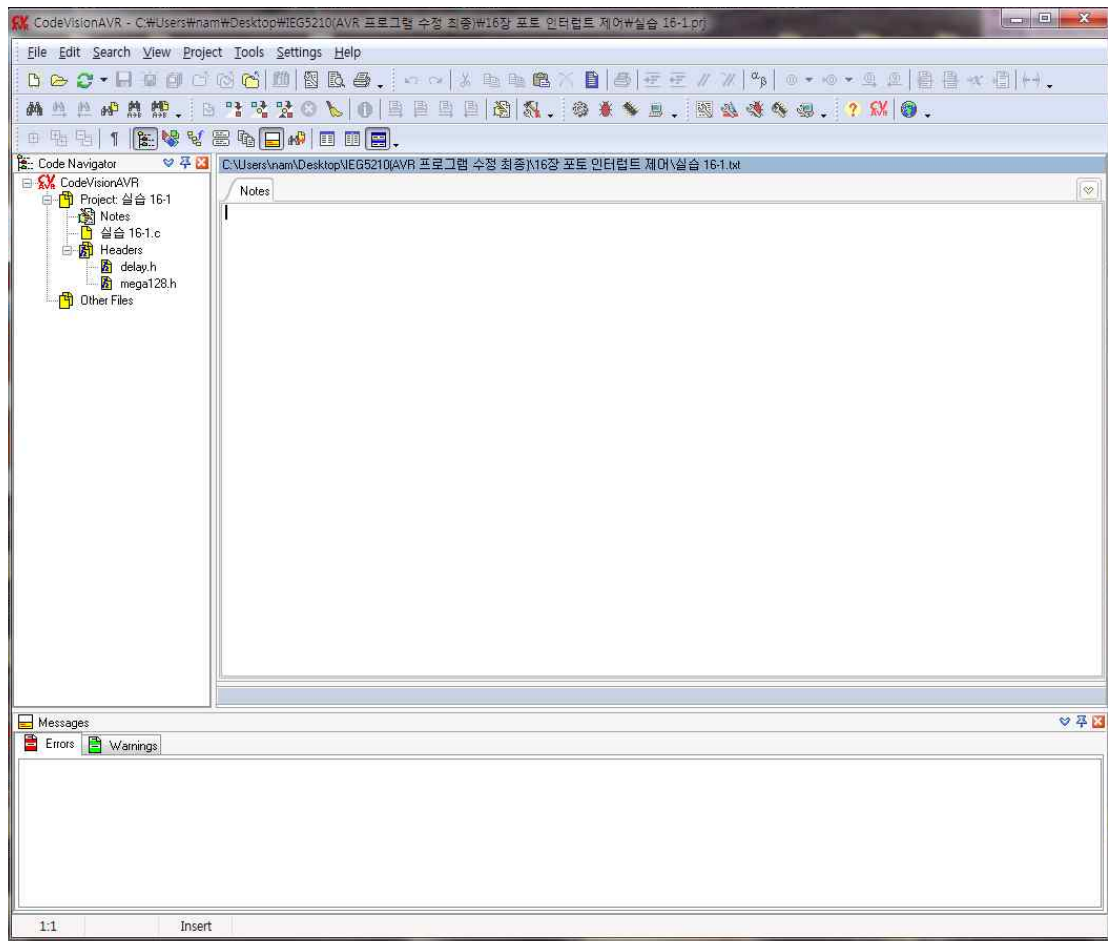
- Relay × 4EA (DC 24V)
- 입출력 Port (Header 8pin × 1EA, 2pin × 2EA)
- 4Φ 단자 × 11EA (입력 × 4EA, 출력 × 4EA, COM × 2EA, GND × 1EA)
- TR Relay Driver × 4EA
- Photocoupler : DC 24V Input × 4EA (LED × 4EA)
- 입출력 Port (Header 8pin × 1EA), 4Φ 원형단자 × 8EA  
(입력 × 4EA, 출력 × 4EA)

## 1-5. 개발 환경 구축(CodeVision 설치)

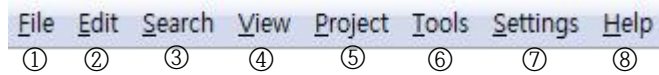
- CodeVision 소프트웨어 툴은 CodeVision 홈페이지(<http://www.hpinfoTech.ro/>)에서 다운로드가 가능하다.



- ☐ CodeVision 다운로드 하여 설치를 하게 되면 아래와 같이 Codevision 화면을 확인 할 수 있다.



## [1] 메뉴 설명



### ① File(기본 기능 설명)

- new : 새로운 Project 만들기, Source 만들기
- Open : 파일 열기(Project, Source 등)
- Save : 파일 저장
- Print : 프린터 출력

### ② Edit(기본 기능 설명)

- Cut : 잘라내기 기능
- Copy : 복사 기능
- Delete : 삭제 기능
- Find : 찾기 기능

### ③ View(기본 기능 설명)

- Toolbar : 툴바 아이콘 보기

### ④ Project(기본 기능 설명)

- Compile : 프로그램 컴파일
- Make : 프로그램 체크 및 error 체크

### ⑤ Tools(기본 기능 설명)

- CodeWizardAVR : one chip 기본 설정
- Chip Programmer : chip 프로그램 라이팅
- Terminal : 시리얼 통신 프로그램

### ⑥ Settings(기본 기능 설명)

- Programmer : ISP통신 포트 설정

### ⑦ Windows

- Tile Horizontal : 화면 설정
- Tile Vertical : 화면 설정

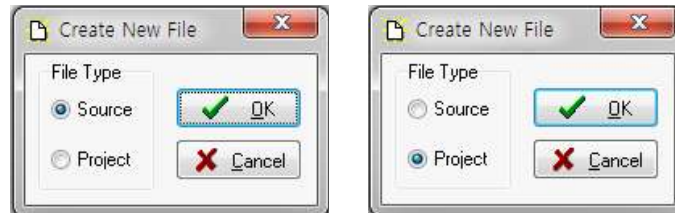
### ⑧ Help

- 프로그램 도움말, 프로그램 버전 정보 표시

## 2) Project 만들기

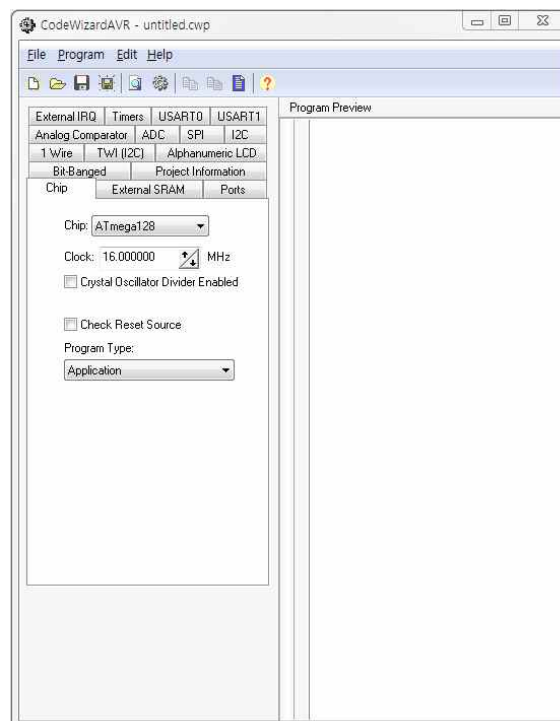
### [1] 새로운 Project 만들기

※ File → New 클릭 → Project 체크 → ok 클릭

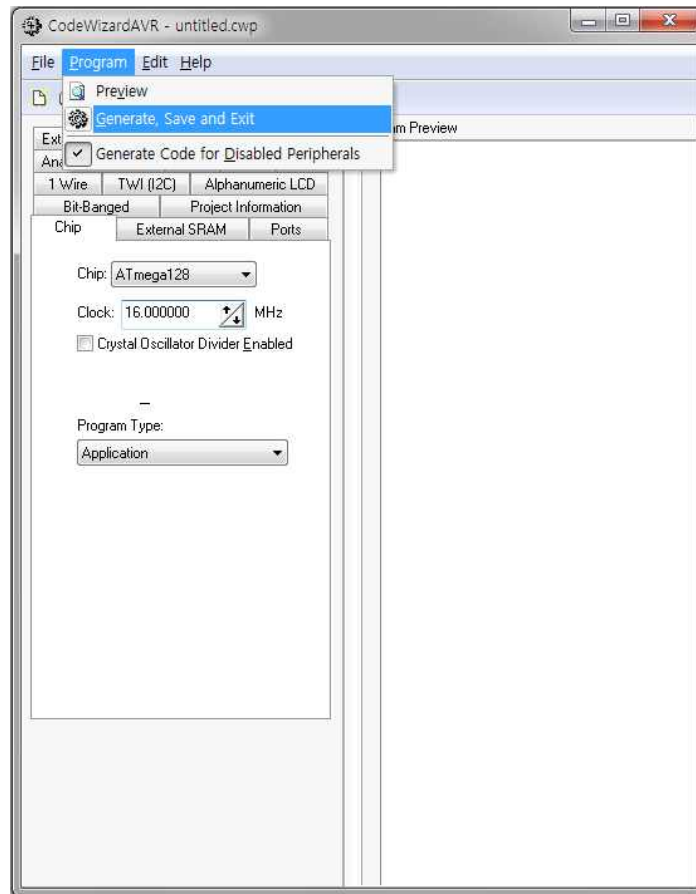


### [2] CodeWizardAVR 사용

※ “Yes 클릭”

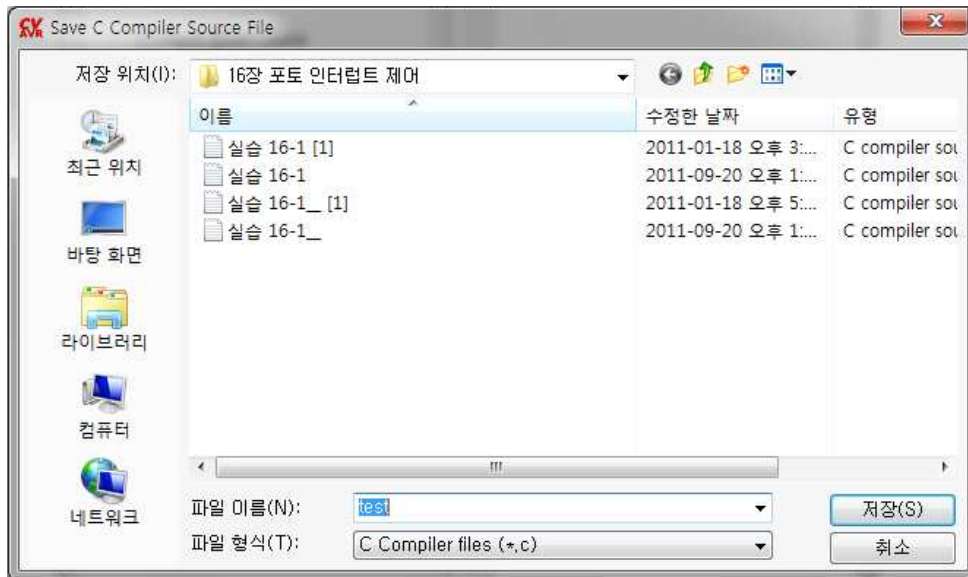


### [3] 프로그램 저장 위치 및 세팅 저장



#### [4] C Compiler Source file 만들기

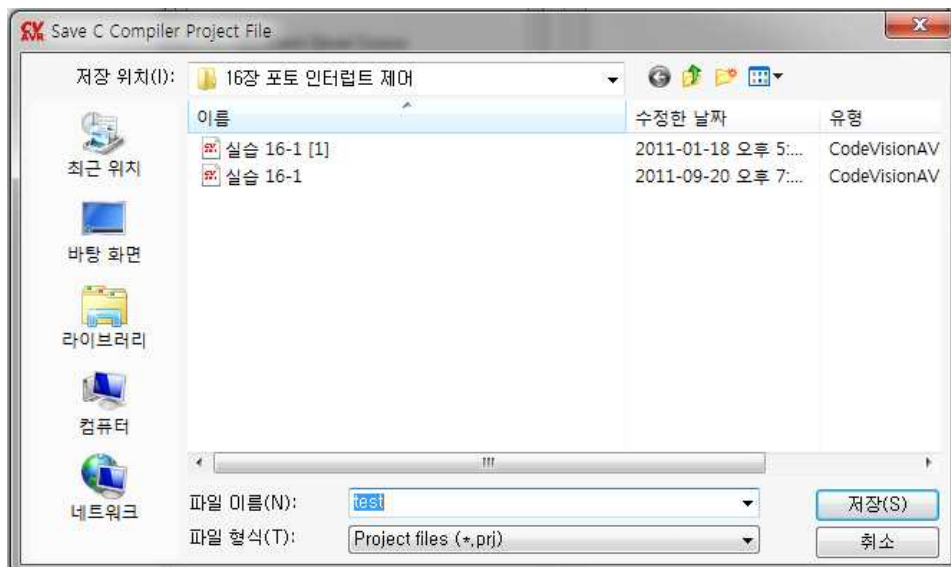
※ 저장 위치 지정, 파일이름 입력, 저장위치와 파일 이름 입력 후 “저장클릭”



#### [5] C Compiler Project file 만들기

※ 저장 위치 지정, 파일이름 입력, 저장위치와 파일 이름 입력 후 “저장클릭”

★ C Compiler Source file, C Compiler Project file 이름은 동일해야 함

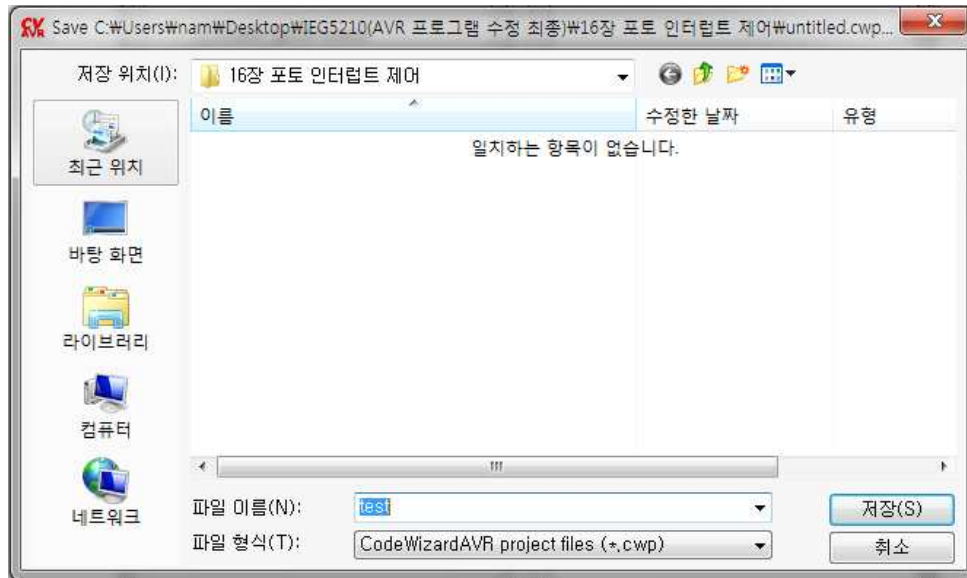




## [6] untitled.cwp file 만들기

※ 저장 위치 지정, 파일이름 입력, 저장위치와 파일 이름 입력 후 “저장클릭”

★ C Compiler Source file, C Compiler Project file,untitled.cwp file 이름은 동일해야 함



## [7] Project 완성 화면

```

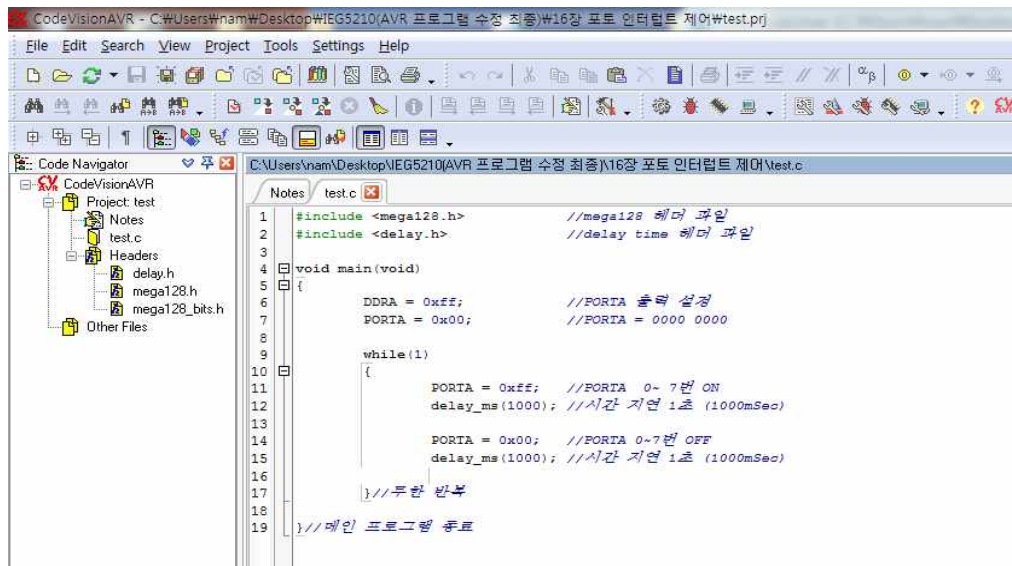
C:\Users\nam\Desktop\IEG5210(AVR 프로그램 수정 최종)\16장 포토 인터럽트 제어\test.c
Notes test.c
1 //*****
2 This program was produced by the
3 CodeWizardAVR V2.05.0 Evaluation
4 Automatic Program Generator
5 ?Copyright 1998-2010 Pavel Haiduc, HP InfoTech s.r.l.
6 http://www.hpinfotech.com
7
8 Project :
9 Version :
10 Date : 2011-09-20
11 Author : Freeware, for evaluation and non-commercial use only
12 Company :
13 Comments:
14
15
16 Chip type : ATmega128
17 Program type : Application
18 AVR Core Clock frequency: 16.000000 MHz
19 Memory model : Small
20 External RAM size : 0
21 Data Stack size : 1024
22 //*****
23
24 #include <mega128.h>
25
26 // Declare your global variables here
27
28 void main(void)
29 {
30 // Declare your local variables here
31
32 // Input/Output Ports initialization
33 // Port A initialization
34 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
35 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
36 PORTA=0x00;
37 DDRA=0x00;
38

```



### 3) 간단한 프로그램 작성 및 chip 라이팅

#### (1) portA(0 ~ 7번) 1초 on/off 프로그램 작성

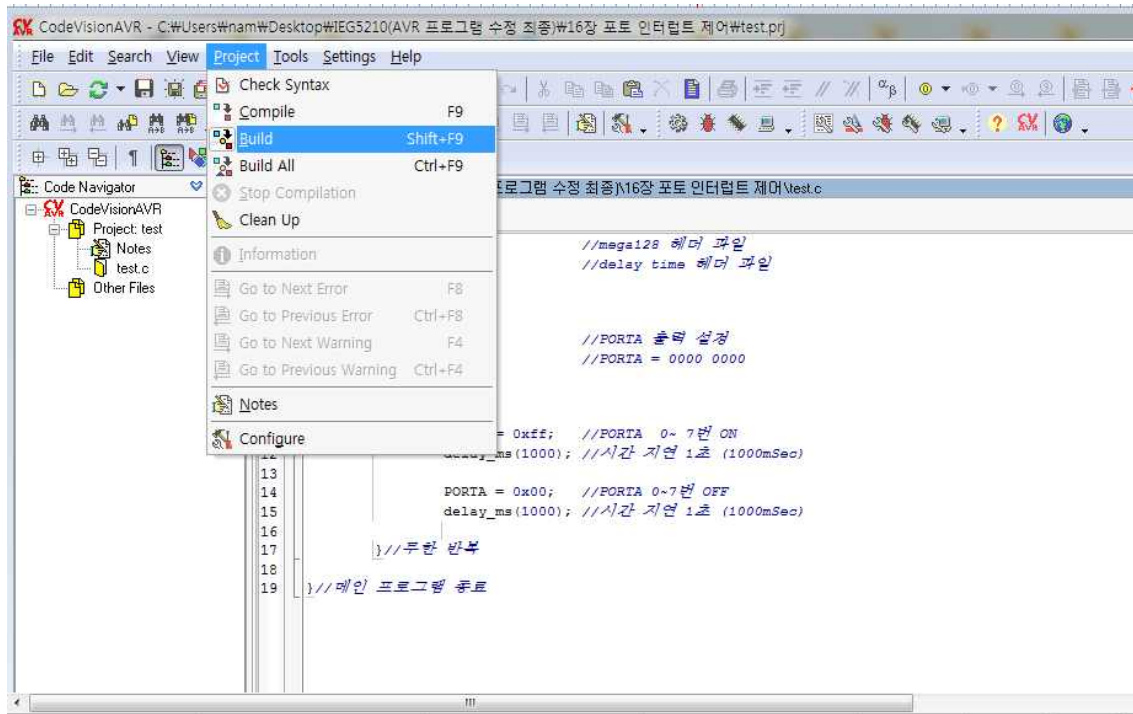


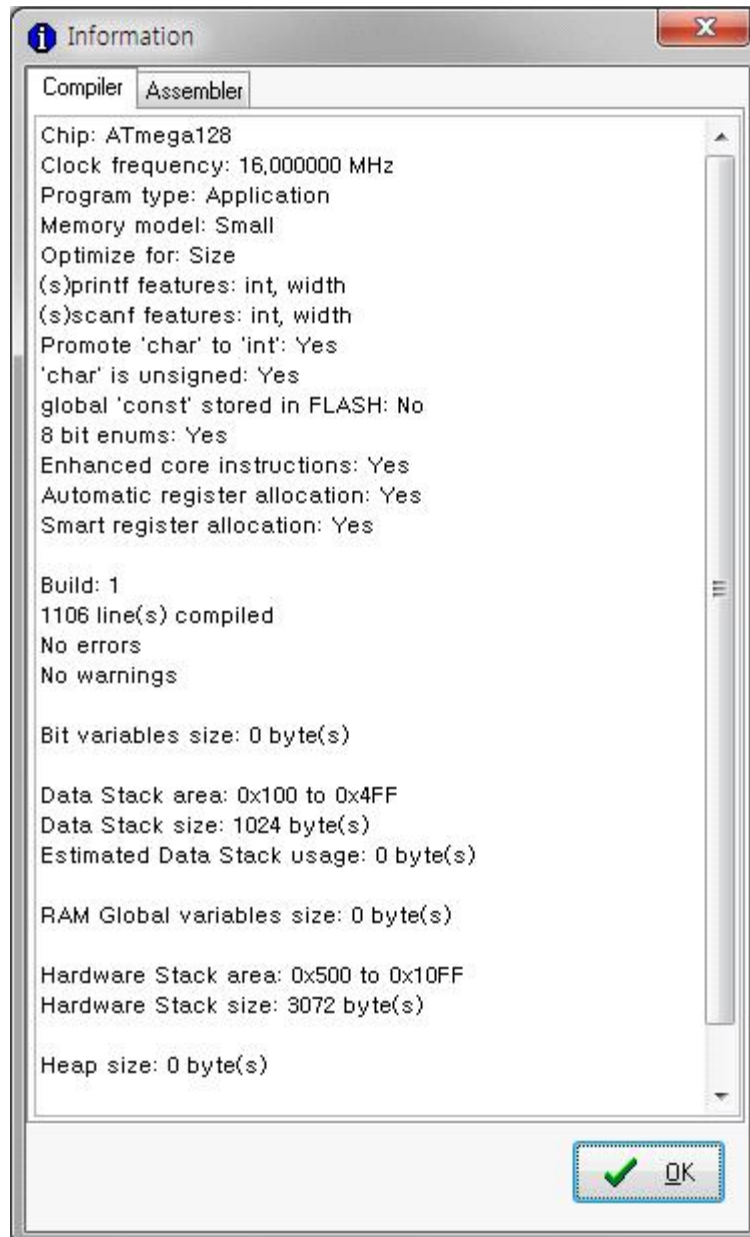
```

#include <mega128.h>           //mega128 헤더 파일
#include <delay.h>             //delay time 헤더 파일
void main(void)
{
    DDRA = 0xff;               //PORTA 출력 설정
    PORTA = 0x00;              //PORTA = 0000 0000
    while(1)
    {
        PORTA = 0xff;           //PORTA 0~ 7번 ON
        delay_ms(1000);         //시간 지연 1초 (1000msec)
        PORTA = 0x00;           //PORTA 0~7번 OFF
        delay_ms(1000);         //시간 지연 1초 (1000msec)
    } //무한 반복
} //메인 프로그램 종료
  
```

## [2] 프로그램 ERROR 체크

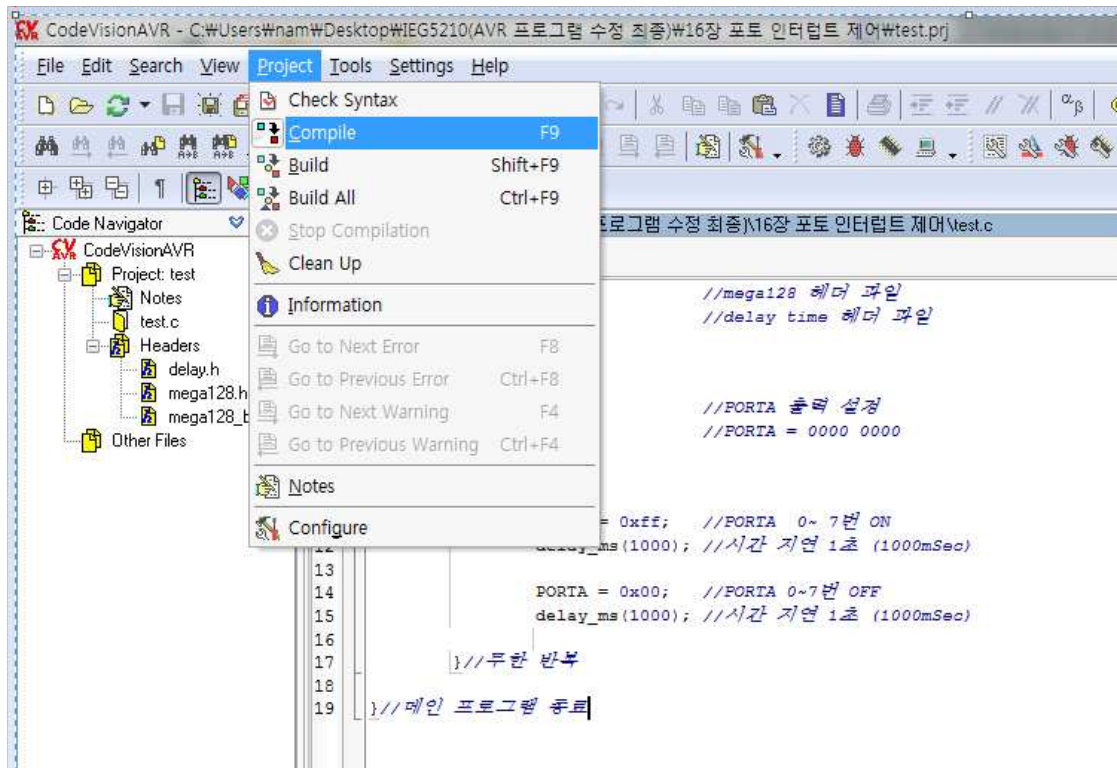
※Project →Build 클릭(단축키 : Shift + F9)

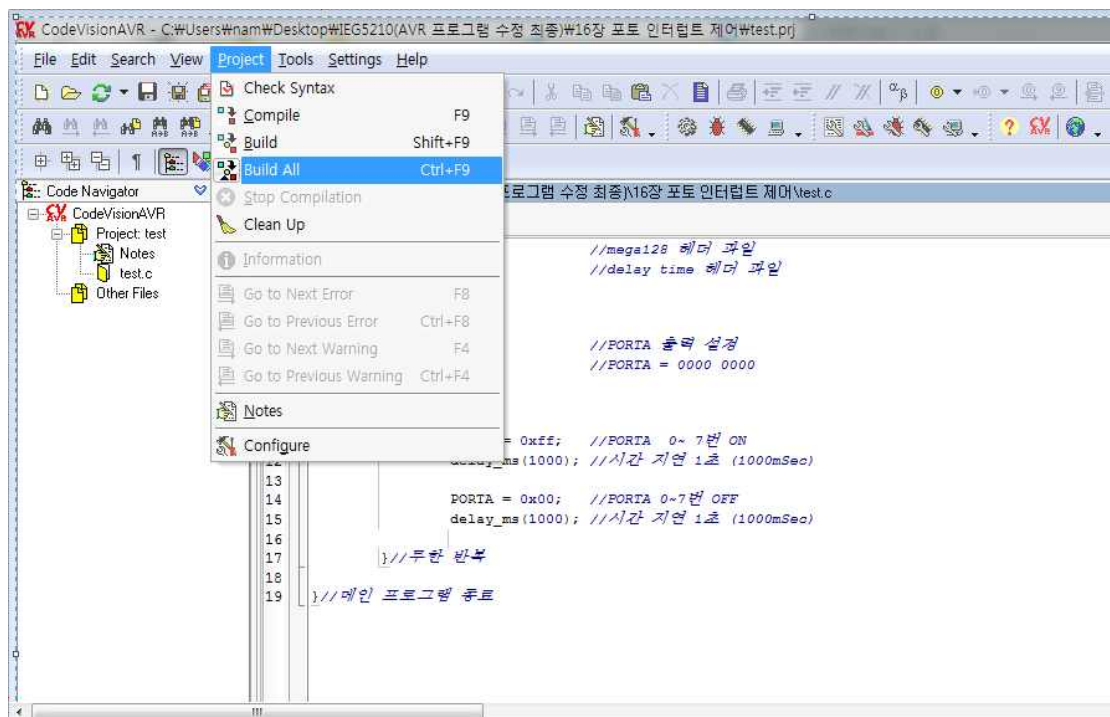
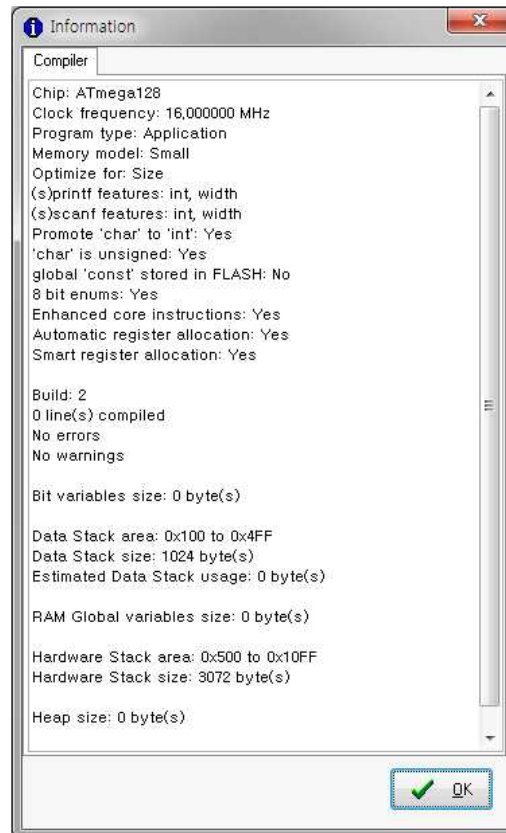


**(3) 프로그램 ERROR 코드 확인 후 “OK 클릭”**

#### [4] Compile-파일 생성

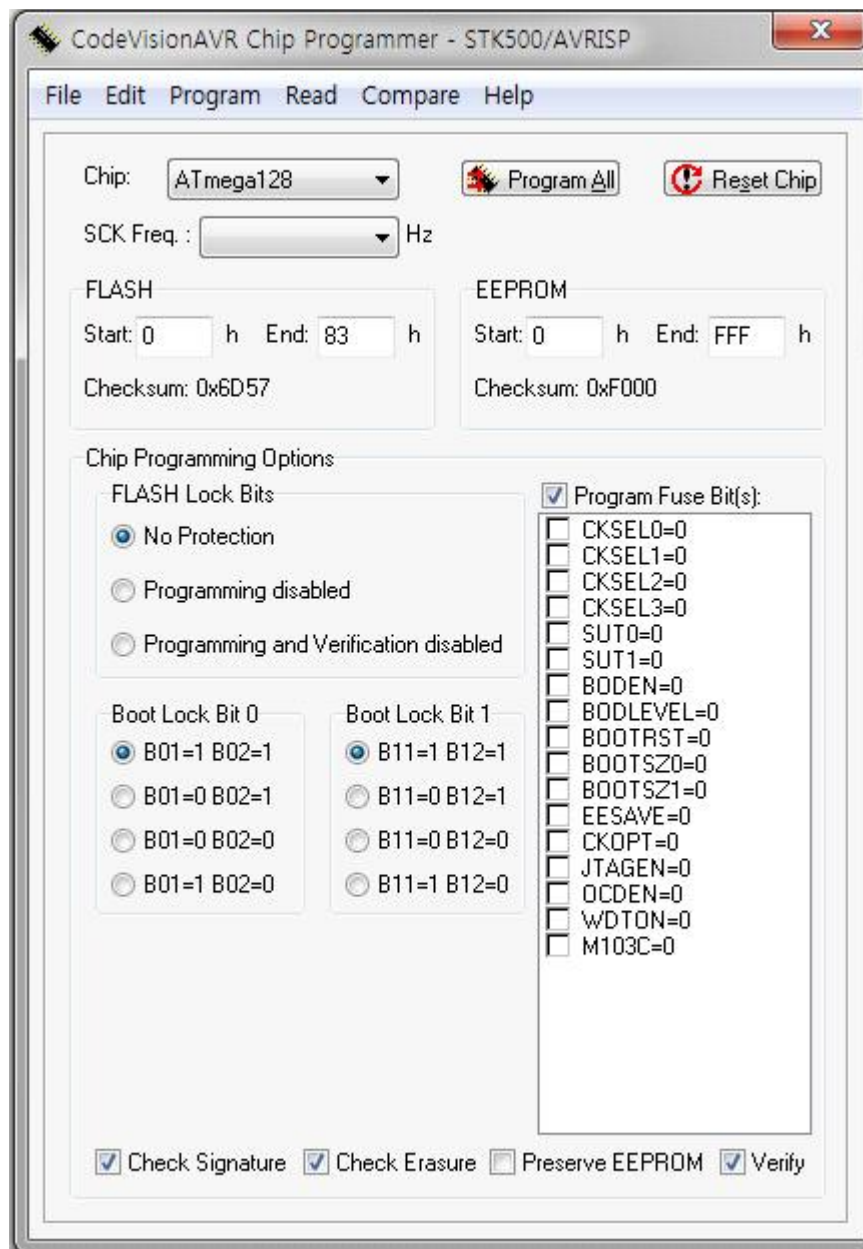
※Project →Compile 클릭(단축키 : F9)



**(5) 프로그램 Compile ERROR 확인 후 “OK 클릭”**

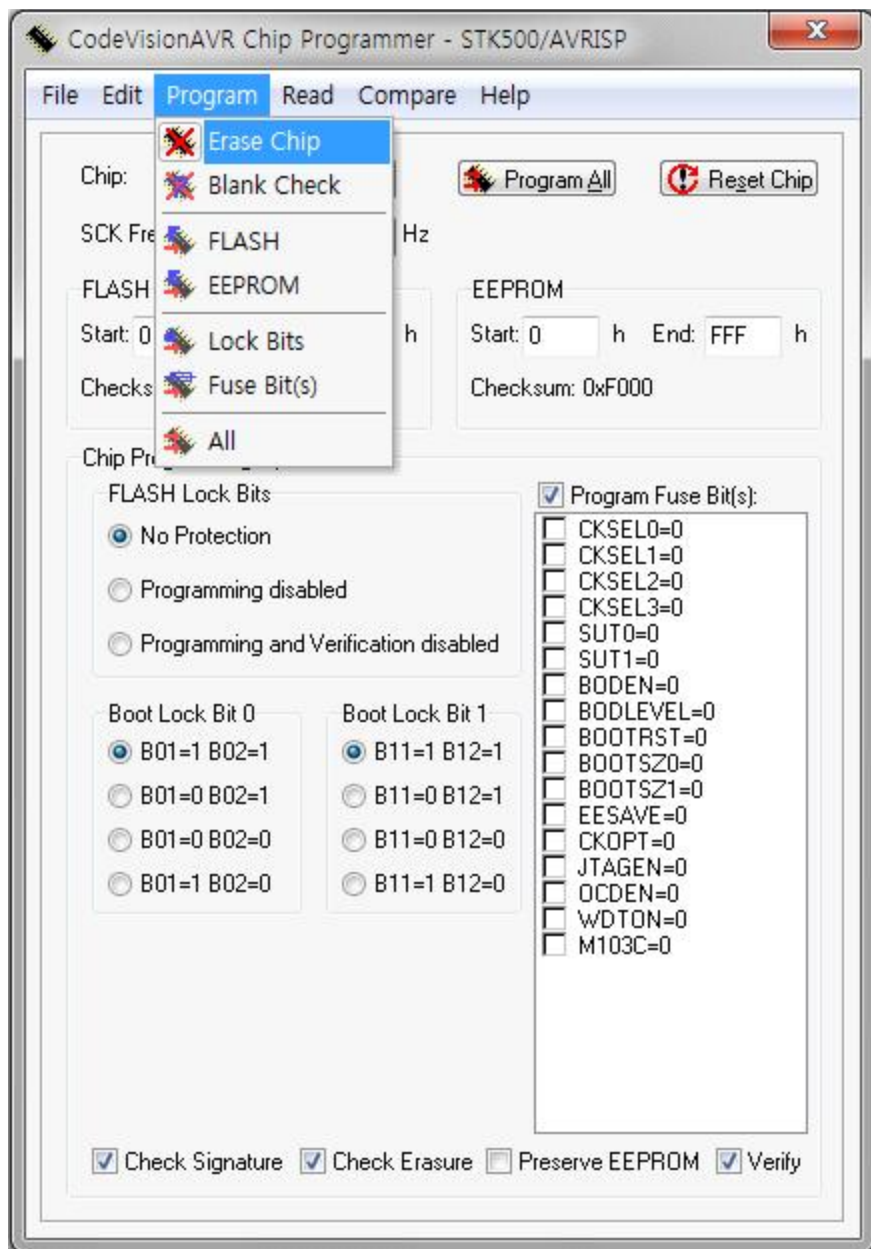
## [6] chip write

※Tools →Chip Programmer 클릭(단축키 : Shift+F4)

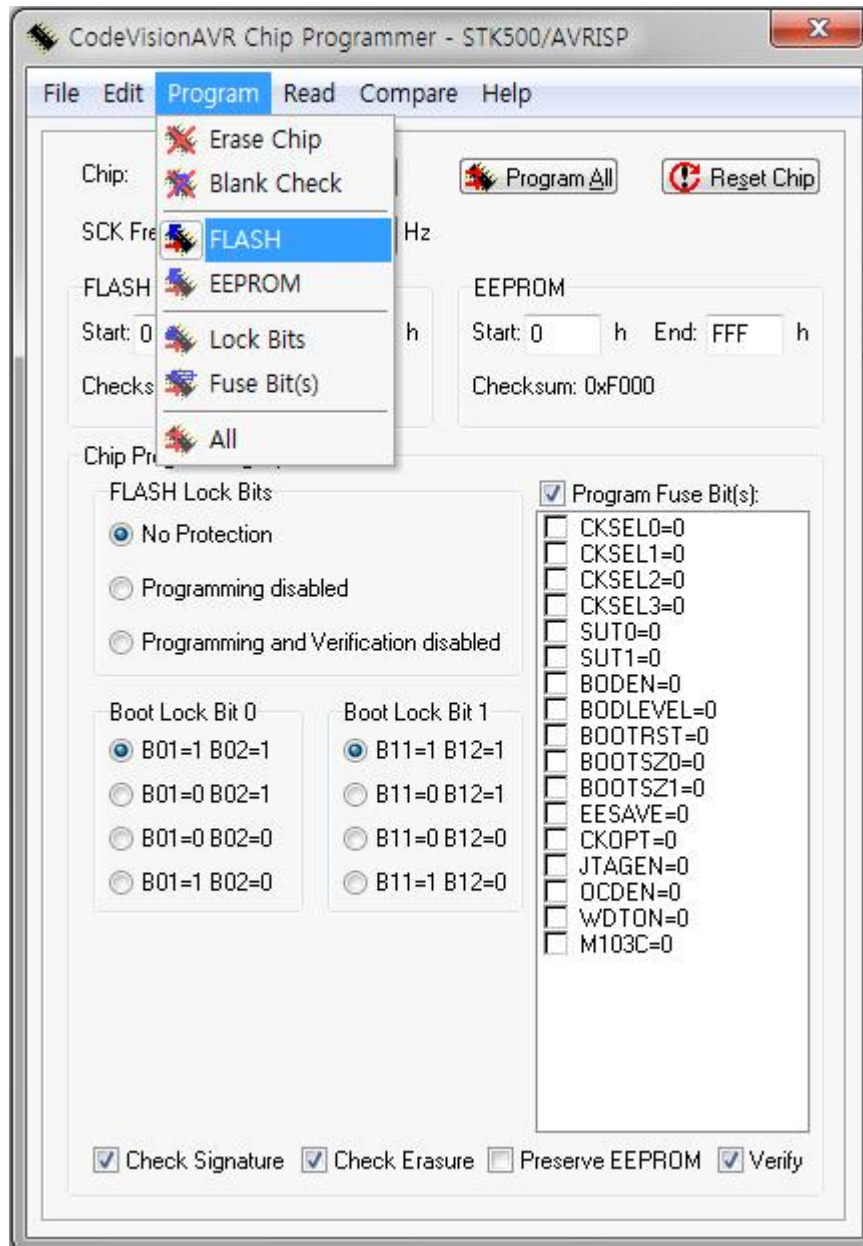


## (7) chip Erase

※Erase chip 클릭





**(8) chip Flash**



## SECTION

# 02 LED 제어 실습

## 2-1. 학습 목표

- LED 소자의 동작 방법
- ATmega128을 이용하여 LED ON 제어
- ATmega128을 이용한 ON/OFF 시간 지연 제어
- LED 왼쪽 또는 오른쪽 쉬프트 이동 제어

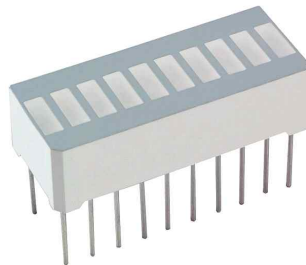
## 2-2. 기초 이론

### (1) LED 란?

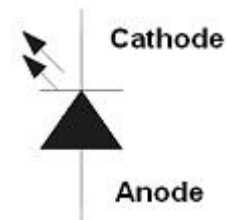
LED란? 전기적인 신호를 빛에너지로 변환하는 소자이다.



