AVR ATtiny2313 입문

Ver. 0.93

정 종 호

B5 용지로 편집되어 있습니다.

2007.3.29 V. 0.93 차례추가, 오타수정
2007.3.14 V. 0.90

차 례

제 1 장

ATtiny2313의 구조와 기능

1.1 ATtiny2313의 기본구조와 기능

1. ATtiny2313의 특징

ATtiny2313은 AVR RISC(Reduced Instruction Set Computer)구조를 기반으로 하는 저전력 CMOS 8비트 마이크로콘트롤러로 단일 클록으로 명령을 수행함으로써 1MHz 당 1 MIPS의 처리속도를 낼 수 있다.

AVR 코어(core)는 32개의 범용 레지스터를 가지고 있으며, 이 범용 레지스터는 한 개의 클록 사이클 내에 한 개의 명령을 수행 하도록 두개의 독립된 레지스터를 액세스할 수 있도록 산술연산장치(ALU)에 직접 연결되어 있다. 이러한 구조는 MCS-51계열의 CISC 마이크로콘트롤러에 비해 10배 이상의 빠른 처리 속도를 낼 수 있다.

ATtiny2313은 2K바이트의 ISP 플래시 프로그램메모리와 128바이트의 EEPROM, 128바이트의 SRAM을 가지고 있으며, 18개의 범용 입·출력 라인, 32개의 범용 레지스터, 온-칩 디버깅이 가능한 싱글-와이어 인터페이스, 비교 모드가 있는 타이머/카운터, 내부와 외부 인터럽트, 직렬통신 USART , 범용 직렬 인터페이스(USI), 내부 오실레이터를 가지는 워치독 타이머, 아이들모드·파워-다운모드·대기모드와 같은 세 개의 전원 절약 모드 등이 있다.

ATtiny2313은 고밀도 비휘발성 메모리 기술이 적용되어 온-칩 ISP 플래시 메모리는 SPI 직렬통신을 통해서 프로그램 메모리에 재프로그래밍 할 수 있다. 모노리식 칩에 8비트 RISC CPU와 ISP 프로그래밍이 가능한 플래시 메모리가 결합함으로써 ATtiny2313은 다양한 임베이디드 제어응용 분야에 적용이 가능하다.

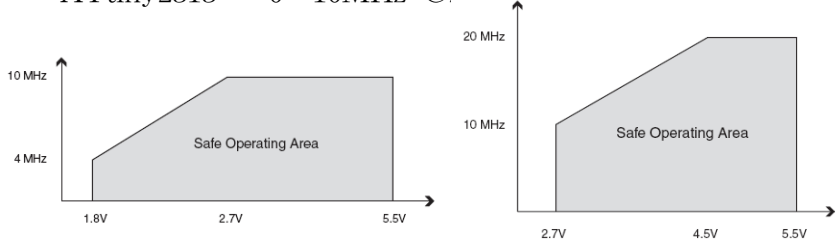
RISC구조의 8비트 마이크로콘트롤러 AVR인 ATtiny2313의 특징은 다음과 같다.

- AVR RISC(Reduced Instruction Set Computer)구조를 사용.
- 고성능, 저전력 RISC구조의 8비트 마이크로콘트롤러.
  - 대부분 단일 클록 사이클에 실행되는 120종의 명령세트.
  - 32개의 8비트 범용 레지스터.

- 20MHz에서 20MIPS의 명령처리 속도.

· 데이터와 비휘발성 프로그램 메모리와 데이터 메모리 구조를 사용.
  - 2KB의 ISP방식 프로그램용 플래시 메모리를 가지며, 10,000번까지 지우고 쓸 수 있음.
  - 128바이트의 데이터 저장용 EEPROM을 가지며, 100,000번까지 지우고 쓸 수 있음.
  - 128바이트의 데이터 저장용 SRAM.
  - 플래시 프로그램과 EEPROM 데이터 보호를 위한 프로그래밍 잠금이 가능.

· 주변장치의 특징
  - 8비트 타이머/카운터 1개, 16비트 타이머/카운터 1개.
  - 2개의 8비트 PWM 출력, 2개의 8~10비트 PWM 출력.
  - 아날로그 비교기.
  - 오실레이터를 내장한 프로그램 가능한 워치독 타이머.
  - 유니버설 직렬 인터페이스.
  - 전이중 통신이 가능한 USART 직렬통신 포트.
  - 8비트, 7비트, 3비트 병렬 I/O포트 각 1개.
  - ATtiny2313V : 0~4MHz @1.8~5.5V, 0~10MHz @2.7~5.5V
  - ATtiny2313 : 0~10MHz @2.7~5.5V, 0~20MHz @ 4.5~5.5V



(a) ATtiny2313V (b) ATtiny2313

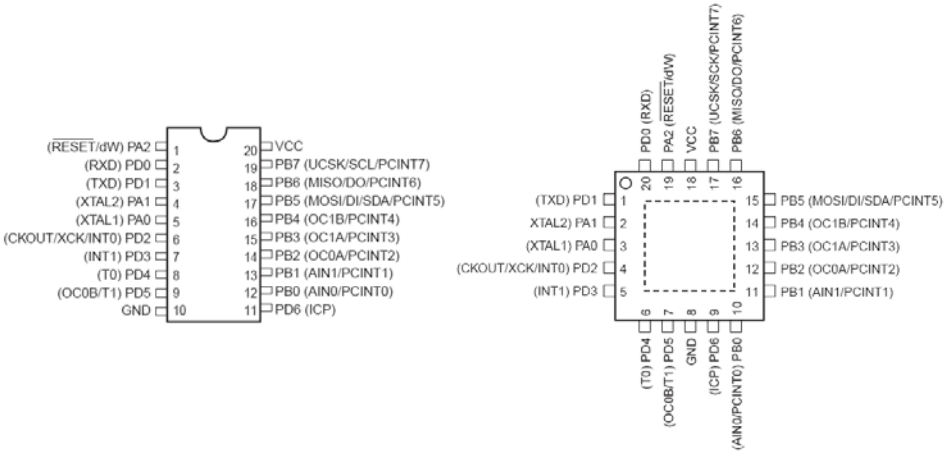
<그림1.1.1> ATtiny2313의 클록 주파수와 Vcc의 관계

· ATtiny2313의 전용특징
  - debugWIRE 온칩 디버깅.
  - SPI포트를 통한 ISP프로그래밍.

- 외부와 내부 인터럽트 소스.
- 파워 온 리셋 회로.
- 저전력 아이들 모드, 파워 다운모드, 대기모드

2. ATtiny2313의 외부 구조

ATtiny2313소자는 20핀의 PDIP/SOIC 패키지와 MLF패키지가 있다. PDIP/SOIC 패키지는 핀번호와 기능이 동일하며, MLF 패키지는 PDIP /SOIC 패키지와 구성만 다를 뿐 기능은 동일하다. PDIP/SOIC 패키지와 MLF 패키지는 <그림 1.1.2>와 같으며, 각 핀의 기능을 요약하면 <표 1.1.1>과 같다.



(a) PDIP/SOIC 패키지 (b) MLF 패키지

<그림1.1.2> ATtiny2313의 외부 구조

<표 1.1.1> ATtiny2313의 외부 핀의 기능 요약

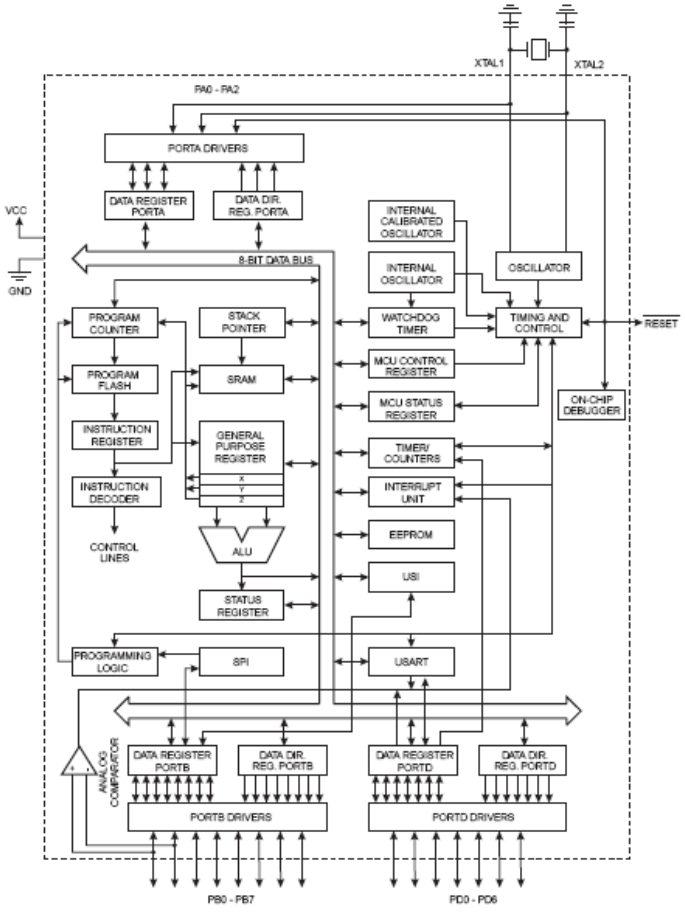
신호 이름	핀번호	기 능
PA0~PA2	5,4,1	· 내부적으로 풀업(20kΩ~50kΩ)된 3비트 양방향 병렬포트. DDRA 레지스터를 사용하여 포트의 입력과 출력을 설정. 입력 데이터는 PINA 레지스터를 통하여 읽으며, 출력 데이터는 PORTA 레지스터를 통하여 출력.
PB0~PB7	12~19	· 내부적으로 풀업(20kΩ~50kΩ)된 8비트 양방향 병렬포트. DDRB 레지스터를 사용하여 포트의 입력과 출력을 설정. 입력 데이터는 PINB 레지스터를 통하여 읽으며, 출력 데이터는 PORTB 레지스터를 통하여 출력.
PD0~PD6	2~3,6~9, 11	· 내부적으로 풀업(20kΩ~50kΩ)된 7비트 양방향 병렬포트. DDRD 레지스터를 사용하여 포트의 입력과 출력을 설정. 입력 데이터는 PIND 레지스터를 통하여 읽으며, 출력 데이터는 PORTD 레지스터를 통하여 출력.
$\overline{\text{RESET}}$	1	· 리셋 신호가 입력되면 마이크로콘트롤러는 초기상태로 다시 시작된다. 올바른 리셋 동작을 위해서는 최소 50ns이상 로우(low)레벨을 유지해야 함.
INT0 INT1	6 7	· 외부 인터럽트 요청신호. MCUCR레지스터에서 인터럽트 신호의 상승/하강에지, 로우(low)레벨일 때, 인터럽트가 발생하도록 설정.
PCINT0~ PCINT7	12~19	· 외부 인터럽트 요청신호. PCMSK 레지스터의 PCINT0~PCINT7가 세트되고 GIMSK의 PCIE가 세트되면 핀 변화 인터럽트는 대응하는 I/O핀을 활성되고 PCINT0~PCINT7가 크리어 되면, 대응하는 I/O핀에 대한 핀 변화 인터럽트는 비활성된다.
T0 T1	8 9	· 타이머/카운터0의 클록 입력 신호. · 타이머/카운터1의 클록 입력 신호.

<표 1.1.1> ATtiny2313의 외부 핀의 기능 요약 (계속)

신호 이름	핀번호	기 능
ICP	11	· 타이머/카운터1의 캡쳐 모드에서 트리거 신호로 사용.
RXD	2	· 직렬통신 포트 USART의 수신 데이터 신호.
TXD	3	· 직렬통신 포트 USART의 송신 데이터 신호.
MOSI	17	· Master Output/Slave Input :ATtiny2313이 ISP프로그래머에게 보내는 데이터신호.
MISO	18	· Master Input/Slaver Output :ISP프로그래머가 ATtiny2313에게 보내는 데이터 신호.
SCK	19	· Serial Clock :ISP프로그래머가 ATtiny2313에게 보내는 프로그래밍 클록신호.
SDA	17	· TWI 채널의 데이터 입출력 신호.
SCL	19	· TWI 채널의 클록 입출력 신호.
OC0A	14	· 각 타이머/카운터의 비교 출력신호. 각 타이머/카운터 채널을 PWM모드로 설정하면 이 단자를 통해 PWM신호가 출력됨.
OC0B	9	
OC1A	15	
OC1B	16	
AIN0	12	· 아날로그 비교기 +입력.
AIN1	13	· 아날로그 비교기 -입력.
XTAL1	5	· 내부 클록발생 증폭회로의 입력신호.
XTAL2	4	· 내부 클록발생 증폭회로의 출력신호
VCC	20	· 회로의 +5V(±10%)전원.
GND	10	· 회로의 전원접지.

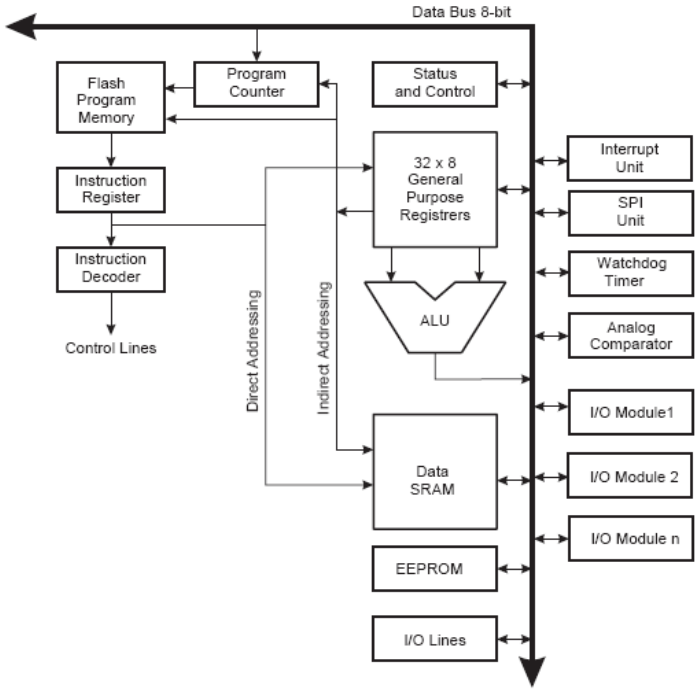
3. ATtiny2313의 내부 구조

ATtiny2313의 내부 구조는 <그림1.1.3>과 같으며, 내부버스가 프로그램 액세스용 버스와 데이터 액세스용 버스로 구성되어 있는 RISC구조이다. 프로그램 액세스용 버스는 내장되어 있는 프로그램 메모리인 플래시 메모리 뿐만 아니라 프로그램 카운터, 명령 해독기, 명령 레지스터 등과도 연결되어 있다. 데이터 액세스용 버스는 범용 레지스터와 SRAM, EEPROM과 같은 데이터 메모리 뿐만 아니라 각종 I/O들과도 연결되어 있다.



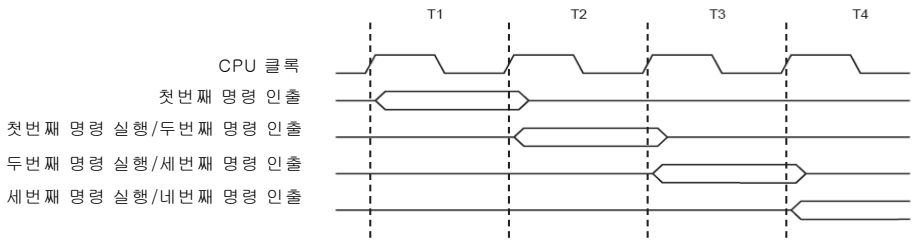
<그림 1.1.3> ATtiny2313의 내부구조

ATtiny2313의 CPU 코어는 <그림 1.1.4>와 같으며, 프로그램 카운터가 지시하는 플래시 메모리 어드레스에서 명령을 인출하고 해독하는 부분과 산술・논리연산장치(ALU), 32개의 범용 레지스터, 상태 레지스터, 스택 포인터 등과 같은 명령 처리부와 데이터 메모리 및 인터럽트 처리기 등으로 구성된다.



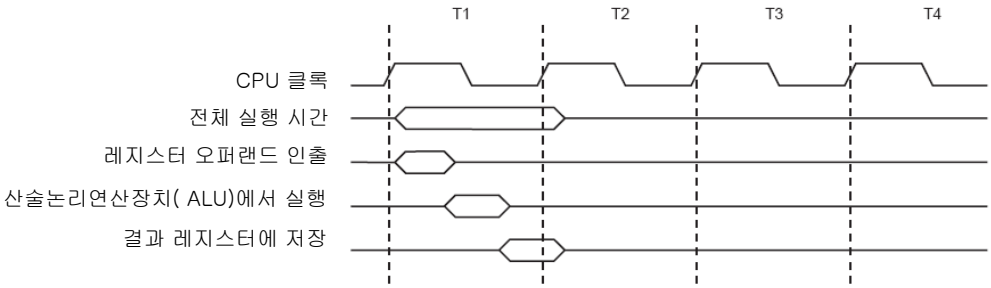
<그림 1.1.4> ATtiny2313의 CPU 코어구조

ATtiny2313이 병렬처리방식으로 프로그램 명령을 인출(fetch)하고 실행(execute)하는 동작은 <그림 1.1.5>와 같다.



<그림 1.1.5> 프로그램 명령의 인출과 실행을 병렬처리하는 동작

단일 클록 사이클의 액세스 시간을 갖는 32개의 범용 레지스터는 산술논리연산장치(ALU) 동작이 단일 클록 사이클에 이루어진다. 이는 한 개의 명령에서 두 개의 오퍼랜드가 범용 레지스터에서 인출되어 산술논리연상장치(ALU)에서 연산되고, 그 결과가 범용 레지스터에 다시 저장되는 과정이 <그림 1.1.6>과 같이 단일 클록 사이클에 실행된다는 것을 의미한다.



<그림 1.1.6> 단일 사이클에 처리되는 산술논리연산장치의 동작

(1) 산술논리연산처리장치(ALU)

ATtiny2313의 산술논리연산처리장치(Arithmetic Logic Unit)는 32개의 범용 레지스터와 직접 연결되어 있어 산술연산, 논리연산, 비트 처리등과 같은 기능을 단일 클록 사이클에 처리하며, 강력한 하드웨어 곱셈기는 부호있는 정수나 부호없는 정수의 곱셈연산을 빠르게 처리한다.

(2) 상태 레지스터(SREG)

상태 레지스터(Status Register)는 가장 최근에 실행된 산술연산 명령의 결과를 표시한다. 상태 레지스터에 저장된 상태 정보는 조건부 처리명령에 의하여 프로그램의 흐름을 변경하는 데 사용한다. 상태 레지스터는 인터럽트 루틴으로 들어갈 때 자동으로 저장되지 않으며, 인터럽트로부터 복귀할 때에 자동으로 복구되지 않으므로 반드시 소프트웨어로 처리해 주어야 한다. ATtiny2313의 상태 레지스터는 <그림 1.1.7>과 같으며, 각 비트의 기능은 표<1.1.2>와 같다.

비트	7	6	5	4	3	2	1	0	
0x3F(0x5F)	I	T	H	S	V	N	Z	C	SREG
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 1.1.7> 상태 레지스터(SREG)

<표 1.1.2> ATtiny2313의 상태 레지스터

비트	비트명	기 능
Bit 7	I	Global Interrupt Enable : 인터럽트 허용을 설정하는 비트. 개별적인 인터럽트를 허용하는 것은 일반 인터럽트 제어 레지스터 및 각 주변 장치의 제어 레지스터에 의해 설정됨.
Bit 6	T	Bit Copy Storage : 상태 레지스터의 이 비트와 다른 레지스터의 한 비트 사이에 비트 복사가 가능.
Bit 5	H	Half carry Flag : 산술연산에서 하프캐리를 지시하는 비트. BCD연산에서 유용함.
Bit 4	S	Sign Bit, S=N⊕V : 네가티브 플랙 N과 2의 보수 오버플로 플랙 V의 배타적OR 결과값을 저장.
Bit 3	V	Two's Complement Overflow Flag : 2의 보수 연산에서 오버플로의 발생을 나타냄.
Bit 2	N	Negative Flag : 산술연산이나 논리연산의 결과가 음수임을 나타냄.
Bit 1	Z	Zero Flag : 산술연산이나 논리연산 결과가 0임을 나타냄.
Bit 0	C	Carry Flag : 산술연산이나 논리연산에서 캐리의 발생을 나타냄.

(3) 범용 레지스터

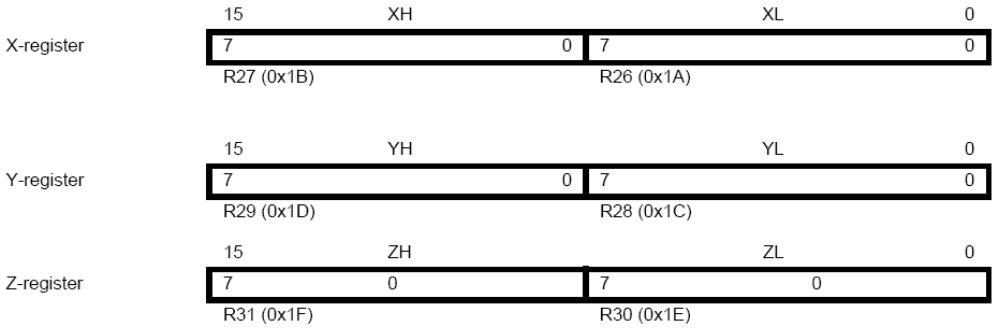
ATtiny2313은 <그림 1.1.8>과 같이 CPU 안에 32개의 범용 레지스터(R0~R31)를 가지고 있다. 이들 레지스터는 기본적인 사칙연산을 수행하며, 즉치 데이터를 사용하는 연산명령은 R16~R31에서만 수행된다.

7	0	Addr.	
R0		0x00	
R1		0x01	
R2		0x02	
…			
R13		0x0D	
R14		0x0E	
R15		0x0F	
R16		0x10	
R17		0x11	
…			
R26		0x1A	X-register Low Byte
R27		0x1B	X-register High Byte
R28		0x1C	Y-register Low Byte
R29		0x1D	Y-register High Byte
R30		0x1E	Z-register Low Byte
R31		0x1F	Z-register High Byte

<그림 1.1.8> ATtiny2313의 범용 레지스터

(4) X-레지스터, Y-레지스터, Z-레지스터

R26~R31의 레지스터는 <그림 1.1.9>와 같이 두 개의 범용 레지스터가 더해져서 16비트 레지스터인 X-레지스터, Y-레지스터, Z-레지스터로 사용된다. 즉, X-레지스터의 경우 범용 레지스터 R26이 하위 8비트를 범용 레지스터 R27이 상위 8비트를 구성하여 16비트 X-레지스터가 된다. 이들 레지스터는 데이터 영역의 간접 어드레스 지정을 위한 16비트 어드레스 포인터로 사용된다.



<그림 1.1.9> X-레지스터, Y-레지스터, Z-레지스터의 구성

(5) 스택 포인터(SP)

스택은 주로 임시 데이터를 저장하거나 또는 지역변수나 인터럽트와

서브루틴 호출 후 복귀 어드레스를 저장하는 데에도 사용된다. 스택 포인터(Stack Pointer) 레지스터는 항상 스택의 상단을 가리킨다. 스택 포인터는 푸시 (PUSH)명령에 의해 데이터를 스택에 저장될 때 1씩 감소하고, 팝(POP)명령에 의해 1씩 증가한 후에 스택 메모리 번지에서 데이터를 꺼낸다. 스택에 푸시 명령으로 서브루틴 호출이나 인터럽트에 의해 16비트의 주소가 저장될 때 스택 포인터가 2씩 감소하며, 팝 명령으로 서브루틴의 RET명령이나 인터럽트의 RETI명령에 의해 스택에서 주소를 가져오면 스택 포인터는 2씩 증가한다. ATtiny2313의 스택 포인터는 서브루틴 호출 이나 인터럽트가 인에이블되기 전에 프로그램에 의해 데이터 SRAM에 스택영역을 정의해야 하며, 사용자 프로그램에서 스택 포인터의 초기값은 내부 SRAM에 위치시키기 위해서 적어도 0x0060번지 이상의 값으로 설정해야 하지만, 일반적으로 내부 SRAM의 끝번지인 0x00DF로 지정한다.

비트	7	6	5	4	3	2	1	0	
0x3E(0x5E)	-I	-	-	-	-	-	-	-	
0x3D(0x5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

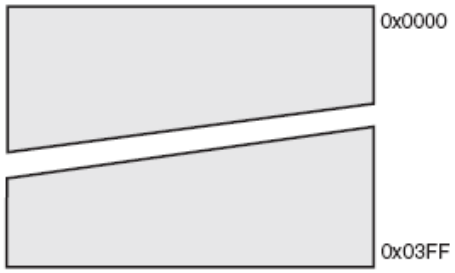
<그림 1.1.10> 스택 포인터(SP)

4. ATtiny2313의 메모리 구조

ATtiny2313의 메모리는 데이터 메모리와 프로그램 메모리로 구분할 수 있으며, 데이터를 저장하기 위한 EEPROM 메모리를 추가로 가지고 있다.

(1) 플래시 프로그램 메모리

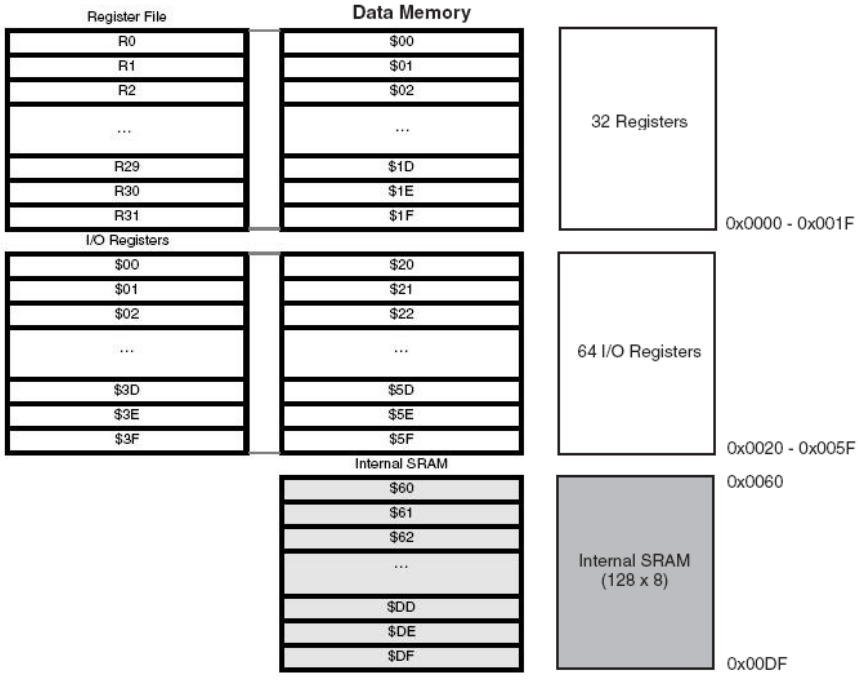
ATtiny2313은 <그림 1.1.11>과 같이 내부에 프로그램 메모리로 2KB의 플래시 메모리를 가지고 있다. ATtiny2313는 8비트 마이크로콘트롤러 이지만 모든 명령은 16비트 또는 32비트 이므로 플래시 메모리는 1KB×16비트=2KB의 용량을 갖는다. 이 플래시 메모리는 SPI 직렬통신을 통하여 최소한 10,000회 이상 쓰기와 지우기를 할 수 있다.



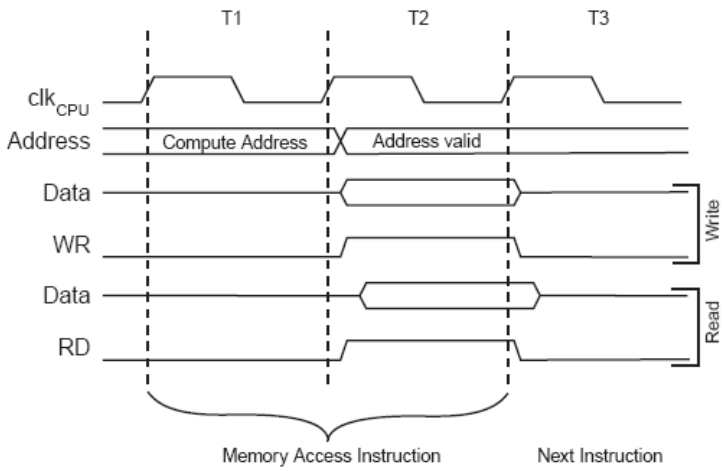
<그림 1.1.11> 플래시 프로그램 메모리

(2) SRAM 데이터 메모리

ATtiny2313의 SRAM 데이터 메모리는 <그림 1.1.12>와 같다. SRAM 데이터 메모리는 32개의 범용 레지스터(0x0000～0x001F), 64개의 I/O 메모리(0x0020～0x005F), 128개의 내부 데이터 SRAM(0x0060～0x00DF)으로 구성되어 있다. SRAM 데이터 메모리를 액세스 하는 데는 <그림 1.1.13>과 같이 CPU의 2클록 사이클이 소요된다.



