WAT-AVR128

(ATMEGA128 Module)

WhiteAT

HOMEPAGE: http://WhiteAT.com

E-MAIL: whiteat@whiteat.com

1부 - ATMEGA128A

2부 - ATMEGA128A 준비

3부 - ATMEGA128A 연습

1부

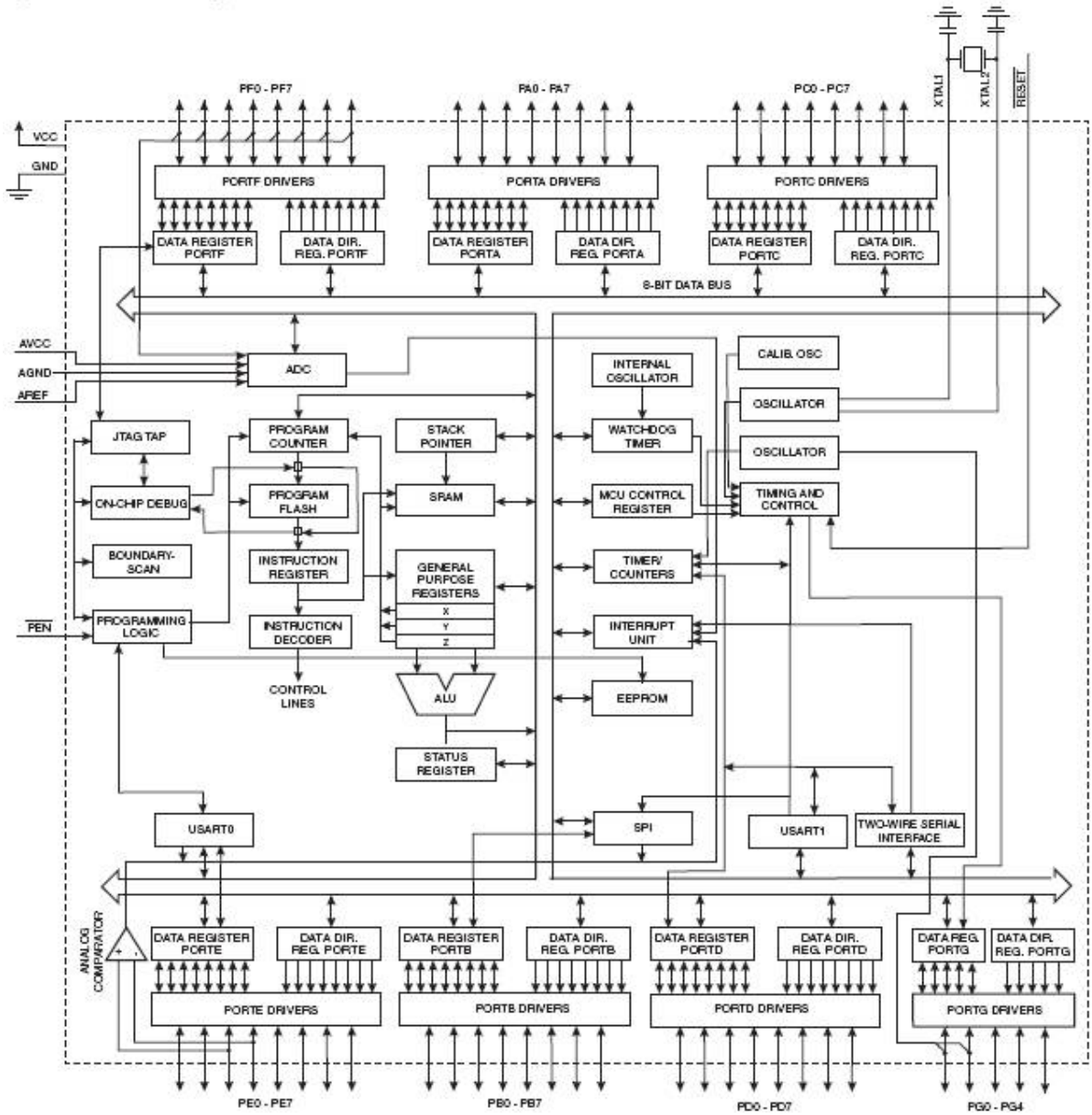
ATMEGA128A

1. ATMEGA128A

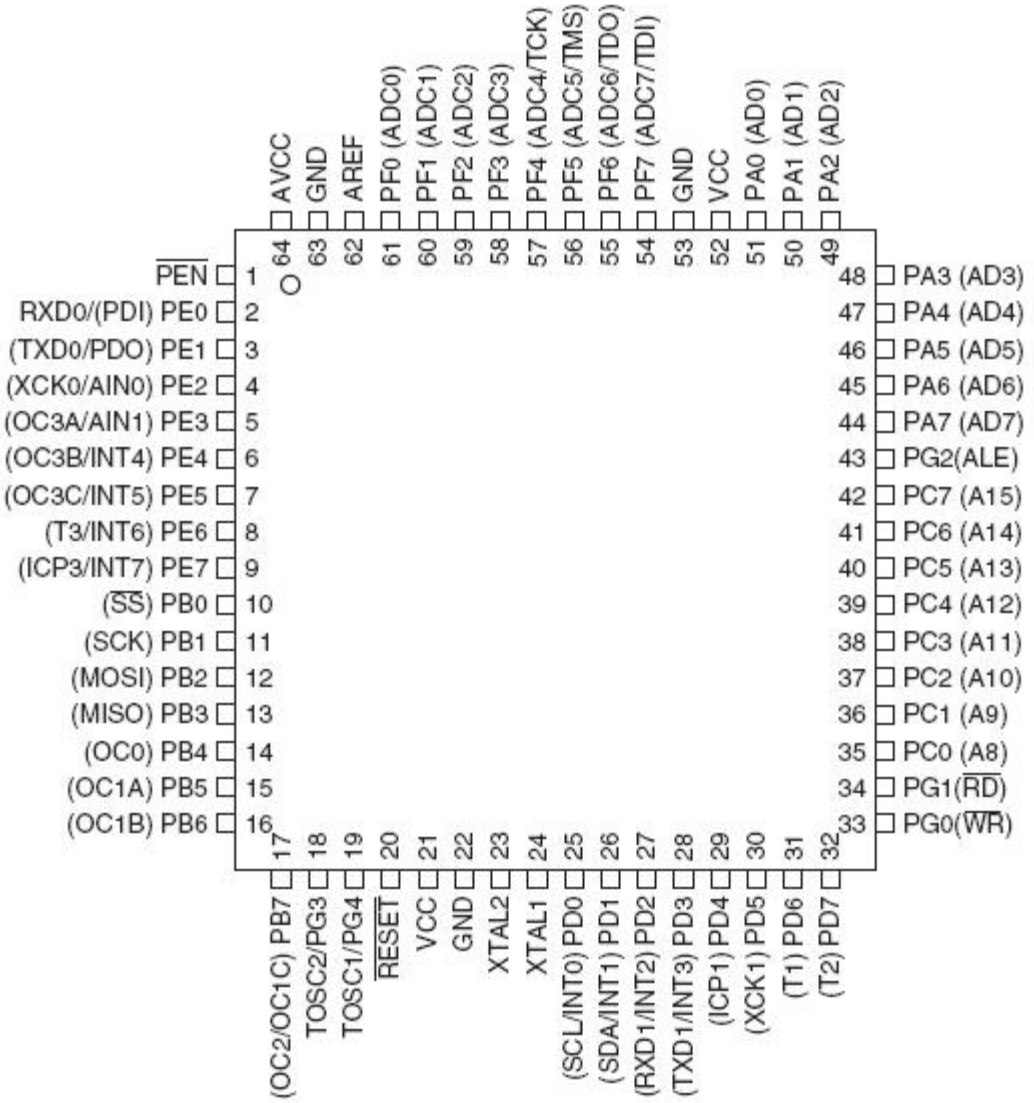
1.1. 특징

- 고성능, 저전력, Atmel AVR 8 비트 마이크로컨트롤러
- 향상된 RISC 아키텍처
  - 133 개의 강력한 명령 - 대부분 싱글사이클 수행
  - 32 개의 범용 레지스트리 + 패리패럴 제어 레지스트리
  - 16Mhz 까지 가능
  - 2 사이클 곱셈기
- 메모리
  - 128Kbytes 프로그램 가능한 Flash 프로그램 메모리
  - 4Kbytes EEPROM
  - 4Kbytes 내부 SRAM
  - 외부 64Kbytes 메모리 확장 가능
  - 프로그램 보안 락 제공
  - SPI 인터페이스
- JTAG 인터페이스
- 기능
  - 2개의 8비트 타이머/카운터
  - 2개의 16비트 타이머/카운터
  - 2개의 8비트 PWM
  - 6개의 2 ~16 비트 변경 가능한 PWM
  - 출력 비교기
  - 8 채널 10비트 아날로그디지털 컨버터
  - 2 채널 USARTS
  - Master/Slave SPI 시리얼 인터페이스
  - 와치도그 타이머
- 53 개의 I/O Lines
- 2.7 ~ 5.5V, 16MHz

1.2. 블록도


PF0 - PF7
PA0 - PA7
PC0 - PC7
XTAL1
XTAL2
RESET
VCC
GND
PORTF DRIVERS
PORTA DRIVERS
PORTC DRIVERS
DATA REGISTER PORTF
DATA DIR. REG. PORTF
DATA REGISTER PORTA
DATA DIR. REG. PORTA
DATA REGISTER PORTC
DATA DIR. REG. PORTC
8-BIT DATA BUS
AVCC
AGND
AREF
ADC
INTERNAL OSCILLATOR
CALIB. OSC
OSCILLATOR
OSCILLATOR
JTAG TAP
PROGRAM COUNTER
STACK POINTER
WATCHDOG TIMER
ON-CHIP DEBUG
PROGRAM FLASH
SRAM
MCU CONTROL REGISTER
TIMING AND CONTROL
BOUNDARY-SCAN
INSTRUCTION REGISTER
GENERAL PURPOSE REGISTERS
X
Y
Z
TIMER/ COUNTERS
PEN
PROGRAMMING LOGIC
INSTRUCTION DECODER
INTERRUPT UNIT
CONTROL LINES
ALU
EEPROM
STATUS REGISTER
USART0
SPI
USART1
TWO-WIRE SERIAL INTERFACE
ANALOG COMPARATOR
DATA REGISTER PORTE
DATA DIR. REG. PORTE
DATA REGISTER PORTB
DATA DIR. REG. PORTB
DATA REGISTER PORTD
DATA DIR. REG. PORTD
DATA REG. PORTG
DATA DIR. REG. PORTG
PORTE DRIVERS
PORTB DRIVERS
PORTD DRIVERS
PORTG DRIVERS
PE0 - PE7
PB0 - PB7
PD0 - PD7
PG0 - PG4

1.3. 핀 정의



Note: The Pinout figure applies to both TQFP and MLF packages. The bottom pad under the QFN/MLF package should be soldered to ground.

- VCC : 전원용
- GND : 그라운드
- Port A (PA7:PA0)
  - 일반 I/O
  - 외부 메모리 주소/데이터 (bit7 ~ bit0)
- Port B (PB7:PB0)
  - 일반 I/O
  - 특수 I/O
- Port C (PC7:PC0)
  - 일반 I/O
  - 외부 메모리 주소/데이터 (bit15 ~ bit8)
- Port D (PD7:PD0)
  - 일반 I/O
  - 특수 I/O
- Port E (PE7:PE0)
  - 일반 I/O
  - 특수 I/O
- Port F (PF7:PF0)
  - 일반 I/O
  - A/D 용
- Port G (PG4:PG0)
  - 특수 I/O
- /RESET : 리셋 핀
- XTAL1 : 크리스탈 연결용1 (또는 오실레이터 연결)
- XTAL2 : 크리스탈 연결용2
- AVCC : AD용 전원 핀
- AREF : 레퍼런스 전압
- PEN : AVR 프로그램 가능하게 설정 핀

1.4. 상태 레지스터 (SREG)

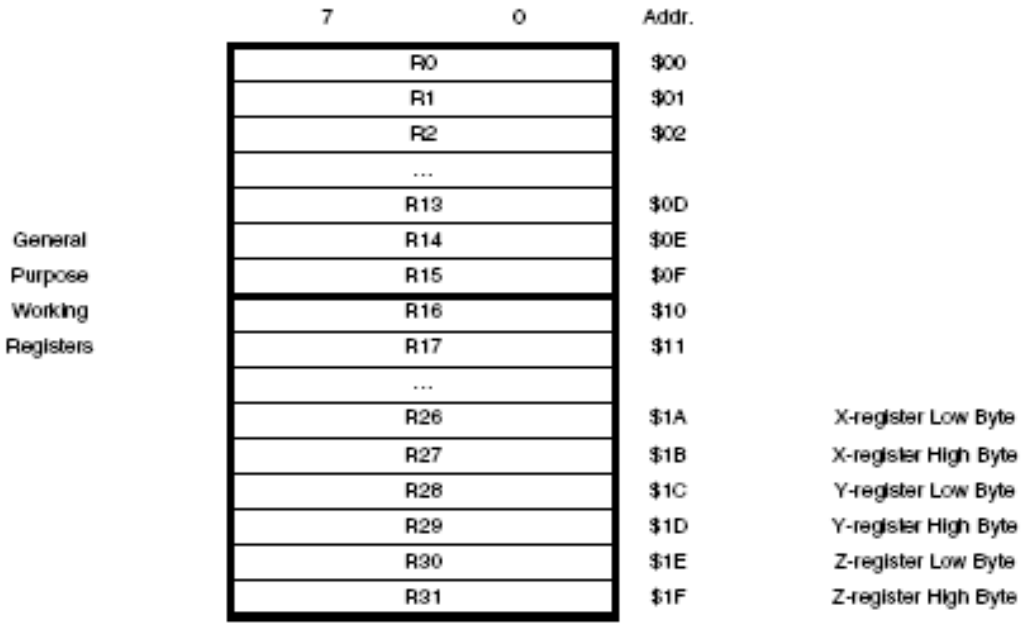
• 상태 레지스터는 연산의 결과에 대한 정보를 가지고 있습니다.

Status Register 주소 : 0x5F

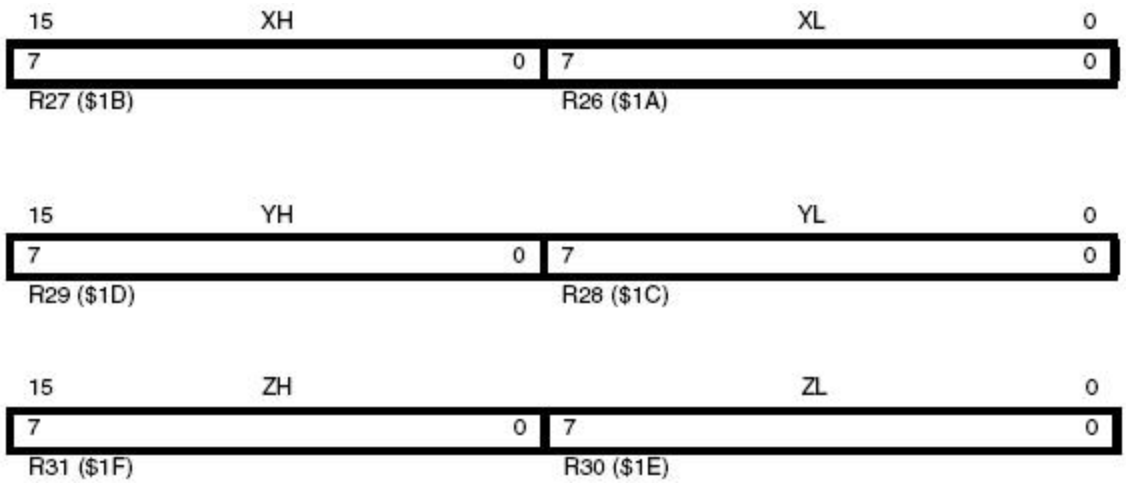
Bit	7	6	5	4	3	2	1	0
	I	T	H	S	V	N	Z	C
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

1.5. 범용 레지스터

• ATMEGA128A 에는 범용 레지스터가 32bytes 가 존재하며 RISC 의 연산을 위해 필요합니다. R0 ~ R31 까지의 이름을 가집니다.



R26,R27 을 X 레지스터, R28, R29 를 Y레지스터, R30, R31 을 Z 레지스터라고 하며 X,Y,Z 레지스터는 16비트로 액세스 할 때 사용됩니다.



1.6. 스택 포인터 (Stack Pointer)

• 스택은 데이터를 임시로 저장할 때 매우 중요하게 사용됩니다. 주로 함수 호출에서 사용되며 함수 호출시 현재 상태를 스택에 저장하고 리턴 될 때 저장된 값으로 복구하는데 사용됩니다.

Status Register 주소 : 0x5D ~ 0x5E

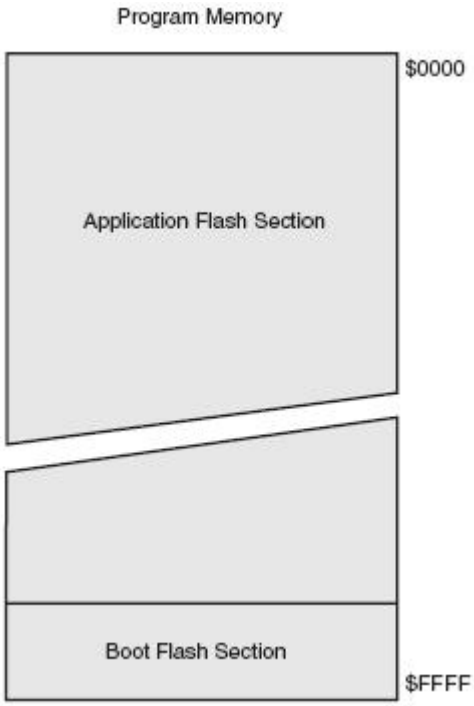
Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

1.7. RAMPZ - RAM Page Z Select Register

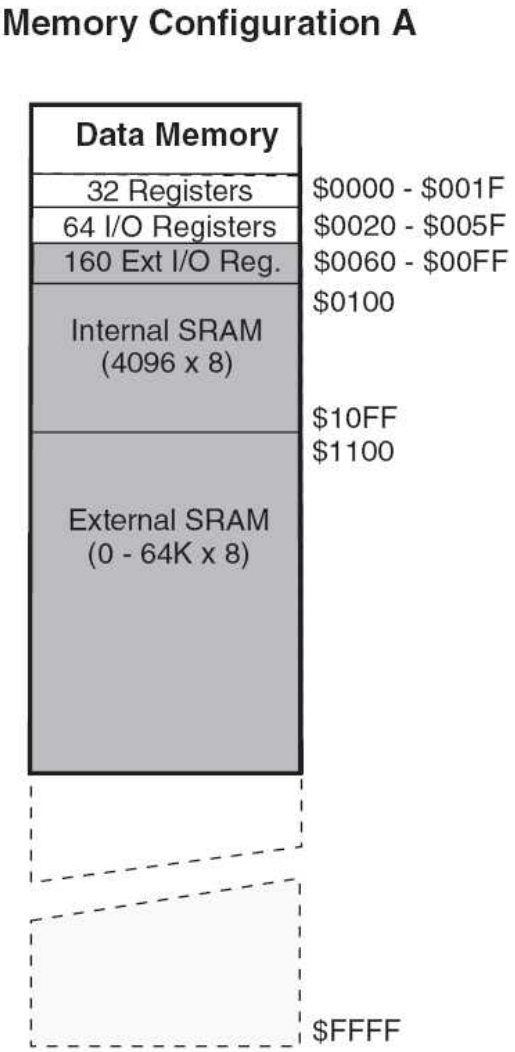
• Z 포인터를 다룰 때 0x0000 ~ 0x7FFF (lower 64Kbytes) 를 액세스 할 것인지, 0x8000 ~ 0xFFFF (higher 64Kbytes) 를 액세스 할 것인지를 결정합니다.

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	-	RAMPZ0	RAMPZ
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

1.8. 프로그램 메모리

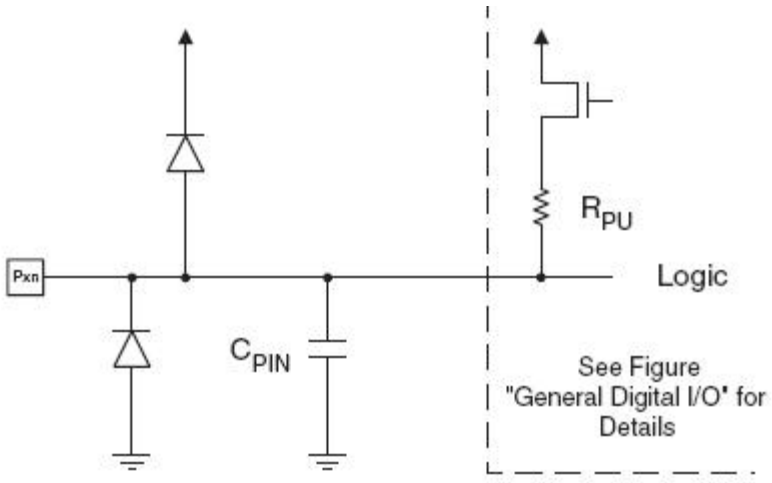

Program Memory
$0000
Application Flash Section
Boot Flash Section
$FFFF

1.9. 데이터 메모리


Memory Configuration A
Data Memory
32 Registers
64 I/O Registers
160 Ext I/O Reg.
Internal SRAM
(4096 x 8)
External SRAM
(0 - 64K x 8)
$0000 - $001F
$0020 - $005F
$0060 - $00FF
$0100
$10FF
$1100
$FFFF

2. General Digital I/O

• I/O 는 대부분 내부 풀업저항을 옵션으로 가지고 있으며 필요에 따라 Tri-state 도 만들 수 있습니다.



< I/O Pin Equivalent Schematic >

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

< Port Pin Configurations >

2.1. PORT A

• PORTA 는 외부 메모리 주소/데이터 (bit7 ~bit 0), 일반 디지털 입출력으로 사용되며 여기서는 일반 디지털 입출력에 대해 설명하겠습니다.

Port Pin	Alternate Function
PA7	AD7 (External memory interface address and data bit 7)
PA6	AD6 (External memory interface address and data bit 6)
PA5	AD5 (External memory interface address and data bit 5)
PA4	AD4 (External memory interface address and data bit 4)
PA3	AD3 (External memory interface address and data bit 3)
PA2	AD2 (External memory interface address and data bit 2)
PA1	AD1 (External memory interface address and data bit 1)
PA0	AD0 (External memory interface address and data bit 0)

< 참고 : PORTA 의 다른 기능 >

• PORTA - Port A 데이터 레지스터

Port A 의 출력 값이 HIGH, LOW 인지 설정한다. (1 : HIGH, 0 : LOW)

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• DDRA - Port A 데이터 방향 레지스터

Port A 를 입/출력 방향을 결정합니다. (1 : 출력, 0 : 입력)

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• PINA - Port A 입력 데이터

Port A 를 입력으로 사용 시 입력 값이 들어옵니다.

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

• PORTA 비트0를 HIGH, 로 나머지는 LOW로 만드는 예제

```
DDRA = 0xFF;      // 포트 A 를 출력으로 설정
PORTA = 0x01      // 포트 A의 비트0만 HIGH로 만들기
```

• PORTA 비트7를 HIGH, 로 나머지는 LOW로 만드는 예제

```
DDRA = 0xFF;  // 포트 A 를 출력으로 설정
PORTA = 0x80 // 포트 A의 비트7만 HIGH로 만들기
```

• PORTA 를 입력으로 사용하여 데이터 읽는 예제

```
DDRA = 0x00;  // 포트 A 를 입력으로 설정
byte byteData = PINA;  // 포트 A의 입력값을 byteData 변수에 입력
```

• PORTA 비트7 ~ 4 를 입력으로 비트3 ~ 0 을 출력으로 설정하는 예제

```
DDRA = 0x0F;  // 포트 A 상위4비트를 입력, 하위4비트를 출력으로 설정
PORTA = 0xFF; // 하위 4비트을 HIGH로 출력, 상위 4비트는 내부 풀업 사용
byte byteData = PINA;  //포트 A의 입력값을 byteData 변수에 입력 (상위 4비트만 유효)
```

2.2. PORT B

• PORTB 는 특별한 기능(표 참조), 일반 디지털 입출력으로 사용되며 여기서는 일반 디지털 입출력에 대해 설명하겠습니다.

Port Pin	Alternate Functions
PB7	OC2/OC1C(1) (Output Compare and PWM Output for Timer/Counter2 or Output Compare and PWM Output C for Timer/Counter1)
PB6	OC1B (Output Compare and PWM Output B for Timer/Counter1)
PB5	OC1A (Output Compare and PWM Output A for Timer/Counter1)
PB4	OC0 (Output Compare and PWM Output for Timer/Counter0)
PB3	MISO (SPI Bus Master Input/Slave Output)
PB2	MOSI (SPI Bus Master Output/Slave Input)
PB1	SCK (SPI Bus Serial Clock)
PB0	$\overline{SS}$ (SPI Slave Select input)

Note: 1. OC1C not applicable in ATmega103 compatibility mode.

< 참고 : PORTB 의 다른 기능 >

- PORTB - Port B 데이터 레지스터
- DDRB - Port B 데이터 방향 레지스터
- PINB - Port B 입력 데이터

  PORTA 의 레지스터 설정 방법과 동일합니다.

2.3. PORT C

• PORT C 는 외부 메모리 주소/데이터 (bit15 ~bit 8), 일반 디지털 입출력으로 사용되며 여기서는 일반 디지털 입출력에 대해 설명하겠습니다.

Port Pin	Alternate Function
PC7	A15
PC6	A14
PC5	A13
PC4	A12
PC3	A11
PC2	A10
PC1	A9
PC0	A8

< 참고 : PORTC 의 다른 기능 >

- PORTC - Port C 데이터 레지스터
- DDRC - Port C 데이터 방향 레지스터
- PINC - Port C 입력 데이터

  PORTA 의 레지스터 설정 방법과 동일합니다.

2.4. PORT D

• PORTD 는 특별한 기능(표 참조), 일반 디지털 입출력으로 사용되며 여기서는 일반 디지털 입출력에 대해 설명하겠습니다.

Port Pin	Alternate Function
PD7	T2 (Timer/Counter2 Clock Input)
PD6	T1 (Timer/Counter1 Clock Input)
PD5	XCK1(1) (USART1 External Clock Input/Output)
PD4	ICP1 (Timer/Counter1 Input Capture Pin)
PD3	$\overline{INT3}$/TXD1(1) (External Interrupt3 Input or UART1 Transmit Pin)
PD2	$\overline{INT2}$/RXD1(1) (External Interrupt2 Input or UART1 Receive Pin)
PD1	$\overline{INT1}$/SDA(1) (External Interrupt1 Input or TWI Serial DAta)
PD0	$\overline{INT0}$/SCL(1) (External Interrupt0 Input or TWI Serial CLock)

< 참고 : PORTD 의 다른 기능 >

- PORTD - Port D 데이터 레지스터
- DDRD - Port D 데이터 방향 레지스터
- PIND - Port D 입력 데이터

  PORTA 의 레지스터 설정 방법과 동일합니다.

2.5. PORT E

• PORTE 는 특별한 기능(표 참조), 일반 디지털 입출력으로 사용되며 여기서는 일반 디지털 입출력에 대해 설명하겠습니다.

Port Pin	Alternate Function
PE7	INT7/ICP3[1] (External Interrupt 7 Input or Timer/Counter3 Input Capture Pin)
PE6	INT6/ T3[1] (External Interrupt 6 Input or Timer/Counter3 Clock Input)
PE5	INT5/OC3C[1] (External Interrupt 5 Input or Output Compare and PWM Output C for Timer/Counter3)
PE4	INT4/OC3B[1] (External Interrupt4 Input or Output Compare and PWM Output B for Timer/Counter3)
PE3	AIN1/OC3A [1] (Analog Comparator Negative Input or Output Compare and PWM Output A for Timer/Counter3)
PE2	AIN0/XCK0[1] (Analog Comparator Positive Input or USART0 external clock input/output)
PE1	PDO/TXD0 (Programming Data Output or UART0 Transmit Pin)
PE0	PDI/RXD0 (Programming Data Input or UART0 Receive Pin)

< 참고 : PORTE 의 다른 기능 >

- PORTE - Port E 데이터 레지스터
- DDRE - Port E 데이터 방향 레지스터
- PINE - Port E 입력 데이터

  PORTA 의 레지스터 설정 방법과 동일합니다.

2.6. PORT F

• PORTF 는 Analog Digital Converter 기능(표 참조), 일반 디지털 입출력으로 사용되며 여기서는 일반 디지털 입출력에 대해 설명하겠습니다.

Port Pin	Alternate Function
PF7	ADC7/TDI (ADC input channel 7 or JTAG Test Data Input)
PF6	ADC6/TDO (ADC input channel 6 or JTAG Test Data Output)
PF5	ADC5/TMS (ADC input channel 5 or JTAG Test Mode Select)
PF4	ADC4/TCK (ADC input channel 4 or JTAG Test ClocK)
PF3	ADC3 (ADC input channel 3)
PF2	ADC2 (ADC input channel 2)
PF1	ADC1 (ADC input channel 1)
PF0	ADC0 (ADC input channel 0)

< 참고 : PORTF 의 다른 기능 >

- PORTF - Port F 데이터 레지스터
- DDRF - Port F 데이터 방향 레지스터
- PINF - Port F 입력 데이터

  PORTA 의 레지스터 설정 방법과 동일합니다

2.7. PORT G

• PORTG 는 특별한 기능(표 참조)으로 사용되며 일반 디지털 입출력으로 사용이 불가능 합니다.

Port Pin	Alternate Function
PG4	TOSC1 (RTC Oscillator Timer/Counter0)
PG3	TOSC2 (RTC Oscillator Timer/Counter0)
PG2	ALE (Address Latch Enable to external memory)
PG1	$\overline{RD}$ (Read strobe to external memory)
PG0	$\overline{WR}$ (Write strobe to external memory)

< 참고 : PORTG 의 다른 기능 >

• 일반 디지털 포트로 사용 불가하여 일반 디지털 I/O에 관련된 레지스터가 없습니다.

여기까지 ATMEGA128A 의 많이 사용되는 기본적인 기능을 알아보았습니다. 좀 더 자세히 또는 정확한 기능을 알려면 데이터시트를 봐야 합니다.

2부

ATMEGA128A
준비

1. 개별 환경

ATMEGA128 컨트롤러의 컴파일툴은 AVRStudio, Codevision, IAR 등이 있습니다. AVR Studio는 가장 많이 사용되는 컴파일툴입니다. Codevision 은 코드위저드가 있어 초기에 코드 생성할 때 편리합니다. IAR 은 주로 업체에서 사용되는 컴파일툴입니다.

1.1. 컴파일툴

AVRStudio 와 WinAVR 로 컴파일 할 수 있습니다. AVRStudio 는 코딩하는데 편리한 환경을 제공하고 WinAVR 은 컴파일러를 제공합니다.

- **프로그램 다운로드**

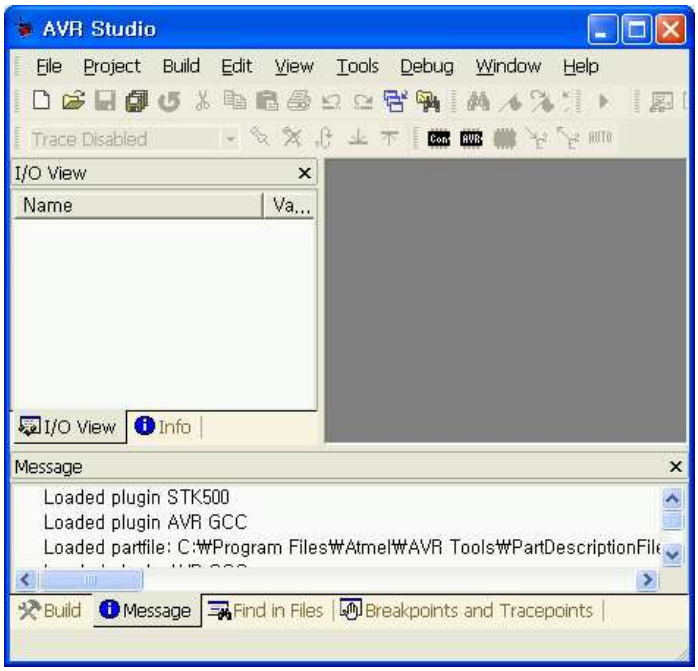
http://www.atmel.com 에 가입 후 AVRStudio 프로그램을 다운받을 수 있습니다.

http://winavr.sourceforge.net/ 에서 가입 없이 WinAVR을 다운받을 수 있습니다.

- **프로그램 설치**

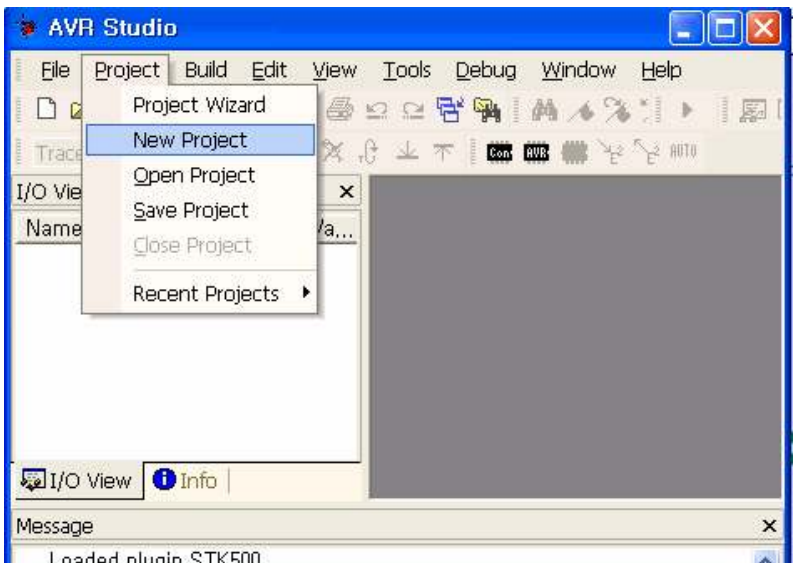
먼저 WinAVR을 설치한 후 AVRStudio를 설치합니다.

