타이머

마이크로프로세서

HRI 연구실

김동한

6.1 LED 제어

6.1.5. ATmega128의 타이머/카운터 이해하기

- 사람의 시간표 **12**시 **(**점심시간**),** **19**시 **(**퇴근**),** **23** 시 **(**취침**)**
  라면 끓이기 **4**분 **30**초
  사람이 시간을 알려면 시계가 필요

- **ATmega 128**도 마찬가지로 시간을 알려면 시계가 필요
  그래서 사용하는 것이 "타이머/카운터"

- 클럭의 단위 **: Hz (**주파수 **: 1**초당 발생 횟수**)**

- 주기 **:** 클럭의 역수**,** 한 개의 **Pulse**가 발생되는 시간

- **ex) 1**초에 **6** 번**, 6Hz ,** 그림의 주기 **: 1/6Hz = 0.17**초

KYUNG HEE UNIVERSITY

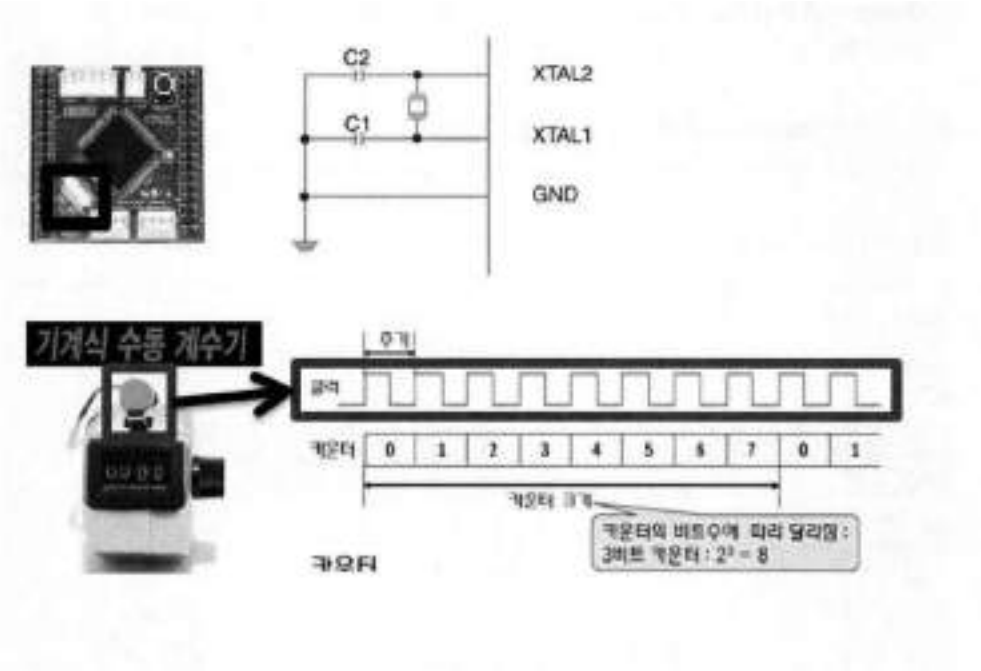
6.1 LED 제어

6.1.6. Clock과 카운터의 이해

- 클럭은 일정한 시간 간격으로 **0** 과 **1**의 값이 번갈아 나타나는 펄스

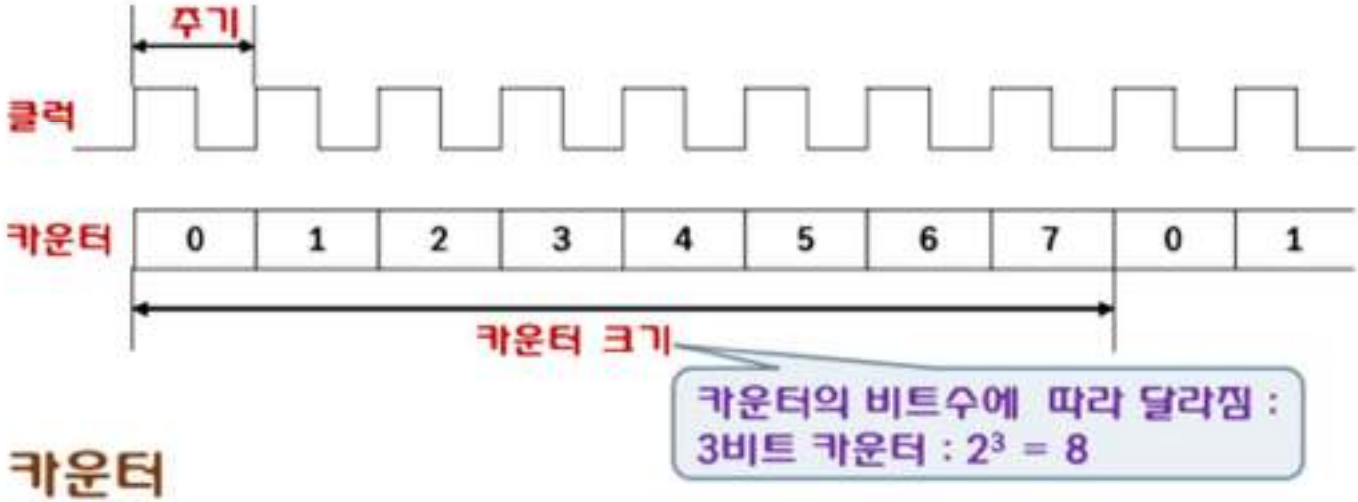
- **ATmega128**은 클럭을 기준으로 동작

- 수정발진자**(Crystal, X-TAL)**을 사용하여 클럭을 생성



중요. 타이머와 카운터

- **클럭**
  - 시계
  - 일정한 시간 간격으로 0과 1의 값이 번갈아 나타남
  - 주어진 일을 순서대로 정확한 시간에 처리하기 위해 사용