<그림 1.1.12> SRAM 데이터 메모리 맵



<그림 1.1.13> SRAM 데이터 메모리 액세스 사이클

(3) EEPROM 데이터 메모리

ATtiny2313에는 128바이트 EEPROM이 내장되어 있으므로 메모리 번지는 0x0000~0x007F로 할당된다. 100,000번까지 데이터를 읽고 쓸 수 있으며, EEPROM 어드레스 레지스터 EEAR, EEPROM 데이터 레지스터 EEDR, EEPROM 제어 어드레스 EECR 등을 사용하여 1바이트씩 액세스 할 수 있다.

EEPROM을 읽을 때는 4클록 사이클 동안 CPU를 정지시켰다 다음 명령을 실행해야하며, EEPROM을 쓸 때는 2클록 사이클 동안 CPU를 정지시켰다 다음 명령을 실행시켜야 한다.

1) EEPROM 어드레스 레지스터(EEAR)

비트	7	6	5	4	3	2	1	0	
0x1E(0x3E)	-	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEAR
읽기 / 쓰기	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 1.1.14> EEPROM 어드레스 레지스터(EEAR)

<표 1.1.3> ATtiny2313의 EEPROM 어드레스 레지스터(EEAR)

비트	기 능
Bit6~Bit0	· 128바이트 EEPROM영역에서 EEPROM번지 지정.

2) EEPROM 데이터 레지스터(EEDR)

비트	7	6	5	4	3	2	1	0	
0x1D(0x3D)	MSB							LSB	EEDR
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 1.1.15> EEPROM 데이터 레지스터(EEDR)

<표 1.1.4> ATtiny2313의 EEPROM 데이터 레지스터(EEDR)

비트	기 능
Bit7~Bit0	· 쓰기할 데이터를 저장하거나, 읽기할 주소를 저장.

3) EEPROM 제어 레지스터(EECR)

비트	7	6	5	4	3	2	1	0	
0x1C(0x3C)	-	-	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	EECR
읽기 / 쓰기	R	R	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 1.1.16> EEPROM 제어 레지스터(EECR)

<표 1.1.5> ATtiny2313의 EEPROM 제어 레지스터(EECR)

비트	비트명	기 능
Bit5~Bit4	EEPM1, EEPM0	· EEPROM 프로그래밍 모드 비트.
Bit 3	EERIE	· SREG의 I비트를 '1'로 하고 EERIE 비트를 '1'로 설정하면 EEPROM Ready 인터럽트를 인에이블하고, EERIE 비트를 '0'으로 설정하면 EEPROM Ready 인터럽트는 디스에이블 됨.
Bit 2	EEMPE	· EEMPE='1'이고, EEPE='1'일때, 4클록 사이클 이내에 데이터를 지정된 EEPROM주소에 프로그램함.
Bit 1	EEPE	· EEPROM 프로그램 인에이블 신호.
Bit 0	EERE	· EEPROM 읽기 인에이블 신호.

1.2 ATtiny2313의 기본 하드웨어

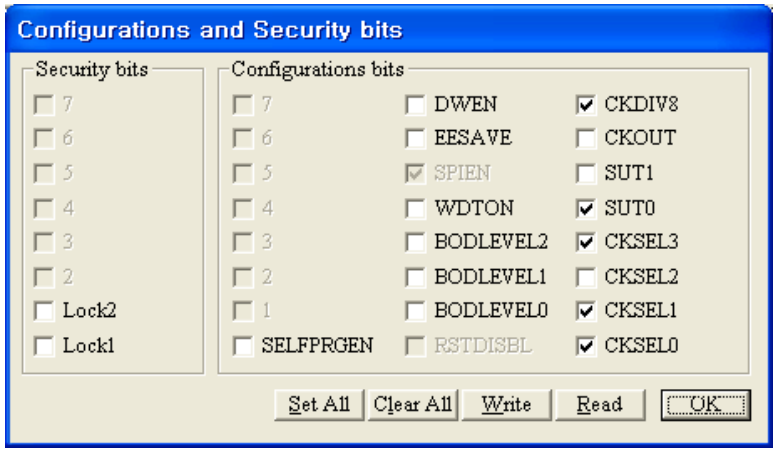
1. 메모리 록 비트와 퓨즈 비트

ATtiny2313은 메모리의 보호 기능을 가지는 2개의 록비트가 있으며, 시스템 설정용 3개의 퓨즈 바이트를 가지고 있다.

(1) 메모리 록 비트

프로그램 메모리와 데이터 메모리 프로그래밍의 호보기능을 설정하는 메모리 록 비트는 1바이트 중 2개의 비트만 사용한다. 메모리 록 비트는 기본값으로 1(unprogrammed)로 주어지며, 프로그램이 되면 0(programmed)로 설정된다. 메모리 록 비트를 기본값 1(unprogrammed)로 환원하려면, 칩 삭제(chip erase) 명령만으로 할 수 있다.

AVR 다운로드 프로그램 중 'ToastProg 2005'의 경우에는 <그림 1.2.1>과 같이 메모리 록 비트와 퓨즈 비트를 설정한다. ☑로 체크한 비트는 '0'으로 설정되고 ☐로 체크가 되어 있지 않은 비트는 '1'로 설정된다.



<그림 1.2.1> ToastProg 2005의 메모리 록 비트와 퓨즈비트

ATtiny2313의 메모리 록 비트의 구성은 <표 1.2.1>과 같다. <표 1.2.2>에서 LB모드 2와 LB모드 3은 LB1과 LB2를 프로그래밍 하기 전에 퓨즈(fuse) 비트와 부트(boot) 비트를 프로그램해야 한다.

<표 1.2.1> ATtiny2313의 메모리 록 비트의 구성

비트번호	비트이름	기 능	기본값
7	-	-	1(unprogrammed)
6	-	-	1(unprogrammed)
5	-	-	1(unprogrammed)
4	-	-	1(unprogrammed)
3	-	-	1(unprogrammed)
2	-	-	1(unprogrammed)
1	LB2	록 비트	1(unprogrammed)
0	LB1	록 비트	1(unprogrammed)

<표 1.2.2> 메모리 록 비트에 의한 메모리 보호 기능

메모리 록 비트			보호 기능
LB모드	LB2	LB1	
1	1	1	메모리 록 기능이 설정되지 않음.
2	1	0	플래시 메모리나 EEPROM을 병렬프로그램 모드나 직렬 프로그램 모드로 프로그램 금지. 퓨즈비트 프로그래밍도 금지.
3	0	0	플래시 메모리나 EEPROM을 병렬 프로그램 모드나 직렬 프로그램 모드로 검증(verify)하는 것을 금지. 부트 록 비트와 퓨즈 비트 프로그래밍도 금지.

(2) 퓨즈 비트

퓨즈 비트는 프로그램하지 않으면 1(unprogrammed)로 설정되며, 프로그램하면 0(programmed)으로 설정된다. 메모리 록 비트의 경우에는 칩 삭제 명령에 의해 다시 기본값이 '1'로 되지만, 퓨즈비트는 이에 영향을 받지 않는다. 퓨즈 비트는 메모리 록 비트의 LB1을 사용하여 퓨즈 비트의 변경을 금지시키도록 설정할 수 있다. 메모리 록 비트에서 설명한 것 처럼 LB1을 이용하여 퓨즈 비트의 변경을 금지하는 LB모드 2와 LB 모드 3의 경우에는 퓨즈 비트와 부트 비트를 설정한 후에 메모리 록 비트를 설정해야 한다.

1) 퓨즈 확장 바이트(Fuse Extended Byte)

퓨즈 확장 바이트는 <표1.2.3>과 같이 1바이트 중 1 비트 만 사용한다. 셀프 프로그래밍을 허용하는 SELPRGEN 비트 1개 만으로 구성되어 있다.

<표 1.2.3> 퓨즈 확장 바이트

비트번호	비트이름	기 능	기본값
7	-	-	1(unprogrammed)
6	-	-	1(unprogrammed)
5	-	-	1(unprogrammed)
4	-	-	1(unprogrammed)
3	-	-	1(unprogrammed)
2	-	-	1(unprogrammed)
1	-	-	1(unprogrammed)
0	SELFPRGEN	셀프 프로그래밍 허용	1(unprogrammed)

2) 퓨즈 하이 바이트(Fuse High Byte)

퓨즈 하이 바이트는 <표1.2.4>와 같다. ATtiny2313은 SPIEN=0로 기본값이 되어 있으므로 SPI에 의한 프로그래밍이 가능하며, EESAVE=1인 상태로 칩 삭제 명령을 사용하면 EEPROM의 내용이 보존되지 않고 삭제된다는 점에 유의해야 한다. 즉, EESAVE=0로 설정된 경우에는 칩 삭제 명령을 사용하더라도 EEPROM 메모리의 내용이 보존된다.

<표 1.2.4> 퓨즈 하이 바이트

비트번호	비트이름	기 능	기본값
7	DWEN	debugWIRE 허용	1(unprogrammed)
6	EESAVE	칩 삭제시 EEPROM메모리 보존.	1(unprogrammed, EEPROM보존안됨.)
5	SPIEN	SPI 직렬 프로그래밍 허용	0(programmed)
4	WDTON	워치독 타이머 항상 on	1(unprogrammed)
3	BODLEVEL2	BOD 트리거 레벨	1(unprogrammed)
2	BODLEVEL1		1(unprogrammed)
1	BODLEVEL0		1(unprogrammed)
0	RSTDISBL	외부 리셋 금지	1(unprogrammed)

3) 퓨즈 로우 바이트(Fuse Low Byte)

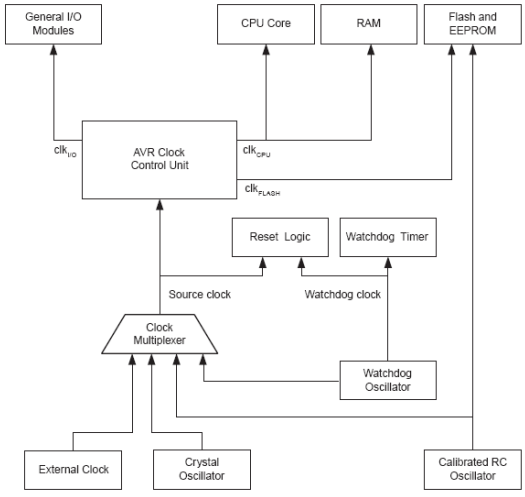
퓨즈 로우 바이트는 <표1.2.5>와 같다. ATtiny2313은 기본값 상태에서 기동시간(start-up)이 최대값으로 되어 있고, 시스템 클록은 내부 RC 오실레이터에 의해 1MHz로 발진된다.

<표 1.2.5> 퓨즈 로우 바이트

비트번호	비트이름	기 능	기본값
7	CKDIV8	클록을 1/8로 분주	0(programmed)
6	CKOUT	CKOUT핀으로 클록 출력	1(unprogrammed)
5	SUT1	기동시간 선택	1(unprogrammed)
4	SUT0		0(programmed)
3	CKSEL3	클록 소스 선택	0(programmed)
2	CKSEL2		1(unprogrammed)
1	CKSEL1		0(programmed)
0	CKSEL0		0(programmed)

2. 시스템 클록과 클록 옵션

<그림 1.2.2>는 ATtiny2313에서 사용할 수 있는 시스템 클록의 종류와 클록을 분배하는 계통도를 나타낸다.



<그림 1.2.2> ATtiny2313의 클록 분배

AVR 클록 제어 유니트(AVR Clock Control unit)로부터 공급되는 클록은 CPU클록(clk_{CPU})과 I/O클록($clk_{I/O}$), 플래시 클록(clk_{FLASH})등이 있다.

(1) 클록 소스

ATtiny2313의 클록 소스는 <표 1.2.6>과 같다. 클록 소스는 퓨즈 로우 바이트의 CKSEL3~CKSEL0를 사용하여 설정할 수 있다.

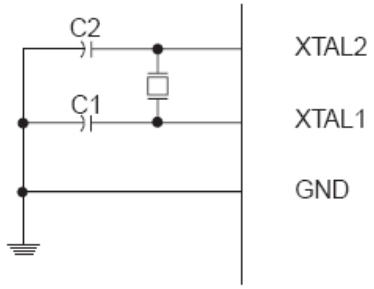
<표 1.2.6> 클록 소스 설정

클록 소스	CKSEL3~CKSEL0
외부 클록(External Clock)	0000
교정된 내부 RC오실레이터 4MHz (Calibrated Internal RC Oscillator 4MHz)	0010
교정된 내부 RC오실레이터 8MHz (Calibrated Internal RC Oscillator 8MHz)	0100
워치독 오실레이터 128kHz (Watchdong Oscillator 128kHz)	0110
외부 크리스탈/ 세라믹 레조네이터 (External Crystal/Ceramic Resonator)	1000~1111

ATtiny2313의 클록 소스 기본값은 CKSEL="0100", SUT="10", CKDIV8="0"으로 되어 있으므로, 퓨즈 비트를 사용자가 별도로 설정하지 않은 초기상태에서는 내부 RC오실레이터 8MHz가 1/8분주된 1MHz의 시스템 클록으로 동작한다.

(1) 크리스탈 오실레이터

크리스탈 이나 세라믹 레조네이터를 사용하는 경우 <그림 1.2.3>과 같이 XTAL1 입력단자와 XTAL2 출력단자에 접속한다. 크리스탈을 사용하는 경우 C1과 C2의 캐패시터 값과 CKSEL3~CKSEL1 비트의 설정은 <표1.2.7>과 같다. CKSEL0 비트에 대한 설정은 SUT1~SUT0 비트와 기동시간을 <표 1.2.8>에서 선택하여 사용한다.



<그림 1.2.3> 크리스탈 오실레이터

<표 1.2.7> 크리스탈 오실레이터 동작 모드

CKSEL3~1	주파수범위(MHz)	크리스탈에 대한 C1, C2 권장값
100	0.4 ~ 0.9	세라믹 레조네이터에서만 사용
101	0.9 ~ 3.0	12 ~ 22pF
110	3.0 ~ 8.0	12 ~ 22pF
111	8.0 ~	12 ~ 22pF

<표 1.2.8> 크리스탈 오실레이터 클록에 대한 기동시간

CKSEL0	SUT1~SUT0	기동시간	리셋에 대한 추가 지연시간(Vcc=5.0V)	권장되는 사용법
0	00	258 클록	14CK + 4.1ms	세라믹 레조네이터 (fast rising power)
0	01	258 클록	14CK + 65ms	세라믹 레조네이터 (slowly rising power)
0	10	1K 클록	14CK	세라믹 레조네이터 (BOD 허용)
0	11	1K 클록	14CK + 4.1ms	세라믹 레조네이터 (fast rising power)
1	00	1K 클록	14CK + 65ms	세라믹 레조네이터 (slowly rising power)
1	01	16K 클록	14CK	크리스탈 오실레이터 (BOD 허용)
1	10	16K 클록	14CK + 4.1ms	크리스탈 오실레이터 (fast rising power)
1	11	16K 클록	14CK + 65ms	크리스탈 오실레이터 (slowly rising power)

(2) 내부 RC 오실레이터

내부 RC 오실레이터(calibrated internal RC oscillator) 는 4.0, 8.0MHz 클록 중 하나를 <표 1.2.9>와 같이 선택할 수 있으며, 내부 RC오실레이터 클록을 선택하면 XTAL1과 XTAL2 단자에 외부 소자를 접속하지 않고 동작한다.

<표 1.2.9> 내부 RC 오실레이터 동작 범위

CKSEL3~0	주파수범위(MHz)
0010 ~ 0011	4.0
0100 ~ 0101	8.0

내부 RC 오실레이터가 사용되는 경우의 기동시간은 <표 1.2.10>과 같이 SUT1~SUT0비트에 따라 선택한다.

<표 1.2.10> 내부 RC 오실레이터에 대한 기동시간

SUT0~SUT0	기동시간	리셋에 대한 추가 지연시간(Vcc=5.0V)	권장되는 사용법
00	6 클록	14CK	BOD 허용
01	6 클록	14CK + 4.1ms	Fast rising power
10	6 클록	14CK + 65ms	Slowly rising power
11	-		

내부 RC 오실레이터의 클록 주파수는 비교적 부정확하므로 이를 조정하는 데는 <그림 1.2.4>와 같이 오실레이터 교정 레지스터 OSCCAL(Oscillator Calibration Register)를 사용하며, 내부 RC 오실레이터 주파수 범위는 <표 1.2.11>과 같다.

비트	7	6	5	4	3	2	1	0	
0x31(0x51)	-	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
읽기 / 쓰기	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

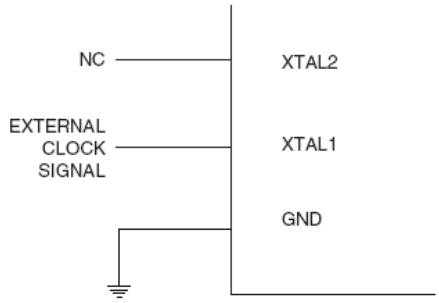
<그림 1.2.4> 오실레이터 교정 레지스터 OSCCAL

<표 1.2.11> 내부 RC 오실레이터 동작 범위

OSCCAL값	최저 주파수	최대 주파수
0x00	50 %	100 %
0x3F	75 %	150 %
0x7F	100 %	200 %

(3) 외부 클록

외부 클록 소스를 XTAL1에 <그림 1.2.5>와 같이 접속하고, CKSEL 3~CKSEL0의 퓨즈 비트를 “0000”으로 설정해야 한다. 기동시간은 <표 1.2.12>와 같이 선택한다.



<그림 1.2.5> 외부 클록 소스

<표 1.2.12> 외부 클록 소스에 대한 기동시간

SUT0~SUT0	기동시간	리셋에 대한 추가 지연시간(Vcc=5.0V)	권장되는 사용법
00	6 클록	14CK	BOD 허용
01	6 클록	14CK + 4.1ms	Fast rising power
10	6 클록	14CK + 65ms	Slowly rising power
11	-		

(4) 128kHz 내부 오실레이터

128kHz 내부 오실레이터는 128kHz의 클록을 공급하는 저전력 오실레이터로 CKSEL3~CKSEL0의 퓨즈 비트를 “0110”으로 설정하고, 기동시간은 <표1.2.13>과 같이 선택한다.

<표 1.2.13> 128kHz 내부 오실레이터에 대한 기동시간

SUT0~SUT0	기동시간	리셋에 대한 추가 지연시간(Vcc=5.0V)	권장되는 사용법
00	6 클록	14CK	BOD 허용
01	6 클록	14CK + 4ms	Fast rising power
10	6 클록	14CK + 64ms	Slowly rising power
11	-		

3. 슬립 모드

슬립 모드는 마이크로콘트롤러의 사용하지 않는 모듈의 동작을 정지시켜 소비전력을 낮추는 동작이다. ATtiny2313은 3가지 슬립모드를 제공한다.

슬립 모드로 들어가기 위해서는 <그림 1.2.6>의 MCU 제어 레지스터 MCUCR의 SE=1로 설정하고 SM1과 SM0비트를 이용하여 <표 1.2.14>와 같이 슬립 모드를 3가지 중 하나를 지정하고 SLEEP 명령을 실행한다.

비트	7	6	5	4	3	2	1	0	
0x35(0x55)	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 1.2.6> MCU 제어 레지스터 MCUCR

<표 1.2.14> 슬립 모드 선택

SM1	SM0	슬립 모드
0	0	Idle 모드
0	1	Power-down 모드
1	0	Power-down 모드
1	1	Standby 모드

(1) Idle 모드

Idle 모드에서는 CPU의 동작은 정지하지만, UART, 아날로그 비교기, USI, 타이머/카운터, 워치독, 인터럽트 시스템의 동작은 계속된다. 이 모드에서는 clk_{CPU}와 clk_{FLASH}는 정지하지만, 다른 클록은 동작된다.

Idle 모드는 외부 트리거 인터럽트나 타이머 오버플로어, UART 송신 종료 인터럽트로 해제하여 마이크로컨트롤러를 동작시킨다. 아날로그 비교기 인터럽트로 Idle 모드 해제를 원하지 않는다면, 아날로그 비교기 제어와 상태 레지스터인 ACSR에서 ACD비트를 '1'로 하여 아날로그 비교기의 전원을 차단하면 Idle 모드에서의 소비전력이 감소한다.

(2) Power-down 모드

Power-down 모드에서는 외부 오실레이터의 발진은 정지하지만, 외부 인터럽트, USI 시작 조건 검출, 워치독은 계속 동작한다. 이 모드에서는 외부 리셋, 워치독 리셋, Brown-out 리셋, USI 시작 조건 인터럽트, INT0 외부 레벨 인터럽트 등에 의해서 만 해제할 수 있다.

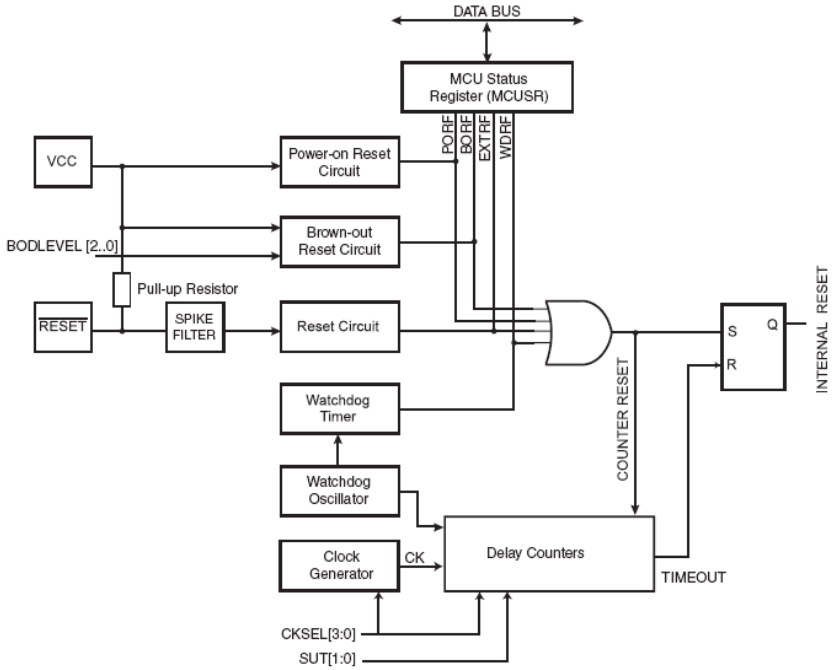
Power-down 모드에서는 비동기 모듈의 클록을 제외한 모든 클록의 발진이 정지된다.

(3) Standby 모드

Standby 모드는 오실레이터가 계속 동작하는 것을 제외하고 Power-down 모드와 동일하다.

4. 시스템 리셋

ATtiny2313이 리셋 되면, 모든 I/O레지스터는 초기값으로 되고, 프로그램은 리셋 벡터로 부터 시작된다. <그림 1.2.7>은 리셋동작과 관련된 계통도이다.



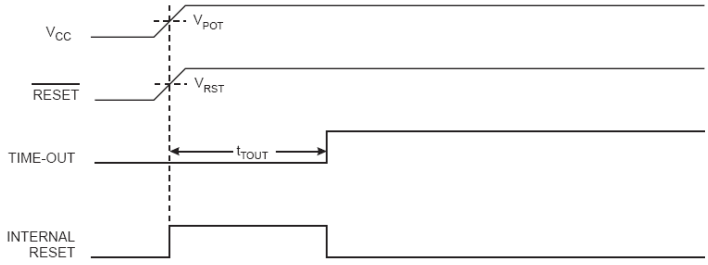
<그림 1.2.7> ATtiny2313의 리셋 계통도

리셋 소스가 입력되면 ATtiny2313의 I/O포트는 즉시 초기상태로 된다. 리셋 소스의 입력이 완료되더라도 지연 카운터(delay counter)동작되어 내부적인 리셋을 지연시켜, 마이크로프로세서가 정상적인 동작을 하기 전에 전원이 안정 상태에 도달하도록 한다. 지연 카운터의 지연시간 주기는 퓨즈 비트 SUT1~SUT0와 CKSEL0로 설정한다.

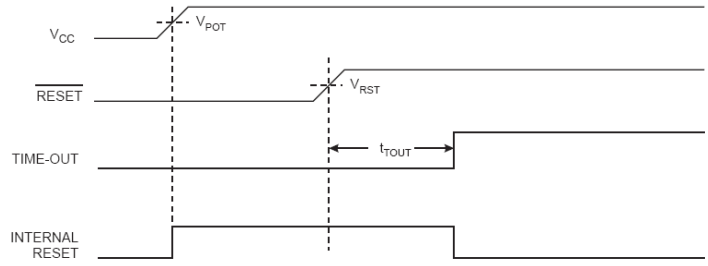
ATtiny2313의 시스템 리셋 소스는 전원-ON 리셋, 외부 리셋, 워치독 리셋, 저전압 검출 리셋 등 4가지가 있다.

(1) 전원-ON 리셋

전원-ON 리셋(Power-on reset)은 인가전압이 전원-ON 리셋 문턱전압(threshod voltage) V_{POT}이하로 내려갔을 때, ATtiny2313이 리셋된다. ATtiny2313의 전원-ON 리셋 문턱전압의 상승시 대표값은 1.2V이고, 하강시 대표값은 1.1V이다. 전원-ON 리셋은 <그림1.2.8>과 같이 리셋 핀을 Vcc에 접속한 경우와 <그림 1.2.9>와 같이 외부 리셋 신호에 접속한 경우가 있으며, 안정된 동작을 위해 외부 리셋 신호에 의한 전원-ON 리셋의 내부 리셋 지연시간이 더 길어진다.



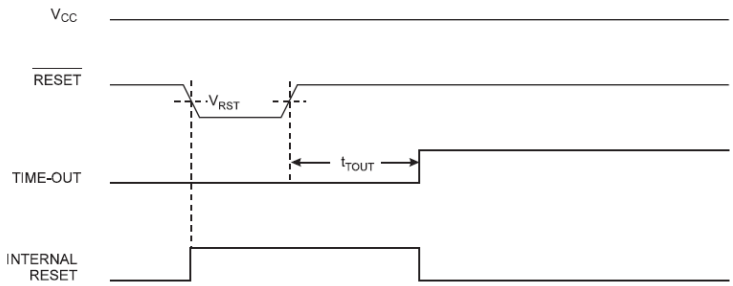
<그림 1.2.8> $\overline{\text{RESET}}$핀을 Vcc에 접속한 경우



<그림 1.2.9> $\overline{\text{RESET}}$핀을 외부리셋 신호에 접속한 경우

(2) 외부 리셋

외부 리셋은 $\overline{\text{RESET}}$핀의 전압이 리셋 문턱 전압(Reset threshold voltage) V_{RST}이하로 최소 펄스 폭이 2.5us이상 지속되었을 시스템 리셋이 되며, 이 보다 펄스 폭이 짧은 경우에는 리셋이 발생하지 않는다.



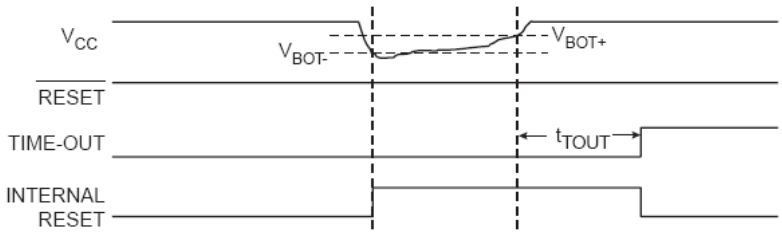
<그림 1.2.10> 외부 리셋에 의한 동작

(3) 저전압 검출 리셋

ATtiny2313에 내장 되어 있는 저전압 검출 회로(Brown-out Detection circuit)는 Vcc의 전압이 트리거 레벨 이하로 2ns이상 지속되면 시스템을 리셋 시킨다. 저전압 검출 트리거 레벨은 <표1.2.15>와 같이 퓨즈비트의 BODLEVEL에 의해 설정할 수 있다.