> AVRStudio 는 컴파일을 쉽게 도와주는 툴을 제공하고
> WinAVR 은 여러 헤더파일과 라이브러리와 컴파일러(AVR-gcc)를 제공합니다.
>
> WinAVR 설치후에 AVRStudio 를 설치해야 하며,
> **WinAVR은 항상 최신버전을 사용해야 합니다.**



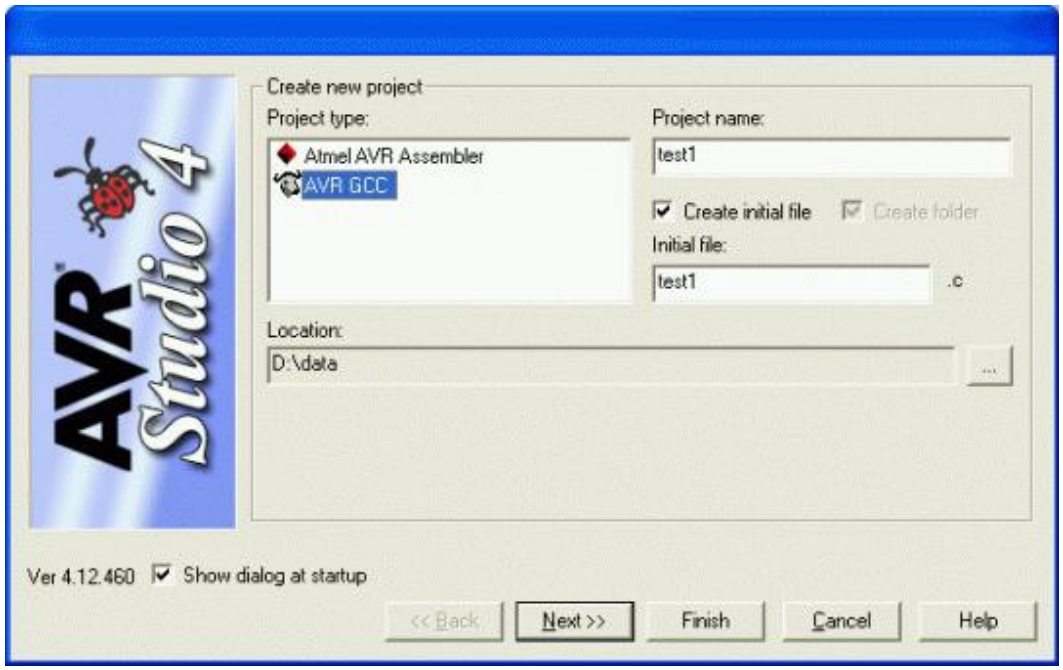
< AVRStudio 실행 화면 >

먼저 새로운 프로젝트를 생성한다.

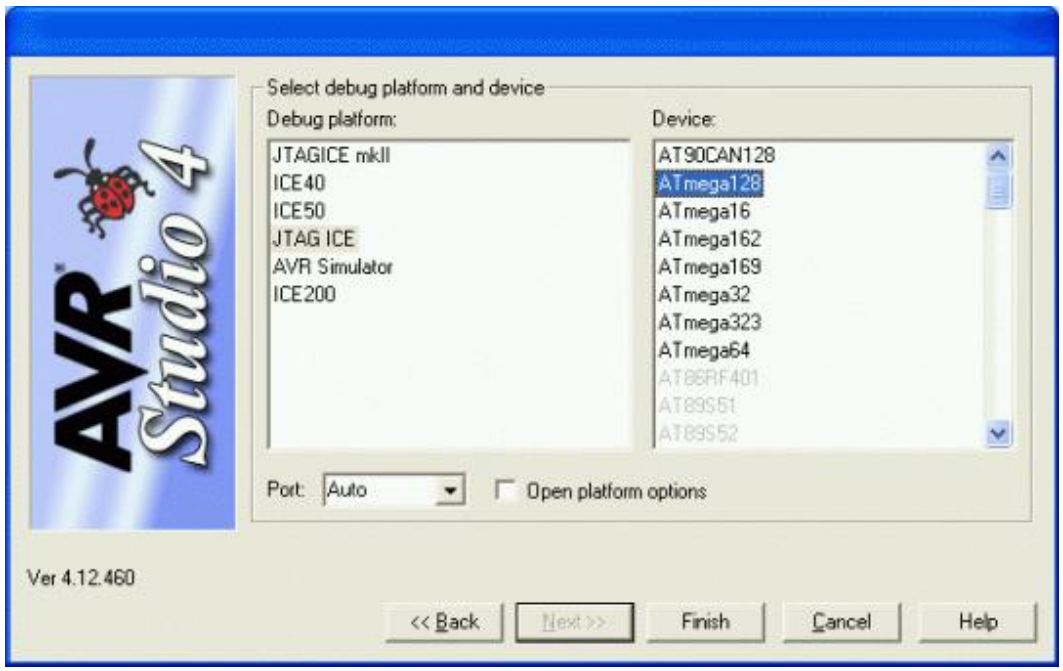


< 프로젝트 생성 >

다음으로 프로젝트명을 결정 한다



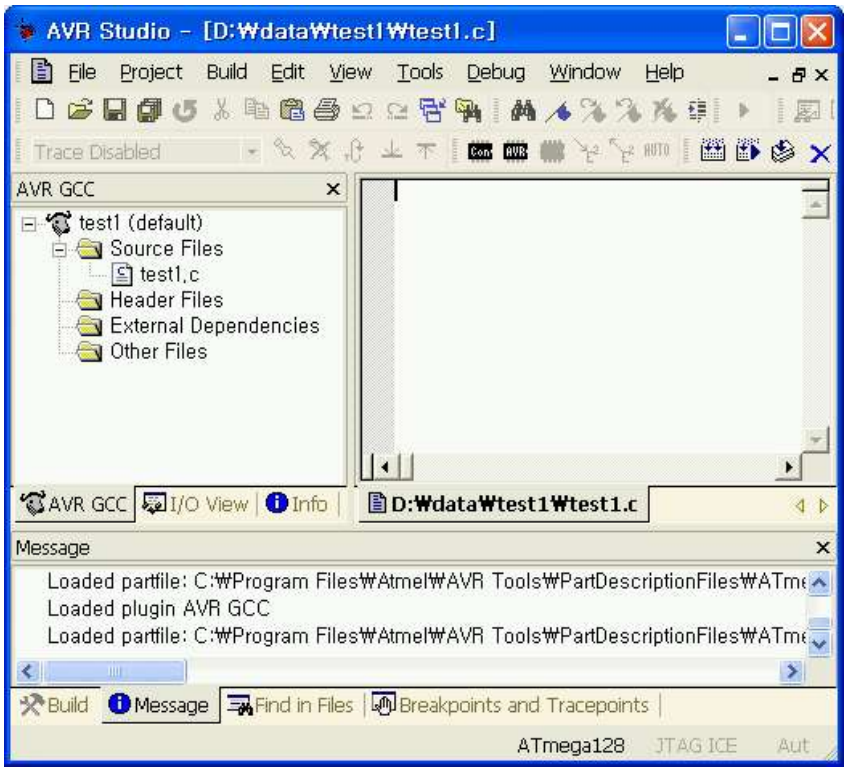
< 프로젝트명 결정 >



< DEVICE 결정 >

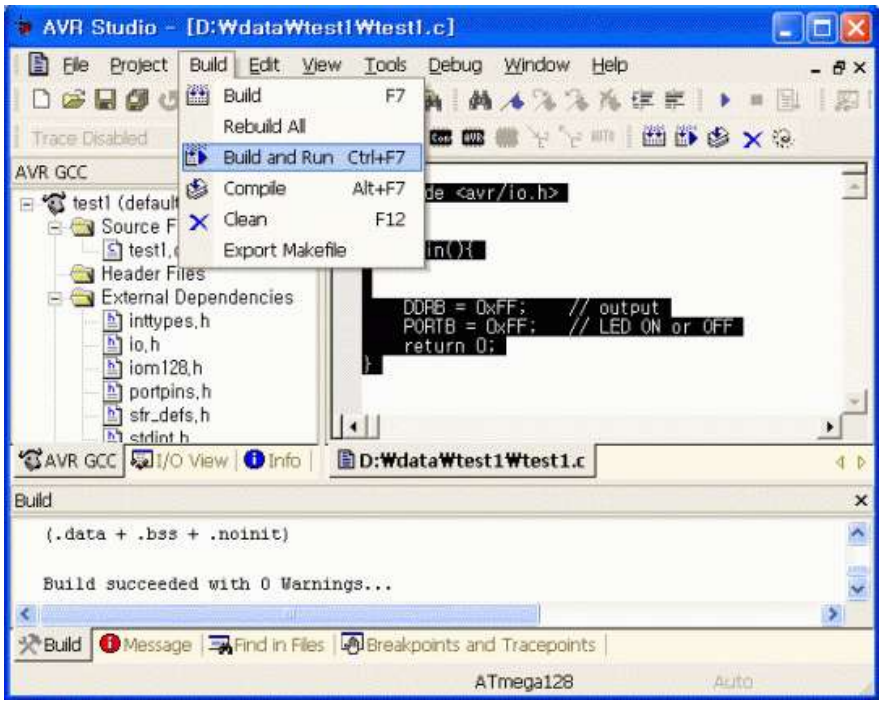
Finish를 눌러 종료하면, 이제 소스를 편집할 수 있는 창이 열린다.

이제 코딩하면 된다.



간단하게 소스를 편집하고,

```
#include <avr/io.h>
int main(){
        DDRB = 0xFF;        // output
        PORTB = 0x00      // LED ON
    return 0;
}
```



< 컴파일 하기 >

Build and Run 를 실행하여 test1.hex 파일이 생성되었는지 확인합니다.

라이팅 하는 방법은 다음 장에 있습니다.

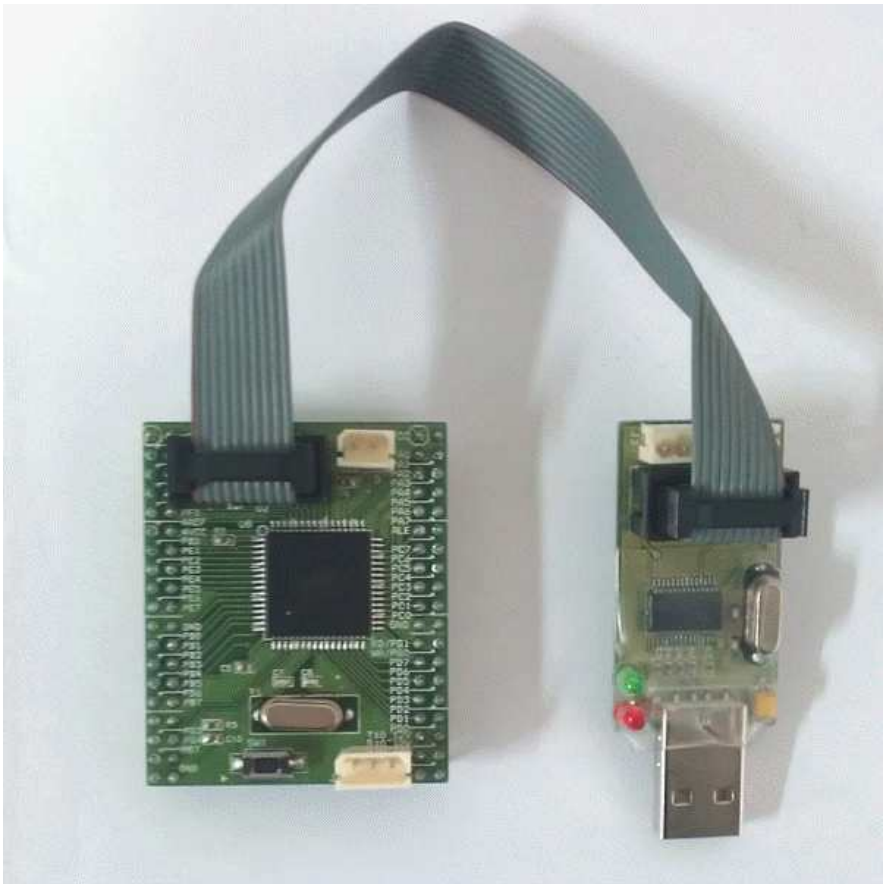
2. 프로그램 라이팅

컴파일하여 생성된 결과 파일(헥사파일)을 다음과 같은 방법으로 라이팅 할 수 있습니다.

- Ponyprog 프로그램과 프린터 포트를 이용
- AVRStudio 프로그램과 WAT-AVR ISP (USB 용 AVR ISP)를 이용

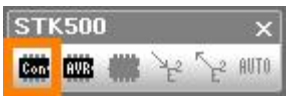
2.1. USB 용 AVR ISP

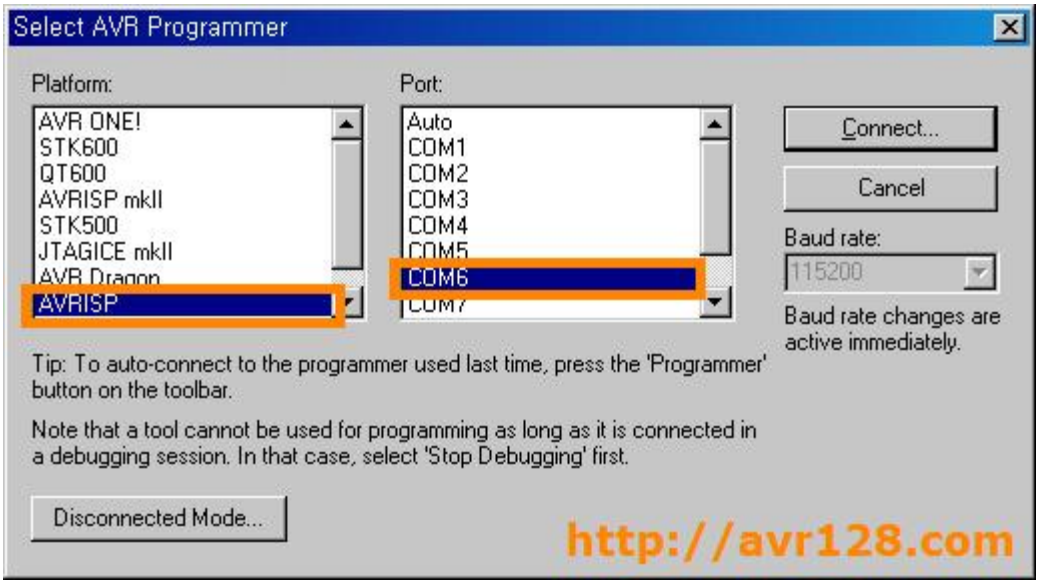
그림처럼 AVR 모듈과 USB ISP 케이블을 연결하고 USB ISP 케이블을 PC의 USB 포트에 연결합니다.



2.1.1. 연결

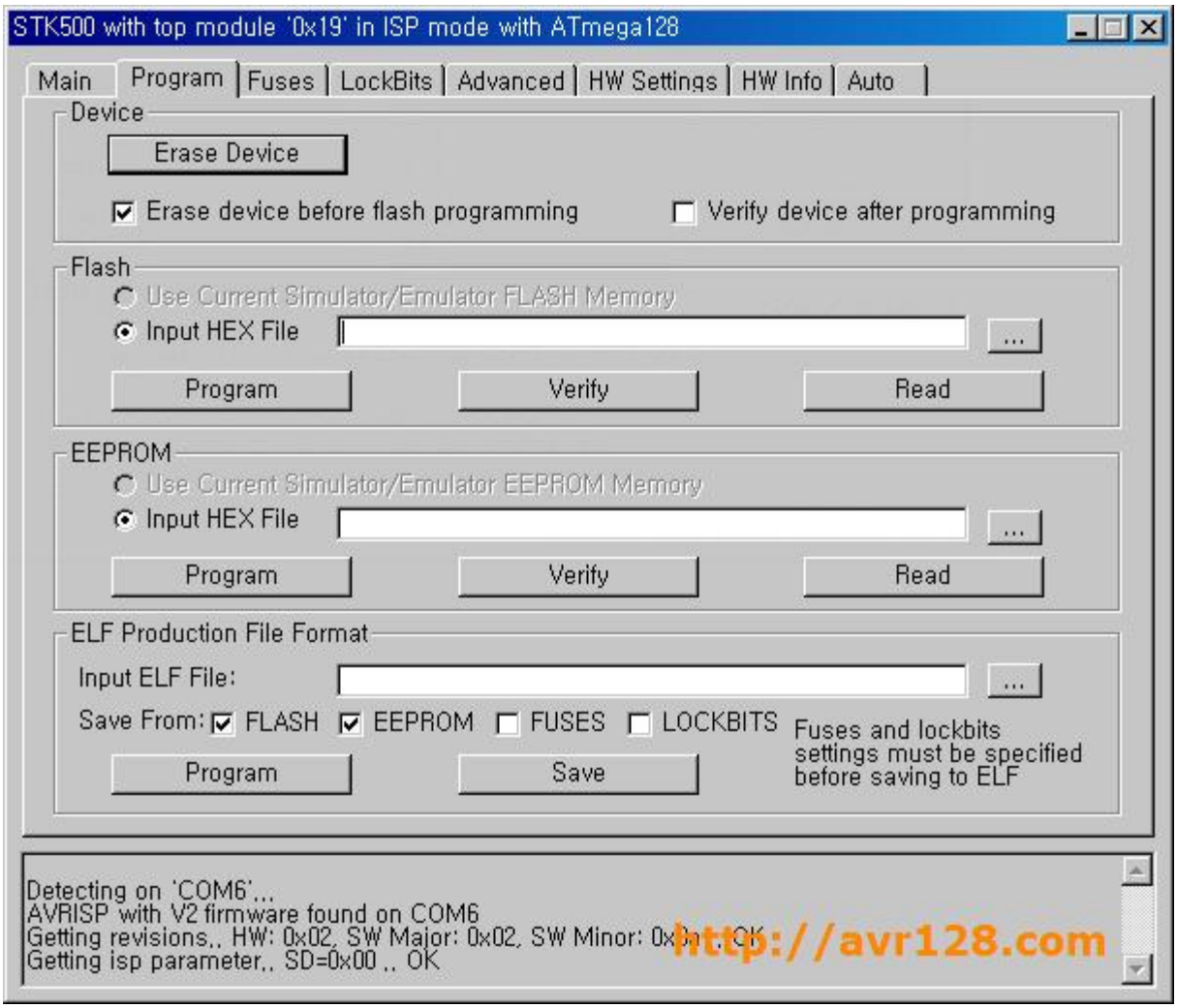
AVRStudio 의 STK500 도구바에서 'Display the Connect Dialog' 버튼을 클릭하면 ISP 연결 설정에 관한 다이어로그 박스가 나타납니다.





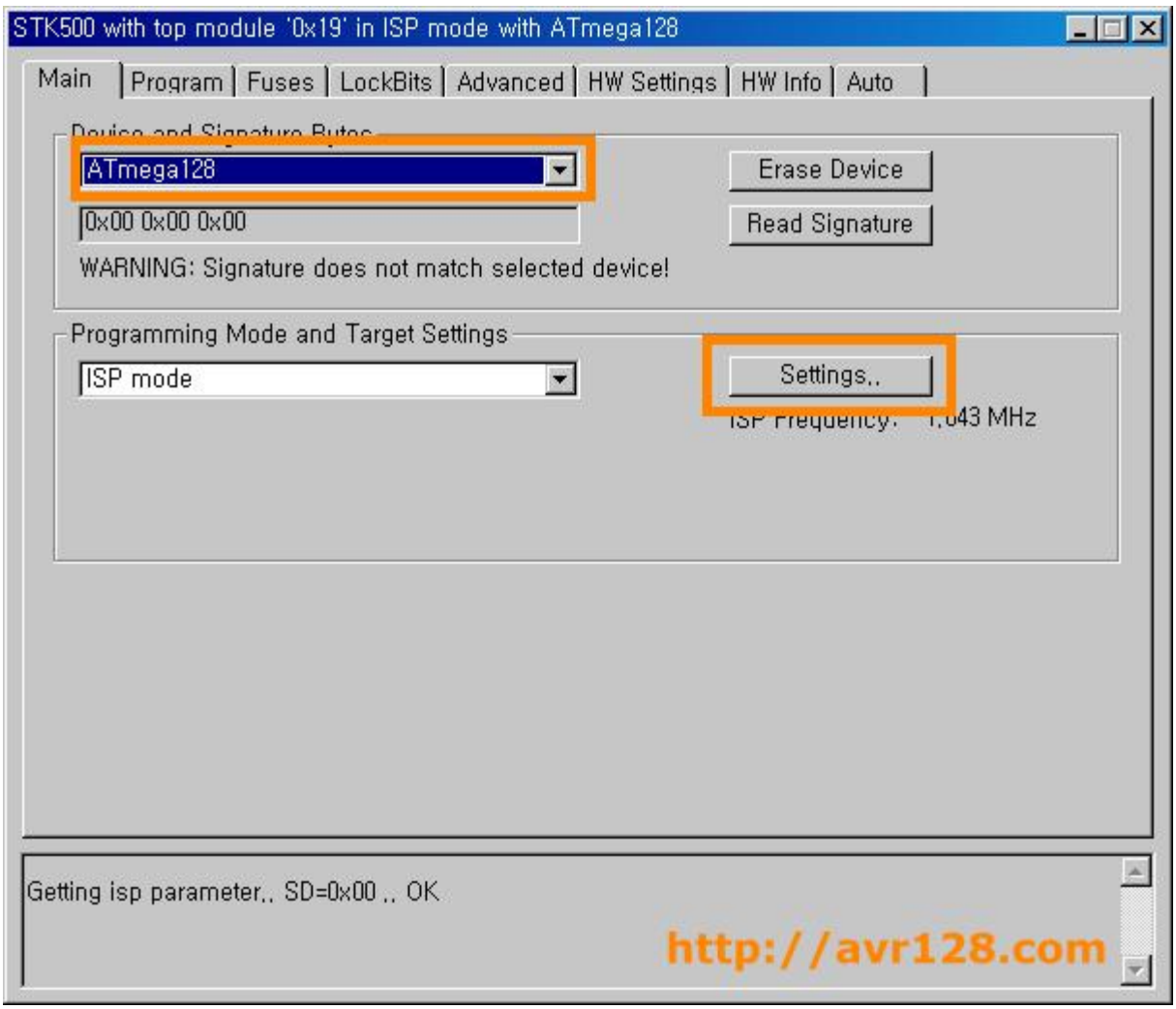
Platform에서 AVRISP 를 선택하고 Port에서 연결된 포트 번호를 선택한 후 [Connect]를 클릭합니다. 연결을 성공하면 프로그램 라이팅에 관련된 다이어로그박스가 뜨게 되고 연결이 실패하면 다시 선택하라고 같은 창이 뜹니다.

지금까지는 ISP 케이블에 AVR 모듈을 연결하지 않아도 상관없습니다.


STK500 with top module '0x19' in ISP mode with ATmega128
Main
Program
Fuses
LockBits
Advanced
HW Settings
HW Info
Auto
Device
Erase Device
Erase device before flash programming
Verify device after programming
Flash
Use Current Simulator/Emulator FLASH Memory
Input HEX File
Program
Verify
Read
EEPROM
Use Current Simulator/Emulator EEPROM Memory
Input HEX File
Program
Verify
Read
ELF Production File Format
Input ELF File:
Save From: FLASH EEPROM FUSES LOCKBITS
Fuses and lockbits settings must be specified before saving to ELF
Program
Save
Detecting on 'COM6'...
AVRISP with V2 firmware found on COM6
Getting revisions,, HW: 0x02, SW Major: 0x02, SW Minor: 0x0a ,, OK
Getting isp parameter,, SD=0x00 ,, OK
http://avr128.com

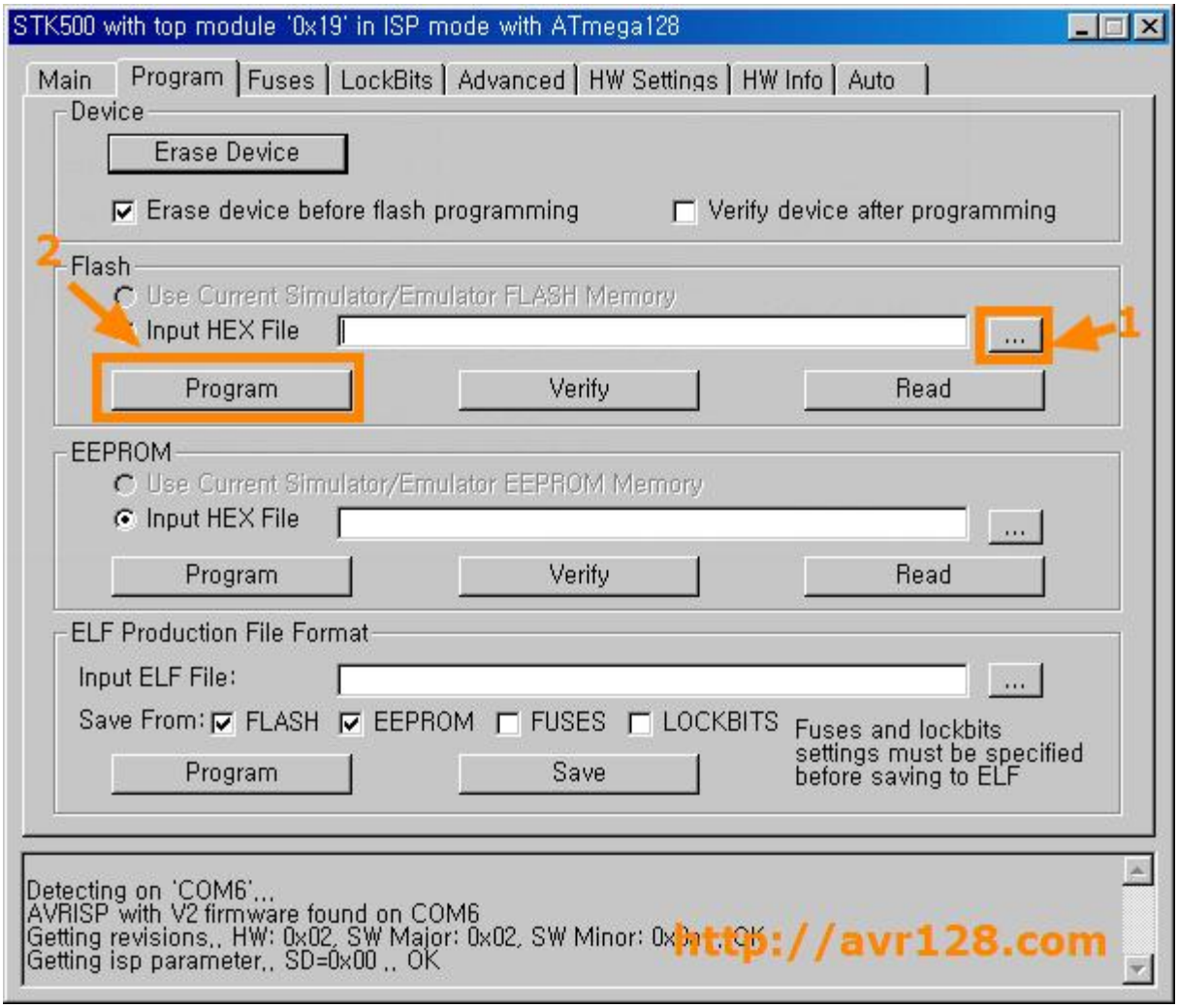
2.1.2. 디바이스 설정

[Device and Signature Bytes]에서 사용할 디바이스명을 선택하고 [Settings]에서 적당한 라이팅 속도를 결정합니다.



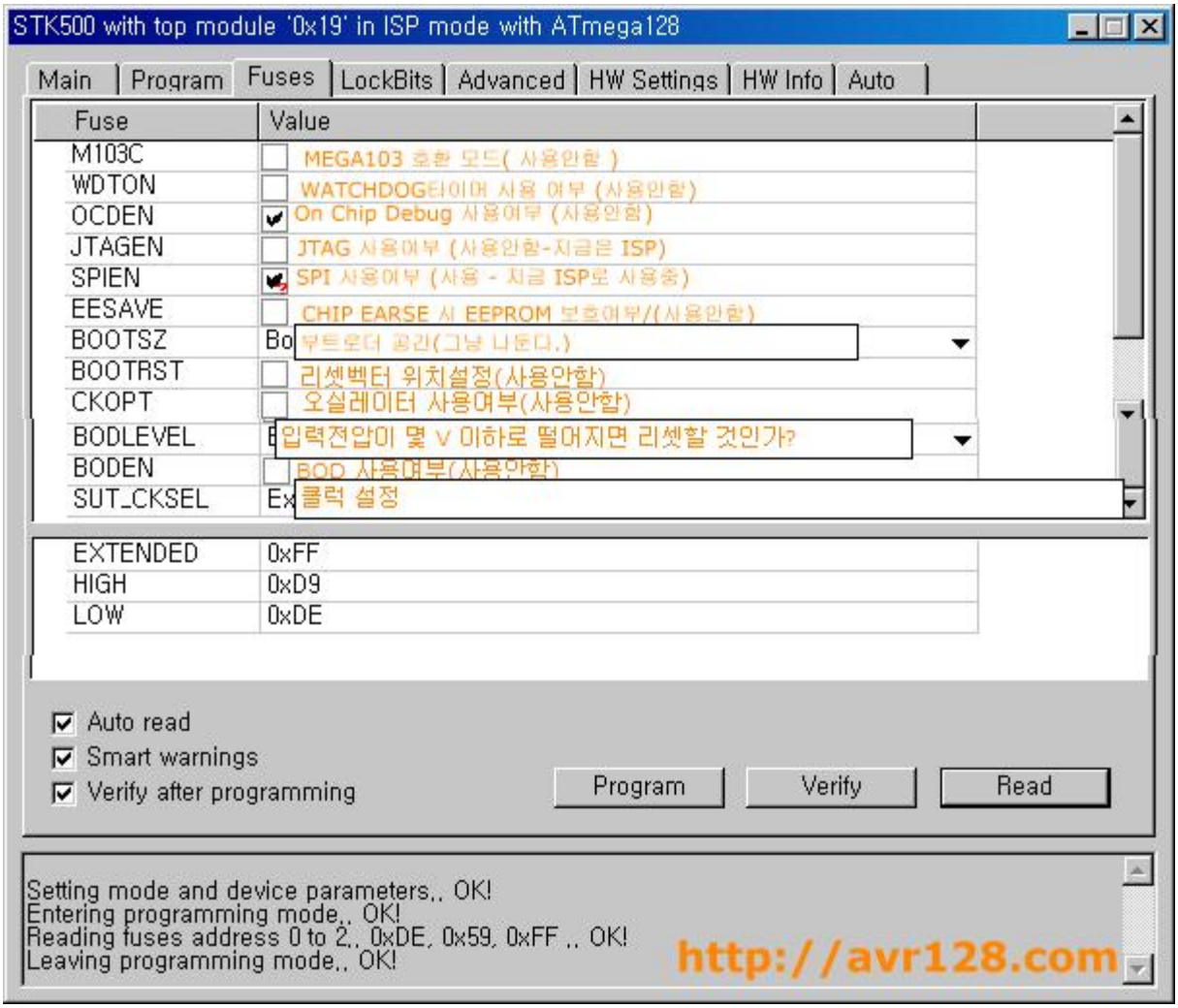
2.1.3. 프로그램 라이팅

[...]으로 헥사파일을 선택한 후 [Program]으로 라이팅을 할 수 있습니다.



2.1.4. FUSE(퓨즈) 설정

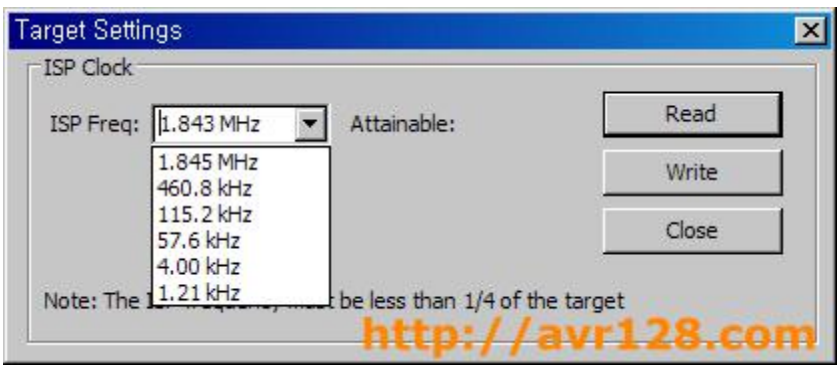
아래와 같이 설정할 수 있습니다. 가장 많이 사용되는 설정이며 필요에 따라 변경할 수 있습니다.



2.1.5. 에러 발생시

만약 다음과 같은 에러가 발생한다면 AVR 모듈이 연결되었는지 확인하고 연결되어 있다면 SCK 속도를 낮춰가며 적당한 라이팅 속도를 찾아야 합니다.

AVRStudio 에서는 총 6가지의 ISP 라이팅 속도를 제공합니다.



2.2. 프린터 포트용 AVR ISP

그림처럼 AVR 모듈과 ISP 케이블을 연결하고 ISP 프로그램케이블을 PC의 프린터 포트에 연결합니다.

http://www.lancos.com 에서 Ponyprog 프로그램을 다운받을 수 있습니다.