- **카운터**
  - 클럭을 세는 장치

중요. 타이머와 카운터

타이머와 카운터

- **타이머** : MCU 내부 클럭을 세는 장치
  - 동기모드
  - 타이머는 MCU의 내부클럭을 세어 일정시간 간격의 펄스를 만들어 내거나 일정시간 경과 후에 인터럽트를 발생
- **카운터** : MCU의 외부에서 입력되는 클럭을 세는 장치
  - 비동기모드
  - 카운터는 외부 핀(TOSC1, TOSC2, T1, T2, T3)을 통해서 들어오는 펄스를 계수(Edge Detector)하여 Event Counter로서 동작

중요. 타이머와 카운터

- ATMega128의 타이머 카운터
  - 타이머 0~3 모두 4개의 타이머/카운터를 보유
    - 타이머 0,2 : 8비트 타이머로 서로 기능 유사
    - 타이머 1,3 : 16비트 타이머로 서로 기능 유사
  - 인터럽트 기능
    - 오버플로우 인터럽트 : 카운터의 값이 오버플로우되는 경우 발생
    - 출력비교 인터럽트 : 카운터 값이 출력비교 레지스터의 값과 같게 되는 순간에 발생
    - 입력 캡쳐 인터럽트 : 외부로부터의 트리거 신호에 의해서 카운터의 초기값을 입력캡쳐
  - PWM 출력 기능
    - Pulse Width Modulation

중요. 타이머와 카운터

- 8비트 타이머/카운터의 특징
  - 4개의 타이머/카운터 중 0번과 2번 타이머/카운터
  - PWM 및 비동기 동작 모드를 갖는 8비트 업/다운(Up/Down) 카운터
  - 8비트 카운터 : 2^8 = 256, 즉 0~255까지 셀 수 있음
  - 10비트의 프리스케일러(prescaler) 보유
  - 각종 인터럽트 기능
    - 오버플로우 인터럽트(overflow interrupt)
    - 출력비교 인터럽트(output compare match interrupt)
  - PWM 기능 제공
  - 타이머 0는 부가적으로 32.768KHz의 수정 진동자를 TOSC1,2 단자에 연결해 Real Time Clock으로 사용할 수 있는 기능 제공

중요. 타이머와 카운터

- 8비트 타이머/카운터 레지스터
  - 타이머/카운터 제어 레지스터(TCCRn)
  - 타이머/카운터 레지스터(TCNTn)
  - 출력 비교 레지스터(OCRn)
  - 인터럽트 관련
    - 타이머/카운터 인터럽트 플래그 레지스터(TIFR)
    - 타이머/카운터 인터럽트 마스크 레지스터 (TIMSK)

중요. 타이머와 카운터

- **프리스케일러(Prescaler)**
  - 고속의 클럭을 사용하여 타이머를 동작시킬 때 나타나는 문제를 해결하기 위해 클럭을 분주하여 더 느린 타이머 클럭을 만듦.
    - 4MHz 클럭을 사용하는 경우 그 클럭의 주기는 250ns
    - 이 클럭으로 256까지 센다고 해도 64us 이하를 세는 카운터밖에는 만들 수가 없음.
  - 10비트, 프리스케일러 보유(최대 2^{10} = 1024배 가능)
    - 프리스케일러를 1024로 하면 4MHz 클럭의 타이머 클럭은 주기가 250ns*1024=256us가 되고, 이 클럭을 256까지 센다면 256us*256=65.536ms 크기의 타이머를 만들 수 있음.

중요. 타이머와 카운터

	타이머/카운터0	타이머/카운터1	타이머/카운터2	타이머/카운터3
기본구조	8bit	16bit	8bit	16bit
타이머입력	clk I/O	clk I/O	clk I/O	clk I/O
카운터입력	TOSC1와TOSC2 (32.768kHz) TOSC1	T1	T2	T3
타이머 프리스케일러	1,8,32,64, 128,256,1024	1,8,64,256,1024	1,8,64,256,1024	1,8,64,256,1024
레지스터	TCCR0 TCNT0 OCR0 ASSR SFIOR TIMSK TIFR	TCCR1A TCCR1B TCCR1C TCNT1H, TCNT1L OCR1AH, OCR1AL OCR1BH, OCR1BL OCR1CH, OCR1CL ICR1H, ICR1L SFIOR TIMSK, ETIMSK TIFR, ETIFR	TCCR2 TCNT2 OCR2 SFIOR TIMSK TIFR	TCCR3A TCCR3B TCCR3C TCNT3H, TCNT3L OCR3AH, OCR3AL OCR3BH, OCR3BL OCR3CH, OCR3CL ICR3H, ICR3L SFIOR TIMSK, ETIMSK TIFR, ETIFR