부품 외형A (일반 LED)



부품 외형B (Bar LED)



회로도 기호

Light Emitting Diode의 약자로 발광 다이오드를 뜻하며 이는 화합물반도체의 특성을 이용해 전기 신호를 적외선 또는 빛으로 변환시켜 신호를 보내고 받는데 사용되는 반도체의 일종으로 가정용 가전제품, 리모컨, 전광판, 표시기, 각종 자동화기기 등에 사용된다.

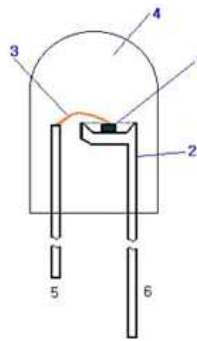
LED는 크게 IRED(Infrared Emitting Diode)와 VLED(Visible Light Emitting Diode)로 나뉜다.

## [2] Infrared LED chip

IRED(Infrared Emitting Diode)Chip을 뜻하는 것이다. 이 칩은 가공되어 IR Emitter, TV 리모컨, 광학스위치, IR LAN, 무선 디지털 데이터 통신용 모듈 등에 쓰인다.

## [3] Visible LED chip

VLED(Visible Light Emitting Diode)Chip을 뜻하며 빨강, 녹색, 오렌지색 등이 개발되어 있다. 이 칩은 램프로 조립(LED)되어 각종 전자제품의 표시나 신호등 및 전광판 등의 광원으로 쓰인다.



1. Chip, 2. Leadframe, 3. Goldwire, 4. Epoxy, 5. Cathode, 6. Anode

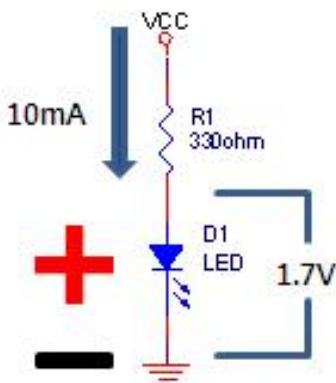
LED 구조

## [4] LED 부분별 명칭

LED는 반도체라는 특성으로 인해 처리속도, 전력소모, 수명 등의 제반사항에서 큰 장점을 보여 각종 전자제품의 전자표시부품으로 각광받고 있다. 그리고 높은 휘도의 제품들이 생산되면서 앞으로 조명기구의 역할도 대체할 수 있을 것이다.

기존 전구램프처럼 눈이 부시거나 엘러먼트가 단락되는 경우가 없는 LED는 소형으로 제작되어 각종 표시소자로 폭넓게 사용되고 있으며 반영구적인 수명(약 1백만 시간)으로 그 활용도가 높다.

특히 청색LED의 상용화로 LED의 풀 컬러 구현이 가능해지고 가격도 크게 낮출 수 있게 되면서 제품의 활용도는 급속히 높아질 전망이다.

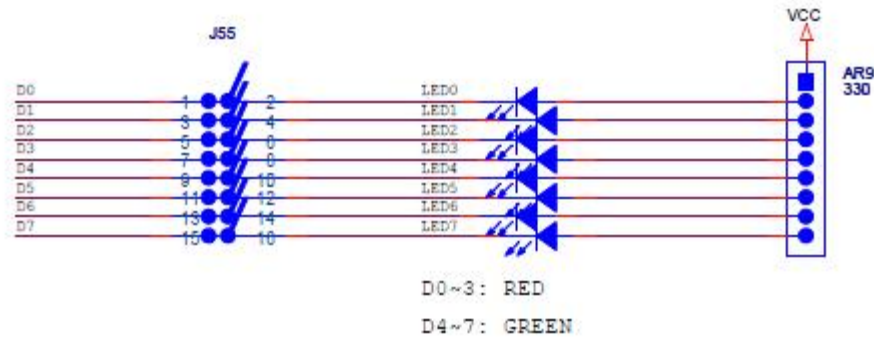
	$R = V / I$ $R = (VCC - 1.7V) / 10mA$ $R = (5V - 1.7V) / 10mA$ $R = 3.3V / 10mA$ $R = 3300mV / 10mA$ $R = 330$
기본 구동 회로	옴의 법칙에 의한 계산

LED 구동 회로

LED를 동작시키기 위해서는 일반적으로 1.7V의 전압과 10mA의 전류가 공급되어야 한다. 이러한 동작 특성을 가지고 있으므로 그림에서와 같은 회로에서 R값은 옴의 법칙에 의해서 위와 같다. 그러므로 330ohm을 직렬로 연결해야 한다.

## 2-3. LED 구성

### (1) LED 회로도



### (2) 케이블 결선도

장비와 같이 제공되는 USB ISP A/B 케이블을 이용하여 ATmega128 보드와 아래와 같이 연결한다.



### [3] 장비 결선도

아래와 같이 ATmega128 IO 포트와 LED를 연결 하도록 한다.

ATmega128 PORTA	LED 핀
0	D7
1	D6
2	D5
3	D4
4	D3
5	D2
6	D1
7	D0

## 2-4. LED 제어

### [1] 프로그램 순서

1-1. LED 제어를 위한 프로그램 작성

#### ■ Program Source

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

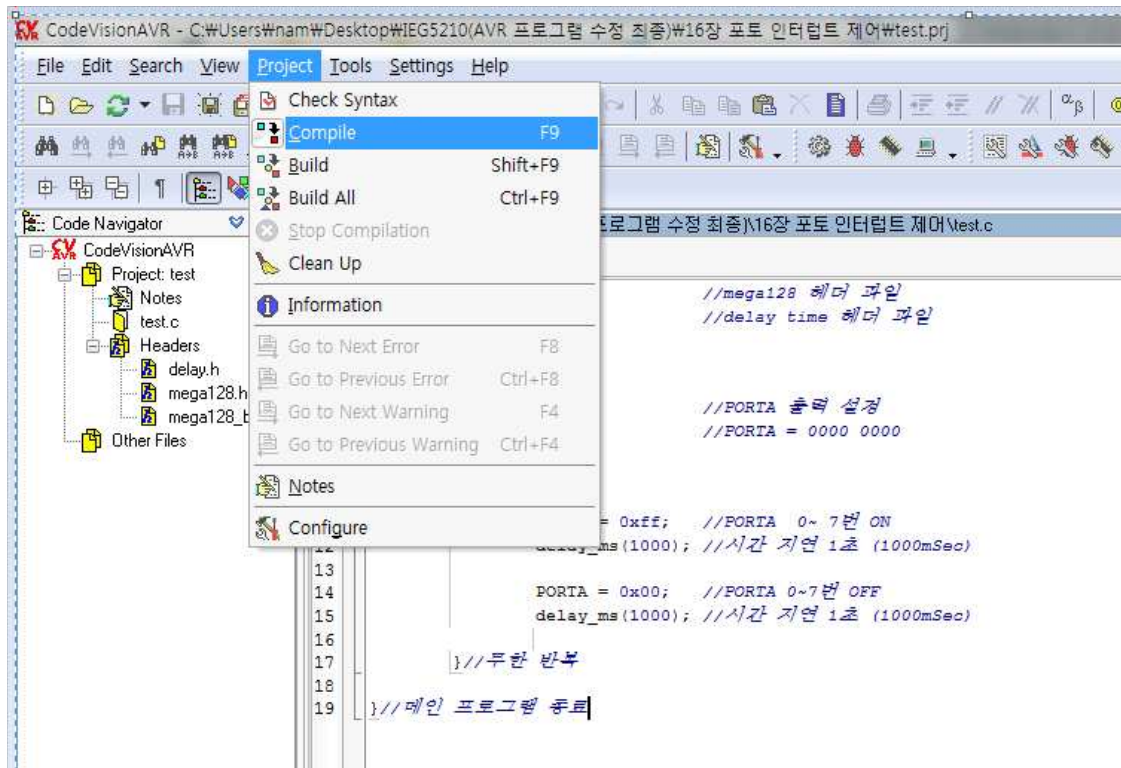
void main(void)
{
    DDRA = 0xff;    //DDR A 출력 설정 1111 1111
    PORTA = 0x00;    //PORT A 출력 0000 0000
    while(1)//반복
    {
        PORTA = 0xff; //PORTA = 1111 1111 출력
        delay_ms(500); //시간 지연 500ms Time

        PORTA = 0x00; //PORTA = 0000 0000 출력
        delay_ms(500); //시간 지연 500ms Time

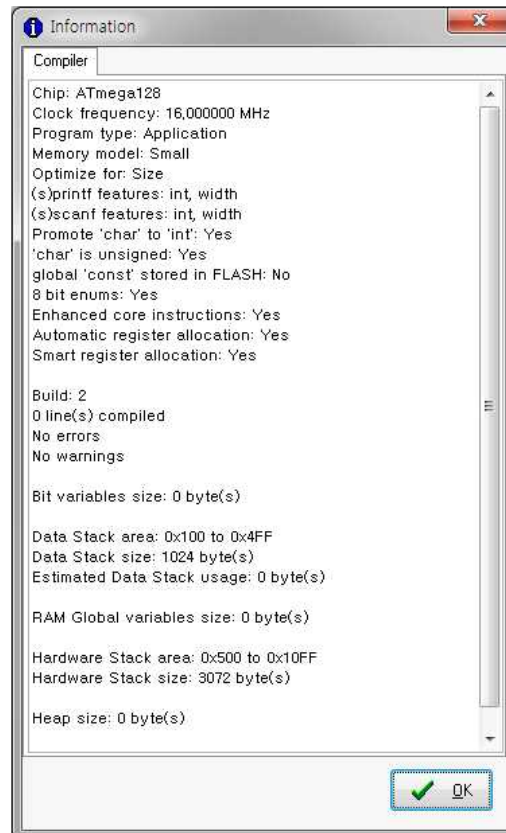
    } //while end
} //main program end
```

## [2] CodeVision 프로그램 컴파일

- CodeVision 프로그램에서 Project 메뉴를 선택하여 Compile을 실행 하도록 한다.

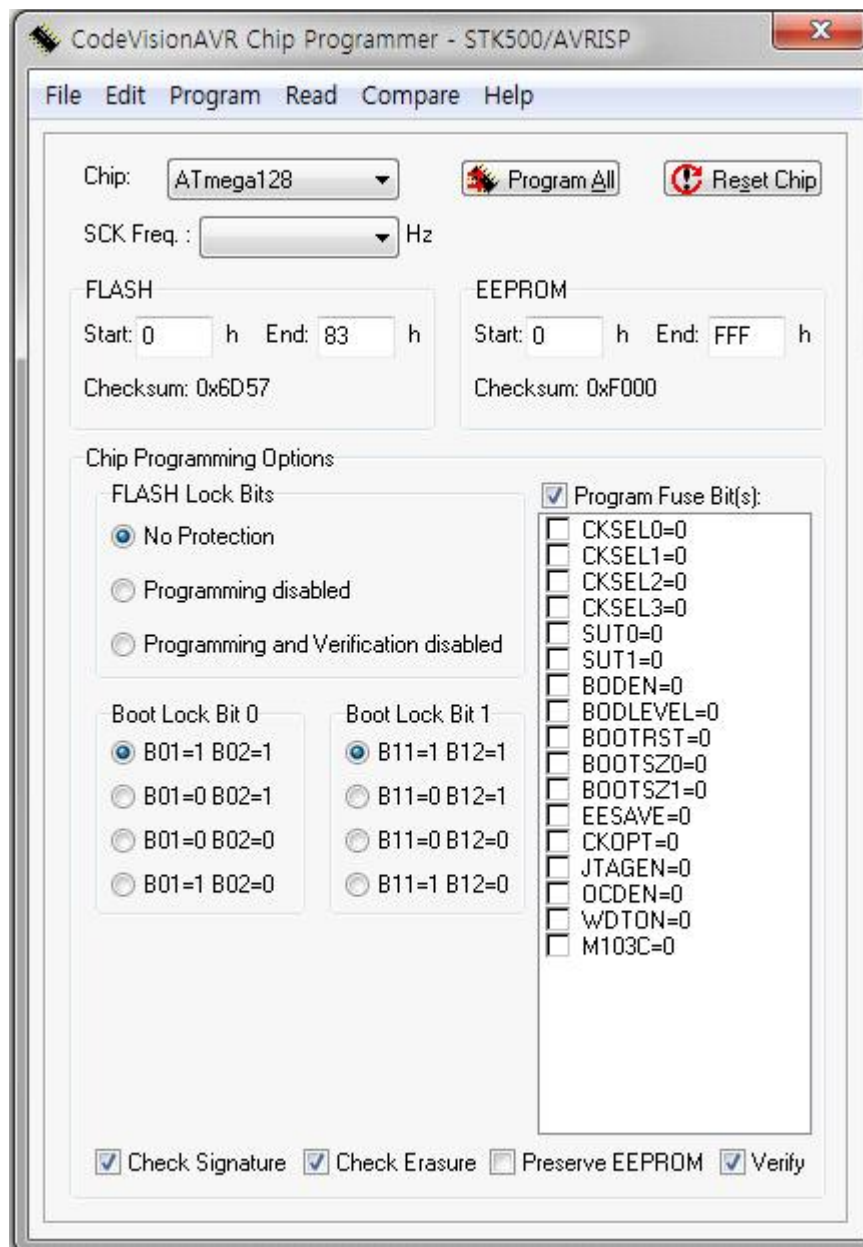


- 프로그램에서 에러가 없으면 아래와 같이 컴파일 정보가 출력 된다.

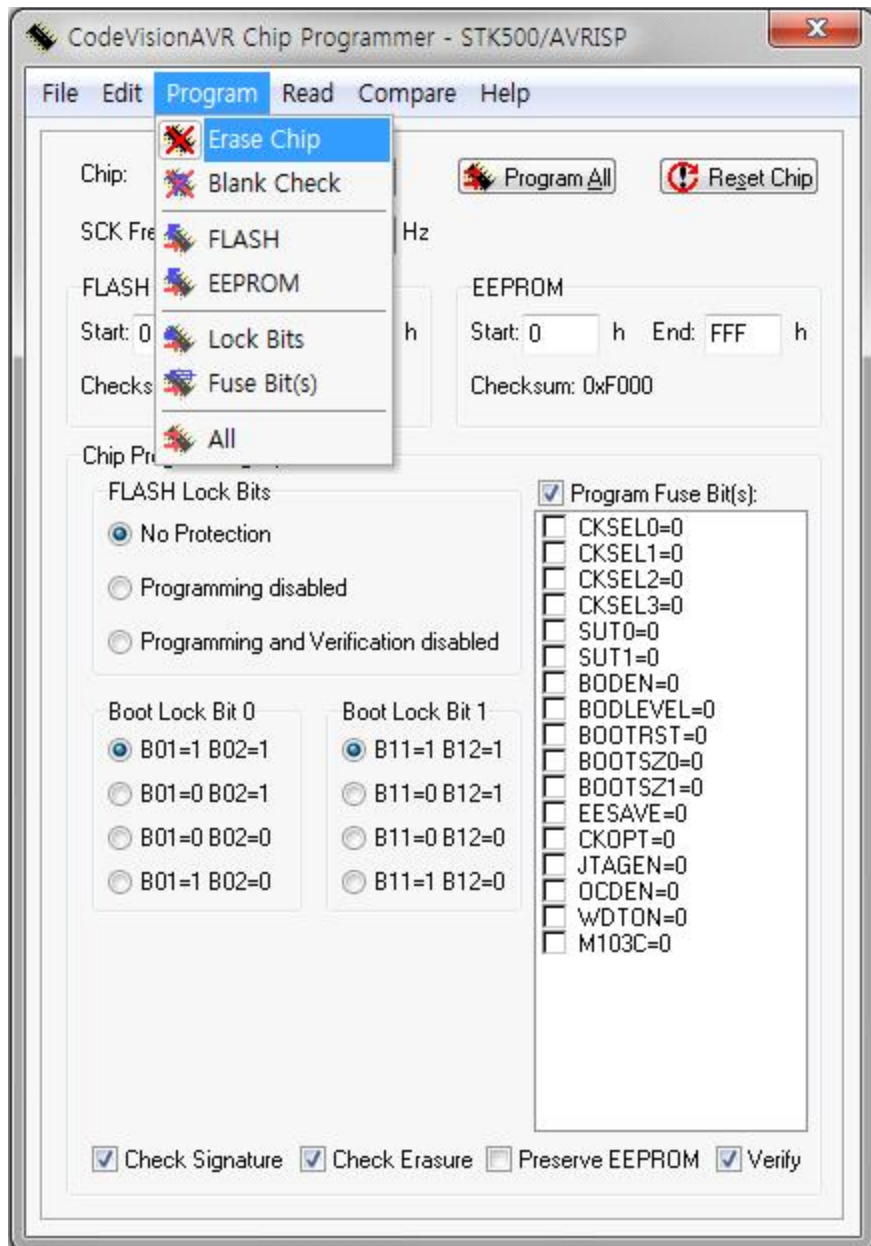




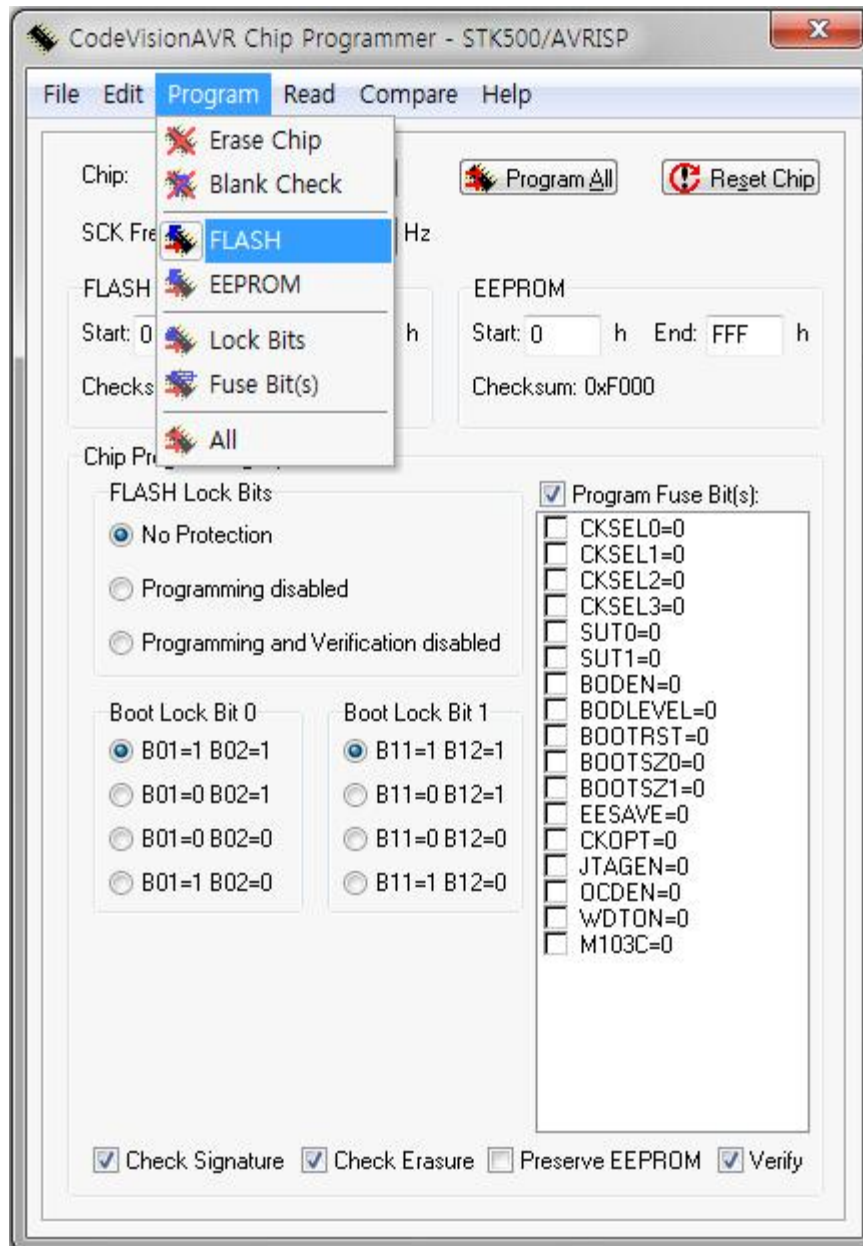
- Tools → Chip Programmer 클릭(단축키 : Shift+F4)



- Erase chip 클릭(chip 내에 프로그램이 남아 있을수 있으므로 프로그램 다운로드 전에 반드시 Erase Chip을 권장하도록 한다. )

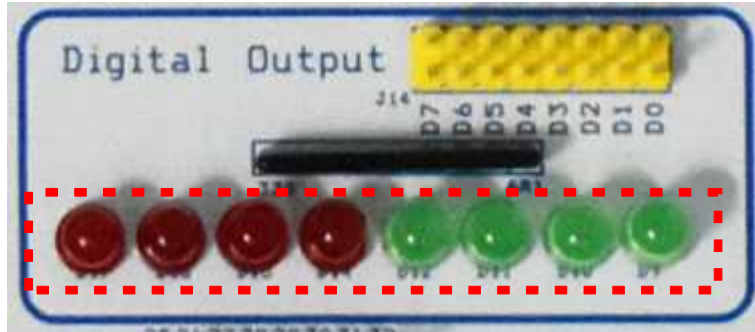


- Chip Flash 클릭하여 프로그램을 다운로드 하도록 한다.



### [3] 동작 확인

- 작성한 프로그램에 따라서 LED가 0.5초 간격으로 ON/OFF 반복되는 것을 확인 할 수 있다.



## [4] 실습 과제

### ☐ LED 제어 포트 : PORTA

4-1. 과제명 : LED 8개를 1초 간격으로 ON/OFF 제어

#### ■ Program Source

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

void main(void)
{
    DDRA = 0xff;    //DDR A 출력 설정 1111 1111
    PORTA = 0x00;    //PORT A 출력 0000 0000
    while(1)//반복
    {
        PORTA = 0xff; //PORTA = 1111 1111 출력
        delay_ms(1000); //시간 지연 1000ms Time

        PORTA = 0x00; //PORTA = 0000 0000 출력
        delay_ms(1000); //시간 지연 1000ms Time

    } //while end
} //main program end
```

## 4-2. 과제명 : LED 8개를 오른쪽으로 1개씩 이동 쉬프트 제어

- 1개씩 ON/OFF 하여 우측으로 이동 하도록 한다. (시간 간격은 1초 단위)

## ■ Program Source

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

void main(void)
{
    unsigned char buff;//DATA 저장을 위한 BUFF 변수

    DDRA = 0xff;    //DDR A 출력 설정 1111 1111
    PORTA = 0x00;   //PORT A 출력 0000 0000

    buff = 0x00;    //buff fast data Setting

    while(1)//반복
    {
        PORTA = buff; //buff data를 porta로 출력한다.
        PORTB = buff; //buff data를 porta로 출력한다.

        if(buff == 0xff) //만약 전체 소등이 될 경우

        {
            buff = 0x00; //led 전부 on

        }

        //if program end

        else//전체 소등이 아니라면
        {
            buff = buff<<1;//1bit 좌측 쉬프트 한다
            buff = buff | 0x01;//저장된 쉬프트 data를 or 연산후
buff로 저장
        }

        delay_ms(1000);//시간 지연 함수 1000ms Time

    }

    //while end

}

//main program end
```

## 4-3. 과제명 : LED 8개를 좌측으로 1개씩 이동 쉬프트 제어

- 1개씩 ON/OFF 하여 좌측으로 이동 하도록 한다. (시간 간격은 1초 단위)

## ■ Program Source

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

void main(void)
{
    unsigned char buff;//DATA 저장을 위한 BUFF 변수

    DDRA = 0xff;    //DDR A 출력 설정 1111 1111
    PORTA = 0x00;   //PORT A 출력 0000 0000

    buff = 0x00;    //buff fast data Setting

    while(1)//반복
    {
        PORTA = buff; //buff data를 porta로 출력한다.
        PORTB = buff; //buff data를 porta로 출력한다.

        if(buff == 0xff) //만약 전체 소등이 될 경우

        {
            buff = 0x00; //led 전부 on

        }

        //if program end

        else//전체 소등이 아니라면
        {
            buff = buff>>1;//1bit 좌측 쉬프트 한다
            buff = buff | 0x80;//저장된 쉬프트 data를 or 연산후
buff로 저장
        }

        delay_ms(1000);//시간 지연 함수 1000ms Time

    }

    //while end

}

//main program end
```

## SECTION

# 03 스위치 제어 실습

## 3-1. 학습 목표

- 스위치 구조와 동작 방법
- 스위치를 이용한 LED 출력
- 스위치를 이용한 LED 제어

## 3-2. 기초 이론

### 1) Tick Tack 스위치

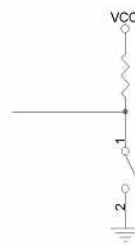
- Kit에서 스위치의 기본 값은 Logic 'H'이며, 스위치를 누르면 Logic 'L'가 된다.



부품 외형



회로도 기호



구동 회로

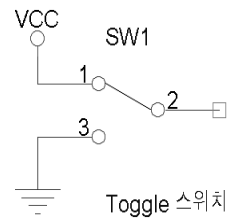


## 2) Toggle 스위치

- 스위치 특성 중 상태 유지를 위해 쓰이는 스위치이다. Kit에서 스위치는 아래로 내리면 Logic 'L', 위로 올리면 Logic 'H'가 연결된다.



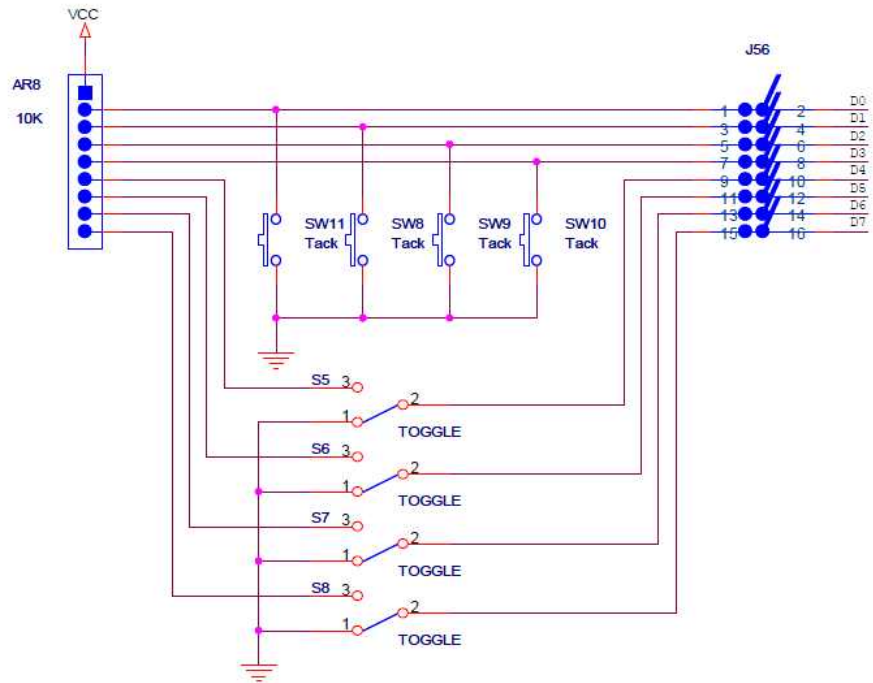
부품 외형



구동 회로

### 3-3. 스위치 구성

#### [1] 스위치 회로도



#### [2] 케이블 결선도

장비와 같이 제공되는 USB ISP A/B 케이블을 이용하여 ATmega128 보드와 아래와 같이 연결한다.



### [3] 장비 결선도

아래와 같이 ATmega128 IO 포트와 스위치를 연결 하도록 한다.

ATmega128 PORTC	SW 핀
7	SW7
6	SW6
5	SW5
4	SW4
3	SW3
2	SW2
1	SW1
0	SW0

ATmega128 PORTA	LED 핀
7	D7
6	D6
5	D5
4	D4
3	D3
2	D2
1	D1
0	D0

## 3-4. 스위치 제어

### [1] 프로그램 순서

1-1. 스위치 입력을 받아 PORTA(LED)로 출력하기 위한 프로그램 작성

#### ■ Program Source

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

void main(void)
{
    DDRA = 0xff;    //DDR A 출력 설정 1111 1111
    PORTA = 0x00;    //PORT A 출력 0000 0000

    DDRC = 0x00;    //DDR C 입력 설정 0000 0000
    PORTC = 0x00;    //PORT C 입력 0000 0000

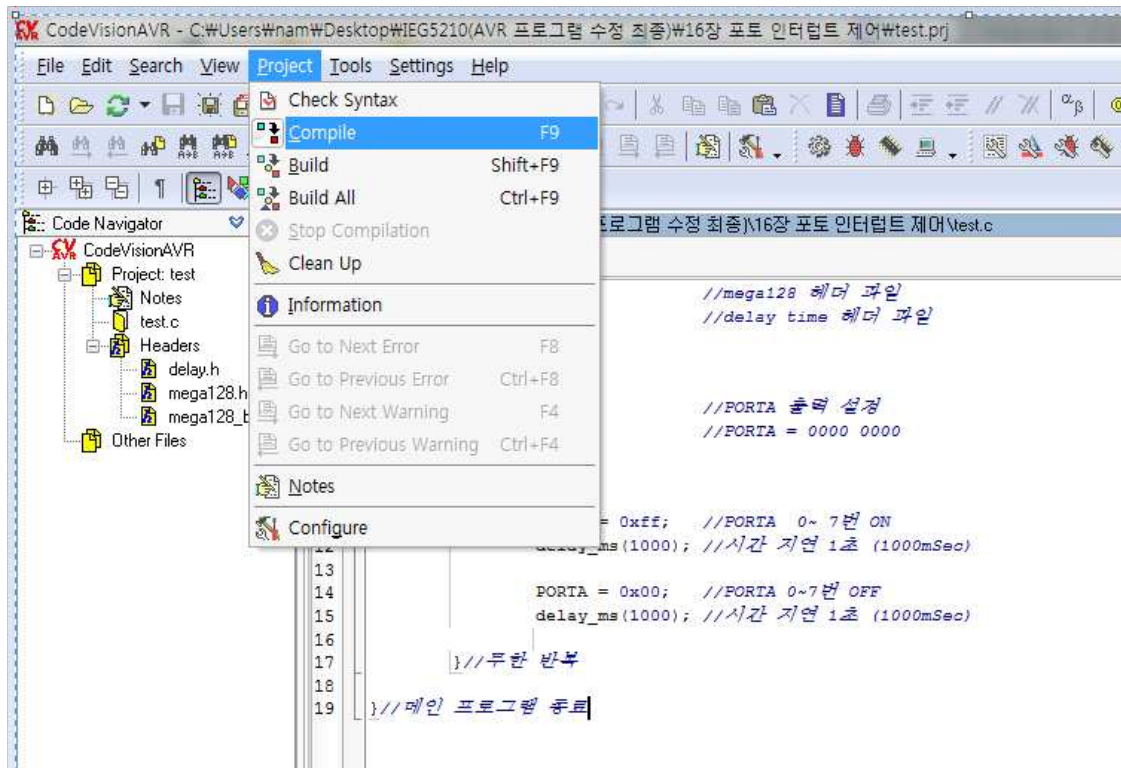
    while(1)//반복
    {
        PORTA = ~PINC;//PORTC에서 스위치를 ON시 LED를 ON
        delay_ms(50);//스위치 입력 시간 지연 함수 50ms Time