< AVR128 모듈과 패러럴 ISP 연결 >

PonyProg 프로그램을 실행하면 아래와 같이 프로그램이 실행됩니다.

< PONYPROG 실행 화면 >

Device Family 선택에서 ATMEGA128일 경우에는 AVR Micro를 선택합니다.

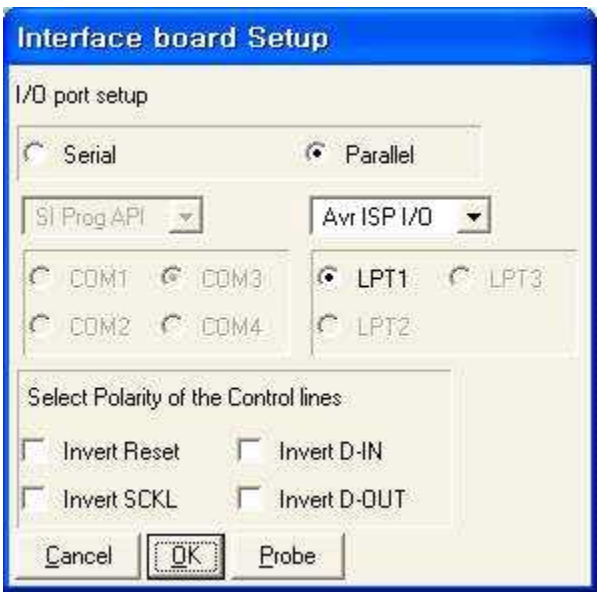
< Device Family 선택 >

Device Type을 선택에서 ATMEGA128일 경우에는 ATMEGA128을 선택합니다.

< Device Type 선택 >

다음으로 프로그램을 라이팅에 사용할 포트을 설정합니다..
〔Setup〕 => 〔Interface Setup〕 메뉴를 선택하여 설정할 수 있습니다.

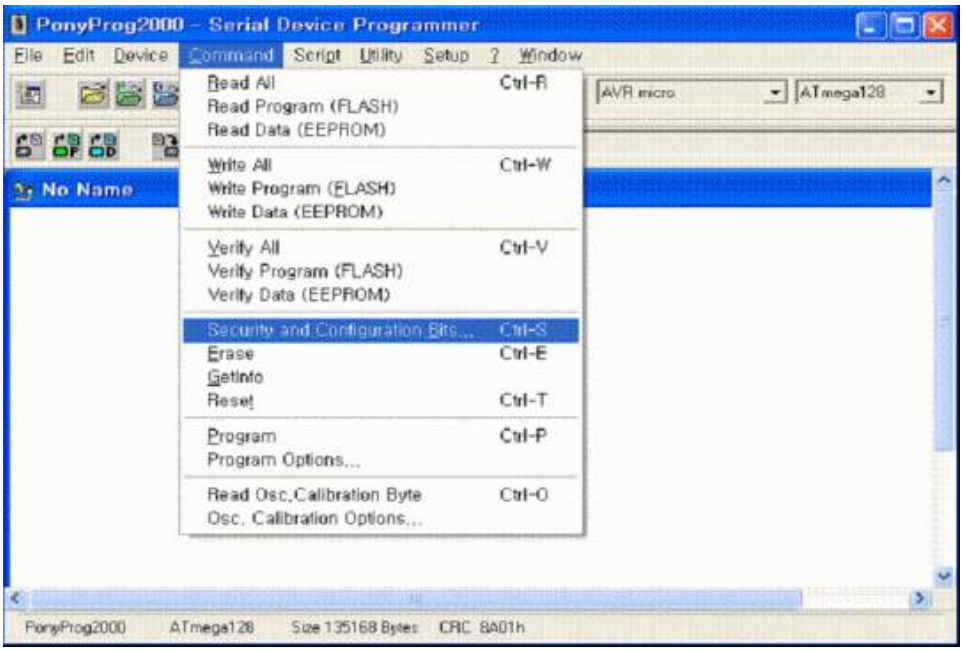
PC의 프린터 포트(LPT1)로 AVR-ISP를 사용할 경우의 셋팅은 아래와 같습니다.



< interface 설정 >

AVR을 사용하기 전에 fuse를 설정해야 합니다..

〔Command〕 => 〔Security and Configuration Bits〕 메뉴를 선택하여 Configuration Bilts 창을 열어 하드웨어에 맞게 설정합니다.

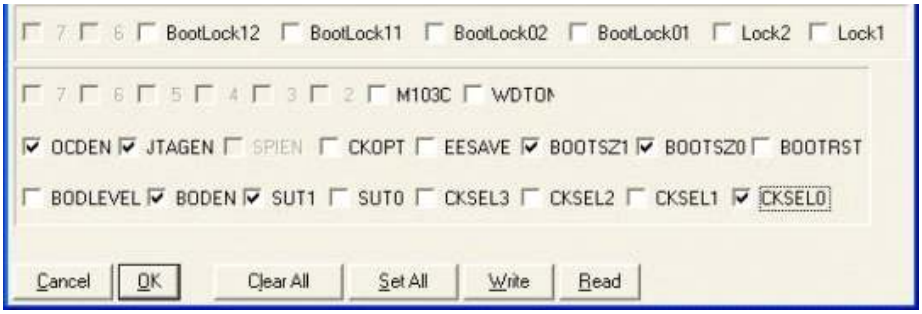


주의

AVR 시리즈는 fuse 라는 것이 있는데 매우 중요하다.
초급 사용자라면 AVR의 fuse 설정을 변경하지 말아야 한다.
설정이 잘못될 경우 모듈이 오동작할 수도 있고, 불능상태가 될 수도 있다.

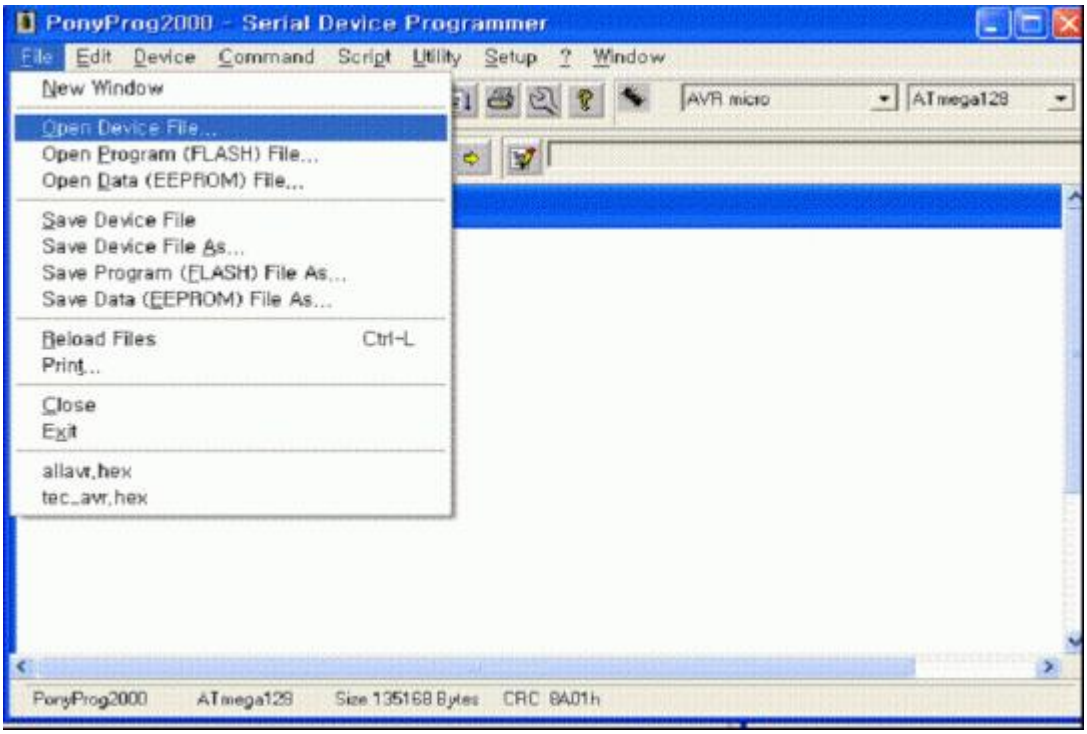
fuse에 대해 자세한 내용은 ATMEGA128A 매뉴얼을 참조하면 된다.

WAT-AVR128 모듈에서 기본으로 제공하는 퓨즈 셋팅은 다음과 같다.

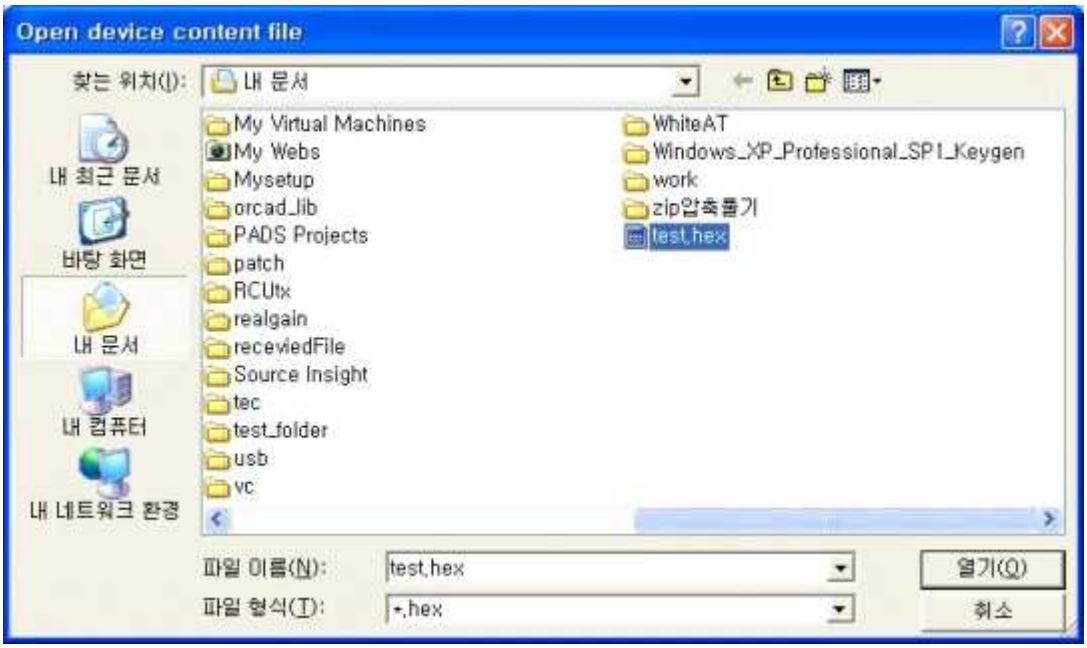


- 외부 크리스탈(11.0592MHz) 사용

이제 기본적인 세팅은 끝났으며 이제 헥사파일을 라이팅 하면 됩니다.
〔File〕 => 〔Open Device File〕 메뉴를 선택하여 헥사파일을 선택합니다.

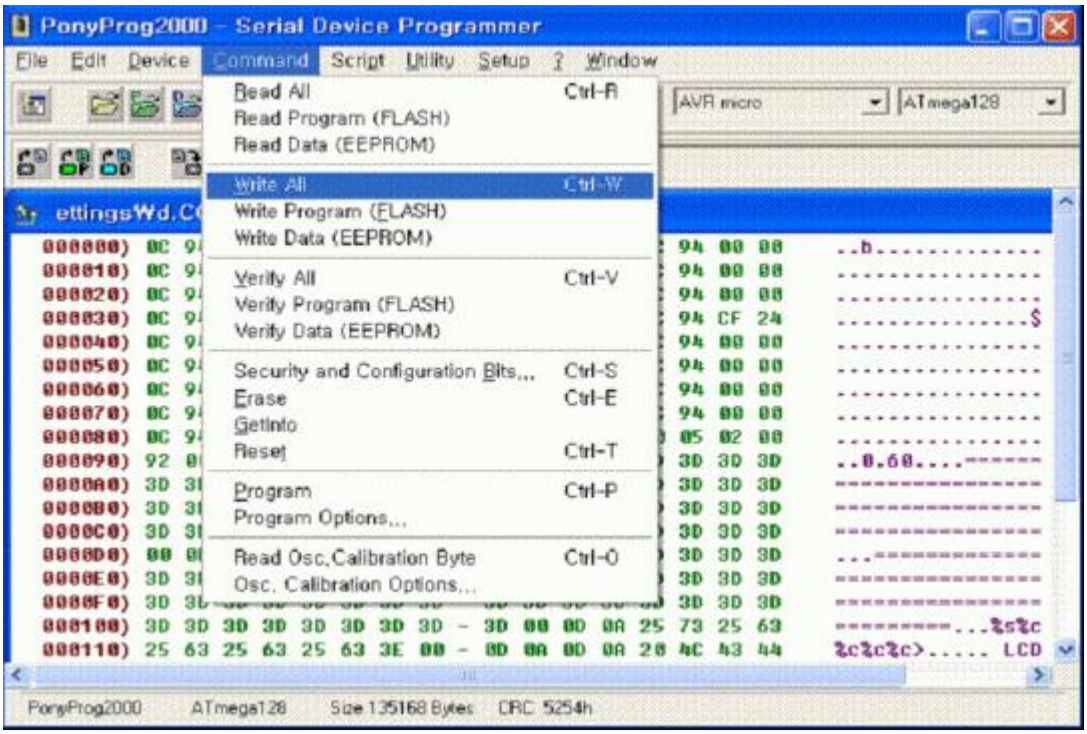


< PONYPROG 프로그램 라이팅 >



< 헥사 파일 선택 >

헥사파일을 선택한 후에 [Command] => [Write All]을 실행하면 라이팅이 진행됩니다.



3부

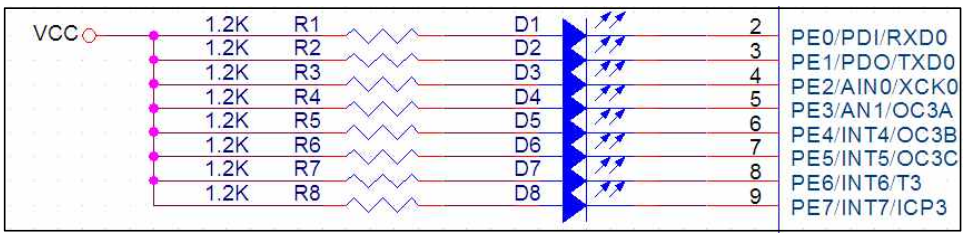
ATMEGA128A
연습

1. I/O 포트

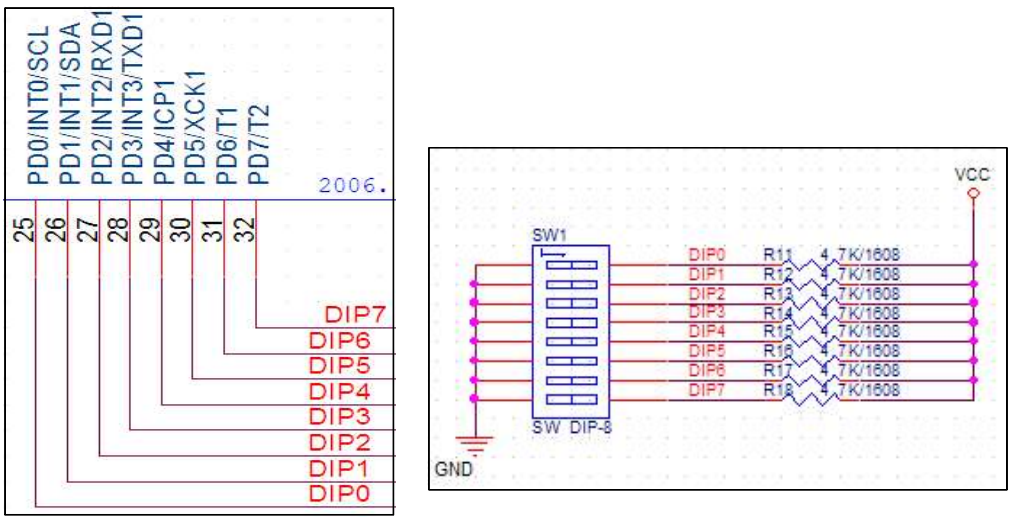
INPUT/OUPUT는 ATMEGA128 의 I/O 레지스터를 통해서 제어할 수 있습니다.

1.1. 준비

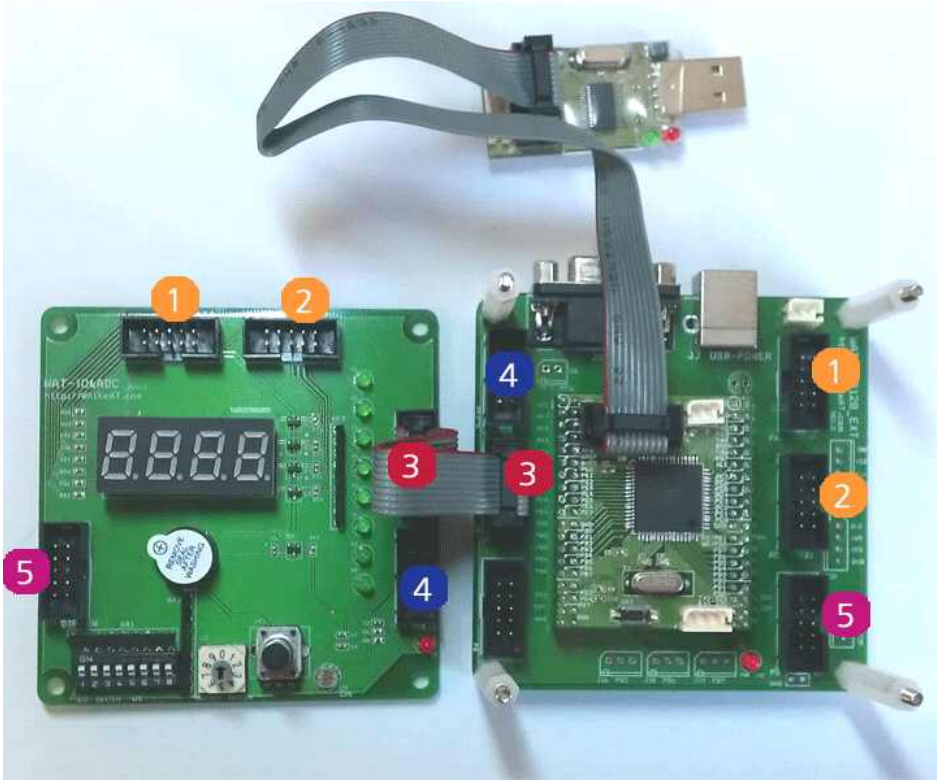
I/O 포트에 대한 실험을 해보겠습니다. 실험에 사용할 회로와 WAT128 보드의 연결 방법은 아래와 같습니다. PORTA 는 FND DATA 신호, PORTC 는 FND 선택 신호, PORTD는 딥스위치, PORTE는 LED, PORTF 는 ADC 신호에 연결되어 있습니다.



< LED 연결 회로도 >

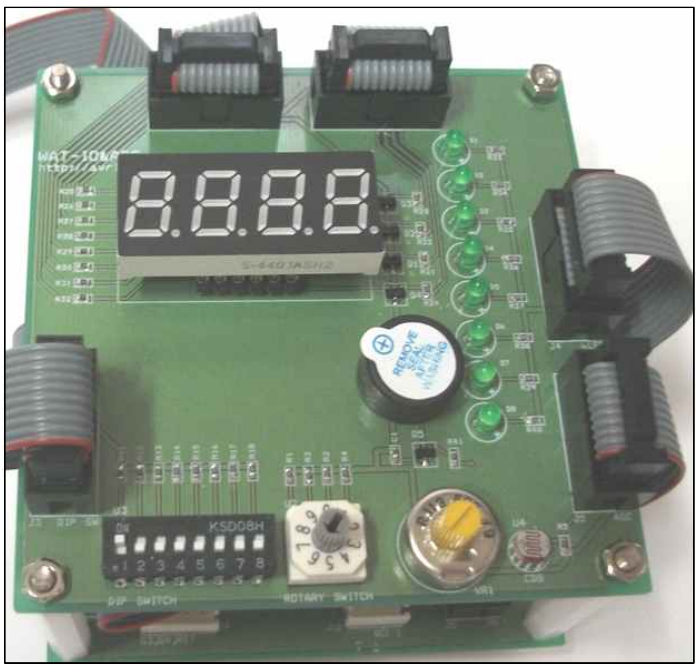


< DIP SWITCH 연결 회로도 >



< WAT-AVR128_EXT 보드와 WAT-IO&ADC 보드를 연결 중 >

< AVR128_EXT 보드와 IO&ADC 보드의 LED만 연결 >

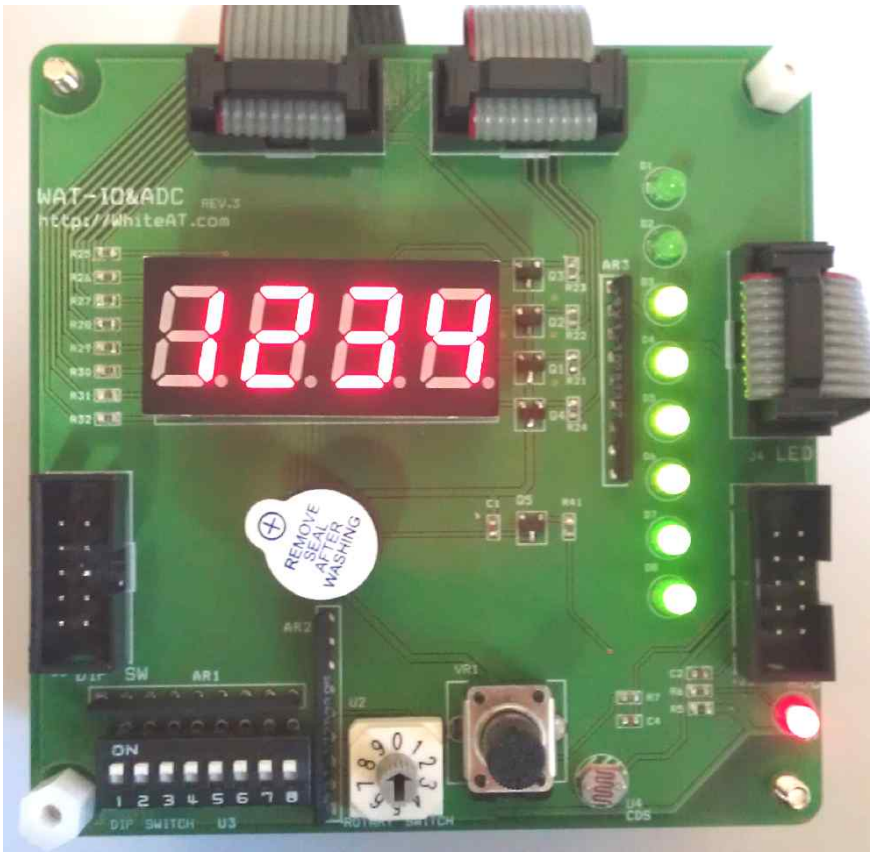


< AVR128_EXT, IO&ADC 보드의 모든 핀 연결 >

1.2. 보드 동작

ATMEGA128 MCU 와 I/O 테스트 모듈을 연결하였으니 제대로 연결되었는지를 확인해 봐야 하는데 먼저 http://whiteat.com/57035 에 있는 펌웨어 압축파일 안의 헥사 파일을 ATMEGA128 MCU로 다운로딩 합니다.

아래와 같이 우측 하단에 전원 연결을 알려 주는 LED가 켜지고, FND에는 "1234" 가 보이고 LED는 D1 은 꺼지고, D2는 깜빡이고, D3 ~ D8 이 켜지면 정상적으로 연결을 한 것입니다.



위처럼 동작하지 않는다면 케이블 연결을 확인한 후 다시 연결하시면 됩니다.

1.3. LED 구동

LED는 양 쪽에 걸리는 전압차로 인해 ON 되는 소자입니다. 양쪽에 모두 0V 또는 5V를 인가하면 LED는 켜지지 않습니다. 한쪽에는 5V, 다른 족에는 0V를 연결하고 적당한 저항이 직렬로 연결되어 있어야 LED를 안전하게 ON 할 수 있습니다.

ATMEGA128의 PORTE 에 연결된 LED를 ON하려면 PORTE를 LOW(0V) 로 출력해야 합니다. PORTE로 HIGH(5V)를 출력하면 LED 양 쪽에 인가되는 전압이 모두 5V 이기 때문에 LED 는 켜지지 않습니다.

1.4. PORT 출력

ATMEGA128의 포트 제어는 ATMEGA128 레지스터 값을 READ/WRITE 하여 가능 한 것입니다. 아래 코드를 예로 들겠습니다.

```
#include <avr/io.h>
int main(){
          DDRE = 0xFF;
          PORTE = 0x00;
        while(1){}
       return 0;
}
```

< 기본 코드 >

```
#include <avr/io.h>   // ATMEGA128 을 제어하기 위한 기본 함수와
                      //  레지스터 제공


int main()            // C 언어의 시작은 main
{
          DDRE = 0xFF;       // PORTE 를 출력으로 설정
          PORTE = 0x00;      // LED 켜기
         while(1){}          // 무한히 대기
       return 0;
}
```

C언어를 조금이라도 배운 분이라면 main, while, return은 아실 겁니다. 하지만 도대체 DDRE, PORTE 는 무엇인가? 변수 같으나 선언한 적이 없고 에러도 없이 컴파일 됩니다.

DDRE, PORTE 는 **레지스터(Register)**입니다.

레지스터는 C언에서 변수처럼 사용되며 레지스터에 8비트 값을 Writing 하여 ATMEGA128 을 제어가 가능한 것입니다. DDRE는 PORTE의 INPUT/OUTPUT을 설정하는 레지스터이고 PORTE는 PORTE의 출력 값을 제어하는 레지스터입니다.

이 실험에서 자주 사용하게 되는 레지스터는 DDRA, PORTA, DDRB, PORTB, DDRC, PORTC, DDRD, PORTD, DDRE, PORTE, DDRF, PORTF 이며 ATMEGA128에는 100개 이상의 레지스터가 있습니다.

다시 코드로 돌아 와서 DDRE = 0xFF 는 DDRE = 0b11111111 과 같고 PORTE = 0xFF 는 PORTE = 0b11111111 와 같습니다.

즉, PORTE의 모든 포트를 출력으로 설정하고 모든 포트에 LOW(0V)를 출력하는 명령입니다. 이렇게 되면 모든 LED는 ON 됩니다.

만약 LED를 ON, OFF, ON, OFF, ON, OFF, ON, OFF로 하려면 아래와 같이 코딩하면 됩니다.

```
#include <avr/io.h>   // ATMEGA128 을 제어하기 위한 기본 함수와
                      //  레지스터 제공

int main()            // C 언어의 시작은 main
{
          DDRE = 0xFF;      // PORTE 를 출력으로 설정
          PORTE = 0x55;     // 0b01010101
         while(1){}         // 무한히 대기
       return 0;
}
```

이 때 **컴파일하고** ATMEGA128 보드로 **라이팅**하는 거 잊으시면 안 됩니다.!!

왜 DDRE = 0x55로 하지 않을까요?

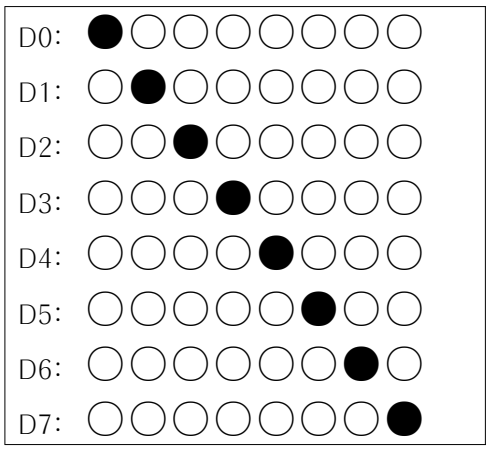
PORTE의 값이 HIGH(5V)이거나 LOW(0V)에 상관없이 PORTE는 출력으로 사용됩니다. PORTE의 출력은 PORTE 레지스터로 제어하며 포트를 출력으로 설정(DDRE=0xFF)해야 정확한 제어가 가능합니다.

만약 DDRE=0x55 로 설정하면 원하는 대로 동작할 가능성도 있지만 나머지 4개의 핀은 입력으로 설정되어 입력된 임의의 값으로 제어되어 원치 않은 결과를 얻을 수 있으므로 주의해야 합니다.

연습 1-1

PORTE의 0 ~ 7 핀에 LED 8개가 연결되어 있고 LED에 불이 들어오게 하려면 해당되는 핀에 '0' 을 출력하면 됩니다. D0(PORTA.0에 연결된 LED)부터 D7까지 차례대로 일정시간 ON된 후 OFF 되게 만들어 보겠습니다.

단 마지막 LED 가 ON/OFF 된 후에는 다시 처음으로 되돌아가서 무한 반복해야 합니다.



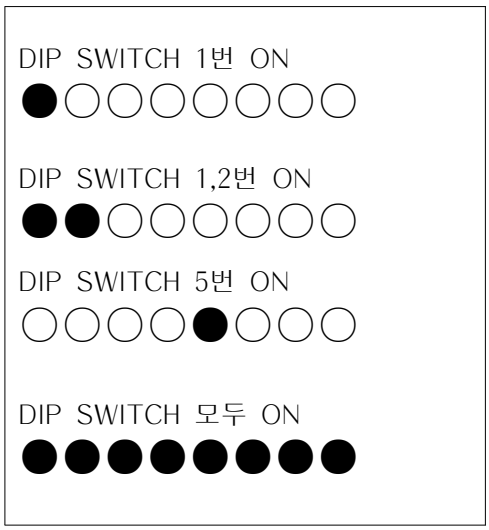
코드 1-1

```
/*
   EX_01_01.c

   LED 순차적으로 ON 하기
   AVRStudio  4.18


*/

#include <avr/io.h>

// 일정 시간 딜레이 (약 1초)
void Delay()
{
   register unsigned long i;
   for(i = 0; i < 300000; i++)
   {
            asm volatile(" PUSH  R0 ");
            asm volatile(" POP   R0 ");
            asm volatile(" PUSH  R0 ");
            asm volatile(" POP   R0 ");
            asm volatile(" PUSH  R0 ");
            asm volatile(" POP   R0 ");
```

```
            asm volatile(" PUSH  R0 ");
            asm volatile(" POP   R0 ");
            asm volatile(" PUSH  R0 ");
            asm volatile(" POP   R0 ");
    }
}

int main()
{
    DDRE = 0xFF;  // ALL OUTPUT
    PORTE = 0xFE; // PORTE.0    LED ON

    while(1)
    {
            Delay();

            if(PORTE == 0x7F)      // 마지막 LED 가 ON 되었다면.
            {
                    // 처음 LED ON 되게
                    PORTE = 0xFE; // PORTE.0    LED ON
            }
            else                   // 그렇지 않다면
            {
                PORTE <<= 1;  // 한 칸 이동 ( PORTE = PORTE<< 1; 와 동일 )
                PORTE |= 1;    // 마지막 LED 는 OFF ( PORTE = PORTE | 1; 와 동일 )
            }
    }
}
```

연습 1-2

PORTE의 0 ~ 7 핀에 LED 8개가 연결되어 있고 LED에 불이 들어오게 하려면 해당되는 핀에 '0' 을 출력하면 됩니다. PORTD 에 연결된 딥스위치 ON/OFF로 LED 를 제어할 수 있는데 딥스위치 1번을 ON 하면 D0가 ON, 딥스위치 1번을 OFF 하면 D0가 OFF 되고, 나머지 딥스위치 2 ~ 7번와 LED도 마찬가지로 ON/OFF 하는 예제입니다.

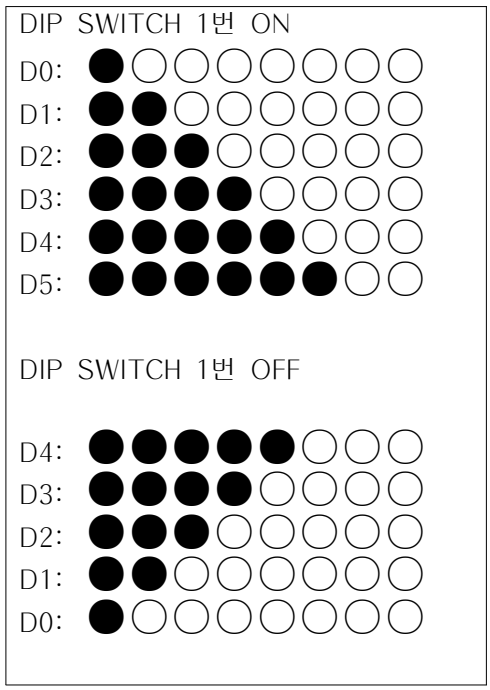


코드 1-2

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


int main()
{
   DDRE = 0xFF;  // PORTE ALL OUTPUT
   PORTE = 0xFF; // ALL LED ON

   DDRD = 0x00;  // INPUT for DIP SWITCH

   while(1)
   {
          PORTE = PIND;  // 딥스위치 값 PORTE 로 바로
   }
}
```

연습 1-3

PORTE의 0 ~ 7 핀에 LED 8개가 연결되어 있고 LED에 불이 들어오게 하려면 해당 핀에 '0' 을 출력하면 됩니다. 딥스위치 1번을 ON 하면 D0(PORTE.0에 연결된 LED)부터 D7까지 차례대로 ON되고 딥스위치 1번을 OFF 하면 마지막 켜진 LED부터 OFF 되는 예제입니다.



코드 1-3

```
/*
   EX_01_03.c

   PORTE의 0 ~ 7 포트에 LED 8개가 연결되어 있고
   LED에 불이 들어오게 하려면 해당되는 포트에
    '0' 을 출력하면 됩니다.

   딥스위치 1번을 ON 하면
   D0(PORTE.0에 연결된 LED)부터 D7까지 차례대로 ON되고
   딥스위치 1번을 OFF 하면 마지막 켜진 LED부터 OFF 되는
   예제입니다.