중요. 타이머와 카운터

	타이머/카운터0	타이머/카운터1	타이머/카운터2	타이머/카운터3
동작모드	Normal CTC Fast PWM Phase Correct PWM	Normal, CTC, Fast PWM, Phase Correct PWM, Phase and Frequency Correct PWM	Normal CTC Fast PWM Phase Correct PWM	Normal, CTC, Fast PWM, Phase Correct PWM, Phase and Frequency Correct PWM
입력신호	TOSC1, TOSC2	T1, IC1	T2	T3, IC3
출력신호	OC0	OC1A OC1B OC1C	OC2	OC3A OC3B OC3C
인터럽트	Overflow, Output Compare Match	Overflow, Output Compare Match A/B/C, Input Capture	Overflow, Output Compare Match	Overflow, Output Compare Match A/B/C, Input Capture
기타	RTC기능 타이머카운터모두 프리스케일러 사용	Capture		Capture

중요. 타이머와 카운터

- Normal Mode(일반 동작모드)
  - 카운터는 업 카운터로서만 동작
    - MAX(0xFF)값이 되면, BOTTOM(0x00)값부터 다시 시작
    - MAX 위치에서 오버플로우 인터럽트 발생
  - TCNTn의 초기값을 설정하여 전체 타이머 주기를 결정
  - TCCRn레지스터의 WGMn1:n0 = 00으로 설정

중요. 타이머와 카운터

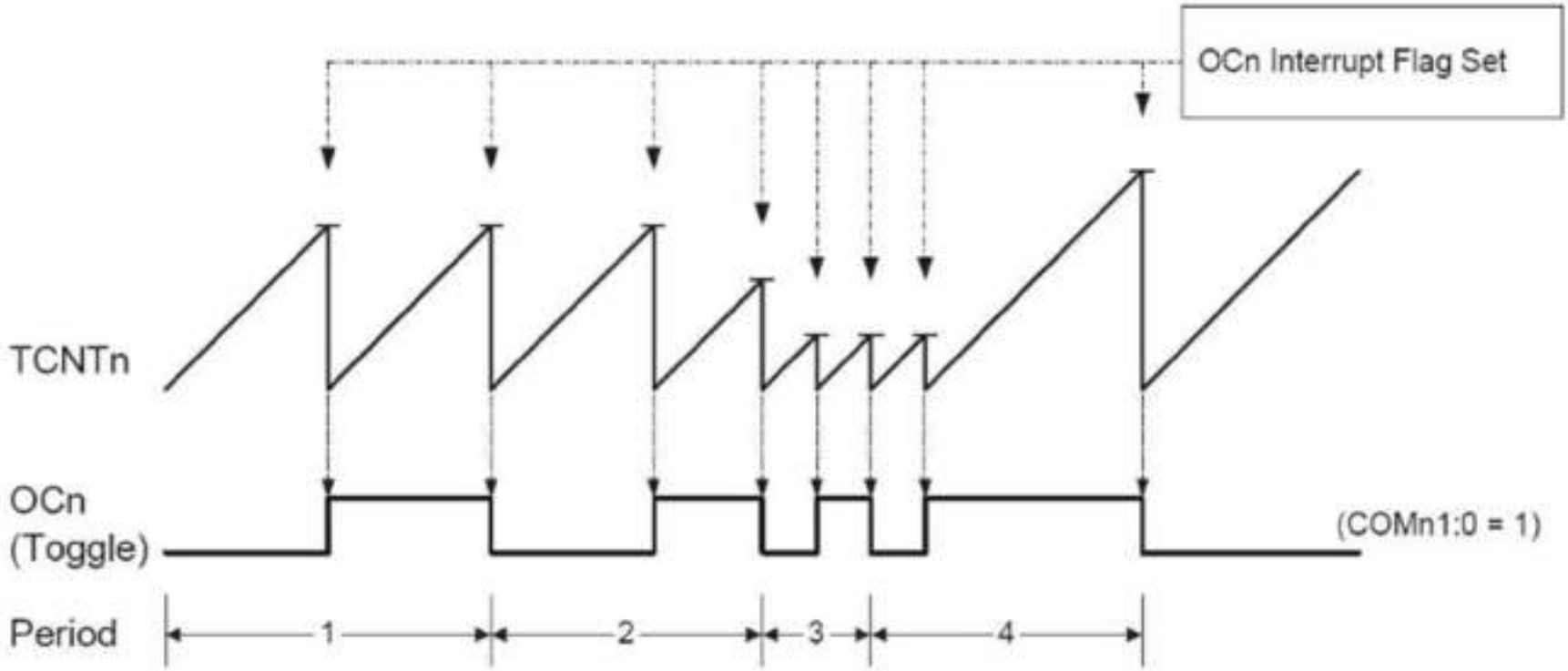
- CTC(Clear Timer on Compare match) Mode
  - 카운트의 한계값(최대로 세는 수)을 설정
  - 카운터는 업 카운터로서만 동작
    - 0으로부터 설정된 한계값까지 세고 다시 0으로 클리어
    - TCNT 값이 증가하여, OCR값과 일치하면 출력 비교 인터럽트 발생
    - OCRn의 값을 바꾸면 그 다음 카운터 주기를 원하는 대로 변경 가능
  - OCn단자를 이용하여 출력파형 발생 가능
    - TCCRn의 COMn1~n0을 01로 설정
    - OCRn레지스터 값을 바꿔가면서 출력비교에 의해 OCn의 신호를 토글
    - 출력되는 파형의 주기는 foc = fclk / (2*N*(1+OCR0))로 계산
  - TCCRn레지스터의 WGMn1:n0를 “10” 으로 설정