    }//while end
}

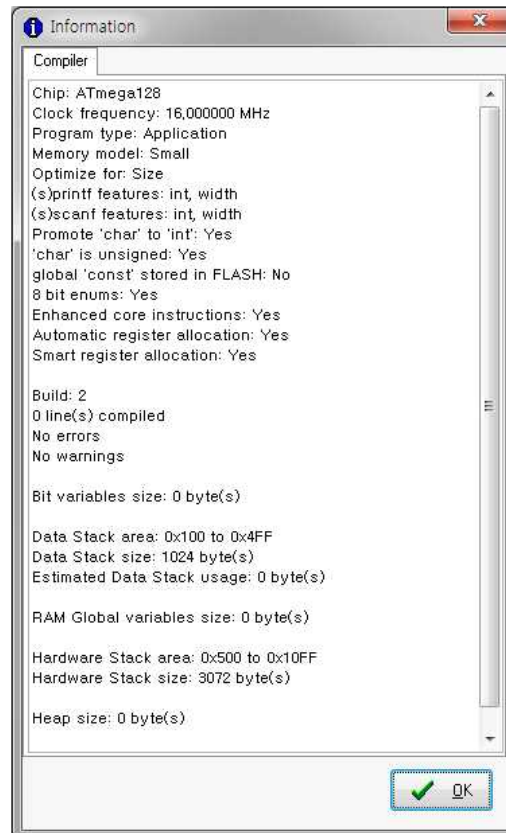
//main program end
```

## [2] CodeVision 프로그램 컴파일

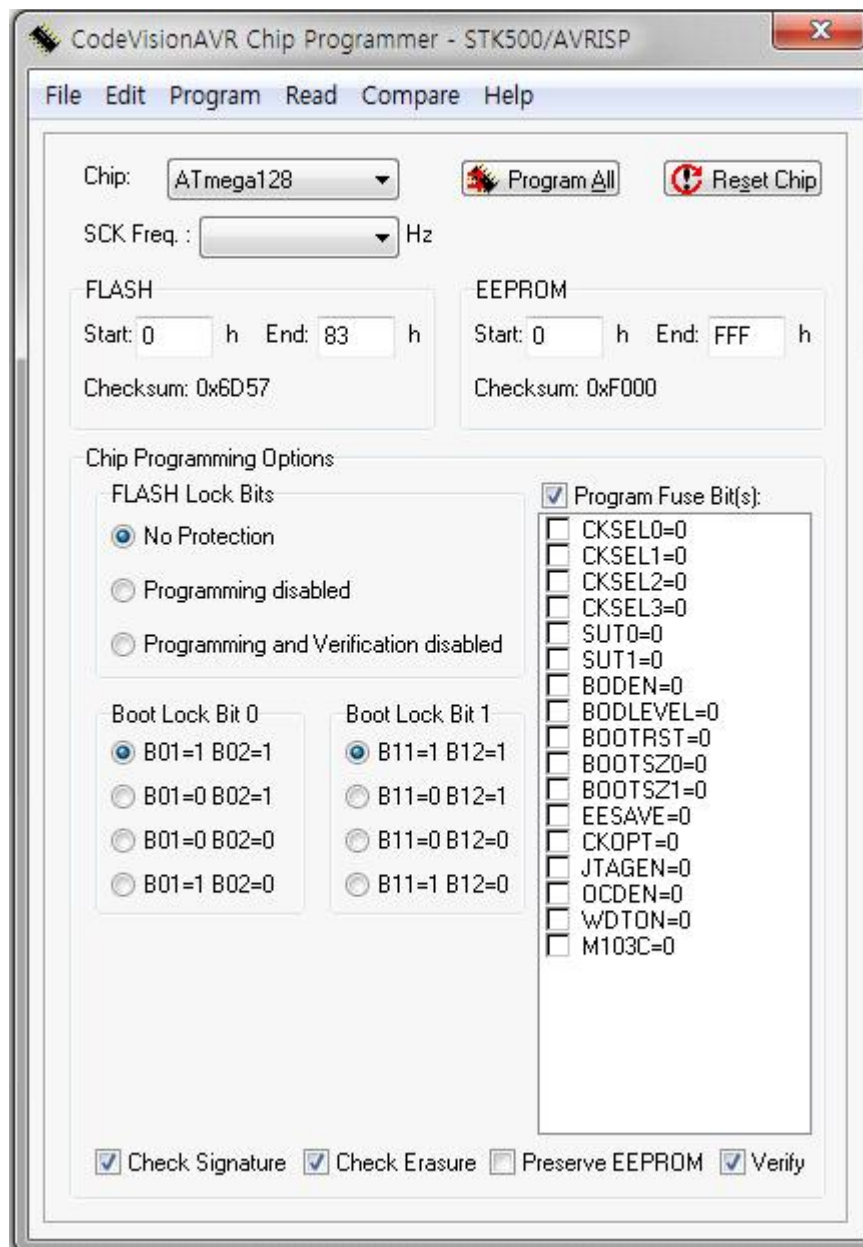
- CodeVision 프로그램에서 Project 메뉴를 선택하여 Compile을 실행 하도록 한다.



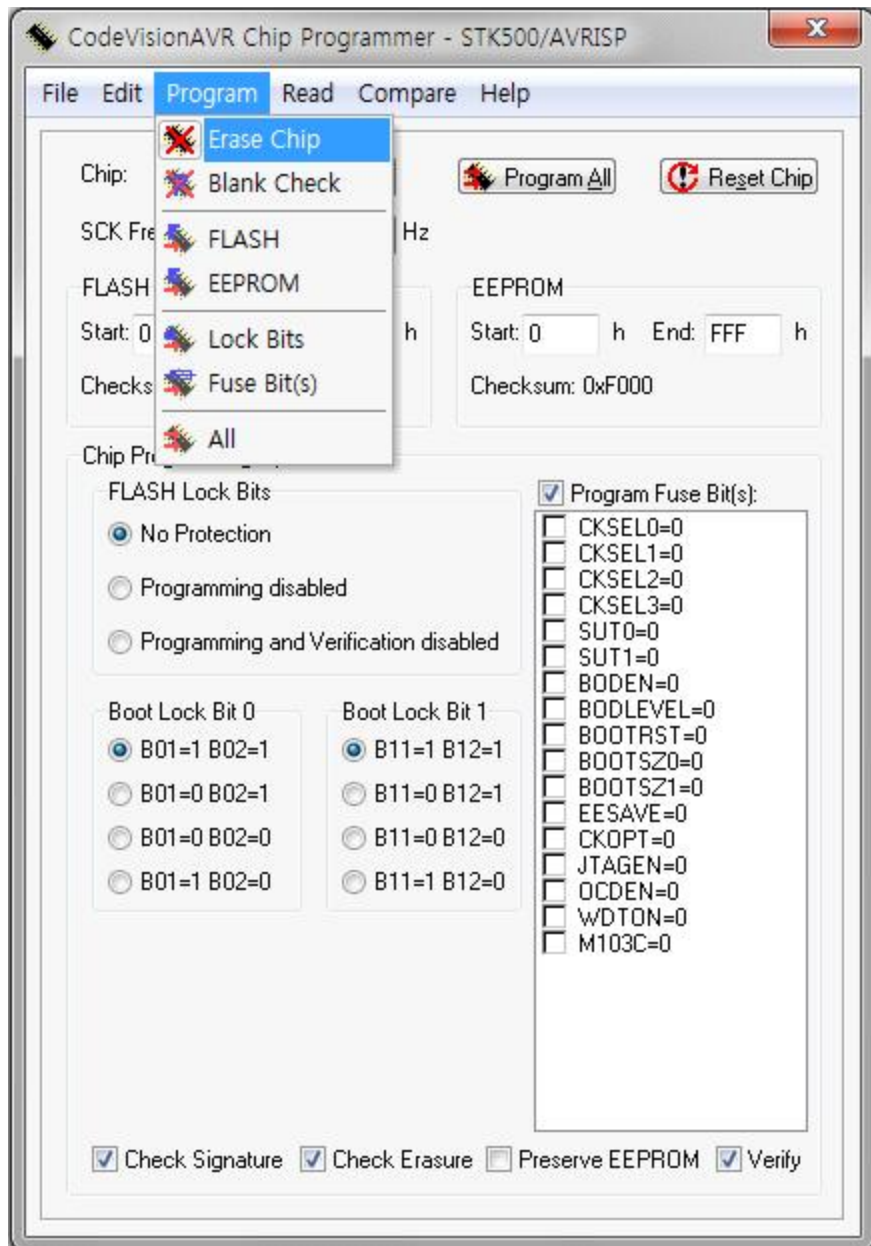
- 프로그램에서 에러가 없으면 아래와 같이 컴파일 정보가 출력 된다.



- Tools → Chip Programmer 클릭(단축키 : Shift+F4)

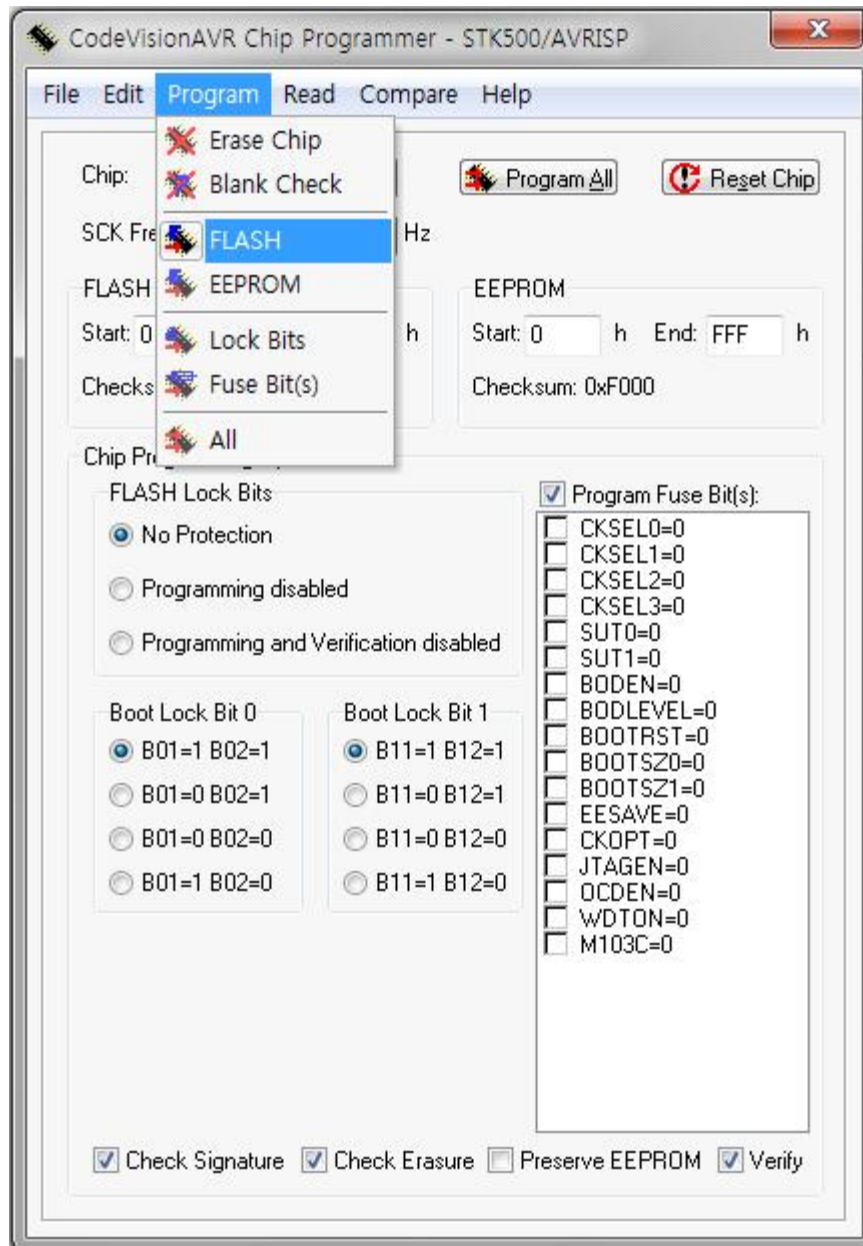


- Erase chip 클릭(chip 내에 프로그램이 남아 있을수 있으므로 프로그램 다운로드 전에 반드시 Erase Chip을 권장하도록 한다. )



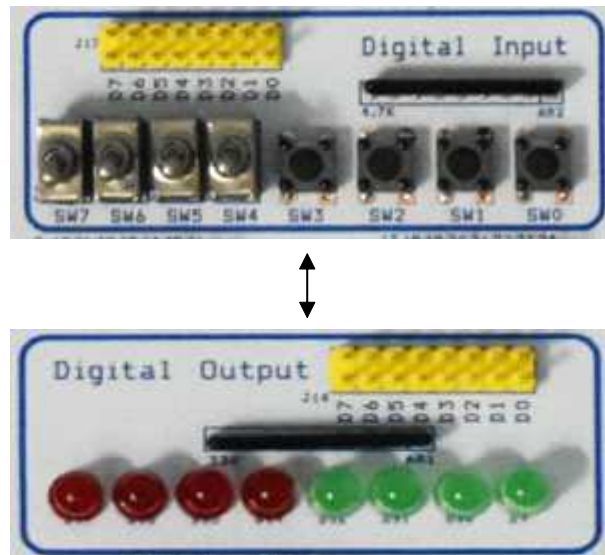


- Chip Flash 클릭하여 프로그램을 다운로드 하도록 한다.



### [3] 동작 확인

- 스위치 입력 신호에 따라서 LED에 출력을 되는 것을 확인 할수 있다.



### [4] 실습 과제

- ☐ 스위치 Input 포트 : PORTC
- ☐ LED Output 포트 : PORTA

- 4-1. 과제명 : 스위치 1번에 "HIGH" 가 입력 되면 LED 8개가 1초 간격으로 ON/OFF 제어
- 4-2. 과제명 : 스위치 2번에 "HIGH" 가 입력 되면 LED 가 오른쪽으로 1개씩 ON 출력 반복(단 8개의 LED가 전부 ON이면 OFF 후 반복)
- 4-3. 과제명 : 스위치 3번에 "HIGH" 가 입력 되면 LED 가 좌측으로 1개씩 ON 출력 반복(단 8개의 LED가 전부 ON이면 OFF 후 반복)

## SECTION

# 04 Segment 제어 실습

## 4-1. 학습 목표

- Segment 구조와 동작 방법
- Segment 제어
- Segment 숫자 출력
- Segment 숫자 카운트 제어

## 4-2. 기초 이론

### (1) Segment 란?

Digital 회로에서 정보를 Display하는 방법에는 여러 가지가 있다.

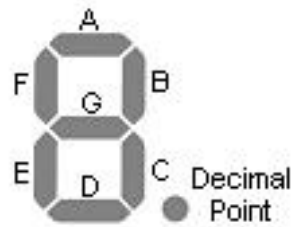
그 중 가장 보편적인 방법으로는 LED를 사용하는 것이다. 하지만 LED로는 많은 정보를 쉽게 표현할 수가 없다는 단점이 있다. 그래서 만들어진 Device가 7-Segment이다.

7-Segment는 실생활에서 많이 사용되는 Display Device로 엘리베이터에서 층을 나타내는 숫자 표시기 등이 있다.

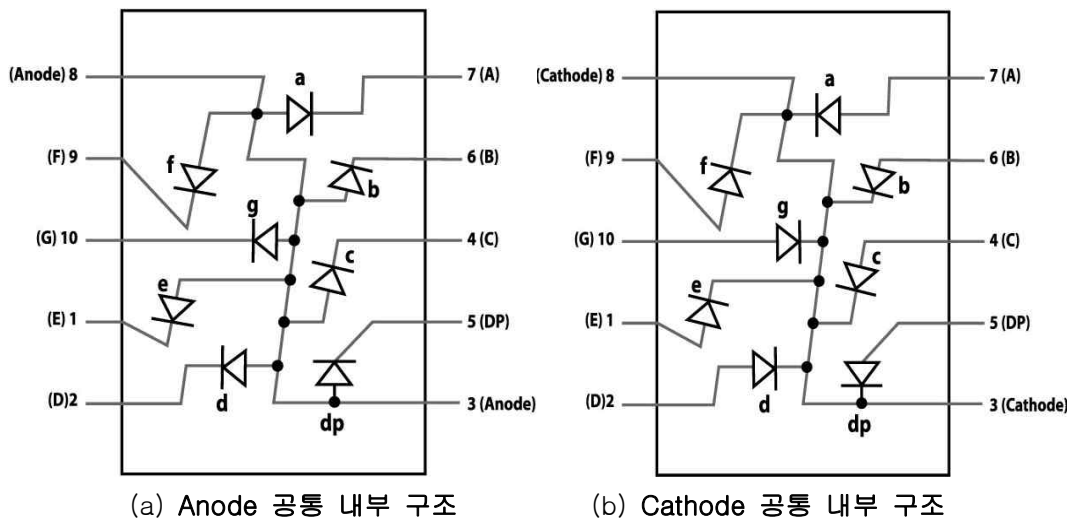
7-Segment의 외형과 구조는 다음과 같다.



7-Segment 외형



[그림 II-3-2] 7-Segment 배열



7-Segment 외형, 배열, 내부 구조

간단하게 말해 8개의 LED가 하나의 Device를 이루는 것이다.

사용자가 7-Segment의 표시를 '0'으로 하려면 「Segment 배열」에서 a, b, c, d, e, f의 LED는 점등, g, dot의 LED는 소등 시키면 된다. 다시 말해, 양극 공통(Anode Common) 7-Segment에서 a, b, c, d, e, f의 LED에 로직 '0'을 출력하면 LED는 점등, g, dot의 LED에 로직 '1'을 출력하면 LED는 소등된다.

## (2) Infrared LED chip

IRED(Infrared Emitting Diode)Chip을 뜻하는 것이다. 이 칩은 가공되어 IR Emitter, TV 리모컨, 광학스위치, IR LAN, 무선 디지털 데이터 통신용 모듈 등에 쓰인다.

Segment Hex 출력 Data (Anode Common)

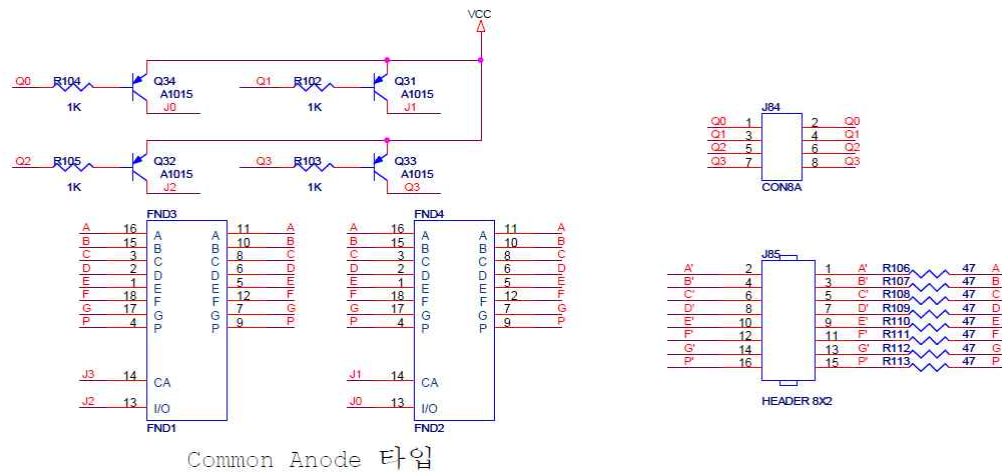
Display	HEX	dot	g	f	e	d	c	b	a
0	0xc0	1	1	0	0	0	0	0	0
1	0xf9	1	1	1	1	1	0	0	1
2	0xa4	1	0	1	0	0	1	0	0
3	0xb0	1	0	1	1	0	0	0	0
4	0x99	1	0	0	1	1	0	0	1
5	0x92	1	0	0	1	0	0	1	0
6	0x82	1	0	0	0	0	0	1	0
7	0xd8	1	1	1	0	1	0	0	0
8	0x80	1	0	0	0	0	0	0	0
9	0x98	1	0	0	1	1	0	0	0
A	0x88	1	0	0	0	1	0	0	0
B	0x83	1	0	0	0	0	0	1	1
C	0xc6	1	1	0	0	0	1	1	0
D	0xa1	1	0	1	0	0	0	0	1
E	0x86	1	0	0	0	0	1	1	0
F	0x8e	1	0	0	0	1	1	1	0
.	0x7f	0	1	1	1	1	1	1	1

Segment Hex 출력 Data (Cathode Common)

Display	HEX	dot	g	f	e	d	c	b	a
0	0x3f	0	0	1	1	1	1	1	1
1	0x06	0	0	0	0	0	1	1	0
2	0x5b	0	1	0	1	1	0	1	1
3	0x4f	0	1	0	0	1	1	1	1
4	0x66	0	1	1	0	0	1	1	0
5	0x6d	0	1	1	0	1	1	0	1
6	0x7d	0	1	1	1	1	1	0	1
7	0x17	0	0	0	1	0	1	1	1
8	0x7f	0	1	1	1	1	1	1	1
9	0x67	0	1	1	0	0	1	1	1
A	0x77	0	1	1	1	0	1	1	1
B	0x7c	0	1	1	1	1	1	0	0
C	0x39	0	0	1	1	1	0	0	1
D	0x5e	0	1	0	1	1	1	1	0
E	0x79	0	1	1	1	1	0	0	1
F	0x71	0	1	1	1	0	0	0	1
.	0x80	1	0	0	0	0	0	0	0

## 4-3. Segment 구성

### [1] Segment 회로도



### [2] 케이블 결선도

장비와 같이 제공되는 USB ISP A/B 케이블을 이용하여 ATmega128 보드와 아래와 같이 연결한다.



### [3] 장비 결선도

아래와 같이 ATmega128 IO 포트와 Segment를 연결 하도록 한다.

ATmega128 PORTB	Segment Enable 핀
3	Q3
2	Q2
1	Q1
0	Q0

ATmega128 PORTA	Segment 핀
7	DP
6	G
5	F
4	E
3	D
2	C
1	B
0	A



## 4-4. Segment 제어

### [1] 프로그램 순서

1-1. Segment 숫자 출력을 위한 프로그램 작성

#### ■ Program Source

```
#include <mega128.h> // ATmega128 header file
#include <delay.h>

#define Q0    PORTB.0           //PortB.0 bit FND Q0
#define Q1    PORTB.1           //PortB.1 bit FND Q1
#define Q2    PORTB.2           //PortB.2 bit FND Q2
#define Q3    PORTB.3           //PortB.3 bit FND Q3
#define FND    PORTA           //Port A를 FND 설정

void main(void)
{
    // FND data Array 0 ~ F
    // data value '0' '1' '2' '3' '4' '5' '6' '7' '8' '9' 'A' 'B' 'C' 'D' 'E' 'F' 'dot'
    unsigned char fnd[17]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xd8,0x80,0x98,0x88,0x83,0xc6,0xa1,0x86,0x8e,0x7f};

    unsigned char i;//fnd 출력 변수

    PORTA = 0x00;           //PORT A 초기화
    DDRA = 0xff;            //Port A 설정, 출력으로 사용

    PORTB = 0x00;           //Port B 초기화
    DDRB = 0xff;            //Port B 설정, 출력으로 사용

    while(1)
    {
        for(i=0;i<10;i++) // FND data Array Fetch
        {
            FND = fnd[i]; // FND data display '0' ~ '9'

            Q0 = 0;         // PORTB.0 출력, Q0에 0을 출력
            Q1 = 0;         // PORTB.1 출력, Q1에 0을 출력
            Q2 = 0;         // PORTB.2 출력, Q2에 0을 출력
            Q3 = 0;         // PORTB.3 출력, Q3에 0을 출력
        }
    }
}
```

## ■ Program Source

```
        delay_ms(1000); // 시간지연함수 호출

        Q0 = 1;        // PORTB.0 출력, Q0에 1을 출력
        Q1 = 1;        // PORTB.1 출력, Q1에 1을 출력
        Q2 = 1;        // PORTB.2 출력, Q2에 1을 출력
        Q3 = 1;        // PORTB.3 출력, Q3에 1을 출력

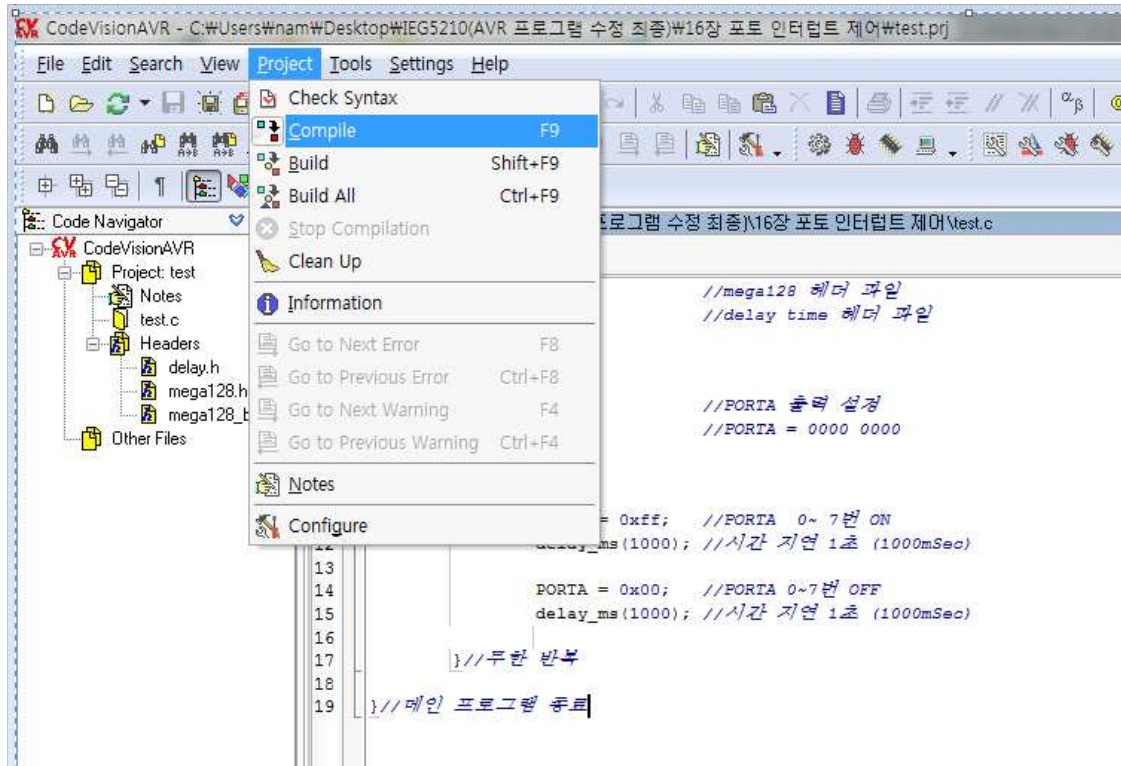
    }//for program end

} //while program end

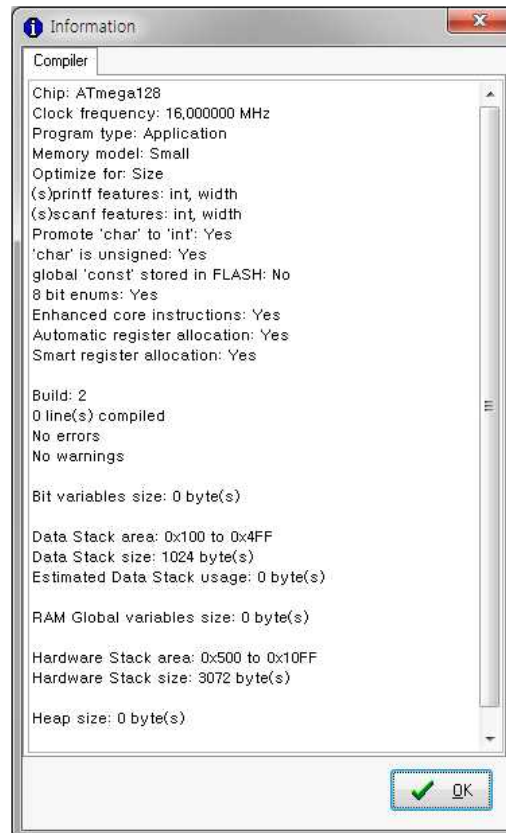
} //main program end
```

## [2] CodeVision 프로그램 컴파일

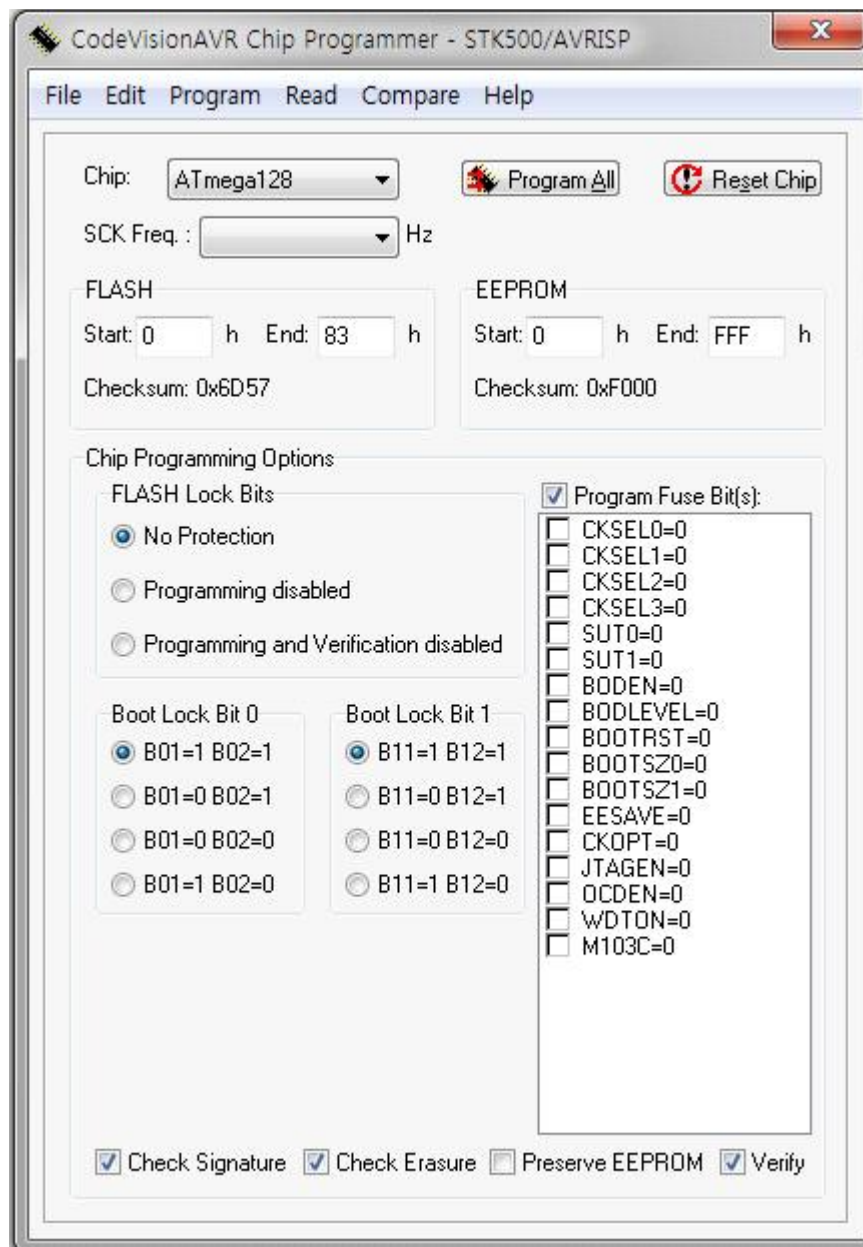
- CodeVision 프로그램에서 Project 메뉴를 선택하여 Compile을 실행 하도록 한다.



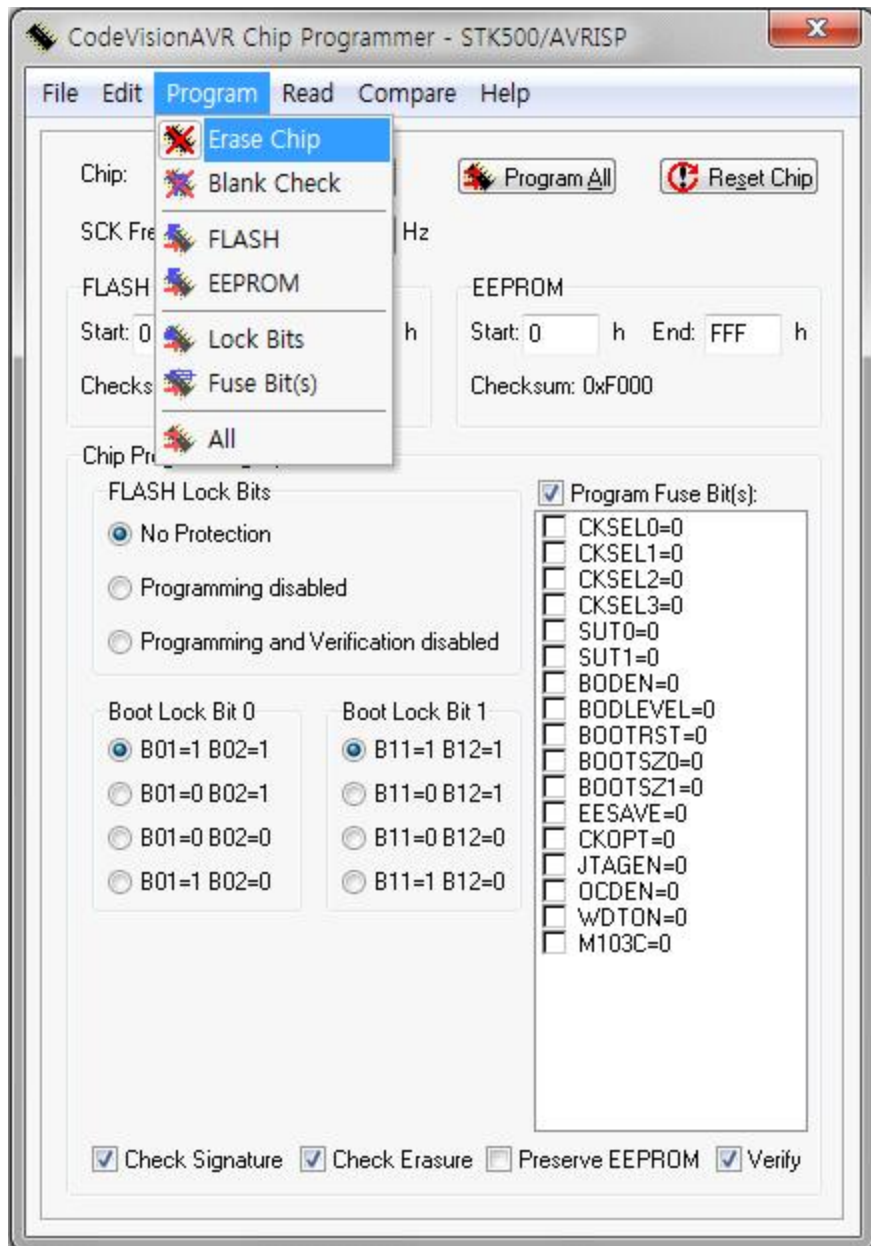
- 프로그램에서 에러가 없으면 아래와 같이 컴파일 정보가 출력 된다.



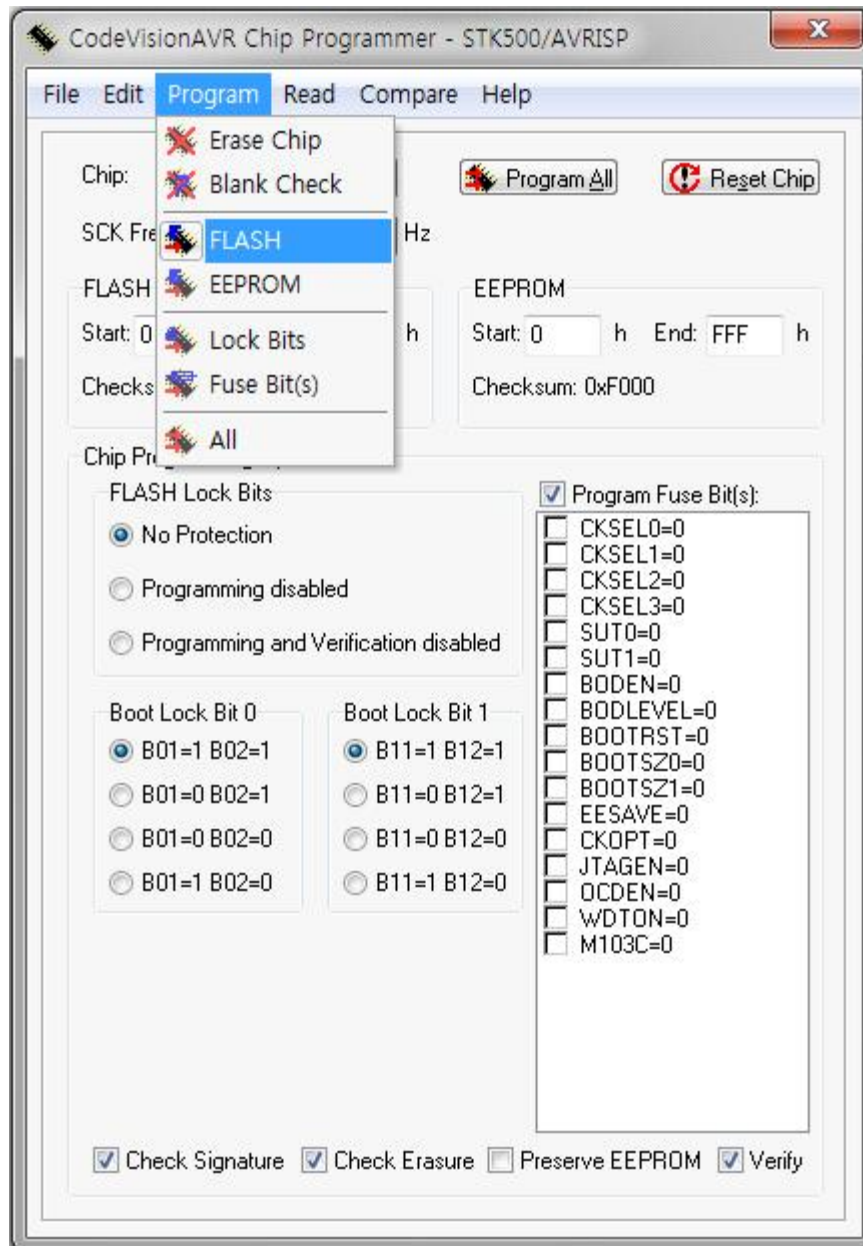
- Tools → Chip Programmer 클릭(단축키 : Shift+F4)



- Erase chip 클릭(chip 내에 프로그램이 남아 있을수 있으므로 프로그램 다운로드 전에 반드시 Erase Chip을 권장하도록 한다. )



- Chip Flash 클릭하여 프로그램을 다운로드 하도록 한다.



### [3] 동작 확인

- 작성한 프로그램에 따라서 1초 간격으로 Segment에 0~9까지 Display가 반복되는 것을 확인 할 수 있다.



### [4] 실습 과제

☐ Segment Data 포트 : PORTA

☐ Segment Control 포트 : PORTB

4-1. 과제명 : Segment 4개를 1초 간격으로 0 ~ 9까지 Display 출력

4-2. 과제명 : Segment 1 ~ 4번을 1초 간격으로 0 ~ 9 까지 각각 Display 출력

4-3. 과제명 : 0~9999 까지 증가하는 값을 Segment에 Display 출력



## SECTION

# 05 Character LCD 제어 실습

## 5-1. 학습 목표

- Character LCD 구조와 동작 방법
- Character LCD 제어
- Character LCD Display 출력
- Character LCD 문자 출력

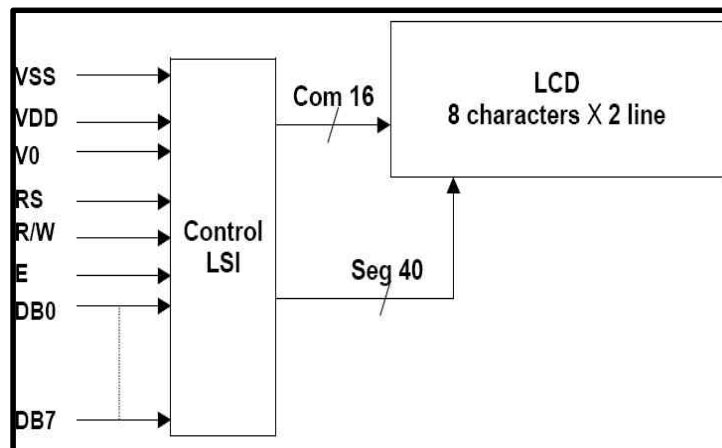
## 5-2. 기초 이론

### (1) Character LCD 란?

Digital 회로에서 정보를 Display하는 방법에는 여러 가지가 있다. 그중 가장 보편적인 방법으로는 LED(Light Emitting Diode)와 7-Segment, DOT Matrix를 사용하는 것이다.

그러나 위의 방법으로는 보다 많은 정보를 표현할 수가 없다. 그래서 만들어진 Device가 LCD 이다. 하지만 LCD는 LED처럼 스스로 발광하는 능력을 가지고 있지 않아서 주위의 밝기가 낮은 곳에서는 사용하기 어렵고, LCD를 보는 각도에 따라서 상태가 다르다. 그래서 백라이트가 내장된 제품들이 개발, 생산되고 있다.

아래는 LCD 내부 구조를 나타낸 그림이다.



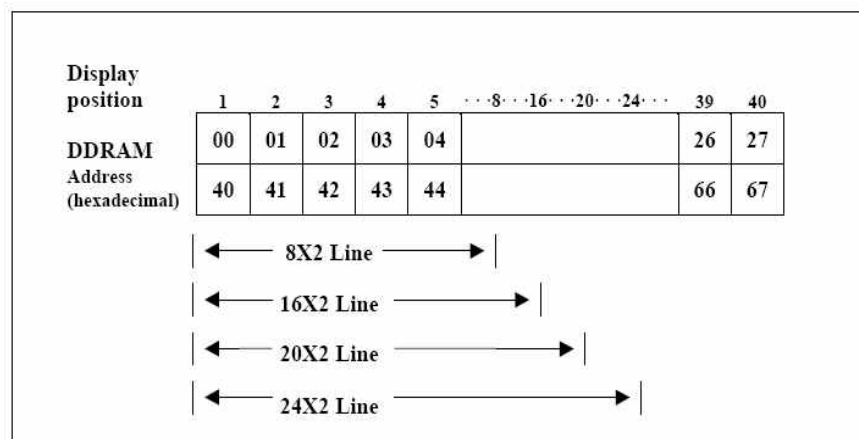
LCD 블록 다이어그램

Char. code

xxxx0000			0	0	P	^	P		-	9	3	α	P
xxxx0001		!	1	A	Q	a	q	.	ア	チ	△	ä	Q
xxxx0010		"	2	B	R	b	r	「	イ	ツ	×	ß	θ
xxxx0011		#	3	C	S	c	s	」	ウ	テ	モ	ε	ω
xxxx0100		\$	4	D	T	d	t	、	エ	ト	ヤ	μ	Ω
xxxx0101		%	5	E	U	e	u	.	オ	ナ	ユ	ü	Ü
xxxx0110		&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111		'	7	G	W	g	w	ア	キ	ヌ	ラ	Q	π
xxxx1000		(	8	H	X	h	x	イ	ク	ネ	リ	フ	Σ
xxxx1001		)	9	I	Y	i	y	ッ	ケ	ル	リ	ウ	
xxxx1010		*	:	J	Z	j	z	エ	コ	ハ	レ	i	チ
xxxx1011		+	;	K	[	k	[	オ	サ	ヒ	ロ	×	π
xxxx1100		,	<	L	¥	l	l	ハ	シ	フ	ワ	Φ	π
xxxx1101		-	=	M	]	m	]	ユ	ズ	ヘ	ン	も	÷
xxxx1110		.	>	N	^	n	^	ヨ	セ	ホ	ッ	ん	
xxxx1111		/	?	O	_	o	_	ッ	ッ	マ	マ	ö	■

내장된 CGRAM 의 문자표 패턴

장비에서 사용되는 16 × 2 Character Type LCD 가로 16행, 세로 2열에 32개의 문자판으로 구성되어 있으며, 백라이트가 내장된 제품이다



LCD크기에 따른 출력위치 DDRAM 주소 값

GCRAM 어드레스와 DDRAM 패턴의 및 패턴은 다음과 같다.

Character Codes (DDRAM data)								CGRAM Address								Character Patterns (CGRAM data)									
7	6	5	4	3	2	1	0	5	4	3	2	1	0	7	6	5	4	3	2	1	0				
High				Low				High				Low				High				Low					
0 0 0 0 * 0 0 0								0 0 0								* * *								}	Character pattern (1)
																1 1 1 1 0									
																1 0 0 0 1									
																1 0 0 0 1									
																1 1 1 1 0									
																1 0 1 0 0									
																1 0 0 1 0									
																1 0 0 0 1									
0 0 0 0 * 0 0 1								0 0 1								* * *								}	Cursor position
																1 0 0 0 1									
																0 1 0 1 0									
																1 1 1 1 1									
																0 0 1 0 0									
																1 1 1 1 1									
																0 0 1 0 0									
																0 0 1 0 0									
0 0 0 0 * 1 1 1								1 1 1								* * *								}	Cursor position
																1 0 0									
																1 0 1									
																1 1 0									
																1 1 1									
																1 1 1									
																1 1 1									
																1 1 1									

LCD 패널과 LCD controller 및 driver가 일치 형으로 내장된 제품으로서, 문자를 이루는 DOT는 5×8로서 ASCII 문자를 출력한다.

LCD controller의 내부에는 DISPLAY 데이터를 저장하는 DDRAM과, DISPLAY 데이터 코드를 표시할 문자 폰트로 변환하는 CG ROM, 사용자 정의 문자를 저장하는 CG RAM등이 내장되어 있다.

그 외에도 LCD를 초기화하거나 제어하는데 사용되는 명령을 DECODE할 수 있는 DECODER가 있으며, 각 문자의 폰트를 시프트 할 수 있는 시프트 레지스터가 있어서 좌, 우로 움직일 수 있다. 또한 BUSY FLAG를 출력하여 LCD 모듈의 사용여부를 확인할 수 있다.

## 배열 및 역할

핀 번호	기호	기능	
1	V <sub>SS</sub>	0V	전원
2	V <sub>DD</sub>	5V	
3	V <sub>L</sub>	3-13V	CONTRAST CONTROL
4	RS	H: 데이터 입력, L: 인스트럭션 입력	
5	R/W	H: 데이터 리드, L: 데이터 라이트	
6	E1	H: D; SD; D[Q, F T;SG]	
7	D0	데이터 버스 4비트 인터페이스 모드에서는 DB4 - DB7을 사용한다.	
8	D1		
9	D2		
10	D3		
11	D4		
12	D5		
13	D6		
14	D7		
15	A	LCD 백라이트 전원	
16	K		

LCD 명령어 표

명령어	데이터								실행 시간	기능
	D7	D6	D5	D4	D3	D2	D1	D0		
표시 클리어	0	0	0	0	0	0	0	1	1.64ms	Clear Display 후, 커서 홈 위치
커서 홈	0	0	0	0	0	0	1	*	1.64ms	커서를 홈 위치로
엔트리 모드	0	0	0	0	0	1	I/D	S	40us	데이터 읽기/쓰기 시 커서 진행방향 I/D(1:증가, 0:감소) S(1:표시 시프트)
표시 On/Off	0	0	0	0	1	D	C	B	40us	D (1:화면 표시) C(1:커서 표시) B(1:커서 깜빡임)
커서/표시 시프트	0	0	0	1	S/C	R/L	*	*	40us	S/C (1:화면, 0:커서) R/L(1:오른쪽, 0:왼쪽)
기능 설정	0	0	1	DL	N	F	*	*	40us	D/L (1:8비트 0:4비트 인터페이스) N(1:2라인, 0:1라인) F(1:5X10dots,0:5X7dots)
CGRAM Address 설정	0	1	CGRAM Address						40us	CGRAM Address Set, 이후 전송. 데이터는 CGRAM 데이터로 인식
DDRAM Address설정	1	L	DDRAM Address						40us	DDRAM Address Set, 이후 전송 데이터는 DDRAM 데이터로 인식 L(0:1라인, 1:2라인)
비지플래그/ 어드레스읽기	BF	Address							0us	Busy Flag 체크 및 어드레스 읽기 CG/DDRAM 모두 사용 가능 BF(0:처리 끝남, 1:처리 중)
CG/DD RAM 데이터 Write	Data								40us	CG/DD RAM 에 데이터를 써 넣는다
CG/DD RAM 데이터 Read	Data								40us	CG/DD RAM 의 데이터를 읽다.

## [2] 제어 명령

LCD 모듈을 사용하기 위해서는 제어 명령(Instruction)을 알아야 한다. 이들 명령은 MCU가 LCD 모듈을 초기화하거나 제어하는 프로그램에서 사용되며 데이터 버스 D0 ~ D7을 통하여 전송된다.

### ☐ 표시 클리어

Instruction Register에 [00000001]을 입력하여야 한다. LCD의 화면을 지우고 주소 카운터를 DD RAM 어드레스 0번지로 set한 후, 커서를 home위치(0 번지)로 옮긴다.

### ☐ 커서 home

InstructionRegister에 [00000010]을 입력하여야 한다. DD RAM의 내용은 변경하지 않고 커서만 home 위치로 옮긴다.

### ☐ 엔트리 모드

Instruction Register에 [000001(I/D)S]로 입력하여야 한다. 데이터를 리드 / 라이트 할 때 커서의 위치를 오른쪽으로 이동시킬 것인가, 왼쪽으로 이동시킬 것인가를 결정하며, 또 이 때 화면을 시프트 동작의 유, 무를 결정한다.

- I/D : DD RAM, CG RAM에 문자코드를 리드, 라이트 할 때 DD RAM, CG RAM의 어드레스를 +1, -1한다.

I/D=1 : 어드레스 +1, 커서의 위치를 오른쪽 이동

I/D=0 : 어드레스 -1, 커서의 위치를 왼쪽 이동

- S : S=1일 때, DD RAM의 전체 데이터를 좌, 우 방향으로 이동시킨다.

S=1 : 화면을 시프트 시킨다.

S=0 : 화면을 시프트 시키지 않는다.

위에서 설명한 I/D, S를 조합하여 LCD에 Display 하는 데이터를 이동시킬 수 있다.

S=1,I/D=1 : LCD의 Display Data 전체가 좌로 이동한다.

S=1,I/D=0 : LCD의 Display Data 전체가 우로 이동한다.

## □ 표시 ON/OFF

Instruction Register에 [00001DCB]을 입력하여야 한다.

화면상의 DISPLAY를 ON/OFF 한다.

- D = 1 : 화면 표시 ON.
- D = 0 : 화면 표시 OFF.
- C = 1 : 커서 표시 ON.
- C = 0 : 커서 표시 OFF.
- B = 1 : 커서를 깜박이게 한다.
- B = 0 : 커서를 깜박이지 않는다.

## □ 커서/표시 시프트

Instruction Register에 [0001(S/C)(R/L)\*\*]을 입력하여야 한다.

Display data, 커서를 오른쪽, 왼쪽으로 시프트 하도록 한다.

S/C	R/L	동 작
0	0	커서 위치를 좌로 이동한다. (어드레스 카운터 -1)
0	1	커서 위치를 우로 이동한다. (어드레스 카운터 +1)
1	0	표시 전체를 좌로 이동, 표시는 커서에 따라 이동한다.
1	1	표시 전체를 우로 이동, 커서는 표시에 따라 이동하지 않음

## □ 평선 셋

Instruction Register에 [001(DL)NF\*\*]을 입력하여야 한다. MCU와의 인터페이스에서 데이터의 길이를 8비트, 4비트로 설정하고, 화면 표시 행수를 2행, 1행으로 지정하며, 문자의 폰트를 5×10 DOT, 5×7 DOT로 설정한다.

- DL = 1 : 데이터를 8BIT 사용
- DL = 0 : 데이터를 4BIT 사용.

정확한 데이터를 보내기 위해서는 8BIT를 전송해야 한다. 그래서 상위 4BIT를 먼저 전송하고, 하위 4BIT를 나중에 전송한다.

- N : 표시 행수를 설정한다.
- F : 문자 폰트를 설정한다.

N	F	표시행 수	문자 폰트	듀티비
0	0	1	5 x 7	1/8
0	1	1	5 x 10	1/11
1	*	2	5 x 7	1/18

### □ CG RAM 어드레스 셋

Instruction Register에 [01000000]을 입력하여야 한다. CG RAM의 어드레스를 지정한다. 이후에 송수신하는 데이터는 CG RAM의 데이터이다.

### □ DD RAM 어드레스 셋

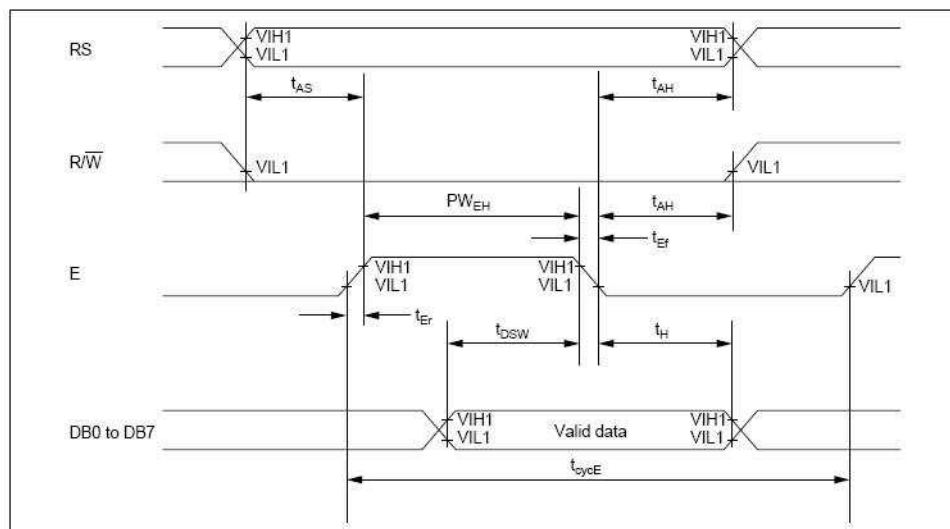
Instruction Register에 [10000000]을 입력하여야 한다. DD RAM의 어드레스를 지정한다. 이후에 송수신하는 데이터는 DD RAM의 데이터이다.

### □ 비지플래그/어드레스 리드

Instruction Register에 [(BF)0000000]을 입력하여야 한다. LCD 모듈이 현재 동작하고 있음을 나타내는 Busy Flag 및 어드레스 카운터의 내용을 리드한다.

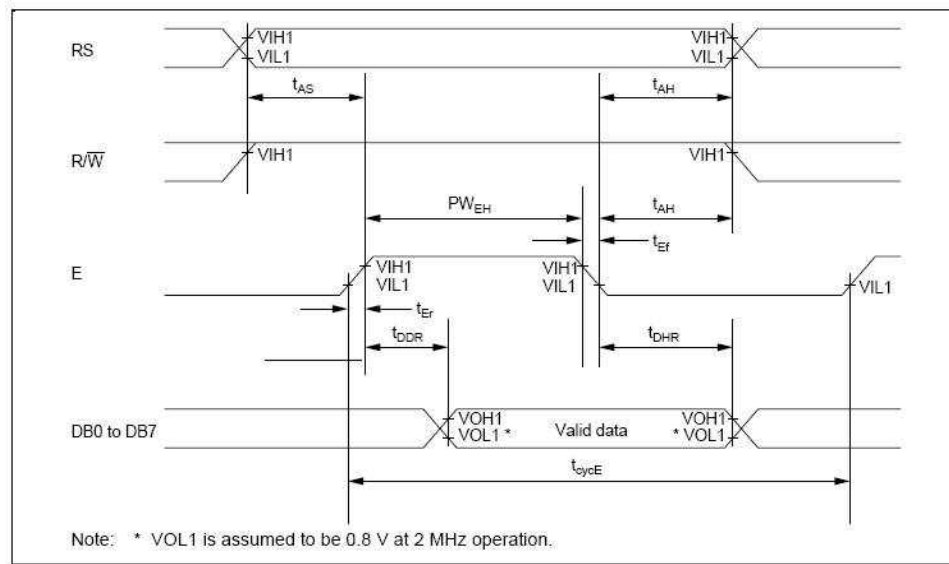
LCD모듈이 각 제어 코드를 실행하는 데는 지정된 시간이 필요하므로 CPU는 다음 제어 코드를 보내기 전에 충분히 대기하여야 한다.

하지만, Busy Flag를 읽어 1이면 LCD 모듈이 활동 중이므로 기다리다가, 0이 되면 활동을 멈춘 상태여서 다음 명령을 보내면 빠르게 DISPLAY할 수 있다.



Write 오퍼레이션 타이밍도

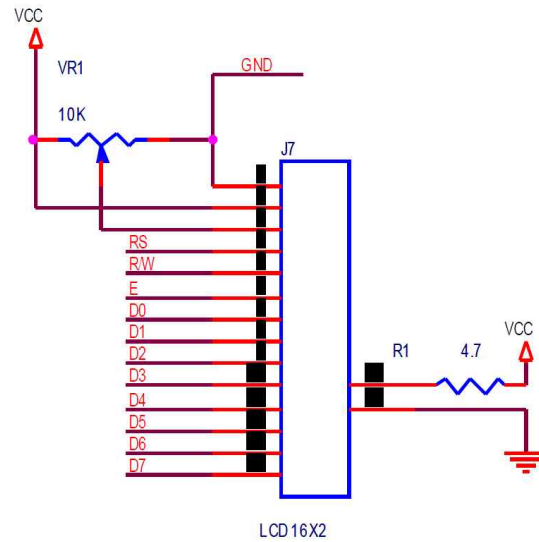




Read 오퍼레이션 타이밍도

## 5-3. Character LCD 구성

### [1] Character LCD 회로도



### [2] 케이블 결선도

장비와 같이 제공되는 USB ISP A/B 케이블을 이용하여 ATmega128 보드와 아래와 같이 연결한다.



### [3] 장비 결선도

아래와 같이 ATmega128 IO 포트와 LCD를 연결 하도록 한다.

ATmega128 PORTC	LCD
0	RS
1	RW
2	E
3	공란

ATmega128 PORTC	LCD
4	D4
5	D5
6	D6
7	D7

## 5-4. Character LCD 제어

### (1) 프로그램 순서

1-1. Character LCD를 사용하기 위한 프로그램 작성

#### ■ Program Source