<표 1.2.15> BODLEVEL 퓨즈

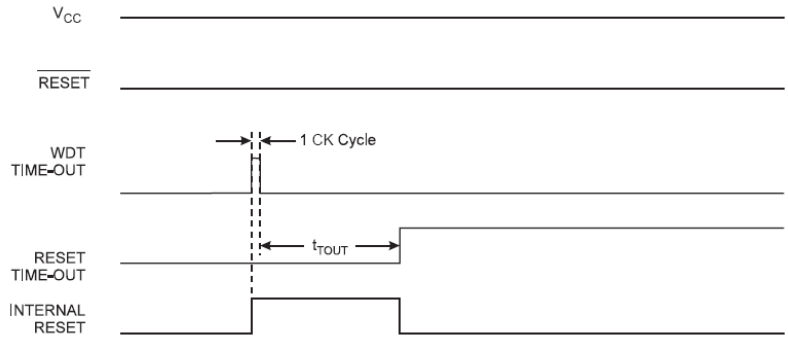
BODLEVEL2~BODLEVEL0	최소 V_{BOT}	대표V_{BOT}	최대V_{BOT}	단위
111	BOD 허용하지 않음			
110		1.8		V
101		2.7		
100		4.3		



<그림 1.2.9> 저전압 검출 리셋의 동작

(4) 워치독 리셋

ATtiny2313의 워치독 타이머가 타임 아웃되었을 때 1클록 사이클의 리셋 펄스를 발생하여 리셋된다.



<그림 1.2.10> 워치독 리셋의 동작

(5) MCU 상태 레지스터

MCU 상태 레지스터(MCU Status Register) MCUSR은 MCU리셋을 발생시키는 리셋 소스에 대한 정보를 제공한다.

비트	7	6	5	4	3	2	1	0	
0x34(0x54)	-	-	-	-	WDRF	BORF	EXTRF	PORF	MCUSR
읽기 / 쓰기	R	R	R	R	R/W	R/W	R/W	R/W	

<그림 1.2.11> MCU 상태 레지스터(MCUSR)

<표 1.2.16> ATtiny2313의 MCU 상태 레지스터(MCUSR)

비트	비트명	기 능
Bit 3	WDRF	· Watchdog Reset Flag : 워치독 리셋이 발생하면, '1'로 되고, 전원-ON 리셋 또는 로직'0'를 써 넣으면 '0'으로 됨.
Bit 2	BORF	· Brown-out Reset Flag : 저전력 검출 리셋이 발생하면, '1'로 되고, 전원-ON 리셋 또는 로직'0'를 써 넣으면 '0'으로 됨.
Bit 1	EXTRF	· External Reset Flag : 외부 리셋이 발생하면, '1'로 되고, 전원-ON 리셋 또는 로직'0'를 써 넣으면 '0'으로 됨.
Bit 0	PORF	· Power-on Reset Flag : 전원-ON 리셋이 발생하면, '1'로 되고, 로직'0'를 써 넣으면 '0'으로 됨.

(6) 내부 기준전압

ATtiny2313은 내부적인 밴드갭(bandgap) 기준 전압을 가지고 있으며, 이것을 저전압 검출이나 아날로그 비교기의 입력에 사용한다. 이 기준전압은 소비전력을 줄이기 위해서 저전압 검출(BOD)가 허용되거나, 밴드갭 기준이 아날로그 비교기에 접속되어 있을 때만 동작한다.

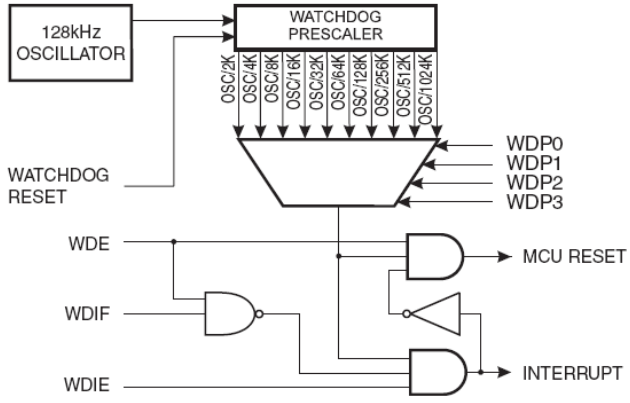
내부 기준전압의 특성은 <표 1.2.17>과 같다.

<표 1.2.17> 내부 기준전압의 특성

기호	파라미터	조건	최소값	대표값	최대값	단위
V_{BG}	밴드갭기준전압	Vcc =2.7V T_A=25℃	1.0	1.1	1.2	V
t_{BG}	밴드갭 기준 시작시간	Vcc =2.7V T_A=25℃		40	70	us
I_{BG}	밴드갭 기준 전류 소비	Vcc =2.7V T_A=25℃		15		uA

5. 워치독 타이머

워치독 타이머(WDT)는 <그림 1.2.12>와 같이 내부에서 독립적으로 만들어지는 128kHz의 오실레이터의 사이클을 카운팅하는 타이머이다. 워치독 타이머의 카운터가 타임아웃값에 도달하면 인터럽트나 시스템 리셋을 발생시킨다.



<그림 1.2.12> 워치독 타이머의 구조

비트	7	6	5	4	3	2	1	0	
0x34(0x54)	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 1.2.13> 워치독 타이머 제어 레지스터(WDTCSR)

<표 1.2.18> ATtiny2313의 워치독 타이머 제어 레지스터(WDTCSR)

비트	비트명	기 능
Bit 7	WDIF	· Watchdog Interrupt Flag : 워치독 타이머가 타임아웃 되었을 때 '1'로 되고, 대응하는 인터럽트 벡터가 실행될 때, 하드웨어에 의해 '0'으로 됨.
Bit 6	WDIE	· Watchdog Interrupt Enable : SREG의 I비트가 '1'일 때, 워치독 인터럽트를 허용.
Bit 4	WDCE	· Watchdog Change Enable : WDE와 프리스케일 비트를 바꿈. WDE 비트를 '0'으로 하고, 프리스케일 비트를 바꾸려면, WDCE가 반드시 '1'이어야 함.
Bit 3	WDE	· Watchdog System Reset Enable : WDRF가 '1'이면 WDE는 항상 '1'.
Bit5, Bit2~Bit0	WDP3~ WDP0	· Watchdog Timer Prescaler : 워치독 타이머의 프리스케일러를 설정.

워치독 타이머는 <표 1.2.19>의 프리스케일러 비트(WDP3～WDP0)에 의해 워치독의 타임아웃 시간을 설정하는 데 사용한다.

<표 1.2.19> 워치독 타이머 프리스케일러 선택

WDP3	WDP2	WDP1	WDP0	WDT 오실레이터 사이클 수	Vcc=5.0V에서 타임아웃
0	0	0	0	2K(2048)사이클	16ms
0	0	0	1	4K(4096)사이클	32ms
0	0	1	0	8K(8192)사이클	64ms
0	0	1	1	16K(16384)사이클	0.125s
0	1	0	0	32K(32768)사이클	0.25s
0	1	0	1	64K(65536)사이클	0.5s
0	1	1	0	128K(131072)사이클	1.0s
0	1	1	1	256K(262144)사이클	2.0s
1	0	0	0	512K(524288)사이클	4.0s
1	0	0	1	1024K(1048576)사이클	8.0s

WDE가 '1'이면, 워치독 타이머는 인터럽트・시스템 리셋 모드이고, 워치독 타이머가 처음으로 타임아웃 되어 WDIF가 '1'로 된다. 이때 대응하는 인터럽트 벡터가 수행되면 WDIE와 WDIF는 '0'으로 된다. 이

것은 인터럽트를 사용하는 동안 워치독 타이머 보안을 유지하는 데 유용하다. 인터럽트·시스템 리셋 모드를 유지하려면, 인터럽트가 종료된 후에 WDIE를 '1'로 해야 한다. 다음 타임아웃 전에 인터럽트를 수행하지 않으면, 시스템 리셋이 적용된다.

<표 1.2.20> 워치독 타이머 제어 레지스터 설정

WDTON	WDE	WDIE	모드	동작
1	0	0	정지모드	-
1	0	1	인터럽트 모드	인터럽트
1	1	0	시스템 리셋 모드	리셋
1	1	1	인터럽트·시스템 리셋 모드	인터럽트 후, 시스템을 리셋함.
0	×	×	시스템 리셋 모드	리셋

6. 인터럽트

ATtiny2313은 총 19가지의 리셋과 인터럽트 벡터를 가지고 있으며, 인터럽트는 SREG 레지스터의 글로벌 인터럽트 허용 비트 I를 '1'로 함으로써 개별적으로 허용할 수 있다.

리셋과 인터럽트 벡터는 <표 1.2.21>과 같이 프로그램 메모리의 최하위 주소에 정의되며, 인터럽트가 발생할 때, 글로벌 인터럽트 허용 I비트가 '0'으로 되고 다른 인터럽트는 허용되지 않는다. 다시 인터럽트를 허용하려면 사용자가 글로벌 인터럽트 허용 I비트를 소프트웨어로 '1'이 되도록 해야 한다.

(1) 리셋과 인터럽트 처리

인터럽트에는 두 가지 종류가 있다. 첫째는 인터럽트 플랙을 '1'로 하는 이벤트에 의해 인터럽트가 발생된다. 이때 프로그램 카운터는 인터럽트 처리 루틴을 실행하기 위해서 해당 인터럽트 벡터를 지시하며, 하드웨어는 대응하는 인터럽트 플랙을 '0'으로 한다. 대응하는 인터럽트 허용 비트가 '0'으로 되는 동안 인터럽트 조건이 발생하면, 인터럽트 플랙은 '1'로 되고 인터럽트가 허용될 때까지 기억되며 플랙은 소프트웨어에 의해 '0'으로 된다. 글로벌 인터럽트 허용 비트가 '0'으로 되어 있는 동안

1개 이상의 인터럽트 조건이 발생하면 대응하는 인터럽트 플랙은 '1'로 되고 글로벌 인터럽트 허용 비트가 '1'이 될 때 까지 기억되며, 인터럽트 우선순위에 따라 실행된다.

둘째는 인터럽트 조건에 따라 인터럽트가 발생한다. 이 인터럽트는 인터럽트 플랙을 필요로 하지 않으며, 인터럽트가 허용되기 전에 인터럽트 조건이 사라지면, 인터럽트가 발생되지 않는다.

인터럽트로부터 탈출할 때는 인터럽트가 발생하기 전의 주 프로그램으로 돌아간다. 상태 레지스터는 인터럽트 루틴으로 들어갈 때 자동으로 저장되지 않으며, 인터럽트 루틴으로부터 돌아올 때에도 자동으로 복구되지 않으므로 소프트웨어로 처리해 주어야 한다.

<표 1.2.21> 리셋과 인터럽트 벡터

벡터번호	프로그램 주소	인터럽트 소스	인터럽트 발생 조건
1	0x0000	RESET	외부핀, 전원-ON리셋, 저전원검출(BOD)리셋, 워치독 리셋
2	0x0001	INT0	외부 인터럽트 요청0
3	0x0002	INT1	외부 인터럽트 요청1
4	0x0003	TIMER1 CAPT	타이머/카운터1 캡쳐 이벤트
5	0x0004	TIMER1 COMPA	타이머/카운터1 비교A
6	0x0005	TIMER1 OVF	타이머/카운터1 오버플로우
7	0x0006	TIMER0 OVF	타이머/카운터0 오버플로우
8	0x0007	USART0, RX	USART0, Rx 완료
9	0x0008	USART0, UDRE	USART0 데이터 레지스터 Empty
10	0x0009	USART0, TX	USART0, Tx 완료
11	0x000A	ANALOG COMP	아날로그 비교기
12	0x000B	PCINT	핀 변화 인터럽트
13	0x000C	TIMER1 COMPB	타이머/카운터1 비교B
14	0x000D	TIMER0 COMPA	타이머/카운터0 비교A
15	0x000E	TIMER0 COMPB	타이머/카운터0 비교B
16	0x000F	USI START	USI 시작 조건
17	0x0010	USI OVERFLOW	USI 오버플로우
18	0x0011	EE READY	EEPROM 준비
19	0x0012	WDT OVERFLOW	워치독 타이머 오버플로우

(2) 인터럽트 응답 시간

허용된 인터럽트에 대한 인터럽트 응답은 최소 4클록 사이클이며, 4클

록 사이클 후에는 해당 인터럽트 처리 루틴에 대한 프로그램 벡터 주소가 실행된다. 4클록 사이클 주기 동안 프로그램 카운터는 스택에 PUSH하고, 벡터는 정상적으로 인터럽트 루틴으로 3 클록 사이클 동안에 점프한다. 다중 사이클 명령을 실행하는 동안 인터럽트가 발생하면, 인터럽트를 수행하기 전에 이 명령을 마친다. MCU가 슬립 모드일 때, 인터럽트가 발생하면, 인터럽트 실행 응답시간은 4 클록 사이클 이상으로 증가한다. 이렇게 응답시간이 증가하는 것은 슬립 모드로부터 기동시간에 추가를 하면 된다.

인터럽트 처리 루틴으로부터 돌아오는 데 4 클록 사이클이 걸린다. 4 클록 사이클 동안 프로그램 카운터(2바이트)는 스택으로부터 POP되고, 스택 포인터는 2만큼 증가되며, SREG 레지스터의 글로벌 인터럽트 허용 비트 I는 '1'로 된다.

(3) 외부 인터럽트

외부 인터럽트는 INT1~INT0 핀 또는 PCINT7~PCINT0 핀을 통해서 입력되는 신호에 의해서 발생되는 인터럽트이다. 인터럽트는 INT1~INT0 핀 또는 PCINT7~PCINT0 핀을 출력으로 설정하여도 발생되며, 이런 기능을 사용하려면 소프트웨어적으로 데이터를 출력하여 인터럽트를 발생시켜야 한다. 핀 변화 인터럽트(pin change interrupt) PCIF는 PCINT7~PCINT0 핀을 토글로 허용하면 인터럽트가 발생한다. PCMSK 레지스터가 핀 변환 인터럽트를 제어하며, PCINT7~PCINT0에 대한 핀 변화 인터럽트가 비동기적으로 검출되므로 슬립 모드나 Idle모드를 해제하는 데 사용할 수 있다.

INT0와 INT1 인터럽트는 입력되는 신호의 상승에지나 하강에지 또는 로우레벨에서 발생되며, 인터럽트 발생 방법은 MCU 제어 레지스터(MCUCR)로 설정한다. INT0 또는 INT1 인터럽트가 허용되고 레벨 인터럽트로 설정되었다면, 핀이 로우(low)로 되었을 때 인터럽트가 발생한다. INT0와 INT1의 상승에지 또는 하강에지 인터럽트를 인식하는 데에는 I/O클록이 필요하며, 로우레벨 인터럽트는 비동기적으로 검출 되므로 슬립 모드나 Idle 모드를 해제 하는 데 사용된다. 또한, I/O 클록은 Idle 모드를 제외한 슬립 모드에서는 정지된다.

로우레벨 인터럽트는 파워-다운 모드의 해제에 사용되며, 이 경우에는 MCU가 정상적인 동작을 하기까지 충분히 긴 로우레벨 인터럽트 신호가 있어야 한다. 기동시간이 끝나기 전에 로우레벨 인터럽트 신호가 사라지면, 파워-다운 모드는 해제되지만 인터럽트는 발생하지 않는다. 기동시간은 SUT0~SUT1와 CKSEL0 퓨즈로 설정한다.

1) MCU 제어 레지스터(MCUCR)

외부 인터럽트 제어 레지스터는 <그림 1.2.14>와 같이 인터럽트를 감지하고 제어하는 제어 비트 ISC11~ISC10와 ISC01~ISC00이 있다.

비트	7	6	5	4	3	2	1	0	
0x35(0x55)	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 1.2.14> MCU 제어 레지스터 MCUCR

외부 인터럽트 1은 SREG I-플랙과 대응하는 인터럽트 마스크가 '1'이면, 외부 핀 INT1의 인터럽트 신호에 의해 동작한다. 외부 INT1핀에 대한 레벨 인터럽트나 에지 인터럽트는 <표1.2.22>의 설정에 따라 발생한다.

<표 1.2.22> 외부 INT1의 인터럽트 방식 설정

ISC11	ISC10	인터럽트 발생 방식
0	0	INT1 핀의 로우레벨 입력이 인터럽트 요청.
0	1	INT1 핀에 대한 논리변화가 인터럽트 요청.
1	0	INT1 핀의 하강에지에서 인터럽트 요청.
1	1	INT1 핀의 상승에지에서 인터럽트 요청.

외부 인터럽트 0은 외부인터럽트 1과 동일하게 동작하므로 레벨 인터럽트나 에지 인터럽트는 <표1.2.23>의 설정에 따라 발생한다.

<표 1.2.23> 외부 INT0의 인터럽트 방식 설정

ISC01	ISC00	인터럽트 발생 방식
0	0	INT0 핀의 로우레벨 입력이 인터럽트 요청.
0	1	INT0 핀에 대한 논리변화가 인터럽트 요청.
1	0	INT0 핀의 하강에지에서 인터럽트 요청.
1	1	INT0 핀의 상승에지에서 인터럽트 요청.

2) 인터럽트 마스크 레지스터(GIMSK)

비트	7	6	5	4	3	2	1	0	
0x3B(0x5B)	INT1	INT0	PCIE	-	-	-	-	-	GIMSK
읽기 / 쓰기	R/W	R/W	R/W	R	R	R	R	R	

<그림 1.2.15> 인터럽트 마스크 레지스터(GIMSK)

<표 1.2.24> 인터럽트 마스크 레지스터(GIMSK)

비트	비트명	기 능
Bit 7	INT1	· External Interrupt Request 1 Enable : INT1=1이고 SREG의 I=1 일때, 외부 인터럽트 핀 허용.
Bit 6	INT0	· External Interrupt Request 0 Enable : INT1=1이고 SREG의 I=1 일때, 외부 인터럽트 핀 허용.
Bit 5	PCIE	· Pin Change Interrupt Enable : PCIE=1이고 SREG의 I=1 일때, 핀 변화 인터럽트 1 허용. PCINT0 ~ PCINT7핀은 PCMSK레지스터에 의해 개별적인 허용이 가능.

3) 외부 인터럽트 플랙 레지스터(EIFR)

비트	7	6	5	4	3	2	1	0	
0x3A(0x5A)	INTF1	INTF0	PCIF	-	-	-	-	-	EIFR
읽기 / 쓰기	R/W	R/W	R/W	R	R	R	R	R	

<그림 1.2.16> 외부 인터럽트 플랙 레지스터(EIFR)

<표 1.2.25> 외부 인터럽트 플랙 레지스터(EIFR)

비트	비트명	기 능
Bit 7	INTF1	· External Interrupt Flag 1 : INT1핀의 에지나 논리가 변할 때, 인터럽트 요청하면, INTF1=1. SREG의 I=1이고, GIMSK의 INT1=1이면, 대응하는 인터럽트 벡터로 점프. 인터럽트 루틴이 실행되면, 플랙=0로 됨.
Bit 6	INTF0	· External Interrupt Flag 0 : INT0핀의 에지나 논리가 변할 때, 인터럽트 요청하면, INTF0=1. SREG의 I=1이고 GIMSK의 INT0=1이면, 대응하는 인터럽트 벡터로 점프. 인터럽트 루틴이 실행되면, 플랙=0로 됨.
Bit 5	PCIF	· Pin Change Interrupt Flag : PCINT0 ~ PCINT7핀에 논리 변화가 있을 때, 인터럽트 요청. SREG의 I=1이고, GIMSK의 PCIE=1이면, 대응하는 인터럽트 벡터로 점프. 인터럽트 루틴이 실행되면, 플랙=0로 됨.

4) 핀 변화 마스크 레지스터(PCMSK)

비트	7	6	5	4	3	2	1	0	
0x20(0x40)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK
읽기 / 쓰기	R/W	R/W	R/W	R	R	R	R	R	

<그림 1.2.17> 핀 변화 마스크 레지스터(PCMSK)

<표 1.2.26> 핀 변화 마스크 레지스터(PCMSK)

비트	비트명	기 능
Bit 7~Bit0	PCINT7~ PCINT0	· Pin Change Enable Mask 7~0 : 핀 변화 인터럽트를 허용할 PCINT0~PCINT7를 선택. PCINT0~PCINT7을 '1'로 하고, GIMSK PCIE='1'하면, 핀 변화 인터럽트가 허용됨. PCINT0~PCINT7를 '0'으로 하면 핀 변화 인터럽트가 해제됨.


0604
ATTINY2313-20PU

제 2 장

ATtiny2313의 I/O 기능


0604
ATTINY2313-20PU

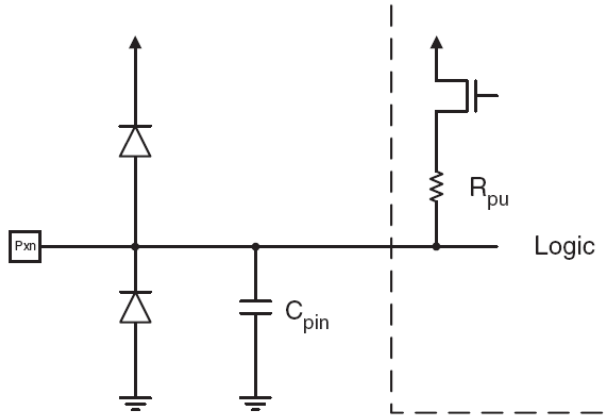
2.1 I/O 포트

1. I/O 포트의 구조

ATtiny2313는 모두 18개의 8비트 양방향 I/O포트(PORTA는 3비트 , PORTB는 8비트, PORTD는 7비트)를 가지고 있다. 모든 I/O포트는 입출력 방향을 개별적으로 변경할 수 있는 Read-Modify-Write 동작이 가능하다.

각 포트는 소스전류 드라이브(source drive)능력과 싱크전류 드라이브(sink drive)능력이 대칭적이며, I/O핀 당 전류는 40mA 정도 이므로 LED를 직접 구동할 수 있다.

각 I/O핀에는 <그림 2.1.1>과 같이 개별적으로 사용 여부를 선택할 수 있는 20[kΩ]~50[kΩ]의 풀업저항 R_{PU}가 있고, Vcc와 GND사이에 보호용 다이오드가 있다.



<그림 2.1.1> I/O핀의 구조

ATtiny2313의 각 포트에 데이터 레지스터(PORTx), 데이터 방향 레지스터(DDRx), 포트 입력 핀(PINx) 등이 3개의 I/O 메모리 주소 영역이 있다. (여기서, 아래 첨자 x는 포트번호를 의미함.) 포트 입력 핀 I/O는 읽기(read) 만 가능한 반면에, 데이터 레지스터와 데이터 방향 레지스터는 읽기/쓰기가 가능하다. PINx 레지스터의 비트를 논리 '1'로 하면, 데이터 레지스터의 해당 비트가 토글된다.

MCUCR의 PUD비트를 '1'로 하여 해제하면, 모든 포트의 모든 핀에 대한 풀업 저항 기능이 해제된다. PUD 비트를 '0'으로 설정하면 내부 풀업 저항 기능을 사용할 수 있다.

비트	7	6	5	4	3	2	1	0	
0x35(0x55)	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 2.1.2> MCU 제어 레지스터 MCUCR

<표 2.1.1> MCU 제어 레지스터(MCUCR)

비트	비트명	기 능
Bit 7	PUD	· Pull-up Disable : PUD=1이면, I/O포트의 풀업 저항기능 해제.

각 포트의 핀은 DDxn, PORTxn, PINxn 등 3개의 레지스터로 구성되어 있다. 여기서, 아래첨자 x는 포트번호, n은 비트번호이다. 예를 들어, PORTB3는 포트B(B가 아래첨자 x에 해당됨)의 3번 비트(3이 아래첨자 n에 해당됨)를 의미한다.

DDRx 레지스터의 DDxn비트는 핀의 방향을 설정한다. DDxn이 '1'이면, Pxn은 출력 핀으로 설정되며, DDxn이 '0'이면 Pxn은 입력 핀으로 설정된다.

PORTxn이 '1'이면 핀이 입력 핀으로 설정되었을 때, 풀업 저항이 동작한다. 풀업 저항 기능을 해제하려면, PORTxn을 '0'으로 하거나 핀을 출력 핀으로 설정해야 한다. 풀업 저항 기능을 해제한 상태에서 리셋이 되면 포트 핀은 3스테이트 상태로 된다.

PORTxn이 '1'이면 핀이 출력 핀으로 설정되었을 때, 포트 핀은 하이(high)출력이다. PORTxn이 '0'이면, 핀이 출력 핀으로 설정되었을 때, 포트 핀은 로우(low) 출력이다.

<표 2.1.2> I/O 포트 핀 동작 설정

DDxn	PORTxn	PUD	I/O	풀업	설 명
0	0	×	입력	안됨	3-상태(Hi-Z)
0	1	0	입력	됨	Pxn는 전류원
0	1	1	입력	안됨	3-상태(Hi-Z)
1	0	×	출력	안됨	출력 로우(sink)
1	1	×	출력	안됨	출력 하이(source)

2. I/O 포트의 레지스터와 부가기능

ATtiny2313의 I/O포트는 기본적인 입출력 포트로서의 기능 이외에 부가적인 기능을 가지고 있다.

(1) 포트A의 레지스터와 부가 기능

포트A는 <표 2.1.3>과 같이 입출력 포트로서의 기능 이외에 리셋 및 크리스탈 오실레이터를 접속할 수 있다.

<표 2.1.3> 포트A의 부가기능

포트 핀	부가 기능
PA2	리셋, debugWIRE
PA1	XTAL2
PA0	XTAL1

① 포트A 데이터 레지스터 – PORTA

비트	7	6	5	4	3	2	1	0	
0x1B(0x3B)	-	-	-	-	-	PORTA2	PORTA1	PORTA0	PORTA
읽기 / 쓰기	R	R	R	R	R	R/W	R/W	R/W	

② 포트A 데이터 방향 레지스터 – DDRA

비트	7	6	5	4	3	2	1	0	
0x1A(0x3A)	-	-	-	-	-	DDRA2	DDRA1	DDRA0	DDRA
읽기 / 쓰기	R	R	R	R	R	R/W	R/W	R/W	

③ 포트A 입력핀 어드레스 – PINA

비트	7	6	5	4	3	2	1	0	
0x19(0x39)	-	-	-	-	-	PINA2	PINA1	PINA0	PINA
읽기 / 쓰기	R	R	R	R	R	R/W	R/W	R/W	

(2) 포트B의 레지스터와 부가 기능

포트B는 <표 2.1.4>과 같이 입출력 포트로서의 기능 이외에 3-와이어 모드, 2-와이어 모드, 타이머/카운터, 아날로그 비교기 등의 부가적인 기능을 수행한다.

<표 2.1.4> 포트B의 부가기능

포트 핀	부가 기능
PB7	· USCK : 3-와이어 모드 범용 직렬 인터페이스(USI) 클록 · SCL : USI 2-와이어 모드를 위한 2-와이어 직렬 클록 · PCINT7 : 핀 변화 인터럽트 소스 7
PB6	· DO : 3-와이어 모드 범용 직렬 인터페이스 데이터 출력 · PCINT6 : 핀 변화 인터럽트 소스 6
BP5	· DI : 3-와이어 모드 범용 직렬 인터페이스 데이터 입력 · SDA : 2-와이어 모드 직력 인터페이스 데이터 · PCINT5 : 핀 변화 인터럽트 소스 5
PB4	· OC1B : 타이머/카운터1의 출력 비교 매치 B 출력 · PCINT4 : 핀 변화 인터럽트 소스 4
PB3	· OC1A : 타이머/카운터1의 출력 비교 매치 A 출력 · PCINT3 : 핀 변화 인터럽트 소스 3
PB2	· OC0A : 타이머/카운터0의 출력 비교 매치 A 출력 · PCINT2 : 핀 변화 인터럽트 소스 2
PB1	· AIN1 : 아날로그 비교기 (-) 입력 · PCINT1 : 핀 변화 인터럽트 소스 1
PB0	· AIN0 : 아날로그 비교기 (+) 입력 · PCINT0 : 핀 변화 인터럽트 소스 0

① 포트B 데이터 레지스터 – PORTB

비트	7	6	5	4	3	2	1	0	
0x18(0x38)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

② 포트B 데이터 방향 레지스터 – DDRB

비트	7	6	5	4	3	2	1	0	
0x17(0x37)	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0	DDRB
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

③ 포트B 입력핀 어드레스 – PINB

비트	7	6	5	4	3	2	1	0	
0x16(0x36)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

(3) 포트D의 레지스터와 부가 기능

포트D는 <표 2.1.5>와 같이 입출력 포트로서의 기능 이외에 타이머/카운터, 외부 인터럽트 USART 등의 부가적인 기능을 수행한다.