중요. 타이머와 카운터

- CTC(Clear Timer on Compare match) Mode

중요. 타이머와 카운터

□ CTC(Clear Timer on Compare match) Mode

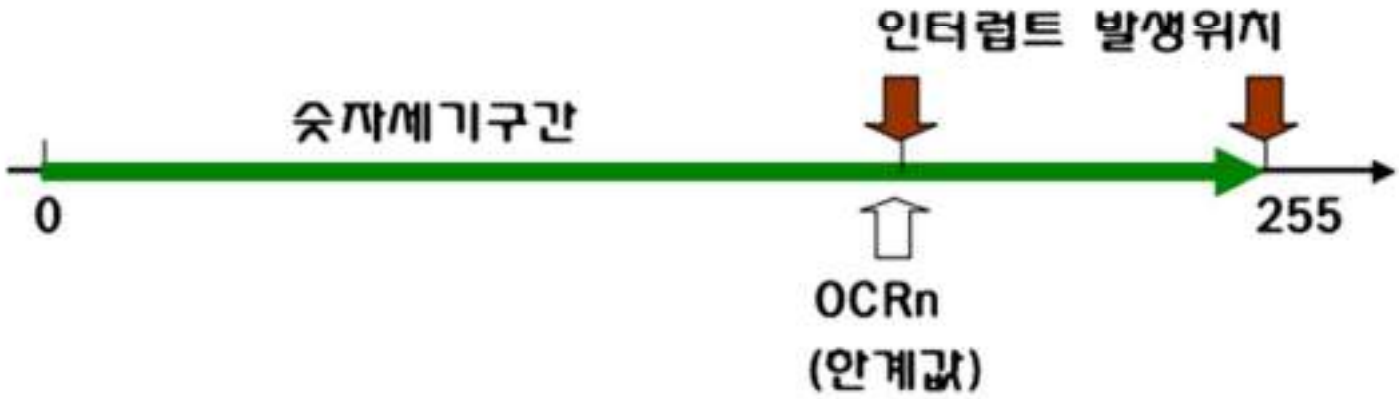


중요. 타이머와 카운터

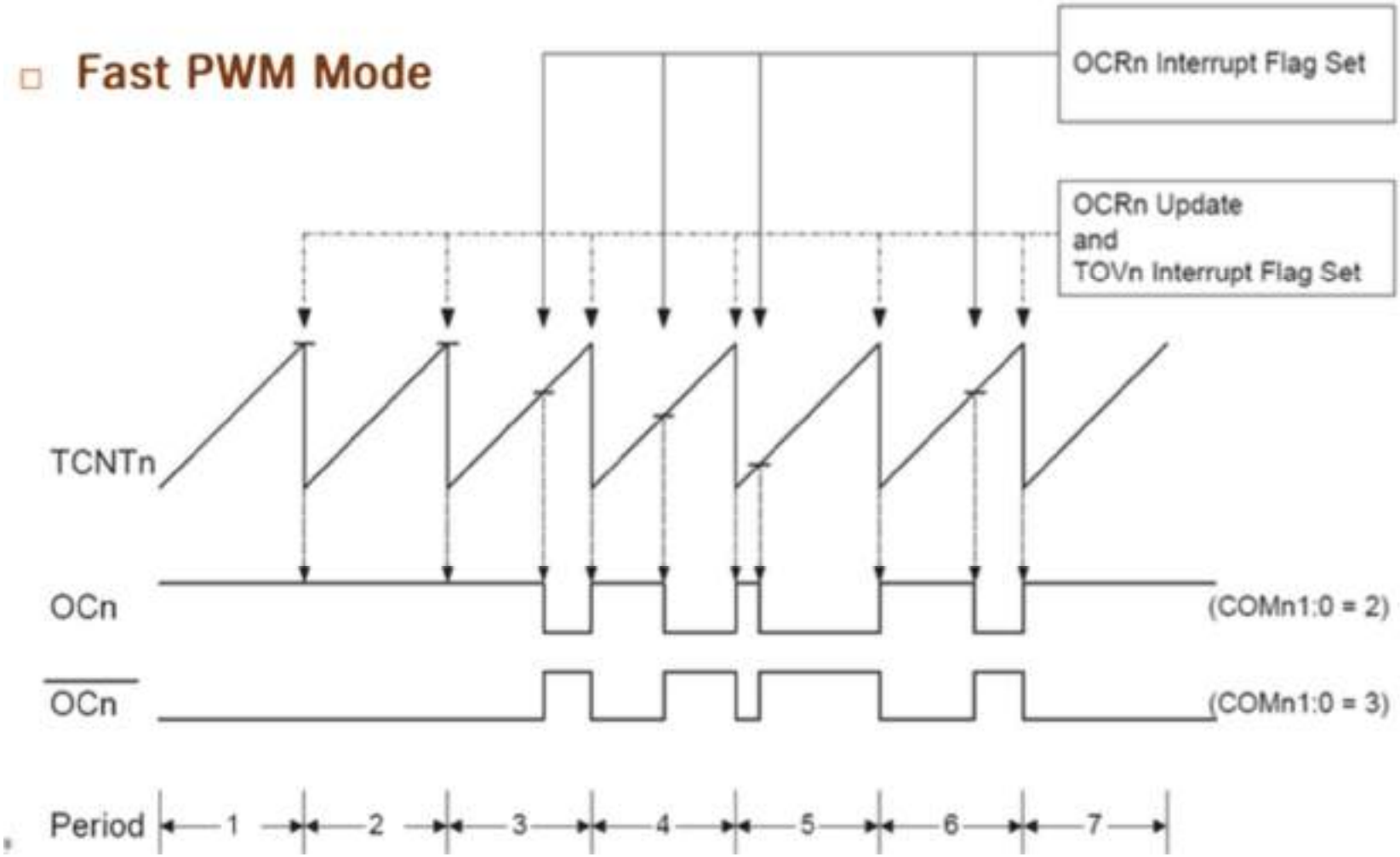
- **Fast PWM Mode**
  - 0에서 255까지 세는동안 두번의 인터럽트 발생 가능
  - 카운터는 업 카운터로서만 동작
    - TCNT 값이 증가하여, OCR값과 일치하면 출력 비교 인터럽트 발생
    - TCNTn는 업카운팅을 계속하여 255까지 증가했다가 0으로 바뀌는 순간 오버플로우 인터럽트 발생
    - OCRn의 값을 바꾸면 그 다음 카운터 주기를 원하는 대로 변경 가능
  - 두가지 모드로 OC0핀에 구형파 출력 가능
    - 비반전 비교 출력 모드
      - TCCRn 레지스터의 COM 비트를 "10" 로 설정
      - TCNT0가 OCR0와 일치하면, OC0 핀에 0를 출력하고 TCNT0가 00이 되면 OC0 핀에 1을 출력
    - 반전 비교 출력 모드
      - TCCRn 레지스터의 COM 비트를 "11" 로 설정
      - TCNT0가 OCR0와 일치하면, OC0 핀에 1을 출력하고, TCNT0가 00이 되면 OC0 핀에 0을 출력
    - 출력 파형 주파수 foc = fclk / (N*256)