```
#include <mega128.h> // Atmega 128 header file
#include <delay.h>
// Port C 0,1,2,3,4,5,6,7 -> RS,RD,EN,free,D4,D5,D6,D7
// Alphanumeric LCD Module functions
#asm
.equ _lcd_port=0x15
#endasm
#include <lcd.h>

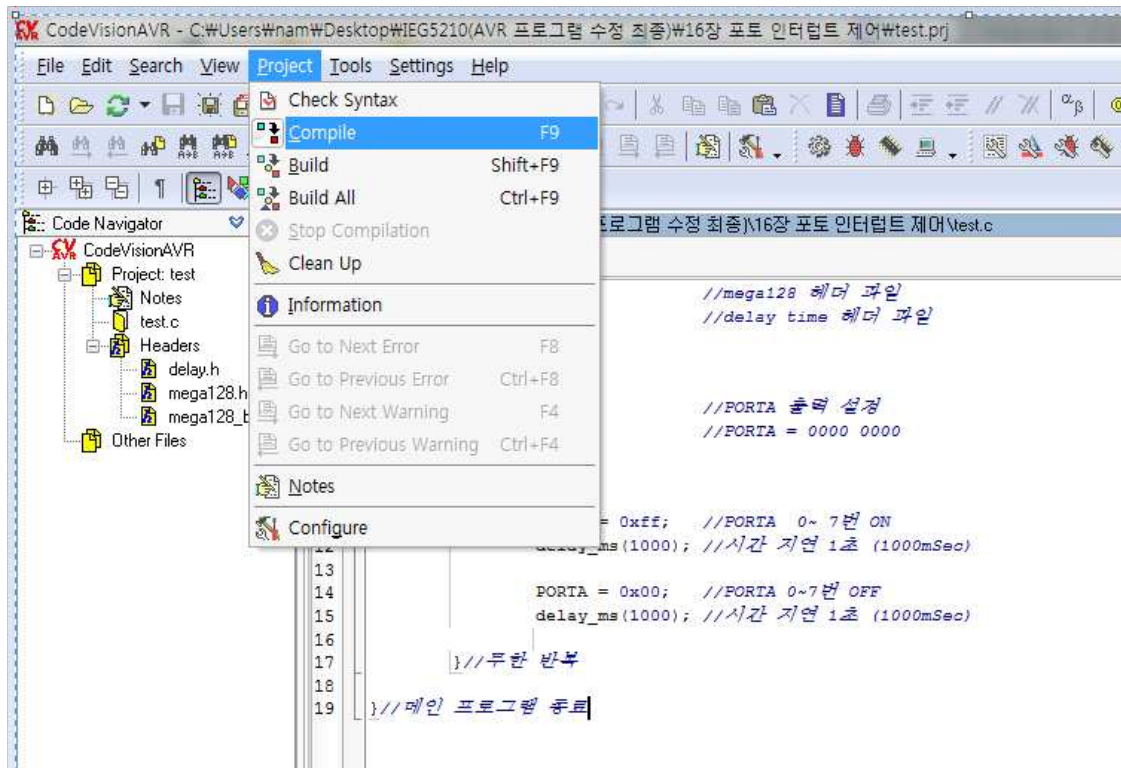
void main(void)
{
    char buff[17]="SPG Korea";    // LCD 출력 Data 저장 변수 설정

    PORTC=0x00; // output
    DDRC=0xFF;

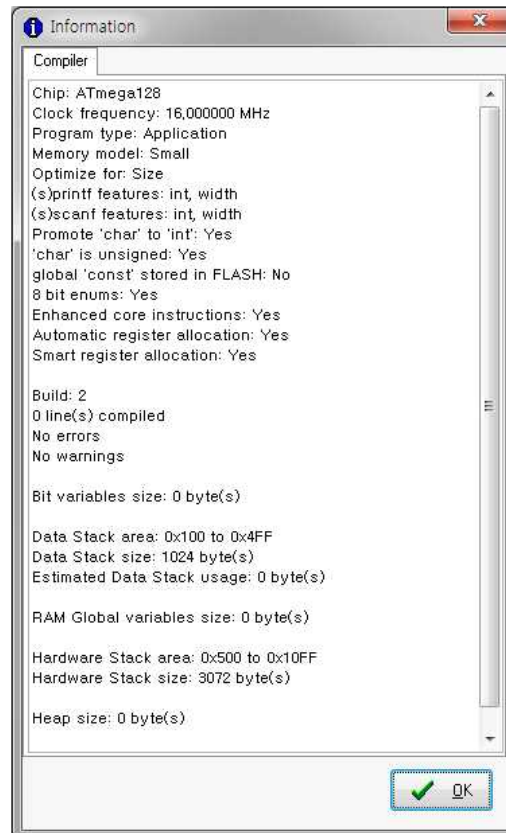
    lcd_init(16);           // Character LCD 16 * 2 사용 선언
    lcd_gotoxy(0,0); // LCD 1 line 선택 함수, x=0, y=0
    delay_ms(100);         // 시간 지연 함수
    lcd_puts(buff);         // buff변수 Data를 LCD로 출력
    lcd_gotoxy(0,1); // LCD 2 line 선택 함수, x=0, y=1
    lcd_putsf("SPG Co,Ltd"); // Data를 LCD로 출력
}
```

## [2] CodeVision 프로그램 컴파일

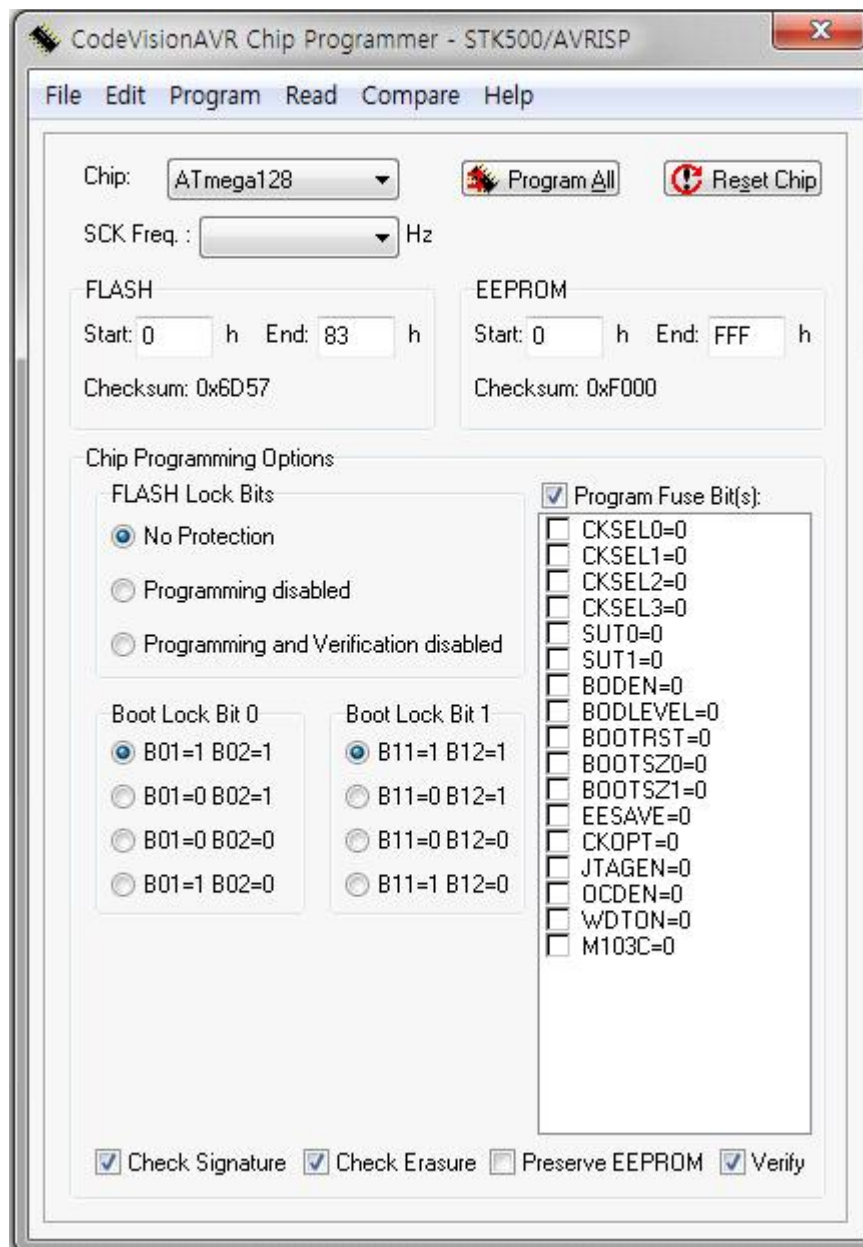
- CodeVision 프로그램에서 Project 메뉴를 선택하여 Compile을 실행 하도록 한다.



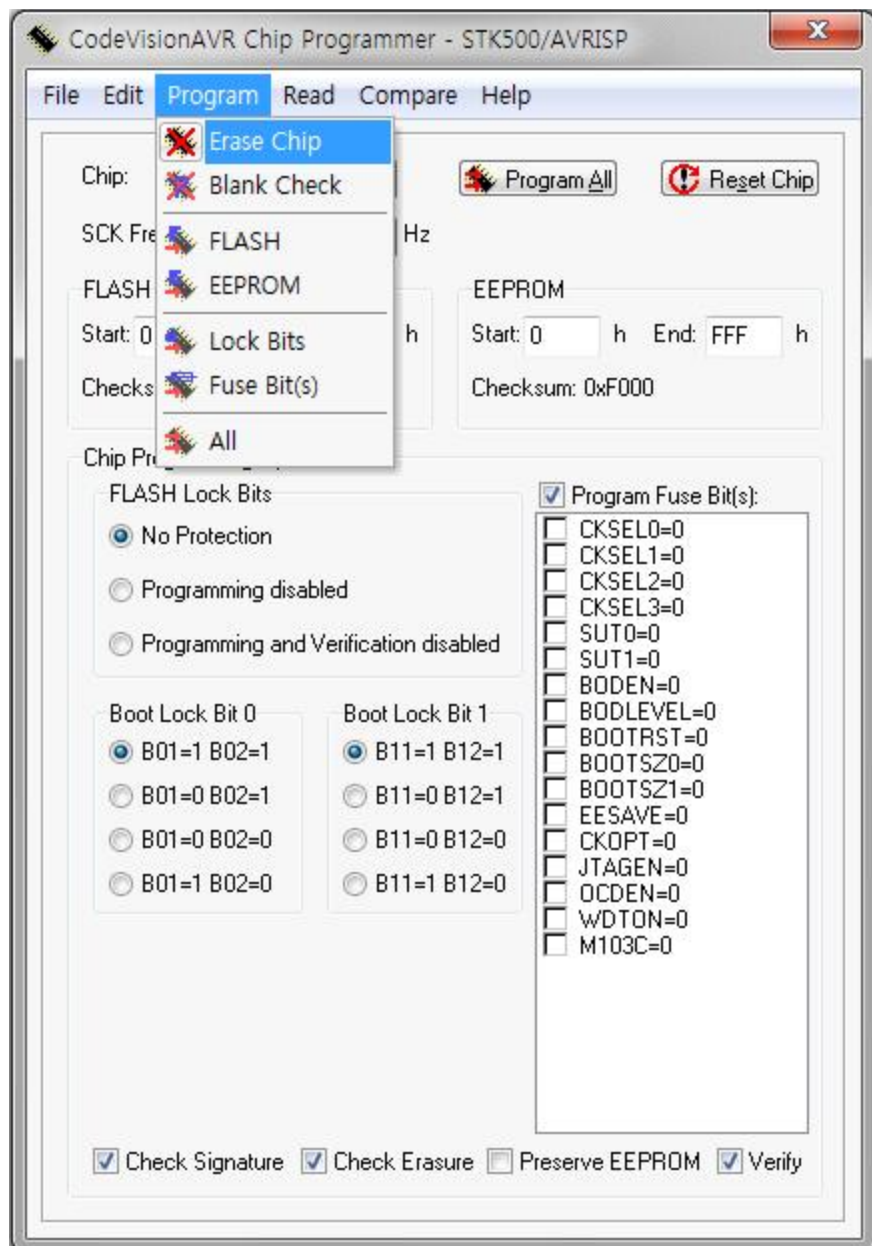
- 프로그램에서 에러가 없으면 아래와 같이 컴파일 정보가 출력 된다.



- Tools → Chip Programmer 클릭(단축키 : Shift+F4)

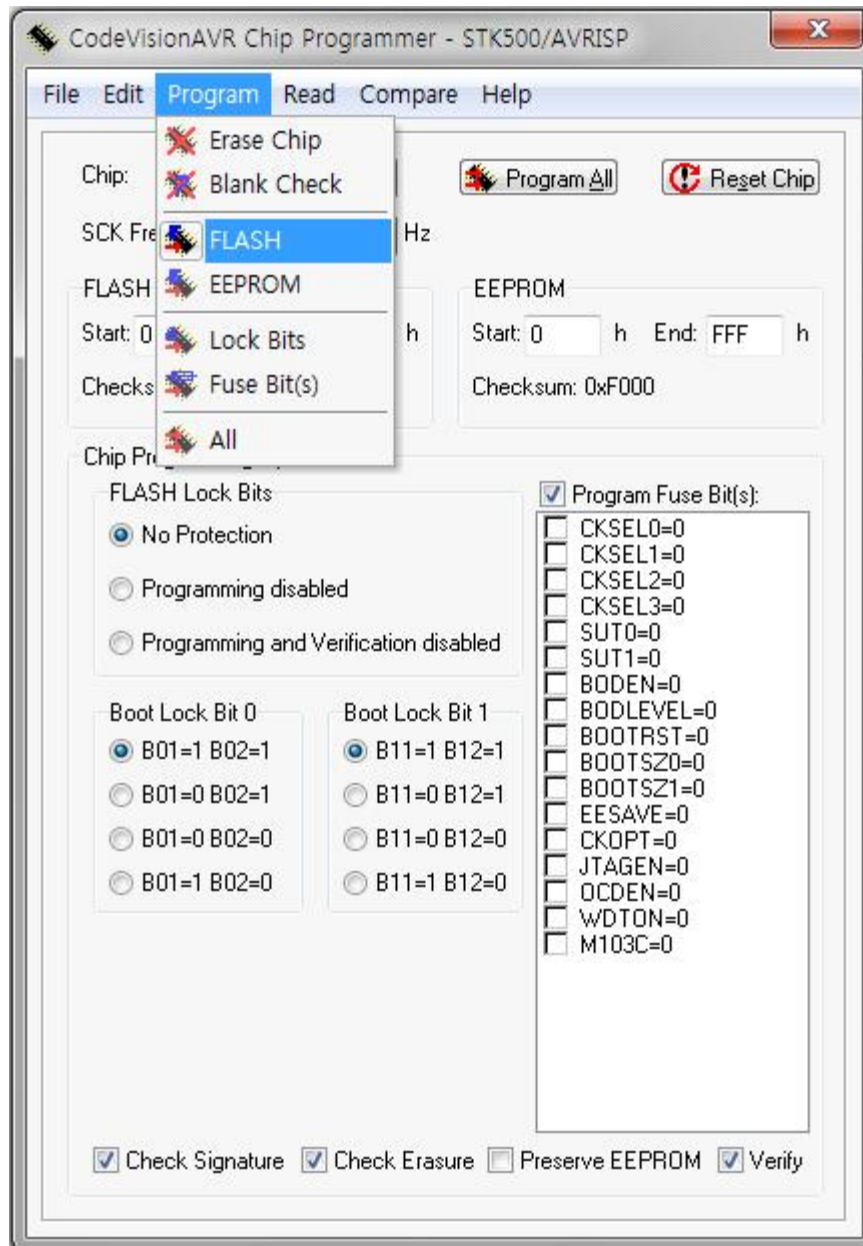


- Erase chip 클릭(chip 내에 프로그램이 남아 있을수 있으므로 프로그램 다운로드 전에 반드시 Erase Chip을 권장하도록 한다. )



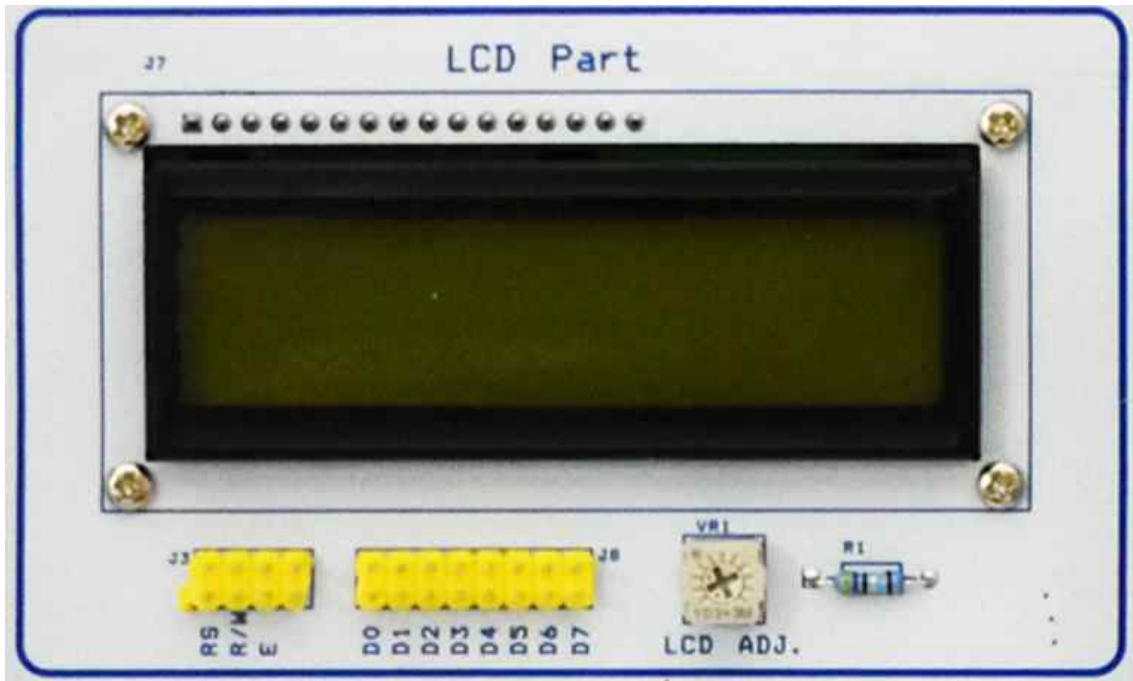


- Chip Flash 클릭하여 프로그램을 다운로드 하도록 한다.



### [3] 동작 확인

- 작성한 프로그램이 LCD 첫 번째 줄 1번째 칸에 커서가 이동하여 'SPG Korea'를 출력하는 것을 확인 할 수 있다.



### [4] 실습 과제

4-1. 과제명 : 첫 번째 줄 1번째 칸에는 'HELLO'를 출력하고 두 번째 줄 1번째 칸에 'ATmega128'를 출력한다.

4-2. 과제명 : 첫 번째 줄 1번째 칸에 'Korea'를 출력하고 문자가 오른쪽으로 이동 출력을 하도록 한다.

## SECTION

## 06

## KEY-Matrix 제어 실습

---

## 6-1. 학습 목표

- KEY-Matrix 구조와 동작 방법
- KEY-Matrix 제어
- KEY-Matrix 값을 Segment 출력

## 6-2. 기초 이론

### [1] KEY-Matrix 란?

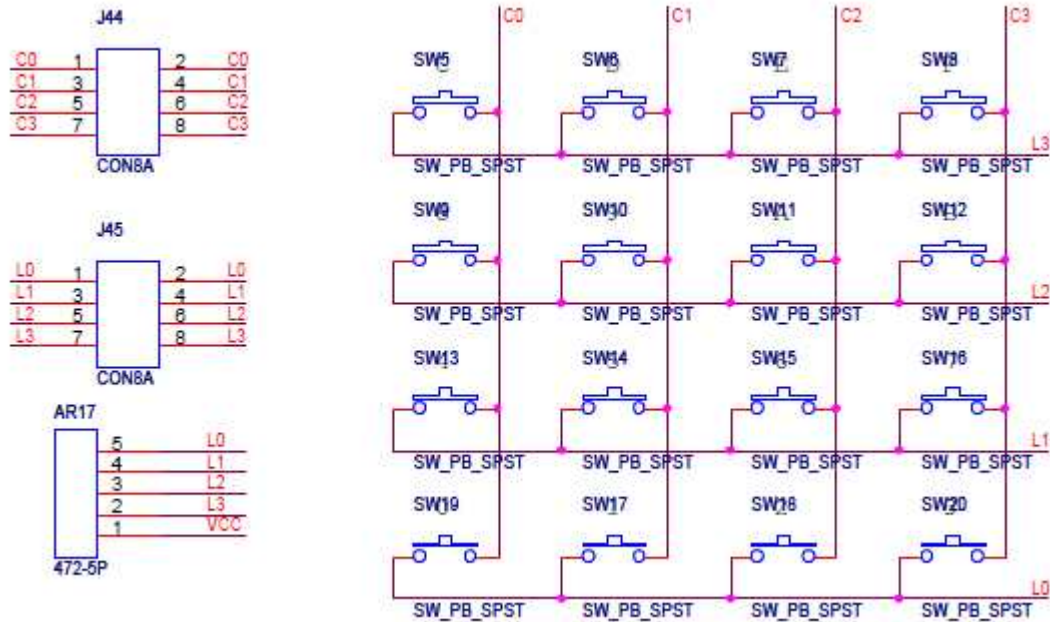
Digital 회로에서 어떠한 정보를 입력하는 방법에는 여러 가지가 있다. 그중 적은 Port로 많은 입력을 받을 수 있는 방법으로는 Key Matrix를 이용하는 것이다.

다시 말해, Matrix 구조의 Key 조합으로 4개의 out Port 와 4개의 in Port로 16개의 입력을 받을 수 있다. 입력을 0x0e로 하면 Key 값은 C, 8, 4, 0의 값 중 하나를 ('0'값을 가지고 있는 Line의 값을 인식) 택하려는 것이다. 그리고 다시 0x0d를 출력하면 D, 9, 5, 1의 값 중 하나를 택할 것이다.

이렇게 0x0b, 0x07을 순차적으로 입력하고 Key Data Output의 값을 읽으면 0에서 f까지의 Key 값을 읽을 수 있다.

## 6-3. Key-Matrix 구성

### [1] Key-Matrix 회로도



### [2] 케이블 결선도

장비와 같이 제공되는 USB A/B 케이블을 이용하여 ATmega128 보드와 아래와 같이 연결한다.



### [3] 장비 결선도

아래와 같이 ATmega128 IO 포트와 Key-Matrix를 연결 하도록 한다.

ATmega128 PORTC	Key-Matrix 핀
7	L3
6	L2
5	L1
4	L0

ATmega128 PORTC	Key-Matrix 핀
3	C0
2	C1
1	C2
0	C3

아래와 같이 ATmega128 IO 포트와 Segment를 연결 하도록 한다.

ATmega128 PORTB	Segment Enable핀
0	Q0
1	Q1
2	Q2
3	Q3

ATmega128 PORTA	Segment 핀
0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	DP

## 6-4. Key-Matrix 제어

### (1) 프로그램 순서

1-1. key-Matrix 제어를 위하여 프로그램 작성

#### ■ Program Source

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

#define Q0          PORTB.0          //PortB.0 bit FND Q0
#define Q1          PORTB.1          //PortB.1 bit FND Q1
#define Q2          PORTB.2          //PortB.2 bit FND Q2
#define Q3          PORTB.3          //PortB.3 bit FND Q3

#define FndDisplay(cmd) PORTA = fnd[cmd]; //Port A를 FND 설정

#define Key_data_input PINC
#define Key_data_output PORTC

// '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F', 'dot'
unsigned char
fnd[17]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xd8,0x80,0x98,0x88,0x83,0xc6,0xa1,0x86,
0x8e,0x7f};

int KeyMatrix();
//Key Matrix 함수 선언

void main(void)
{
    // Port 초기화 함수

    DDRA = 0xFF; // Port A 설정, 출력으로 사용
    DDRB = 0xFF; // Port B 설정, 출력으로 사용
    DDRC = 0x0F; // Port C 설정, PORTC0,1,2,3출력, 4,5,6,7 입력으로 사용
    DDRD = 0xFF;

    PORTA = 0xFF; // Port A 초기값
    PORTB = 0xFF; // Port B 초기값
    PORTC = 0x00; // Port C 초기값
    PORTD = 0xFF; // Port D 초기값
```

## ■ Program Source

```

Q3 = 0;          // PortB.3 0 출력, Q3에 0 출력
Q2 = 0;          // PortB.2 0 출력, Q2에 0 출력
Q1 = 0;          // PortB.1 0 출력, Q1에 0 출력
Q0 = 0;          // PortB.0 0 출력, Q0에 0 출력

FndDisplay(0);           //초기값으로 FND에 0출력

while(1)
{
    FndDisplay( KeyMatrix());           // 키 매트릭스 함수 호출

} //while program end

} //main program end

// FND 출력 코드 함수

// 키 매트릭스 함수 정수형으로 현재 입력된 값을 반환 받을 수 있다.
int KeyMatrix()
{
    int keyout = 0xfe;           // 키 매트릭스 신호
    int i;
    for( i = 0; i <= 3; i++ )
    {
        Key_data_output = keyout; // Port C에 1111 1110 출력,
0,4,8,C line Enable
        delay_ms(1);    // 출력 딜레이

        switch(Key_data_input & 0xf0)
        {
            case 0xe0: return 0+i;
            case 0xd0: return 4+i ;
            case 0xb0: return 8+i;
            case 0x70: return 12+i;
        }
        keyout = (keyout<<1) + 0x01;    //키보드 신호 출력 카운터
    }
}

```

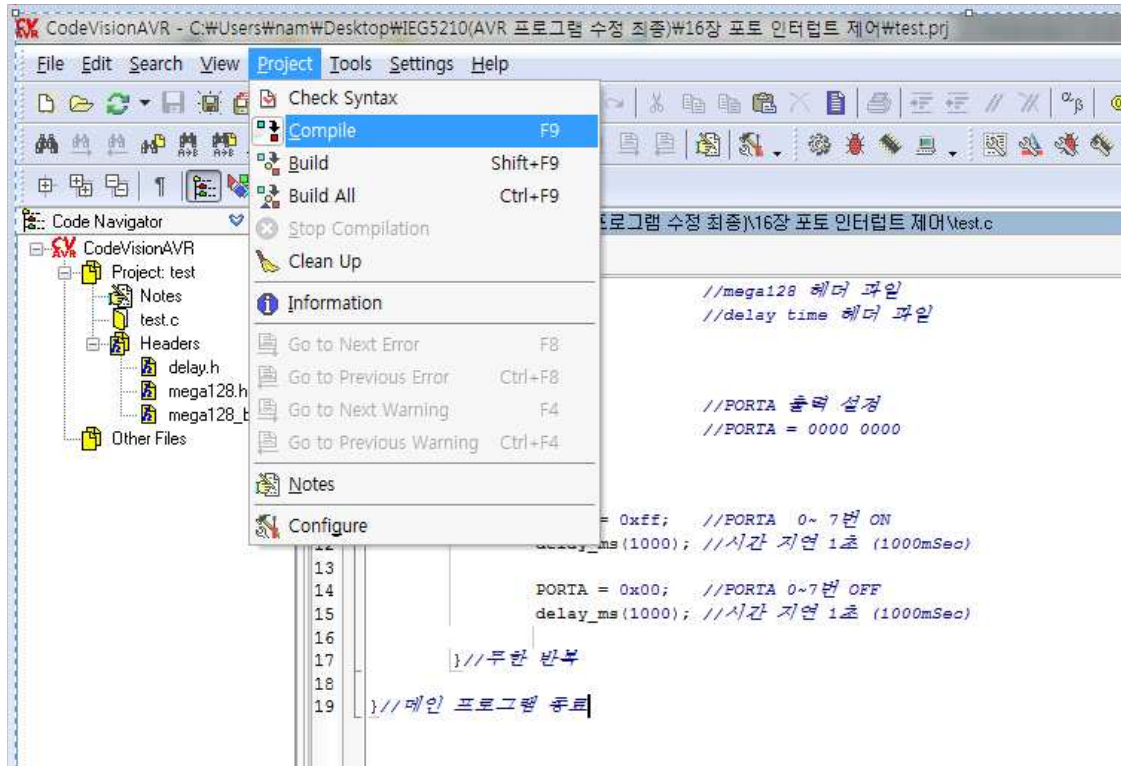


■ Program Source

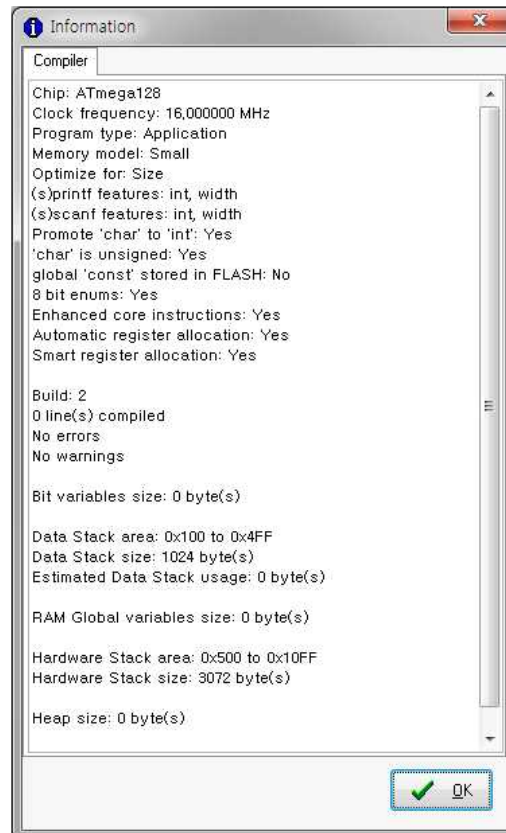
```
    }  
    return 0;  
  
} //key matrix program end
```

## [2] CodeVision 프로그램 컴파일

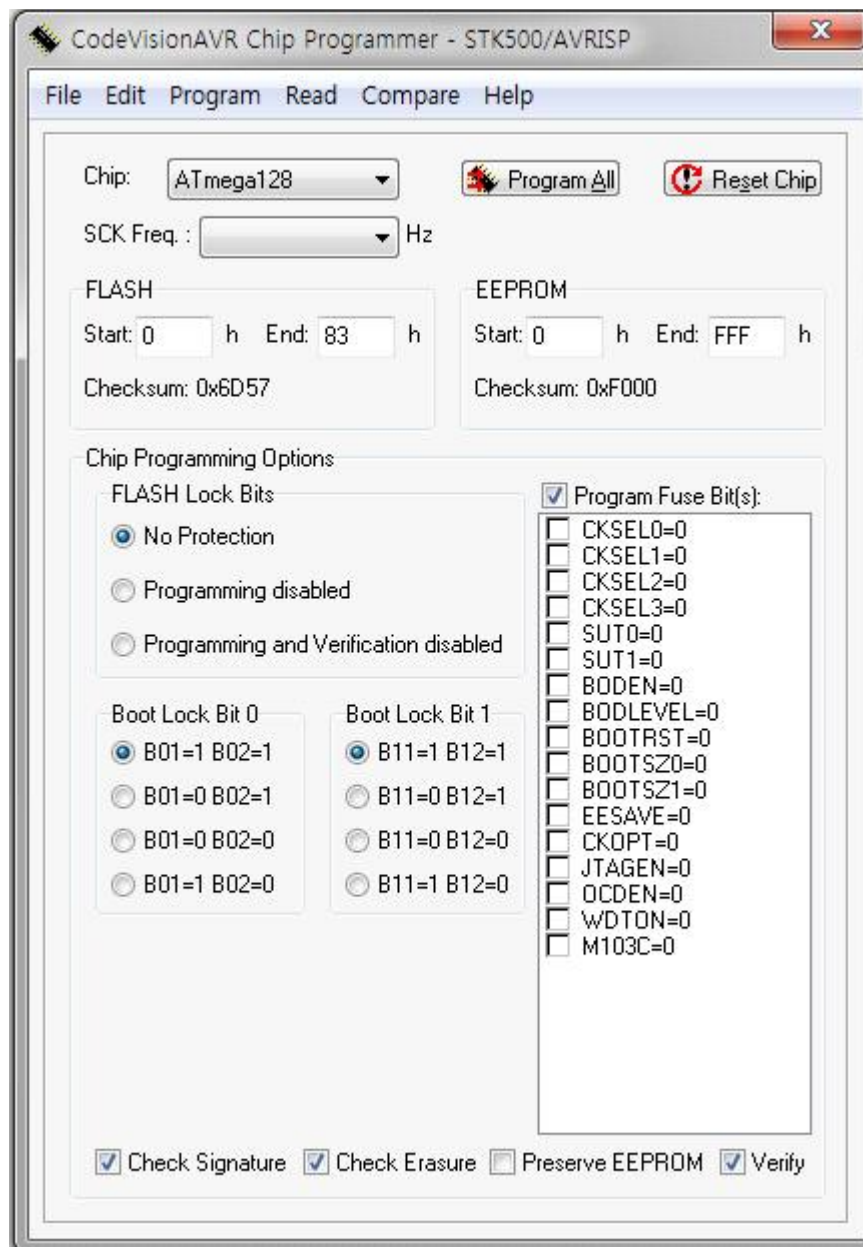
- CodeVision 프로그램에서 Project 메뉴를 선택하여 Compile을 실행 하도록 한다.