   LED: PORTE 에 연결
   DIP SWITCH: PORTD 에 연결


   AVRStudio  4.18

*/

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

```
// 일정 시간 딜레이
void Delay()
{
   register unsigned long i;
   for(i = 0; i < 300000; i++)
   {
            asm volatile(" PUSH  R0 ");
            asm volatile(" POP   R0 ");
            asm volatile(" PUSH  R0 ");
            asm volatile(" POP   R0 ");
            asm volatile(" PUSH  R0 ");
            asm volatile(" POP   R0 ");
            asm volatile(" PUSH  R0 ");
            asm volatile(" POP   R0 ");
            asm volatile(" PUSH  R0 ");
            asm volatile(" POP   R0 ");
   }
}

int main()
{
   DDRE = 0xFF;  // PORTE OUTPUT
   PORTE = 0xFF; // ALL LED OFF

   DDRD = 0x00;          // DIP SWITCH 용으로 입력

   while(1)
   {

            Delay();

            if((PIND & 0x01) == 0x00)
            {

                     // DIP 1 이 ON 이면
                     PORTE <<= 1;
                     PORTE |= 0;           // 마지막 LED 는 ON 되게

            }
            else
            {

                     //      DIP 1이 OFF 이면
                     PORTE >>=1;
```

```
                PORTE |=0x80; // 처음 LED 는 OFF 되게

            }
        }
    }
```

연습 1-4

로타리 스위치 값을 Anode Type의 FND 의 첫 번째 자리에 출력하며 로터리 스위치 값이 '0' 일 때 잠깐 동안 '삑' 소리를 내고 두 번째 자리의 수를 증가하는 예제입니다.

코드 1-4

```
/*******************************************

로터리(Rotary) 스위치 값을 Anode FND 1자리에 표시하는 예제

WAT-AVR128 모듈의 PORTA => WAT-IO&ADC 보드의 FND DATA(J1) 에 연결
WAT-AVR128 모듈의 PORTC => WAT-IO&ADC 보드의 FND CONTROL(J2) 에 연결
WAT-AVR128 모듈의 PORTF => WAT-IO&ADC 보드의 ADC(J5) 에 연결

PORTC.0 : 로터리 0에 연결 (입력)
PORTC.1 : 로터리 1에 연결 (입력)
PORTC.2 : 로터리 2에 연결 (입력)
PORTC.3 : 로터리 3에 연결 (입력)


PORTC.4 : FND 1자리에 SEL 신호 (출력)
PORTC.5 : FND 2자리에 SEL 신호 (출력)
PORTC.6 : FND 3자리에 SEL 신호 (출력)
PORTC.7 : FND 4자리에 SEL 신호 (출력)


PORTA : FND 데이터 신호(출력)


Main Clock : 11.0592Mhz

Tools : AVR Studio     4.16
실험보드 : WAT-AVR128 보드 + WAT-IO&ADC
```

```
*****************************************/

#include <avr/io.h>
#include "wat128.h"

int main()
{
   BYTE byteLastFND0 = 0;

   // FND4 초기화
   InitFND4();

   // 로터리 스위치 초기화
   InitRotary();
   g_FND[0] = 0;
   g_FND[1] = 0;
   g_FND[2] = 3;
   g_FND[3] = 4;

   BUZZER_INIT;

   while (1)
   {
         // 로터리 스위치 값 받기
         g_FND[0] =  GetRotaryInt();

         if(g_FND[0] != byteLastFND0)
         {
               byteLastFND0 = g_FND[0];

               if(byteLastFND0 ==0)
               {

                     // 비프음 내기
                     BUZZER_ON;
                     DelayMS(10);
                     BUZZER_OFF;

                     // 두번째 자리수 증가
                     if(++g_FND[1]>=10)
                           g_FND[1] = 0;
               }
         }
```

```
            // FND 4자리 표시
            DisplayFND4(g_FND[0],g_FND[1],g_FND[2],g_FND[3]);
        }
    }
```

연습 1-5

키 매트릭스(WAT-KEY4x4) 제어 예제입니다. 키 값을 실시간으로 PC로 전송하여
모니터에 출력하는 예제입니다.

코드 1-5

```
/*

필요한 보드
   1. WAT-AVR128 (모듈)
   2. WAT-AVR128 EXT (확장보드)
   3. WAT-KEY 4x4 (키패드)
   => WAT-KEY 4x4 보드를 PORTA 에 연결한 예

기능
   4x4 배열의 키 상태를 시리얼 통신으로 PC에서 실시간으로 감시한다.
   - C#  (VS2008)


*/


#include <avr/io.h>
#include "WAT128.h"

void OperatingFromPC();

int main(){

   // 시리얼 통신으로 PC에 전송하기 위한 보레이트 설정
   OpenSCI0(57600);


   // 키패드 초기화
   InitKey4x4();


   while(1)
```

```
    {
        // 키패드의 값을 읽어서 PC로 전송
        OperatingFromPC();


    }
}


BOOL    bPCHeadCheckOK = FALSE;
INT16US g_uiPCRecevingDataCount = 0;
BYTE    g_bytePCData[RX_BUFFER_SIZE];

//! RX 버퍼 초기화
void RXClearFromPC(){
    g_uiPCRecevingDataCount = 0;
    bPCHeadCheckOK = FALSE;
}

//! RX 처리 루틴
void OperatingFromPC(){
    BYTE byteCommand;

    INT16S iRxData;
    iRxData=GetByte0();



    if( 0<= iRxData && iRxData<=255 ){


        if (TRUE == bPCHeadCheckOK)
        {
            // STX 를 통과했다면(이미 나왔다면)
            g_bytePCData[g_uiPCRecevingDataCount] = iRxData;
            if(++g_uiPCRecevingDataCount>=RX_BUFFER_SIZE )
            {
                g_uiPCRecevingDataCount =0;
            }

            if (COMM_ETX == iRxData )
            {
                byteCommand = g_bytePCData[0];
                switch(byteCommand)
                {
```

```
                        case 0x10:      // key 4x4
                                if ( g_uiPCRecevingDataCount>=7)
                                {
                                        if (COMM_ETX == g_bytePCData[6] )
                                        {
                                                SendKey4x4( GetKey4x4());
                                        }
                                        RXClearFromPC();
                                }
                                break;

                        default:
                                RXClearFromPC();
                                break;
                        }
                }
        }
        else
        {
                // STX 가 나오지 않았다면 STX 가 나올때까지 기다린다.
                if (COMM_STX == iRxData)
                {
                        bPCHeadCheckOK = TRUE;
                        g_uiPCRecevingDataCount = 0;

                }else{

                }
        }
    }
}
```

2. LCD 제어

2.1. Character LCD

16x2 배열의 Character LCD 는 일반적으로 16개의 핀이 있으며 각 기능은 다음과 같습니다.

핀번호	핀명	기능
1	GND	Ground, 0V
2	VDD	Logic power supply, +5V
3	VO	Voltage for LCD drive
4	RS	Data / Instruction register select
5	R/W	Read / Write
6	E	Enable signal, start data read/write
7	D0	Data Bus Line
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	
15	A	LED Anode, power supply +
16	K	LED Cathode, ground 0V

Character LCD 는 총 11개의 Instruction(명령)이 있으며 다음과 같습니다.

Instruction	Instruction Code										DESCRIPTION	Executed Time(fosc =270KHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write “20H” to DDRAM and set DDRAM address to “00H ” from AC	1.53mS
Cursor At Home	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to “00H” from AC and return cursor to its original Position if shifted. The contents of DDRAM are not changed.	1.53mS
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display.	39μS
Display On/Off Control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor(C), and Blinking of cursor(B) ON/OFF control bit.	39μS
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	-	-	Set cursor moving and display shifts cursor bit, and the direction, without changing of DDRAM data.	39μS
Function Set	0	0	0	0	1	DL	N	F	-	-	Sets interface data length (DL:8-BIT/4-BIT), number of display lines(N:2-line/1-line) and, display font type (F:5x11dots/5x8 dots).	39μS
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter.	39μS
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter.	39μS
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0μS
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM / CGRAM)	43μS
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Reads data from internal RAM (DDRAM / CGRAM).	43μS

*"-":don't care

NOTE : When an MPU program with checking the Busy Flag(DB7) is made, it must be necessary 1/2Fosc is necessary for executing the next instruction by the falling edge of the 'E' signal after the Busy Flag(DB7)goes to "LOW" .

LCD화면을 CLEAR, 커서를 좌측 상단으로 이동하는 방법입니다.

화면 CLEAR

핀번호	핀명	데이터
4	RS	LOW
5	R/W	LOW
7 ~ 14	D0 ~ D7	0x01

커서를 좌측 상단으로 이동

핀번호	핀명	데이터
4	RS	LOW
5	R/W	LOW
7 ~ 14	D0 ~ D7	0x02

이렇게 LCD에 명령하는 작업을 타이밍을 고려하여 함수로 만들어 보면 다음과 같습니다.

```
void CLCD_Command(BYTE data)
{
        while(CLCD_BusyCheck()){}   // 앞의 LCD 명령이 완료될 때까지 기다림
        ClearBit(CLCD_CONTROL_PORT,CLCD_RS);
        ClearBit(CLCD_CONTROL_PORT,CLCD_EA);
        DelayUS(1);  // 1uS 딜레이
        CLCD_DATA_PORT= data;
        DelayUS(50);  // 50uS 딜레이
        SetBit(CLCD_CONTROL_PORT,CLCD_EA);
        DelayUS(20);    // 20uS 딜레이
        ClearBit(CLCD_CONTROL_PORT,CLCD_EA);
        DelayUS(50);       // 50uS 딜레이
}

CLCD_Command(1); // 화면 CLEAR
CLCD_Command(2); // 커서를 HOME 으로
```

LCD화면에 한 문자를 출력하는 방법입니다.

1 바이트 문자 출력

핀번호	핀명	데이터
4	RS	HIGH
5	R/W	LOW
7 ~ 14	D0 ~ D7	출력할 문자 값

1문자를 출력하는 작업을 타이밍을 고려하여 함수로 만들어 보면 다음과 같습니다.

```
void CLCD_PutChar(BYTE data)
{
        while(CLCD_BusyCheck()){}    // 앞의 LCD 명령이 완료될 때까지 기다림
        ClearBit(CLCD_CONTROL_PORT,CLCD_EA);
        SetBit(CLCD_CONTROL_PORT,CLCD_RS);
        DelayUS(2);                  // 2 us delay
        CLCD_DATA_PORT= data;
        DelayUS(50);                 //    delay
        SetBit(CLCD_CONTROL_PORT,CLCD_EA);
        DelayUS(50);                 //    delay
        ClearBit(CLCD_CONTROL_PORT,CLCD_EA);
}

CLCD_PutChar( 'A' ); // 'A' 출력
CLCD_PutChar( 'y' ); // 'y' 출력
CLCD_PutChar(0x57); // 'W' 출력
```

2.2. Character LCD 실험

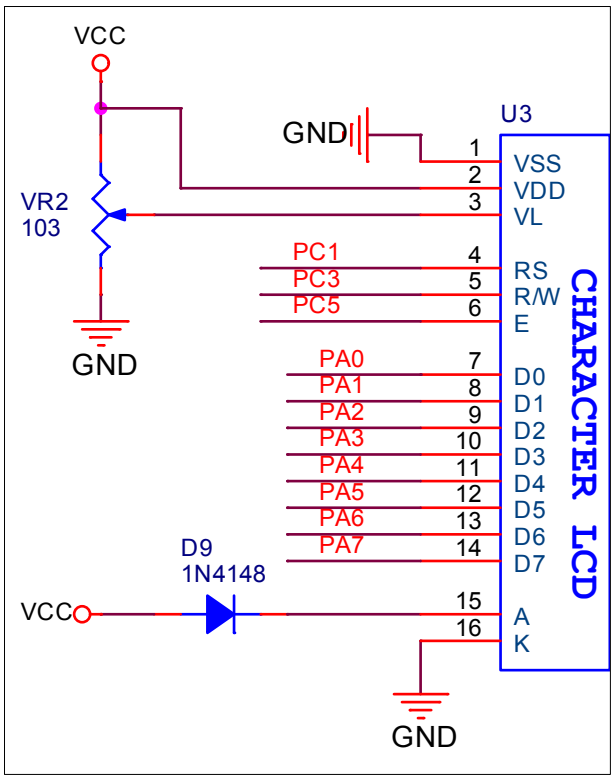
WAT-CLCD와 WAT-AVR128 모듈을 아래와 같이 연결하여 실험을 진행하겠습니다. PORTA 에 LCD 데이터 신호를 연결하고 PORTC 에는 컨트롤(RW, CS, E)를 연결합니다.

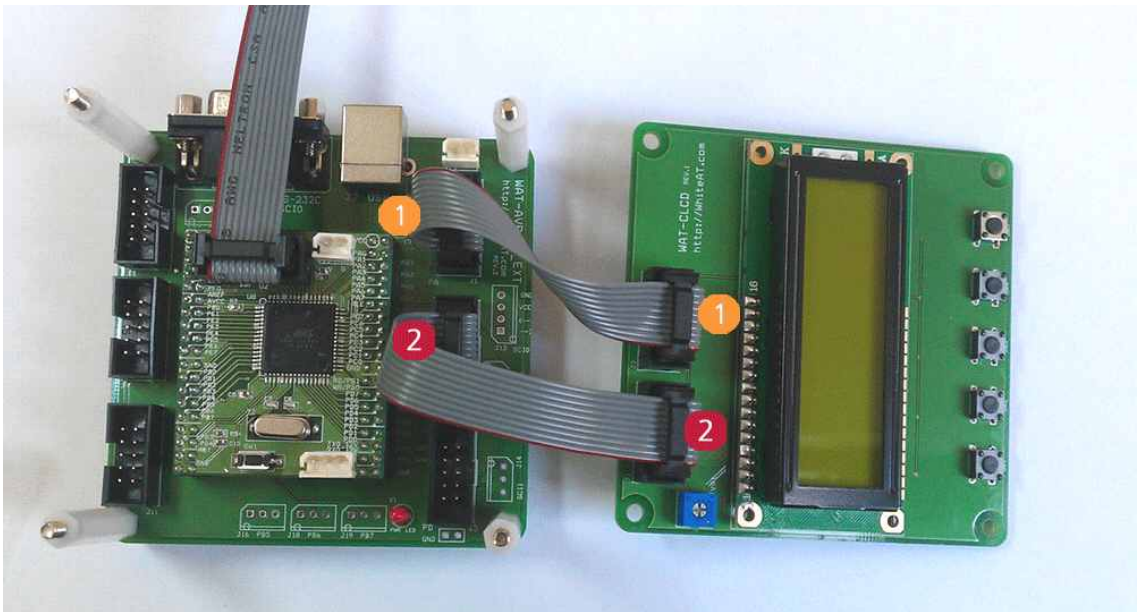
VR1의 5K 가변 저항으로 Graphics LCD의 백라이트 밝기를 조절할 수 있습니다.

- RS : PC1
- R/W: PC3
- E : PC5

- D0 : PA0
- D1 : PA1
- D2 : PA2
- D3 : PA3
- D4 : PA4
- D5 : PA5
- D6 : PA6
- D7 : PA7

- BUTTON1 : PC0
- BUTTON2 : PC2
- BUTTON3 : PC4
- BUTTON4 : PC6
- BUTTON5 : PC7





< WAT-AVR128_EXT 보드와 WAT-CLCD 보드를 연결 중 >

< AVR128_EXT, CLCD 보드 연결 완료 >

ATMEGA128 MCU 와 WAT-CLCD 모듈을 연결하였으니 제대로 연결되었는지를 확인해 봐야 하는데 먼저 http://whiteat.com/68839 에 있는 펌웨어 압축파일 안의 헥사 파일을 ATMEGA128 MCU로 다운로딩 합니다.

아래와 같이 LCD에 “abcdefghijklmnop" 와 ” 0123456789ABCDEF" 가 출력되고 하단의 버튼을 누르면 아랫줄의 숫자가 변경되면 정상적으로 연결을 한 것입니다.

< AVR128_EXT CLCD 보드 동작 >

위처럼 동작하지 않는다면 케이블 연결을 확인한 후 다시 연결하시면 됩니다.

WAT128.h에서 기본적으로 제공하는 함수는 아래와 같습니다.

선언

```
#define CLCD_RS 1
#define CLCD_RW 3
#define CLCD_EA 5
#define CLCD_DATA_PORT PORTA
#define CLCD_DATA_PORT_DIR     DDRA

#define CLCD_CONTROL_PORT      PORTC
#define CLCD_CONTROL_PORT_DIR DDRC


void CLCD_XY(BYTE x1, BYTE y1);            // 출력할 좌표 이동
void CLCD_Command(BYTE data);              // CLCD 명령
void CLCD_On();                            // CLCD 켜기
BYTE CLCD_BusyCheck();                     // CLCD 대기 상태인지 체크
void CLCD_PutString(BYTE x, BYTE y, char  *str); // x,y 좌표에 문자열 출력
void CLCD_PutChar(BYTE chr);               // 한 문자 출력
void CLCD_Clear();                         // CLCD 클리어
void CLCD_Init();                          // CLCD 초기화
```

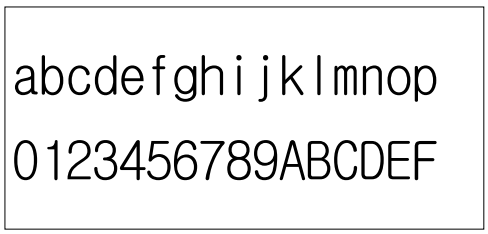
```
BYTE CLCD_BusyCheck()
{
        DelayUS(200);
        DelayUS(200);
        DelayUS(200);
        DelayUS(200);
        DelayUS(200);
        DelayUS(200);
        DelayUS(200);

        return (0);
}

void CLCD_PutString(BYTE x, BYTE y,  char *str)
{
        int len = 0;

        CLCD_XY(x,y);

        while(*str ){
                if(len>16) break;
                CLCD_PutChar(*str++);
                len++;
        }

}

void CLCD_PutChar(BYTE data)
{
        while(CLCD_BusyCheck());
        ClearBit(CLCD_CONTROL_PORT,CLCD_EA);
        SetBit(CLCD_CONTROL_PORT,CLCD_RS);
        DelayUS(2);
        CLCD_DATA_PORT= data;
        DelayUS(50);
        SetBit(CLCD_CONTROL_PORT,CLCD_EA);
        DelayUS(50);
        ClearBit(CLCD_CONTROL_PORT,CLCD_EA);
}
```

```
void CLCD_Clear()
{
        CLCD_Command(1);
        DelayMS(1);
        CLCD_Command(2);
        DelayMS(1);
}

void CLCD_XY(BYTE x, BYTE y)
{
        BYTE position = 0;
        if(x>16) x = 0;
        switch(y)
        {
        case 0 :
                position = 0x80;
                break;
        case 1 :
                position = 0xC0;
                break;
        }
        CLCD_Command(position + x);
}

void CLCD_Command(BYTE data)
{
        while(CLCD_BusyCheck()){}
        ClearBit(CLCD_CONTROL_PORT,CLCD_RS);
        ClearBit(CLCD_CONTROL_PORT,CLCD_EA);
        DelayUS(1);
        CLCD_DATA_PORT= data;
        DelayUS(50);
        SetBit(CLCD_CONTROL_PORT,CLCD_EA);
        DelayUS(20);
        ClearBit(CLCD_CONTROL_PORT,CLCD_EA);
        DelayUS(50);
}
```

```
void CLCD_On()
{
	CLCD_Command(0x01);// dummy
	CLCD_Command(0x38);
	CLCD_Command(0x0C );
	CLCD_Command(0x01);
	CLCD_Command(0x06);
	CLCD_Command(0x01);
	DelayMS(2);
}

void CLCD_Init()
{
	SetBit(CLCD_CONTROL_PORT_DIR,CLCD_EA);
	SetBit(CLCD_CONTROL_PORT_DIR,CLCD_RS);
	SetBit(CLCD_CONTROL_PORT_DIR,CLCD_RW);

	ClearBit(CLCD_CONTROL_PORT,CLCD_RW);

	CLCD_DATA_PORT_DIR = 0xFF;

	CLCD_On();
	CLCD_Clear();
}
```

연습 2-1

캐릭터 LCD에 우측처럼 영문과 숫자를 표시하며
버튼이 눌릴 경우 버튼 번호를 표시하는 예제입니다.



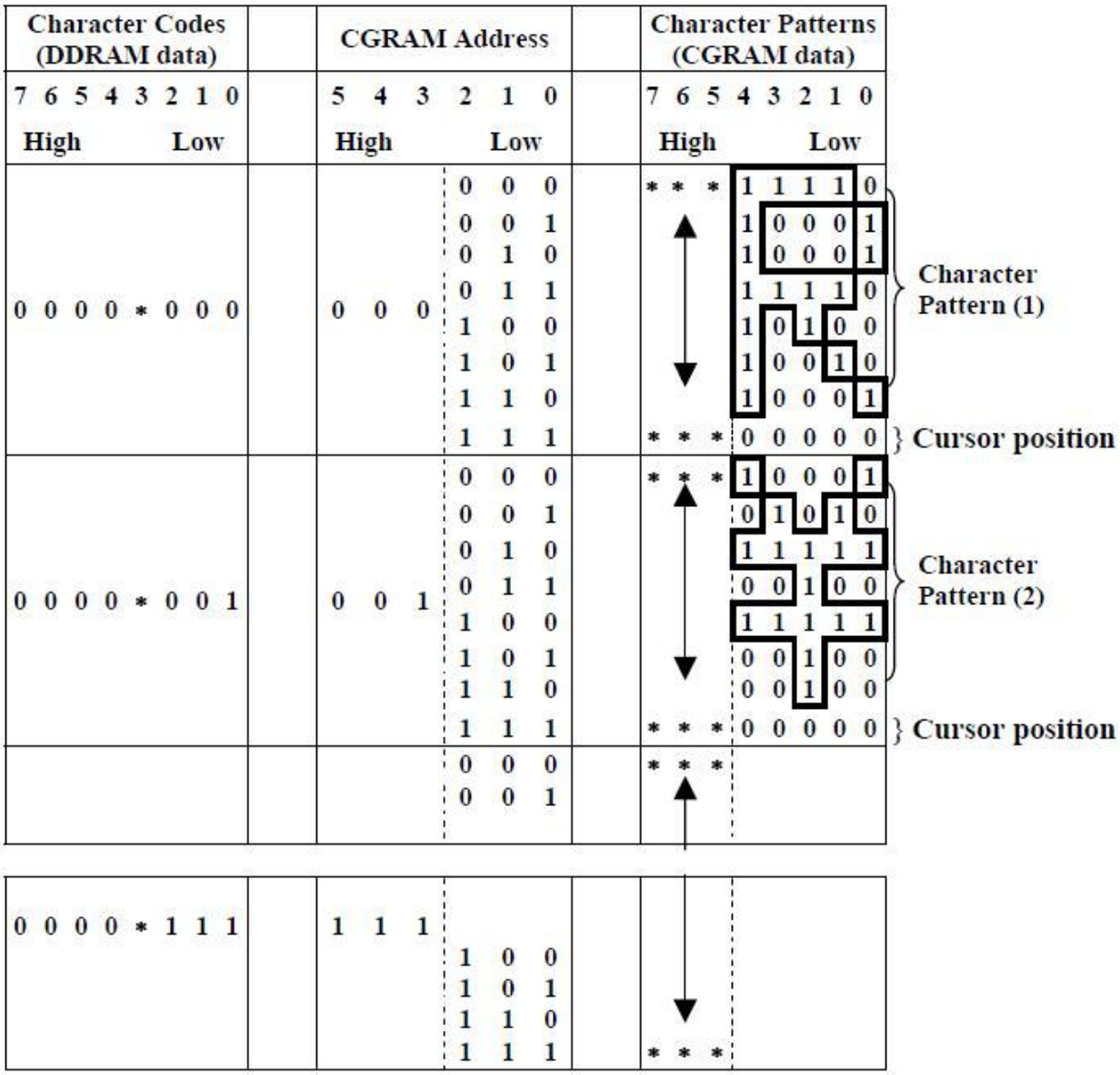
코드 2-1

```
#include <avr/io.h>
#include "WAT128.h"

int main()
{

    CLCD_Init();              // LCD 초기화

    CLCD_PutChar('a');
    CLCD_PutString(1,0,"bcdefghijklmnop");
    CLCD_PutString(0,1,"0123456789ABCDEF");

    while(1)
    {

        if((CLCD_CONTROL_PIN & 0x01) ==0)
        {
          CLCD_PutString(0,1,"1111111111111111");
        }
        else if((CLCD_CONTROL_PIN & 0x04) ==0)
        {
            CLCD_PutString(0,1,"2222222222222222");
        }
        else if((CLCD_CONTROL_PIN & 0x10) ==0)
        {
           CLCD_PutString(0,1,"3333333333333333");
        }
        else if((CLCD_CONTROL_PIN & 0x40) ==0)
        {
            CLCD_PutString(0,1,"4444444444444444");
        }
        else if((CLCD_CONTROL_PIN & 0x80) ==0)
        {
            CLCD_PutString(0,1,"5555555555555555");
        }
        else
        {
            CLCD_PutString(0,1,"0123456789ABCDEF");
        }
}
}
```

2.3. 사용자 지정 폰트 출력

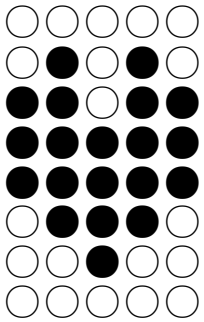
LCD에서 'A'. 'B' 등의 문자는 CGROM에 저장되어 있어서 아스키코드 값으로 바로 출력 가능합니다. . '가나다♡' 를 출력해 보겠습니다. 하지만 CGROM에 없는 폰트는 어떻게 출력할까요? CGRAM 을 사용하면 됩니다.

CGRAM 은 5x7폰트를 최대 8 문자가 저장됩니다.



CGRAM의 상위 2바이트는 01으로 고정(즉, 0x40)이며, 비트5,4,3은 8개의 데이터 공간이며, 하위 3비트는 데이터 주소입니다.

하트. 출력을 예로 들어 보겠습니다. 일단 폰트를 만들면 다음처럼 됩니다.



0x00
0x0A
0x1B
0x1F
0x1F
0x0E
0x04
0x00

폰트 데이터를 CGRAM에 입력합니다.

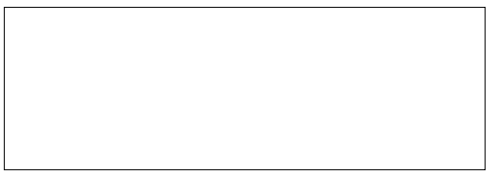
```
CLCD_Command(0x40);   // CGRAM 의 0번지부터 저장하겠다.

CLCD_Data(0x00);
CLCD_Data(0x0A);
CLCD_Data(0x1B);
CLCD_Data(0x1F);
CLCD_Data(0x1F);
CLCD_Data(0x0E);
CLCD_Data(0x04);
CLCD_Data(0x00);
```

폰트 데이터화면에 출력합니다.

```
CLCD_Command( 0x80); DDRAM 으로 보내겠다.(화면에 출력하겠다.)
CLCD_Data( 0x00);  // CGRAM의 0번지의 하트를 출력
```

연습 2-2

캐릭터 LCD에 우측처럼 가나다❤(사용자 지정 폰트)를 표시하는 예제입니다.

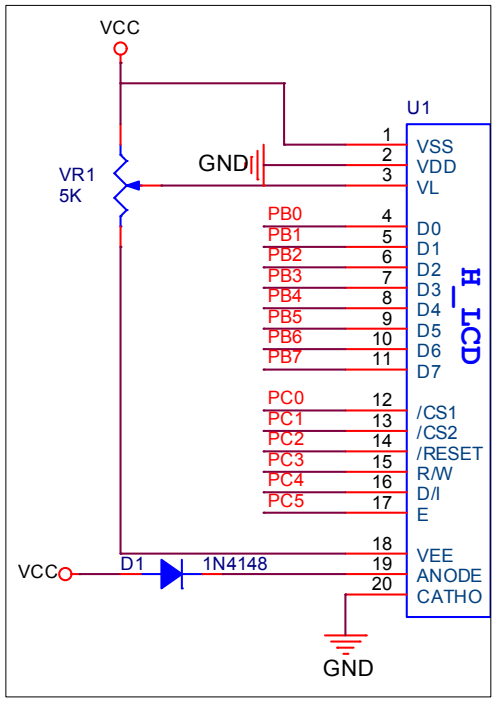


코드 2-2

```
#include <avr/io.h>
#include "WAT128.h"

// 폰트지정
void SetFont()
{
int i
unsigned char font[] = {0x02, 0x1a, 0x0a, 0x0a, 0x0b, 0x0a, 0x0a, 0x02,    //가
                        0x02, 0x12, 0x12, 0x12, 0x1B, 0x02, 0x02, 0x02,  //나
                        0x02, 0x1A, 0x12, 0x12, 0x1B,0x02, 0x02, 0x02,  //다
                        0x00, 0x0a, 0x1b, 0x1f, 0x1f, 0x0e, 0x04, 0x00};   //하트

    CLCD_Command(0x40);
    for(i=0; i<32; ++i)
    {
        CLCD_Data( font[i]);
    }
    return
}


int main()
{

    CLCD_Init();            // LCD 초기화

    SetFont();


    CLCD_Command( 0x80); //set ddram
    CLCD_Data( 0x00);  //가
    CLCD_Data( 0x01);  //나
    CLCD_Data( 0x02);  //다
    CLCD_Data( 0x03); // heart

    while(1)
    {
        // 무한대기
    }
}
```

2.4. Graphics LCD 연결

128x64 배열의 Graphics LCD (WAT-GCLD)를 제어해 보겠습니다.



< Graphics LCD 연결 회로도 >

LCD 의 DATA line은 ATMEGA128의 PORTA 에 연결하고, CONTROL line은 PORTC를 사용합니다. VR1의 5K 가변 저항은 Graphics LCD의 백라이트 밝기를 조절할 수 있습니다.

연습 2-3

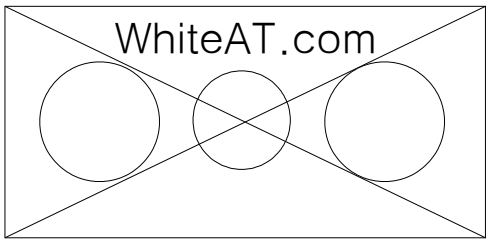
그래픽 LCD 에 우측에 있는 그림처럼 출력하고 3번째 줄부터는 영문 'E'를 계속 출력하는 예제입니다. 단 이때 현재 위치에 커서를 표시해야 합니다.

코드 2-3

```
#include <avr/io.h>
#include "WAT128.h"


int main()
{

        GLCD_Init();
        GLCD_ShowCursor(1);   // 커서를보이게하자

        GLCD_String(0,0,"    화이트앳    ");
        GLCD_String(1,0,"   WhiteAT.com  ");

        while(1)
        {
                // 약200ms 마다'E' 출력
                GLCD_English('E',0);
                DelayMS(200);
        }
}
```

연습 2-4

그래픽 LCD 에 사각형, 원, 선을 그리는 예제입니다.



코드 2-4

```
int main(){

        GLCD_Init();
        GLCD_String(0,0,"   WhiteAT.com  ");


        // 외각선(사각형) 그리기
        GLCD_DrawRect(0,0,127,63);

        // 대각선 그리기
        GLCD_DOT_Line(0,0,127,63);
        GLCD_DOT_Line(0,63,127,0);

        // 원 그리기
        GLCD_DOT_DrawCircle(64,32,10);
        GLCD_DOT_DrawCircle(32,32,14);
        GLCD_DOT_DrawCircle(96,32,14);

        while(1)
        {

        }
}
```

3. 인터럽트

인터럽트는 프로그램이 진행되는 흐름을 변경하는 한 방법입니다. 인터럽트가 걸리면 다음 명령을 실행하지 않고 인터럽트 서비스 루틴을 먼저 실행한 후 다음 명령이 실행됩니다. 인터럽트 서비스 루틴과 관련된 인터럽트 벡터, 인터럽트 우선순위, 관련 레지스터 등에 대한 설명을 주로 다룰 것입니다.

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	TIMER2 COMP	Timer/Counter2 Compare Match
11	$0014	TIMER2 OVF	Timer/Counter2 Overflow
12	$0016	TIMER1 CAPT	Timer/Counter1 Capture Event
13	$0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	$001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	$001C	TIMER1 OVF	Timer/Counter1 Overflow
16	$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	$0020	TIMER0 OVF	Timer/Counter0 Overflow
18	$0022	SPI, STC	SPI Serial Transfer Complete
19	$0024	USART0, RX	USART0, Rx Complete
20	$0026	USART0, UDRE	USART0 Data Register Empty
21	$0028	USART0, TX	USART0, Tx Complete
22	$002A	ADC	ADC Conversion Complete
23	$002C	EE READY	EEPROM Ready
24	$002E	ANALOG COMP	Analog Comparator
25	$0030(3)	TIMER1 COMPC	Timer/Countre1 Compare Match C
26	$0032(3)	TIMER3 CAPT	Timer/Counter3 Capture Event
27	$0034(3)	TIMER3 COMPA	Timer/Counter3 Compare Match A

< 리셋, 인터럽트 벡터 테이블 >

Vector No.	Program Address(2)	Source	Interrupt Definition
28	$0036(3)	TIMER3 COMPB	Timer/Counter3 Compare Match B
29	$0038(3)	TIMER3 COMPC	Timer/Counter3 Compare Match C
30	$003A(3)	TIMER3 OVF	Timer/Counter3 Overflow
31	$003C(3)	USART1, RX	USART1, Rx Complete
32	$003E(3)	USART1, UDRE	USART1 Data Register Empty
33	$0040(3)	USART1, TX	USART1, Tx Complete
34	$0042(3)	TWI	Two-wire Serial Interface
35	$0044(3)	SPM READY	Store Program Memory Ready

Notes:
1. When the BOOTRST fuse is programmed, the device will jump to the Boot Loader address at reset, see "Boot Loader Support – Read-While-Write Self-Programming" on page 277.
2. When the IVSEL bit in MCUCR is set, interrupt vectors will be moved to the start of the Boot Flash section. The address of each interrupt vector will then be address in this table added to the start address of the boot Flash section.
3. The Interrupts on address $0030 - $0044 do not exist in ATmega103 compatibility mode.