중요. 타이머와 카운터

- Fast PWM Mode
  - TCCRn레지스터의 WGMn1:n0를 11 으로 설정
  - 높은 주파수의 PWM 파형 발생시 유용



중요. 타이머와 카운터


Fast PWM Mode
OCRn Interrupt Flag Set
OCRn Update
and
TOVn Interrupt Flag Set
TCNTn
OCn
(COMn1:0 = 2)
OCn
(COMn1:0 = 3)
Period
1
2
3
4
5
6
7

중요. 타이머와 카운터

- PCPWM(Phase Correct Pulse Width Modulation) Mode
  - Fast PWM과 유사
  - 업카운팅과 다운카운팅이 번갈아 일어남
    - TCNT 값이 증가하여, OCR값과 일치하면 출력 비교 인터럽트 발생
    - TCNTn는 업카운팅을 계속하여 TCNTn는 255에 도달하면 다운카운팅 시작.
    - 다운카운팅을 하다가 0에 도달하면 오버플로우 인터럽트가 발생
    - OCRn의 값을 바꾸면 그 다음 카운터 주기를 원하는 대로 변경 가능
  - PWM 주기를 변경하기 위해 OCRn 레지스터에 새로운 값을 기록하더라도 즉시 변경되지 않고 TCNTn이 255에 도달하면 갱신됨.

중요. 타이머와 카운터

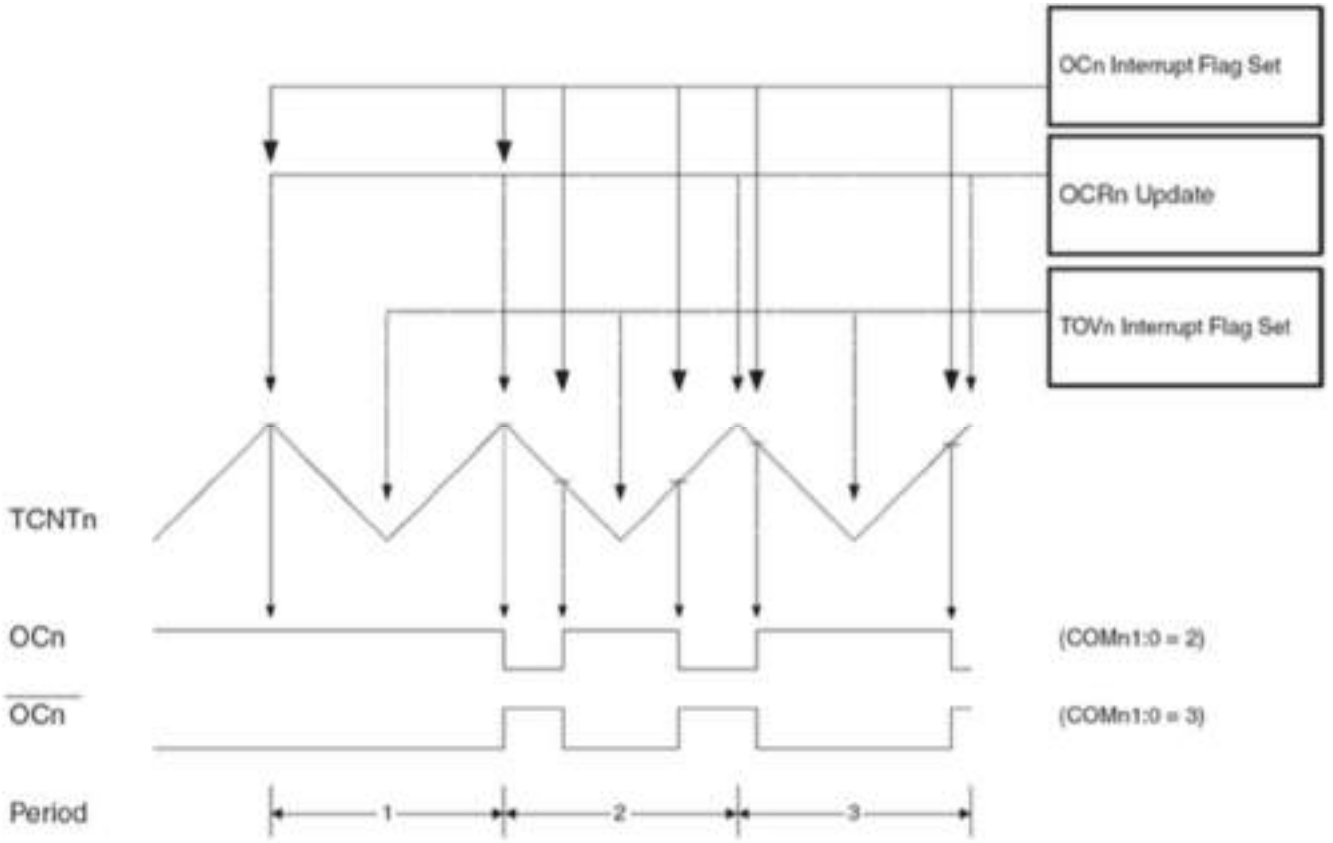
- PCPWM(Phase Correct Pulse Width Modulation) Mode
  - 두가지 모드로 OC0핀에 구형파 출력 가능
    - 비반전 비교 출력 모드(TCCRn 레지스터의 COM 비트를 “10” 로 설정)
      - 업 카운트 중에 TCNT0와 OCR0가 일치하면, OC0핀에 0를 출력
      - 다운 카운트 중에 일치하면, OC0핀에 1을 출력
    - 반전 비교 출력 모드(TCCRn 레지스터의 COM 비트를 “11” 로 설정)
      - 업 카운트 중에 TCNT0와 OCR0가 일치하면, OC0핀에 1을 출력
      - 다운 카운트 중에 일치하면, OC0핀에 0를 출력
    - 출력 파형 주파수 foc = fclk / (N*256)