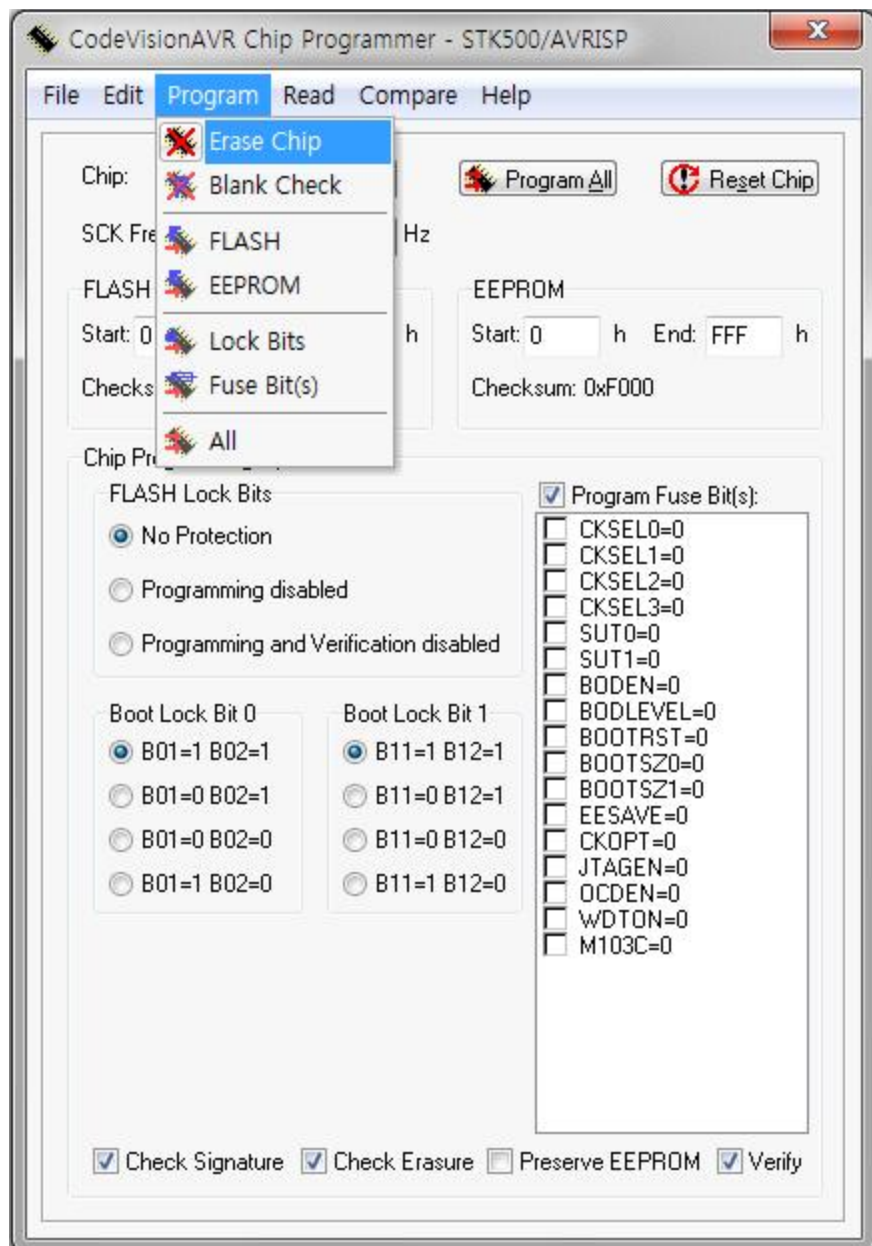
- 프로그램에서 에러가 없으면 아래와 같이 컴파일 정보가 출력 된다.



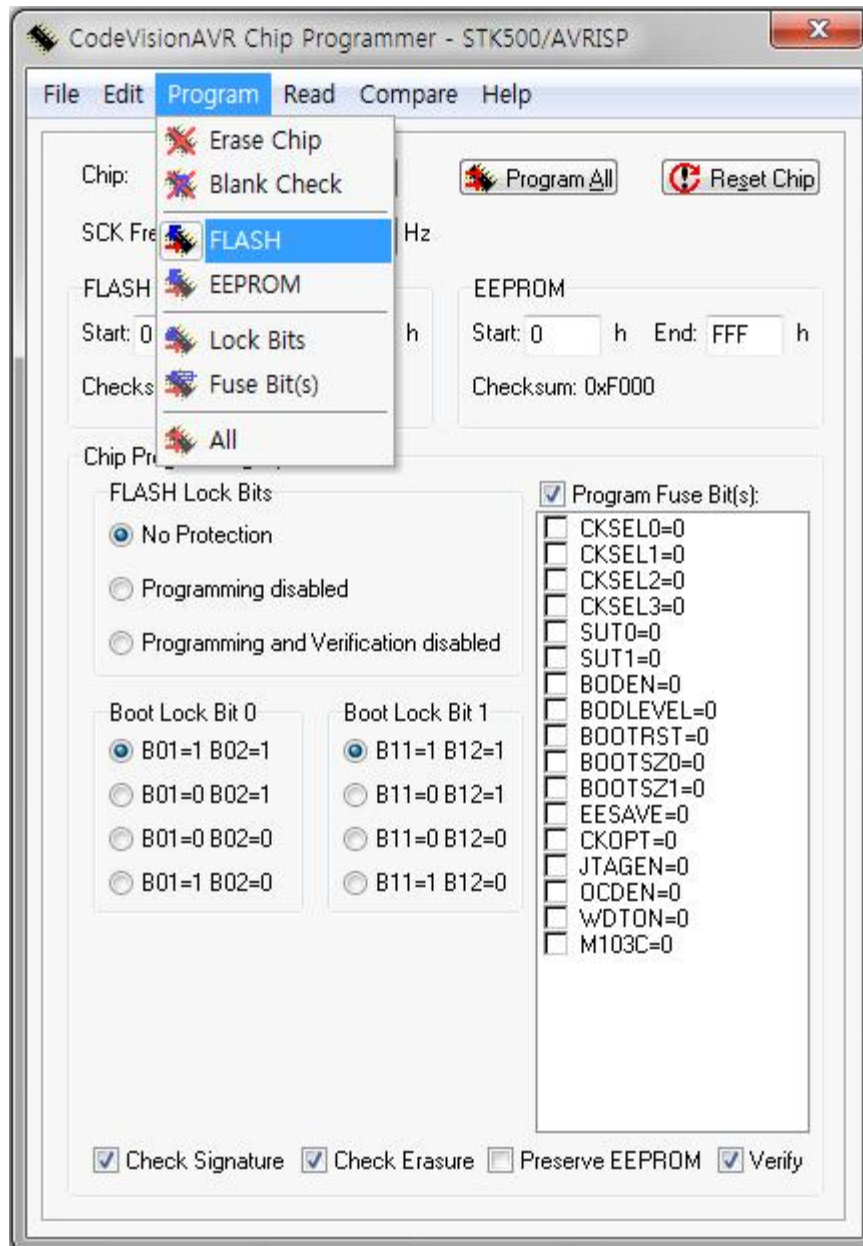
- Tools → Chip Programmer 클릭(단축키 : Shift+F4)



- Erase chip 클릭(chip 내에 프로그램이 남아 있을수 있으므로 프로그램 다운로드 전에 반드시 Erase Chip을 권장하도록 한다. )

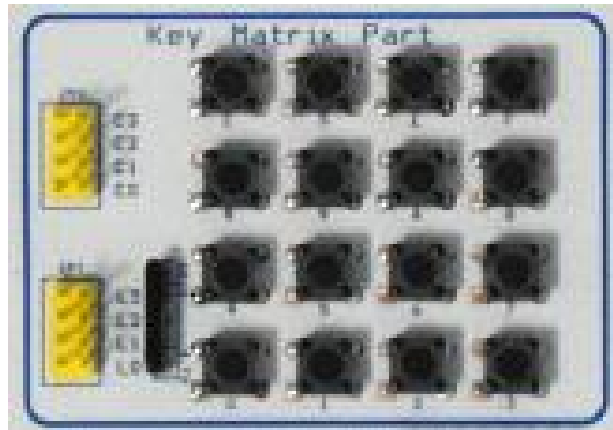


- Chip Flash 클릭하여 프로그램을 다운로드 하도록 한다.



### [3] 동작 확인

- Key를 누를 때 마다 Segment에 값이 출력 되는 것을 확인 할 수 있다.



### [4] 실습 과제

4-1. 과제명 : Key를 누를 때 마다 LED에 출력값을 출력하도록 한다.

(단 LED는 8개의 포트 이므로 8421 코드를 이용하여 출력 하도록 한다. )

4-2. 과제명 : key를 누를 때 마다 LCD에 숫자를 출력하도록 한다.

(단 LCD는 1개를 이용하여 0 ~ 9까지 값만 출력 하도록 한다. )

## SECTION

# 07 Servo Motor 제어 실습

## 7-1. 학습 목표

- Servo Motor 구조와 동작 방법
- Servo Motor 회전 방향 제어
- Servo Motor 회전 각도 제어

## 7-2. 기초 이론

### (1) Servo Motor 란?

Servo Motor는 주어진 신호에 의해서 일정한 각을 유지하거나 일정한 위치를 유지하는 모터를 말한다. Servo Motor는 일반적으로 무선조종 자동차, 모형비행기에 조향장치 등으로 많이 사용하는 모터이다. Servo Motor에 일정한 PWM신호를 넣어주면 일정 각도를 유지한다. 동작영역은 약 0~180°이다.



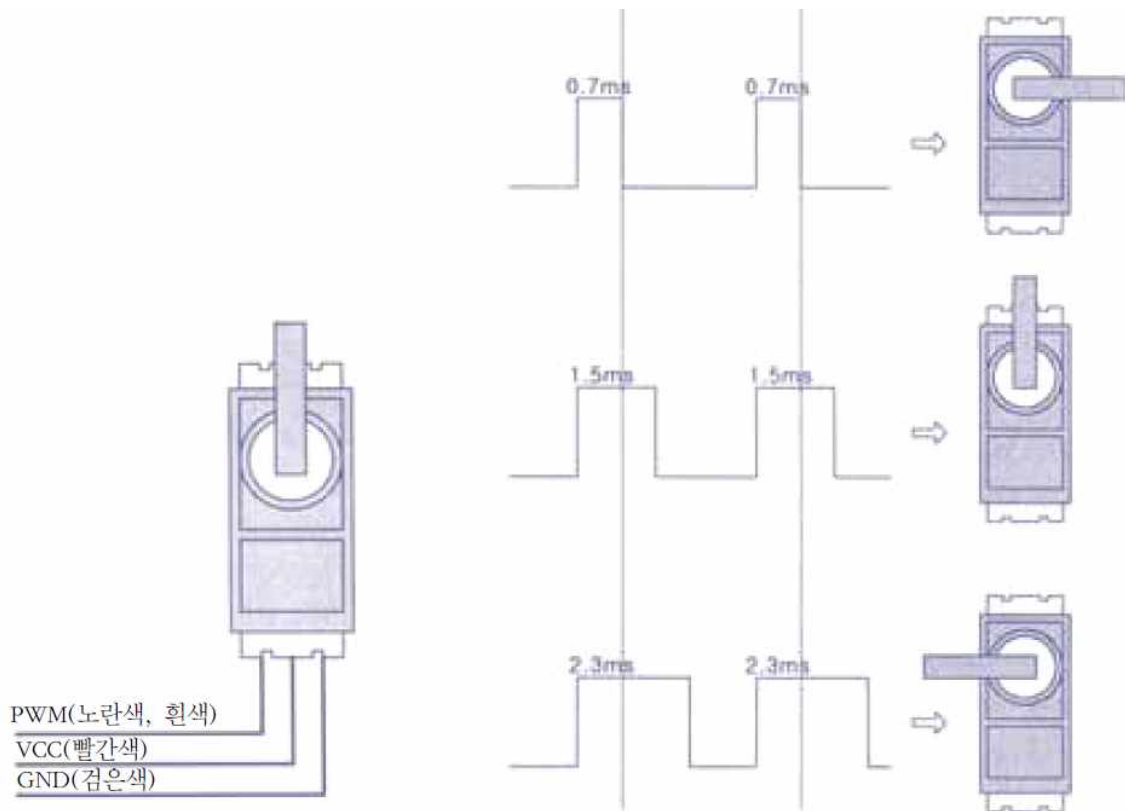
RC 서보 모터 외형



## □ Servo Motor 구동방법

Servo Motor 모터에는 3개의 선이 있다. 각각 GND(검은색), VCC(빨간색), PWM(노란색, 흰색)이다. 보통 전원은 4~6V에서 동작한다. 모터의 동작은 PWM 입력선으로 일정주기의 펄스신호를 주면 모터가  $-90^{\circ} \sim 90^{\circ}$ 의 사이에서 동작한다.

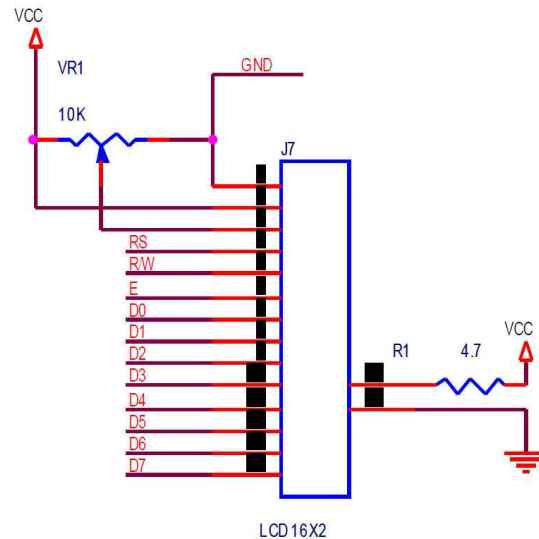
20ms의 주기로 1.5ms의 신호를 넣어주면  $0^{\circ}$ 의 각도로 동작을 하고, 0.7ms일 때는  $-90^{\circ}$ , 2.3ms의 신호를 넣어주면  $90^{\circ}$ 로 동작한다. 이밖에 다른 각도 제어한다면 0.7~2.3ms까지 다른 펄스의 신호를 넣어주면 된다. 펄스를 입력해주는 도중 신호가 중단된다면 모터는 각도 제어를 실행하지 않게 된다.



Servo Motor 제어펄스

## 7-3. Servo Motor 구성

### [1] Servo Motor 회로도 - Servo Motor 도면 없음



### [2] 케이블 결선도

장비와 같이 제공되는 USB A/B 케이블을 이용하여 ATmega128 보드와 아래와 같이 연결한다.



### [3] 장비 결선도

아래와 같이 ATmega128 IO 포트와 Servo Motor를 연결 하도록 한다.

ATmega128 PORTA	Servo Motor 핀
0	RC_IN

## 7-4. Servo Motor 제어

### (1) 프로그램 순서

1-1. Servo Motor 제어를 위한 프로그램 선언

#### ■ Program Source

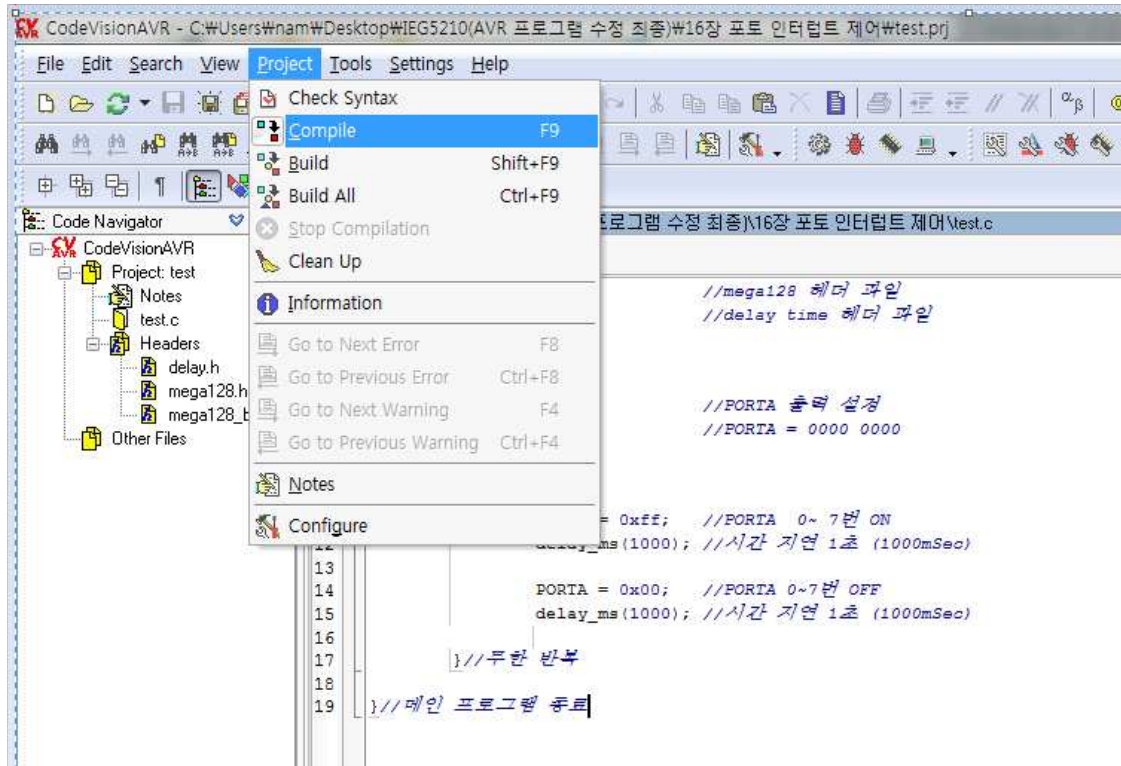
```
#include <mega128.h>
#include <delay.h>

void main(void)
{
    int i;
    DDRA=0xFF;
    while(1)
    {
        for(i=0;i<25;i++){ PORTA.0=1; delay_us(1500); PORTA.0=0; delay_ms(20); }
        // 0도
        for(i=0;i<25;i++){ PORTA.0=1; delay_us(2400); PORTA.0=0; delay_ms(20); }
        // 90도

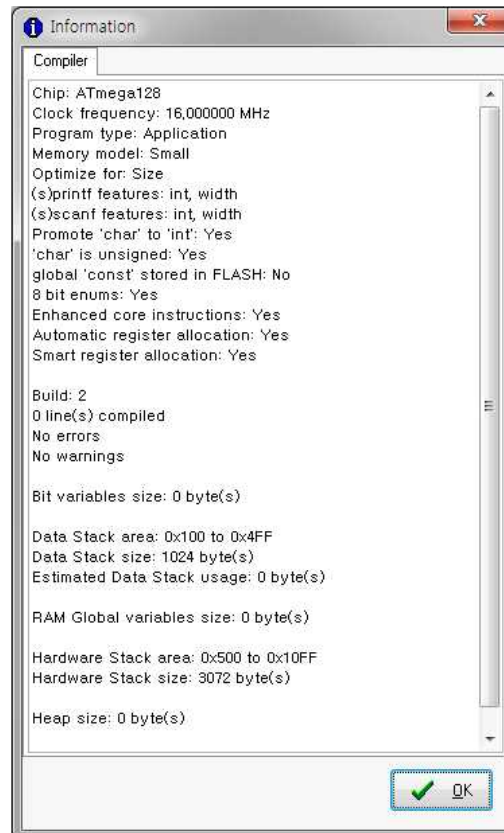
        for(i=0;i<25;i++){ PORTA.0=1; delay_us(1500); PORTA.0=0; delay_ms(20); }
        // 0도
        for(i=0;i<25;i++){ PORTA.0=1; delay_us( 600); PORTA.0=0; delay_ms(20); }
        // -90도
    }
}
```

## [2] CodeVision 프로그램 컴파일

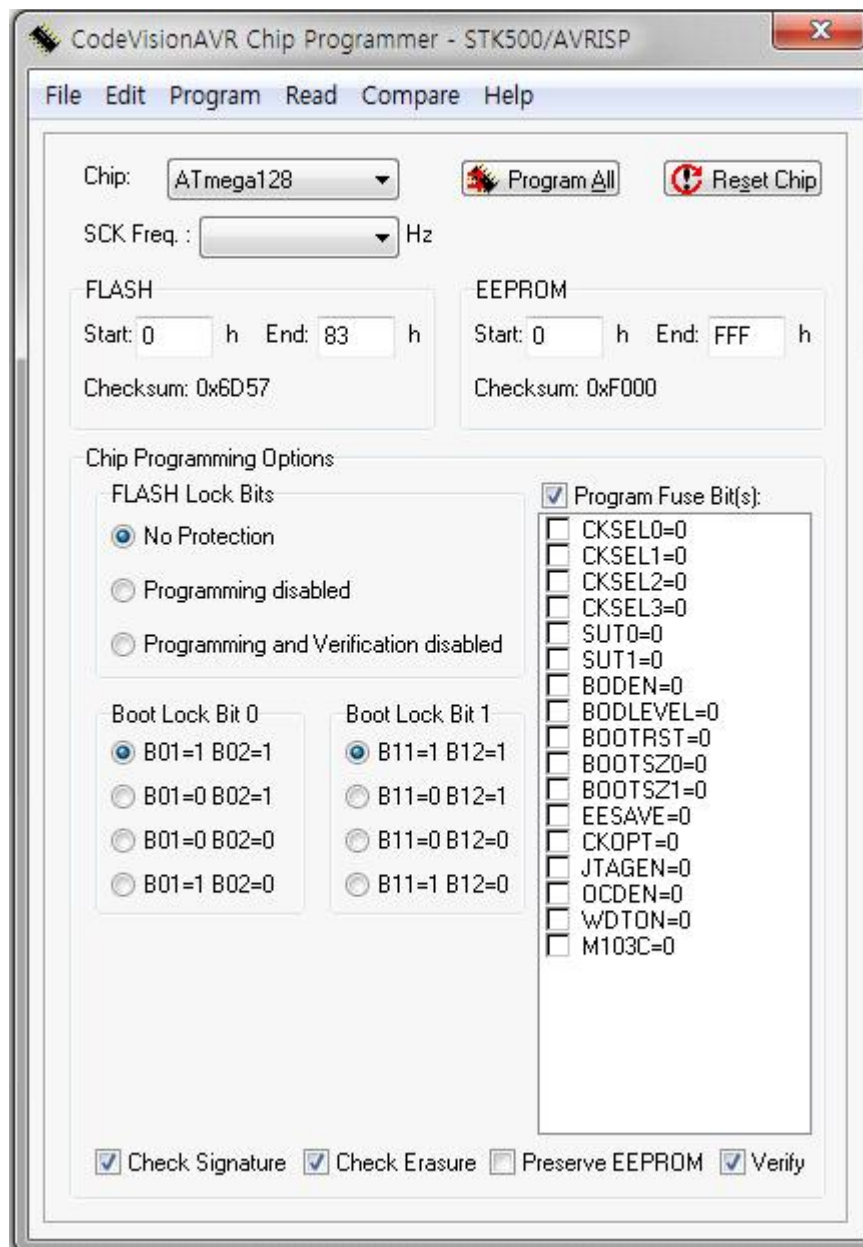
- CodeVision 프로그램에서 Project 메뉴를 선택하여 Compile을 실행 하도록 한다.



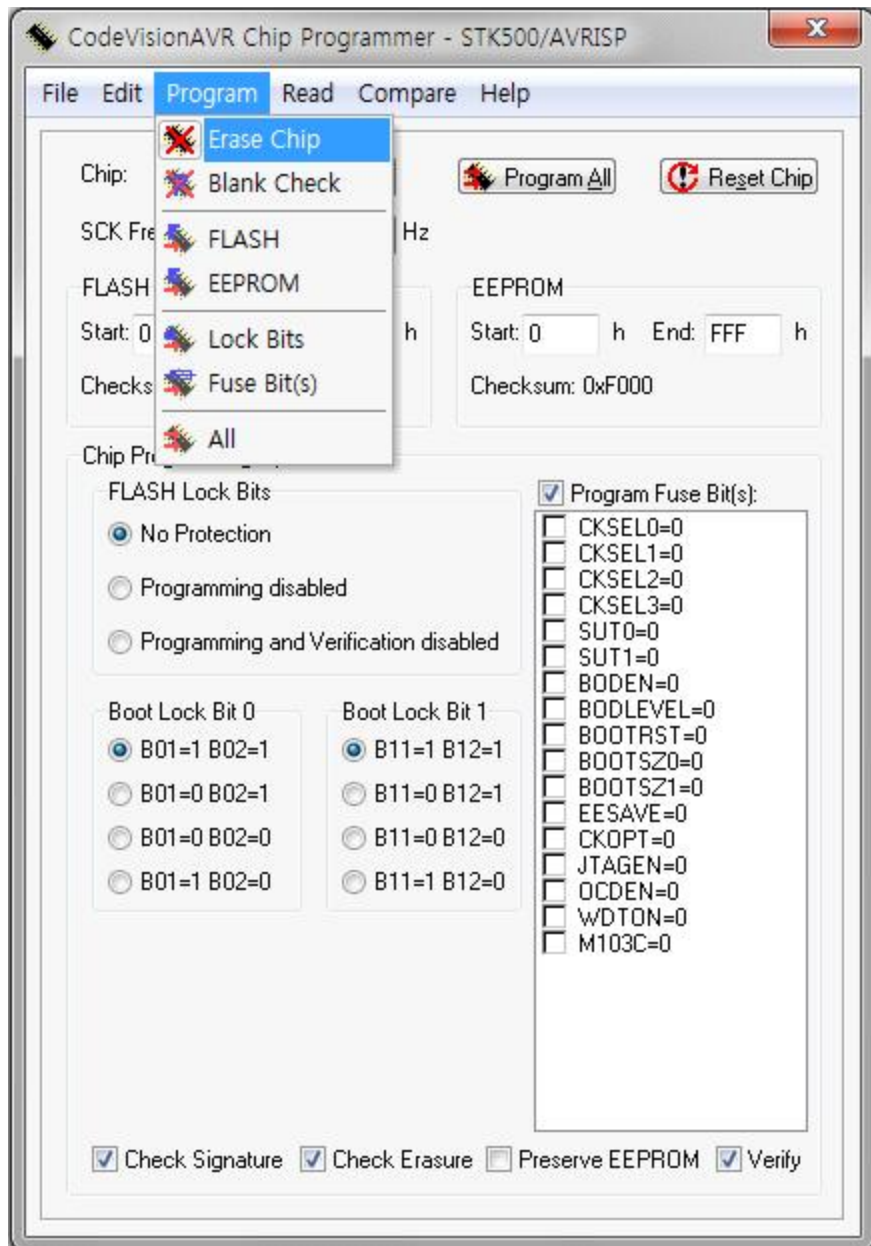
- 프로그램에서 에러가 없으면 아래와 같이 컴파일 정보가 출력 된다.



- Tools → Chip Programmer 클릭(단축키 : Shift+F4)

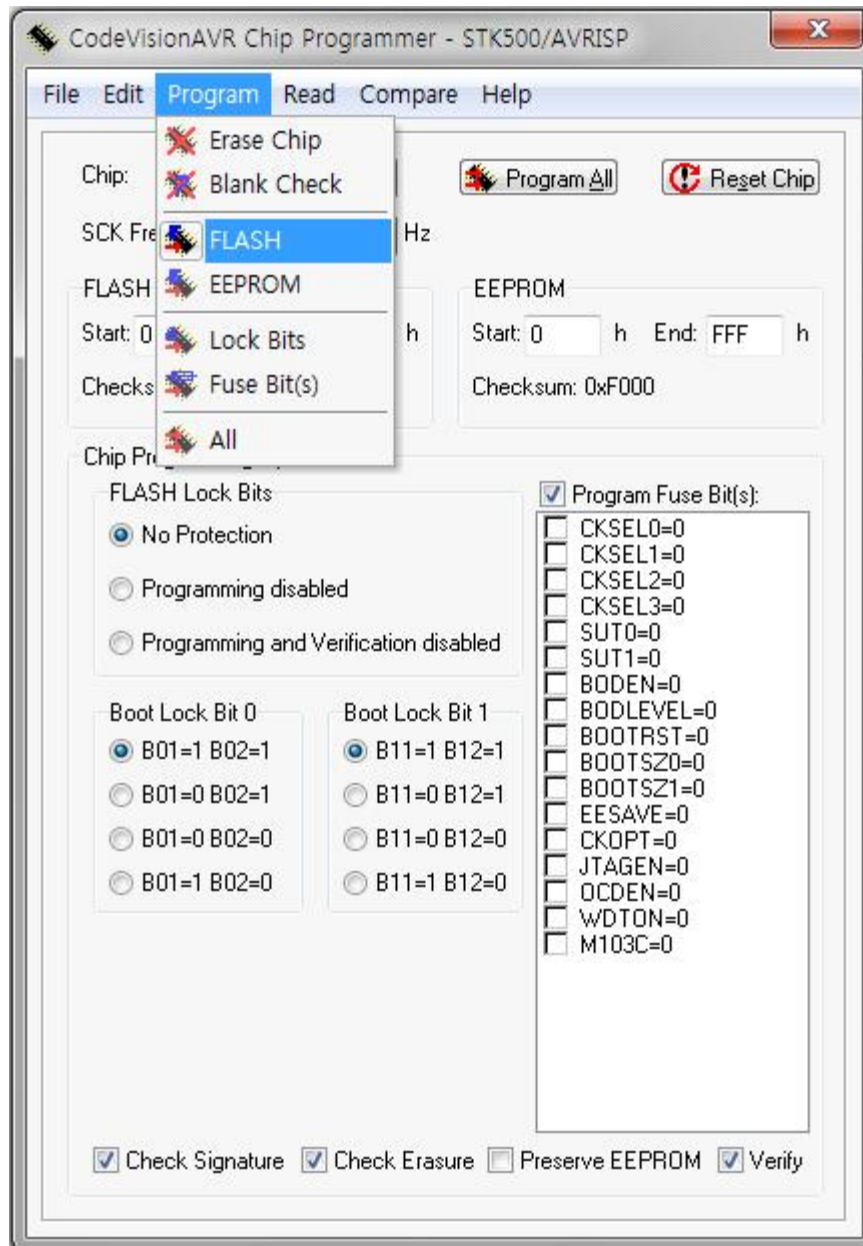


- Erase chip 클릭(chip 내에 프로그램이 남아 있을수 있으므로 프로그램 다운로드 전에 반드시 Erase Chip을 권장하도록 한다. )





- Chip Flash 클릭하여 프로그램을 다운로드 하도록 한다.



### [3] 동작 확인

- 0°, 90°, 0°, -90°를 계속적으로 Servo 모터각도가 동작하는 것을 확인 할 수 있다.



### [4] 실습 과제

4-1. 과제명 : 1번 스위치를 누르면 90°, 2번 스위치를 누르면 0°, 3번 스위치를 누르면 180° 동작을 하도록 한다.

4-2. 과제명 : 1번 스위치를 누르면 180° 방향으로 1°씩 증가를 하고 2번 스위치를 누르면 0° 방향으로 -1°씩 감소 하도록 한다.

단 0° 이하, 180°이상 동작은 증가/감소를 적용하지 않는다.

## SECTION

# 08 Step Motor 제어 실습

## 8-1. 학습 목표

- Step Motor 구조와 동작 방법
- Step Motor 회전 방향 제어
- Step Motor 회전 각도 제어

## 8-2. 기초 이론

### (1) Step Motor 란?

STEP Motor는 Digital 펄스를 입력받아 회전 운동을 하는 전동기이다. STEP Motor는 STEPPING Motor 또는 STEPPER 등으로도 불리며 직류 전압이나 전류 입력을 가하면 주어진 각도만큼 회전한다. STEP Motor를 적절히 제어하면 그 회전각은 항상 입력 펄스의 수에 비례한다.

각각의 펄스는 회전자를 1STEP만큼 씩 회전시키며 회전자는 자기적으로 정확한 위치에 정지한다.

## [2] Step Motor의 장·단점

### ■ 장점

- 피드백 없이 오픈 루프만으로 구동할 수 있다.
- 안정도에 문제가 없다.
- 디지털 입력 펄스에 의해 구동되므로 디지털 컴퓨터로 쉽게 제어된다.
- 기계적 구조가 간단하다. 따라서 유지 보수가 거의 필요 없다.
- 브러시가 없으므로 오염으로부터 안전하다.
- 필요시 발열을 쉽게 발산시킬 수 있다.
- 상대적으로 견고하고 튼튼하다.

### ■ 단점

- 고정된 스텝 각도만큼 이동하므로 분해능에 제약이 따른다.
- 보통의 드라이버로는 효율이 낮다.
- 스텝 응답에 대해 상대적으로 큰 오버슈트와 진동을 나타낸다.
- 관성이 큰 부하를 다루기 어렵다.
- 오픈 루프 제어 시 마찰이 증가할수록 위치 오차가 커진다.  
단 오차가 누적되지는 않는다.
- 출력과 크기에 제약을 받는다.

이상이 스텝모터의 장단점들이다.

## [3] Step 모터의 분류

다음은 스텝모터의 분류에 대해서 알아보자. 스텝모터는 권선의 상의 개수에 의해 분류한다. 바이폴라(양극성)와 유니폴라(단극성)방식으로 나뉘는데, 바이폴라 방식은 다음과 같다.

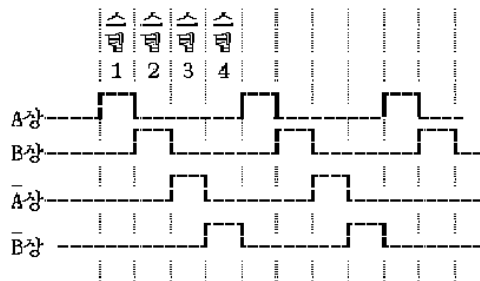
양극성 스텝 모터는 2상 브러시 없는 모터로 4개의 단자선이 나와 있다. 양극성 스텝 모터는 2상, 즉 2개의 권선을 가지므로 회전하기 위해서는 하나의 권선에 서로 다른 방향으로 전류가 흐를 수 있어야 한다. 즉 양극성 권선이 2개 필요하다.

유니폴라 방식은 4상 브러시 없는 모터로 5~6개의 단자선이 나와 있다. 단극성 스텝 모터는 4상, 즉 4개의 권선이 있으므로 한 번에 한 권선씩 차례로 전류를 흘려주면 영구자석 회전자가 회전하게 된다.

### ■ 스텝 모터의 여자 방식

- 1상 여자 : 한번에 1개의 상을 여자 하는 방식

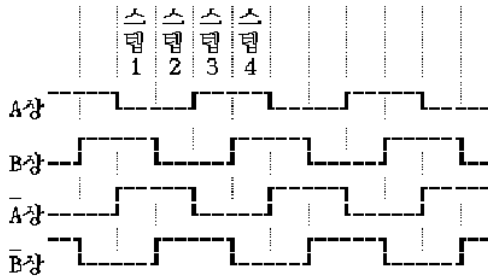
입력이 1상뿐이므로 모터의 온도 상승이 낮고, 전원이 낮아도 된다. 출력토크는 크지만 스텝 했을 때에 감쇠 진동이 큰 난조를 일으키기 쉬우므로 광범위한 스텝 레이트로 회전시킬 때는 주의를 요한다.



스텝	1	2	3	4
전류	$A' \rightarrow A$	$B' \rightarrow B$	$\bar{A}' \rightarrow \bar{A}$	$\bar{B}' \rightarrow \bar{B}$
회전자 위치				

- 2상 여자 : 한번에 2개의 상을 여자 하는 방식

항상 2상이 여자 되어 있으므로 기동 토크가 주어져 난조가 일어나기 어렵다. 상 전환 시에도 반드시 1상은 여자 되어 있으므로 동작 시 제동 효과가 있다. 다만, 모터의 온도 상승이 있고 1상 여자에 비해 2배의 전원 용량을 필요로 한다.