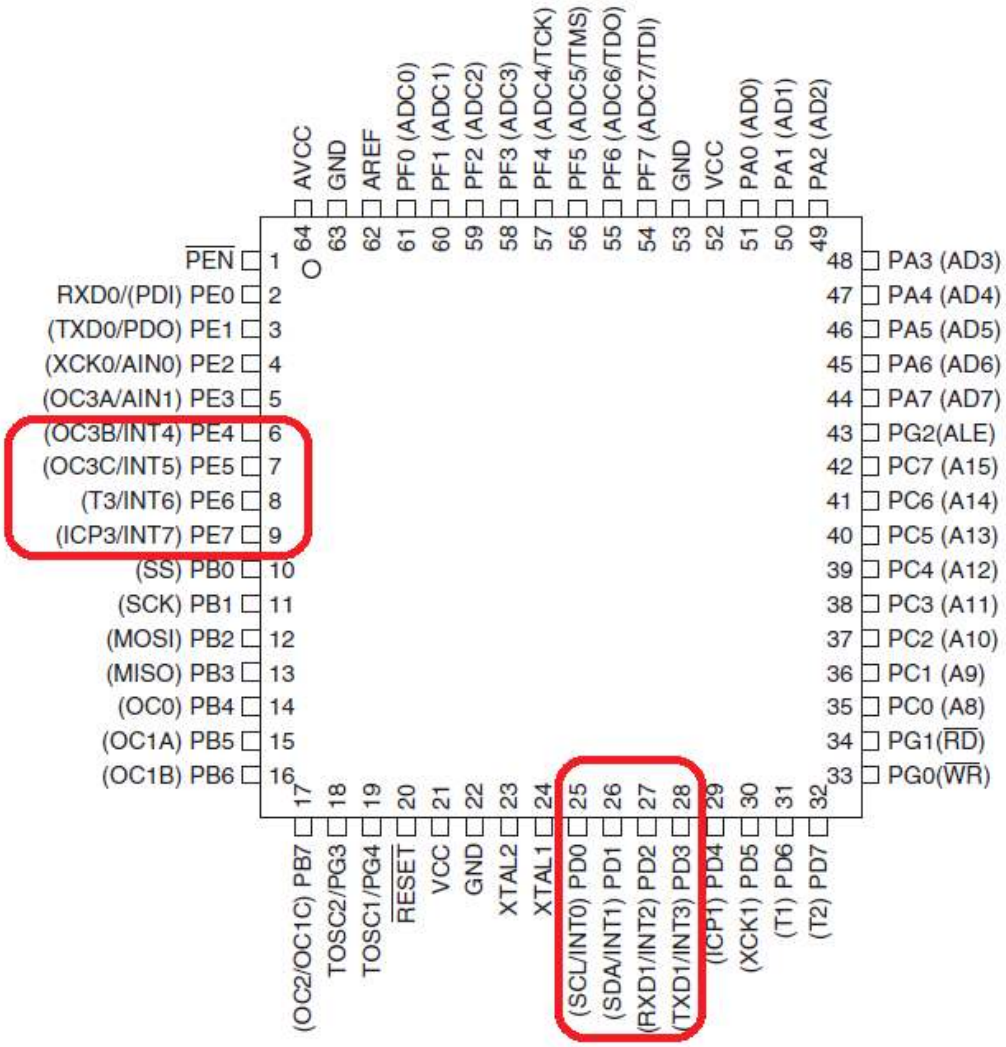
< 리셋, 인터럽트 벡터 테이블 >

4. 외부 인터럽트

4.1. 외부 인터럽트

외부 인터럽트는 외부의 상태가 변했을 때 발생합니다. 신호가 상승할 때, 신호가 하강할 때, 신호가 상승하거나 하강할 때 선택적으로 사용할 수 있습니다.

외부 인터럽트를 사용할 수 있는 핀은 INT0 ~ INT7까지 8개의 핀이며 그림으로 보면 아래와 같습니다.



4.2. 외부 인터럽트 레지스터

외부 인터럽트를 사용하려면 EICRA, EICRB, EIMSK, EIFR 레지스터를 설정해야 합니다.

- EICRA - External Interrupt Control Register A
- EICRB - External Interrupt Control Register B
  EICRA, EICRB 는 트리거 방식을 설정합니다.

Bit	7	6	5	4	3	2	1	0	
	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

설정 값은 2 바이트씩 묶이는데 기능은 다음과 같습니다.

ISCn1	ISCn0	기능
0	0	Low level에서 인터럽트 요청
0	1	없음
1	0	하강 에지에서 인터럽트 요청
1	1	상승 에지에서 인터럽트 요청

• **EIMSK** - External Interrupt Mask Register
EIMSK 는 외부 인터럽트를 사용할 것인지 아닌지를 설정하는 레지스터입니다.

Bit	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	IINT0	EIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• **EIFR** - External Interrupt Flag Register
EIFR 는 외부 인터럽트 플래그 레지스터로 외부인터럽트 발생여부를 나타냅니다.

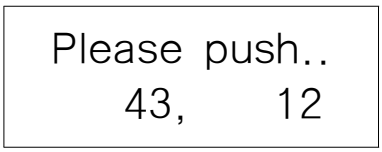
Bit	7	6	5	4	3	2	1	0	
	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	IINTF0	EIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• **SREG** - Status Register
SREG 의 bit7 로 전체 인터럽트를 사용할 것인지 아닌지를 결정합니다.

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

연습 4-1

WAT-CCLD 모듈을 사용하여 캐릭터 LCD 에 SW1, SW2 버튼의 눌림 횟수를 카운터 하는 예제입니다. (falling edge 사용)



코드 4-1

```
#include <stdio.h>

#include <avr/io.h>
#include "WAT128.h"

// 카운터용변수
int g_iSW1Count = 0;
int g_iSW2Count = 0;

// 출력용임시변수
char g_Temp[20];

int main()
{
    EICRA = 0x22;  // 7,6,4 번falling edge 0010 0010
    EIMSK = 0x05;     //   0000 0101
    sei(); // 전체인터럽트

    CLCD_Init();                // LCD 초기화
    CLCD_PutString(0,0,"Please push..");

    // 수시로카운터출력
    while(1)
    {
        sprintf(g_Temp,"   %4d,    %4d",g_iSW1Count,g_iSW2Count);
        CLCD_PutString(0,1,g_Temp);

    }
}

// INT0 걸리면
ISR(INT0_vect)
{
    g_iSW1Count++; // g_iSW1Count 증가
}

// INT2 걸리면
ISR(INT2_vect)
{
    g_iSW2Count++;      // g_iSW2Count 증가
}
```

5. Timer/Counter

ATMEGA128에는 2개의 8비트 타이머/카운터와 2개의 16비트 타이머/카운터가 있습니다. 타이머와 카운터는 펄스를 입력받는 장치인데 펄스 입력을 어디에서 받는가에 따라 구분됩니다. MCU의 메인클럭을 입력으로 사용하면 타이머, 외부 카운터 핀을 입력으로 사용하면 카운터가 됩니다.

타이머/카운터로 인터럽트를 발생하거나 PWM을 발생하여 여러 가지 시스템에 응용할 수 있습니다.

	Timer/Counter 0	Timer/Counter 2	Timer/Counter 1	Timer/Counter 3
비트	8		16	
카운터 입력 핀	TOSC1	T2	T1, IC1	T3, IC3
프리스케일러	1, 8, 32, 64, 128, 256, 1024		1, 8, 64, 256, 1024	
관련 레지스터	TCNT0, TCCR0, OCR0, ASSR, SFIOR, TIMSK, TIFR	TCCR2, TCNT2, OCR2, SFIOR, TIMSK, TIFR	TCCR1A, TCCR1B, TCCR1C, TCNT1H, TCNT1L, OCR1AH, OCR1AL, OCR1BH, OCR1BL, OCR1CH, OCR1CL, ICR1H, ICR1L, SFIOR, TIMSK, ETIMSK, TIFR, ETIFR	TCCR3A, TCCR3B, TCCR3C, TCNT3H, TCNT3L, OCR3AH, OCR3AL, OCR3BH, OCR3BL, OCR3CH, OCR3CL, ICR3H, ECR3L, SFIOR, TIMSK, ETIMSK, TIFR, ETIFR
동작 모드	Normal, CTC, Fast PWM, Phase Correct Match		Normal, CTC, Fast PWM, Phase Correct PWM, Phase and Frequency Correct PWM	
출력 핀	OC0	OC2	OC1A, OC1B, OC1C	OC3A, OC3B, OC3C
인터럽트	Overflow, Output Compare Match		Overflow, Output Compare Match A/B/C, Imput Capture	

5.1. Timer/Counter 인터럽트

타이머/카운터는 총 14개의 인터럽트를 사용할 수 있으며 주로 Compare, Overflow, Capture 인터럽트가 사용됩니다.

10	$0012	TIMER2 COMP	Timer/Counter2 Compare Match
11	$0014	TIMER2 OVF	Timer/Counter2 Overflow
12	$0016	TIMER1 CAPT	Timer/Counter1 Capture Event
13	$0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	$001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	$001C	TIMER1 OVF	Timer/Counter1 Overflow
16	$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	$0020	TIMER0 OVF	Timer/Counter0 Overflow
25	$0030(3)	TIMER1 COMPC	Timer/Countre1 Compare Match C
26	$0032(3)	TIMER3 CAPT	Timer/Counter3 Capture Event
27	$0034(3)	TIMER3 COMPA	Timer/Counter3 Compare Match A
28	$0036(3)	TIMER3 COMPB	Timer/Counter3 Compare Match B
29	$0038(3)	TIMER3 COMPC	Timer/Counter3 Compare Match C
30	$003A(3)	TIMER3 OVF	Timer/Counter3 Overflow

5.2. Timer/Counter 0

타이머/카운터0 는 8비트이며 ASSR, TCCR0, TCNT0, OCR0, TIMSK 레지스터와 관련 있습니다.

- ASSR - Asynchronous Status Register

Bit	7	6	5	4	3	2	1	0
	–	–	–	–	AS0	TCN0UB	OCR0UB	TCR0UB
Read/Write	R	R	R	R	R/W	R	R	R
Initial Value	0	0	0	0	0	0	0	0

AS0가 '0' 이면 I/O 클럭을 사용하고 '1' 이면 TOSC1에 연결된 클럭을 사용합니다.

- TCCR0 - Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit 7 -- FOC0: Force Output Compare
Bit 6, 3 -- WGM01:00 Waveform Generation Mode

Table 14-2. Waveform Generation Mode Bit Description

Mode	WGM01[1] (CTC0)	WGM00[1] (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0 at	TOV0 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

Note: 1. The CTC0 and PWM0 bit definition names are now obsolete. Use the WGM01:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

Bit 5:4 -- COM01:00 Compare Match Output Mode

Table 14-3. Compare Output Mode, non-PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

Table 14-4. Compare Output Mode, Fast PWM Mode(1)

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at BOTTOM, (non-inverting mode)
1	1	Set OC0 on compare match, clear OC0 at BOTTOM, (inverting mode)

Note: 1. A special case occurs when OCR0 equals TOP and COM01 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 99 for more details.

Table 14-5 shows the COM01:0 bit functionality when the WGM01:0 bits are set to phase correct PWM mode.

Table 14-5. Compare Output Mode, Phase Correct PWM Mode(1)

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match when up-counting. Set OC0 on compare match when downcounting.
1	1	Set OC0 on compare match when up-counting. Clear OC0 on compare match when downcounting.

Note: 1. A special case occurs when OCR0 equals TOP and COM01 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 101 for more details.

- **Bit 2:0 – CS02:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter, see Table 14-6.

Bit 2:0 -- CS02:00 Clock Select

Table 14-6. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk_{T0S}/(No prescaling)
0	1	0	clk_{T0S}/8 (From prescaler)
0	1	1	clk_{T0S}/32 (From prescaler)
1	0	0	clk_{T0S}/64 (From prescaler)
1	0	1	clk_{T0S}/128 (From prescaler)
1	1	0	clk_{T0S}/256 (From prescaler)
1	1	1	clk_{T0S}/1024 (From prescaler)

- TCNT0 - Timer/Counter Register

Bit	7	6	5	4	3	2	1	0
	TCNT0[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

8비트 타이머/카운터 레지스터

- OCR0 - Output Compare Register

Bit	7	6	5	4	3	2	1	0
	OCR0[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

8비트 비교 값

- TIMSK - Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 1 -- OCIE0: Timer/Counter0 Output Compare Match Interrupt Enable
- Bit 0 -- TOIE0: Timer/Counter0 Overflow Interrupt Enable

5.3. Timer/Counter 0 Mode

타이머/카운터0 동작모드는 TCCR0 레지스터의 WGM01~00에서 결정하며 OC0 핀에 어떤 파형을 출력할지 결정할 수 있습니다.

Table 14-2. Waveform Generation Mode Bit Description

Mode	WGM01[1] (CTC0)	WGM00[1] (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0 at	TOV0 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

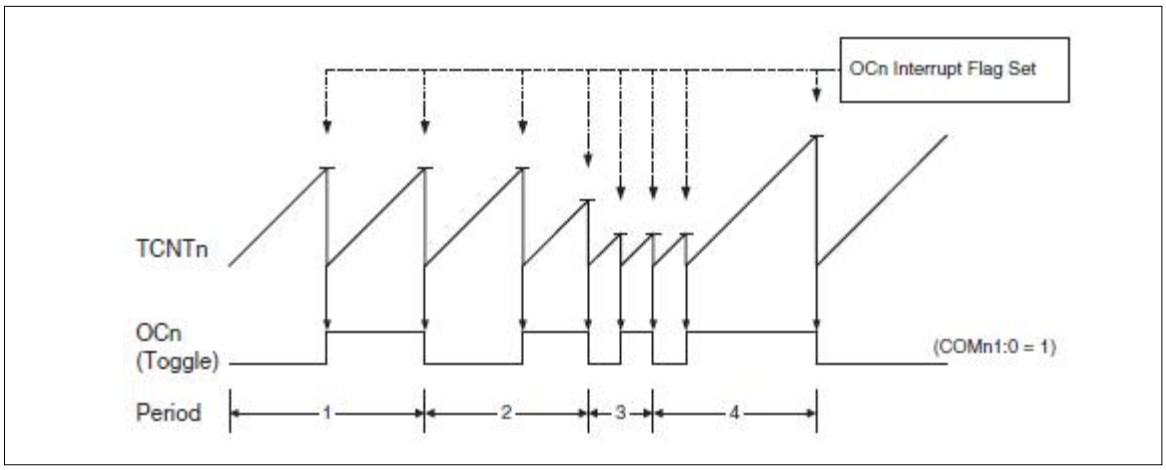
Note: 1. The CTC0 and PWM0 bit definition names are now obsolete. Use the WGM01:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

• Normal Mode

- WGM01:00 == 00 일 때 적용
- TCNT0 값이 0 ~ 0xFF 까지 반복적으로 증가
- TCNT0 값이 0xFF에서 0x00으로 될 때 Overflow 인터럽트 발생
- TCNT0와 OCR0의 값이 같아질 때 Compare 인터럽트 발생

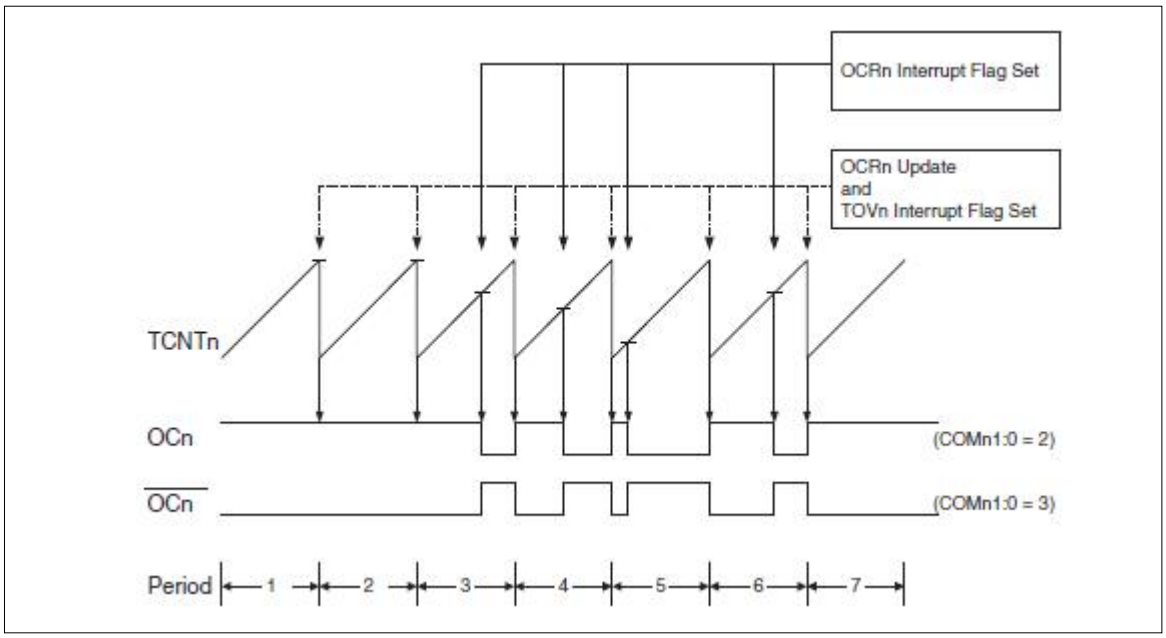
• Clear Timer on Compare Match (CTC) Mode

- WGM01:00 == 01 일 때 적용
- TCNT0 값이 0 ~ OCR0 까지 반복적으로 증가
- TCNT0와 OCR0의 값이 같아질 때 Compare 인터럽트 발생



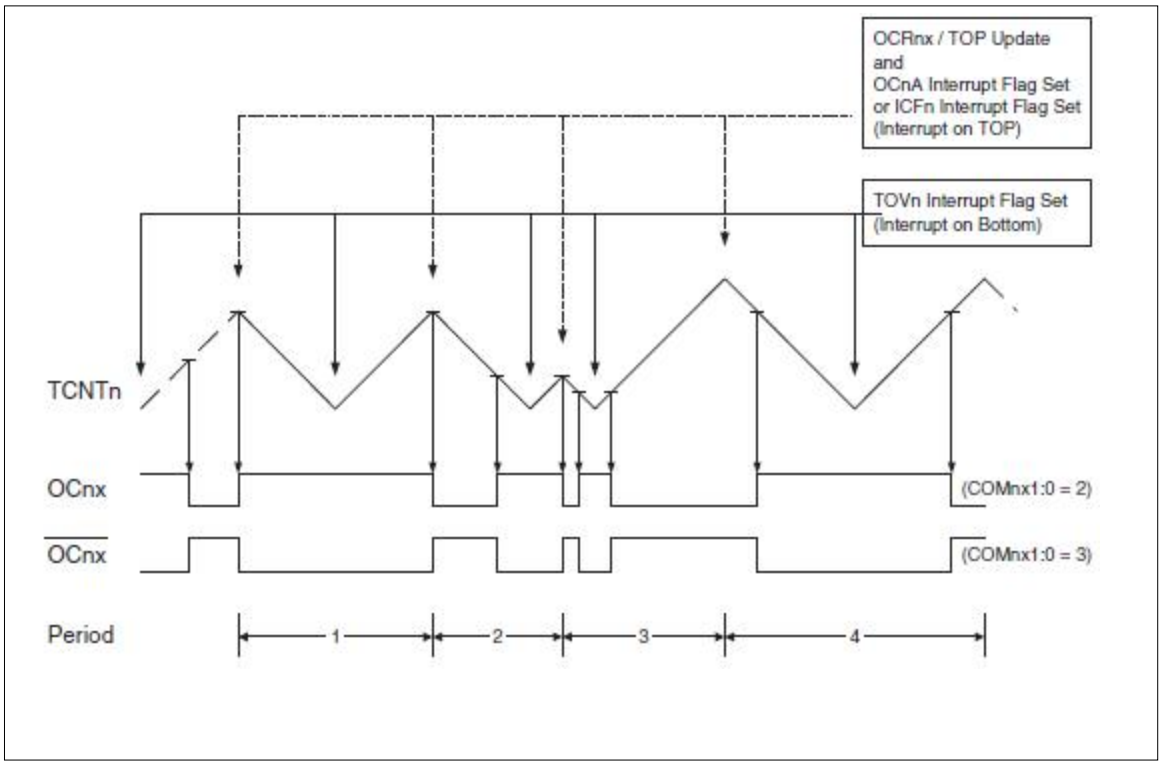
• Fast PWM Mode

- 다른 PWM 모드에 비해 2배의주파수를 갖으며 단순한 PWM 제어에 사용
- WGM01:00 =11로 설정하며 높은 주파수의 PWM 출력 파형을 발생
- 타이머/카운터 레지스터 TCNT0가 항상 BOTTOM에서 TOP의 범위에서 증가하는 방향으로만 반복적으로 수행
- TCNT0와 OCR0의 값이 일치할 경우 OC0핀으로 데이터가 출력되고 TOP값이 될 경우 다시 OC0 핀으로 데이터가 출력
- COM01:00=10 일 때
  TCNT0 == OCR0 일 경우 OC0 출력신호가 '0'
  TCNT0 == TOP 일 경우 OC0 출력신호가 '1'
- COM01:00=11 일 때
  TCNT0 == OCR0 일 경우 OC0 출력신호가 '1'
  TCNT0 == TOP 일 경우 OC0 출력신호가 '0'



• Phase Correct PWM Mode

- Fast PWM 모드에 비하여 1/2 낮은 주파수
- WGM01:00 = 01로 설정하며 높은 분해능의 PWM출력 파형을 발생에 사용
- 타이머/카운터 레지스터 TCNT0는 상향 카운터로서 BOTTOM 에서 TOP 으로 증가하였다가 다시 하향 카운터로서 BOTTOM 으로 감소하는 동작을 반복적으로 수행
- 이 동작에서 TCNT0와 OCR0의 값이 일치하면 OC0 핀을 통하여 신호를 출력 (상향 카운터와 하향 카운터의 출력신호는 반대이며 COM01에서 결정됨)
- COM01:00=10일 때,
  상향카운터 라면 TCNT0와 OCR0가 일치하면 OC00이 '0' 출력
  하향카운터 라면 TCNT0와 OCR0가 일치하면 OC00이 '1' 출력
- COM01:00=11일 때,
  상향카운터 라면 TCNT0와 OCR0가 일치하면 OC00이 '1' 출력
  하향카운터 라면 TCNT0와 OCR0가 일치하면 OC00이 '0' 출력



5.4. Timer/Counter 2

타이머/카운터2 는 8비트이며 TCCR2, TCNT2, OCR2, SFIOR, TIMSK 레지스터와 관련 있습니다.

TCCR2 - Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7 -FOC2: Force Output Compare
- Bit 6, 3 -WGM21:0: Waveform Generation Mode

Mode	WGM21 (CTC2)	WGM20 (PWM2)	Timer/Counter Mode of Operation	TOP	Update of OCR2 at	TOV2 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR2	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

- Bit 5:4 -COM21:0: Compare Match Output Mode

Table 17-3. Compare Output Mode, Non-PWM Mode

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Toggle OC2 on compare match
1	0	Clear OC2 on compare match
1	1	Set OC2 on compare match

Table 17-4 shows the COM21:0 bit functionality when the WGM21:0 bits are set to fast PWM mode.

Table 17-4. Compare Output Mode, Fast PWM Mode(1)

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Reserved
1	0	Clear OC2 on compare match, set OC2 at BOTTOM, (non-inverting mode)
1	1	Set OC2 on compare match, clear OC2 at BOTTOM, (inverting mode)

Note: 1. A special case occurs when OCR2 equals TOP and COM21 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 154 for more details.

Table 17-5. Compare Output Mode, Phase Correct PWM Mode[1]

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Reserved
1	0	Clear OC2 on compare match when up-counting. Set OC2 on compare match when downcounting.
1	1	Set OC2 on compare match when up-counting. Clear OC2 on compare match when downcounting.

Note: 1. A special case occurs when OCR2 equals TOP and COM21 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 156 for more details.

- Bit 2:0 – CS22:0: Clock Select

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T2 pin. Clock on falling edge
1	1	1	External clock source on T2 pin. Clock on rising edge

• Bit 2:0 -- CS22:20 Clock Select

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T2 pin. Clock on falling edge
1	1	1	External clock source on T2 pin. Clock on rising edge

• TCNT2 - Timer/Counter Register

Bit	7	6	5	4	3	2	1	0
	TCNT2[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

8비트 타이머/카운터 레지스터

• OCR2 - Output Compare Register

Bit	7	6	5	4	3	2	1	0	
	OCR2[7:0]								OCR2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

8비트 비교 값

• TIMSK - Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

• Bit 7 -- OCIE2: Timer/Counter2 Output Compare Match Interrupt Enable
• Bit 6 -- TOIE2: Timer/Counter2 Overflow Interrupt Enable

• TIFR - Timer/Counter Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

• Bit 7 -OCF2: Output Compare Flag 2
• Bit 6 -TOV2: Timer/Counter2 Overflow Flag

5.5. Timer/Counter 2 동작 모드

타이머/카운터2 동작모드는 TCCR2 레지스터의 WGM21~20에서 결정하며 OC2 핀에 어떤 파형을 출력할지 결정할 수 있습니다.

Mode	WGM21 (CTC2)	WGM20 (PWM2)	Timer/Counter Mode of Operation	TOP	Update of OCR2 at	TOV2 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR2	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

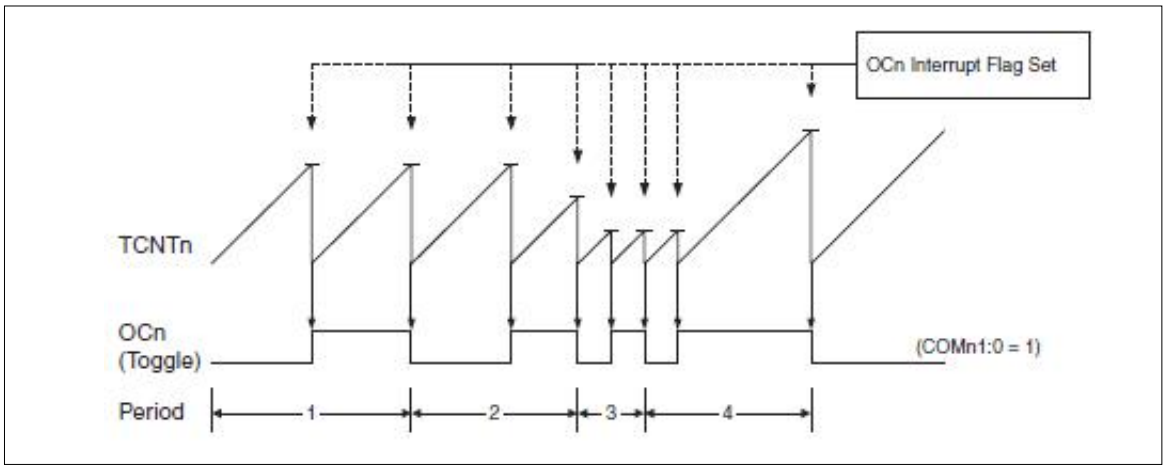
Note: The CTC2 and PWM2 bit definition names are now obsolete. Use the WGM21:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

• Normal Mode

- WGM21:20 == 00 일 때 적용
- TCNT2 값이 0 ~ 0xFF 까지 반복적으로 증가
- TCNT2 값이 0xFF에서 0x00으로 될 때 Overflow 인터럽트 발생
- TCNT2와 OCR2의 값이 같아질 때 Compare 인터럽트 발생

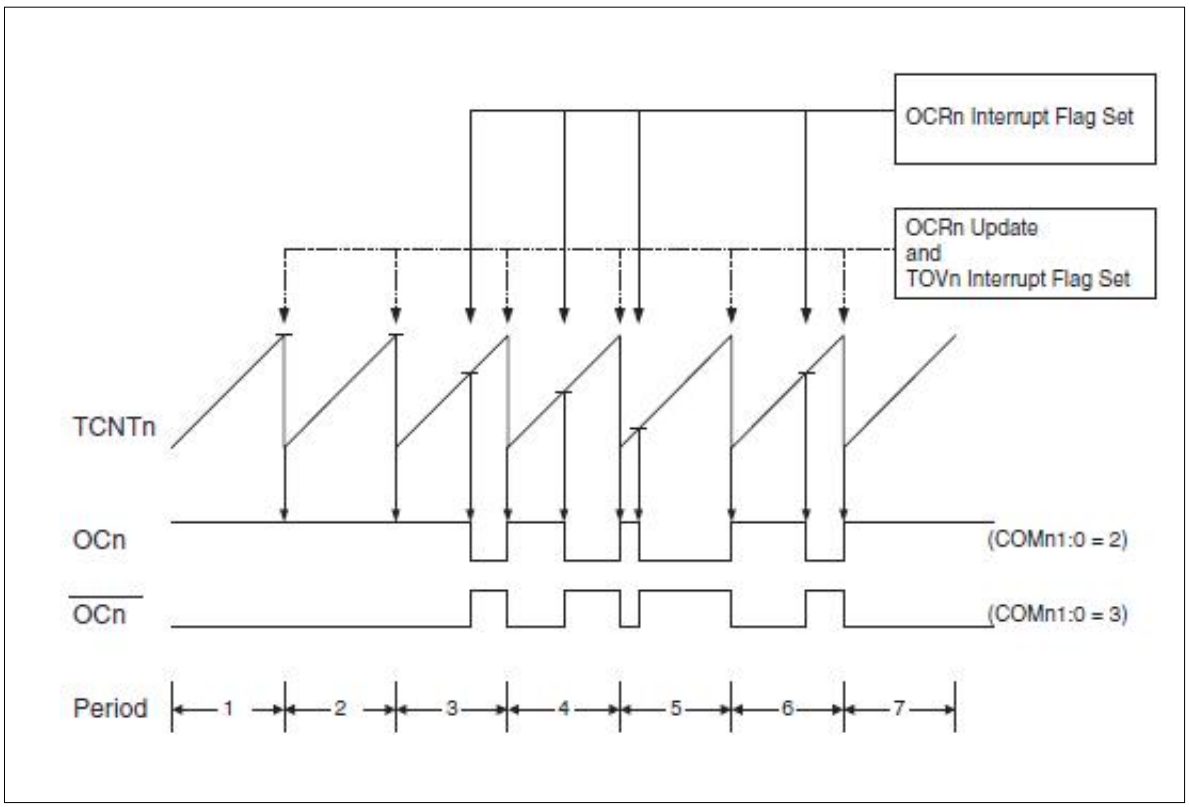
• Clear Timer on Compare Match (CTC) Mode

- WGM21:20 == 01 일 때 적용
- TCNT0 값이 0 ~ OCR0 까지 반복적으로 증가
- TCNT0와 OCR0의 값이 같아질 때 Compare 인터럽트 발생



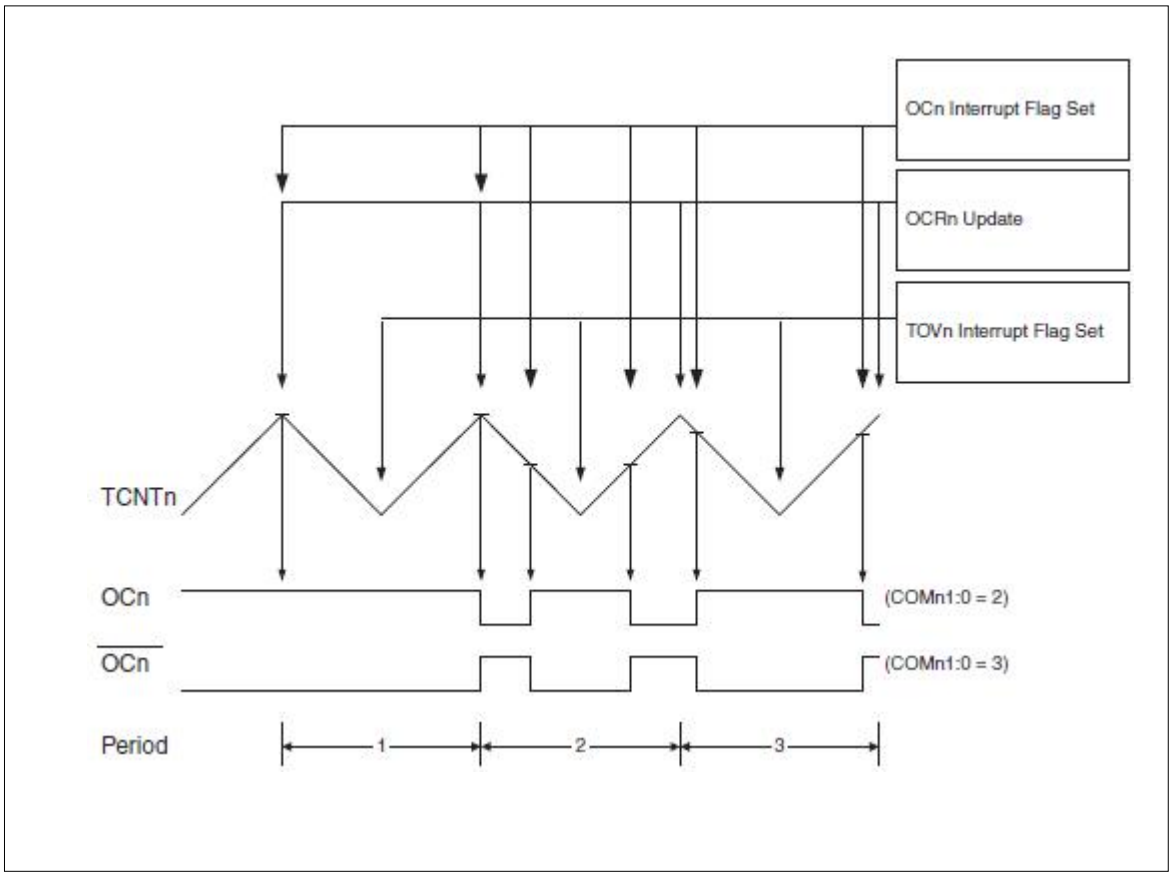
• Fast PWM Mode

- 다른 PWM 모드에 비해 2배의주파수를 갖으며 단순한 PWM 제어에 사용
- WGM21:20 =11로 설정하며 높은 주파수의 PWM 출력 파형을 발생
- 타이머/카운터 레지스터 TCNT2가 항상 BOTTOM에서 TOP의 범위에서 증가하는 방향으로만 반복적으로 수행
- TCNT2와 OCR2의 값이 일치할 경우 OC2핀으로 데이터가 출력되고 TOP값이 될 경우 다시 OC2 핀으로 데이터가 출력
- COM21:20=10 일 때
  TCNT2 == OCR2 일 경우 OC2 출력신호가 '0'
  TCNT2 == TOP 일 경우 OC2 출력신호가 '1'
- COM21:20=11 일 때
  TCNT2 == OCR2 일 경우 OC2 출력신호가 '1'
  TCNT2 == TOP 일 경우 OC2 출력신호가 '0'



• Phase Correct PWM Mode

- Fast PWM 모드에 비하여 1/2 낮은 주파수
- WGM21:20 = 01로 설정하며 높은 분해능의 PWM출력 파형을 발생에 사용
- 타이머/카운터 레지스터 TCNT2는 상향 카운터로서 BOTTOM 에서 TOP 으로 증가하였다가 다시 하향 카운터로서 BOTTOM 으로 감소하는 동작을 반복적으로 수행
- 이 동작에서 TCNT2와 OCR2의 값이 일치하면 OC2 핀을 통하여 신호를 출력 (상향 카운터와 하향 카운터의 출력신호는 반대이며 COM21에서 결정됨)
- COM21:20=10일 때,
  상향카운터 라면 TCNT2와 OCR2가 일치하면 OC20이 '0' 출력
  하향카운터 라면 TCNT2와 OCR2가 일치하면 OC20이 '1' 출력
- COM21:20=11일 때,
  상향카운터 라면 TCNT2와 OCR2가 일치하면 OC20이 '1' 출력
  하향카운터 라면 TCNT2와 OCR2가 일치하면 OC20이 '0' 출력



5.6. Timer/Counter 1, 3

타이머/카운터1은 16비트 이며 PWM 출력 및 캡쳐 입력을 가지고 있습니다. (OC1A, OC1B, OC1C) 타이머/카운터1도 16비트 이며 PWM 출력 및 캡쳐 입력을 가지고 있습니다. (OC3A, OC3B, OC3C) 프리스케일러에 의해 1, 8, 64, 256, 1024 중 하나를 선택하여 분주할 수도 있고 차단할 수도 있습니다. 16비트라서 0 ~ 0xFFFF 까지 카운팅됩니다.