중요. 타이머와 카운터

- PCPWM(Phase Correct Pulse Width Modulation) Mode
  - TCCRn레지스터의 WGMn1:n0를 01으로 설정
  - 높은 분해능의 PWM 파형 발생시 유용

중요. 타이머와 카운터

PCPWM(Phase Correct Pulse Width Modulation) Mode



중요. 타이머와 카운터

TCCRn(Timer/Counter Control Register n)

- 타이머/카운터 제어 레지스터 n(n=0 or 2)
- 동작 모드, 프리스케일러등 타이머/카운터의 전반적인 동작 영태를 결정

7	6	5	4	3	2	1	0
FOCn	WGMn0	COMn1	COMn0	WGMn1	CSn2	CSn1	CSn0

- 비트7 : FOCn
- 비트 6,3 : WGM
- 비트 5,4 : COM
- 비트 2, 1, 0 : CSn

중요. 타이머와 카운터

TCCRn(Timer/Counter Control Register n)

FOC(Force Output Compare)

- FoC가 1로 세트되면 출력 비교 실시 (OC핀 작동)
- FoC가 0로 세팅되면 출력 비교를 실시하지 않음.

WGM(Waveform Generation Mode)

- 타이머/카운터의 동작모드 설정
- 카운터의 카운팅 방향, 최대 카운터 값, Waveform Generation 방식 등을 결정

모드	WGMn1 (CTCn)	WGMn0 (PWMn)	타이머/카운터 동작모드	TOP	OCRn의 업데이트	TOVn Flag Set-on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCRn	Immediate	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

중요. 타이머와 카운터

TCCRn(Timer/Counter Control Register n)

COM(Compare Output Mode)

- OCn핀의 동작을 조정
- COMn1/COMn0에 따른 Ocn 핀의 동작
  - Non-PWM 모드

COMn1	COMn0	내용
0	0	normal 포트 동작, OCn 연결을 끊다.
0	1	Toggle OCn on compare match
1	0	Clear OCn on compare match
1	1	Set OCn on compare match

중요. 타이머와 카운터

TCCRn(Timer/Counter Control Register n)

- Fast-PWM 모드

COMn1	COMn0	내용
0	0	normal 포트 동작, OCn 연결을 끊다.
0	1	예약
1	0	Clear OCn on compare match, set OCn at TOP
1	1	Set OCn on compare match, clear OCn at TOP

- PC PWM Mode

COMn1	COMn0	내용
0	0	normal 포트 동작, OCn 연결을 끊다.
0	1	예약
1	0	up카운팅일때 Clear OCn on compare match, down카운팅일때 set Ocn
1	1	up카운팅일때 Set OCn on compare match, down카운팅일때 clear Ocn

중요. 타이머와 카운터

- TCCRn(Timer/Counter Control Register n)
  - CSn : 클럭 선택(Clock Select)
    - 타이머/카운터에 사용할 클럭과 프리스케일러를 선택
    - 타이머/카운터 0

CS02	CS01	CS00	설명
0	0	0	클럭소스가 없다.
0	0	1	No 프리스케일러
0	1	0	8분주
0	1	1	32분주
1	0	0	64분주
1	0	1	128분주
1	1	0	256분주
1	1	1	1024분주

중요. 타이머와 카운터

TCNTn(Timer/Counter Register n)

- 타이머/카운터 레지스터 n (n은 0 or 2)
  - 타이머/카운터 n의 8비트 카운터 값을 저장하고 있는 레지스터
  - 이 레지스터는 쓸 수도 있고 읽을 수도 있다.
  - 임의의 값을 써주면 타이머의 주기를 더 빠르게 할 수 있다.
  - 레지스터의 값은 인터럽트가 걸리면 자동으로 0으로 클리어 되고 다시 분주된 주기마다 한 개씩 값이 늘어난다.