스텝	1	2	3	4
전류	$A' \rightarrow A$ $B' \rightarrow B$	$B' \rightarrow B$ $\bar{A}' \rightarrow \bar{A}$	$\bar{A}' \rightarrow \bar{A}$ $\bar{B}' \rightarrow \bar{B}$	$\bar{B}' \rightarrow \bar{B}$ $A' \rightarrow A$
회전자 위치				

- 1-2상 여자 : 1상 여자와 2상 여자를 조합하여 여자 하는 방식

1 상, 2 상 여자의 용량을 특징을 가지며 스텝 각이 1 상, 2 상에 비교해서 1/2이 된다. 응답 스텝 레이트는 1 상, 2 상 여자의 2배가 된다.



스텝	1	2	3	4
전류	$A' \rightarrow A$	$A' \rightarrow A$ $B' \rightarrow B$	$B' \rightarrow B$	$B' \rightarrow B$ $A' \rightarrow A$
회전자 위치				

- 스텝 모터의 상 진행 순서

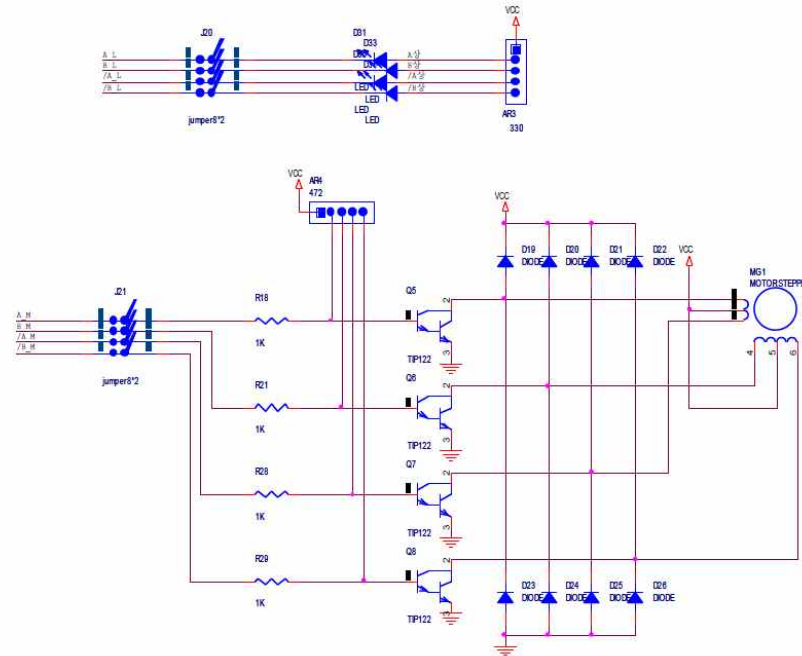
- 전 스텝 : 스텝 모터의 기본 동작

- 반 스텝 : 전 스텝 회전각의 1/2로 회전, 2배의 분해능

- 마이크로 스텝 : 전 스텝 회전각을 수십 조각으로 나눠 회전한다. 수십 배의 분해능으로 부드럽게 회전한다.

## 8-3. Step Motor 구성

### [1] Step Motor 회로도



### [2] 케이블 결선도

장비와 같이 제공되는 USB ISP A/B 케이블을 이용하여 ATmega128 보드와 아래와 같이 연결한다.



### [3] 장비 결선도

아래와 같이 ATmega128 IO 포트와 Step Motor를 연결 하도록 한다.

ATmega128 PORTD	Step Motor 핀
3	A_M
2	B_M
1	/A_M
0	/B_M

## 8-4. Step Motor 제어

### [1] 프로그램 순서

1-1. Step Motor 제어를 위한 프로그램 작성

#### ■ Program Source

```
#include <mega128.h> // Atmega 128 header file
#include <delay.h>    // delay header file

#define STEP_MOTOR    PORTC // STEPPING Motor Data Port
#define STEP_LED      PORTD // STEPPING Motor LED Data Port
#define Motor_Time    10

void STEP_M_CW();      // STEP Motor CW 구동 함수
void STEP_M_CCW();     // STEP Motor CCW 구동 함수
void STEP_M_STOP();    // STEP Motor STOP함수

void main(void)
{
    unsigned char buff0;

    PORTA=0xff; // Port A 초기값
    DDRA=0x00; // Port A 설정, 입력으로 사용

    PORTB=0xff; // Port B 초기값
    DDRB=0xff; // Port B 설정, 출력으로 사용

    PORTC=0xff; // Port C 초기값
    DDRC=0xff; // Port C 설정, 출력으로 사용

    PORTD=0xff; // Port D 초기값
    DDRD=0xff; // Port D 설정, 출력으로 사용

    while (1)
    {
        buff0 = PINA; // buff0에 Port A 값을 저장한다.
        delay_ms(50); // 시간 지연함수 호출
        buff0 = buff0 & 0x03; // buff값에 0x03 AND Mask

        switch(buff0)
        {
```



#### ■ Program Source

```

        case 0x01:                // SW0이 눌리면
            STEP_M_CW();           // STEP Motor CW 구동
            break;
        case 0x02:                // SW1이 눌리면
            STEP_M_CCW();          // STEP Motor CCW 구동
            break;
        default:                  // default값이면
            STEP_M_STOP();         // STEP Motor
STOP
            break;

        } // Switch program

    } // while program

} // main program end

void STEP_M_CW()
{
    STEP_MOTOR = 0x01;            // STEPPING Motor A상
    STEP_LED = 0x01;             // STEPPING Motor LED A상
    delay_ms(Motor_Time);

    STEP_MOTOR = 0x02;            // STEPPING Motor B상
    STEP_LED = 0x02;             // STEPPING Motor LED B상
    delay_ms(Motor_Time);

    STEP_MOTOR = 0x04;            // STEPPING Motor /A상
    STEP_LED = 0x04;             // STEPPING Motor LED /A상
    delay_ms(Motor_Time);

    STEP_MOTOR = 0x08;            // STEPPING Motor /B상
    STEP_LED = 0x08;             // STEPPING Motor LED /B상
    delay_ms(Motor_Time);

} // step cw program end

void STEP_M_CCW()
{

```

## ■ Program Source

```
STEP_MOTOR = 0x08;          // STEPPING Motor /B상
STEP_LED = 0x08;            // STEPPING Motor LED /B상
delay_ms(Motor_Time);

STEP_MOTOR = 0x04;          // STEPPING Motor /A상
STEP_LED = 0x04;            // STEPPING Motor LED /A상
delay_ms(Motor_Time);

STEP_MOTOR = 0x02;          // STEPPING Motor B상
STEP_LED = 0x02;            // STEPPING Motor LED B상
delay_ms(Motor_Time);

STEP_MOTOR = 0x01;          // STEPPING Motor A상
STEP_LED = 0x01;            // STEPPING Motor LED A상
delay_ms(Motor_Time);

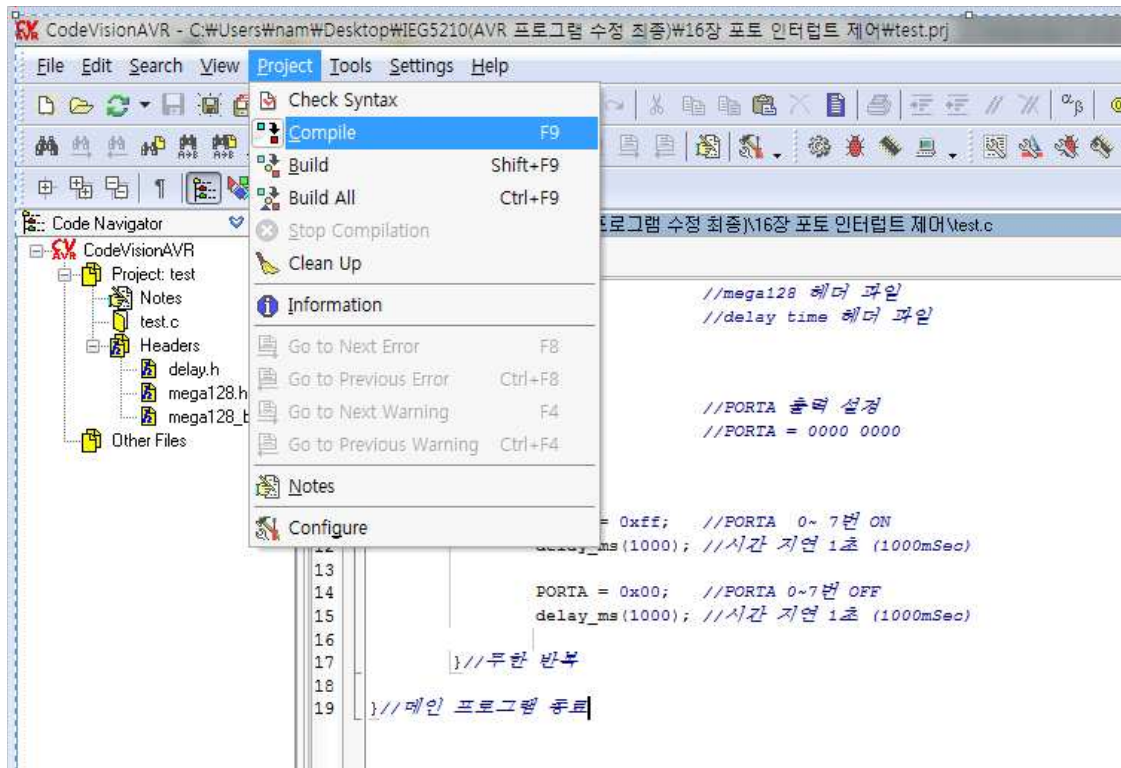
} //step ccw program end

void STEP_M_STOP()
{
    STEP_MOTOR = 0x00;        // STEPPING Motor A상
    STEP_LED = 0x00;          // STEPPING Motor LED

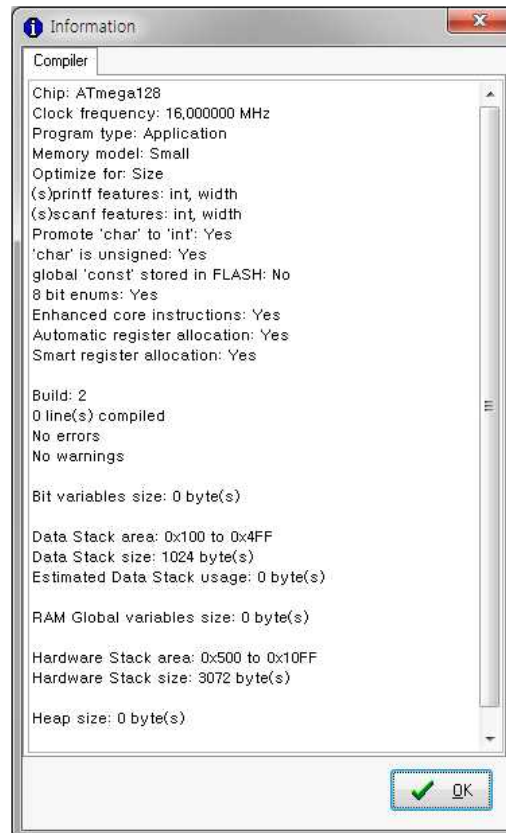
} //step stop program end
```

## [2] CodeVision 프로그램 컴파일

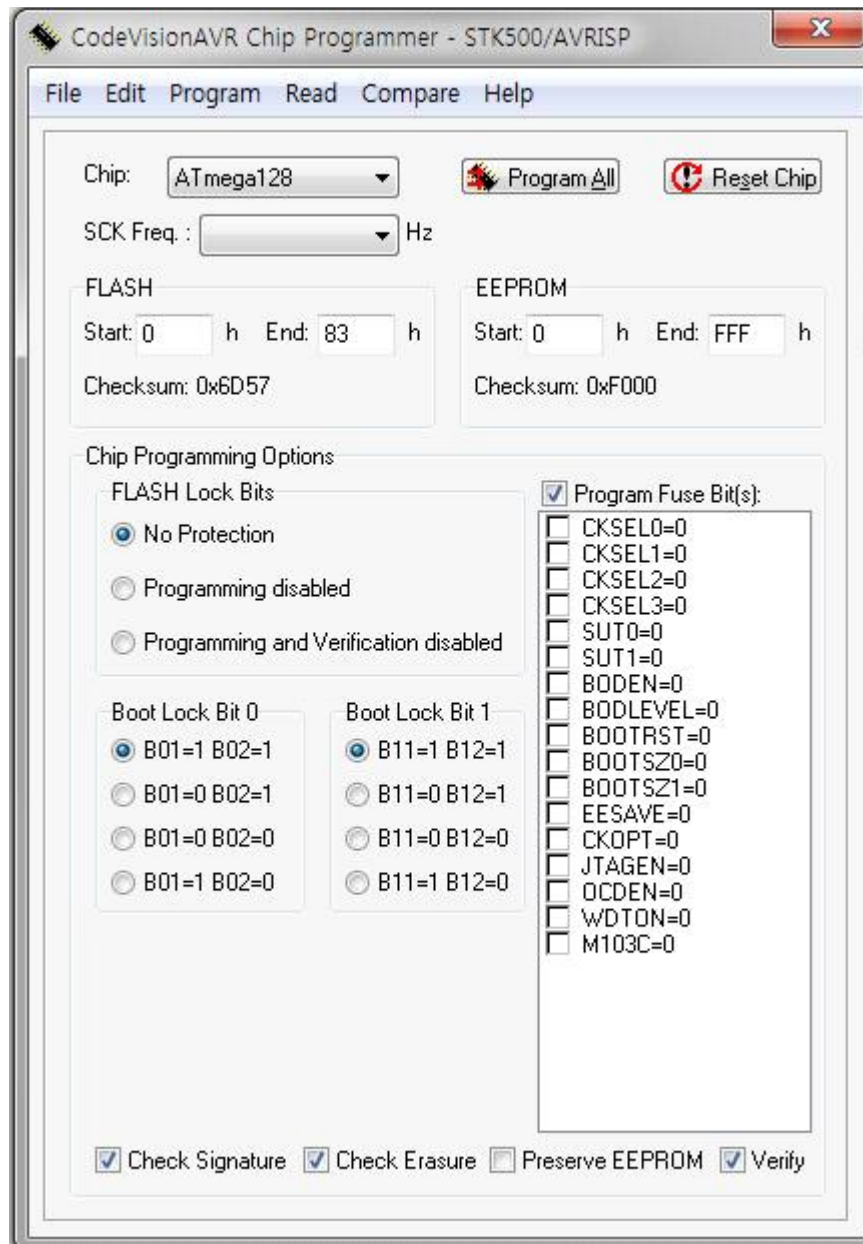
- CodeVision 프로그램에서 Project 메뉴를 선택하여 Compile을 실행 하도록 한다.



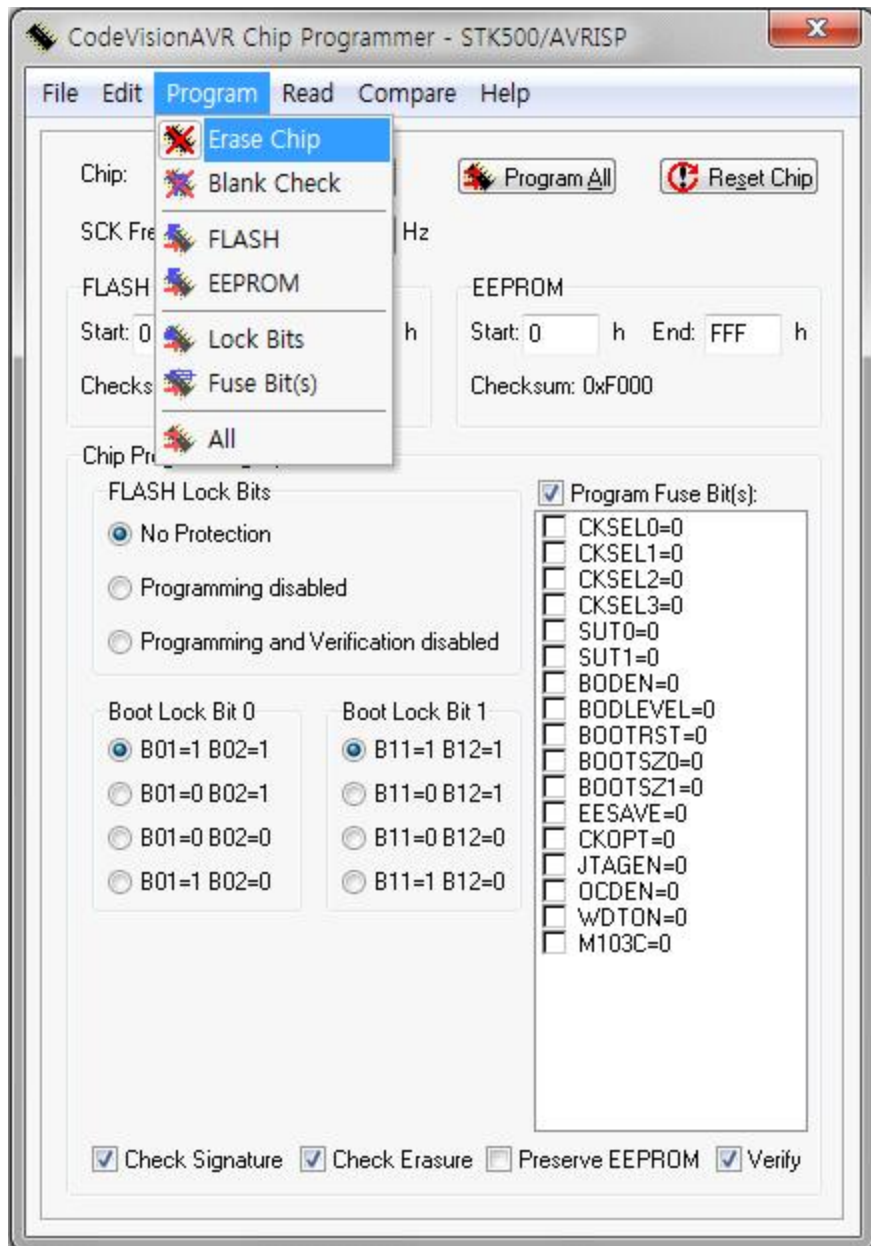
- 프로그램에서 에러가 없으면 아래와 같이 컴파일 정보가 출력 된다.



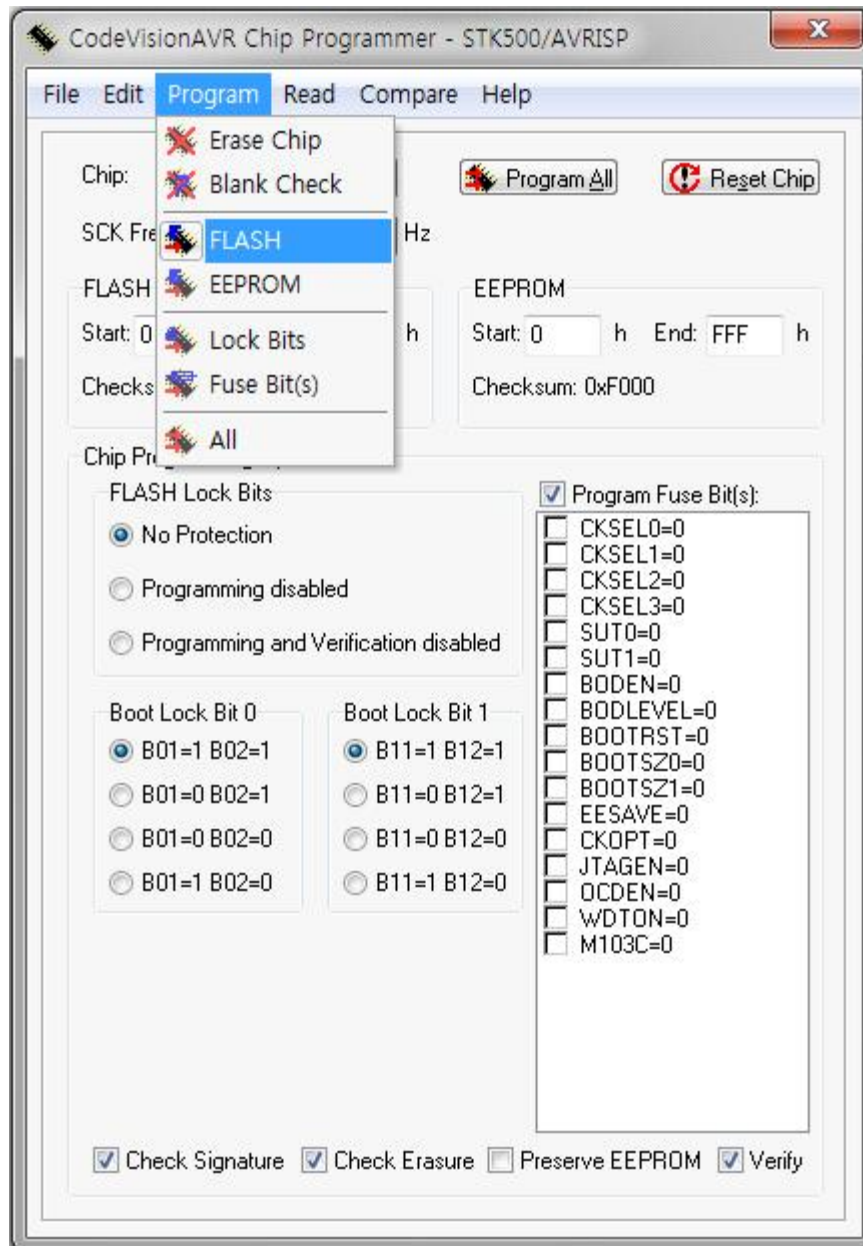
- Tools → Chip Programmer 클릭(단축키 : Shift+F4)



- Erase chip 클릭(chip 내에 프로그램이 남아 있을수 있으므로 프로그램 다운로드 전에 반드시 Erase Chip을 권장하도록 한다. )

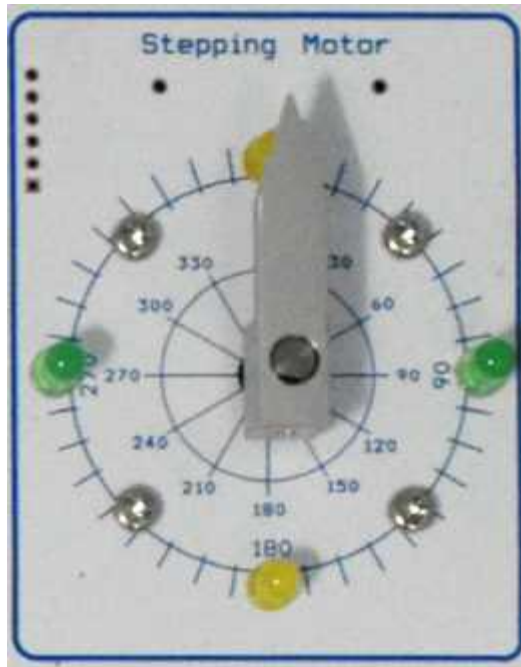


- Chip Flash 클릭하여 프로그램을 다운로드 하도록 한다.



### [3] 동작 확인

- Step Motor가 시계 방향으로 360°씩 회전하는 것을 확인 할수 있다.



### [4] 실습 과제

- 4-1. 과제명 : 180°씩 끊어서 1상 제어를 CW 방향으로 회전하도록 제어 한다.
- 4-2. 과제명 : 180°씩 끊어서 2상 제어를 CCW 방향으로 제어 하도록 한다.



## SECTION

## 09

## DC Motor 제어 실습

## 9-1. 학습 목표

- DC Motor 구조와 동작 방법
- DC Motor 회전 방향 제어
- L298 DC Motor Driving IC를 이용한 모터 제어

## 9-2. 기초 이론

## (1) Step Motor 란?

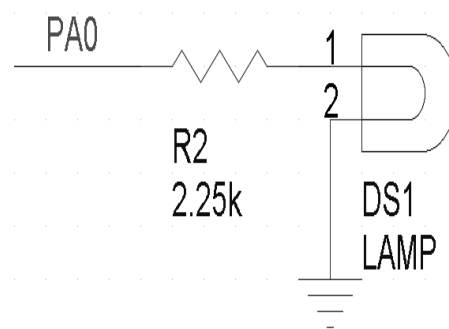
Digital은 Logic 'L', '0'과 'H', '1' 동작을 한다. 일반적인 TTL, CMOS 계열 IC의 조합으로 회로가 구성되어 있으면 구동에 있어서 크게 문제가 없다.

그러나 그 외의 Device ( Motor, Lamp 등)를 사용 할 때는 구동하는데 있어서 문제가 많이 있다.

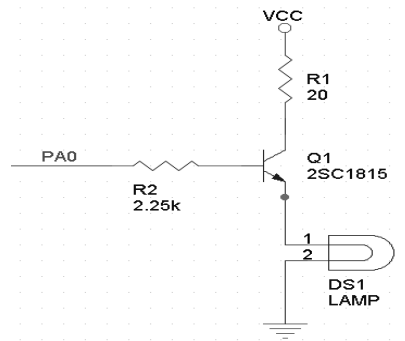
Motor, Lamp등의 Device들이 정상적인 동작을 하려면 일정량의 전압과 전류가 필요하다. 하지만, Digital Signal은 +5V와 소량의 전류로만 동작을 하므로 위의 Device는 정상적으로 동작할 수 없다.

그래서 구동 Driver가 필요한 것이다. 이것은 소량의 전압과 전류로서 큰 전압과 전류를 만들어 올바른 동작을 유도한다.

아래의 그림은 구동 driver의 미사용 예이다. Digital 출력인 Port에 Logic 'H', '1', +5V를 출력하여도 Lamp는 점등이 안 된다.



구동 driver 미사용 예



구동 driver 사용 예

그 이유는 Lamp가 동작을 하기 위해서는 최소한의 전압과 전류가 있어야 하지만 digital 출력에서는 전류량이 아주 작아서 Lamp를 구동시키지 못한다.

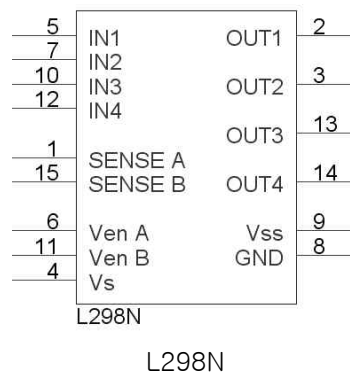
하지만 위 그림은 구동 driver TR을 사용하여 위에서 설명한 실험을 수행하면 TR의 On, Off 동작에 의하여 Lamp는 main 전원(VCC)이 인가되어 점등된다.

## □ L298N Motor Driver ( DC Motor용 )

■ L298N Motor Driver는 앞에서 설명한 driver IC이다.

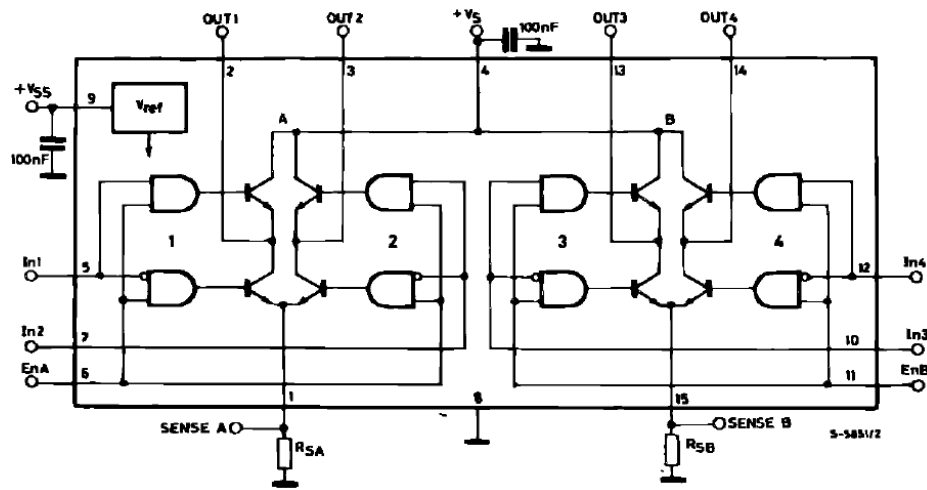
다시 말해, TTL Level의 5V, 소량의 전류 출력으로 DC Motor를 구동시킬 때 필요한 Device이다.

동작 사항을 살펴보면 아래의 표와 같이 IN1, IN2와 OUT1,2가 1조, IN3, IN4 와 OUT3,4가 2조이다.

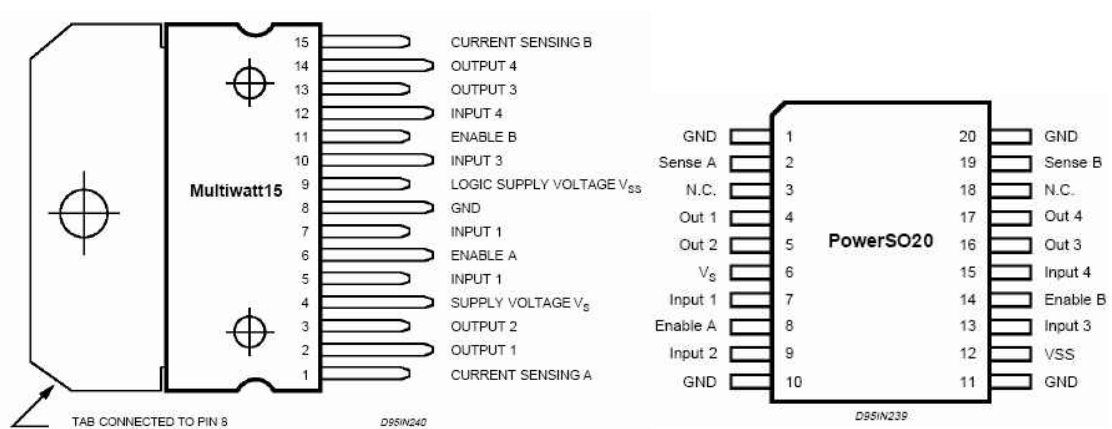


DC Motor 구동 신호

IN1,3	IN2,4	구동
H	L	Forward
L	H	Reverse
IN1 = IN2		Fast motor stop



L298N 구조



L298N 핀 배치도

## □ DC Motor

DC Motor는 말 그대로 DC전압을 사용하여 구동하는 Motor이다. 입력 전류에 비례해서 회전력이 발생한다.

회전력  $T = K_t \cdot I$  , 여기서  $K_t$ 는 토크 상수

입력 전류는 입력 전압에 비례하고 모터 권선 저항에 반비례한다.

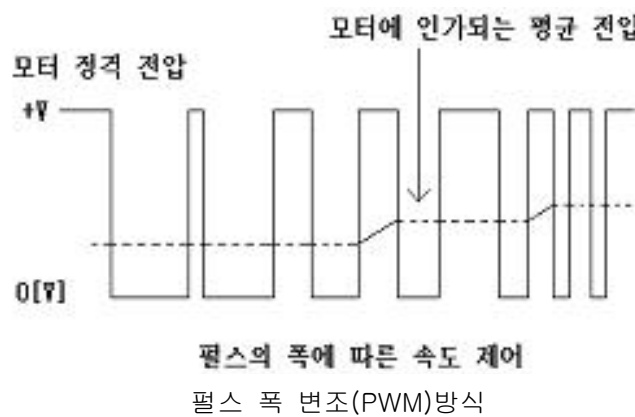
입력 전류  $I = V / R$  , 여기서  $R$ 은 전기자 저항

DC Motor는 입력단자에 순 전압을 가하면 시계방향으로 회전하고, 입력단자에 역 전압을 가하면 반시계방향으로 회전한다.



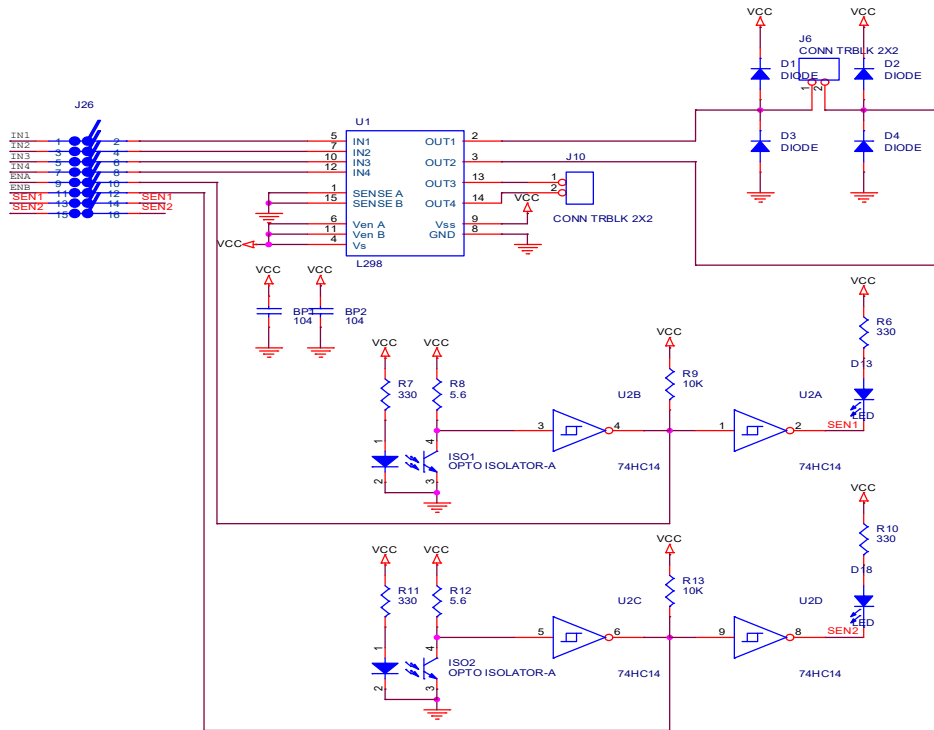
DC Motor 및 전압배선방법

일반적인 DC Motor의 속도제어 방식은 펄스 폭 변조(PWM)방식을 많이 사용한다.



## 9-3. DC Motor 구성

### [1] DC Motor 회로도



### [2] 케이블 결선도

장비와 같이 제공되는 USB ISP A/B 케이블을 이용하여 ATmega128 보드와 아래와 같이 연결한다.



### [3] 장비 결선도

아래와 같이 ATmega128 IO 포트와 DC Motor를 연결 하도록 한다.

ATmega128 PORTB	Step Motor 핀
0	IN1
1	IN2
2	ENA

## 9-4. DC Motor 제어

### (1) 프로그램 순서

1-1. DC Motor 제어를 위한 프로그램 작성

#### ■ Program Source

```
#include <mega128.h> // Atmega 128 header file
#include <delay.h>    // delay header file

#define DCM_IN1      PORTB.0      // DC Motor L298N IN1 Input신호
#define DCM_IN2      PORTB.1      // DC Motor L298N IN2 Input신호
#define DCENA        PORTB.2      // DC Motor EN Input신호

void DC_M_CW();          // DC Motor CW 구동 함수
void DC_M_CCW();        // DC Motor CCW 구동 함수
void DC_M_STOP();       // DC Motor STOP함수

void main(void)
{
    unsigned char buff0;

    PORTA=0xFF; // Port A 초기값
    DDRA=0x00;  // Port A 설정, 입력으로 사용

    PORTB=0x00; // Port B 초기값
    DDRB=0xFF;  // Port B 설정, 출력으로 사용

    while (1)
    {
        buff0 = PINA;          // buff0에 Port A 값을 저장한다.
        delay_ms(50);          // 시간 지연함수 호출
        buff0 = buff0 & 0x03;   // buff값에 0x03 AND Mask

        switch(buff0)
        {
            case 0x02:          // SW0이 눌리면
                DCENA = 1;
                DC_M_CW();      // DC Motor CW 구동
                break;
            case 0x01:          // SW1이 눌리면
                DCENA = 1;
```

## ■ Program Source

```

        DC_M_CCW();          // DC Motor CCW 구동
        break;

    default:                  // default값이면
        DCENA = 0;
        DC_M_STOP();         // DC Motor STOP
        break;

    } // Switch program end

} // while program end

} // main program end

void DC_M_CW()
{
    DCM_IN1 = 1;             // DC Motor L298N IN1 '1' Input
    delay_ms(100);
    DCM_IN2 = 0;             // DC Motor L298N IN2 '0' Input
    delay_ms(100);

    DCM_IN1 = 0;             // DC Motor L298N IN1 '0' Input
    delay_ms(100);
    DCM_IN2 = 0;             // DC Motor L298N IN2 '0' Input

} // motor cw program end

void DC_M_CCW()
{
    DCM_IN1 = 0;             // DC Motor L298N IN1 '0' Input
    delay_ms(100);
    DCM_IN2 = 1;             // DC Motor L298N IN2 '1' Input
    delay_ms(100);
    DCM_IN1 = 0;             // DC Motor L298N IN1 '0' Input
    delay_ms(100);
    DCM_IN2 = 0;             // DC Motor L298N IN2 '0' Input

} // motor ccw program end

void DC_M_STOP()

```

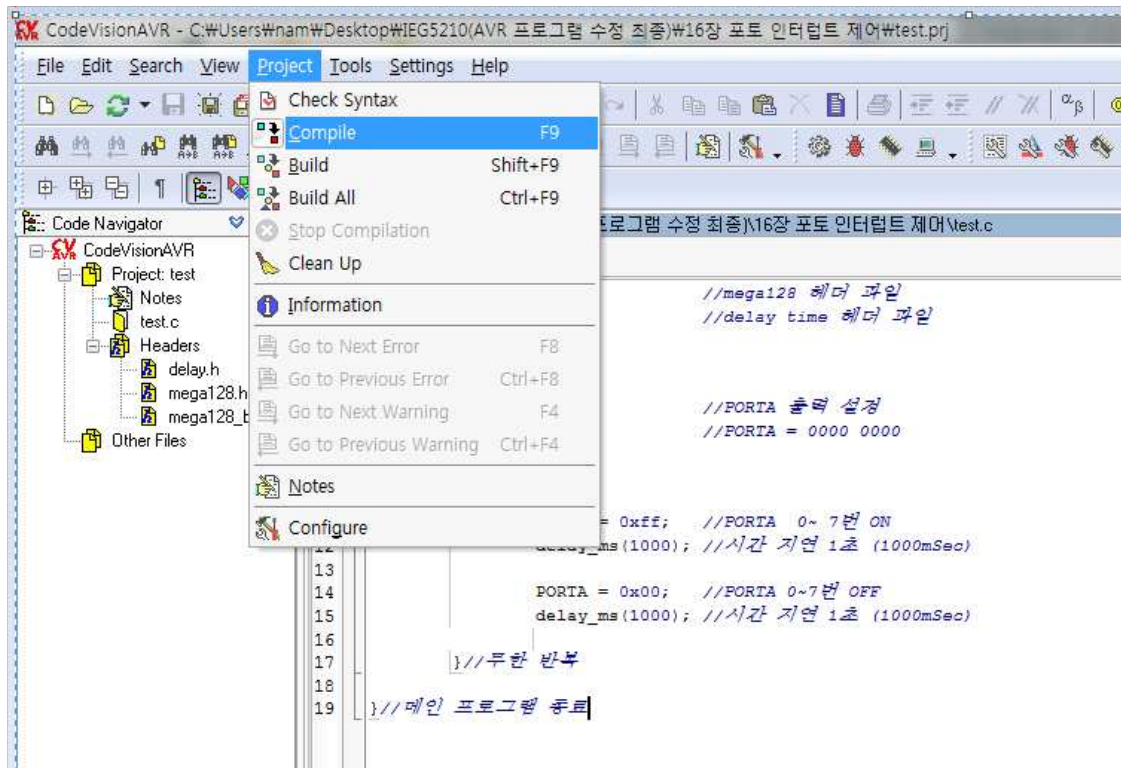


## ■ Program Source

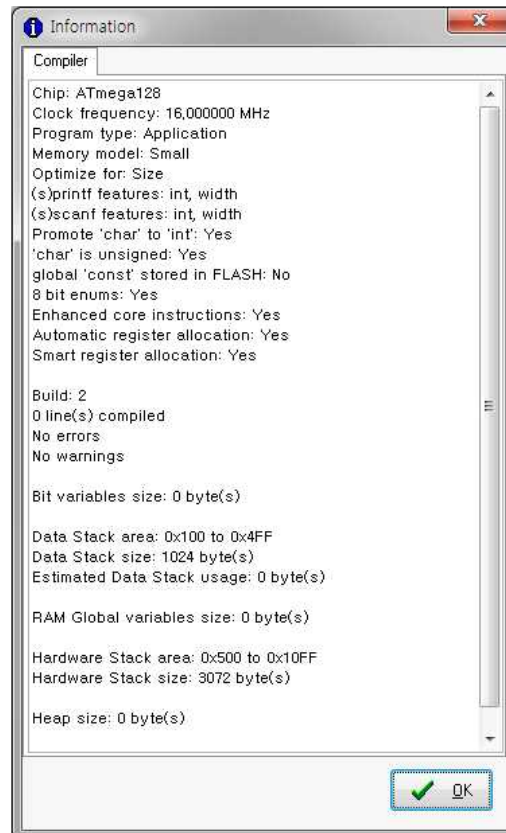
```
{  
    DCM_IN1 = 0;          // DC Motor L298N IN1 '0' Input  
    DCM_IN2 = 0;          // DC Motor L298N IN2 '0' Input  
  
} //motor stop program end
```

## [2] CodeVision 프로그램 컴파일

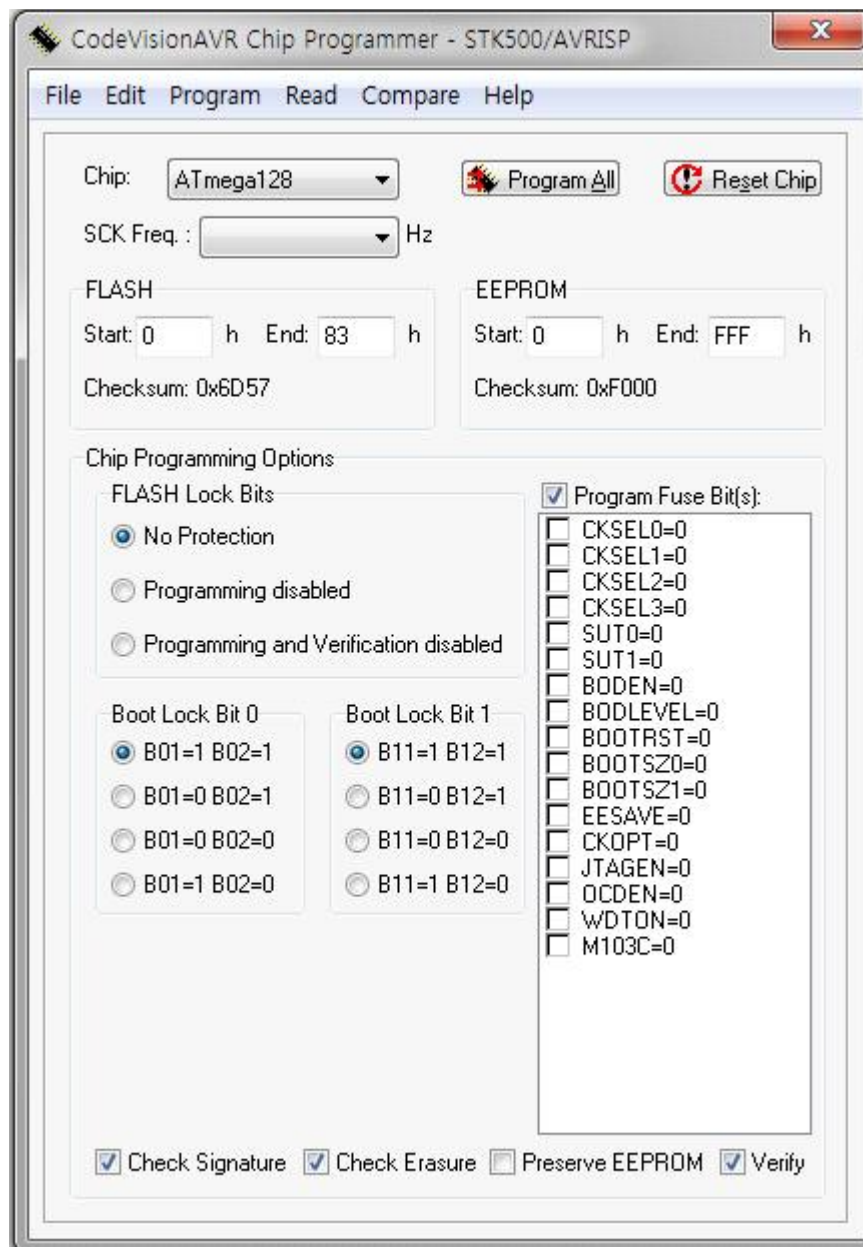
- CodeVision 프로그램에서 Project 메뉴를 선택하여 Compile을 실행 하도록 한다.