타이머/카운터1은 SFIOR, TCCR1A, TCCR1B,TCCR1C,TCNT1H, TCNT1L, OCR1AH, OCR1AL, OCR1BH, OCR1BL, OCR1CH, OCR1CL, ICR1H, ICR1L, TIMSK, ETIMSK, TIFR, ETIFR 레지스터와 관련 있고, 타이머/카운터3은 SFIOR, TCCR3A, TCCR3B, TCCR3C, TCNT3H, TCNT3L, OCR3AH, OCR3AL, OCR3BH, OCR3BL, OCR3CH, OCR3CL, ICR3H, ECR3L,TIMSK, ETIMSK, TIFR, ETIFR 와 관련 있습니다.

SFIOR - Special Function IO Register

Bit	7	6	5	4	3	2	1	0
	TSM	–	–	–	ACME	PUD	PSR0	PSR321
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7 -- TSM: Timer/Counter Synchronization Mode
- Bit 0 -- PSR321: Prescaler Reset Timer/Counter3, Timer/Counter2, and Timer/Counter1

TCCR1A - Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TCCR3A - Timer/Counter3 Control Register A

Bit	7	6	5	4	3	2	1	0
	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7:6 -- COMnA1:0: Compare Output Mode for Channel A
- Bit 5:4 -- COMnB1:0: Compare Output Mode for Channel B

• Bit 3:2 -- COMnC1:0: Compare Output Mode for Channel C

Table 15-2. Compare Output Mode, non-PWM

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
0	1	Toggle OCnA/OCnB/OCnC on compare match.
1	0	Clear OCnA/OCnB/OCnC on compare match (set output to low level).
1	1	Set OCnA/OCnB/OCnC on compare match (set output to high level).

Table 15-3 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to the fast PWM mode

Table 15-3. Compare Output Mode, Fast PWM

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
0	1	WGMn3:0 = 15: Toggle OCnA on Compare Match, OCnB/OCnC disconnected (normal port operation). For all other WGMn settings, normal port operation, OCnA/OCnB/OCnC disconnected.
1	0	Clear OCnA/OCnB/OCnC on compare match, set OCnA/OCnB/OCnC at BOTTOM, (non-inverting mode)
1	1	Set OCnA/OCnB/OCnC on compare match, clear OCnA/OCnB/OCnC at BOTTOM, (inverting mode)

Note: A special case occurs when OCRnA/OCRnB/OCRnC equals TOP and COMnA1/COMnB1/COMnC1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 126. for more details.

Table 15-3 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to the phase correct and frequency correct PWM mode.

Table 15-4. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
0	1	WGMn3:0 = 9 or 11: Toggle OCnA on Compare Match, OCnB/OCnC disconnected (normal port operation). For all other WGMn settings, normal port operation, OCnA/OCnB/OCnC disconnected.
1	0	Clear OCnA/OCnB/OCnC on compare match when up-counting. Set OCnA/OCnB/OCnC on compare match when downcounting.
1	1	Set OCnA/OCnB/OCnC on compare match when up-counting. Clear OCnA/OCnB/OCnC on compare match when downcounting.

Note: 1. A special case occurs when OCRnA/OCRnB/OCRnC equals TOP and COMnA1/COMnB1//COMnC1 is set. See "Phase Correct PWM Mode" on page 128. for more details.

- Bit 1:0 -- WGMn1:0: Waveform Generation Mode

Table 15-5. Waveform Generation Mode Bit Description

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation(1)	TOP	Update of OCRnX at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

Note: 1. The CTCn and PWMn1:0 bit definition names are obsolete. Use the WGMn2:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

TCCR1B - Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TCCR3B - Timer/Counter3 Control Register B

Bit	7	6	5	4	3	2	1	0
	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7 -- ICNCn: Input Capture Noise Canceler
- Bit 6 -- ICESn: Input Capture Edge Select
- Bit 5 -- Reserved Bit
- Bit 4:3 -- WGMn3:2: Waveform Generation Mode
- Bit 2:0 -- CSn2:0: Clock Select

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0
	FOC1A	FOC1B	FOC1C	–	–	–	–	–
Read/Write	W	W	W	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

TCCR3C – Timer/Counter3 Control Register C

Bit	7	6	5	4	3	2	1	0
	FOC3A	FOC3B	FOC3C	–	–	–	–	–
Read/Write	W	W	W	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

- Bit 7 -- FOCnA: Force Output Compare for Channel A
- Bit 6 -- FOCnB: Force Output Compare for Channel B
- Bit 5 -- FOCnC: Force Output Compare for Channel C
- Bit 4:0 -- Reserved

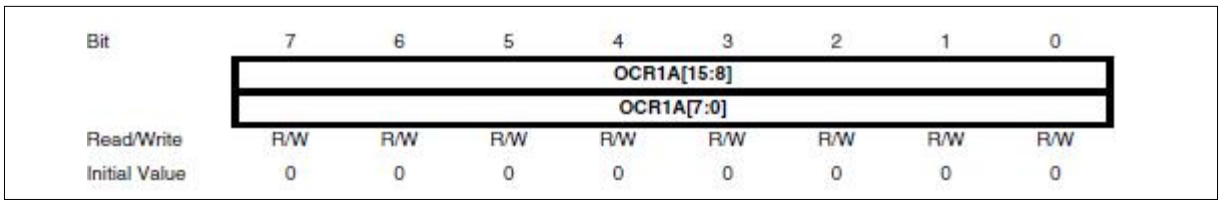
TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0
	TCNT1[15:8]							
	TCNT1[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TCNT3H and TCNT3L – Timer/Counter3

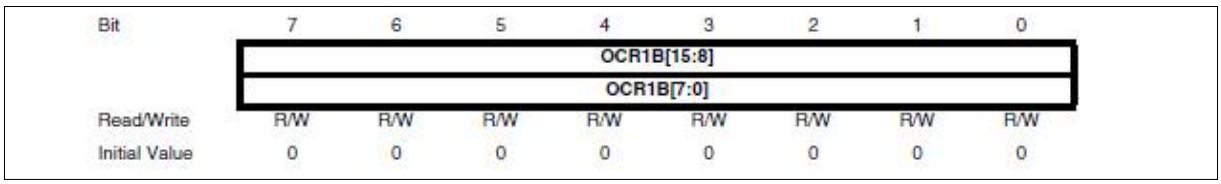
Bit	7	6	5	4	3	2	1	0
	TCNT3[15:8]							
	TCNT3[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

OCR1AH and OCR1AL – Output Compare Register 1 A



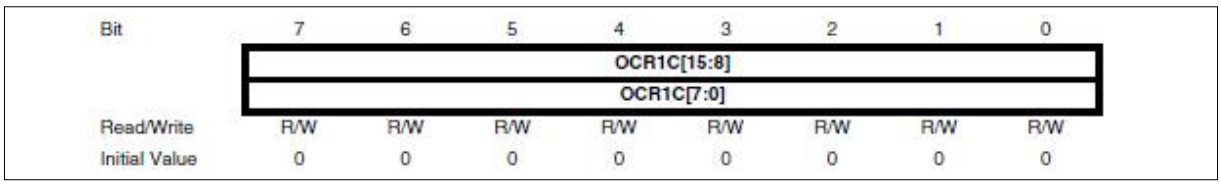
Bit	7	6	5	4	3	2	1	0
	OCR1A[15:8]							
	OCR1A[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

OCR1BH and OCR1BL – Output Compare Register 1 B



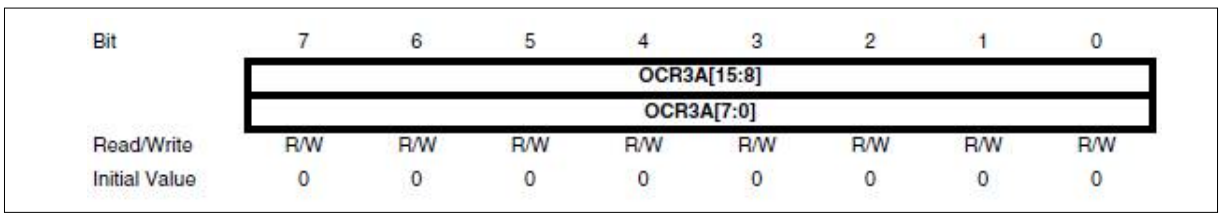
Bit	7	6	5	4	3	2	1	0
	OCR1B[15:8]							
	OCR1B[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

OCR1CH and OCR1CL – Output Compare Register 1 C



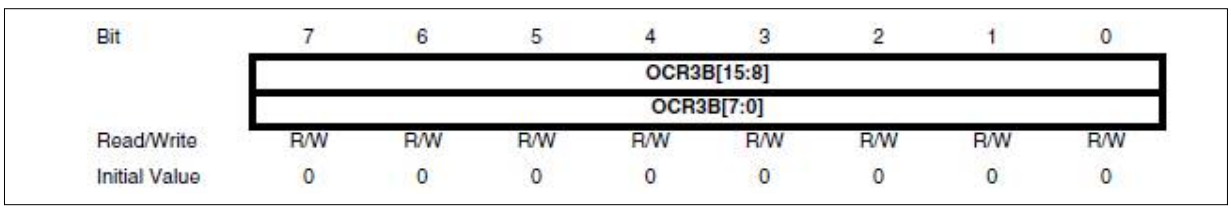
Bit	7	6	5	4	3	2	1	0
	OCR1C[15:8]							
	OCR1C[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

OCR3AH and OCR3AL – Output Compare Register 3 A



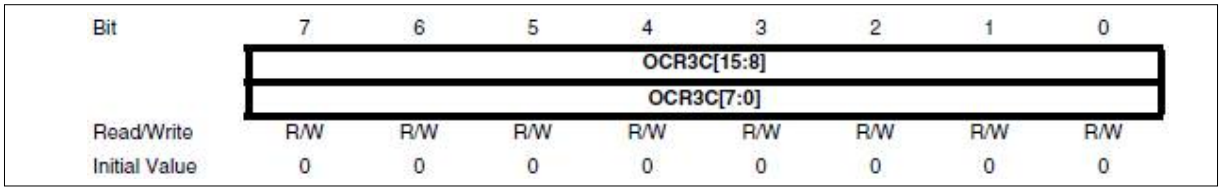
Bit	7	6	5	4	3	2	1	0
	OCR3A[15:8]							
	OCR3A[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

OCR3BH and OCR3BL – Output Compare Register 3 B



Bit	7	6	5	4	3	2	1	0
	OCR3B[15:8]							
	OCR3B[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

OCR3CH and OCR3CL – Output Compare Register 3 C



Bit	7	6	5	4	3	2	1	0
	OCR3C[15:8]							
	OCR3C[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

ICR1H and ICR1L - Input Capture Register 1

Bit	7	6	5	4	3	2	1	0
	ICR1[15:8]							
	ICR1[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

ICR3H and ICR3L - Input Capture Register 3

Bit	7	6	5	4	3	2	1	0
	ICR3[15:8]							
	ICR3[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TIMSK - Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 5 -- TICIE1: Timer/Counter1, Input Capture Interrupt Enable
- Bit 4 -- OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable
- Bit 3 -- OCIE1B: Timer/Counter1, Output Compare B Match Interrupt Enable
- Bit 2 -- TOIE1: Timer/Counter1, Overflow Interrupt Enable

ETIMSK - Extended Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
	–	–	TICIE3	OCIE3A	OCIE3B	TOIE3	OCIE3C	OCIE1C
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7:6 -- Reserved
- Bit 5 -- TICIE3: Timer/Counter3, Input Capture Interrupt Enable
- Bit 4 -- OCIE3A: Timer/Counter3, Output Compare A Match Interrupt Enable
- Bit 3 -- OCIE3B: Timer/Counter3, Output Compare B Match Interrupt Enable
- Bit 2 -- TOIE3: Timer/Counter3, Overflow Interrupt Enable
- Bit 1 -- OCIE3C: Timer/Counter3, Output Compare C Match Interrupt Enable
- Bit 0 -- OCIE1C: Timer/Counter1, Output Compare C Match Interrupt Enable

TIFR - Timer/Counter Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 5 -- ICF1: Timer/Counter1, Input Capture Flag
- Bit 4 -- OCF1A: Timer/Counter1, Output Compare A Match Flag
- Bit 3 -- OCF1B: Timer/Counter1, Output Compare B Match Flag
- Bit 2 -- TOV1: Timer/Counter1, Overflow Flag

ETIFR - Extended Timer/Counter Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
	-	-	ICF3	OCF3A	OCF3B	TOV3	OCF3C	OCF1C
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7:6 -- Reserved
- Bit 5 -- ICF3: Timer/Counter3, Input Capture Flag
- Bit 4 -- OCF3A: Timer/Counter3, Output Compare A Match Flag
- Bit 3 -- OCF3B: Timer/Counter3, Output Compare B Match Flag
- Bit 2 -- TOV3: Timer/Counter3, Overflow Flag
- Bit 1 -- OCF3C: Timer/Counter3, Output Compare C Match Flag
- Bit 0 -- OCF1C: Timer/Counter1, Output Compare C Match Flag

5.7. Timer/Counter 1,3 Mode

타이머/카운터1의 동작모드는 TCCR1A ,TCCR1B 레지스터의 WGM13~10에서 결정하며 COM1x1:0에 의해 OC1A, OC1B, OC1C 핀에 어떤 파형을 출력할지 결정할 수 있습니다.

타이머/카운터3의 동작모드는 TCCR3A ,TCCR3B 레지스터의 WGM33~30에서 결정하며 COM3x1:0에 의해 OC3A, OC3B, OC3C 핀에 어떤 파형을 출력할지 결정할 수 있습니다.

Table 15-5. Waveform Generation Mode Bit Description

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation[1]	TOP	Update of OCRnX at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

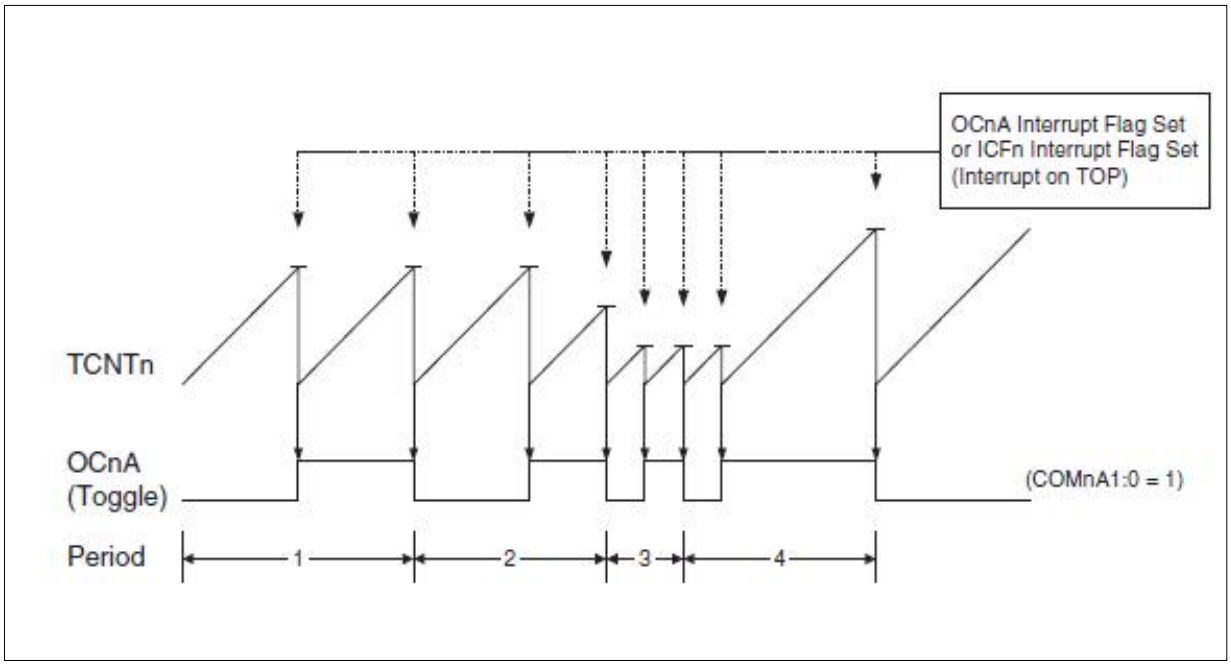
Note: 1. The CTCn and PWMn1:0 bit definition names are obsolete. Use the WGMn2:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

• Normal Mode

- WGMn3:0 == 00 일 때 적용
- TCNTn 값이 0 ~ 0xFF 까지 반복적으로 증가
- TCNTn 값이 0xFF에서 0x00으로 될 때 Overflow 인터럽트 발생
- TCNTn와 OCRn의 값이 같아질 때 Compare 인터럽트 발생

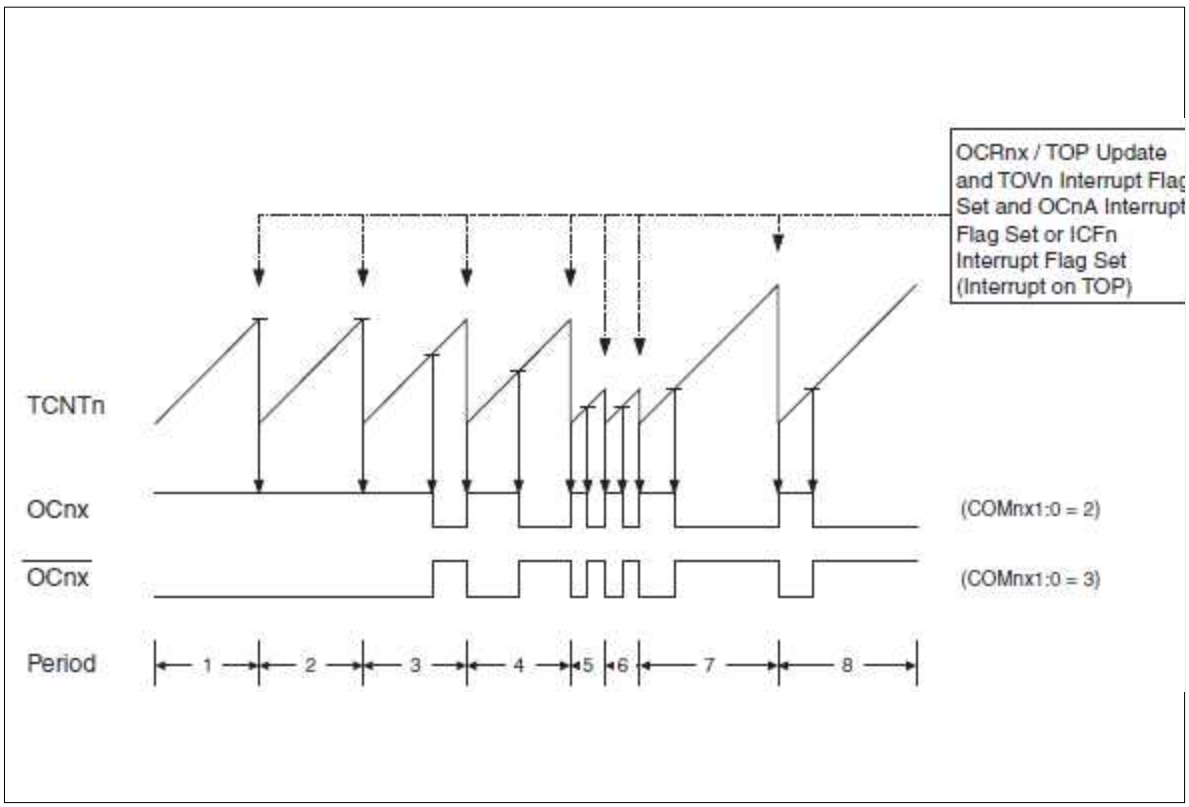
• Clear Timer on Compare Match (CTC) Mode

- WGMn3:0 == 01 일 때 적용
- TCNTn 값이 0 ~ OCRnA 까지 반복적으로 증가
- TCNTn와 OCRn의 값이 같아질 때 Compare 인터럽트 발생



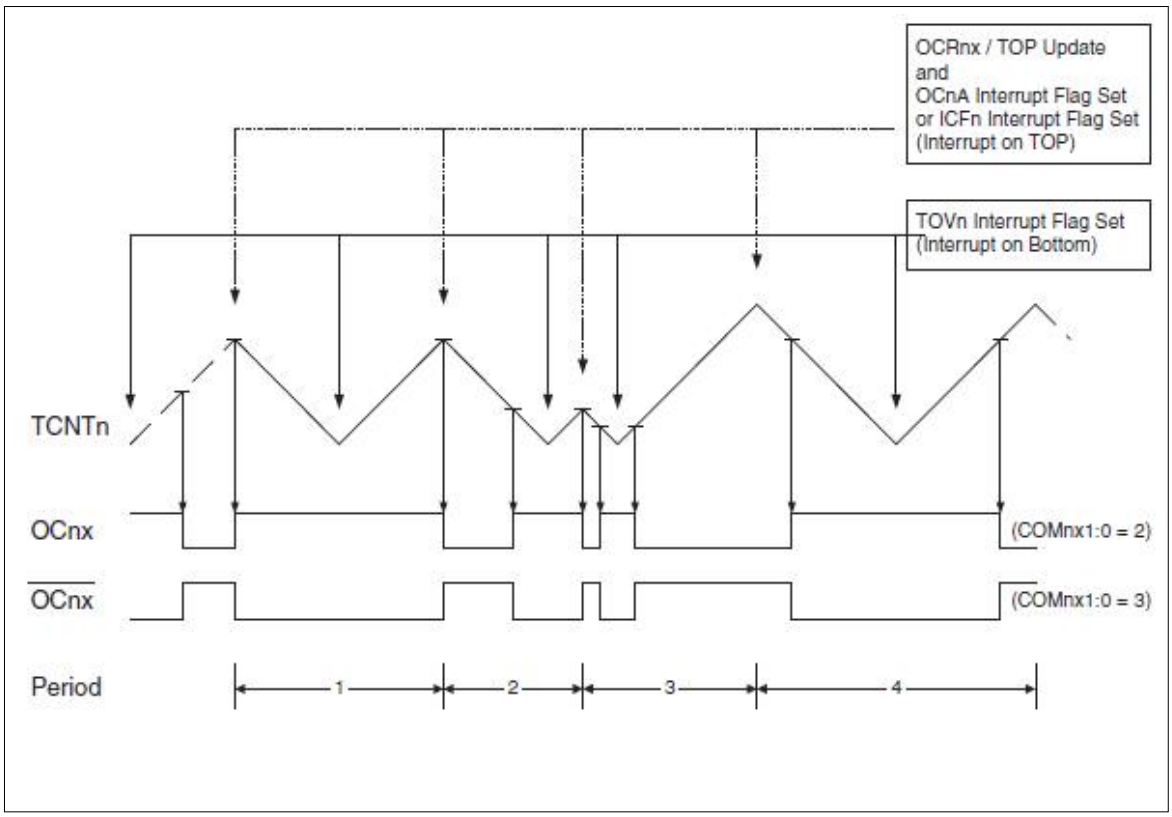
• Fast PWM Mode

- 다른 PWM 모드에 비해 2배의 주파수를 갖으며 단순한 PWM 제어에 사용
- WGMn3:0 == 5,6,7,14, 15 로 설정하며 높은 주파수의 PWM 출력 파형을 발생
- 타이머/카운터 레지스터 TCNTn가 항상 BOTTOM에서 TOP의 범위에서 증가하는 방향으로만 반복적으로 수행
- TCNTn와 OCRnx의 값이 일치할 경우 OCnx핀으로 데이터가 출력되고 TOP값이 될 경우 다시 OCnx 핀으로 데이터가 출력
- COMn1:0 == 10 일 때
  TCNTn == OCRnx 일 경우 OCnx 출력신호가 '0'
  TCNTn == TOP 일 경우 OCnx 출력신호가 '1'
- COMn1:0 == 11 일 때
  TCNTn == OCRnx 일 경우 OCnx 출력신호가 '1'
  TCNTn == TOP 일 경우 OCnx 출력신호가 '0'



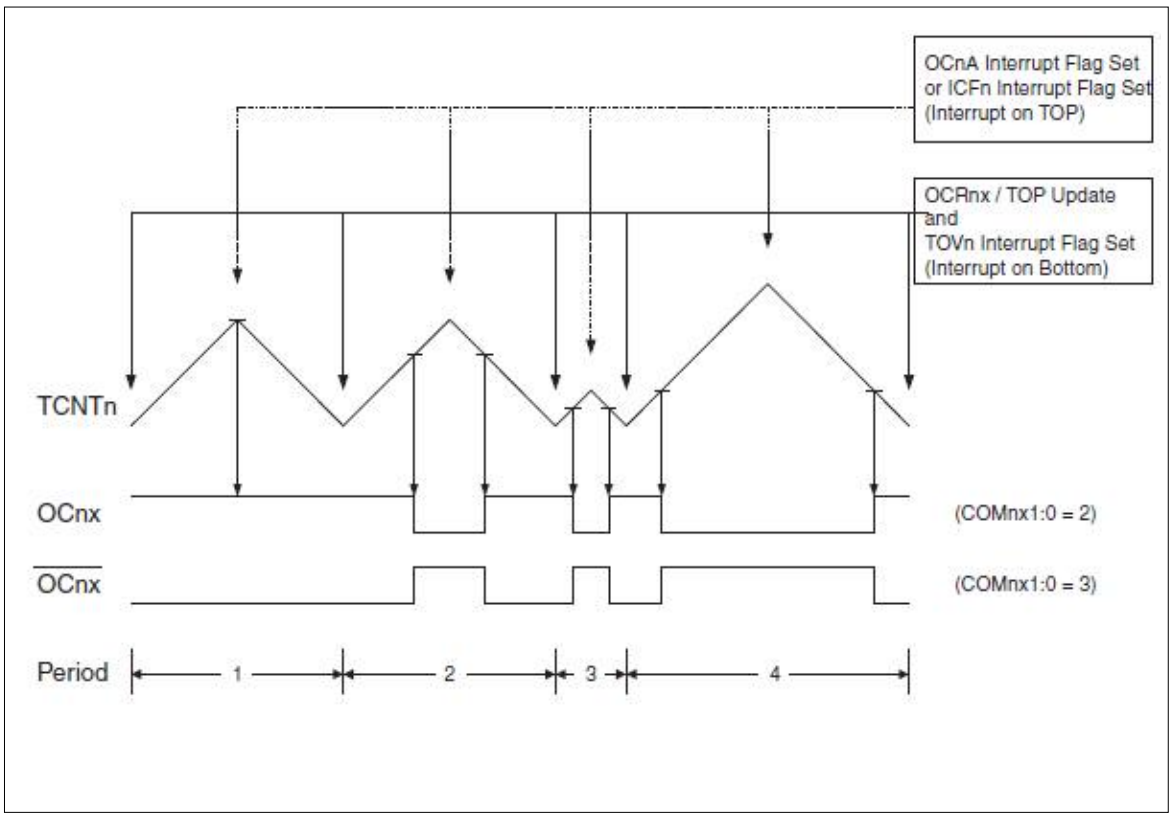
• Phase Correct PWM Mode

- Fast PWM 모드에 비하여 1/2 낮은 주파수
- WGMn3:0 == 1,2,3,10,11 로 설정하며 높은 주파수의 PWM 출력 파형을 발생
- 타이머/카운터 레지스터 TCNTn는 상향 카운터로서 BOTTOM 에서 TOP 으로 증가하였다가 다시 하향 카운터로서 BOTTOM 으로 감소하는 동작을 반복적으로 수행
- 이 동작에서 TCNTn와 OCRnx의 값이 일치하면 OCnx 핀을 통하여 신호를 출력 (상향 카운터와 하향 카운터의 출력신호는 반대이며 COMnx에서 결정됨)
- COMnx1:0 == 10일 때,
  상향카운터 라면 TCNTn와 OCRnA가 일치하면 OCnx 이 '0' 출력
  하향카운터 라면 TCNTn와 OCRnA가 일치하면 OCnx 이 '1' 출력
- COMnx1:0 == 11일 때,
  상향카운터 라면 TCNTn와 OCRnA가 일치하면 OCnx 이 '1' 출력
  하향카운터 라면 TCNTn와 OCRnA가 일치하면 OCnx 이 '0' 출력
  (동작 모드에 따라서 TOP 값은 0x00FF, 0x01FF, 0x3FF, ICRn, OCRnA 로 지정함)



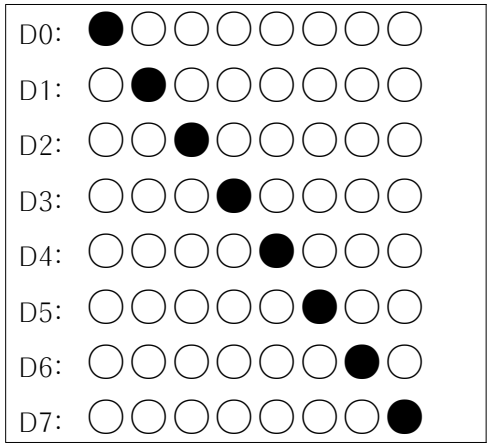
• Phase and Frequency Correct PWM Mode

- WGMn3:0 == 8,9 로 설정하며 높은 주파수의 PWM 출력 파형을 발생
- 타이머/카운터 레지스터 TCNTn 이 0x0000 ~ TOP 으로 증가하였다가 TOP ~ 0x0000으로 감소하는 동작을 반복적으로 수행
- Phase Correct PWM Mode와 거의 같지만 다른 점은 TOP으로 설정되는 레지스터 값이 갱신되는 시점이 Phase Correct PWM Mode에서는 TCNTn 이 TOP에서 갱신되고 이 모드에서는 0x0000에서 갱신
- COMnx1:0 == 10일 때,
  상향카운터 라면 TCNTn와 OCRnA가 일치하면 OCnx 이 '0' 출력
  하향카운터 라면 TCNTn와 OCRnA가 일치하면 OCnx 이 '1' 출력
- COMnx1:0 == 11일 때,
  상향카운터 라면 TCNTn와 OCRnA가 일치하면 OCnx 이 '1' 출력
  하향카운터 라면 TCNTn와 OCRnA가 일치하면 OCnx 이 '0' 출력
  (동작 모드에 따라서 TOP 값은 ICRn, OCRnA 로 지정함)



연습 5-1

delay 함수를 사용하지 않고 Timer0 Compare 인터럽트를 이용하여 PORTE에 연결된 LED을 시프트하는 예제입니다. 약 1초 간격으로 이동하며 마지막 LED 가 ON 되면 다시 처음 LED부터 ON됩니다.