7	6	5	4	3	2	1	0
TCNT7	TCNT6	TCNT5	TCNT4	TCNT3	TCNT2	TCNT1	TCNT0

중요. 타이머와 카운터

- TIMSK(Timer Interrupt Mask)
  - 타이머 인터럽트 마스크 레지스터
  - 타이머/카운터0, 타이머/카운터1, 타이머/카운터2가 발생하는 인터럽트를 개별적으로 enable하는 레지스터

7	6	5	4	3	2	1	0
OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0

  - 비트 1,7 : OCIE0/OCIE2
    - 타이머/카운터0/2의 출력비교 인터럽트 인에이블
  - 비트 0,6 : TOIE0/TOIE2
    - 타이머/카운터0/2 오버플로우 인터럽트 인에이블

중요. 타이머와 카운터

- TIFR (Timer Interrupt Flag Register)
  - 타이머 인터럽트 프래그 레지스터
  - 타이머/카운터0, 타이머/카운터1, 타이머/카운터2가 발생하는 인터럽트의 플래그를 저장하는 레지스터

7	6	5	4	3	2	1	0
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

  - 비트 1,7 : OCF0/OCF2
    - TCNT0/2레지스터와 출력비교 레지스터 OCR0/2값을 비교해서, 같으면 이 비트가 "1" 로 세트되어 출력 비교인터럽트가 발생.
  - 비트 0,6 : TOV0/TOV2
    - 타이머/카운터0/2에서 오버플로우가 발생하면 이 비트가 "1" 로 세트되어 오버플로우 인터럽트가 발생.

중요. 타이머와 카운터

ASSR(ASynchronous Status Register)

- 비동기 상태 레지스터
- 타이머/카운터00이 외부 클럭에 의하여 비동기 모드로 동작하는 경우 관련된 기능을 수행하는 레지스터

7	6	5	4	3	2	1	0
–	–	–	–	AS0	TCN0UB	OCR0UB	TCR0UB

- 비트 3 (AS0 : ASynchronous timer 0)
- 비트 2 (TCN0UB : Timer/CouNter0 Update Busy)
- 비트 1 (OCR0UB : Output Compare Register 0 Update Busy)
- 비트 0 (TCR0UB : Timer Conter Register 0 Update Busy)

중요. 타이머와 카운터

SFIOR(Special Function I/O Register)

- 특수 기능 I/O 레지스터
- 타이머/카운터들을 동기와 하는데 관련된 기능을 담당하는 레지스터

7	6	5	4	3	2	1	0
TSM	–	–	–	ACME	PUD	PSR0	PSR321

- **비트 7 (TSM : Timer Synchronization Mode)**
  - '0' : PSR0, PSR321비트를 하드웨어적으로 클리어,
  - 이로 인해, 타이머들이 동시에 카운팅 동작을 시작한다
- **비트 1 (PSR0 : Prescaler Reset Timer 0)**
  - '1' : 타이머0의 프리스케일러를 리셋
- **비트 0 (PSR321 : Prescaler Reset Timer 321)**
  - '1' : 타이머1/2/3의 프리스케일러를 리셋

6.1 LED 제어

6.1.7. ATmega128 내부의 Timer/Counter 이해하기

ATmega128 내부의 **Timer/Counter는 0, 2(8 bit) / 1, 3(16 bit)** 등 4개가 존재한다.
Counter 가능 범위 : 0 ~ 255 (8 bit)와 0 ~ 65535 (16 bit)이고, PWM 및 비동기 동작 모드를 가지는 8 bit , 16bit 업/다운 Counter이다.

ATmega128의 기본 Timer는 16MHz이고 Timer/Counter로 사용할 때는 분주하여 사용해야 한다.
예) 분주비가 ½ 이면 16/2 = 8 MHz로 Timer/Counter의 timer가 시작된다.
즉, 분주비가 ¼이면 16/4 = 4 MHz, 분주비가 1/64이면 16/64=0.25MHz이다.

오버플로우 인터럽트(Overflow interrupt) : TCNT 값이 증가하다가 최댓값 (255)에서 하나 더 증가하여 00이 되었을 때 발생하는 것이 오버플로우 인터럽트라 한다.
Output Compare Match (출력 비교 일치) 인터럽트
→ TCNT 값이 증가하다가 출력 비교 레지스터 (OCR) 값과 일치하면 발생하는 인터럽트

Overflow (오버플로우) 흘러넘치다, 저장할 수 있는 값을 넘어섰을 때를 뜻함
8bit일 때, 최댓값은 0b11111111 = 255이고, 256을 넣으면 0b00000000 = 0 이 되는 현상

6.1 LED 제어

6.1.8. Timer/Counter0(8비트) overflow 계산방법

TCNT0(Timer 0 레지스터)의 값이 255에서 256이 되면 0으로 바뀌면서 오버플로우 인터럽트 발생

만일 0이 아닌 246에서 시작하면 256까지 10번만 세면 인터럽트가 발생한다.