<표 2.1.5> 포트D의 부가기능

포트 핀	부가 기능
PD6	· ICP : 타이머/카운터1 입력 캡쳐 핀
BD5	· OC0B : 타이머/카운터0의 출력 비교 매치 B 출력 · T1 : 타이머/카운터1의 외부 카운터 클록 입력
PD4	· T0 : 타이머/카운터0의 외부 카운터 클록 입력
PD3	· INT1 : 외부 인터럽트 소스1
PD2	· INT0 : 외부 인터럽트 소스0 · XCK : 동기전송 모드에서 만 사용되는 USART 전송 클록 · CKOUT : 시스템 클록 출력
PD1	· TXD : UART 데이터 송신기
PD0	· RXD : UART 데이터 수신기

① 포트D 데이터 레지스터 – PORTD

비트	7	6	5	4	3	2	1	0	
0x12(0x32)	-	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
읽기 / 쓰기	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

② 포트D 데이터 방향 레지스터 – DDRD

비트	7	6	5	4	3	2	1	0	
0x11(0x31)	-	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0	DDRD
읽기 / 쓰기	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

③ 포트D 입력핀 어드레스 – PIND

비트	7	6	5	4	3	2	1	0	
0x10(0x306)	-	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
읽기 / 쓰기	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

2.2 타이머/카운터 및 PWM출력

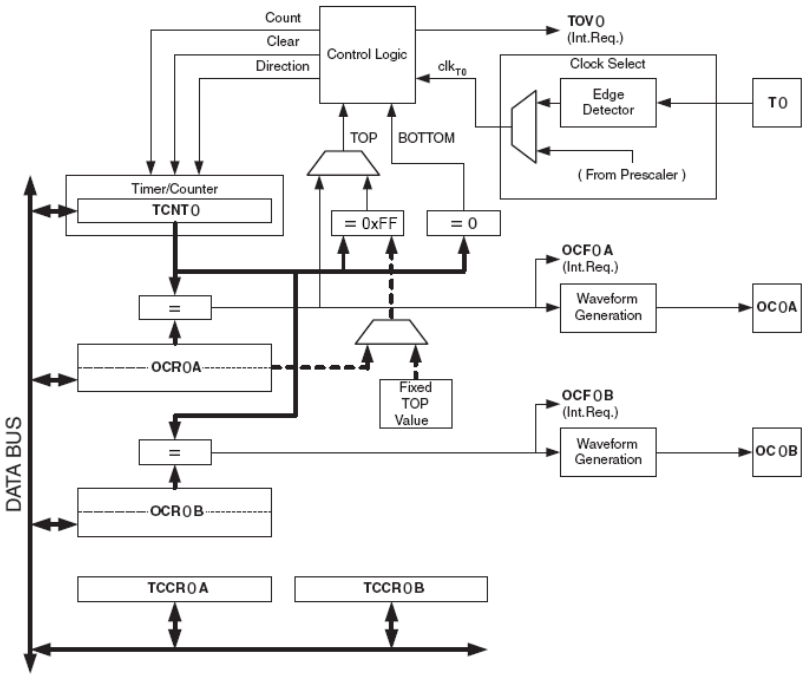
ATtiny2313은 8비트 PWM출력 타이머/카운터0와 16비트 PWM 출력 타이머/카운터1을 가지고 있다. 이들 타이머/카운터의 특징은 <표 2.2.1>과 같다.

<표 2.2.1> ATtiny2313 타이머/카운터의 특징

	타이머/카운터0	타이머/카운터1
기본구조	8비트	16비트
타이머입력	$clk_{I/O}$ (= clk_{CPU})	$clk_{I/O}$ (= clk_{CPU})
카운터입력	T0	T1
타이머 프리스케일러	/ 1, 8, 64, 256, 1024	/ 1, 8, 64, 256, 1024
카운터 프리스케일러		
관련 레지스터	TCNT0 OCR0A, OCR0B TCCR0A, TCCR0B TIFR, TIMSK	TCNT1H/TCNT1L, OCR1AH/OCR1AL, OCR1BH/OCR1BL, ICR1H/ICR1L, TCCR1A, TCCR1B, TCCR1C TIFR, TIMSK
동작모드	Normal, CTC, Fast PWM, Phase Correct PWM	Normal, CTC, Fast PWM, Phase Correct PWM, Phase and Frequency Correct PWM
입력신호단자	T0	T1, ICP
출력신호단자 (PWM가능)	OC0A, OC0B	OC1A, OC1B
인터럽트	Overflow, Output Compare Match	Overflow, Output Compare Match A/B, Input Capture

1. 8비트 타이머/카운터 0

타이머/카운터0은 2개의 독립된 출력 비교기(compare unit)와 PWM 출력을 가지는 범용 8비트 타이머/카운터 모듈이다. 타이머/카운터0의 구조는 <그림 2.2.1>과 같다.



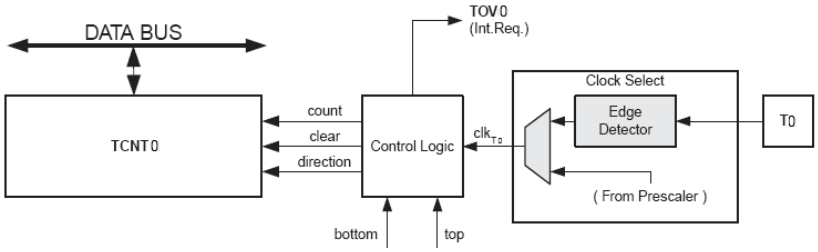
<그림 2.2.1> 타이머/카운터0의 블록도

타이머/카운터0는 클록신호 clk_{T0}에 의해 8비트 타이머/카운터 TCNT0가 상향 카운터(up counter)로 동작하여 카운터의 최대값인 0xFF에 이르면 0x00으로 되면서 오버플로우가 생겨 TOV0신호에 의해 오버플로우 인터럽트가 발생한다. 카운터 값은 출력비교 레지스터 OCR0x과 비교되고 비교한 값이 같으면 내부적으로 OCF0x신호에 의해 인터럽트가 발생되고, 외부적으로 OC0x 단자에 신호가 출력되도록 설정할 수 있다. (여기서, x는 A 또는 B를 나타냄.)

(1) 카운터부

타이머/카운터0의 카운터부는 프리스케일러를 통한 내부 클록 입력과 외부 클록T_0, 상향/하향 카운터로 사용할 수 있는 타이머/카운터 TCNT0 등으로 <그림2.2.2>와 같이 구성된다. 타이머/카운터0의 외부 클록은 T_0 단자로 받아 카운터 동작을 수행한다. 외부 클록 T_0에 대해 카운터가 상승에지에서 동작할 것인지 하강에지에 동작할 것인지는 타이머/카운터0 제어 레지스터B인 TCCR0B의 CS02~CS00비트로 설정한다.

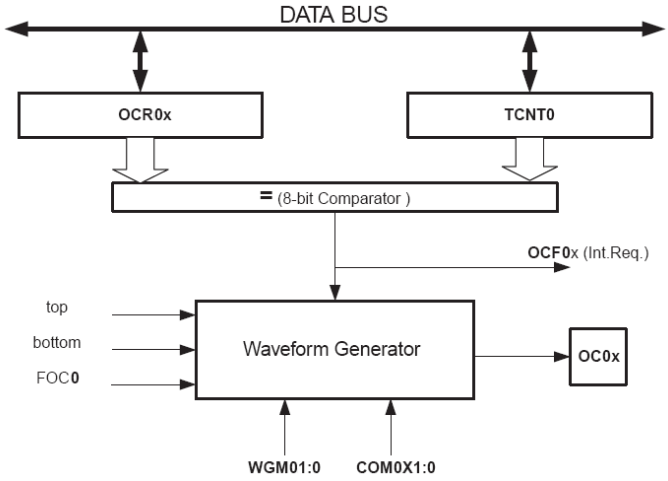
타이머/카운터0가 타이머로 동작할 때는 시스템 클록 주파수를 프리스케일러를 통해 1, 8, 64, 256, 1024의 분주비로 분주하여 클록 소스로 사용할 수도 있고, 클록 소스 입력을 차단할 수도 있다.



<그림 2.2.2>타이머/카운터0의 카운터부 블록도

(2) 출력 비교기부

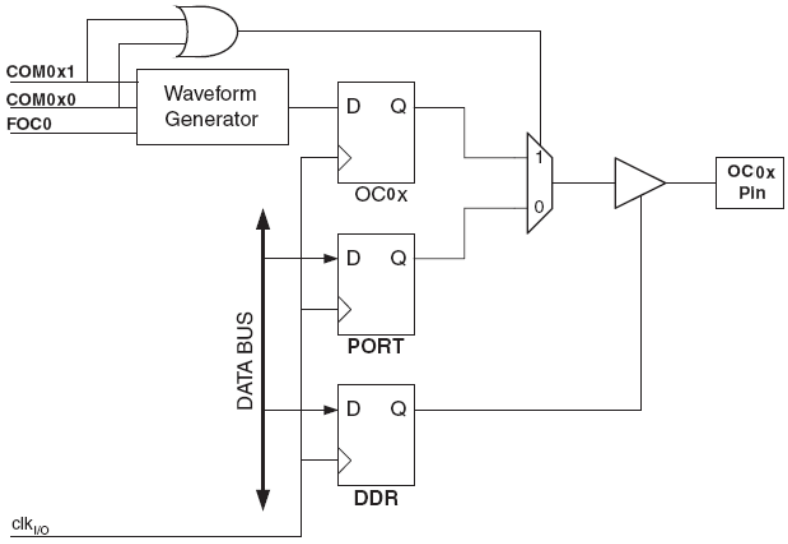
타이머/카운터0에 입력되는 클록 신호 clk_{T0}는 타이머/카운터 TCNT0를 동작시키고 동작중에 카운터 값을 출력비교 레지스터 OCR0x와 비교하여 값이 같아지면, 그 다음 주기에 내부적으로 OCF0x신호에 의해 출력비교 인터럽트를 발생하고, 외부적으로는 OC0x단자에 신호가 출력되도록 설정할 수 있다. OC0x단자로 출력되는 신호는 출력비교 레지스터 OCR0x에 의해 원하는 PWM 신호를 출력할 수 있다. (여기서, x는 A 또는 B를 나타냄.)



<그림 2.2.3> 타이머/카운터0의 출력 비교기부 블록도

(3) 비교 매치 출력부

타이머/카운터0는 <그림 2.2.4>와 같이 타이머/카운터 TCNT0와 출력 비교 레지스터 OCR0x을 비교하여 일치하면 타이머/카운터 제어 레지스터A TCCR0A의 COM0x1와 COM0x0 비트를 설정하여 파형 발생기를 통해 출력할 비교출력 모드를 선택한다. (여기서, x는 A 또는 B를 나타냄.) OC0x단자(ATtiny2313에서는 PB2와 PD5에 해당)를 타이머/카운터의 비교출력 단자로 사용하려면 데이터 방향 레지스터(DDR)를 출력방향으로 설정해야 한다.



<그림 2.2.4> 타이머/카운터0의 비교 매치 출력부 블록도

2. 타이머/카운터 0의 동작 모드

타이머/카운터0의 동작모드는 타이머/카운터 제어 레지스터 TCCR0x의 파형발생 모드(WGM01～WGM00)비트에 의해 결정되며, 출력신호의 동작은 타이머/카운터 제어 레지스터 TCCR0A의 비교출력 모드(COM01～COM00)비트에 의해 OC0x단자로 출력될 파형이 결정된다.

타이머/카운터0의 동작 모드는 일반모드, CTC모드, 고속 PWM모드, 위상 보정PWM모드 등 4가지가 있다.

(1) 일반모드

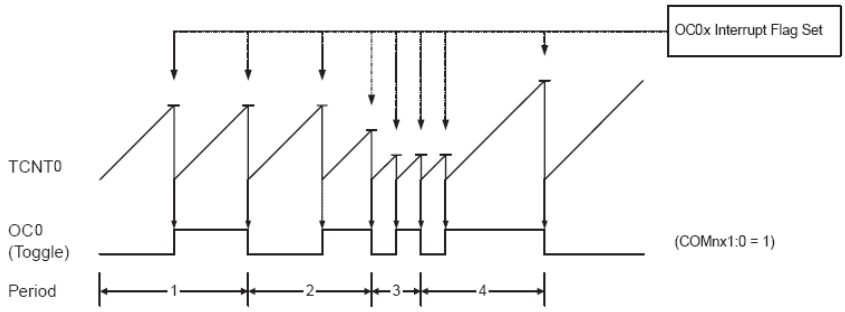
타이머/카운터0의 일반모드(Normal Mode)는 타이머/카운터0 제어 레지스터 TCCR0x의 파형발생모드(WGM02~WGM00)를 '0'으로 설정한다. 이 모드에서 카운터는 상향 카운트만을 수행한다. 또한, 카운트 중에 클리어가 되지 않으며 클록 입력에 의해 8비트의 최대 카운트값인 0xFF가 되면 카운터값이 0x00으로 되어 카운트 동작을 반복한다. 여기서, x는 A 또는 B를 나타냄.)

타이머/카운터 레지스터 TCNT0의 값이 0x00로 됨과 동시에 타이머/카운터 오버플로우 플랙 비트 TOV0가 '1'로 되면서 오버플로우 인터럽트가 발생한다.

일반모드에서 출력비교부는 인터럽트를 발생시키는 데 사용할 수 있으나, 파형을 발생시키기 위한 출력비교는 CPU 타임을 너무 많이 차지하므로 권장되지 않는다.

(2) CTC모드

타이머/카운터0의 CTC모드(Clear Timer on Compare Match Mode)는 타이머/카운터0 제어 레지스터 TCCR0x의 파형발생모드(WGM02~WGM00)를 '2'로 설정한다. 타이머/카운터 레지스터 TCNT0의 값이 출력비교 레지스터 OCR0x와 일치하면, TCNT0는 '0'으로 클리어 된다. 카운터의 최고값은 출력비교 레지스터 OCR0x로 지정하므로 OCR0x를 카운터의 계수 범위를 조정하는 데 사용한다. (여기서, x는 A 또는 B를 나타냄.)



<그림 2.2.5> 타이머/카운터0의 CTC모드 동작

CTC모드에서 OC0x단자로 파형을 출력하려면, 타이머/카운터0 제어 레지스터 TCCR0x의 비교출력 모드비트(COM0x1～COM0x0)를 '1'로 설정하여 토글모드로 하면, OC0x단자로 파형이 출력된다. OC0x단자로 출력되는 파형의 출력 주파수는 다음 식으로 계산된다.

$$f_{OC0x} = \frac{f_{clk_{I/O}}}{2 \cdot N \cdot (1 + OCR0x)}$$

여기서, N은 타이머/카운터0의 프리스케일러 분주비(1, 8, 64, 256, 1024).

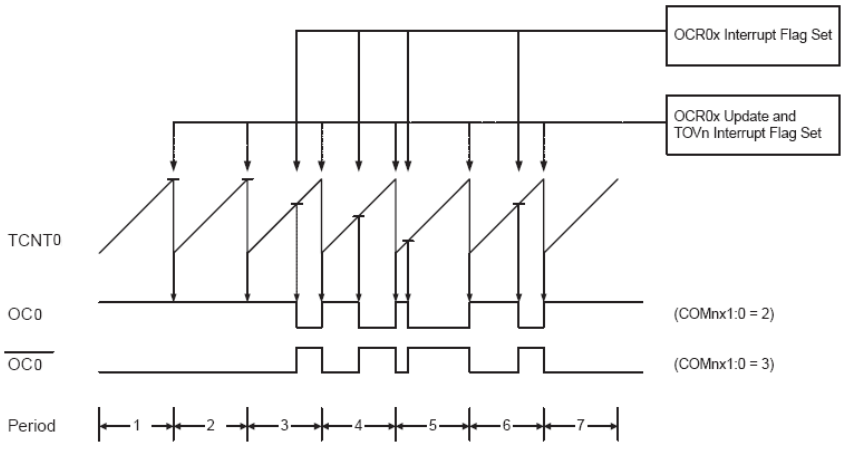
(3) 고속 PWM모드

고속 PWM모드(Fast PWM Mode)는 타이머/카운터 레지스터 TCCR0x에서 파형발생 모드(WGM01～WGM00)를 "3" 또는 "7"로 설정하여 높은 주파수의 PWM 출력 파형을 발생시킨다. 타이머/카운터 레지스터 TCNT0의 카운트 동작이 0x00에서 0xFF까지 반복적으로 수행되며, 이 값은 출력비교 레지스터 OCR0x의 값과 비교되어 일치하면 OC0x 출력신호가 '0'이되고 오버플로우가 발생되면 OC0x 출력신호는 '1'이 된다.

OC0x단자를 통해 출력되는 PWM 출력신호의 주파수는 다음과 같다.

$$f_{OC0xPWM} = \frac{f_{clk_{I/O}}}{N \cdot 256}$$

여기서, N은 타이머/카운터0의 프리스케일러 분주비(1, 8, 64, 256, 1024).



<그림 2.2.6> 타이머/카운터0의 고속 PWM 모드 동작

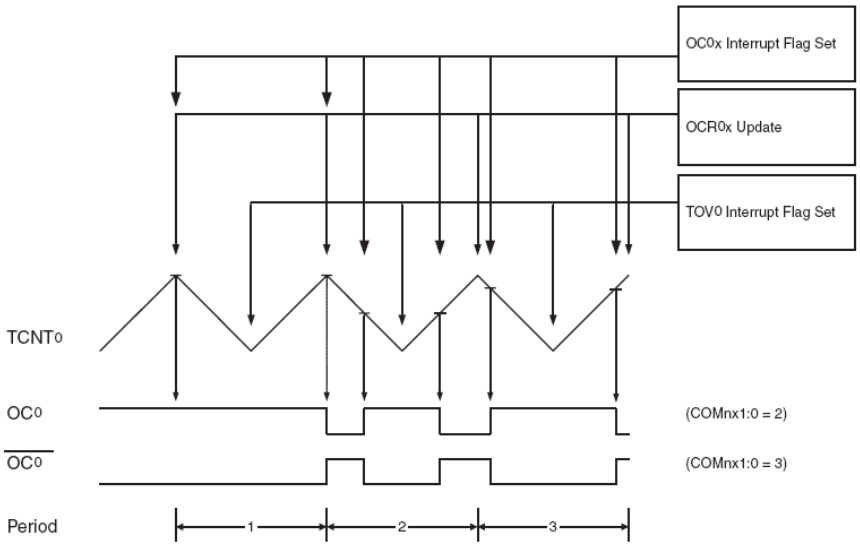
(4) 위상 보정 PWM모드

위상 보정 PWM모드(Phase Correct PWM Mode)는 타이머/카운터 레지스터 TCCRx의 파형발생모드(WGM03~WGM00)를 “1” 또는 “5”로 설정하여 높은 분해능의 PWM 출력 파형을 발생시킨다. 위상 보정 PWM 모드에서는 타이머/카운터 레지스터 TCNT0의 카운트 동작이 0x00에서 0xFF로 증가했다가, 다시 0xFF에서 0x00으로 감소하는 동작을 반복하며, TCNT0의 값은 출력비교 레지스터 OCR0x의 값과 비교하여 0xFF로 증가하는 도중에 일치하면 OC0x 출력신호는 ‘0’이되고 0x00으로 감소하는 도중에 일치하면 OC0x 출력신호는 ‘1’이 된다.

위상 보정 PWM모드는 고속 PWM 모드에 비해 주파수는 약 1/2로 낮아지지만, 고속 PWM 모드의 튜티비 분해능 보다 2배 정도 높아진다. OC0x단자를 통해 출력되는 PWM 출력신호의 주파수는 다음과 같다.

$$f_{OC0xPCPWM} = \frac{f_{clk_{I/O}}}{N \cdot 510}$$

여기서, N은 타이머/카운터0의 프리스케일러 분주비(1, 8, 64, 256, 1024).



<그림 2.2.7> 위상 보정 PWM 모드의 동작

3. 타이머/카운터 0의 레지스터

(1) 타이머/카운터0 제어 레지스터(TCCR0x)

TCCR0x 레지스터는 타이머/카운터0의 동작모드를 설정하고 프리스케일러의 분주비를 설정한다.

비트	7	6	5	4	3	2	1	0	
0x30(0x50)	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00	TCCR0A
읽기 / 쓰기	R/W	R/W	R/W	R/W	R	R	R/W	R/W	

<그림 2.2.8> 타이머/카운터0 제어 레지스터A(TCCR0A)

<그림 2.2.8>에서 비교 매치 출력 x 모드(Compare Match Output x Mode)의 COM0x1와 COM0x0비트는 OC0x 핀의 동작을 설정하며, 파형 발생모드(Waveform Generation Mode)의 WGM01과 WGM00비트는 타이머/카운터0 제어 레지스터B의 WGM02 비트와 조합이 되어 타이머/카운터0의 동작 모드를 설정한다.

비트	7	6	5	4	3	2	1	0	
0x33(0x53)	FOC0A	FOC0B	-	-	WGM01	CS02	CS01	CS00	TCCR0B
읽기 / 쓰기	W	W	R	R	R/W	R/W	R/W	R/W	

<그림 2.2.9> 타이머/카운터0 제어 레지스터B(TCCR0B)

<그림 2.2.9>에서 강제출력 비교 x(Force Output Compare x)의 FOC0x 비트는 PWM모드가 아닌 경우에만 유효하며, TCCR0B 레지스터가 PWM모드로 동작 중일 때는 FOC0x비트를 “1”로 해야 한다. FOC0x가 “1”로 설정되면 OC0x 단자에 출력비교 매치된 값을 출력으로 내보내며, 출력신호의 동작은 COM0x1~COM0x0비트로 설정한다. FOC0x는 인터럽트를 발생시키지도 않으며, CTC모드에서 OCR0A의 값이 0xFF가 되더라도 “0”으로 되지 않는다. FOC0x비트는 항상 “0”로 설정한다.

클록 선택(Clock Select)의 CS02~CS00비트는 내부 클록을 사용할 것인지, 외부 클록을 사용할 것인지를 선택하거나 내부 클록의 경우 클록의 분주비를 설정한다.

<표 2.2.2>는 파형발생모드(WGM02~WGM00)비트에 의한 동작모드 설정하는 것이며, <표 2.2.3>은 비교출력모드(COM0x1~COM0x0)

<표 2.2.2> WGM02~WGM00 비트에 의한 동작모드 설정

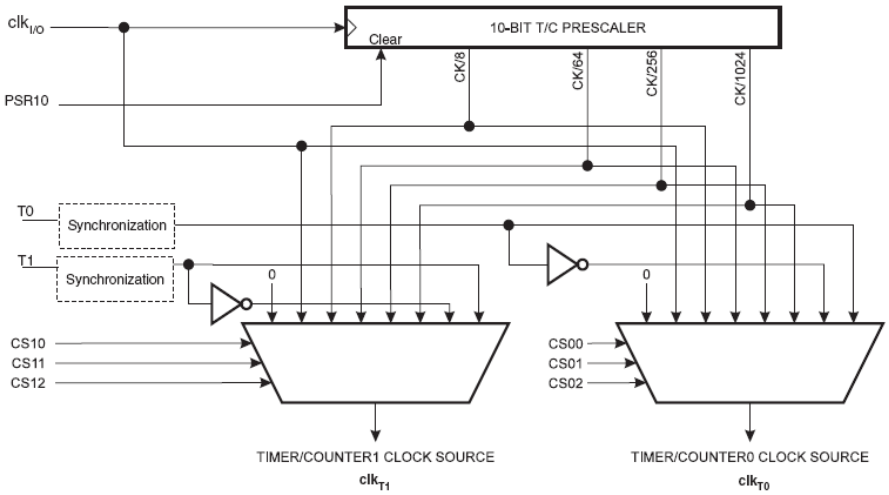
모드	WGM02	WGM01	WGM00	타이머/카운터 동작모드	최대값	OCRx레지스터의 업데이트 시점	TOV 플랙의 셋트 시점
0	0	0	0	일반	0xFF	설정즉시	0xFF
1	0	0	1	PWM, 위상보정	0xFF	0xFF	0x00
2	0	1	0	CTC	OCR0A	설정즉시	0xFF
3	0	1	1	고속PWM	0xFF	0xFF	0xFF
4	1	0	0	-	-	-	-
5	1	0	1	PWM, 위상보정	OCR0A	OCR0A값	0x00
6	1	1	0	-	-	-	-
7	1	1	1	고속PWM	OCR0A	OCR0A값	OCR0A값

<표 2.2.3> COM0x1~COM0x0비트에 의한 OC0x핀 기능 설정

모드	COM0x1	COM0x0	OC0x핀의 기능
PWM모드가 아닌 경우	0	0	정상적인 I/O포트로 동작, OC0x 출력차단
	0	1	비교 매치에서 OC0x 출력을 토글
	1	0	비교 매치에서 OC0x = 0
	1	1	비교 매치에서 OC0x = 1
고속 PWM 모드	0	0	정상적인 I/O포트로 동작, OC0x 출력차단
	0	1	WGM02=0, 정상적인I/O포트로 동작, OC0x 출력차단 WGM02=1, 비교 매치에서 OC0x 출력을 토글 (COM0B1, COM0B0비트의 경우 사용안함)
	1	0	비교 매치에서 OC0x=0, 0xFF에서 OC0x=1
	1	1	비교 매치에서 OC0x=1, 0xFF에서 OC0x=0
위상보정PWM 모드	0	0	정상적인 I/O포트로 동작, OC0x 출력차단
	0	1	WGM02=0, 정상적인I/O포트로 동작, OC0x 출력차단 WGM02=1, 비교 매치에서 OC0x 출력을 토글 (COM0B1, COM0B0비트의 경우 사용안함)
	1	0	상향 카운터 비교 매치에서 OC0x=0, 하향 카운터 비교 매치에서 OC0x=1
	1	1	상향 카운터 비교 매치에서 OC0x=1, 하향 카운터 비교 매치에서 OC0x=0

<표 2.2.4> 타이머/카운터0의 클록소스 선택 비트 설정

CS02	CS01	CS00	클록 소스
0	0	0	클록소스 없음(타이머/카운터 정지)
0	0	1	$clk_{I/O}$ / 1
0	1	0	$clk_{I/O}$ / 8
0	1	1	$clk_{I/O}$ / 64
1	0	0	$clk_{I/O}$ / 256
1	0	1	$clk_{I/O}$ / 1024
1	1	0	T_0핀에 외부 클록소스(하강에지)
1	1	1	T_0핀에 외부 클록소스(상승에지)



<그림 2.2.10> 타이머/카운터0와 타이머/카운터1에 대한 프리스케일러

(2) 타이머/카운터0 레지스터(TCNT0)

타이머/카운터0 레지스터 TCNT0(Timer/Counter Register 0)는 타이머/카운터0의 8비트 카운터값을 저장하고 있으며, 읽기와 쓰기 동작을 할 수 있지만 카운터로 동작하고 있는 동안에 값을 수정하면, TCNT0값과 OCR0x레지스터 사이의 비교 매치에 오류가 발생할 수 있다.

비트	7	6	5	4	3	2	1	0	
0x32(0x52)	TCNT0 [7:0]								TCNT0
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 2.2.11> 타이머/카운터0 레지스터B(TCNT0)

(3) 타이머/카운터0 출력비교 레지스터(OCR0x)

타이머/카운터0 출력비교 레지스터 OCR0x(Output Compare Register x)는 타이머/카운터 레지스터 TCNT0값과 비교하여 OC0x단자에 출력신호를 내보내기 위한 값을 저장하고 있다.

비트	7	6	5	4	3	2	1	0	
0x36(0x56)	OCR0A [7:0]								OCR0A
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 2.2.12> 타이머/카운터0 출력비교 레지스터(OCR0A)

비트	7	6	5	4	3	2	1	0	
0x3C(0x5C)	OCR0B [7:0]								OCR0B
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 2.2.13> 타이머/카운터0 출력비교 레지스터(OCR0B)

(4) 타이머/카운터 인터럽트 마스크 레지스터(TIMSK)

타이머/카운터 인터럽트 마스크 레지스터(Timer/Counter Interrupt Mask Register)는 타이머/카운터0가 발생하는 인터럽트를 개별적으로 허용한다.