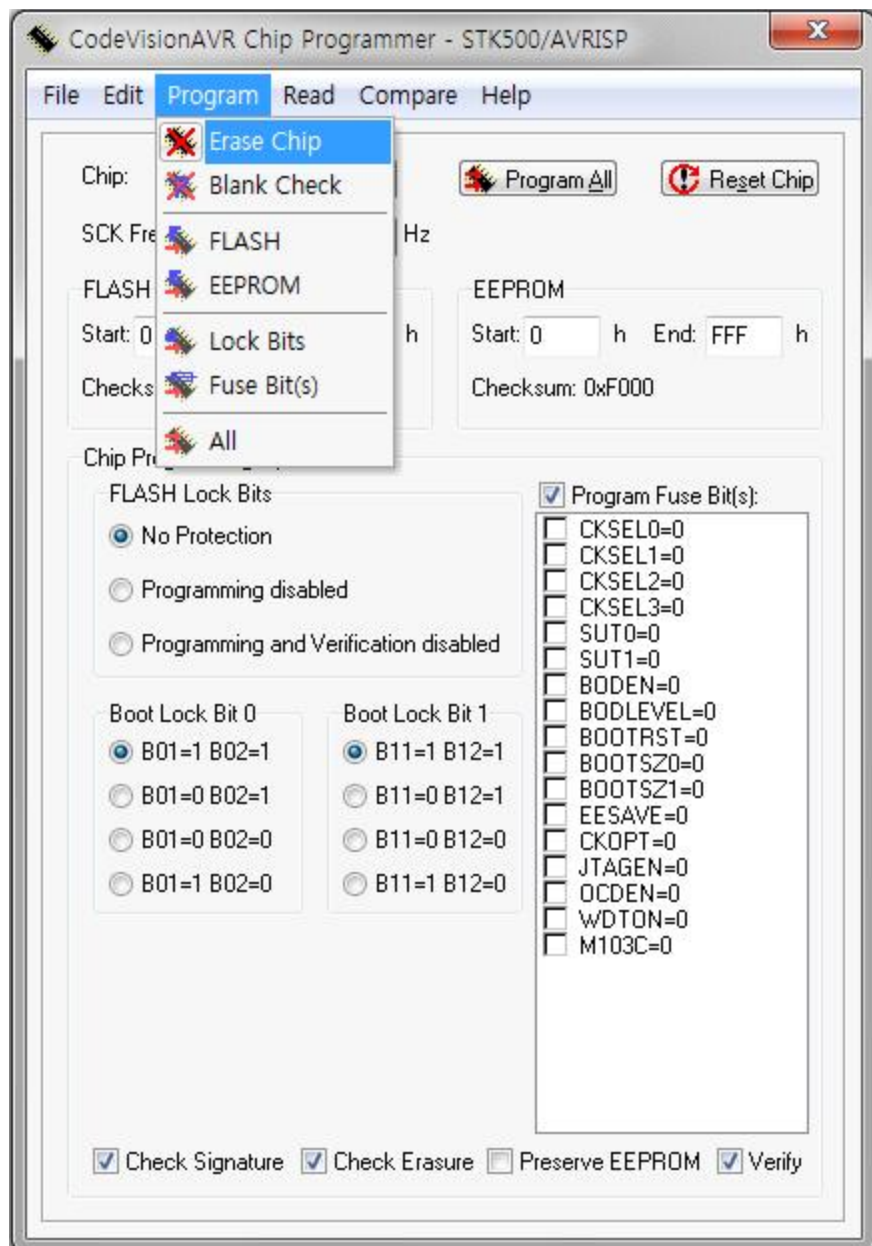
- 프로그램에서 에러가 없으면 아래와 같이 컴파일 정보가 출력 된다.



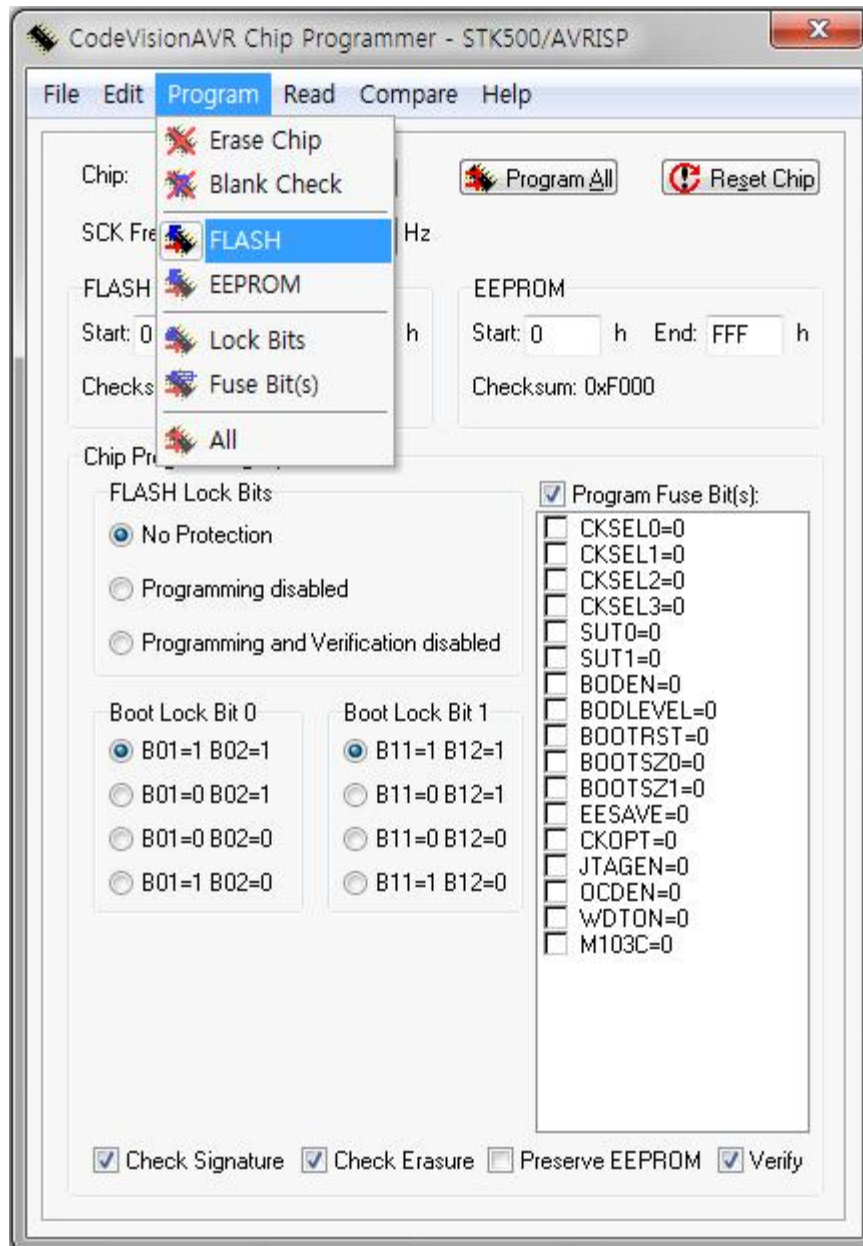
- Tools → Chip Programmer 클릭(단축키 : Shift+F4)



- Erase chip 클릭(chip 내에 프로그램이 남아 있을수 있으므로 프로그램 다운로드 전에 반드시 Erase Chip을 권장하도록 한다. )



- Chip Flash 클릭하여 프로그램을 다운로드 하도록 한다.



### [3] 동작 확인

- DC Motor 가 시계 방향으로 2초간 회전, 시계 반대 방향으로 2초가 회전, 정지 2초를 반복 동작을 하게 된다.



### [4] 실습 과제

4-1. 과제명 : 스위치 1번이 입력되면 CW 방향으로 회전, 스위치 2번이 입력되면 CCW 방향으로 회전을 하도록 한다. 단 2개의 스위치 신호가 없으면 정지 상태이며 1, 2 스위치가 눌렸을때만 동작 하도록 한다.

## SECTION

# 10 CDS 제어 실습

## 10-1. 학습 목표

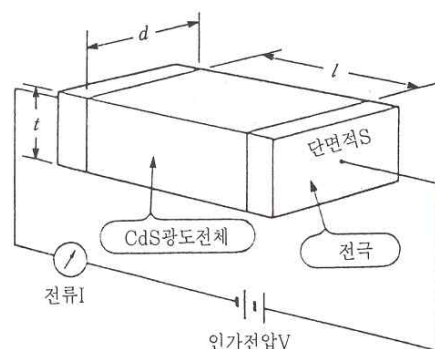
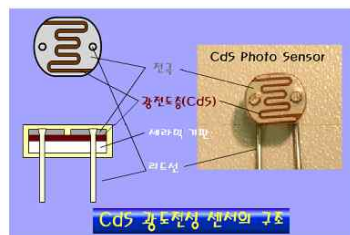
- CDS 구조와 동작 방법
- 광량을 통한 CDS 제어
- CDS 광량값 Segment 출력 제어

## 10-2. 기초 이론

### (1) CDS 란?

크기는 외경이 5 ~ 32mm인 각종 타입이 있다. CdS 셀은 습기에 약하기 때문에 기밀 봉합되어 있는데, 그 방법으로는 글라스 봉입 형과 메탈 케이스 형이 있으며, 저가격화를 꾀한 것으로 플라스틱 케이스 형과 수지 코팅 형이 있으나, 코팅 재료의 개량으로 내후성의 신뢰성이 향상되고 저가격, 형상의 자유도에 의해서 현재는 수지 코팅 형이 많이 보급되어 있다. CdS 셀의 특징은 다음과 같다.

- ▶ 가시광선 영역에 분광 감도를 가지고 있다.
- ▶ 교류 동작이 가능하다.
- ▶ 잡음에 강하다.
- ▶ 다른 포토 센서에 비해 응답 속도가 느리다.
- ▶ 가시광선 영역에 분광 감도를 가지고 있으므로 주위의 빛에 교란당하기 쉽다.



CdS

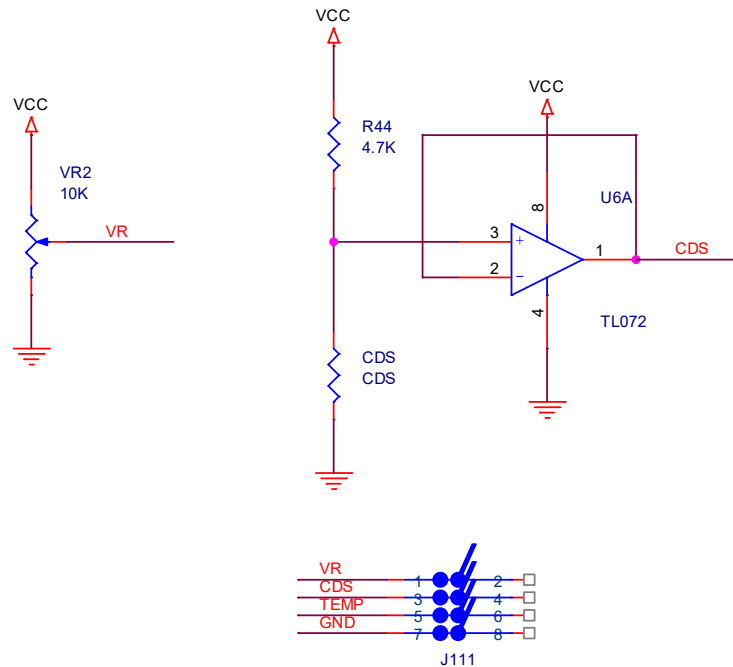
CdS 셀의 구조를 나타내었다. CdS 셀의 양단에 전극을 설치하고 암흑 속에 놓고 전압을 인가하면 전류계에 미소한 전류(암전류)가 흐른다. 이 전류는 미소하기 때문에 소자는 높은 저항을 나타낸다. 이 CdS 셀에 빛을 비추면 전류(명전류)가 흐른다.(명전류는 보통 신호성분을 나타내며, 암전류는 잡음 성분을 나타낸다.)



CdS 셀의 저항 값은 조사 광이 강해지면 낮아지고, 조사 광이 약해지면 높아진다. 이와 같이 CdS 셀의 저항 값은 빛의 강약에 의해 변화한다. 이 성질을 이용하는 것이 CdS 셀의 기본 동작 원리이다.

## 10-3. CDS 구성

### (1) CDS 회로도



### (2) 케이블 결선도

장비와 같이 제공되는 USB ISP A/B 케이블을 이용하여 ATmega128 보드와 아래와 같이 연결한다.



**[3] 장비 결선도**

아래와 같이 아두이노 IO 포트와 Segment를 연결 하도록 한다.

ATmega128 PORTB	Segment Enable 핀
3	Q3
2	Q2
1	Q1
0	Q0

ATmega128 PORTA	Segment 핀
7	DP
6	G
5	F
4	E
3	D
2	C
1	B
0	A

ATmega128 PORTF	CDS 핀
0	CDS

## 10-4. CDS 제어

### (1) 프로그램 순서

1-1. CDS 출력을 위하여 프로그램 작성

#### ■ Program Source

```
#include <mega128.h> // Atmega 128 헤더파일
#include <delay.h>    // delay 헤더파일

#define ADC_VREF_TYPE 0x40

#define Q0    PORTB.0           //PortB.0 bit FND Q0
#define Q1    PORTB.1           //PortB.1 bit FND Q1
#define Q2    PORTB.2           //PortB.2 bit FND Q2
#define Q3    PORTB.3           //PortB.3 bit FND Q3
#define FND    PORTA            //Port A 를 FND 설정

void FND_DISPLAY(unsigned int Speed, unsigned int value)
{
    // FND data Array 0 ~ F
    // data value      '0' '1' '2' '3' '4' '5' '6' '7' '8' '9' 'A' 'B'
    'C' 'D' 'E' 'F' 'dot'
    unsigned char
fnd[17]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xd8,0x80,0x98,0x88,0x83,0xc6,0xa1,0x86,
0x8e,0x7f};

    unsigned int fnd4, fnd3, fnd2, fnd1, remainder;

    fnd4 = value / 1000;           // buff1000변수에 i값을 1000으로 나눈 몫 저장
    remainder = value % 1000;     // remainder변수에 i값을 1000으로 나눈 나머
    지 저장

    fnd3 = remainder / 100;        // buff100변수에 i값을 100으로 나눈
    몫 저장
    remainder = remainder % 100; // remainder변수에 i값을 100으로 나눈 나머
    지 저장

    fnd2 = remainder / 10;         // buff10변수에 i값을 10으로 나눈 몫 저장
    remainder = remainder % 10;    // remainder변수에 i값을 10으로 나눈 나머
    지 저장
```

## ■ Program Source

```

    fnd1 = remainder;                // buff1변수에 remainder변수 값 저장

    while(Speed--)
    {
        FND = fnd[fnd4];            // fnd 변수의 값을 Port 1로 출력
        Q3 = 0;                    // PortB.3 0 출력, Q3에 0 출력
        Q2 = 1;                    // PortB.2 1 출력, Q2에 1 출력
        Q1 = 1;                    // PortB.1 1 출력, Q1에 1 출력
        Q0 = 1;                    // PortB.0 1 출력, Q0에 1 출력
        delay_us(500);

        FND = fnd[fnd3];            // fnd 변수의 값을 Port 1로 출력
        Q3 = 1;                    // PortB.3 1 출력, Q3에 1 출력
        Q2 = 0;                    // PortB.2 0 출력, Q2에 0 출력
        Q1 = 1;                    // PortB.1 1 출력, Q1에 1 출력
        Q0 = 1;                    // PortB.0 1 출력, Q0에 1 출력
        delay_us(500);

        FND = fnd[fnd2];            // fnd 변수의 값을 Port 1로 출력
        Q3 = 1;                    // PortB.3 1 출력, Q3에 1 출력
        Q2 = 1;                    // PortB.2 1 출력, Q2에 1 출력
        Q1 = 0;                    // PortB.1 0 출력, Q1에 0 출력
        Q0 = 1;                    // PortB.0 1 출력, Q0에 1 출력
        delay_us(500);

        FND = fnd[fnd1];            // fnd 변수의 값을 Port 1로 출력
        Q3 = 1;                    // PortB.3 1 출력, Q3에 1 출력
        Q2 = 1;                    // PortB.2 1 출력, Q2에 1 출력
        Q1 = 1;                    // PortB.1 1 출력, Q1에 1 출력
        Q0 = 0;                    // PortB.0 0 출력, Q0에 0 출력
        delay_us(500);
    }
}

// Read the AD conversion result
unsigned int ADC(unsigned char channel)
{
    ADMUX= channel | ADC_VREF_TYPE;

```

## ■ Program Source

```
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
    ADCSRA|=0x10;
return ADCW;
}

void main(void)
{

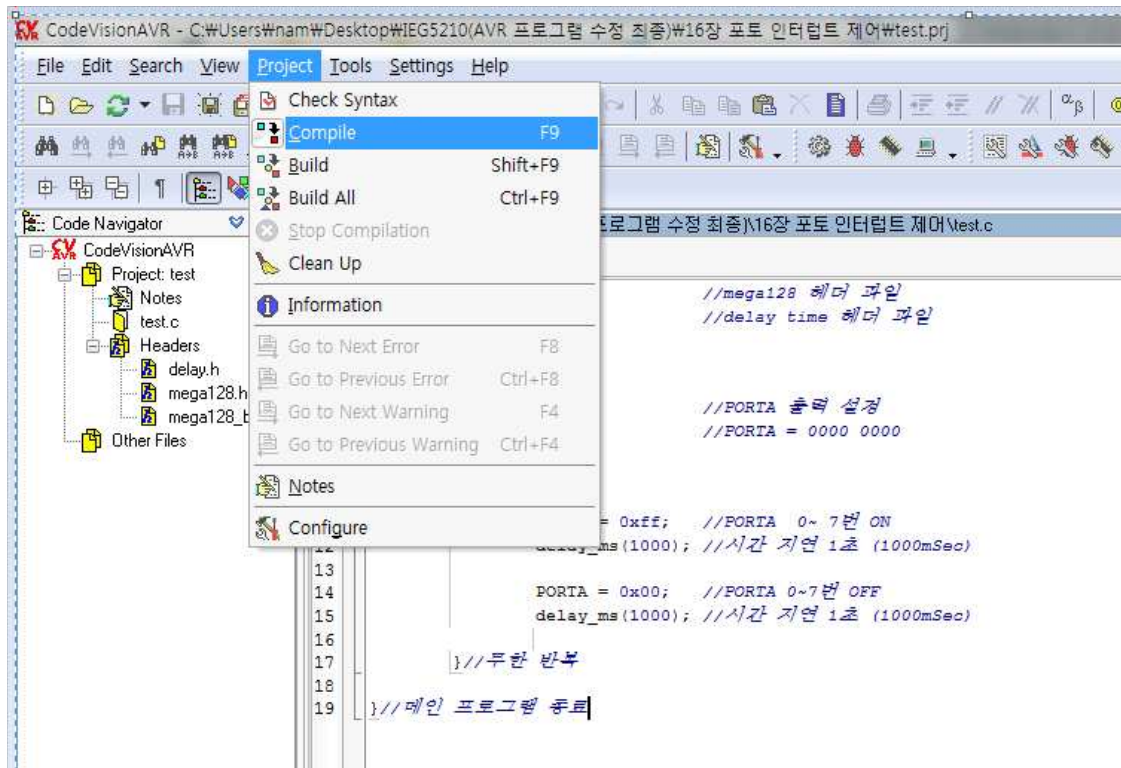
    unsigned int ADC_Value;

    PORTA=0x00;
    DDRA=0xFF;

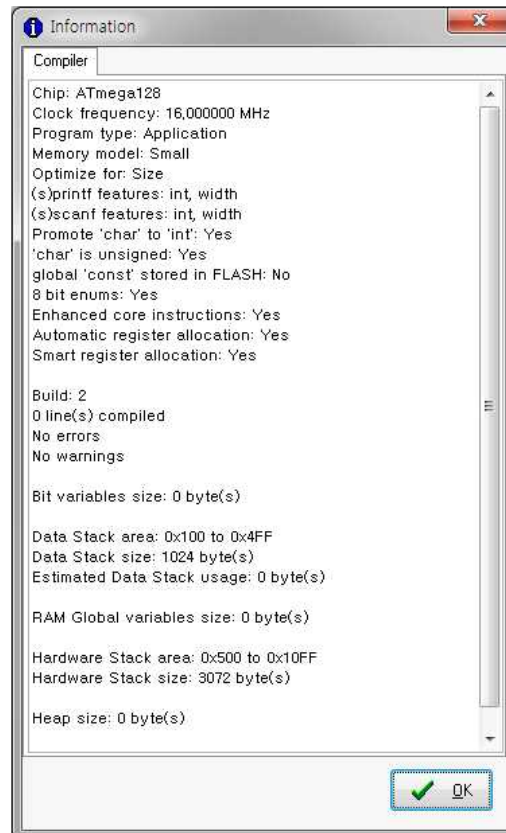
    PORTB=0x00;
    DDRB=0xFF;
    // ADC 클럭 주파수: 125.000 kHz
    // ADC 기준 전압 : AVCC 핀 사용
    ADCSRA=0x87;
    while (1)
    {
        ADC_Value = ADC(0);           //0번 채널 -> PORTF.0 사용
                                     //0~7 까지 8채널을 사용 할수 있다.
        FND_DISPLAY(60,ADC_Value);    //FND ADC값 출력
    }
}
```

## [2] CodeVision 프로그램 컴파일

- CodeVision 프로그램에서 Project 메뉴를 선택하여 Compile을 실행 하도록 한다.

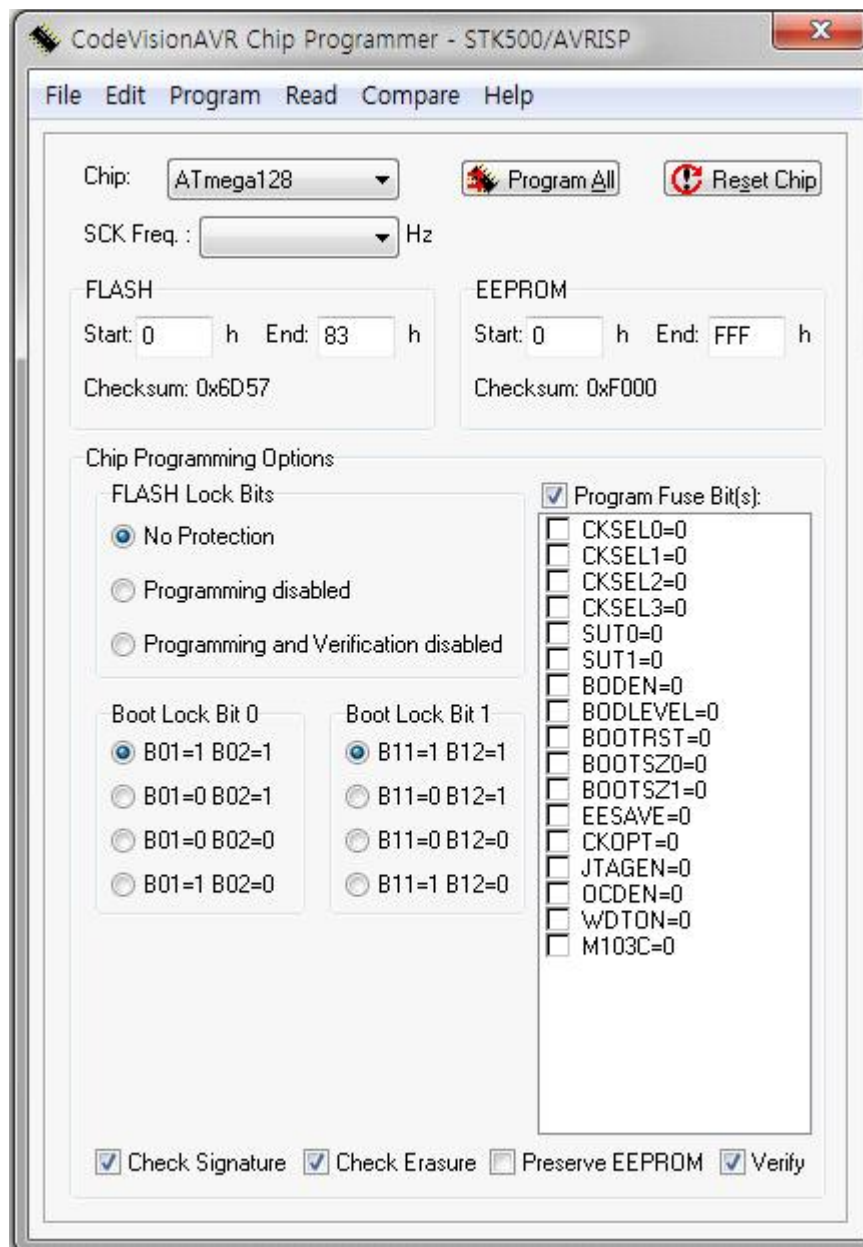


- 프로그램에서 에러가 없으면 아래와 같이 컴파일 정보가 출력 된다.

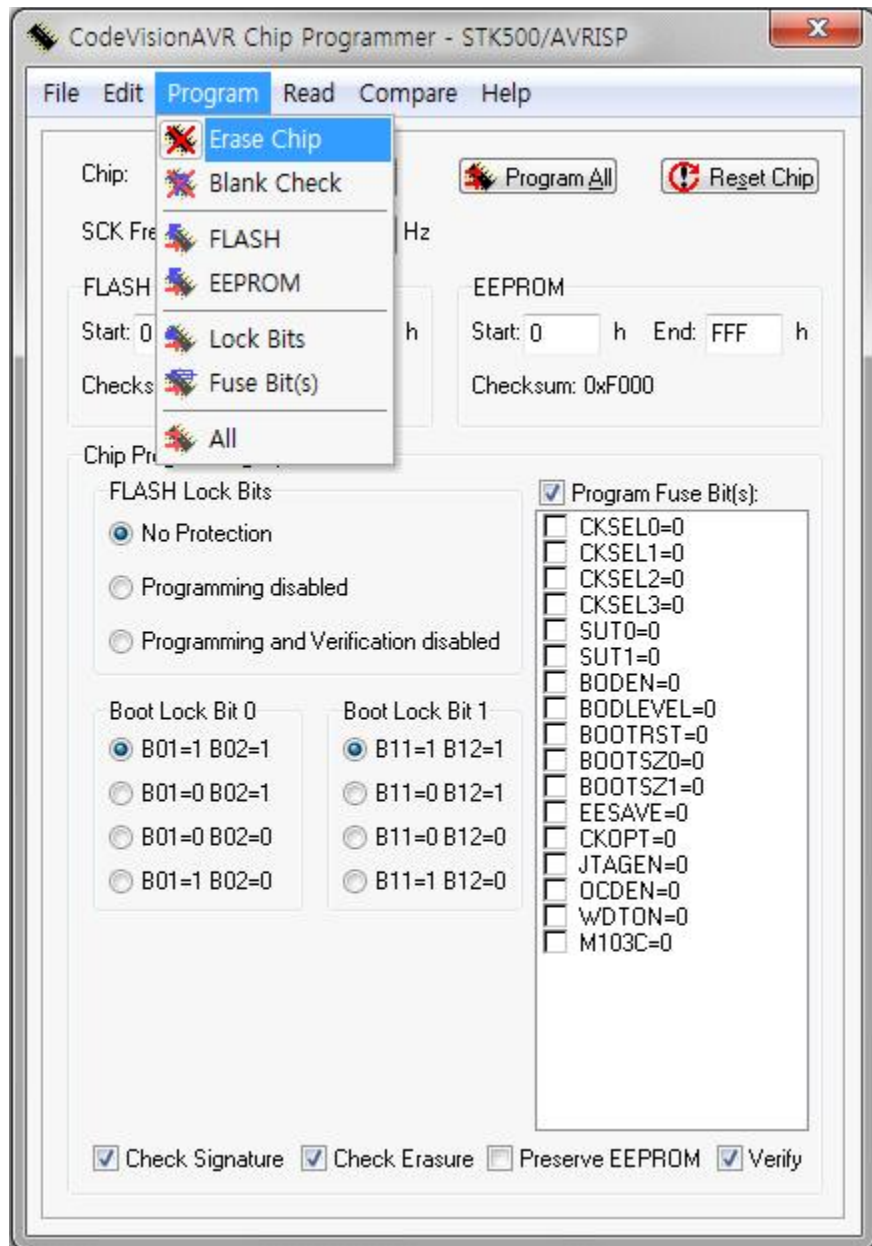




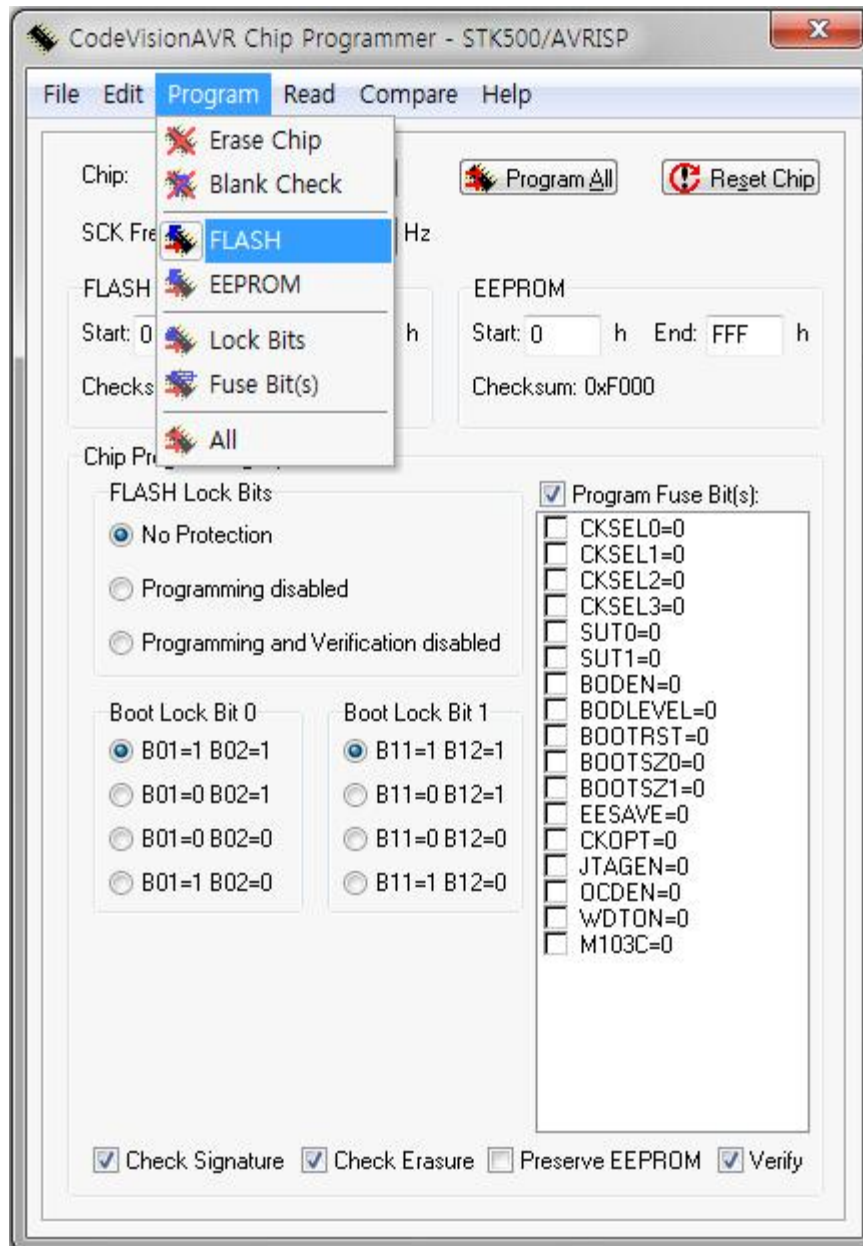
- Tools → Chip Programmer 클릭(단축키 : Shift+F4)



- Erase chip 클릭(chip 내에 프로그램이 남아 있을수 있으므로 프로그램 다운로드 전에 반드시 Erase Chip을 권장하도록 한다. )

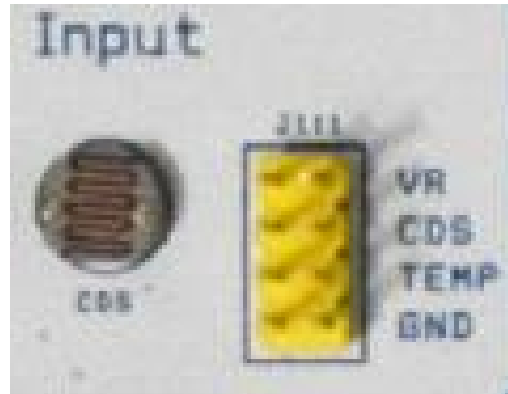


- Chip Flash 클릭하여 프로그램을 다운로드 하도록 한다.



### [3] 동작 확인

- CDS(광량)에 광량을 조정 하였을 때 Segment에 광량 값이 변화 하는 동작을 하게 된다.



### [4] 실습 과제

- 4-1. 과제명 : CDS값에 따른 DC CW, CCW 제어
- 4-2. 과제명 : CDS값에 따른 LCD에 Display 출력

## SECTION

## 11

## Interrupt 제어 실습

## 11-1. 학습 목표

- Interrupt 구조와 동작 방법
- Interrupt Relay 제어
- Interrupt LED 출력 제어

## 11-2. 기초 이론

## (1) Interrrupt 란?

현재 어떤 프로그램이 실행중인 상태에서 인터럽트의 조건을 만족하는 어떤 변화가 감지되었을 경우, 현재 실행되는 프로그램을 중단시키고 인터럽트 처리 프로그램을 실행되는 것을 인터럽트라고 한다. 인터럽트를 볼 수 있는 가장 쉬운 예는 컴퓨터를 사용하다가 갑자기 뜨는 “블루 스크린”을 들 수 있다. 블루스크린이 뜨는 이유는 정확히 알 수 없지만 OS가 실행되는 영역인 커널 모드에서 치명적인 오류가 발생했을 경우 인터럽트를 발생시켜 현재 프로그램을 중지 시키고 블루스크린을 띄워 오류를 알려주게 된다.