코드 5-1

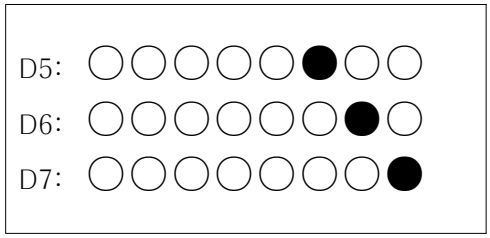
```
/*
        EX_05_01.c

  PORTE에 연결된 LED 를 시프트하는 예제입니다.
  delay 함수를 사용하지 않고 타이머 compare 인터럽트를 사용합니다.

  Main Clock : 11.0592Mhz


        AVRStudio  4.18
        2012-09-08

*/

#include <stdio.h>

#include <avr/io.h>
#include <avr/interrupt.h>
#include "WAT128.h"


#define LED_PORT   PORTE

// 1초계산용
unsigned int g_Timer = 0;
unsigned int g_TimeTick = 0; // 1초경과횟수


// 타이머compare 인터럽트루틴
// 1ms 마다실행됨
ISR(TIMER0_COMP_vect)
{
  TCNT0 =0x00;
  g_Timer++;
```

```
    if(g_Timer>=1000)
    {
        g_TimeTick ++;
        g_Timer = 0;

    }
}

// 1ms 마다실행되는인터럽트
void InitTimer0()
{
    ASSR   = 0x00;
    TCCR0 = 0x04;                    // MAX -> BOTTOM 일때플래그셋

    TCNT0 = 0x00;
    OCR0   = 171;                    // Match Register
}


int main()
{
    InitTimer0();
    // LED 출력용으로설정
    DDRE = 0xFF;
    LED_PORT = 0xFE;

    // 타이머인터럽트
    TIMSK=0x02;

    // 전체인터럽트인에이블
    sei();

    while(1)
    {
        if(g_TimeTick >=1)
        {
            // 1초가경과되었다.

            if(LED_PORT == 0x7F)      // 마지막LED 가ON 되었다면.
            {
                // 처음LED ON 되게
                LED_PORT = 0xFE;       // PORTA.0      LED ON
            }
            else                                 // 그렇지않다면
            {
                LED_PORT <<= 1;          // 한칸이동한후
                LED_PORT |= 1;           // 마지막LED 는OFF 되게
            }

            g_TimeTick   = 0;
        }
    }
}
```

연습 5-2

Timer1 을 사용하여 LED 밝기를 제어하는 예제입니다.

PB5, PB6, PB7 의 밝기를 각각 0 ~ 100%까지 증가한 후 100 ~ 0% 로 감소하는 예제입니다.



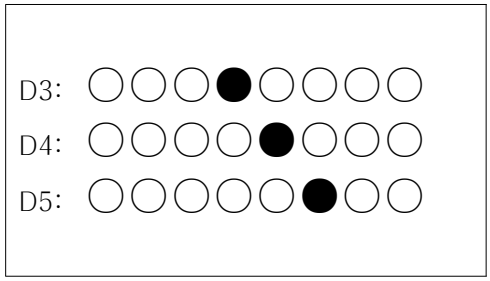
코드 5-2

```
/*
        EX_05_02.c

   TIMER1 을사용하여PORTB.5, PORTB.6, PORTB.7 에연결된LED의밝기를
  0 ~ 100, 100 ~ 0 까지제어하는예제입니다.


  Main Clock : 11.0592Mhz


        AVRStudio  4.18
        2012-12-20

*/

#include <stdio.h>

#include <avr/io.h>
#include <avr/interrupt.h>
#include "WAT128.h"


int main()
{

  SFIOR = 0;
```

```
    DDRB = (1<<PB7 |1<<PB6 |1<<PB5 );   // PB7, PB6, PB5
    TCCR1A=0xAA; // MODE=10,      FAST PWM
    TCCR1B=0x19; // 1분주/ 분주안함
    ICR1=100;    // 0 ~ 100 까지의값을사용

    OCR1A =100;
    OCR1B =100;
    OCR1C = 100;

    while(1)
    {
      OCR1A -=1;
      OCR1B -=2;
      OCR1C -=4;

      if(OCR1A>100)
        OCR1A =100;
      if(OCR1B>100)
        OCR1B =100;
      if(OCR1C>100)
        OCR1C =100;


      DelayMS(20);
    }
}
```

연습 5-3

TIMER3 을사용하여PORTE.3, PORTE.4, PORTE.5 에 연결된 LED의 밝기를 0 ~ 100, 100 ~ 0 까지 제어하는 예제입니다.



코드 5-3

```
/*
        EX_05_03.c

   TIMER3 을사용하여PORTE.3, PORTE.4, PORTE.5 에 연결된 LED의 밝기를
  0 ~ 100, 100 ~ 0 까지 제어하는 예제입니다.


  Main Clock : 11.0592Mhz

        AVRStudio  4.18
        2012-12-20

*/

#include <stdio.h>

#include <avr/io.h>
#include <avr/interrupt.h>
#include "WAT128.h"



int main()
{
```

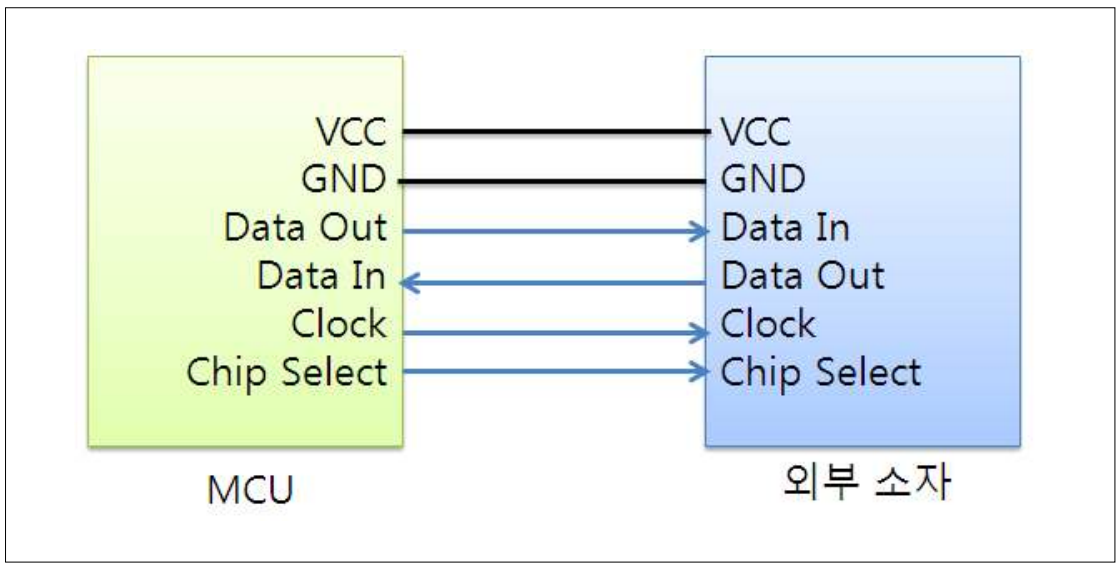
```
  SFIOR = 0;

  DDRE = (1<<PE3 |1<<PE4 |1<<PE5 );
  TCCR3A=0xAA; // MODE=10,      FAST PWM
  TCCR3B=0x1A; // 8분주
  ICR3=100;    // 0 ~ 100 까지의값을사용

  OCR3A =100;
  OCR3B =100;
  OCR3C = 100;

  while(1)
  {
    OCR3A -=1;
    OCR3B -=2;
    OCR3C -=4;

    if(OCR3A>100)
      OCR3A =100;
    if(OCR3B>100)
      OCR3B =100;
    if(OCR3C>100)
      OCR3C =100;


    DelayMS(20);
  }
}
```

6. Serial Peripheral Interface - SPI

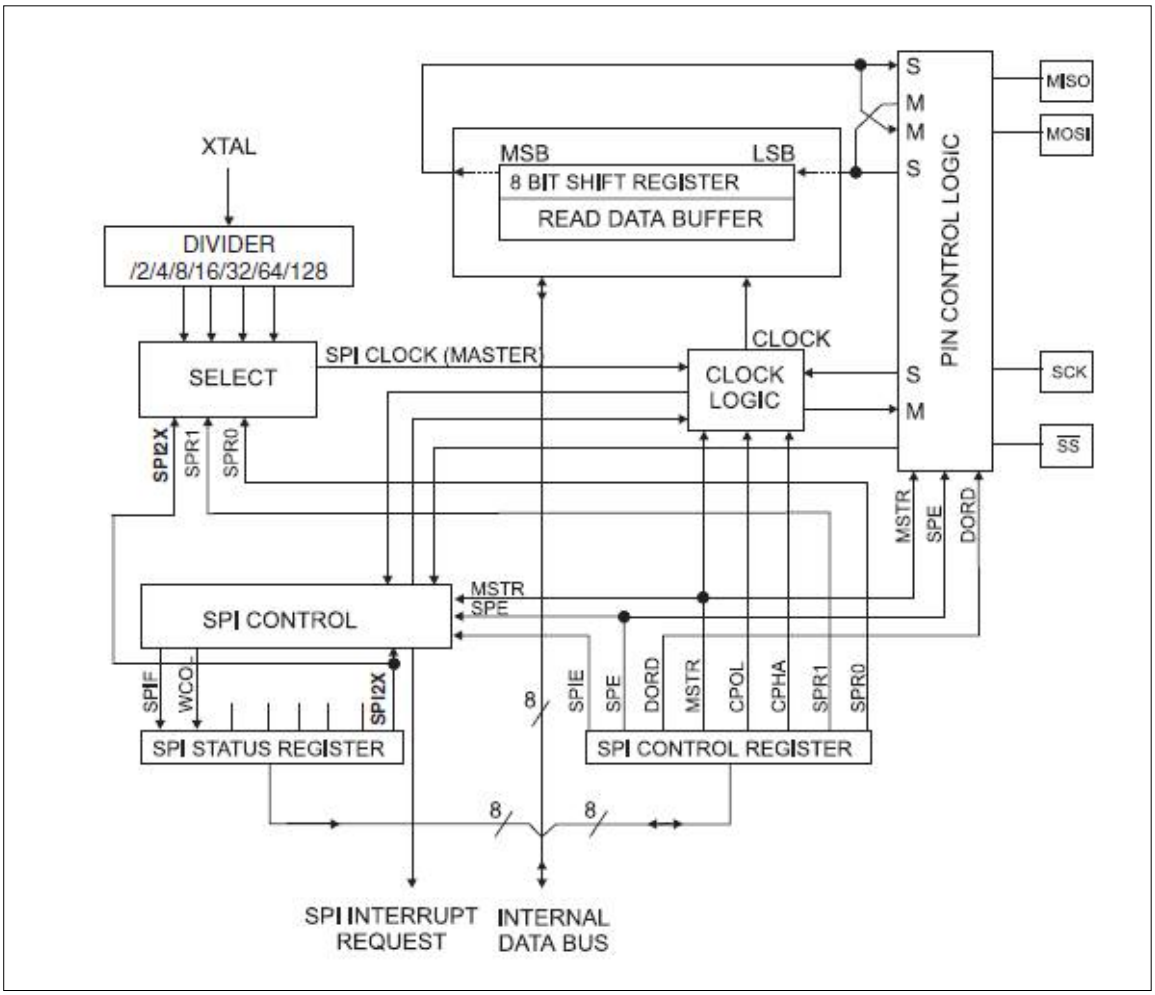
6.1. SPI

SPI는 MCU와 외부 소자 간에 직렬 데이터 통신 규격이며 모토롤라사에서 개발되었습니다. 주로 A/D 변환, D/A 변환 소자, EEPROM, Touch Controller 소자와의 통신에 많이 사용됩니다.

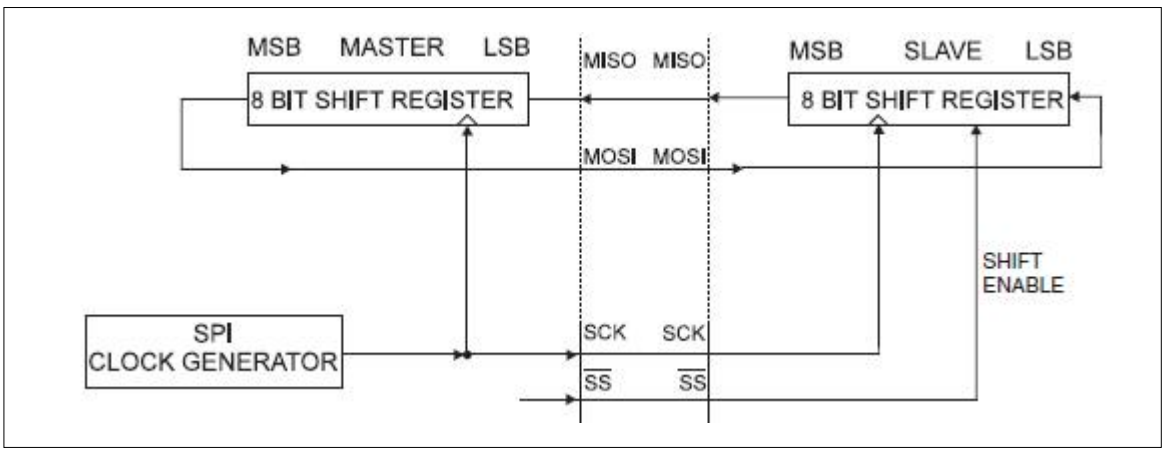
DataIn, DataOut, ChipSelect, Clock 의 4개의 컨트롤 신호와 VCC, GND 만으로 통신이 가능합니다.



< SPI 구조 >



< ATMEGA128 SPI Block Diagram >



< SPI Master-Slave 연결 >

SPI는 SPCR, SPSR, SPDR 레지스터와 관련 있습니다.

• SPCR - SPI Control Register

Bit	7	6	5	4	3	2	1	0
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7 -- SPIE: SPI Interrupt Enable
- Bit 6 -- SPE: SPI Enable
- Bit 5 -- DORD: Data Order
- Bit 4 -- MSTR: Master/Slave Select
- Bit 3 -- CPOL: Clock Polarity

CPOL	Leading edge	Trailing edge
0	Rising	Falling
1	Falling	Rising

- Bit 2 -- CPHA: Clock Phase

CPHA	Leading edge	Trailing edge
0	Sample	Setup
1	Setup	Sample

- Bits 1, 0 -- SPR1, SPR0: SPI Clock Rate Select 1 and 0

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$

• SPSR – SPI Status Register

Bit	7	6	5	4	3	2	1	0
	SPIF	WCOL	–	–	–	–	–	SPI2X
Read/Write	R	R	R	R	R	R	R	R/W
Initial Value	0	0	0	0	0	0	0	0

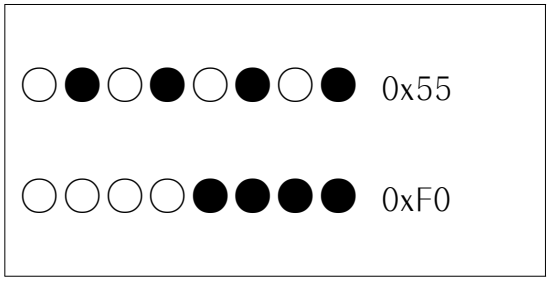
• SPDR – SPI Data Register

Bit	7	6	5	4	3	2	1	0
	MSB							LSB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	X	X	X	X	X	X	X	X

연습 6-1

WAT-AT45DB 모듈를 PORTB에 연결하여 AT45DB041 데이터 플래시에 데이터를 Write/Read 해보겠습니다.

0 ~ 0xFF 까지의 데이터를 쓰고 읽어서 전후 데이터가 동일하면 0x55를 다르면 0xF0 를 출력하는 예제입니다.



코드 6-1

```
/*
        EX_06_01.c

        SPI 로DataFlash 읽고쓰는예제입니다.

    SPI 를 초기화 하고
    데이터를 쓰고 읽은후
    쓰고 읽은 데이터가 동일한지 확인합니다.

    결과를 PORTE로 표시합니다.
        AVRStudio  4.18
        2013-03-08


*/

#include <stdio.h>

#include <avr/io.h>
#include "WAT128.h"
#include "AT45DB161D.h"


BYTE Data[256];
BYTE ReadData[256];


int main()
{
```

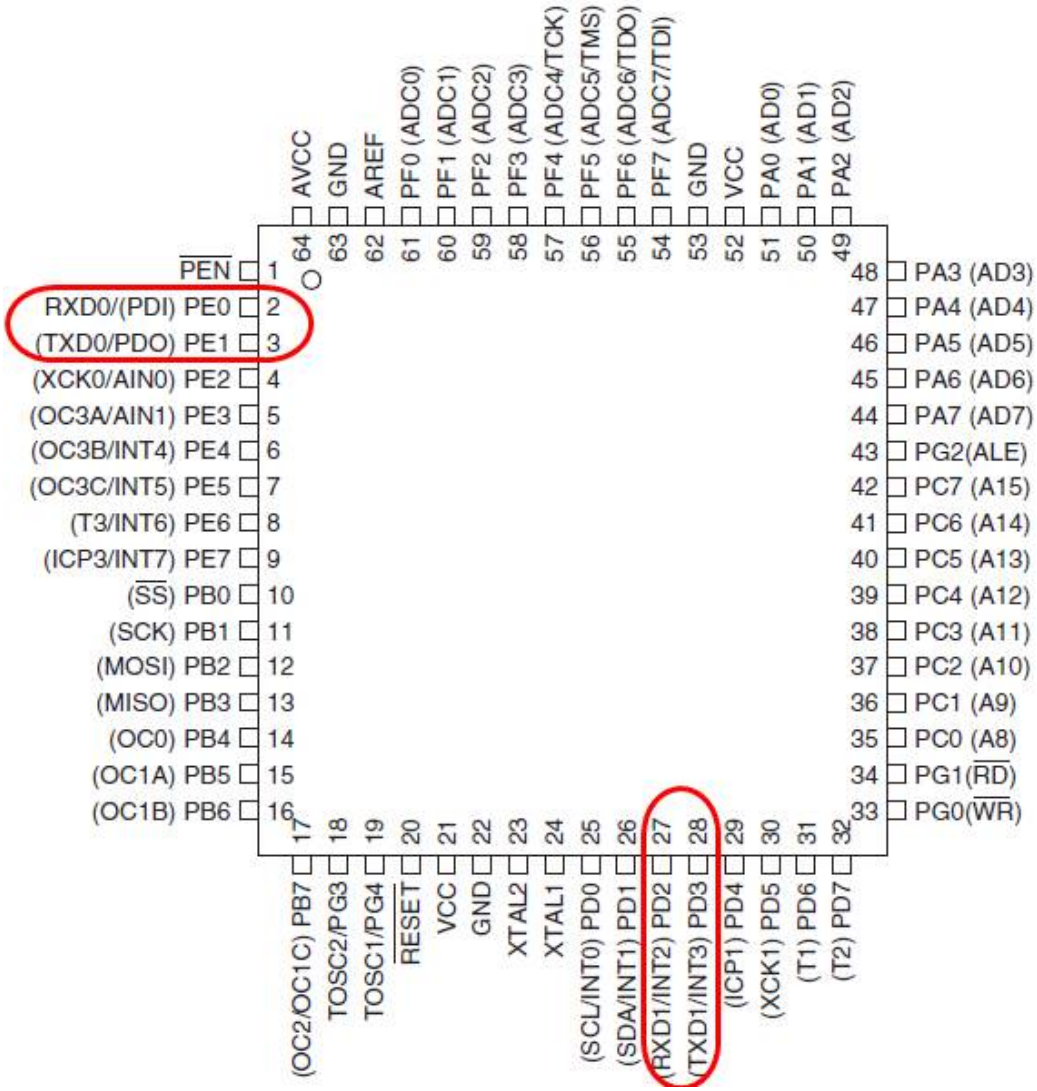
```
const BYTE BufferNum =AT45DB_BUFFER_1;
const BYTE PageNum =4;

DDRE = 0xFF;
PORTE = 0xFF;

// AT45DB 용SPI 초기화
SpiInit();
SFlash_GetID();

// 버퍼로한번써줘야한다.(중요)
PageToBuffer(1,0);

// Data 에임의의값
for(int i=0;i<256;i++)
{
  Data[i] = i;
}

// Data 값을버퍼에넣기
SFlash_WriteBuffer(BufferNum,0,Data,256);

// 버퍼값을페이지(실제저장할위치)에쓰기
SFlash_BufferToPage(BufferNum, PageNum, 1);


// 읽어올변수(ReadData)에임의의값쓰기
for(int i=0;i<256;i++)
{
  ReadData[i] = 0xff;
}

// 페이지데이터를버퍼로가져오기
PageToBuffer(BufferNum,PageNum);

// 버퍼데이터를ReadData로가져오기
GetMemBuffer(ReadData);

//
for(int i=0;i<256;i++)
{
  if(ReadData[i] != Data[i] )
  {
    while (1)
    {
      // 쓰고읽은데이터가다르다면
      PORTE = 0x55;
```

```
        }
      }
    }

    // 쓰고읽은데이터가동일하다면
    PORTE = 0xf0;;
    while(1)
    {
    }
}
```

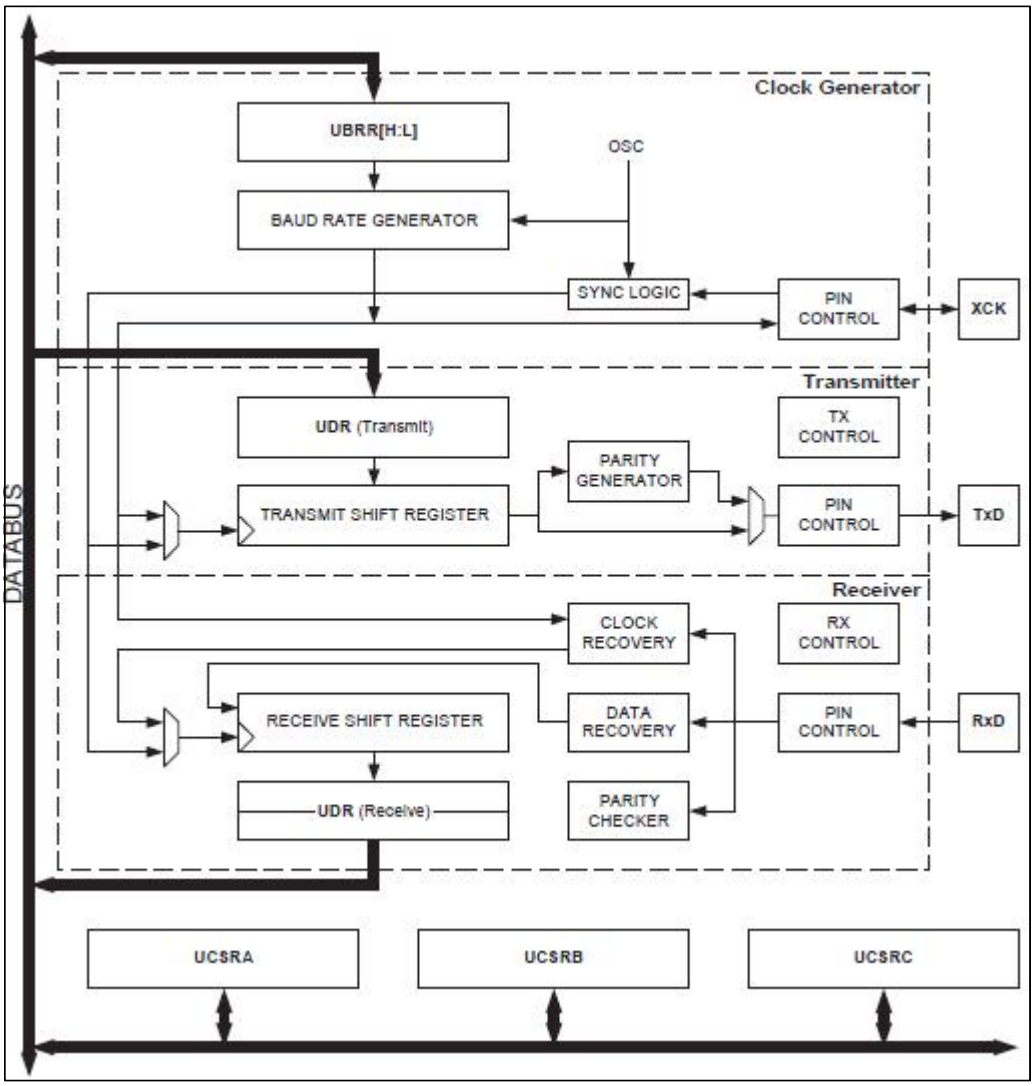
7. Universal Synchronous and Asynchronous Receiver and Transmitter - USART

7.1. USART

ATMEGA128A 에는 2개의 USART 를 내장하고 있습니다. 동기, 비동기 모드에서 전이중 통신이 가능합니다.



< USART 용 핀 >



< USART Block Diagram >

7.2. USART 레지스터

USART 사용하려면 UDR0, UCSR0A, UCSR0B, UCSR0C, UBRR0H, UBRR0L, UDR1, UCSR1A, UCSR1B, UCSR1C, UBRR1H, UBRR1L 레지스터를 설정해야 합니다.

UDRn - USARTn I/O Data Register

Bit	7	6	5	4	3	2	1	0	
	RXBn[7:0]								UDRn (Read)
	TXBn[7:0]								UDRn (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• 데이터 송신은 UDRn 를 쓰고, 수신은 UDRn 을 읽음

UCSRnA - USARTn Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

• Bit 7 – RXCn: USART Receive Complete
• Bit 6 – TXCn: USART Transmit Complete
• Bit 5 – UDREn: USART Data Register Empty
• Bit 4 – FEn: Frame Error
• Bit 3 – DORn: Data OverRun
• Bit 2 – UPEn: Parity Error
• Bit 1 – U2Xn: Double the USART Transmission Speed
• Bit 0 – MPCMn: Multi-Processor Communication Mode

UCSRnB – USARTn Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7 – RXCIEn: RX Complete Interrupt Enable
- Bit 6 – TXCIE: TX Complete Interrupt Enable
- Bit 5 – UDRIEn: USART Data Register Empty Interrupt Enable
- Bit 4 – RXENn: Receiver Enable
- Bit 3 – TXENn: Transmitter Enable
- Bit 2 – UCSZn2: Character Size
- Bit 1 – RXB8n: Receive Data Bit 8
- Bit 0 – TXB8n: Transmit Data Bit 8

UCSRnC – USARTn Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
	–	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- Bit 7 – Reserved Bit

- Bit 6 – UMSELn: USART Mode Select

Table 77. UMSELn Bit Settings

UMSELn	Mode
0	Asynchronous Operation
1	Synchronous Operation

- Bit 5:4 – UPMn1:0: Parity Mode

Table 78. UPMn Bits Settings

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	(Reserved)
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

• Bit 3 - USBSn: Stop Bit Select

Table 79. USBSn Bit Settings

USBSn	Stop Bit(s)
0	1-bit
1	2-bits

• Bit 2:1 - UCSZn1:0: Character Size

Table 80. UCSZn Bits Settings

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

• Bit 0 - UCPOLn: Clock Polarity

UCPOLn	Transmitted Data Changed (Output of TxDn Pin)	Received Data Sampled (Input on RxDn Pin)
0	Rising XCKn Edge	Falling XCKn Edge
1	Falling XCKn Edge	Rising XCKn Edge

UBRRnH, UBRRnL - USARTn Baud Rate Register

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

• Bit 15:12 - Reserved Bits

• Bit 11:0 - UBRRn11:0: USARTn Baud Rate Register

보레이트 설정 예제

Baud Rate (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max (1)	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

1. UBRR = 0, Error = 0.0%

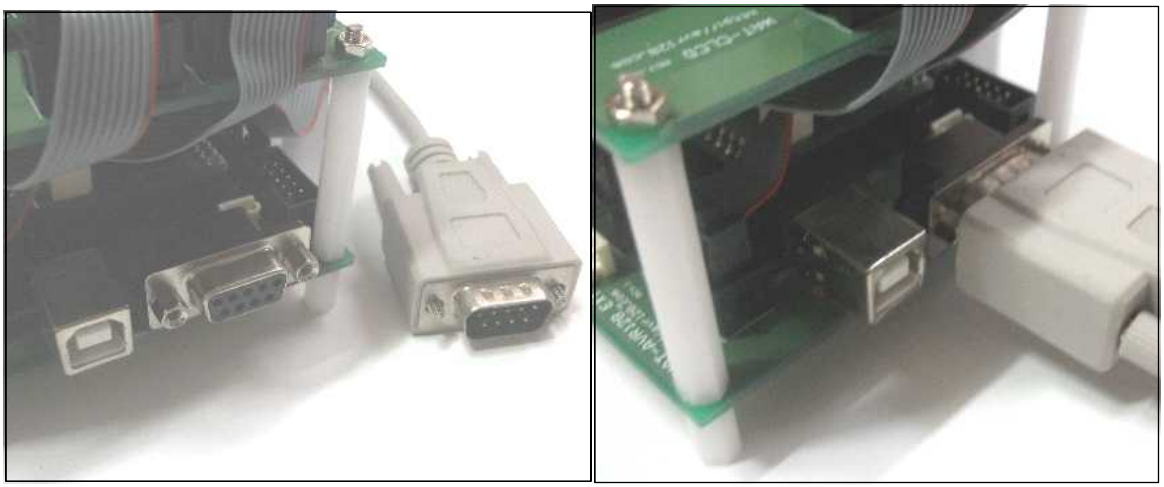
Baud Rate (bps)	f_{osc} = 16.0000 MHz				f_{osc} = 18.4320 MHz				f_{osc} = 20.0000 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max (1)	1 Mbps		2 Mbps		1.152 Mbps		2.304 Mbps		1.25 Mbps		2.5 Mbps	

1. UBRR = 0, Error = 0.0%

위의 표를 보면 알 수 있듯이 에러율을 최대한 줄이기 위해서는 11.0592Mhz를 사용하는게 좋습니다. 16Mhz의 메인클럭에서 115200bps 의 통신을 할 경우 에러률이 3.5%나 됩니다..

7.3. RS-232C 통신 실험

WAT-AVR128 실험키트로 RS-232C 통신을 해 보겠습니다. 시리얼 케이블과 실험키트를 아래와 같이 연결합니다.



WAT-AVR128 보드와 컴퓨터의 시리얼 포트를 시리얼 연장 케이블로 연결합니다. 만약 컴퓨터에 시리얼 포트가 없다면 USBtoSerial 케이블[1]로 대체할 수 있습니다.

1) USBtoSerial http://kit128.com/goods/view.php?seq=49 에서 구매 가능합니다.

연습 7-1

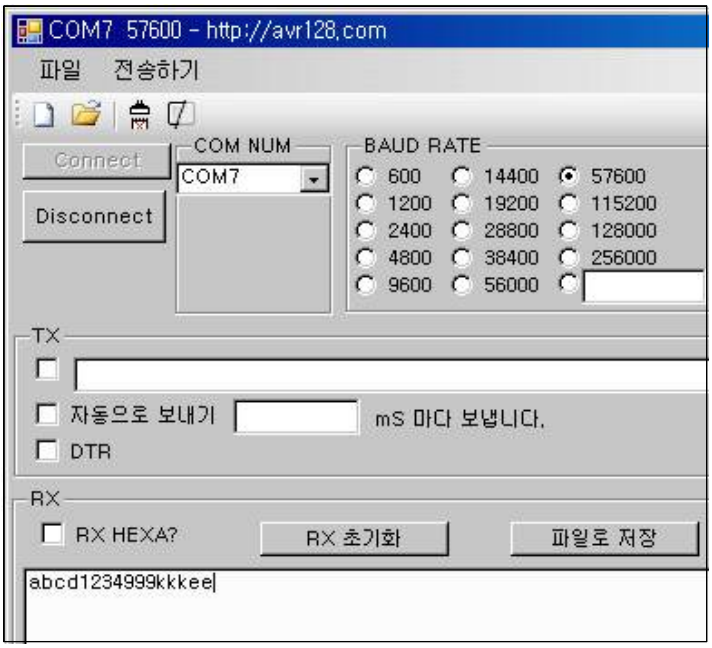
컴퓨터로 'A' ~ 'Z' 를 반복적으로 전송 예제입니다.

코드 7-1

```
/*
        EX_07_01.c

        USART0 로'A' 부터'Z' 데이터를무한히보내는예제
          보레이트: 115200 bps

        AVRStudio  4.18
        2012-06-05

*/

#include <avr/io.h>
#include "WAT128.h"

int main()
{
        char chData = 'A';

        OpenSCI0(115200);           // USART 0 열기

        while(1)
        {
                        DelayMS(300);       // 약간의딜레이

                        PutChar0(chData); // 문자보내기

                        // 'Z' 가출력되면'A'부터시작하자.
                        if(++chData> 'Z')
                                chData = 'A';

        }
}
```

연습 7-2

컴퓨터의 시리얼 통신 프로그램에서 데이터를 받아 LCD 에 출력한 후 에코를 보내는 예제입니다.