비트	7	6	5	4	3	2	1	0	
0x39(0x59)	TOIE1	OCIE1A	OCIE1B	-	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
읽기 / 쓰기	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	

<그림 2.2.14> 타이머/카운터 인터럽트 마스크 레지스터(TIMSK)

<표 2.2.5> 타이머/카운터 인터럽트 마스크 레지스터(TIMSK) 기능

비트	비트명	기 능
Bit 2	OCIE0B	· Timer/Counter0 Output Compare Match B Interrupt Enable : OCIE0B=1이고, 상태 레지스터 SREG의 I=1 일때, 타이머/카운터 비교 매치 B 인터럽트 허용. 타이머/카운터0의 출력비교 인터럽트가 발생되어 TIFR레지스터의 OCF0B=1이면, 인터럽트가 처리됨.
Bit 1	TOIE0	· Timer/Counter0 Overflow Interrupt Enbale : TOIE0=1이고, 상태 레지스터 SREG의 I=1 일때, 타이머/카운터0 오버플로우 인터럽트 허용. 타이머/카운터0의 오버플로우 인터럽트가 발생되어 TIFR 레지스터의 TOV0=1이면, 인터럽트가 처리됨.
Bit 0	OCIE0A	· Timer/Counter0 Output Compare Match A Interrupt Enable : OCIE0A=1이고, 상태 레지스터 SREG의 I=1 일때, 타이머/카운터 비교 매치 A 인터럽트 허용. 타이머/카운터0의 출력비교 인터럽트가 발생되어 TIFR레지스터의 OCF0A=1이면, 인터럽트가 처리됨.

(5) 타이머/카운터 인터럽트 플랙 레지스터(TIFR)

타이머/카운터 인터럽트 플랙 레지스터(Timer/Counter Interrupt Flag Register)는 타이머/카운터0가 발생하는 인터럽트 플랙을 저장한다.

비트	7	6	5	4	3	2	1	0	
0x38(0x58)	TOV1	OCF1A	OCF1B	-	ICF1	OCF0B	TOV0	OCF0A	TIFR
읽기 / 쓰기	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	

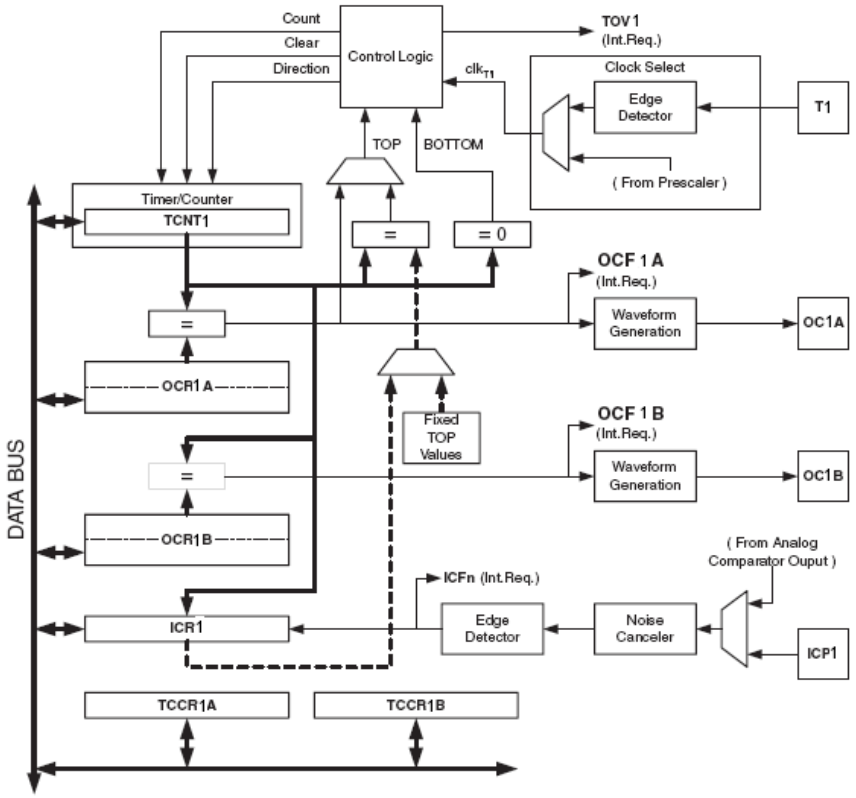
<그림 2.2.15> 타이머/카운터 인터럽트 플랙 레지스터(TIFR)

<표 2.2.6> 타이머/카운터 인터럽트 플랙 레지스터(TIFR) 기능

비트	비트명	기 능
Bit 2	OCF0B	· Output Compare Flag 0 B : 타이머/카운터와 OCR0B의 데이터 사이에 비교 매치가 발생하면, OCF0B=1이 되어 타이머/카운터 출력비교 인터럽트가 발생.
Bit 1	TOV0	· Timer/Counter0 Overflow Flag : T타이머/카운터0에 오버플로우가 발생하면 TOV0=1이 되어 오버플로우 인터럽트가 발생.
Bit 0	OCF0A	· Output Compare Flag 0 A : 타이머/카운터와 OCR0A의 데이터 사이에 비교 매치가 발생하면, OCF0A=1이 되어 타이머/카운터 출력비교 인터럽트가 발생.

4. 16비트 타이머/카운터 1

타이머/카운터1은 16비트 타이머/카운터로 프로그램 실행시간과, 파형 발생, 신호시간측정을 정확하게 할 수 있다. 또한, 2개의 독립된 출력 비교기와 1개의 입력 캡쳐 기능을 가지고 있다. 16비트 타이머/카운터1의 구조는 <그림 2.2.16>과 같다.



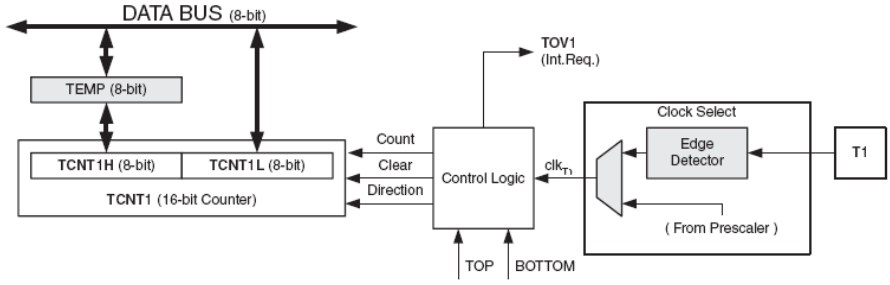
<그림 2.2.16> 타이머/카운터 1의 블록도

타이머/카운터1은 클록신호 clk_{T1}에 의해 16비트 타이머/카운터 TCNT1이 상향 카운터(up counter)로 동작하여 카운터의 최대값인 0xFFFF에 이르면 0x0000으로 되면서 오버플로우가 생겨 TOV1신호에 의해 오버플로우 인터럽트가 발생한다. 카운터 값은 출력비교 레지스터 OCR1x와 비교되고 비교한 값이 같으면 내부적으로 OCF1x신호에 의해 인터럽트가 발생되고, 외부적으로 OC1x 단자에 신호가 출력되도록 설정할 수 있다. (여기서, x는 A 또는 B를 나타냄.)

(1) 카운터부

타이머/카운터1의 카운터부는 프리스케일러를 통한 내부 클록 입력과 외부 클록 T_1, 상향/하향 카운터로 사용할 수 있는 타이머/카운터 TCNT1H와 TCNT1L 등으로 <그림 2.2.17>과 같이 구성된다. 외부 클록 T_1에 대해 카운터가 상승에지에서 동작할 것인지 하강에지에서 동작할 것인지는 타이머/카운터1 제어 레지스터B인 TCCR1B의 C12~C10비트로 설정한다.

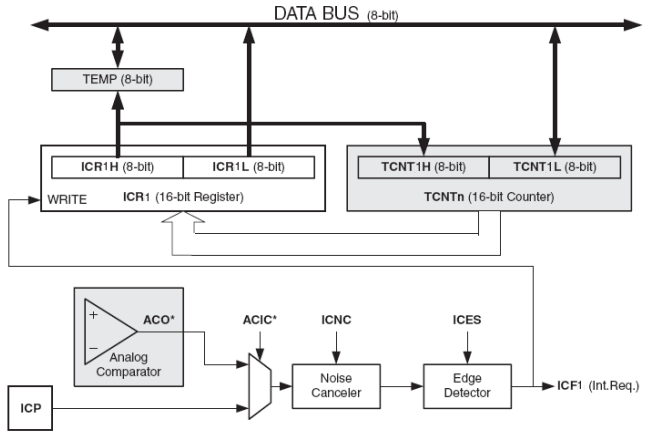
타이머/카운터1이 타이머로 동작할 때는 시스템 클록 주파수를 프리스케일러를 통해 1, 8, 64, 256, 1024의 분주비로 분주하여 클록 소스로 사용할 수도 있고, 클록 소스 입력을 차단할 수도 있다.



<그림 2.2.17> 타이머/카운터1의 카운터부 블록도

(2) 입력 캡쳐부

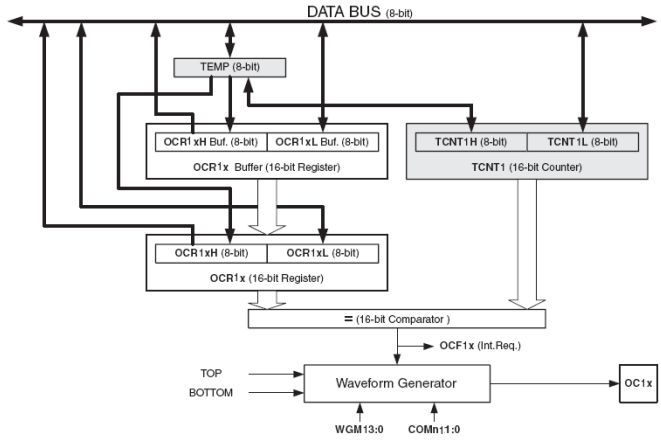
타이머/카운터1의 입력 캡쳐부는 입력핀 ICP에 입력되는 외부신호나 아날로그 비교기의 신호에 대한 타이머/카운터 레지스터 TCNT1의 카운트 값을 입력 캡쳐 레지스터 ICR1에 캡쳐하여 저장한다. 아날로그 비교기의 신호를 사용하려면, 아날로그 비교기 상태레지스터 ACSR의 ACIC=1로 설정해야 한다.



<그림 2.2.18> 타이머/카운터1의 입력 캡쳐부 블록도

(3) 출력 비교기부

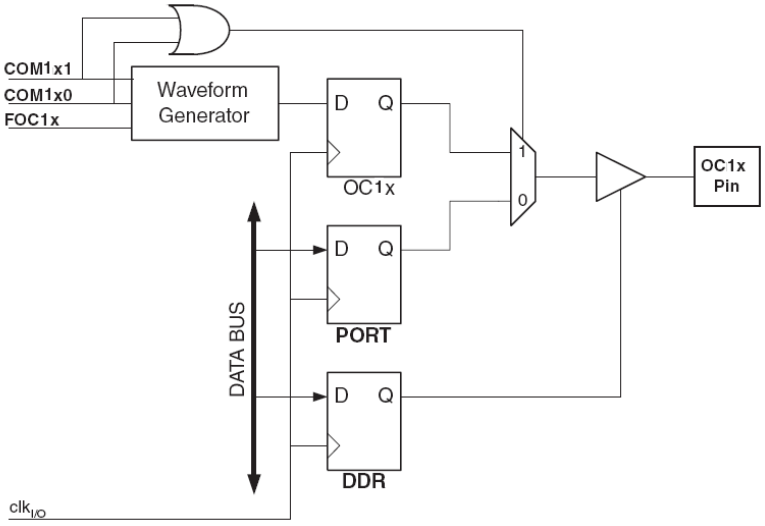
타이머/카운터1에 입력되는 클록 신호 clk_{T1}은 타이머/카운터 TCNT1을 동작시키고 동작중에 카운터 값을 출력비교 레지스터 OCR1x와 비교하여 값이 같아지면, 그 다음 주기에 내부적으로 OCF1x신호에 의해 출력비교 인터럽트를 발생하고, 외부적으로는 OC1x단자에 신호가 출력되도록 설정할 수 있다. OC1x단자로 출력되는 신호는 출력비교 레지스터 OCR1x에 의해 원하는 PWM 신호를 출력할 수 있다. (여기서, x는 A 또는 B를 나타냄.)



<그림 2.2.19> 타이머/카운터1의 출력 비교부 블록도

(4) 비교 매치 출력부

타이머/카운터1는 <그림 2.2.20>과 같이 타이머/카운터 TCNT1과 출력비교 레지스터 OCR1x를 비교하여 일치하면 타이머/카운터 제어 레지스터A TCCR1A의 COM1x1와 OCM1x0 비트를 설정하여 파형 발생기를 통해 출력할 비교출력 모드를 선택한다. (여기서, x는 A 또는 B를 나타냄.) OC1x단자(ATtiny2313에서는 PB3와 PB4에 해당)를 타이머/카운터의 비교출력 단자로 사용하려면 데이터 방향 레지스터(DDR)를 출력방향으로 설정해야 한다.



<그림 2.2.20> 타이머/카운터1의 비교 매치 출력부 블록도

5. 타이머/카운터 1의 동작 모드

타이머/카운터1의 동작모드는 타이머/카운터 제어 레지스터 TCCR1x의 파형발생 모드(WGM13~WGM10)비트에 의해 결정되며, 출력신호의 동작은 타이머/카운터 제어 레지스터 TCCR1A의 비교출력 모드(COM11~COM10)비트에 의해 OC1x단자로 출력될 파형이 결정된다. (여기서, x는 A 또는 B를 나타냄.)

타이머/카운터1의 동작 모드는 일반모드, CTC모드, 고속 PWM모드, 위상 보정 PWM모드, 위상 주파수 보정 PWM모드 등 5가지가 있다.

(1) 일반모드

타이머/카운터1의 일반모드(Normal Mode)는 타이머/카운터1 제어 레지스터 TCCR1x의 파형발생모드(WGM12~WGM10)를 '0'으로 설정한다. 이 모드에서 카운터는 상향 카운트만을 수행한다. 또한, 카운트 중에 클리어가 되지 않으며 클록 입력에 의해 16비트의 최대 카운트값인 0xFFFF가 되면 카운터값이 0x0000으로 되어 카운트 동작을 반복한다. (여기서, x는 A 또는 B를 나타냄.)

타이머/카운터 레지스터 TCNT1의 값이 0x0000로 됨과 동시에 타이머/카운터 오버플로우 플랙 비트 TOV1이 '1'로 되면서 오버플로우 인터럽트가 발생한다.

일반모드에서 출력비교부는 인터럽트를 발생시키는 데 사용할 수 있으나, 파형을 발생시키기 위한 출력비교는 CPU 타임을 너무 많이 차지 하므로 권장되지 않는다.

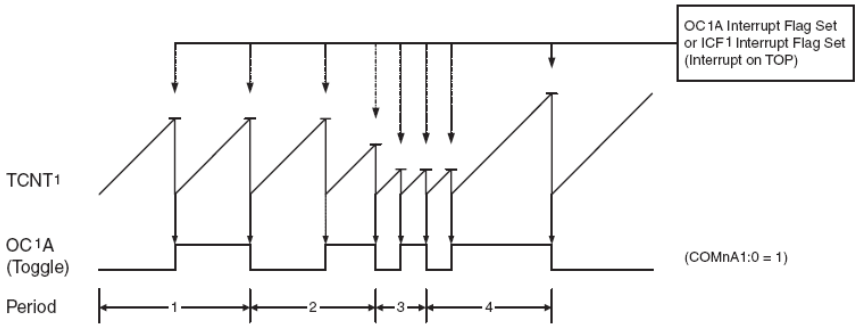
(2) CTC모드

타이머/카운터1의 CTC모드(Clear Timer on Compare Match Mode)는 타이머/카운터1 제어 레지스터 TCCR1x의 파형발생모드(WGM13~WGM10)를 '4' 또는 '12'로 설정한다. (여기서, x는 A 또는 B를 나타냄.) 타이머/카운터1 레지스터 TCNT1의 계수동작은 0x0000에서 0xFFFF로 상향 카운트를 반복하며, 파형발생 모드가 '4'인 경우에는 출력비교 레지스터 OCR1A와 일치하면, TCNT1이 '0'으로 클리어되고, 파형발생 모드가 '12'인 경우에는 입력 캡쳐 레지스터 ICR1과 일치하면, TCNT1이 '0'으로 클리어 된다.

CTC모드에서 OC1x단자로 파형을 출력하려면, 타이머/카운터1 제어 레지스터 TCCR1A의 비교출력 모드(COM1x1~COM1x0)를 '1'로 설정하여 토글 모드로 하면, OC1x 단자로 파형이 출력된다. OC1x단자로 출력되는 파형의 출력 주파수는 다음 식으로 계산된다.

$$f_{OC1x} = \frac{f_{clk_{I/O}}}{2 \cdot N \cdot (1 + OCR1x)}$$

여기서, N은 타이머/카운터1의 프리스케일러 분주비(1, 8, 64, 256, 1024).



<그림 2.2.21> 타이머/카운터1의 CTC 모드 동작

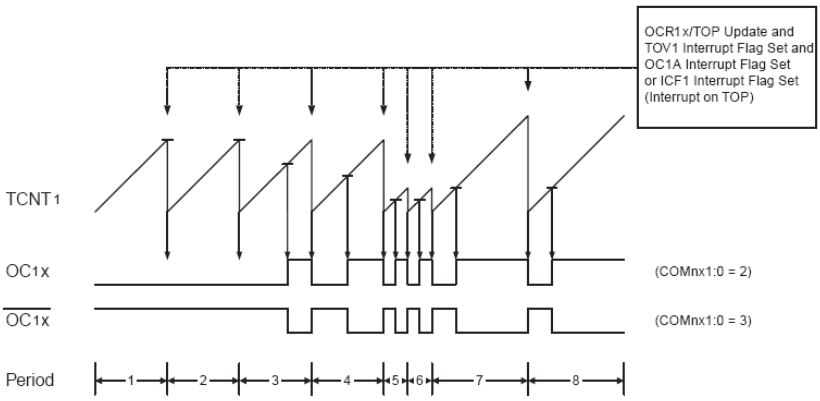
(3) 고속 PWM모드

고속 PWM모드(Fast PWM Mode)는 타이머/카운터 레지스터 TCCR1x에서 파형발생 모드(WGM13~WGM10)를 “5”, “6”, “7”, “14”, “15”로 설정하여 높은 주파수의 PWM 출력 파형을 발생시킨다. 타이머/카운터 레지스터 TCNT1의 카운트 동작이 0x0000에서 0xFFFF까지 반복적으로 수행되며, 이 값은 출력비교 레지스터 OCR1x의 값과 비교되어 일치하면 OC1x 출력신호가 ‘0’이되고 오버플로우가 발생되면 OC1x 출력신호는 ‘1’이 된다.

OC1x단자를 통해 출력되는 PWM 출력신호의 주파수는 다음과 같다.

$$f_{OC1xPWM} = \frac{f_{clk_{I/O}}}{N \cdot (1+TOP)}$$

여기서, N은 타이머/카운터1의 프리스케일러 분주비(1, 8, 64, 256, 1024)이고, TOP값은 파형발생 모드가 5인 경우에는 0x00FF, 6인 경우에는 0x01FF, 7인 경우에는 0x03FF, 14인 경우에는 ICR1값, 15인 경우에는 OCR1A값이다.



<그림 2.2.22> 타이머/카운터1의 고속 PWM 동작 모드

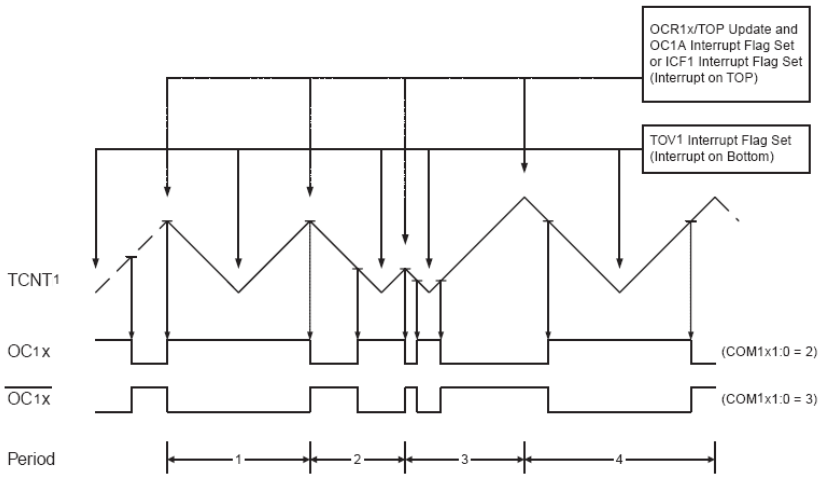
(4) 위상 보정 PWM모드

위상 보정 PWM모드(Phase Correct PWM Mode)는 타이머/카운터 레지스터 TCCR1x의 파형발생모드(WGM13~WGM10)를 “1”, “2”, “3”, “10”, “11”로 설정하여 높은 분해능의 PWM 출력 파형을 발생시킨다. 위상 보정 PWM모드에서는 타이머/카운터 레지스터 TCNT1의 카운트 동작이 0x0000에서 0xFFFF로 증가했다가, 다시 0xFFFF에서 0x0000으로 감소하는 동작을 반복하며, TCNT1의 값은 출력비교 레지스터 OCR1x의 값과 비교하여 0xFFFF로 증가하는 도중에 일치하면 OC1x 출력신호는 ‘0’이되고 0x0000으로 감소하는 도중에 일치하면 OC1x 출력신호는 ‘1’이 된다.

위상 보정 PWM모드는 고속 PWM 모드에 비해 주파수는 약 1/2로 낮아지지만, 고속 PWM 모드의 튜티비 분해능 보다 2배 정도 높아진다. OC1x단자를 통해 출력되는 PWM 출력신호의 주파수는 다음과 같다.

$$f_{\mathrm{OC1xPCPWM}} = \frac{f_{\mathrm{clk_{I/O}}}}{2 \cdot \mathrm{N} \cdot \mathrm{TOP}}$$

여기서, N은 타이머/카운터1의 프리스케일러 분주비(1, 8, 64, 256, 1024)이고, TOP값은 파형발생 모드가 1인 경우에는 0x00FF, 2인 경우에는 0x01FF, 3인 경우에는 0x03FF, 10인 경우에는 ICR1값, 11인 경우에는 OCR1A값이다.



<그림 2.2.23> 타이머/카운터1의 위상 보정 PWM 동작 모드

(5) 위상 및 주파수 보정 PWM모드

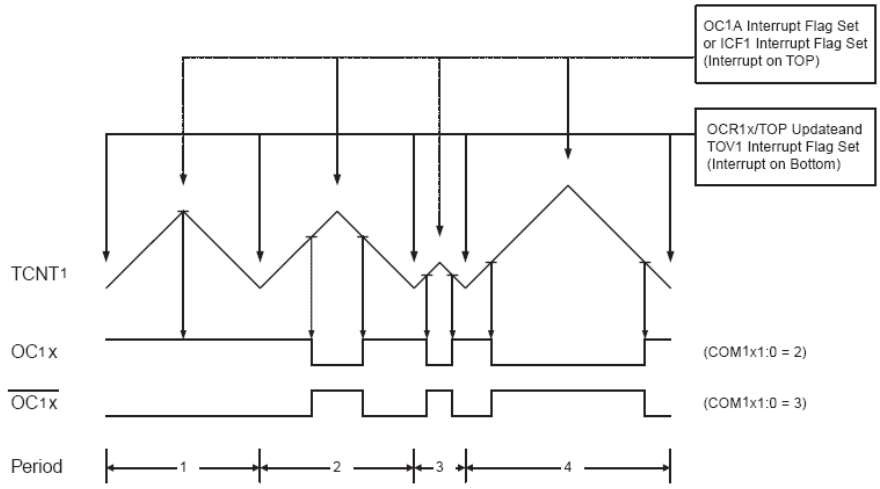
위상 및 주파수 보정 PWM모드(Phase and Frequency Correct PWM Mode)는 타이머/카운터 레지스터 TCCR1x의 파형발생모드(WGM13~WGM10)를 “8” 또는 “9”로 설정하여 높은 분해능의 위상 및 주파수 보정 PWM 발생시킨다. 위상 보정 PWM모드에서는 타이머/카운터 레지스터 TCNT1의 카운트 동작이 0x0000에서 0xFFFF로 증가했다가, 다시 0xFFFF에서 0x0000으로 감소하는 동작을 반복하며, TCNT1의 값은 출력비교 레지스터 OCR1x의 값과 비교하여 0xFFFF로 증가하는 도중에 일치하면 OC1x 출력신호는 ‘0’이되고 0x0000으로 감소하는 도중에 일치하면 OC1x 출력신호는 ‘1’이 된다.

위상 및 주파수 보정 PWM모드는 고속 PWM 모드에 비해 주파수는 약 1/2로 낮아지지만, 고속 PWM 모드의 튜티비 분해능 보다 2배 정도 높아진다. OC1x단자를 통해 출력되는 PWM 출력신호의 주파수는 다음과 같다.

$$f_{OC1xPFCPWM} = \frac{f_{clk_{I/O}}}{2 \cdot N \cdot TOP}$$

여기서, N은 타이머/카운터1의 프리스케일러 분주비(1, 8, 64, 256, 1024)

이고, TOP값은 파형발생 모드가 8인 경우에는 ICR1값, 9인 경우에는 OCR1A값이다.



<그림 2.2.24> 타이머/카운터1의 위상 및 주파수 보정 PWM 동작 모드

6. 16비트 타이머/카운터1의 레지스터

(1) 타이머/카운터1 제어 레지스터(TCCR1x)

TCCR1x 레지스터는 타이머/카운터1의 동작모드를 설정하고 프리스케일러의 분주비를 설정한다.

비트	7	6	5	4	3	2	1	0	
0x2F(0x4F)	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10	TCCR1A
읽기 / 쓰기	R/W	R/W	R/W	R/W	R	R	R/W	R/W	

<그림 2.2.25> 타이머/카운터1 제어 레지스터A(TCCR1A)

<그림 2.2.25>에서 비교 매치 출력 x 모드(Compare Match Output x Mode)의 COM1x1과 COM1x0비트는 OC1x 핀의 동작을 설정하며, 파형발생모드(Waveform Generation Mode)의 WGM11과 WGM10비트는 타이머/카운터0 제어 레지스터B의 WGM12 비트와 조합이 되어 타이머/카운터1의 동작 모드를 설정한다.

비트	7	6	5	4	3	2	1	0	
0x2E(0x4E)	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
읽기 / 쓰기	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	

<그림 2.2.26> 타이머/카운터1 제어 레지스터B(TCCR1B)

<그림 2.2.26>에서 ICNC1(Input Capture Noise Canceler)비트는 입력캡쳐 단자 ICP로 입력되는 신호에 대한 노이즈 제거기의 동작을 설정하며, 노이즈 제거기가 동작하면 입력 캡쳐신호는 4개의 시스템 클록 만큼 지연된다. ICES1(Input Capture Edge Select)비트는 입력캡쳐 단자 ICP로 입력되는 신호가 상승에지에서 캡쳐하려면 '1'로 하강에지에서 캡쳐하려면 '0'으로 설정한다. 캡쳐 신호는 입력캡쳐 레지스터(ICR1)에 저장되며, 타이머/카운터 인터럽트 플랙 레지스터의 입력캡쳐 플랙(ICF1)이 '1'로 되면 입력캡쳐 인터럽트가 발생한다. 입력캡쳐 레지스터가 파형발생 모드의 TOP으로 사용되면 입력캡쳐 기능은 정지된다.

클록 선택(Clock Select)의 CS12~CS10비트는 내부 클록을 사용할 것인지, 외부 클록을 사용할 것인지를 선택하거나 내부 클록의 경우 클록의 분주비를 설정한다.

비트	7	6	5	4	3	2	1	0	
0x22(0x42)	FOC1A	FOC1B	-	-	-	-	-	-	TCCR1C
읽기 / 쓰기	W	W	R	R	R	R	R	R	

<그림 2.2.27> 타이머/카운터1 제어 레지스터C(TCCR1C)

<표 2.2.27>에서 강제출력 비교 x(Force Output Compare x)의 FOC1x 비트는 PWM모드가 아닌 경우에만 유효하며, TCCR1B 레지스터가 PWM모드로 동작 중일 때는 FOC1x비트를 "1"로 해야 한다. FOC1x가 "1"로 설정되면 OC1x 단자에 출력비교 매치된 값을 출력으로 내보내며, 출력신호의 동작은 COM1x1~COM1x0비트로 설정한다. FOC1x는 인터럽트를 발생시키지도 않으며, CTC모드에서 OCR1A의 값이 0xFFFF가 되더라도 "0"으로 되지 않는다. FOC1x비트는 항상 "0"로 설정한다.