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

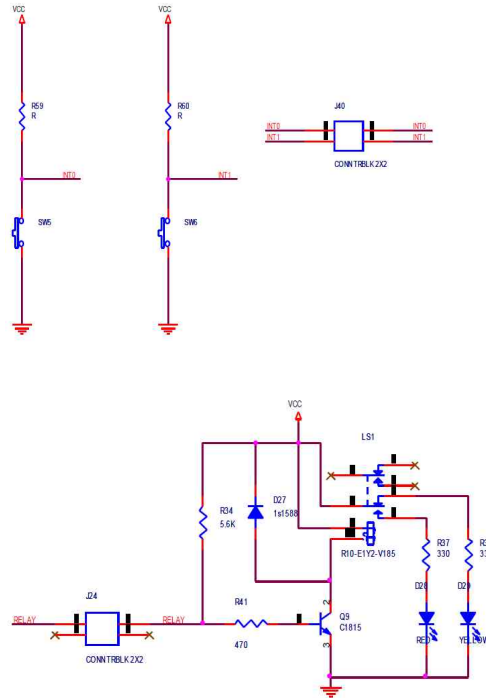
Technical information:

*** STOP: 0x0000000A (0x0227001d, 0x00000002, 0x00000000, 0x804eba3a)

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
```

## 11-3. Interrupt 구성

### (1) Interrupt or Relay 회로도



### (2) 케이블 결선도

장비와 같이 제공되는 USB ISP A/B 케이블을 이용하여 ATmega128 보드와 아래와 같이 연결한다.



### [3] 장비 결선도

아래와 같이 아두이노 IO 포트와 Interrupt, Relay를 연결 하도록 한다.

ATmega128 PORTD	Interrupt
0	Interrupt INT0

ATmega128 PORTC	Relay
0	Relay In

## 11-4. Interrupt 제어

### (1) 프로그램 순서

1-1. Interrupt 제어를 위한 프로그램 작성

#### ■ Program Source

```
#include <mega128.h> // ATmega128 header file
#include <delay.h>    // delay header file

void init_port(void);          // PORT 초기화
void init_intterrupt(void);    // Intterrupt 초기화

unsigned char Int0_test=0;

interrupt [EXT_INT0] void ext_int0_isr(void)
{
    Int0_test++;
    delay_ms(500);
    #asm("sei")                // 전체 인터럽트 허용
}

void main(void)
{
    init_port();
    init_intterrupt();

    while(1)
    {
        PORTC = ~Int0_test;
    }
}

void init_port(void)
{
    // Port C init... Output Port
    PORTC=0x00;
    DDRC=0xff;

    PORTD=0xff;
    DDRD=0xf0;
}
```



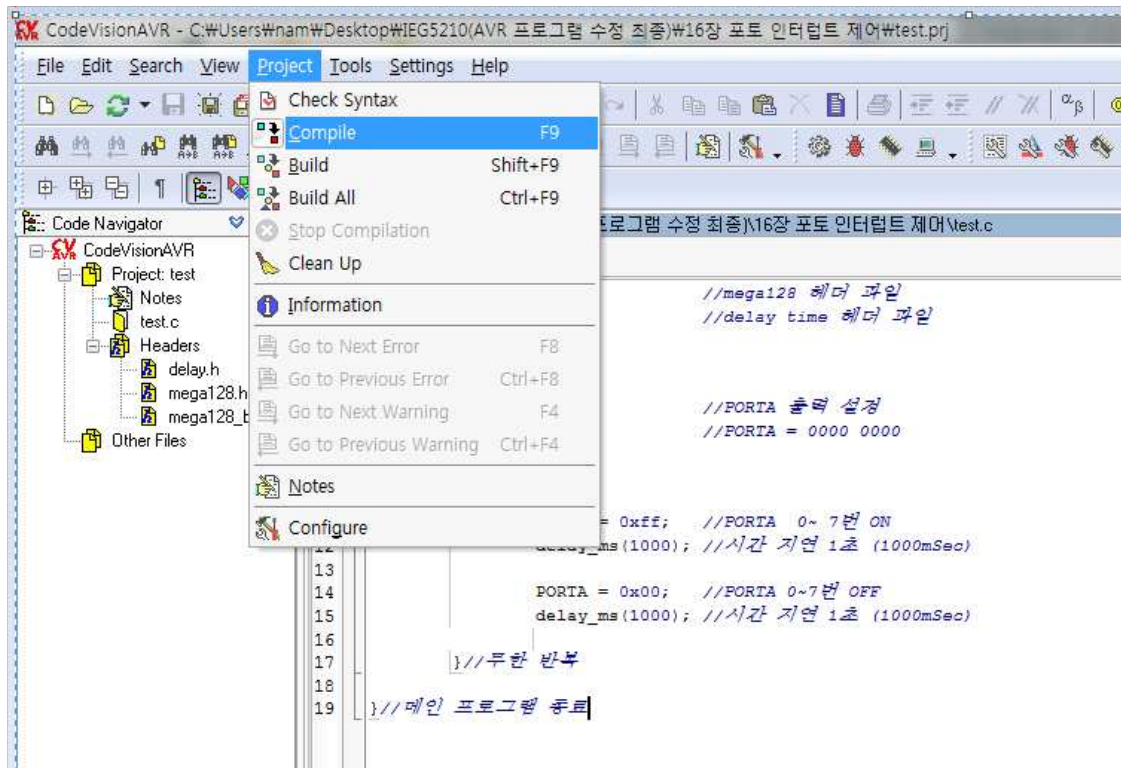
## ■ Program Source

```
void init_intterrupt(void)
{
    // External Interrupt(s) initialization
    // INT0: On
    // INT0 Mode:
    // INT1: Off
    EIMSK = 0x01;           // 개별 인터럽트 허용
    EICRA = 0x20;           // 하강엣지에서 인터럽트 발생
    EIFR = 0x01;

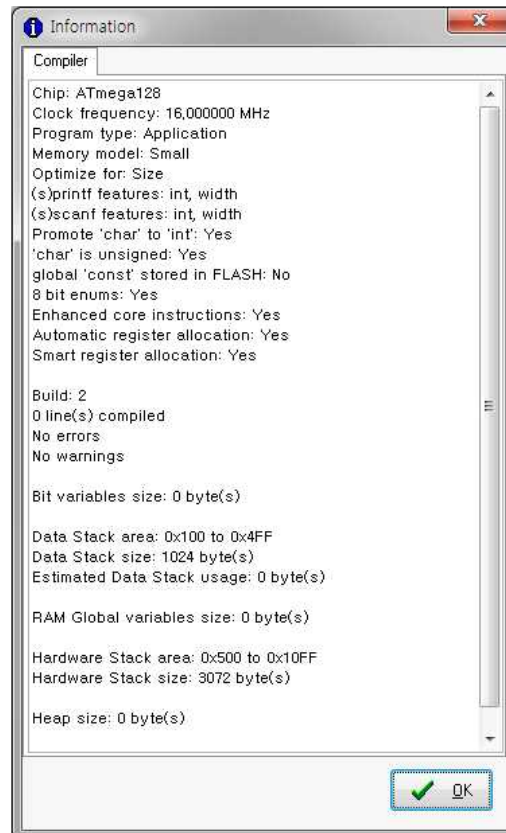
    // Global enable interrupts
    #asm("sei")
}
```

## [2] CodeVision 프로그램 컴파일

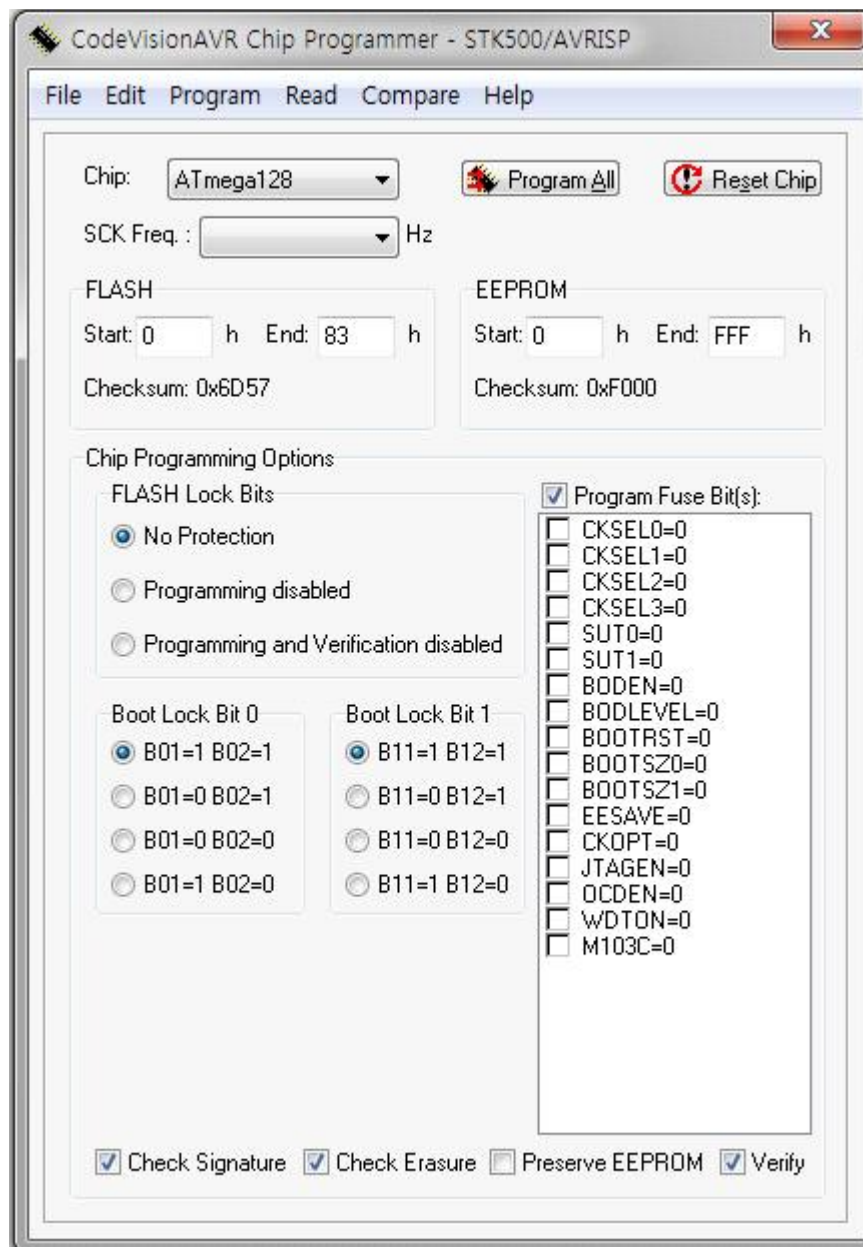
- CodeVision 프로그램에서 Project 메뉴를 선택하여 Compile을 실행 하도록 한다.



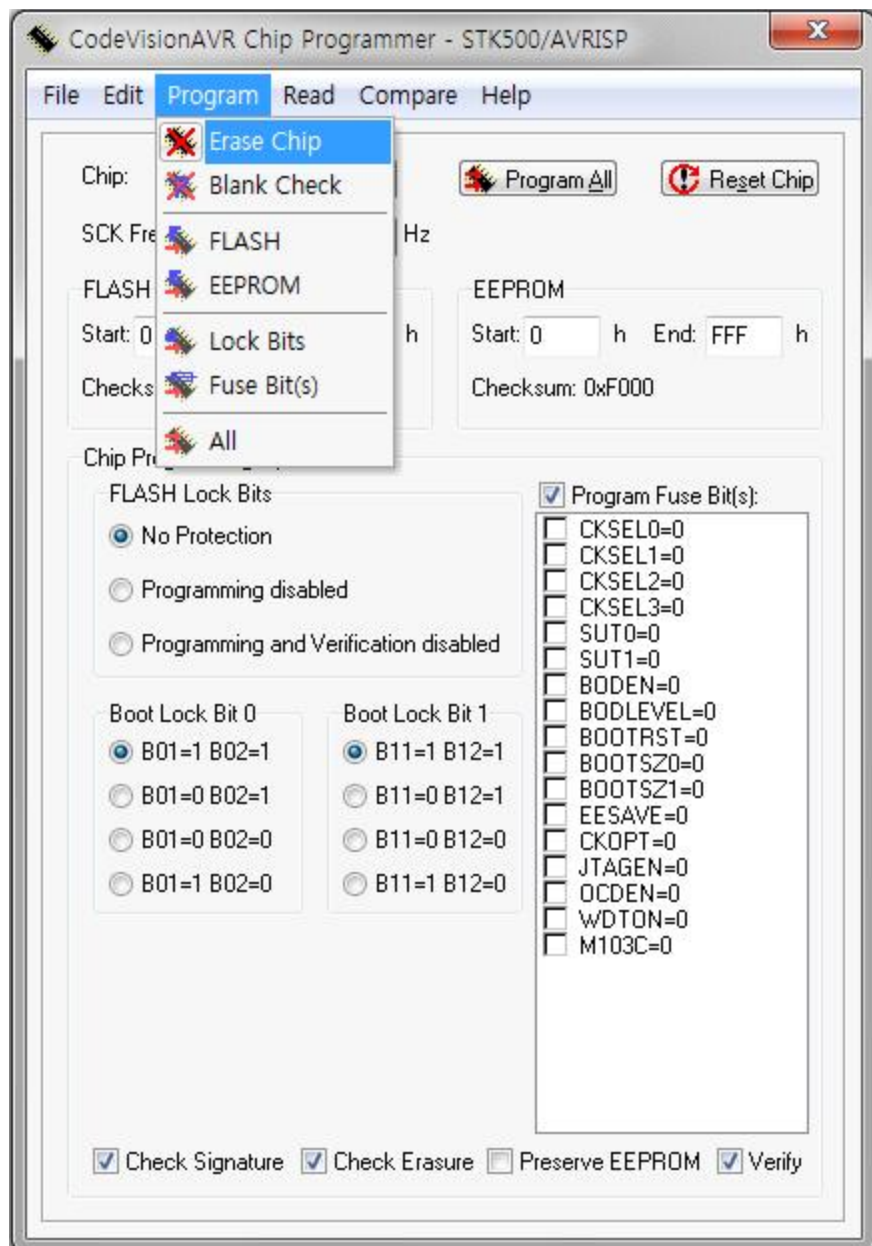
- 프로그램에서 에러가 없으면 아래와 같이 컴파일 정보가 출력 된다.



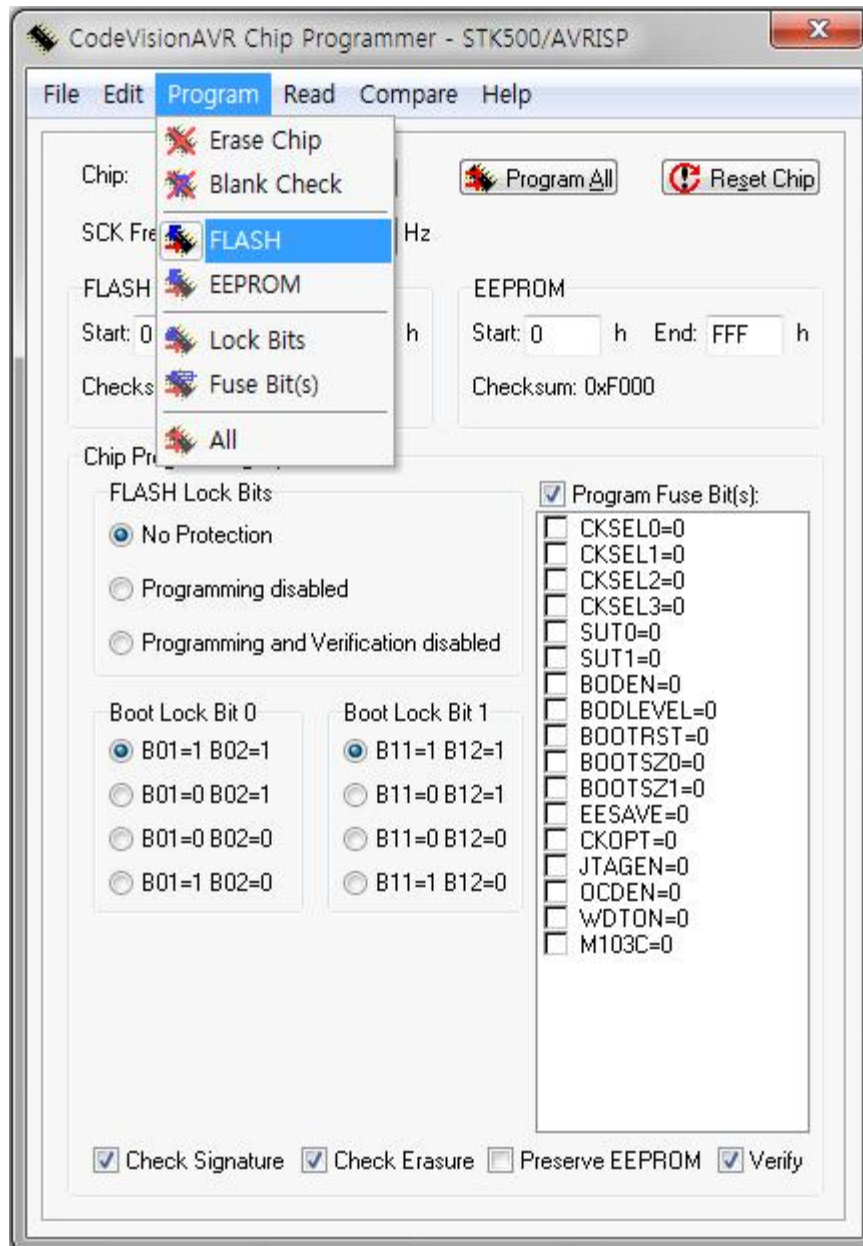
- Tools → Chip Programmer 클릭(단축키 : Shift+F4)



- Erase chip 클릭(chip 내에 프로그램이 남아 있을수 있으므로 프로그램 다운로드 전에 반드시 Erase Chip을 권장하도록 한다. )

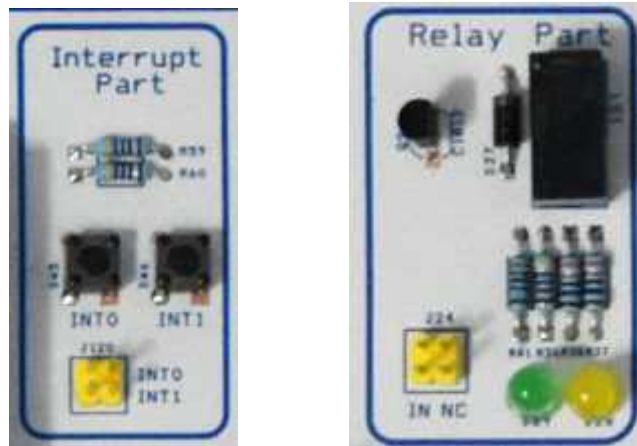


- Chip Flash 클릭하여 프로그램을 다운로드 하도록 한다.



### [3] 동작 확인

- SW를 눌렀을 때 인터럽트가 발생하여 Relay에 신호가 ON/OFF 가 되는 동작을 하게 된다.



### [4] 실습 과제

4-1. 과제명 : Interrupt를 이용하여 SW가 ON 되었을 때 LED 출력, OFF 되었을 때 LED를 OFF 하도록 한다.

4-2. 과제명 : Interrupt를 이용하여 SW0이 ON 되었을 때만 Segment에 숫자 증가를 하고 SW1이 ON이 되면 Segment에 숫자가 감소하는 카운터 디스플레이를 만들도록 한다.

## SECTION

# 12

## 시리얼 통신 실습

### 12-1. 학습 목표

- 시리얼 통신 구조와 동작 방법
- 시리얼 통신을 이용한 PC와 데이터 송수신

### 12-2. 기초 이론

#### [1] 시리얼 통신이란?

시리얼(Serial)은 직렬이란 뜻으로, 시리얼 통신은 직렬통신을 말합니다. 반대되는 개념으로는 병렬(Parallel)통신이 존재하며, 두 가지 모두 다른 기기와 데이터를 주고 받기 위한 통신 방법에 해당 된다.

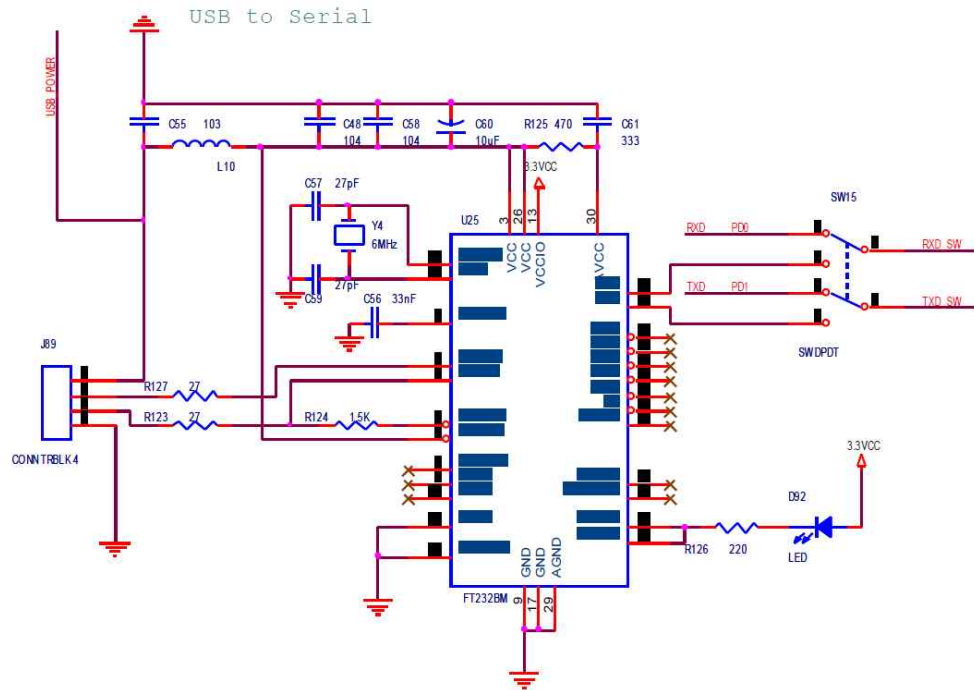
직렬 통신은 데이터를 1개씩 보내는 방법으로, 속도는 느리지만 거리가 길다는 장점이 있으며, 병렬 통신은 데이터를 여러개 씩 보내 속도는 빠르지만 거리가 짧다는 단점이 있다. 시리얼 통신을 하기 위해서는 수신선(RX)과 발신선(TX)이 필요하며 아두이노 우노 보드는 디지털 0번핀(RX)과 1번핀(TX)을 통해 시리얼 통신이 가능하다.

디지털 0번핀과 1번핀은 컴퓨터와 연결되는 USB 단자와도 연결 되어 있습니다. 즉, USB 케이블을 통해 아두이노와 컴퓨터를 연결하면 서로간의 시리얼 통신이 가능하다. USB 단자를 통해 아두이노와 컴퓨터가 연결되어 있는데 아두이노의 디지털 1번핀과 0번 핀도 다른 기기와 연결되어 있다면 중복 되기 때문에 통신이 꼬여 오류가 발생하기도 한다.



## 12-3. 시리얼 통신 구성

### [1] 시리얼 통신 회로도



### [2] 케이블 결선도

장비와 같이 제공되는 USB ISP A/B 케이블을 이용하여 ATmega128 보드와 아래와 같이 연결한다.



### [3] 장비 결선도

아래와 같이 ATmega128 IO 포트와 Serial 를 연결 하도록 한다.

HW 장비	PC
USB	USB

## 12-4. 시리얼 통신 제어

### [1] 프로그램 순서

1-1. 시리얼 통신을 위한 프로그램 작성

#### ■ Program Source

```
#include <mega128.h>
#include <stdio.h>      // Standard Input/Output functions
#include <delay.h>

void main(void)
{
    char buffer;

    UCSRA=0x00;
    UCSRB=0x18;    // 송수신 인에이블
    UCSR0C=0x06;    // 8bit Data통신
    UBRR0H=0x00;
    UBRR0L=0x08;    // 115200 bps

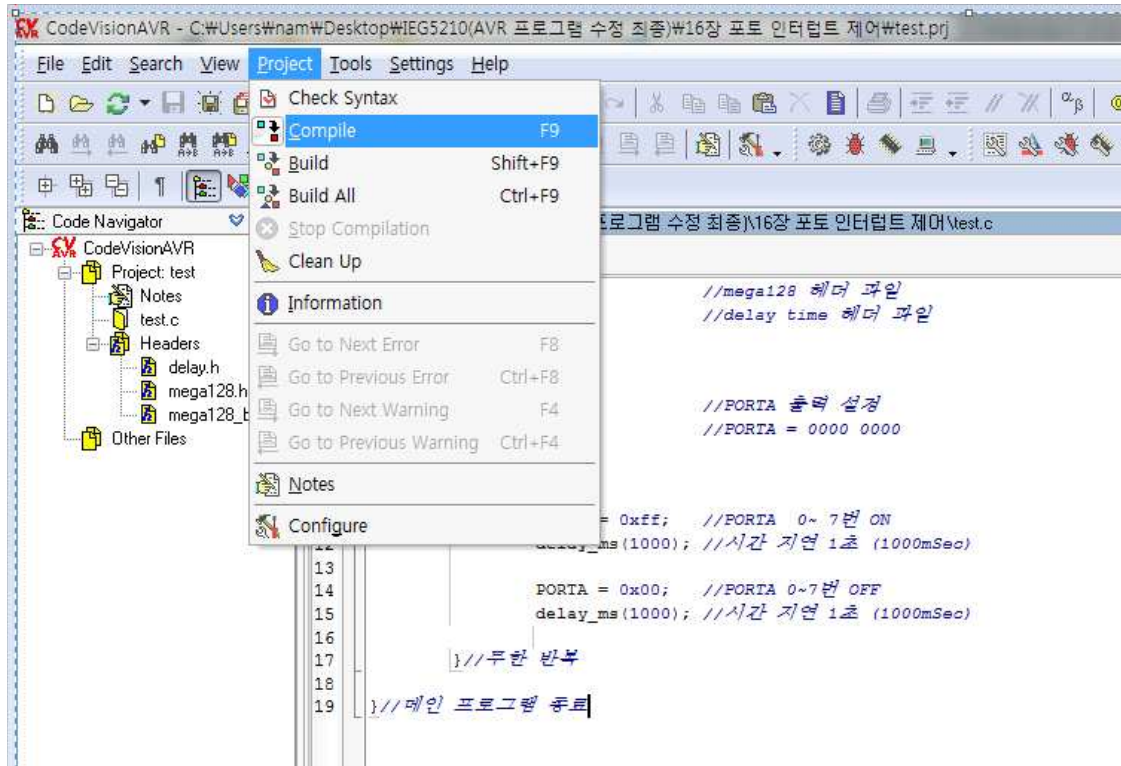
    /*
    UBRR / 16(UBRR+1) = fosc
    UBRR = fosc / (16xBAUD )-1
    7.6805... = 16MHz / (9600x16) -1
    반올림 8 → 16진수변환 → 0x08
    */

    while (1)
    {
        printf("Input Key Data:");           //입력 메시지 출력
        buffer = getchar();                   //문자 입력을 받아 변수에 저장
        printf(" ECO Key Data -> %c\r\n",buffer); //변수에 저장 된 문자를
        출력 한다.

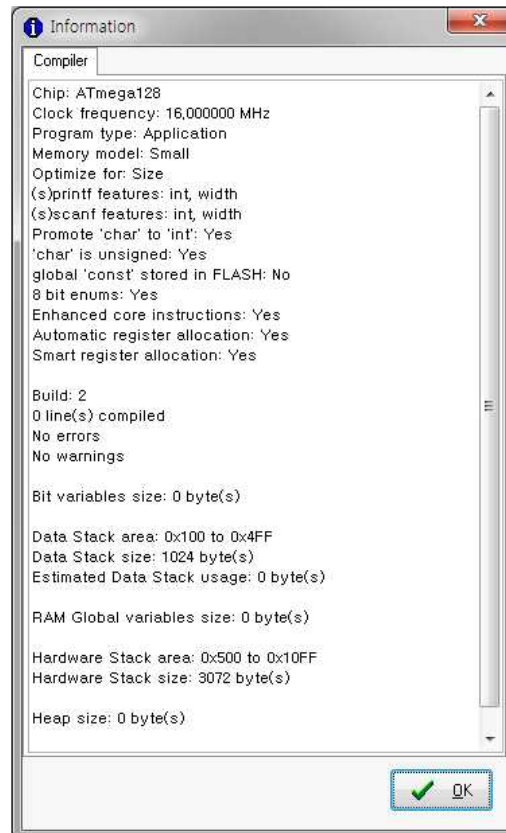
        }//while program end
    }//main program end
```

## [2] CodeVision 프로그램 컴파일

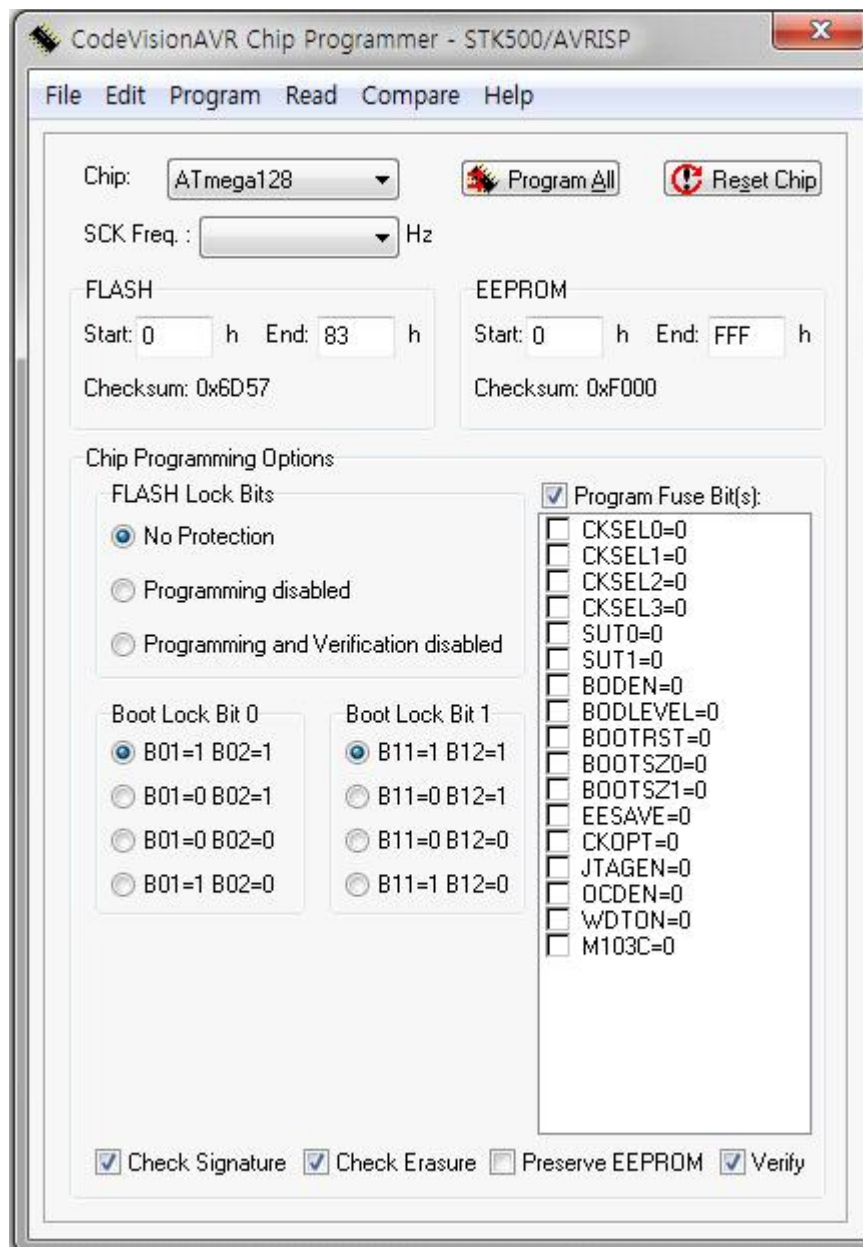
- CodeVision 프로그램에서 Project 메뉴를 선택하여 Compile을 실행 하도록 한다.



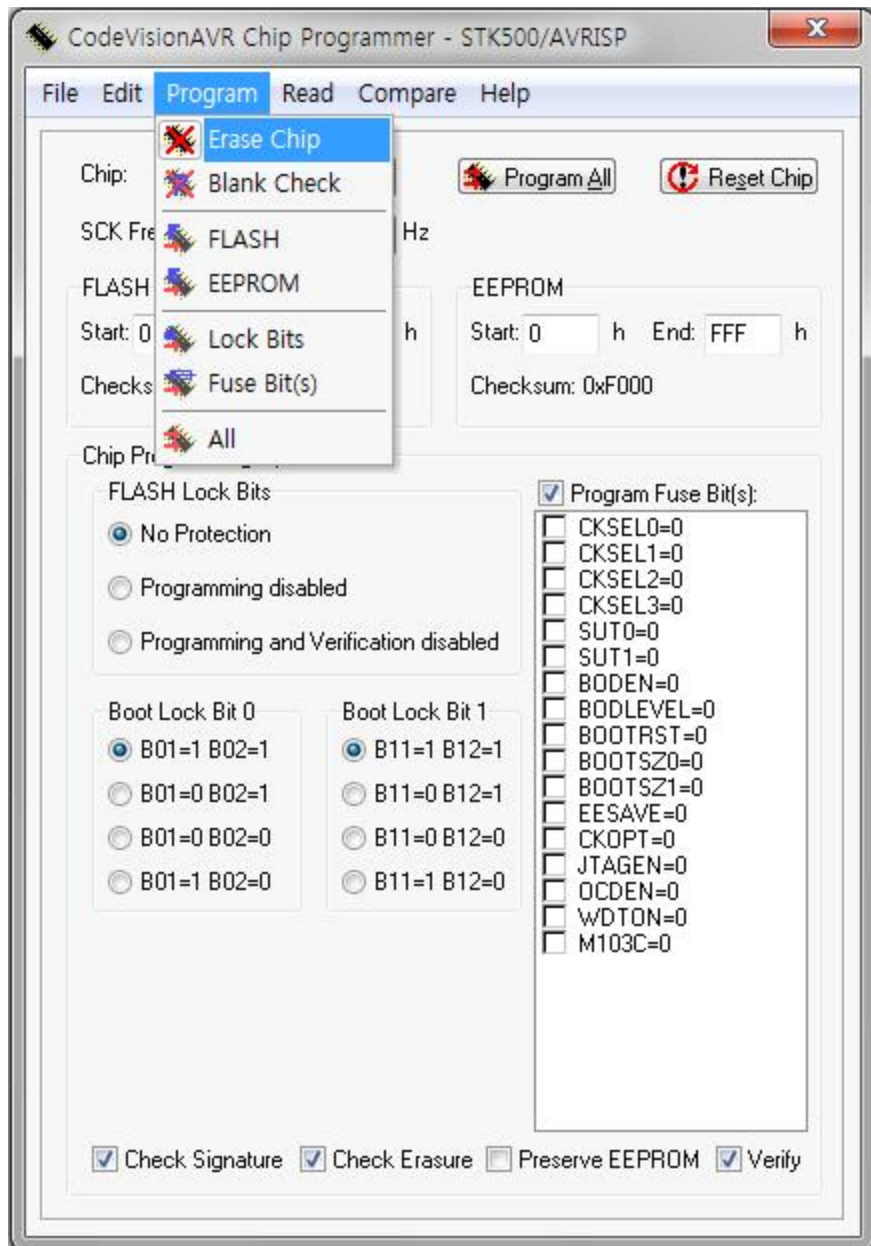
- 프로그램에서 에러가 없으면 아래와 같이 컴파일 정보가 출력 된다.



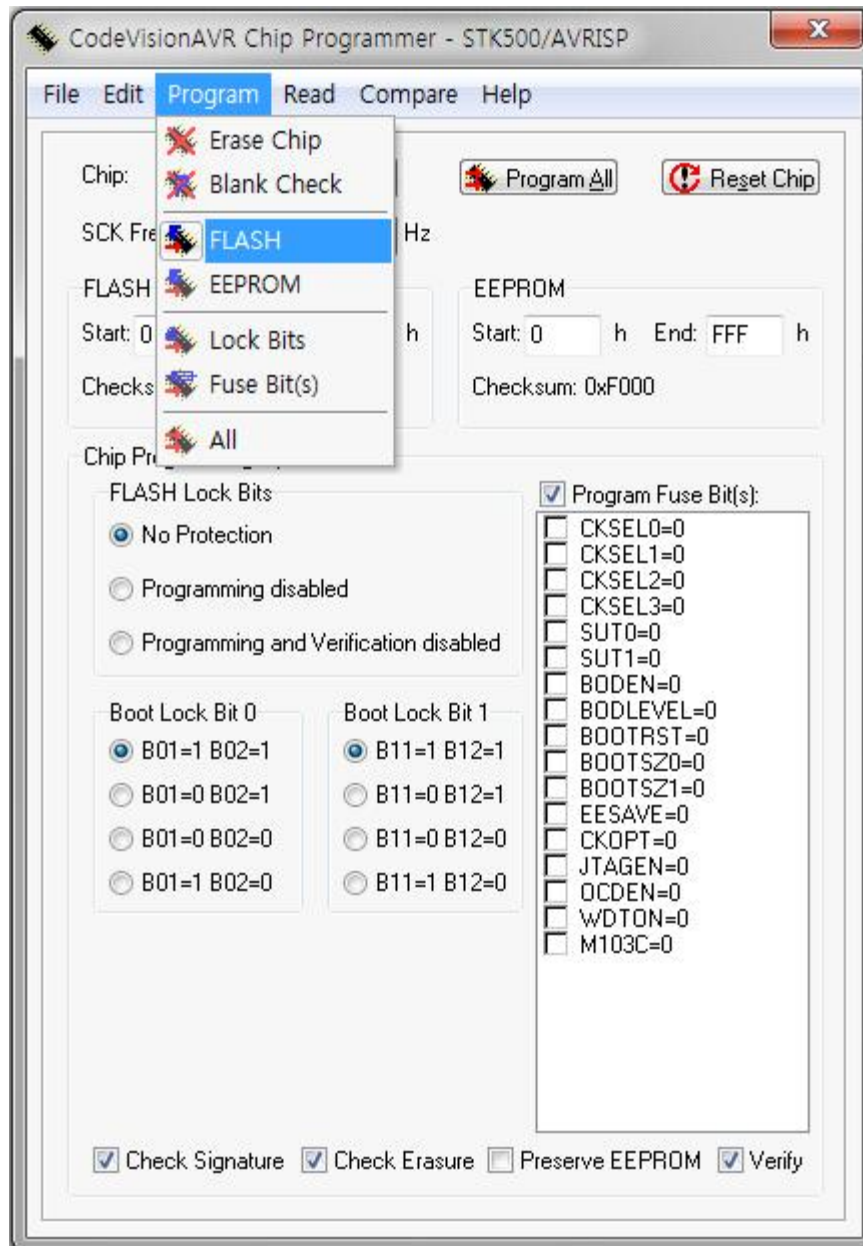
- Tools → Chip Programmer 클릭(단축키 : Shift+F4)



- Erase chip 클릭(chip 내에 프로그램이 남아 있을수 있으므로 프로그램 다운로드 전에 반드시 Erase Chip을 권장하도록 한다. )



- Chip Flash 클릭하여 프로그램을 다운로드 하도록 한다.

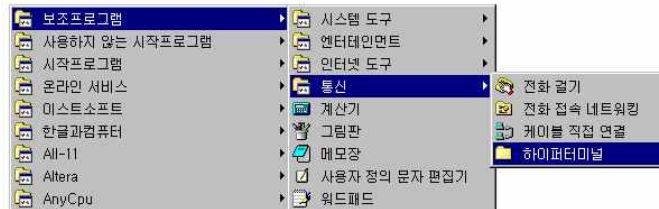




### [3] 시리얼 통신 프로그램 가동

컴퓨터의 「하이퍼 터미널」을 실행하여 컴퓨터 키보드 자판을 클릭하면 ATmega128에서 반환하는 프로그램을 작성한다.

① 하이퍼터미널을 실행한다.



하이퍼 터미널 실행

연결에 사용되는 이름을 입력하고, [확인] 버튼을 누른다.

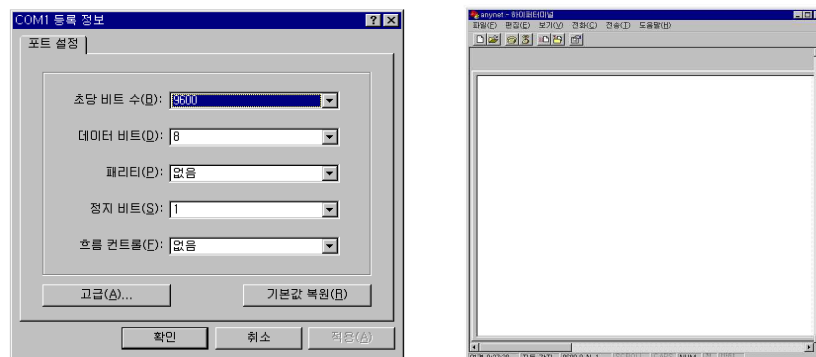
연결할 serial port를 선택하면 된다.



연결 설정

Port의 등록정보를 다음과 같이 변경한다.

내 용	설 정 값
초당 비트 수	9600
데이터 비트	8
패리티	없음
정지 비트	1
흐름 컨트롤	없음



#### [4] 동작 확인

- 시리얼 모니터에 입력 받은 데이터가 출력되는 것을 확인 할 수 있다.



#### [5] 실습 과제

4-1. 과제명 : Interrupt를 이용하여 SW가 ON 되었을 때 LED 출력, OFF 되었을 때 LED를 OFF 하도록 한다.

4-2. 과제명 : Interrupt를 이용하여 SW0이 ON 되었을 때만 Segment에 숫자 증가를 하고 SW1이 ON이 되면 Segment에 숫자가 감소하는 카운터 디스플레이를 만들도록 한다.

MCU & Arduino Control Trainer PAMS - AEB V9.1 (ATmega 128)

<http://www.spgkorea.co.kr>

代 031) 8018 5040