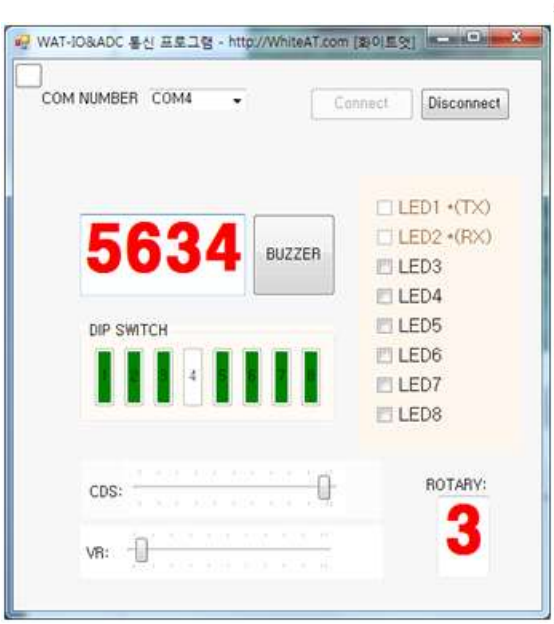
코드 7-2

```
/*
    EX_07_02.c

    컴퓨터에서 USART0 로 받은 데이터를 Character LCD에 출력
    컴퓨터로 에코 보냄
    보레이트: 57600bps
    AVRStudio  4.18
    2011-08-16

*/

#include <avr/io.h>
#include "WAT128.h"

int main()
{
```

```
    INT16S iRxData;

    OpenSCI0(57600);        // USART 0 열기

    CLCD_Init();            // LCD 초기화

    while(1)
    {
            iRxData=GetByte0();
            if( 0<= iRxData && iRxData<=255 ){

                    // CLCD 에 출력
                    CLCD_PutChar(iRxData);
                    PutChar0(iRxData); // echo 보내기
            }
    }
}
```

연습 7-3

컴퓨터에서 WAT-AVR128 보드로 FND데이터, LED 데이터 부저 상태를 전송하고 보드에서 컴퓨터로 DIP스위치 값, 로터리 스위치 값, CDS 값, 가변저항 값 등 실시간으로 전송하는 예제입니다.



< Visual C# 예제>

< Visual C++ 6.0 예제 >

코드 7-3

```
/*
        EX_09_03.c

        USART0 로 보드의 상태 PC로 전송
        PC에서 LED, BUZZER, FND 제어
        AVRStudio  4.18
        2012-06-05

        HOMEPAGE: http://WhiteAT.com
        SHOPPING: http://kit128.com

*/

#include <avr/io.h>
#include "WAT128.h"

BYTE g_FNDData[4]={1,2,3,4};
BYTE g_BUZZER = 0;
BYTE g_LED = 0;

UINT16 g_adcCDS;            // CDS 값보관
UINT16 g_adcVR;         // 가변저항값보관

void OperDisplayFND()
{
        DisplayFND4(g_FNDData[0],g_FNDData[1],g_FNDData[2],g_FNDData[3]);
}

INT16 g_byteOperPCTXTimer = 0;
void OperPCTX()
{
        if(--g_byteOperPCTXTimer>0)
                return;

        PutChar0(0x02);                                 // 0 시작신호
        PutChar0(PIND);                                 // 1 DIP SWITCH
        PutChar0(GetRotaryInt());               // 2 ROTARY
        PutChar0(g_adcCDS>>8);                  // 3 CDS 상위값
        PutChar0(g_adcCDS);                     // 4 CDS 하위값
        PutChar0((g_adcVR)>>8 &0xFF); // 5 가변저항상위값
        PutChar0(g_adcVR&0xFF);                 // 6 가변저항하위값
        PutChar0(0x03); // 7
        PutChar0(0xCC);                         // 8 체크섬
        PutChar0(0x03);                         // 9 끝신호
```

```
		g_byteOperPCTXTimer = 20;

}

// 가변저항, CDS 값을ADC로읽기
void OperReadADC()
{
		INT16 uiTemp; // 임시변수

		g_adcCDS = 0;
		// 노이즈를생각해서값을번읽어평균을낸다.

		for(uiTemp = 0; uiTemp<16;uiTemp++)
		{
			ADMUX=0x40 | 0x00;
			ADCSRA = 0xD7;
			while((ADCSRA & 0x10) != 0X10);
			g_adcCDS += ADCL + (ADCH*256);
		}

		g_adcCDS>>=4;

		g_adcVR = 0;
		// 노이즈를생각해서값을번읽어평균을낸다.

		for(uiTemp = 0; uiTemp<16;uiTemp++)
		{
			ADMUX=0x40 | 0x01;
			ADCSRA = 0xD7;
			while((ADCSRA & 0x10) != 0X10);
			g_adcVR += ADCL + (ADCH*256);
		}

		g_adcVR>>=4;
}

int main()
{

		BUZZER_INIT; // BUZZER 초기화
		OpenSCI0(115200);		// USART 0 열기

		InitFND4(); // FND 초기화

		InitADC();		// ADC 초기화

		DisplayFND4(3,4,5,6);
		InitRotary();
```

```
DDRD = 0x00; // 딥스위치를입력으로설정
PORTD = 0xFF;
DDRE = 0xFE; // LED


while(1)
{
        BUZZER_OFF;
        OperDisplayFND();
        OperReadADC();

        OperPCTX();


        if(0x02 ==GetByte0())
        {
                UINT16 uiData = 0;
                uiData = GetByte0(); //1
                uiData<<=8;
                uiData += GetByte0(); //2

                //buzzer
                g_BUZZER = GetByte0();


                g_LED = GetByte0(); //4
                GetByte0(); //5
                GetByte0(); //6
                GetByte0(); //7
                if(0xCC == GetByte0() && 0x03 == GetByte0() )
                {
                        if( g_BUZZER)
                        {
                                BUZZER_ON;
                                DelayMS(2);
                        }

                        // PC에서받은LED 값을출력
                        PORTE =~((g_LED)&0xFC);

                        // FND 표시
                        g_FNDData[0]=((uiData/1000)%10);
                        g_FNDData[1]=((uiData/100)%10);
                        g_FNDData[2]=((uiData/10)%10);
                        g_FNDData[3]=(uiData%10);
                }
```

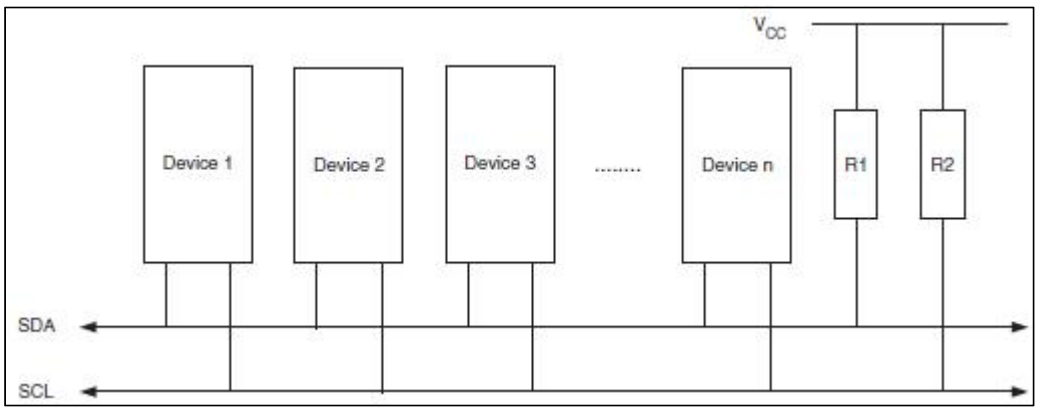
```
                    }
                }
}
```

8. Two-wire Serial Interface - I2C

8.1. TWI

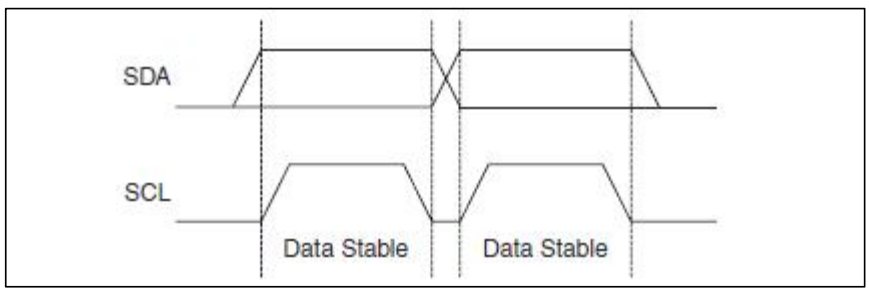
ATMEGA128A 에는 1개의 TWI 를 내장하고 있습니다. TWI는 2개의 선(Data, Clock) 으로 통신하는 방식이며 I2C 라고도 합니다.

하드웨어 구성은 1개의 마스터가 여러 개의 슬레이브 중 한 개와 통신할 수 있으며 슬레이브 전체에 같은 데이터를 송신할 수도 있습니다. SDA, SCK 은 오픈 드레인 방식으로 되어 있어 반드시 풀업 저항(약 4.7K)을 연결해야 합니다.



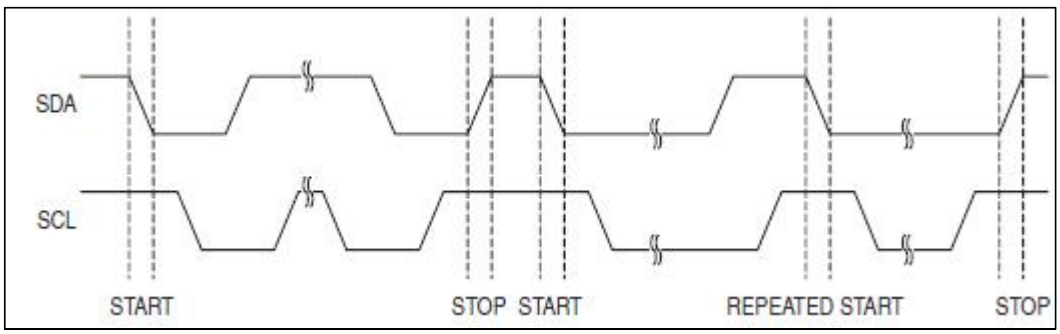
< TWI Bus Interconnection >

TWI 는 비트 단위로 데이터를 전송할 수 있는데 SDA 데이터가 안정된 상태에서 SCK 에 클럭 신호(상승-> 하강)를 발생하면 됩니다.

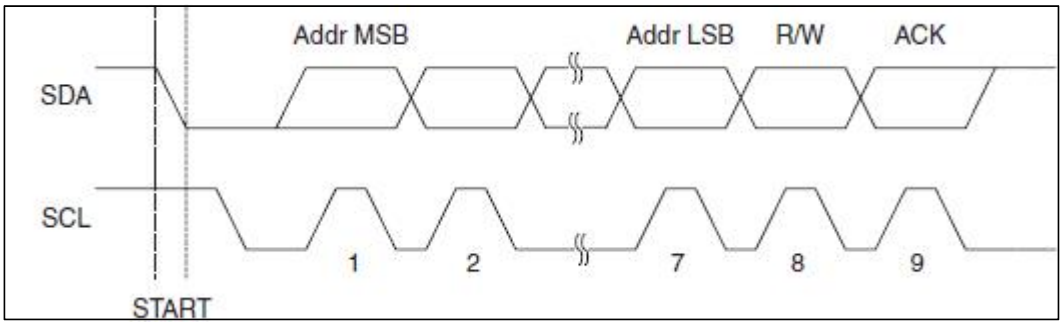


< 데이터 전송 타이밍>

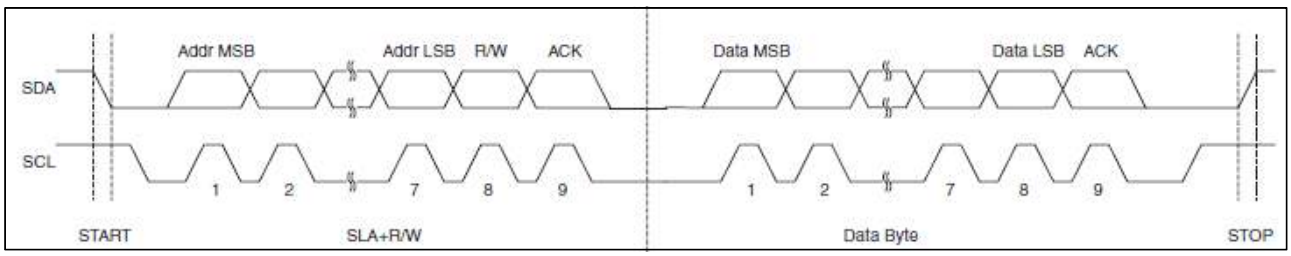
패킷 시작을 알리는 START신호, 패킷 종료를 알리는 STOP 신호는 다음과 같습니다. 만약 연속적인 데이터를 송신하거나 수신한다면 REPEATED START를 사용할 수도 있습니다.



슬레이브를 결정하는 주소 형태는 START 신호 직후에 지정하며 MSB를 먼저 지정합니다.



전체 데이터 포맷은 다음과 같습니다.



8.2. TWI 레지스터

TWI 사용하려면 TWBR, TWCR, TWSR, TWDR, TWAR, 레지스터를 설정해야 합니다.

TWBR - TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• TWI 통신 속도 결정

TWCR - TWI Control Register

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7 - TWINT: TWI Interrupt Flag
- Bit 6 - TWEA: TWI Enable Acknowledge Bit
- Bit 5 - TWSTA: TWI START Condition Bit
- Bit 4 - TWSTO: TWI STOP Condition Bit
- Bit 3 - TWWC: TWI Write Collision Flag
- Bit 2 - TWEN: TWI Enable Bit
- Bit 1 - Res: Reserved Bit
- Bit 0 - TWIE: TWI Interrupt Enable

TWSR - TWI Status Register

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

- Bits 7..3 - TWS: TWI Status
- Bit 2 - Res: Reserved Bit
- Bits 1..0 - TWPS: TWI Prescaler Bits

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

TWDR - TWI Data Register

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

- Write/Read 데이터

TWAR - TWI (Slave) Address Register

Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

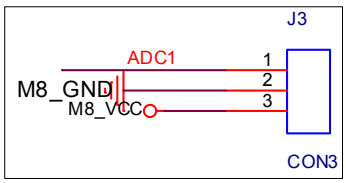
- Bits 7..1 - TWA: TWI (Slave) Address Register
- Bit 0 - TWGCE: TWI General Call Recognition Enable Bit

9. Analog to Digital Convertor - ADC

ATMEGA128 은 8개의 10비트 ADC를 제공합니다.

9.1. 연결

거리측정센서를 ADC1에 연결하고 WAT-CLCD를 연결하여 실험을 해 보겠습니다.



9.2. 거리측정 센서 측정

ADC 값을 받아 LCD에 출력하고 거리를 계산하는 코드입니다.

```
while(1){
        ADMUX=ADC_VREF_TYPE;
        ADCSRA = 0xD7;          // start conversion and clear ADIF
        while((ADCSRA & 0x10) != 0X10);
        sum1 = ADCL + ADCH*256;
        Delay_ms(1);

        sprintf(g_chTemp,"ADC0 Value:%5d ",sum1);
        LCD_PUTSTRINGS(0,0,g_chTemp);
        sprintf(g_chTemp,"ADC0 Value:%5d ",sum1);
        LCD_PUTSTRINGS(0,1,g_chTemp);

        Delay_ms(50);

}
```

10. 교육용 키트 소개

WAT-AVR128 Module은 Atmel 의 ATMEGA128A을 장착한 모듈로 ATMEGA128A 구동에 필요한 기본적인 부품과 회로를 포함하고 있으며, SP3232 RS-232C IC를 장착하여 PC와의 통신을 쉽게 할 수 있게 도와주는 제품입니다.

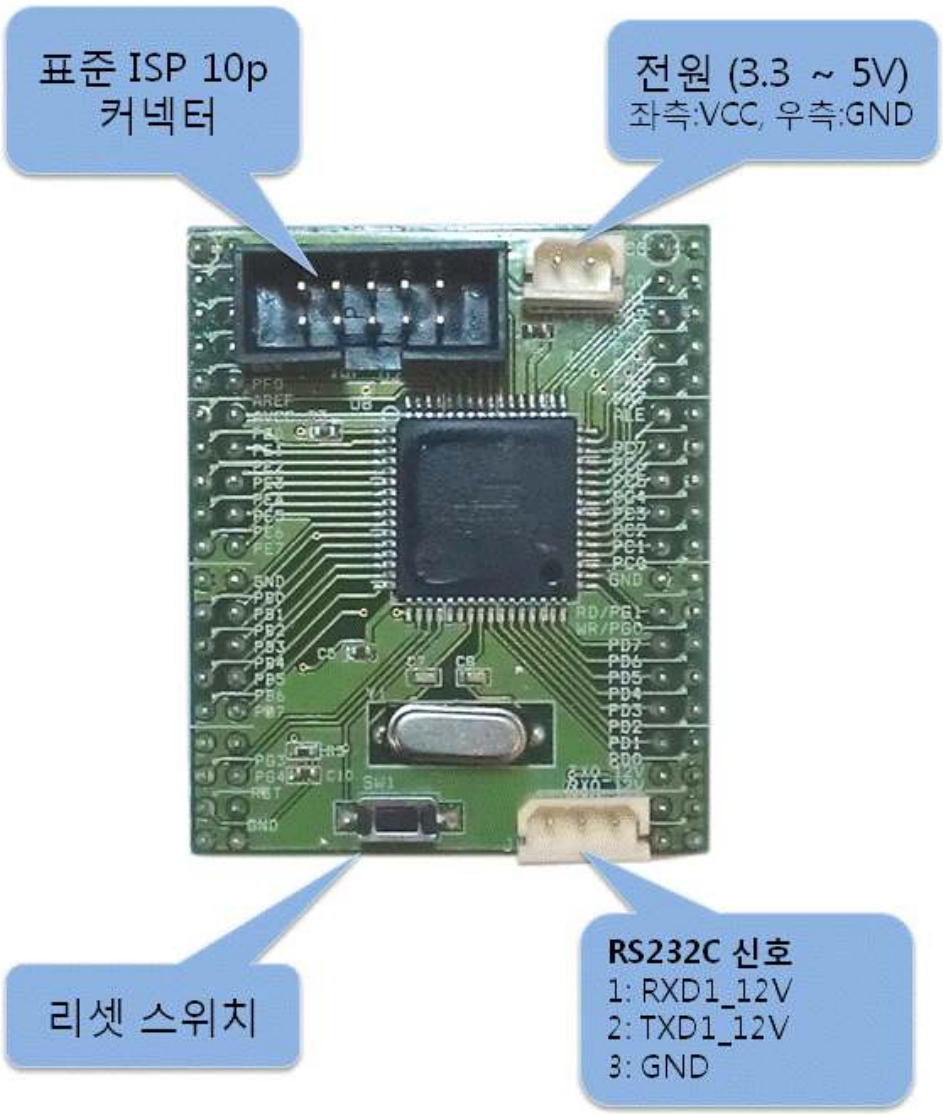
10.1. 특징

항 목	내 용
사용전압	3.3V ~ 5.0V
AD 채널수	8채널 10bit
CLOCK	11.0592 Mhz
시리얼 통신	2채널 2채널을 TTL LEVEL 레벨과 RS-232C 레벨의 두가지로 제공
ISP	ISP 커넥터 제공

SIPEX3232EC 를 사용하여 모든 사용전압(3.3V ~ 5.0V)에서 RS232-C 통신이 가능합니다.

(참고로 일반적인 MAX232 는 5.0V 에서만 RS-232C 통신이 가능합니다.)

10.2. 외형



보드의 크기는 41mm(W) x 49.5mm(H) 이며, 양 쪽에는 2줄짜리 핀헤더(Pitch : 2.54mm)를 제공하여 사용하기 편하게 설계되었습니다.

11. 외부 장치 연결

핀헤더 소켓을 사용하여 외부 장치와 연결하여 사용할 수 있습니다.

11.1. 핀 정의

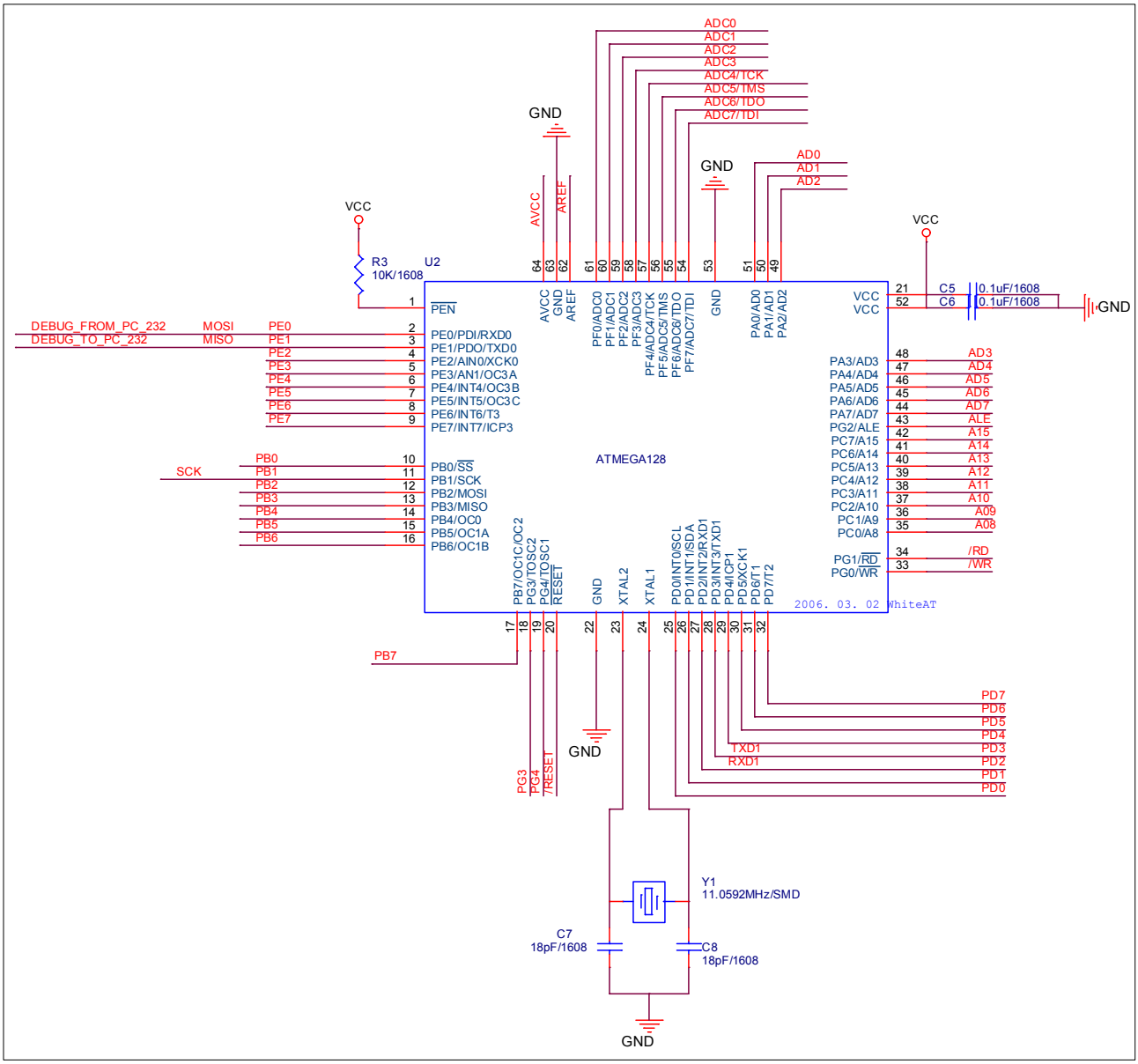
좌 측		우 측	
1. VCC	2. VCC	41. VCC	42. VCC
3. ADC7/TDI	4. ADC6/TDO	43. AD0	44. AD1
5. ADC5/TMS	6. ADC4/TCK	45. AD2	46. AD3
7. ADC3	8. ADC2	47. AD4	48. AD5
9. ADC1	10. ADC0	49. AD6	50. AD7
11. AREF	12. AVCC	51. ALE	52. NC
13. PE0	14. PE1	53. A15	54. A14
15. PE2	16. PE3	55. A13	56. A12
17. PE4	18. PE5	57. A11	58. A10
19. PE6	20. PE7	59. A09	60. A08
21. GND	22. GND	61. GND	62. GND
23. PB0	24. PB1	63. /RD	64. /WR
25. PB2	26. PB3	65. PD7	66. PD6
27. PB4	28. PB5	67. PD5	68. PD4
29. PB6	30. PB7	69. PD3	70. PD2
31. NC	32. NC	71. PD1	72. PD0
33. PG3	34. PG4	73. TXD0_12V	74. RXD0_12V
35. nRESET	36. NC	75. TXD1_12V	76. RXD1_12V
37. GND	38. GND	77.GND	78.GND

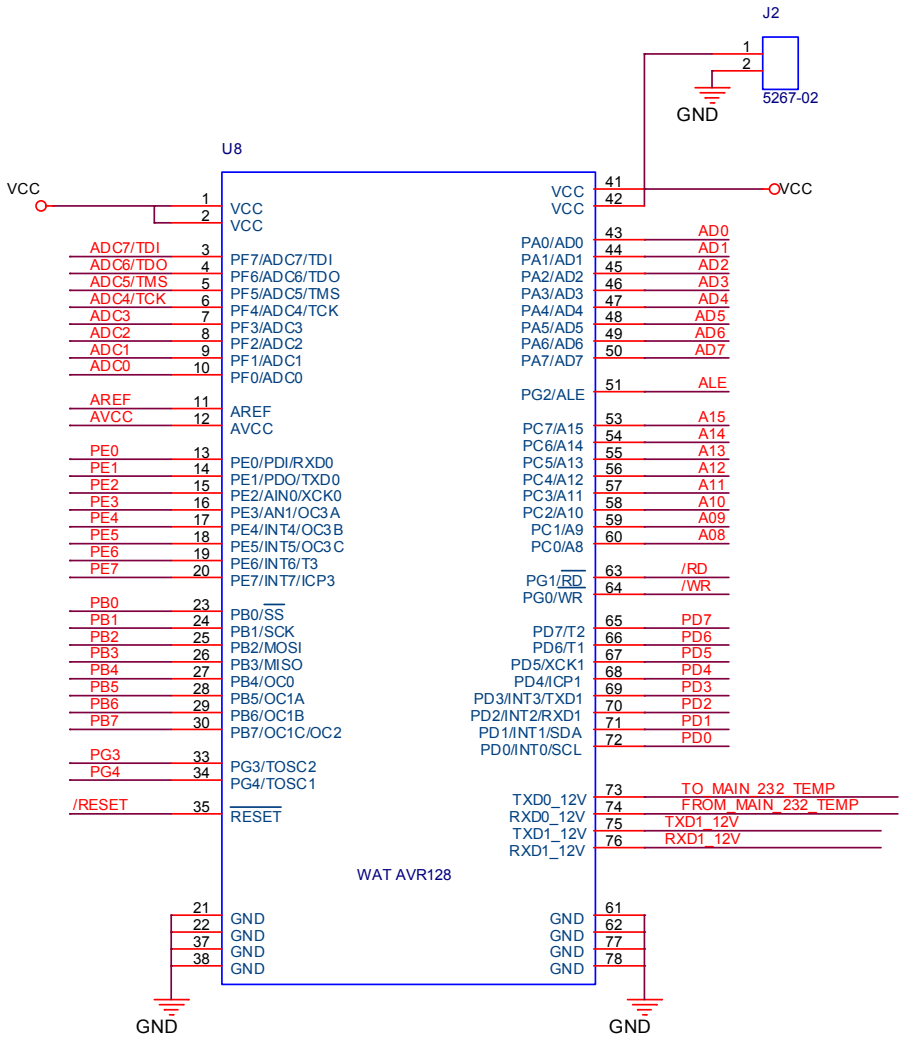
ISP 핀

10핀 ISP용 커넥터를 제공하여 ISP로 프로그램 라이팅이 가능합니다.

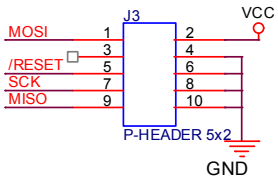
RS-232C 통신용 핀

RS-232C용 통신에 사용되는 핀 3핀으로 PC와의 통신을 쉽게 사용이 가능합니다.

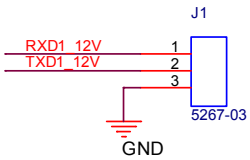

ADC0
ADC1
ADC2
ADC3
ADC4/TCK
ADC5/TMS
ADC6/TDO
ADC7/TDI
GND
AD0
AD1
AD2
VCC
R3
10K/1608
U2
AVCC
AREF
PEN
DEBUG_FROM_PC_232
DEBUG_TO_PC_232
MOSI
MISO
PE0
PE1
PE2
PE3
PE4
PE5
PE6
PE7
PE0/PDI/RXD0
PE1/PDO/TXD0
PE2/AIN0/XCK0
PE3/AN1/OC3A
PE4/INT4/OC3B
PE5/INT5/OC3C
PE6/INT6/T3
PE7/INT7/ICP3
PB0
PB1
PB2
PB3
PB4
PB5
PB6
SCK
PB0/SS
PB1/SCK
PB2/MOSI
PB3/MISO
PB4/OC0
PB5/OC1A
PB6/OC1B
ATMEGA128
PF0/ADC0
PF1/ADC1
PF2/ADC2
PF3/ADC3
PF4/ADC4/TCK
PF5/ADC5/TMS
PF6/ADC6/TDO
PF7/ADC7/TDI
PA0/AD0
PA1/AD1
PA2/AD2
PA3/AD3
PA4/AD4
PA5/AD5
PA6/AD6
PA7/AD7
PG2/ALE
PC7/A15
PC6/A14
PC5/A13
PC4/A12
PC3/A11
PC2/A10
PC1/A9
PC0/A8
PG1/RD
PG0/WR
AD3
AD4
AD5
AD6
AD7
ALE
A15
A14
A13
A12
A11
A10
A09
A08
/RD
/WR
C5
C6
0.1uF/1608
PB7/OC1C/OC2
PG3/TOSC2
PG4/TOSC1
RESET
XTAL2
XTAL1
PD0/INT0/SCL
PD1/INT1/SDA
PD2/INT2/RXD1
PD3/INT3/TXD1
PD4/ICP1
PD5/XCK1
PD6/T1
PD7/T2
PB7
PD7
PD6
PD5
PD4
PD3
PD2
PD1
PD0
TXD1
RXD1
PG3
PG4
/RESET
Y1
11.0592MHz/SMD
C7
18pF/1608
C8
18pF/1608
2006. 03. 02 WhiteAT



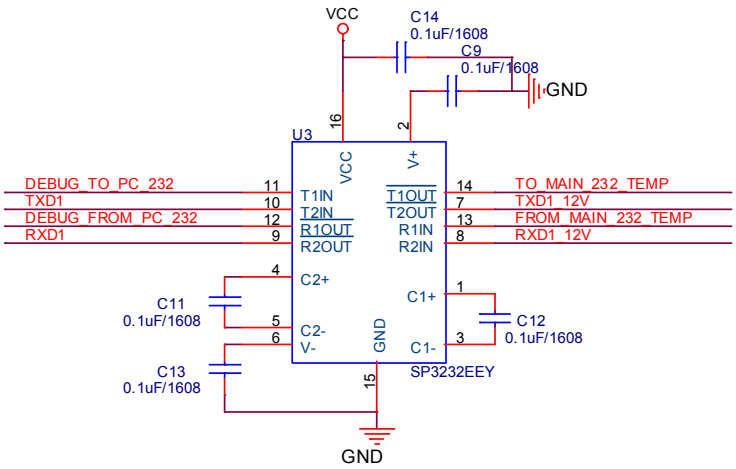
2x19 pin HEADERCONNECTOR



ISP CONNECTOR



RS-232C EXT



SP3232 PART

11.2. 다른 보드에 연결

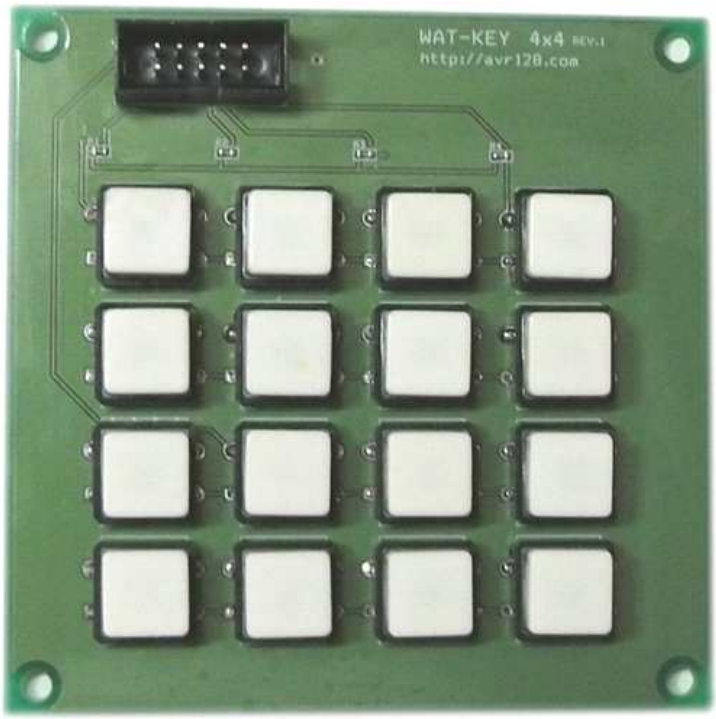
WAT-AVR128_EXT 보드(확장 보드)를 사용하면 다른 보드와 쉽게 연결 할 수 있습니다.

보드명	보드 설명
WAT-KEY 4x4	4x4 배열의 매트릭스 키보드 보드(Input 실험용)
WAT-CLCD	Character LCD 보드
WAT-GLCD	Graphics LCD 보드
WAT-IO&ADC	LED, FND 와 가변 저항 테스트 보드
WAT-KEY 2x4	2x4 배열의 키보드 보드 (Input 실험용)
WAT-LED	8개의 LED 보드(Output 실험용)

< 확장보드와 연결 가능한 보드 >

WAT-KEY 4x4 보드

4x4 배열로 스위치를 장착하여 MCU의 8개의 핀과 VCC 핀만으로 16개의 스위치를 제어하는 예제를 제공합니다.



더 많은 자료와 업데이트된 소스는 홈페이지(http://whiteat.com)에서 확인 및 다운로딩이 가능합니다.