<표 2.2.7>는 파형발생모드(WGM12~WGM10)비트에 의한 동작모드 설정하는 것이며, <표 2.2.8>은 비교출력모드(COM1x1~COM1x0)

<표 2.2.7> WGM13~WGM10 비트에 의한 동작모드 설정

모드	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (WPM10)	동작모드	최대값	OCR1x 업데이트	TOV1플랙
0	0	0	0	0	일반	0xFFFF	설정즉시	0xFFFF
1	0	0	0	1	PWM, 위상보정	0x00FF	0x00FF	0x0000
2	0	0	1	0	PWM, 위상보정	0x01FF	0x01FF	0x0000
3	0	0	1	1	PWM, 위상보정	0x03FF	0x03FF	0x0000
4	0	1	0	0	CTC	OCR1A	설정즉시	0xFFFF
5	0	1	0	1	고속PWM	0x00FF	0x00FF	0x00FF
6	0	1	1	0	고속PWM	0x01FF	0x01FF	0x01FF
7	0	1	1	1	고속PWM	0x03FF	0x03FF	0x03FF
8	1	0	0	0	PWM, PF보정	ICR1	0x0000	0x0000
9	1	0	0	1	PWM, PF보정	OCR1A	0x0000	0x0000
10	1	0	1	0	PWM, PF보정	ICR1	ICR1	0x0000
11	1	0	1	1	PWM, 위상보정	OCR1A	OCR1A	0x0000
12	1	1	0	0	CTC	ICR1	설정즉시	0xFFFF
13	1	1	0	1	-	-	-	-
14	1	1	1	0	고속PWM	ICR1	ICR1	ICR1
15	1	1	1	1	고속PWM	OCR1A	OCR1A	OCR1A

※ PF보정은 '위상 및 주파수 보정 PWM'임.

<표 2.2.8> COM1x1~COM1x0비트에 의한 OC1x핀 기능 설정

모드	COM1x1	COM1x0	OC1x핀의 기능
PWM모드가 아닌 경우	0	0	정상적인 I/O포트로 동작, OC1x 출력차단
	0	1	비교 매치에서 OC1x 출력을 토글
	1	0	비교 매치에서 OC1x = 0
	1	1	비교 매치에서 OC1x = 1
고속 PWM 모드	0	0	정상적인 I/O포트로 동작, OC1x 출력차단
	0	1	WGM13=0, 정상적인I/O포트로 동작, OC1x 출력차단 WGM13=1, 비교 매치에서 OC1A 출력을 토글 (OC1B 사용안함)
	1	0	비교 매치에서 OC1x=0, '최대값'에서 OC1x=1
	1	1	비교 매치에서 OC1x=1, '최대값'에서 OC1x=0
위상보정PWM 모드/위상 및 주파수 보정 PWM 모드	0	0	정상적인 I/O포트로 동작, OC1x 출력차단
	0	1	WGM13=0, 정상적인I/O포트로 동작, OC1x 출력차단 WGM13=1, 비교 매치에서 OC1A 출력을 토글 (OC1B 사용안함)
	1	0	상향 카운터 비교 매치에서 OC1x=0, 하향 카운터 비교 매치에서 OC1x=1
	1	1	상향 카운터 비교 매치에서 OC1x=1, 하향 카운터 비교 매치에서 OC1x=0

<표 2.2.9> 타이머/카운터1의 클록소스 선택 비트 설정

CS12	CS11	CS10	클록 소스
0	0	0	클록소스 없음(타이머/카운터 정지)
0	0	1	$clk_{I/O}$ / 1
0	1	0	$clk_{I/O}$ / 8
0	1	1	$clk_{I/O}$ / 64
1	0	0	$clk_{I/O}$ / 256
1	0	1	$clk_{I/O}$ / 1024
1	1	0	T_1핀에 외부 클록소스(하강에지)
1	1	1	T_1핀에 외부 클록소스(상승에지)

(2) 타이머/카운터1 레지스터(TCNT1)

타이머/카운터1 레지스터 TCNT1(Timer/Counter Register1)는 타이머/카운터1의 16비트 카운터값을 저장하고 있으며, 읽기와 쓰기 동작을 할 수 있지만 카운터로 동작하고 있는 동안에 값을 수정하면, TCNT1값과 OCR1x레지스터 사이의 비교 매치에 오류가 발생할 수 있다.

비트	7	6	5	4	3	2	1	0	
0x2D(0x4D)	TCNT1 [15:8]								TCNT1H
0x2C(0x4C)	TCNT1 [7:0]								TCNT1L
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 2.2.28> 타이머/카운터1 레지스터B(TCNT1)

(3) 타이머/카운터1 출력비교 레지스터(OCR1x)

타이머/카운터1 출력비교 레지스터 OCR1x(Output Compare Register x)는 타이머/카운터 레지스터 TCNT1값과 비교하여 OC1x단자에 출력신호를 내보내기 위한 값을 저장하고 있다.

비트	7	6	5	4	3	2	1	0	
0x2B(0x4B)	OCR1A [15:8]								OCR1AH
0x2A(0x4A)	OCR1A [7:0]								OCR1AL
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 2.2.29> 타이머/카운터1 출력비교 레지스터(OCR1A)

비트	7	6	5	4	3	2	1	0	
0x29(0x49)	OCR1B [15:8]								OCR1BH
0x28(0x48)	OCR1B [7:0]								OCR1BL
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 2.2.30> 타이머/카운터1 출력비교 레지스터(OCR1B)

(4) 타이머/카운터1 입력 캡쳐 레지스터(ICR1)

타이머/카운터1 입력 캡쳐 레지스터(Timer/Counter Input Capture Register)레지스터는 입력캡쳐 신호 ICP에 의해 타이머/카운터 TCNT1의 값을 캡쳐하여 저장한다.

비트	7	6	5	4	3	2	1	0	
0x25(0x45)	ICR1H [15:8]								ICR1H
0x24(0x44)	ICR1L [7:0]								ICR1L
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 2.2.31> 타이머/카운터1 입력 캡쳐 레지스터(ICR)

(5) 타이머/카운터 인터럽트 마스크 레지스터(TIMSK)

타이머/카운터 인터럽트 마스크 레지스터(Timer/Counter Interrupt Mask Register)는 타이머/카운터1가 발생하는 인터럽트를 개별적으로 허용한다.

비트	7	6	5	4	3	2	1	0	
0x39(0x59)	TOIE1	OCIE1A	OCIE1B	-	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
읽기 / 쓰기	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	

<그림 2.2.32> 타이머/카운터 인터럽트 마스크 레지스터(TIMSK)

<표 2.2.10> 타이머/카운터 인터럽트 마스크 레지스터(TIMSK) 기능

비트	비트명	기 능
Bit 7	TOIE1	· Timer/Counter1 Overflow Interrupt Enable : TOIE1=1이고, 상태 레지스터 SREG의 I=1 일때, 타이머/카운터1 오버플로우 인터럽트 허용. TIFR레지스터의 TOV1=1 이면, 대응하는 인터럽트 벡터가 처리됨.
Bit 6	OCIE1A	· Timer/Counter1 Output Compare Match A Interrupt Enable : OCIE1A=1이고, 상태 레지스터 SREG의 I=1 일때, 타이머/카운터1 비교 매치 A 인터럽트 허용. TIFR레지스터의 OCF1A=1 이면, 대응하는 인터럽트 벡터가 처리됨.
Bit 5	OCIE1B	· Timer/Counter1 Output Compare B Interrupt Enbale : OCIE1B=1이고, 상태 레지스터 SREG의 I=1 일때, 타이머/카운터1 비교 매치 B 인터럽트 허용. TIFR 레지스터의 OCF1B=1 이면, 대응하는 인터럽트 벡터가 처리됨.
Bit 3	ICIE1	· Timer/Counter1 Input Capture Interrupt Enable : ICIE1=1이고, 상태 레지스터 SREG의 I=1 일때, 타이머/카운터1 입력 캡쳐 인터럽트 허용. TIFR레지스터의 ICF1=1이면, 대응하는 인터럽트벡터가 처리됨.

(6) 타이머/카운터 인터럽트 플랙 레지스터(TIFR)

타이머/카운터 인터럽트 플랙 레지스터(Timer/Counter Interrupt Flag Register)는 타이머/카운터1가 발생하는 인터럽트 플랙을 저장한다.

비트	7	6	5	4	3	2	1	0	
0x38(0x58)	TOV1	OCF1A	OCF1B	-	ICF1	OCF0B	TOV0	OCF0A	TIFR
읽기 / 쓰기	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	

<그림 2.2.33> 타이머/카운터 인터럽트 플랙 레지스터(TIFR)

<표 2.2.11> 타이머/카운터 인터럽트 플랙 레지스터(TIFR) 기능

비트	비트명	기 능
Bit 7	TOV1	· Timer/Counter1 Overflow Flag : WGM13~WGM10에 따라 세팅됨. 일반모드와 CTC모드에서, 타이머 오버플로우 일때 TOV1=1. 타이머/카운터1에 오버플로우가 인터럽트 벡터가 실행되면 TOV1=0.
Bit 6	OCF1A	· Timer/Counter1 Output Compare A Match Flag : 카운터(TCNT1)값이 출력 비교 레지스터 A(OCR1A)와 매치된 후 OCF1A=1. 출력 비교 매치 A인터럽트 벡터가 실행되면 OCF1A=0.
Bit 5	OCF1B	· Timer/Counter1 Output Compare B Match Flag : 카운터(TCNT1)값이 출력 비교 레지스터 B(OCR1B)와 매치된 후 OCF1B=1. 출력 비교 매치 B인터럽트 벡터가 실행되면 OCF1B=0.
Bit 3	ICF1	· Timer/Counter1 Input Capture Flag : ICP핀에 캡쳐가 발생하면 ICF1=1.입력 캡쳐 레지스터(ICR1)가 WGM13~WGM10의 TOP값으로 사용될 때, 카운터 값이 TOP이 되었을 때 ICF1=1. 입력 캡쳐 인터럽트 벡터가 실행되면 ICF1=0.

2.3 USART 직렬통신 포트

1. ATtiny2313 직렬 통신의 특징

ATtiny2313은 동기 및 비동기 전송모드에서 전이중(Full Duplex)통신이 가능하면, 높은 정밀도의 보레이트 발생기를 내장하고 있다. 전송 데이터 비트는 5~9비트로 설정하고, 정지비트는 1~2비트로 설정할 수 있다. 데이터를 전송하는 동안 패리티 비트를 사용하지 않을 수도 있고, 홀수 또는 짝수 패리티 비트로 설정할 수도 있다. 데이터를 수신하는 동안에는 패리티 에러, 데이터 오버런에러, 프레임 에러 등을 검출할 수 있다.

ATtiny2313의 직렬 통신포트에서는 송신완료, 송신 데이터 레지스터 준비완료, 수신완료 등을 인터럽트 소스로 사용할 수 있다.

(1) USART의 구조

<그림 2.3.1>은 ATtiny2313의 USART의 블록도로 클록 발생부, 송신부, 수신부 등으로 구성된다.

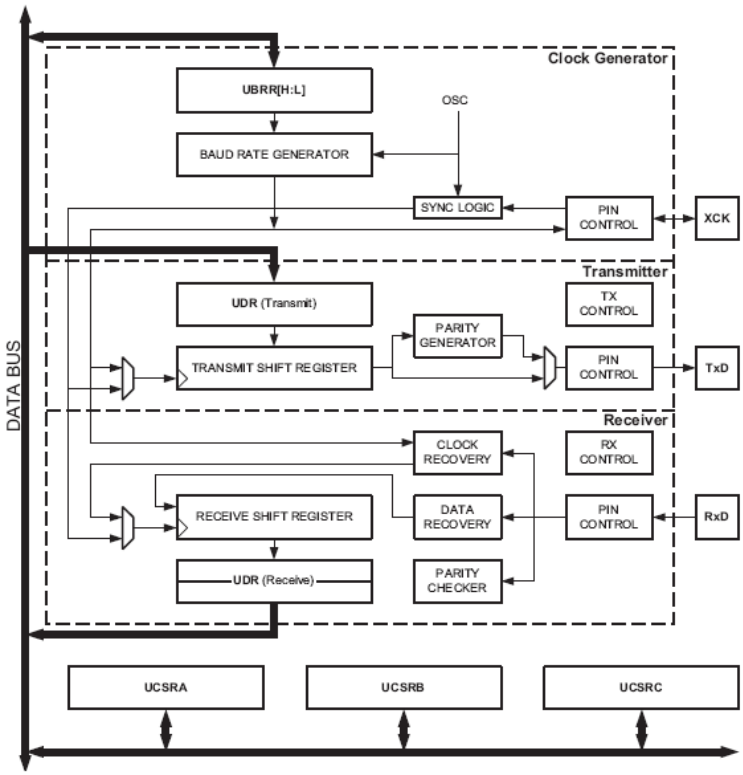
클록 발생부(Clock generator)는 보레이트 발생기와 동기 모드에서 클록과 동기시키는 동기회로 등으로 구성된다. 송신부(Transmitter)는 송신버퍼, 송신 시프트 레지스터, 패리티 발생기 등으로 구성되며, 수신부(Receiver)는 수신버퍼, 수신 시프트 레지스터, 패리티 검사기 등으로 구성된다.

ATtiny2313의 USART는 비동기 모드와 동기 모드로 나뉘며, 비동기 모드에는 일반 모드와 2배속 모드가 있고 동기 모드에는 마스터 모드와 슬레이브 모드가 있다.

USART를 비동기 모드로 동작시킬 것인지 또는 동기 모드로 동작시킬 것인지는 USART 제어 및 상태 레지스터C인UCSRC 레지스터의 USART 모드 선택 비트인 'UMSEL'비트로 선택한다. UMSEL=0으로 설정하여 비동기 모드인 경우, USART 제어 및 상태 레지스터A인 UCSRA의 U2X=0이면, 일반모드이고, U2X=1이면, 2배속 모드이다.

UMSEL=1로 설정하여 동기 모드인 경우, 클록 소스 내부이면 마스터 모드이고, 외부이면 슬레이브 모드이며, 클록 소스가 내부가 될 것인지 외

부가 될 것인지는 XCK핀에 대한 데이터 방향 레지스터에 의해서 결정된다. XCK핀은 동기 모드에서만 사용된다.

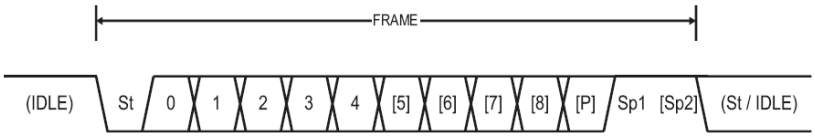


<그림 2.3.1> ATtiny2313의 USART 구조

(2) 전송 데이터 형식

USART에 의한 직렬 전송 데이터 형식은 1 시작 비트, 5~9 데이터 비트, 1 패리티 비트, 1~2 정지 비트로 구성된다. 전송 데이터 형식은 USART 제어 및 상태 레지스터B와 C의 케릭터 크기 비트 USCZ2~USCZ0, 패리티 모드 비트 UPM1~UPM0, 정지 비트 선택 비트 USBS 등을 사용한다. 캐릭터 크기 비트인 USCZ2~USCZ0는 데이터 형식의 데이터 비트 수를 선택하고, 패리티 모드 비트 UPM1~UPM0는 패리티 비트를 허용하며, 형식을 지정한다. 정지 비트 USBS에 의해서 1개 또는 2개의 정지 비트를 선택한다. 수신부 에서는 두번째 정지비트를 무시한다. 그러므로, 데이터 형식 에러는 첫번째 정지비트가 '0'인 경우에만 발

생한다.



<그림 2.3.2> ATtiny2313의 직렬 통신 전송 데이터 형식

<그림2.3.2>에서 St, (n), P, Sp, IDLE 등의 의미는 다음과 같다.

St : 시작비트, 항상 low.

(n) : 데이터 비트(0~8).

P : 패리티 비트 (홀수 또는 짝수)

Sp : 정지 비트, 항상 high.

IDLE : 통신선에 데이터 전송이 없는 휴지기간, 항상 high.

전송 데이터가 n개의 비트일 경우 패리티 비트는 모든 데이터 비트의 배타적-OR로 계산된다. 홀수 패리티 비트는 P_{odd}이고, 짝수 패리티 비트는P_{even}로 표시한다.

$$P_{odd} = d_{n-1} \oplus \cdots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

$$P_{even} = d_{n-1} \oplus \cdots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$

(3) USART 초기화

USART는 통신을 시작하기 전에 초기화를 해야 한다. 초기화 과정은 일반적으로 보우레이트 설정, 데이터 형식 설정과 송신부와 수신부의 사용 허용 으로 구성된다. 인터럽트에 의해서 USART를 동작시키려면, 초기화 과정에서 글로벌 인터럽트 플랙을 '0'으로 한다.

TXC 플랙은 송신부가 전송을 완료했는지를 확인하는 데 사용하고, RXC 플랙은 수신 버퍼에 읽지 않은 데이터가 남아 있는지를 확인하는 데 사용한다. TXC 플랙은 전송전에 '0'으로 해야 한다.

1) USART 송신부의 데이터 전송

USART의 송신부는 USART 제어 및 상태 레지스터B (UCSRB)의 송신허용 비트를 TXEN=1로 설정하면 송신 동작이 허용된다. USART 제

어 및 상태 레지스터A (UCSRA)의 데이터 준비 레지스터(USART Data Register Empty)비트가 UDRE=1이면, 인터럽트가 발생되어 MCU는 송신할 문자를 송신 버퍼에 적재하고, 적재가 끝나면 UDRE=0으로 된다.

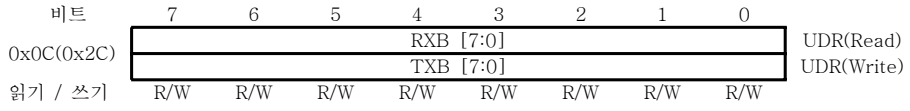
2) USART 수신부의 데이터 수신

USART의 수신부는 USART 제어 및 상태 레지스터B (UCSRB)의 수신허용 비트를 RXEN=1로 설정하면 수신 동작이 허용된다. USART 제어 및 상태 레지스터A (UCSRA)의 수신 완료 레지스터(USART Receive Complete)비트가 RXC=1이면, 인터럽트가 발생되어 수신 버퍼에 수신된 문자가 있다는 것을 MCU에 알리고 USART 제어 및 상태 레지스터A(UCSRA)에서 데이터 형식 오류, 오버런 오류, 패리티 오류를 확인하고, MCU가 수신 버퍼의 데이터를 읽으면 RXC=0으로 된다.

2. USART의 레지스터

(1) USART I/O 레지스터(UDR)

<그림 2.3.3>과 같이 USART 송신 데이터 버퍼 레지스터와 수신 데이터 버퍼 레지스터는 USART 데이터 레지스터(UDR)이라고 하는 I/O 주소를 공유하여 사용한다. 송신할 데이터를 데이터 레지스터(UDR)에 쓰면 송신 데이터 버퍼 레지스터(TXB)에 저장되고, 수신할 데이터를 데이터 레지스터(UDR)에서 읽으면 수신 데이터 버퍼(RXB)에 수신된 값이 읽혀진다. 전송 데이터가 5~7비트로 설정되면, 사용하지 않는 상위 비트에 대해 송신부에서는 무시되고, 수신부에서는 0으로 처리된다.



<그림 2.3.3> USART I/O 데이터 레지스터(UDR)

송신 버퍼는 USART 제어 및 상태 레지스터A(UCSRA)의 데이터 확인 레지스터가 UDRE=1 일때만 쓰기(write)가 가능하고, UDRE=0일 때는 쓰기를 하더라도 무시된다. 송신할 데이터가 송신 버퍼에 저장되고, 송신부가 허용상태이면, 시프트 레지스터가 비어 있는 것을 확인한 다음 데이터

를 시프트레지스터로 보내서 TxD핀을 통해 직렬로 전송한다.

(2) USART 제어 및 상태 레지스터A(UCSRA)

UCSRA레지스터는 직렬 통신 포트의 송수신 동작을 제어하거나 송수신 상태를 저장하는 기능을 한다.

비트	7	6	5	4	3	2	1	0	
0x0B(0x2B)	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM	UCSRA
읽기 / 쓰기	R	R/W	R	R	R	R	R/W	R/W	

<그림 2.3.4> USART 제어 및 상태 레지스터A(UCSRA)

<표 2.3.1> USART 제어 및 상태 레지스터A(UCSRA)

비트	비트명	기 능
Bit 7	RXC	· USART Receive Complete : 수신 버퍼에 데이터가 있을 때 RXC=1, 수신 버퍼가 데이터가 없을 때 RXC=0. 수신부가 허용상태가 아니면 수신버퍼를 비우고 RXC=0로 됨. RXC 플랙은 수신완료 인터럽트 발생에 사용할 수 있음.
Bit 6	TXC	· USART Transmit Complete : 송신 시프트 레지스터가 데이터 전송을 완료하면, TXC=1이고, 송신완료 인터럽트가 발생하면 TXC=0이됨. TXC 플랙은 송신 완료 인터럽트 발생에 사용할 수 있음.
Bit 5	UDRE	· USART Data Register Empty : 송신버퍼가 새로운 송신 데이터를 받을 준비가 되어있으면, UDRE=1이고 송신버퍼에 새로운 데이터가 들어오면 UDRE=1이 됨. UDRE 플랙은 송신 데이터를 받을 준비가 되면 인터럽트 발생에 사용할 수 있음.
Bit 4	FE	· Frame Error : 수신버퍼의 첫번째 정지비트가 '0'일때, FE=1이고 수신버퍼를 읽을 때까지 유효함. 수신된 데이터의 정지비트가 '1'일때, FE=0이됨.
Bit 3	DOR	· Data OverRun : 수신버퍼가 가득차면 데이터 오버런이 발생하며, DOR=1이고, 수신버퍼를 읽을 때까지 유효함.
Bit 2	UPE	· USART Parity Error : UCSRC레지스터의 UPM1=1인 패리티 비트를 사용한 경우 UPE=1이고, 수신버퍼 UDR을 읽을 때까지 유효함.
Bit 1	U2X	· Double the USART Transmission Speed : 비동기 모드에서 U2X=1, 동기 모드에서 U2X=0.
Bit 0	MPCM	· Multi-processor Communication Mode : 멀티프로세서 통신 모드 MPCM=1이고, 주소정보 없이 수신되는 데이터는 모두 무시됨.

(3) USART 제어 및 상태 레지스터B(UCSRB)

UCSRB레지스터는 직렬 통신 포트의 송수신 동작을 제어하거나, 전송 데이터의 9번째 비트를 저장하는 기능을 한다.

비트	7	6	5	4	3	2	1	0	
0x0A(0x2A)	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
읽기 / 쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	

<그림 2.3.5> USART 제어 및 상태 레지스터B(UCSRB)

<표 2.3.2> USART 제어 및 상태 레지스터B(UCSRB)

비트	비트명	기 능
Bit 7	RXCIE	· RX Complete Interrupt Enbale : RXCIE=1이고 SREG의 글로벌 인터럽트 플랙 I=1, UCSRA의 RXC=1일때 만 수신완료 인터럽트 발생.
Bit 6	TXCIE	· TX Complete Interrupt Enbale : TXCIE=1이고 SREG의 글로벌 인터럽트 플랙 I=1, UCSRA의 TXC=1일때 만 송신완료 인터럽트 발생.
Bit 5	UDRIE	· USART Data Register Empty Interrupt Enable : UDRIE=1이고 SREG의 글로벌 인터럽트 플랙 I=1 UCSRA의 UDRE=1일때 만 데이터 레지스터 준비 인터럽트 발생.
Bit 4	RXEN	· Receiver Enable : RXEN=1이면 RxD핀을 수신부로 동작. RXEN=0이면, 수신버퍼 값을 없애고 FE, DOR, UPE플랙도 유효하지 않게됨.
Bit 3	TXEN	· Transmitter Enable : TXEN=1이면 TxD핀을 송신부로 동작. RXEN=0이면, 송신 시프트 레지스터에 데이터가 남아 있을 경우 전송이 완료될 때까지 RXEN=0는 유효하지 않음.
Bit 2	UCSZ2	· Character Size : UCSRC의 UCSZ1~UCSZ0와 함께 송수신부의 데이터 비트의 데이터 형식을 설정함.
Bit 1	RXB8	· Receive Data Bit 8 : RXB8=1이면, 수신된 데이터의 9번째 비트를 저장.
Bit 0	TXB8	· Transmit Data Bit 8 : TXB8=1이면, 송신할 데이터의 9번째 비트를 저장.

(4) USART 제어 및 상태 레지스터C(UCSRC)

UCSRC레지스터는 직렬 통신 포트의 송수신 동작을 제어하는 기능을 한다.

비트	7	6	5	4	3	2	1	0	
0x03(0x23)	-	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
읽기 / 쓰기	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

<그림 2.3.6> USART 제어 및 상태 레지스터B(UCSRC)

<표 2.3.3> USART 제어 및 상태 레지스터B(UCSRC)

비트	비트명	기 능
Bit 6	UMSEL	· USART Mode Select : UMSEL=0이면 비동기모드, UMSEL=1이면 동기모드.
Bit 5	UPM1	· Parity Mode : 패리티 비트 설정 - UPM1=1, UPM0=0이면, 짝수 패리티 비트 사용. - UPM1=1, UPM0=1이며, 홀수 패리티 비트 사용.
Bit 4	UPM0	
Bit 3	USBS	· Stop Bit Select : 정지비트 설정 - USBS=0이면, 정지비트 '1' - USBS=1이면, 정지비트 '2'
Bit 2	UCSZ1	
Bit 1	UCSZ0	· Character Size : 데이터비트 설정 - UCSZ2=0, UCSZ1=0, UCSZ0=0이면, '5비트' - UCSZ2=0, UCSZ1=0, UCSZ0=1이면, '6비트' - UCSZ2=0, UCSZ1=1, UCSZ0=0이면, '7비트' - UCSZ2=0, UCSZ1=1, UCSZ0=1이면, '8비트' - UCSZ2=1, UCSZ1=1, UCSZ0=1이면, '9비트'
Bit 0	UCPOL	· Clock Polarity : 동기모드에서만 사용. - UCPOL=0이면, · XCK클록 상승에지에서 TxD단자의 송신데이터 출력. · XCK클록 하강에지에서 RxD단자의 수신데이터 입력. - UCPOL=1이면, · XCK클록 하강에지에서 TxD단자의 송신데이터 출력. · XCK클록 상승에지에서 RxD단자의 수신데이터 입력.

(5) USART 보우레이트 레지스터(UBRRH, UBRRL)

UBRR레지스터는 USART 포트의 송수신 속도를 설정하는 레지스터이다. UBRR레지스터 값은 시스템 클록의 분주비로 사용되어 보우레이트를 결정한다.

비트	15	14	13	12	11	10	9	8	
0x02(0x22)	-	-	-	-	UBRR [11:8]				UBRRH
0x09(0x29)	UBRR [7:0]								UBRRL
비트	7	6	5	4	3	2	1	0	
읽기 / 쓰기	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	

<그림 2.3.7> USART 보우레이트 레지스터(UBBR)

<표 2.3.4> 오실레이터 주파수에 대한 UBRR값

보우 레이트	11.0592 MHz				16 MHz			
	U2X=0 (비동기 일반)		U2X=1 (비동기 2배속)		U2X=0 (비동기 일반)		U2X=1 (비동기 2배속)	
	UBRR	에러[%]	UBRR	에러[%]	UBRR	에러[%]	UBRR	에러[%]
2,400	287	0.0	575	0.0	416	-0.1	832	0.0
4,800	143	0.0	287	0.0	207	0.2	416	-0.1
9,600	**71**	0.0	143	0.0	**103**	0.2	207	0.2
14,400	47	0.0	95	0.0	68	0.6	138	-0.1
19,200	**35**	0.0	71	0.0	**51**	0.2	103	0.2
28,800	23	0.0	47	0.0	34	-0.8	68	0.6
38,400	17	0.0	35	0.0	25	0.2	51	0.2
57,600	11	0.0	23	0.0	16	2.1	34	-0.8

제 3 장

ATtiny2313의 활용한 프로그래밍


0604
ATTINY2313-20PU

3.1 AVR Studio 및 WinAVR 설치

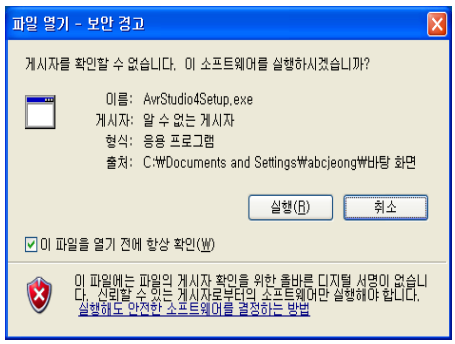
1. AVR Studio의 설치

AVR Studio는 Atmel사에서 무료로 제공하는 AVR 프로그래밍을 지원하는 통합환경(IDE)소프트웨어 이다. AVR Studio는 Atmel사에서 제공하는 매크로 어셈블러만을 포함하고 있었기 때문에, C언어를 이용하여 프로그램을 하고자 했던 사용자들은 주로 코드비전(Code Vision) C컴파일러나, IAR C컴파일러 등과 같은 상업용 C컴파일러를 사용해야 했다.

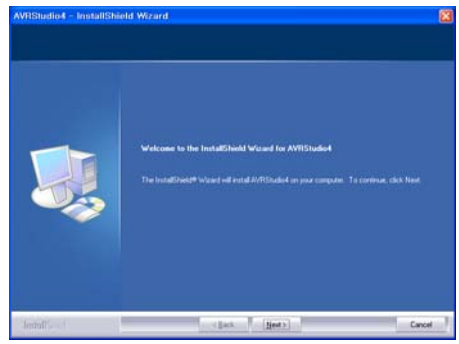
AVR Studio는 새롭게 기능을 향상시키면서, 리눅스에서 사용하고 있는 GNU GCC컴파일러를 AVR에 포팅한 AVR-GCC컴파일러를 플러그-인 하여 사용할 수 있도록 하면서 AVR 사용자들로부터 큰 호응을 받고 있다.

AVR Studio는 Atmel사의 홈페이지를 방문하면 무료로 다운로드 받아 설치할 수 있다. Atmel사의 홈페이지는 http://www.atmel.com이며, AVR Studio 프로그램을 찾으려면, 상단의 'PRODUCTS' 탭을 클릭하고 우측의 상품 메뉴 중에서 'Microcontrollers and DSP'로 들어가서, 'AVR 8bit RISC'로 들어가면 우측의 메뉴 중 'Tools and Software'를 선택하면 'Design Software' 중 'AVR Studio4'로 들어가면 최신버전을 다운로드 할 수 있다. 현재 버전은 'AVR Studio 4.12, build 460'로 파일의 용량은 46MB정도이다.

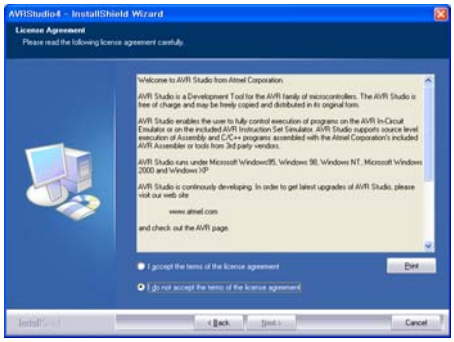
프로그램 설치는 <그림 3.1.1>~<그림 3.1.6> 순서로 설치하면 된다.



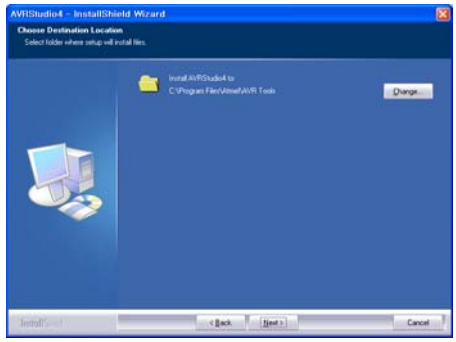
<그림 3.1.1> AVR Studio설치1



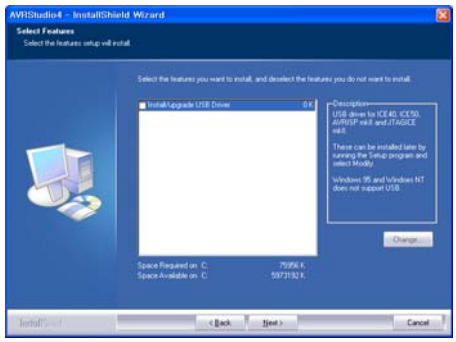
<그림 3.1.2> AVR Studio설치2



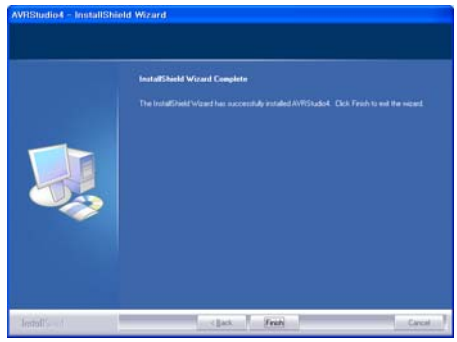
<그림 3.1.3> AVR Studio설치3



<그림 3.1.4> AVR Studio설치4



<그림 3.1.5> AVR Studio설치5



<그림 3.1.6> AVR Studio설치6

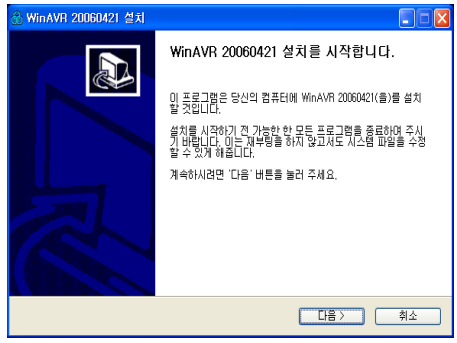
AVR Studio는 C:\Program Files\Atmel\AVR Tools\ 폴더에 설치되어 있다. 필요에 따라 2006년 10월에 발표된 서비스 팩 4(AVR Studio 4.12 Service Pack 4 (build 498))를 설치할 수도 있겠으나, 이 서비스 팩4를 설치하지 않아도 ATtiny2313의 프로그램에는 아무런 지장이 없다.

2. WinAVR의 설치

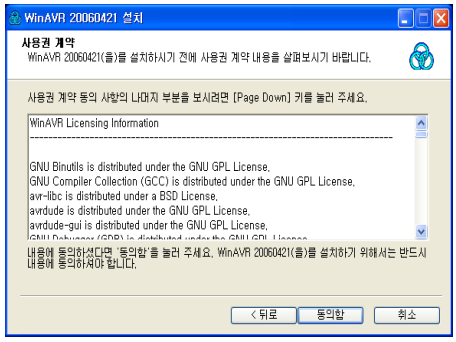
WinAVR은 AVR Studio와 마찬가지로 무료로 다운로드 받아 설치할 수 있다. WinAVR을 제공하고 있는 홈페이지는 http://sourceforge.net/projects/winavr/ 이며, 이곳에서 최신 버전의 WinAVR을 다운로드 받을 수 있다. 현재버전은 2006년 4월에 발표된 것으로 'WinAVR-20060421-Install'이며 용량은 약23MB 정도이다.

프로그램 설치는 <그림 3.1.7>~<그림 3.1.12> 순서로 설치하면 된다.

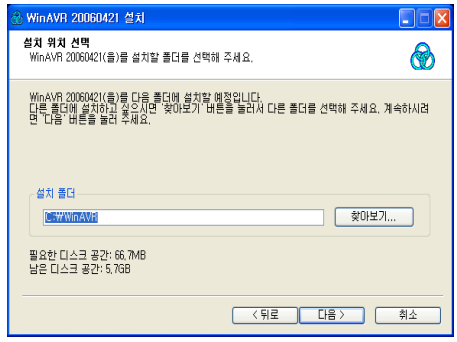
<그림 3.1.7> WinAVR의 설치1



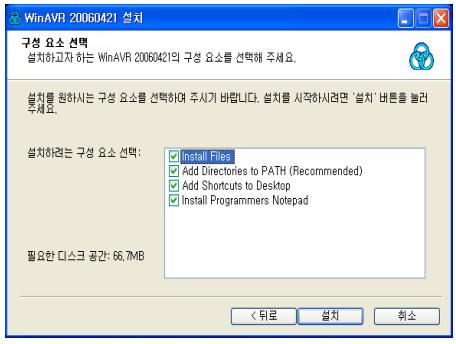
<그림 3.1.8> WinAVR의 설치2



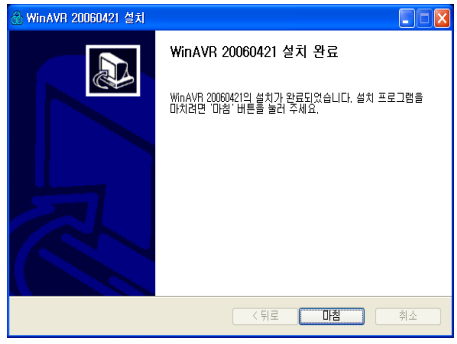
<그림 3.1.9> WinAVR의 설치3



<그림 3.1.10> WinAVR의 설치4



<그림 3.1.11> WinAVR의 설치5

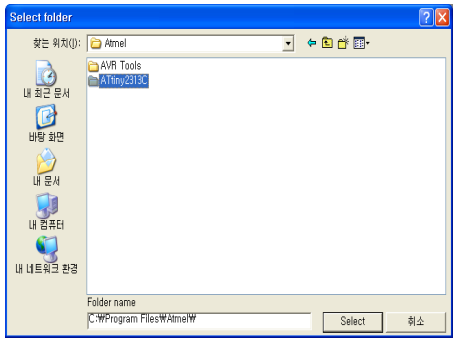


<그림 3.1.12> WinAVR의 설치6

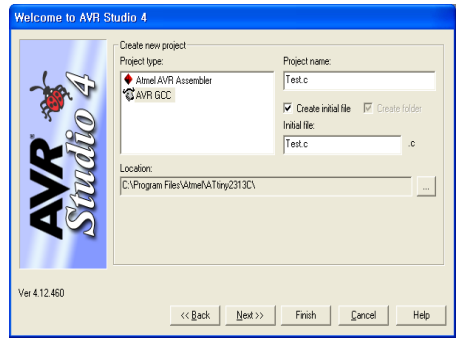
3. AVR Studio에서 WinAVR을 이용한 C언어 프로그램 작성

ATtiny2313의 프로그램을 C언어로 작성하기 위해서 AVR Studio와 WinAVR을 컴퓨터에 설치하였다. AVR Studio에서 WinAVR을 이용하여 C언어 프로그램 작성과정을 설명한다.

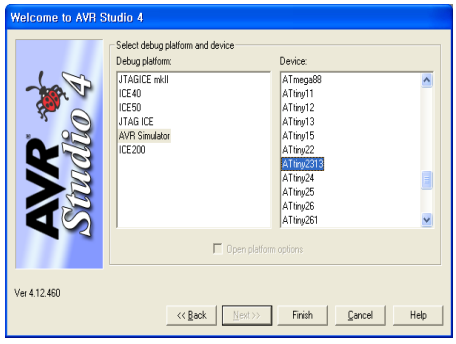
윈도 XP에 AVR Studio와 WinAVR이 설치된 것으로 가정하고 설명하기로 한다. 윈도 XP의 시작 메뉴에서 모든 프로그램의 'Atmel AVR Tools'항목의 'AVR Studio 4' 단축 메뉴를 클릭하여 프로그램을 실행시킨다.



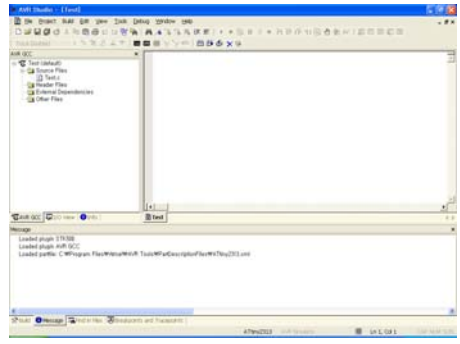
<그림 3.1.13> 작업폴더 만들기



<그림 3.1.14> 프로젝트 만들기

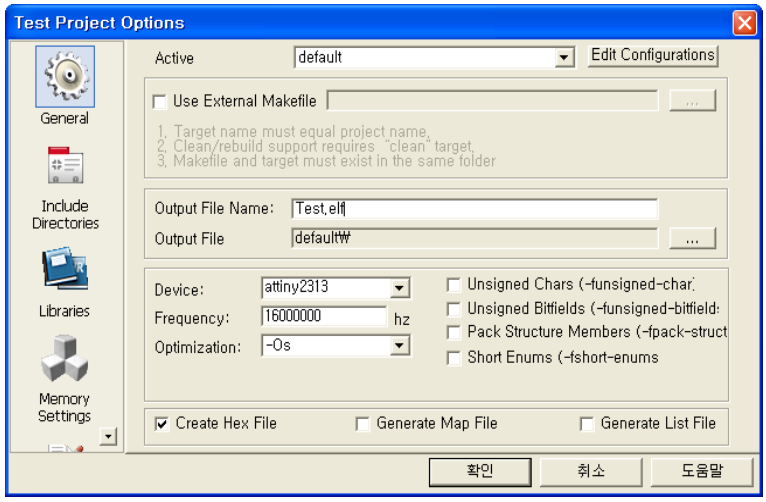


<그림 3.1.15> 디바이스 선택



<그림 3.1.16> 프로젝트 화면

<그림 3.1.13>과 같이 C:\Program Files\Atmel\ 폴더에 'ATtiny2313' 폴더를 추가하여 프로그램에 필요한 작업 폴더를 만든다. <그림 3.1.13>의 Folder name에서 'Select'를 클릭하면, <그림 3.1.14>와 같이 프로젝트 작성 화면이 나온다. AVR GCC를 선택하고 프로젝트 이름을 Project name에 'Test'라는 이름을 넣고 'Next'를 클릭한다. 디버그 플랫폼과 디바이스 선택 화면이 <그림 3.1.15>와 같이 나오면 AVR Simulator와 ATtiny2313을 선택하고 'Finish'를 클릭하면 <그림 3.1.16>과 같은 프로젝터 화면이 나온다.

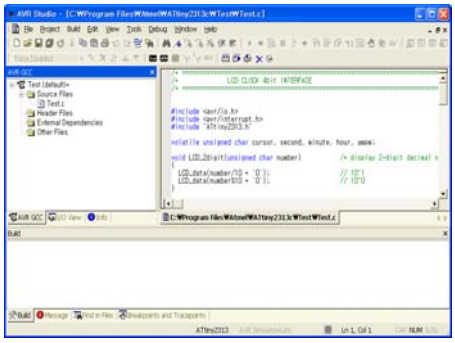


<그림 3.1.17> 프로젝트의 컴파일 옵션

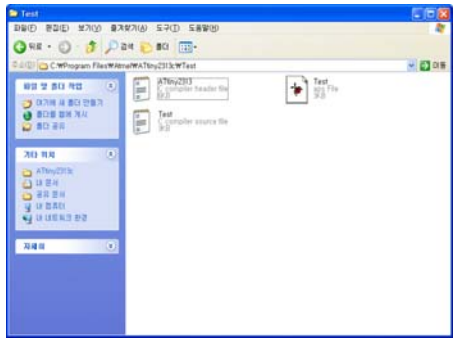
AVR Studio의 'Project'메뉴의 'Configuration Options'를 선택하면 <그림 3.1.17>과 같으며, Device, Frequency, Optimization은 각각 다음과 같이 설정해 준다.

Device : attiny2313, Frequency : 16000000, Optimization : -Os

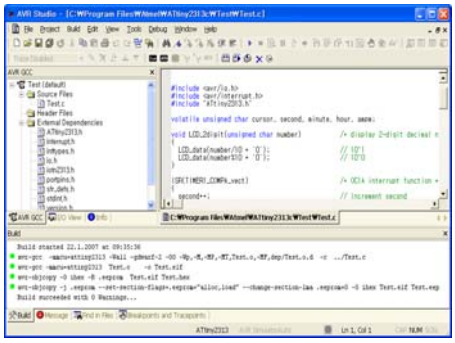
<그림 3.1.18>은 프로그램 소스를 컴파일하기 전의 화면이다. AVR Studio의 아이콘을 클릭하거나, 상단 메뉴의 'Build'에서 Build 또는 F7 키를 누르면 <그림 3.1.20>과 같이 컴파일 된다. 컴파일 결과는 C:\Program Files\Atmel\ATtiny2313\Test\ 폴더에 <그림 3.1.21>과 같이 Test 폴더 하위 폴더인 default 폴더에 Test.hex파일로 생성된다.



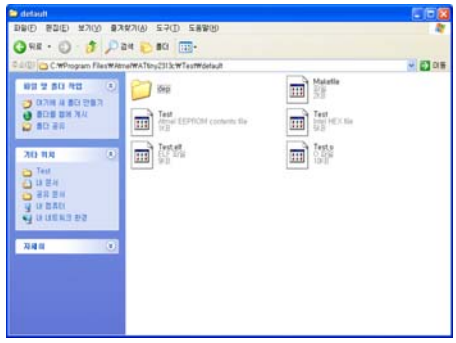
<그림 3.1.18> 프로그램 컴파일 전



<그림 3.1.19> 프로젝트 폴더



<그림 3.1.20> 프로그램 컴파일 후



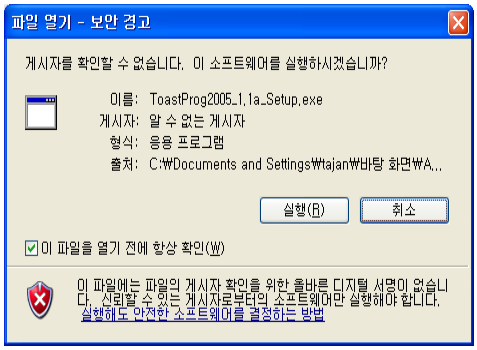
<그림 3.1.21> 컴파일 후 결과 파일

4. ToastProg2005를 사용한 HEX파일 다운로드

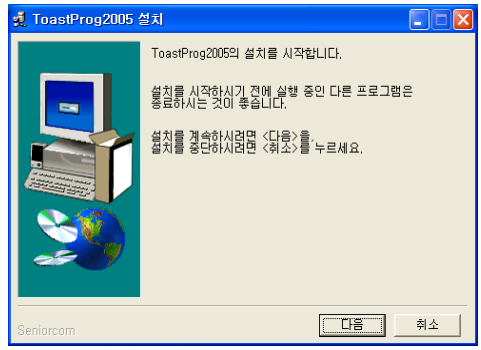
AVR Studio에서 컴파일된 HEX파일을 ATtiny2313의 플래시 프로그램 메모리에 다운로드 하기 위해서는 ISP 다운로드 프로그램인 'ToastPorg2005'와 다운로드 케이블이 필요하다.

(1) ToastProg2005 ISP 다운로드 프로그램 설치

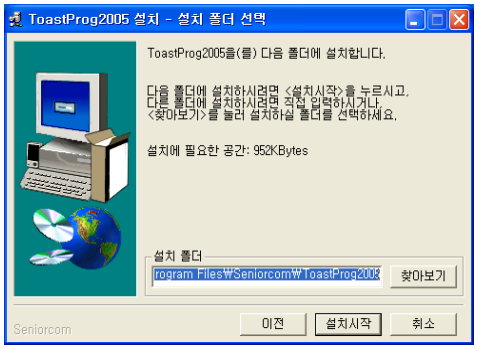
ToastProg2005는 시니어컴의 안도랑님이 개발한 무료 ISP 다운로드 프로그램이다. 이 프로그램은 다음 카페 http://cafe.daum.net/avr8051 에 가면 무료로 다운로드 받아 설치할 수 있다. ToastProg2005프로그램의 설치는 <그림 3.1.22>~<그림 3.1.25>과 같은 과정에 따라 설치한다.



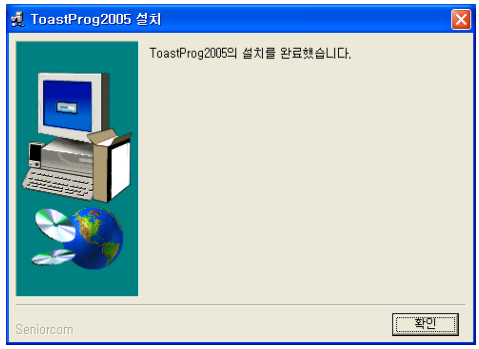
<그림 3.1.22> ToastProg2005설치1



<그림 3.1.23> ToastProg2005설치2



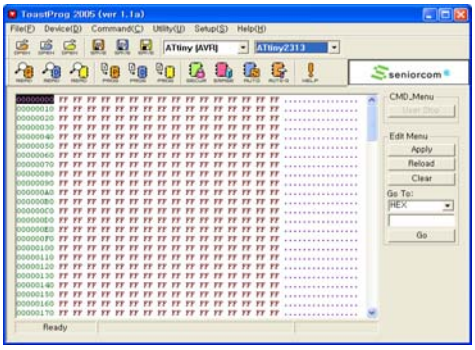
<그림 3.1.24> ToastProg2005설치3



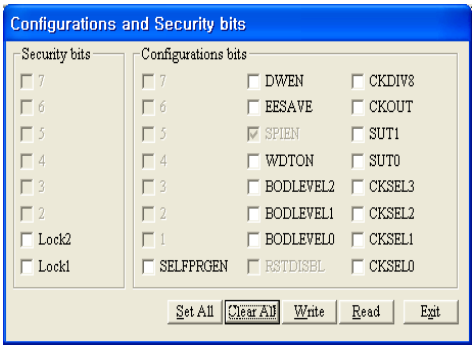
<그림 3.1.25> ToastProg2005설치4

ToastProg2005를 실행시키면 <그림 3.1.26>과 같다. 프린터 포트를 이용하는 ISP 케이블을 사용하기 위해서는 ToastProg2005의 Setup메뉴에서 Interface Setup을 LPT1으로 설정한다.

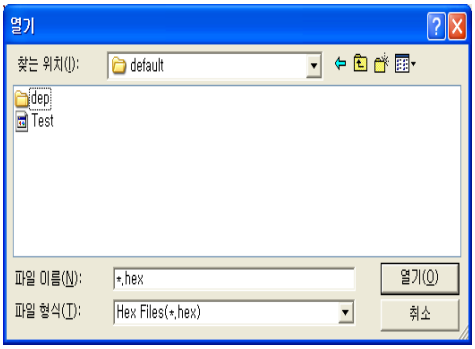
ATtiny2313의 시스템을 설정하기 위한 한 메모리 록 비트와 퓨즈 비트를 <그림 3.1.27>과 같이 ATtiny2313에 프로그램(Write)한다.



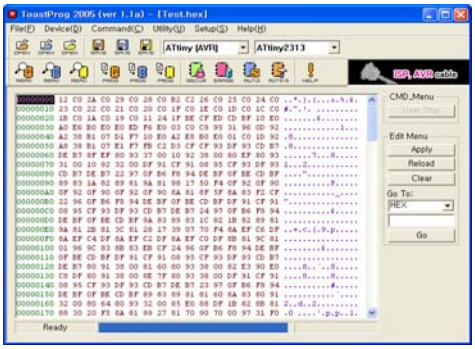
<그림 3.1.26> ToastProg2005실행



<그림 3.1.27> 퓨즈비트 설정



<그림 3.1.28> HEX 파일 열기 1



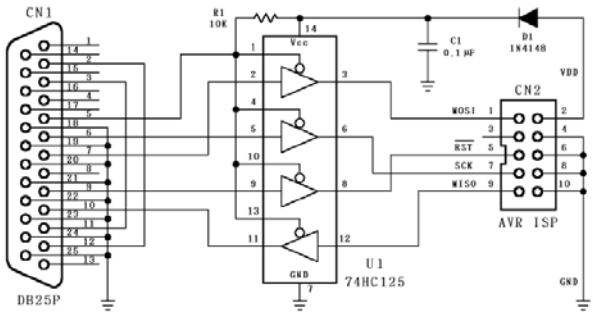
<그림 3.1.29> HEX 파일 열기 2

메모리 록 비트와 퓨즈 비트를 ATtiny2313에 프로그램 한 다음 <그림 3.1.28>과 같이 HEX파일을 열고, ToastProg2005의  아이콘을 클릭하여 HEX파일을 ISP 케이블을 통해 다운로드 한다.

(2) AVR ISP 다운로드 케이블

ToastProg2005 ISP 다운로드 프로그램을 이용하여 ATtiny2313의 플래시 프로그램 메모리에 HEX파일을 다운로드 하기 위해서는 ISP 다운로드 케이블이 필요하다.

다운로드 케이블은 Kanda System사의 스타터 키트 STK200/STK300에 호환하는 AVR ISP 케이블을 주로 사용한다. Kanda System사의 AVR ISP 케이블은 74HC244 소자를 사용하여 프린터 포트를 통해 AVR의 플래시 프로그램 메모리에 HEX파일을 다운로드 할 수 있게 하고 있으나, 이것을 <그림 3.1.30>과 같이 공주대학교 전기전자제어공학부 제어계측공학과 윤덕용 교수가 74HC125소자로 재설계하여 더욱 간단하게 개량한 <그림 3.1.31>과 같은AVR ISP 케이블이 널리 사용되고 있다.



<그림 3.1.30> AVR ISP 다운로드 케이블 회로도

<그림 3.1.31> AVR ISP 다운로드 케이블

3.2 AVR-GCC 컴파일러의 주요기능

1. ATtiny2313의 메모리에서의 데이터 표현 및 읽기

ATtiny2313의 메모리는 데이터 메모리인 SRAM, 프로그램 메모리인 플래시 메모리로 나눌 수 있다. AVR-GCC 컴파일러 사용시 이들 메모리에서의 데이터 표현 방법과 읽는 방법에 대해서 설명한다.

(1) SRAM에서의 데이터 표현

ATtiny2313에서 SRAM은 다른 메모리에 비해 액세스 속도가 빠르며, AVR-GCC에서 속성을 지정하지 않는 변수는 SRAM 영역에 저장된다. 데이터를 표현하기 위해서 **<io.h>를 인클루드**해야 한다.

<표 3.2.1>은 AVR-GCC에서 데이터를 표현하는 방법이다.

<표 3.2.1> AVR-GCC의 데이터 표현

C언어 표준	AVR-GCC의 표현	바이트수	표현범위
signed char	int8_t	1	-128~+127
unsigned char	uint8_t	1	0~255
signed int	int16_t	2	-32,768~+32,768
unsigned int	uint16_t	2	0~65,535
signed long	int32_t	4	-2,147,483,648~+2,147,483,648
unsigned long	uint32_t	4	0~4,294,967,295
singed long long	int64_t	8	-9.22×10^18~+9.22×10^18
unsigned long long	uint64_t	8	0~1.844×10^19

(2) 플래시 메모리에서의 상수표현

ATtiny2313의 프로그램 메모리인 플래시 메모리에는 프로그램이나 상수를 저장할 수 있다. SRAM의 경우에는 읽기와 쓰기가 모두 가능하지만, 플래시 메모리의 경우는 읽기만 가능하다. 플래시 메모리에서 상수를 표현하기 위해서는 **<pgmspace.h>를 인클루드**해야 한다.

<표 3.2.2>는 AVR-GCC에서 플래시 메모리에 상수를 표현하는 방법이다.

<표 3.2.2> AVR-GCC에서 플래시 메모리의 상수 표현

C언어 표준	AVR-GCC의 표현	바이트수	표현범위
signed char	prog_char prog_int8_t	1	-128~+127
unsigned char	prog_uchar prog_uint8_t	1	0~255
signed int	prog_int16_t	2	-32,768~+32,768
unsigned int	prog_uint16_t	2	0~65,535
signed long	prog_int32_t	4	-2,147,483,648~+2,147,483,648
unsigned long	prog_uint32_t	4	0~4,294,967,295
singed long long	prog_int64_t	8	-9.22×10^18~+9.22×10^18
unsigned long long	prog_uint64_t	8	0~1.844×10^19

ATtiny2313의 플래시 프로그램 메모리에는 8비트, 16비트, 32비트 등의 상수 데이터를 표현할 수 있으며 이러한 프로그램 메모리 상수 데이터를 다음과 같은 방법으로 읽는다.

1) signed char 형의 정의와 읽기

```
prog_int8_t data ={5};                    /* 정의 */

int8_t candy;
candy = pgm_read_byte(&data);             /* 읽기 */
```

2) signed char 형 배열의 정의와 읽기

```
prog_int8_t data1[5] = {0,1,2,3,4};       /* 정의 */

int8_t candy;
candy = pgm_read_byte(&data1[2]);         /* 읽기 */
```

2. ATtiny2313의 I/O 포트 액세스

AVR-GCC에서 ATtiny2313의 I/O 포트를 액세스 하는 방법을 요약하면 <표 3.2.3>과 같다. I/O 포트 액세스를 위해서는 **<io.h>를 인클루드**해야 한다

<표 3.2.3> AVR-GCC에서 I/O 액세스 방법

엑세스 방법	avr-libc v1.2.3이전	기 능
PORTB = PORTB | _BV(0);	sbi(PORTB,0);	특정 비트 set
PORTB = PORTB & ~_BV(1);	cbi(PORTB,1);	특정 비트 reset
PORTB = 0xFF;	outp(0xFF,PORTB);	포트로 0xFF 출력
DDRB = 0xFF;	outp(0xFF,DDRB);	포트를 출력으로 설정
DDRB = 0x00;	outp(0x00,DDRB);	포트를 입력으로 설정
unsigned char data; data = PINB;	unsigned char data; data = inp(PINB);	포트로 데이터 읽기
bit_is_set(PINB,3);		특정비트가 set되었는지 조사
bit_is_clear(PINB,3);		특정비트가 reset되었는지 조사

3. 인터럽트 처리

AVR-GCC에서 인터럽트 서비스 루틴을 처리하기 위해서는 지정된 형식을 지켜야 하며, 이를 위해서는 **<interrupt.h>를 인클루드**해야 한다.

```
#include <avr/interrupt.h>
  ISR(vector)
  {
  /* Interrupt service routine
  }
```

ATtiny2313에서 인터럽트가 발생하면 SREG의 글로벌 인터럽트 플랙 I=0이 되므로 인터럽트 함수가 실행되는 동안에는 인터럽트 발생이 금지된다. 이때 인터럽트를 허용하려면 인터럽트 함수의 서두에 sei() 함수를 실행하고, 인터럽트를 금지하려면 cli() 함수를 실행하면된다.

ATtiny2313의 인터럽트 처리를 위한 vector는 <표 3.2.4>와 같다.

<표 3.2.4> ATtiny2313의 인터럽트 처리를 위한 vector

번호	vector	old vector	설 명
1	ANA_COMP_vect	SIG_COMPARATOR	아날로그 비교기
2	EEPROM_READY_vect	SIG_EEPROM_READY, SIG_EE_READY	EEPROM 준비
3	INT0_vect	SIG_INTERRUPT0	외부 인터럽트0
4	INT1_vect	SIG_INTERRUPT1	외부 인터럽트1
5	PCINT_vect	SIG_PIN_CHANGE, SIG_PCINT	핀 변화 인터럽트
6	TIMER0_COMPA_vect	SIG_OUTPUT_COMPARE0A	타이머/카운터0 비교 매치A
7	TIMER0_COMPB_vect	SIG_OUTPUT_COMPARE0B, SIG_OUTPUT_COMPARE0_B	타이머/카운터0 비교 매치B
8	TIMER0_OVF_vect	SIG_OVERFLOW0	타이머/카운터0 오버플로우
9	TIMER0_CAPT_vect	SIG_INPUT_CAPTURE1	타이머/카운터 캡쳐
10	TIMER1_COMPA_vect	SIG_OUTPUT_COMPARE1A	타이머/카운터1 비교 매치A
11	TIMER1_COMPB_vect	SIG_OUTPUT_COMPARE1B	타이머/카운터1 비교 매치B
12	TIMER1_OVF_vect	SIG_OVERFLOW1	타이머/카운터1 오버플로우
13	USART_RX_vect	SIG_USART_RECV, SIG_UART_RECV	USART Rx 완료
14	USART_UDRE_vect	SIG_USART_DATA, SIG_UART_DATA	USART 데이터 레지스터준비
15	USI_OVERFLOW_vect	SIG_USI_OVERFLOW	USI 오버플로우
16	WDT_OVERFLOW_vect	SIG_WATCHDOG_TIMEOUT, SIG_WDT_OVERFLOW	워치독 타이머 오버플로우

4. 인라인 어셈블

AVR-GCC에서는 어셈블리 명령을 사용하여(인라인 어셈블) C언어의 소스에 포함시킬 수 있다. 인라인 어셈블 명령을 사용하여 시간지연 함수를 만드는 예를 보면 다음과 같다.

```
void Delay_us(unsigned char time_us)
{ register unsigned char i;
  for( i = 0; i < time_us; i++)        /* 4 cycles */
    { asm volatile(" PUSH R0 ");   /* 2 cycles */
      asm volatile(" POP   R0 ");  /* 2 cycles */
```

```
        asm volatile(" PUSH R0 ");   /* 2 cycles */
        asm volatile(" POP  R0 ");   /* 2 cycles */
        asm volatile(" PUSH R0 ");   /* 2 cycles */
        asm volatile(" POP  R0 ");   /* 2 cycles = 16cycles = 1us */
    }
}
```

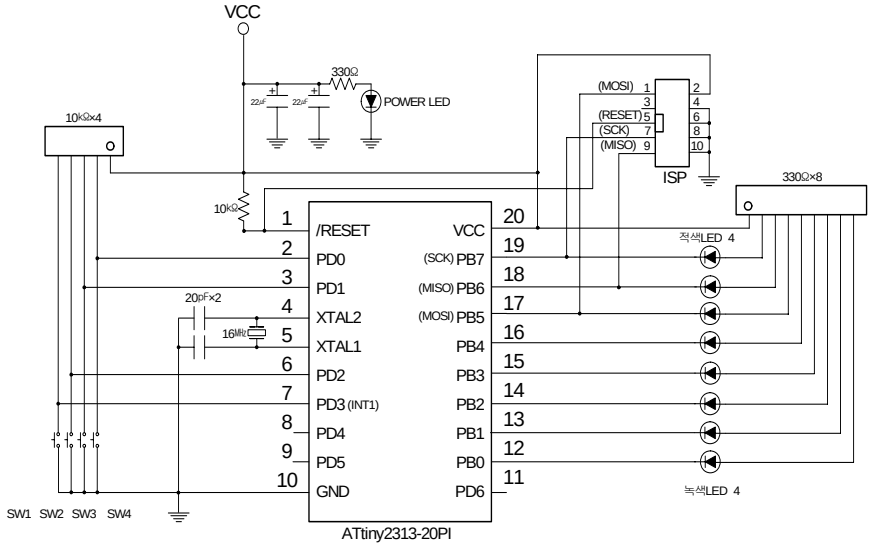
3.3 프로그램 예제

1. 기본 예제

AVR Studio와 WinAVR의 AVR-GCC컴파일러를 이용한 기본 예제 프로그램을 작성하여 실행하기 위해 <표 3.3.1>에 제시된 부품을 이용하여 <그림 3.3.1>과 같은 하드웨어를 만능기판에 구성한다.

<표 3.3.1> 기본 예제를 위한 하드웨어 부품목록

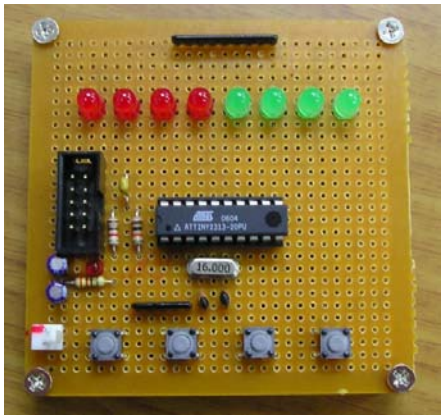
부품	규격	수량	부품	규격	수량
IC	ATtiny2313-20PU	1	저항	10kΩ	1
오실레이터	16MHz ATS형	1	저항	1kΩ	1
LED	적색 5φ	4	저항	330Ω	1
LED	녹색 5φ	4	어레이저항	330Ω (9핀)	1
LED	적색 3φ	1	어레이저항	10kΩ (5핀)	1
IC소켓	20핀	1	세라믹콘덴서	22pF	2
택트스위치	4핀	4	박스형콘넥터	3FB-10P-S/T	1
만능기판	28×28	1	SIP 헤더	2핀	1



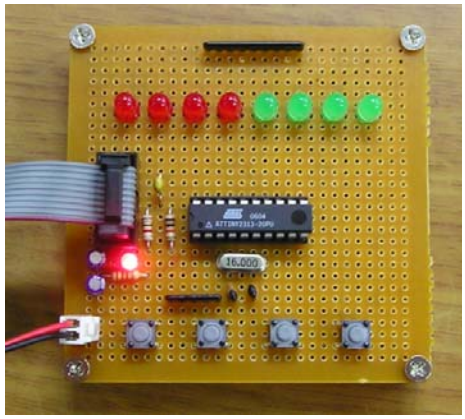
<그림 3.3.1> LED 제어회로

<그림 3.3.1>의 LED제어 회로를 이용하여 기본적인 프로그램을 작성하고 AVR-GCC컴파일러를 이용하여 컴파일을 수행하고 HEX파일 ATtiny2313의 플래시 프로그램 메모리에 다운로드 하여 LED제어회로를 동작시켜 본다.

<그림 3.3.2>는 <그림 3.3.1>의 회로를 만능기판에 완성한 모습을 보여주며, <그림 3.3.3>은 ATtiny2313을 이용한 LED제어회로에 플래시 프로그램 메모리에 HEX파일을 다운로드 하기 위한 ISP케이블과 전원을 연결한 모습이다.



<그림 3.3.2> LED제어회로 1



<그림 3.3.3> LED제어회로 2

(1) 프로그램 예제 1

<그림 3.3.1>의 회로에서 LED는 PORTB에 접속되어 있으며, PORTB의 출력이 '0'일때 LED가 점등된다. PORTB에 접속된 LED의 상위 4비트와 하위 4비트를 구분하기 쉽게 하기 위해 상위 4비트는 적색LED를 사용하고, 하위 4비트는 녹색 LED를 사용하여 회로를 구성한다.

상위 4비트와 하위 4비트의 LED가 1초 간격으로 점멸되는 프로그램은 다음과 같다.

```
/* ========================== */
/*  LED1.C                    */
/* ========================== */
#include <avr/io.h>

void Delay_us(unsigned char time_us)
{ register unsigned char i;
  for(i = 0; i < time_us; i++)
    { asm volatile(" PUSH R0");
      asm volatile(" POP R0");
      asm volatile(" PUSH R0");
      asm volatile(" POP R0");
    }
}

void Delay_ms(unsigned int time_ms)
{ register unsigned int i;
  for(i = 0; i < time_ms; i++)
    { Dealy_us(250);
      Dealy_us(250);
      Dealy_us(250);
      Dealy_us(250);
    }
}

int main(void)
{
  DDRB = 0xFF;
  while(1)
    { PORTB = 0xF0;
      Delay_ms(1000);
      PORTB = 0x0F;
      Delay_ms(1000);
     }
}
```

이 프로그램은 메인함수 main()와 두개의 시간지연함수 Delay_us()와 Delay_ms()함수로 구성되어 있다. 시간지연함수 Delay_us()은 '인라인 어셈블'을 이용하여 정확한 시간지연을 얻기 위한 것이다. for루프에서 4 cycles, PUSH와 POP에서 각각 2 cycles이다. 시스템 클록이 16MHz인 경우에 16 cycles이 1us이다. NOP명령은 1 cycle이므로 NOP명령을 for 루프에 16번 넣어도 같은 시간지연을 얻을 수 있다.

Delay_ms()함수는 Delay_us()를 기초로하여 for루프에서 원하는 시간만큼 돌려서 얻을 수 있다. Delay_us()함수에서는 for루프의 변수형이 'unsigned char'이지만 Delay_ms()의 경우에는 'unsigned int'라는 것에 주

의해야 한다. unsigned char의 표현 범위는 (0~255)이고 unsigned int의 표현 범위는 (0~65,535)이다.

main()함수에서는 PORTB로 데이터를 출력하기 전에 PORTB의 출력 방향을 설정해주어야 한다. DDRB는 포트B의 데이터 방향을 설정하는 레지스터이다. '0'일 경우에는 입력이고, '1'일 경우에는 출력이다. LED를 점등하는 데이터를 출력하기 위해서는 포트B를 모두 '1'로 설정해 주어야 한다. 8비트 이므로 2진수로 '11111111'이며, 이것을 16진수로 표현하면 'FF'가 된다.

시간지연함수 Delay_us()와 Delay_ms()를 별도의 파일로 만들어 두고 인클루드 시켜주면 매번 시간지연함수를 작성하는 수고를 하지 않아도 된다.

```
/* ========================== */
/*  ATtiny2313.h              */
/* ========================== */

void Delay_us(unsigned char time_us)
{ register unsigned char i;
  for(i = 0; i < time_us; i++)
    { asm volatile(" PUSH R0");
      asm volatile(" POP R0");
      asm volatile(" PUSH R0");
      asm volatile(" POP R0");
    }
}

void Delay_ms(unsigned int time_ms)
{ register unsigned int i;
  for(i = 0; i < time_ms; i++)
    { Dealy_us(250);
      Dealy_us(250);
      Dealy_us(250);
      Dealy_us(250);
    }
}
```

파일명을 'ATtiny2313.h'로 하여 저장하도록 한다. ATtiny2313.h를 LED1.C에 #include "ATtiny2313.h" 로 하여 인클루드 시키면 프로그램이 간단해 진다. ATtiny2313.h는 반드시 LED1.C와 같은 폴더에 있어야 한다.

```
/* ========================== */
/*  LED1_1.C                  */
/* ========================== */
#include <avr/io.h>
#include "ATtiny2313.h"

int main(void)
{
  DDRB = 0xFF;
  while(1)
    { PORTB = 0xF0;
      Delay_ms(1000);
      PORTB = 0x0F;
      Delay_ms(1000);
     }
}
```

출력 데이터값을 led 변수로 출력하는 프로그램을 작성해 보도록 한다. 데이터값을 '좌로 시프트'하는 명령을 사용하여 LED의 이동 방향을 보도록 한다.

```
/* ========================== */
/*  LED2.C                    */
/* ========================== */
#include <avr/io.h>
#include "ATtiny2313.h"

int main(void)
{ unsigned char i, led;
  DDRB = 0xFF;
  led  = 0xFE;

  for(i = 0; i < 8; i++)
    { PORTB = led;
      led = led << 1;
      Delay_ms(1000);
     }

  PORTB = 0xFF;

  while(1);
}
```

PORTB0 부터 차례로 LED가 점등된다. 좌로 시프트(<<)하여 비트를 이동시키면, 차례로 비트값을 '0'으로 만든다는 것을 알 수 있다. led = led << 1;은 led <<= 1;로 표기하여도 같다.

```
/* ========================== */
/*  LED3.C                    */
/* ========================== */

#include <avr/io.h>

int main(void)
{ unsigned char key;
  DDRB = 0xFF;          // PortB Output
  PORTB = 0xFF;         // PortB initialize for all LED OFF
  DDRD = 0xF0;          // PortD Lower 4bit Input
  PORTD = 0x00;         // PortD initialize

  while(1)
    {
      key = PIND & 0b00001111;
      switch(key)
        { case 0x0E: PORTB = 0b11111110; // Pressed SW1
                     break;
          case 0x0D: PORTB = 0b11111101; // Pressed SW2
                     break;
          case 0x0B: PORTB = 0b11111011; // Pressed SW3
                     break;
          case 0x07: PORTB = 0b11110111; // Pressed SW4
                     break;
          default:   break;
            }
    }
}
```

LED3.C 예제에서는 스위치 SW1~SW4를 누르면 PORTB의 하위 4비트 LED가 점등되도록 하는 프로그램이다. key변수에는 누른 스위치 값이 저장되고, switch{}문에서 key에 저장된 값과 같은 case를 실행한다.

PORTB로 출력되는 값을 2진수로 표기하였다. SW1를 누른 경우 PORTB로 출력되는 2진수값은 0b11111110로 하여 PORTB0의 LED가 점등된다. 이것을 16진수값으로 나타내면 0xFE이다. 즉, PORTB=0xFE;로 바꾸어도 마찬가지 결과가 된다.

```
/* ========================== */
/*  LED4.C                    */
/* ========================== */

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

ISR(TIMER1_COMPA_vect)
  {
    PORTB = PORTB ^ 0x01;  PortB0 for LED ON
   }

int main(void)
{ DDRB = 0xFF;           // PortB Output
  PORTB = 0xFF;          // PortB initialize

  TCCR1A = 0x00;         // CTC mode(4), don't output OC1A
  TCCR1B = 0x0C;         // 16MHz/256/(1+62499) = 1Hz
  TCCR1C = 0x00;
  OCR1A  = 62499;
  TCNT1  = 0x0000;       // clear Timer/Counter1

  TIMSK = 0x40;          // enable OC1A interrupt
  TIFR  = 0xE8;          // clear all interrupt flags
  sei();

  while(1);
}
```

타이머/카운터1을 이용하여 1초 간격으로 PORTB0의 LED를 점멸하는 프로그램이다. 타이머/카운터1은 1초 간격으로 인터럽트를 발생시키고, 인터럽트 서비스 루틴 ISR(Timer1_COMPA_vect)에서는 PORTB0의 LED를 점멸하는 일을 한다.

OC1A핀으로 파형을 출력하지 않는 CTC모드4를 타이머/카운터1 제어 레지스터 TCCR1A와 TCCR1B의 파형 발생 비트 WGM13~WGM10으로 설정한다.

```
  TCCR1A = 0x00;         // CTC mode(4), don't output OC1A
  TCCR1B = 0x0C;         // 16MHz/256/(1+62499) = 1Hz
  TCCR1C = 0x00;
  OCR1A  = 62499;
  TCNT1  = 0x0000;       // clear Timer/Counter1
```

<표 2.2.7>의 WGM13~WGM10 비트에 의한 동작모드 설정에서 CTC mode 4를 선택하려면, TCCR1A의 WGM11과 WGM10은 각각 '0'이고, TCCR1B의 WGM13과 WGM12는 각각 '0'과 '1'로 설정되어야 한다.

<표 2.2.8>의 OC1x핀 기능 설정에서 OC1A핀으로 파형을 출력하지 않으므로 COM1A1,COM1A0,COM1B1,COM1B0는 '0000'으로 설정되어야 한다. TCCR1A의 COM1A1~COM1B0비트에 이 값을 대입하면되다.

<표 2.2.9>의 클록소스 선택비트 설정에서 프리스케일러의 분주비를 256으로 선택하면 CS12, CS11, CS10은 '100'이다. 이 값은 TCCR1B의 CS12, CS11, CS10비트 대입한다. 프리스케일러가 정해지면, OC1A의 출력 파형 주파수를 얻기 위한 타이머/카운터1의 출력 비교 레지스터 OCR1A의 값이 정해진다. CTC 모드에서 출력비교 인터럽트가 2번 발생하여 1주기의 출력파형을 만들기 때문에 OCR1A(=TOP)값을 구하는 식은 다음과 같다.

$$f_{OC_x} = \frac{f_{CPU_{clock}}}{N \cdot (1+TOP)} \quad \Rightarrow \quad TOP = \frac{16,000,000}{(256 \cdot f_{OC_x})} - 1$$

f_{OC_x}가 1Hz이므로 OCR1A의 값은 16,000,000/256 - 1 = 62,499이다.

비트	7	6	5	4	3	2	1	0	
0x2F(0x4F)	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10	TCCR1A
읽기 / 쓰기	R/W	R/W	R/W	R/W	R	R	R/W	R/W	

비트	7	6	5	4	3	2	1	0	
0x2E(0x4E)	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
읽기 / 쓰기	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	

모드	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (WPM10)	동작모드	최대값	OCR1x 업데이트	TOV1플랙
4	0	1	0	0	CTC	OCR1A	설정즉시	0xFFFF

모드	COM1x1	COM1x0	OC1x핀의 기능
PWM모드가 아닌 경우	0	0	정상적인 I/O포트로 동작, OC1x 출력차단
	0	1	비교 매치에서 OC1x 출력을 토글
	1	0	비교 매치에서 OC1x = 0
	1	1	비교 매치에서 OC1x = 1

CS12	CS11	CS10	클록 소스
1	0	0	$clk_{I/O}$ / 256

```
TIMSK = 0x40;          // enable OC1A interrupt
TIFR  = 0xE8;          // clear all interrupt flags
```

OC1A의 인터럽트를 허용하려면, 타이머/카운터1의 타이머인터럽트 마스크 레지스터 TIMSK의 OCIE1A=1로 한다.

타이머/카운터1의 모든 인터럽트 플랙을 '0'로 하려면, 타이머/카운터1의 타이머 인터럽트 플랙 레지스터 TIFR의 TOV1=1, OCF1A=1, ICF1=1으로 설정한다.

비트	7	6	5	4	3	2	1	0	
0x39(0x59)	TOIE1	OCIE1A	OCIE1B	-	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
읽기 / 쓰기	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	

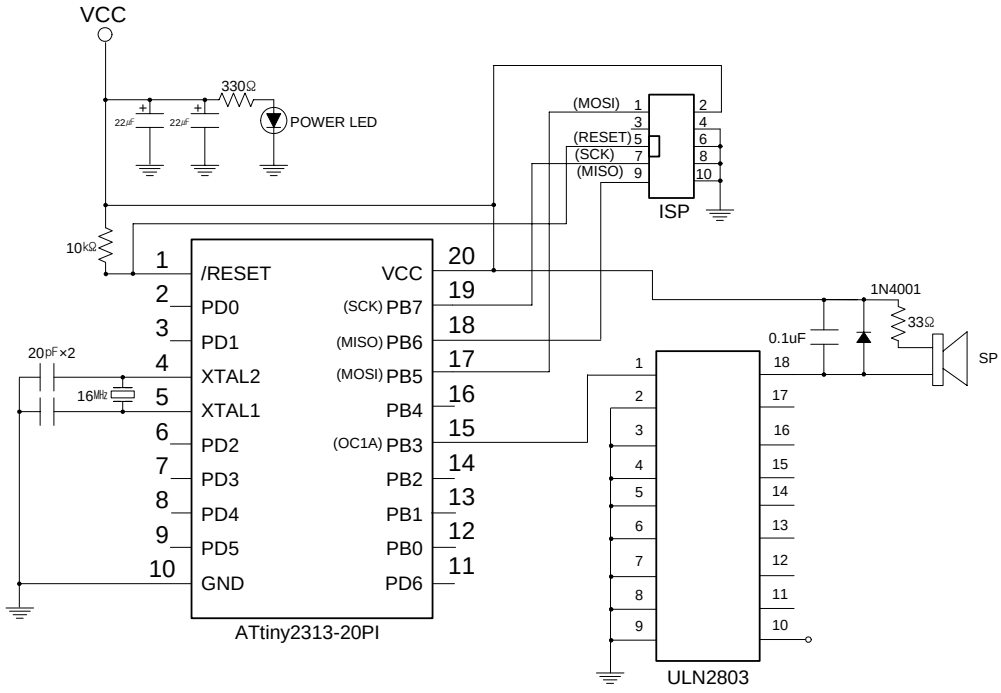
비트	7	6	5	4	3	2	1	0	
0x38(0x58)	TOV1	OCF1A	OCF1B	-	ICF1	OCF0B	TOV0	OCF0A	TIFR
읽기 / 쓰기	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	

2. 뮤직박스 예제

ATtiny2313의 타이머/카운터1의 CTC모드를 사용하여 스피커를 통해 음악을 연주하는 프로그램을 작성하기 위해 <표 3.3.2>에 제시된 부품을 이용하여 <그림 3.3.4>의 회로를 만능기판에 구성한다.

<표 3.3.2> 뮤직박스 부품목록

부품	규격	수량	부품	규격	수량
IC	ATtiny2313-20PU	1	저항	10kΩ	1
오실레이터	16MHz ATS형	1	저항	1kΩ	1
다이오드	1N4001	1	저항	330Ω	1
LED	적색 3φ	1	저항	33Ω	1
IC소켓	20핀	1	IC소켓	18핀	1
스피커	0.5W 8Ω	1	세라믹콘덴서	22pF	2
택트스위치	4핀	1	박스형콘넥터	3FB-10P-S/T	1
만능기판	28×28	1	SIP 헤더	2핀	1



<그림 3.3.4> 뮤직박스 회로

```
/* ========================== */
/*  ATtiny2313.h  for MUSIC   */
/* ========================== */

#define VLA     9008    // 2 OCTAVE
#define VLAX    8580
#define VLB     8098
#define LC      7644    // 3 OCTAVE
#define LCX     7214
#define LD      6810
#define LDX     6427
#define LE      6066
#define LF      5726
#define LFX     5404
#define LG      5101
#define LGX     4815
#define LA      4544
#define LAX     4289
#define LB      4049
#define MC      3821    // 4 OCTAVE
#define MCX     3607
#define MD      3404
```

```
#define MDX     3213
#define ME      3033
#define MF      2862
#define MFX     2702
#define MG      2550
#define MGX     2407
#define MA      2272
#define MAX     2144
#define MB      2024
#define HC      1910    // 5 OCTAVE
#define HCX     1803
#define HD      1702
#define HDX     1606
#define HE      1516
#define HF      1431
#define HFX     1350
#define HG      1275
#define HGX     1203
#define HA      1135
#define HAX     1072
#define HB      1011
#define VHC     955     // 6 OCTAVE
#define VHCX    901
#define VHD     850
#define VHDX    803
#define VHE     757

#define NOTE32  1*3     // define note length
#define NOTE16  2*3
#define NOTE16D 3*3
#define NOTE16T 2*2
#define NOTE8   4*3
#define NOTE8D  6*3
#define NOTE8T  4*2
#define NOTE4   8*3
#define NOTE4D  12*3
#define NOTE4T  8*2
#define NOTE2   16*3
#define NOTE2D  24*3
#define NOTE1   32*3

void MCU_initialize(void)       /* initialize ATtiny2313 MCU */
{
  DDRB = 0xFF;          // PORTB = output
  PORTB = 0x00;
  DDRD = 0xFF;          // PORTD = output
  PORTD = 0x00;
}

void Delay_us(unsigned char time_us)    /* time delay for us */
{ register unsigned char i;

  for(i = 0; i < time_us; i++)          // 4 cycle +
    { asm volatile(" PUSH  R0 ");       // 2 cycle +
```

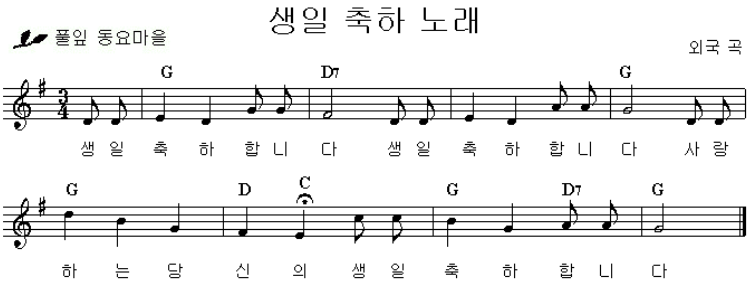
```
      asm volatile(" POP   R0 ");       // 2 cycle +
      asm volatile(" PUSH  R0 ");       // 2 cycle +
      asm volatile(" POP   R0 ");       // 2 cycle +
      asm volatile(" PUSH  R0 ");       // 2 cycle +
      asm volatile(" POP   R0 ");       // 2 cycle = 16 cycle = 1us
    }
}

void Delay_ms(unsigned int time_ms)     /* time delay for ms */
{ register unsigned int i;

  for(i = 0; i < time_ms; i++)
    { Delay_us(250);
      Delay_us(250);
      Delay_us(250);
      Delay_us(250);
    }
}
```

□----------□

□----------□

```
 /* ========================== */
 /*  music_box.c               */
 /* ========================== */

#include <avr/io.h>
#include "ATtiny2313.h"

unsigned char tempo;

void Set_tempo(unsigned char number)  /* set tempo */
{
  tempo = number;                     // set tempo
  TCCR1A = 0x40;                      // CTC mode(12), use OC1A
  TCCR1B = 0x18;                      // speaker off
  TCCR1C = 0x00;
}

void Play_note(unsigned int sound, unsigned int note) /* play note */
{
  ICR1= sound;                        // set ICR1
  TCNT1 = 0x0000;                     // clear Timer/Counter1
  TCCR1B = 0x1A;                      // start CTC mode(prescaler = 8)

  Delay_ms(note*tempo*7);             // note play time

  TCCR1B = 0x18;                      // speaker off
}
```

```
int main(void)
{
  MCU_initialize();                    // initialize MCU
  Delay_ms(50);                        // wait for system stabilization

      Set_tempo(4);                    // set tempo

      Play_note(MD,NOTE8);             // Line 1
      Play_note(MD,NOTE8);
      Play_note(ME,NOTE4);
      Play_note(MD,NOTE4);
      Play_note(MG,NOTE8);
      Play_note(MG,NOTE8);
      Play_note(MFX,NOTE2);
      Play_note(MD,NOTE8);
      Play_note(MD,NOTE8);
      Play_note(ME,NOTE4);
      Play_note(MD,NOTE4);
      Play_note(MA,NOTE8);
      Play_note(MA,NOTE8);
      Play_note(MG,NOTE2);
      Play_note(MD,NOTE8);
      Play_note(MD,NOTE8);

      Play_note(HD,NOTE4);              // Line 2
      Play_note(MB,NOTE4);
      Play_note(MG,NOTE4);
      Play_note(MFX,NOTE4);
      Play_note(ME,NOTE4);
      Play_note(HC,NOTE8);
      Play_note(HC,NOTE8);
      Play_note(MB,NOTE4);
      Play_note(MG,NOTE4);
      Play_note(MA,NOTE8);
      Play_note(MA,NOTE8);
      Play_note(MG,NOTE2);

      Delay_ms(1000);

      while(1);
}
```



<그림 3.3.4> 생일축하 노래 악보