클럭 하나의 시간이 1us라면 10번 세면 10us되는데 따라서 10번을 세야 한다면 **256 - 10 = 246** 의 값을 TCNT0에 넣어주면 된다.

오버플로우 인터럽트가 발생하면 TCNT0의 값을 변경해줘야 함 (0으로 바뀌면서 인터럽트 발생)

	비트							
	7	6	5	4	3	2	1	0
246	1	1	1	1	0	1	1	0
247	1	1	1	1	0	1	1	1
...	...	...	...	...	...	...	...	...
255	1	1	1	1	1	1	1	1
256 = 0	0	0	0	0	0	0	0	0

Overflow Interrupt

KYUNG HEE UNIVERSITY

6.1 LED 제어

6.1.9. ATmega128에서 Timer/Counter 0을 이용한 1ms 인터럽트 만들기

1.MCU(ATmega128_의 기본 클럭 : 16MHz, 인터럽트 주기는 1ms 로 설정한다.

2. 분주비 (Prescaler) 를 이용한 Timer0의 클럭 설정 : 16MHz/분주비

   → 16MHz(1), 2MHz(8), 500kHz(32), **250kHz(64), 125kHz(128),** 62.5kHz(256), 15.625kHz(1024)

4. 클럭의 주기 : 1/클럭 62.5ns(1), 500ns(8), 2us(32), **4us(64), 8us(128),** 16us(256), 64us(1024)

5. 오버플로우가 발생하려면 최대 256개의 클럭이 필요 (8 bit, 256번 : 0 → 1 → ... → 255 → 0)

6. 각 주기의 최대 인터럽트 시간 : 주기*256

   → 16us(1), 128us(8), 512us(32), **1.024ms(64), 2.048ms(128),** 4.096ms(256), 16.384ms(1024)

7. Timer/Counter 0에서는 분주비 1, 8, 32는 불가능하고 64,128,256,1024 사용 가능함.

8. 각 클럭의 주기에 따라 카운터하는 클럭의 개수 계산 : 1ms/주기

   → **250번(64), 125번(128),** 62.5번(256), 31.25번(1024)

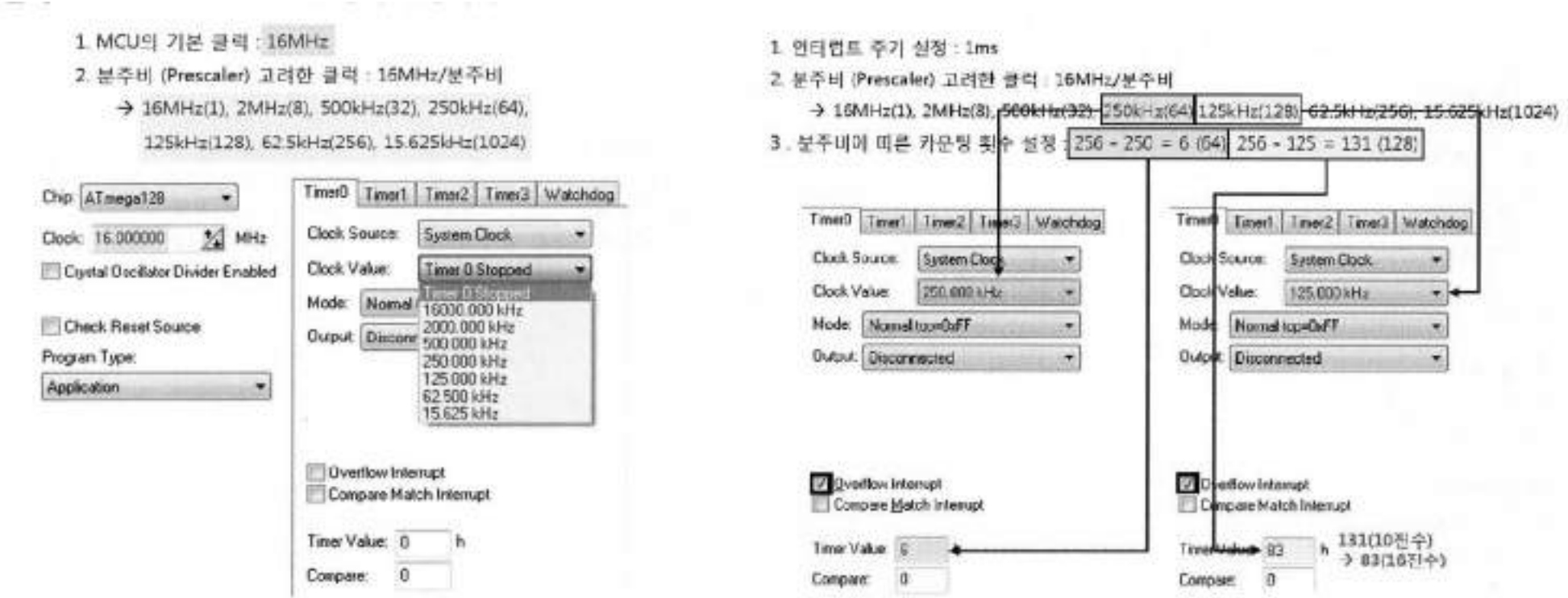
9. 분주비 256, 1024는 정확하지 않으므로 제외, 분주비 64, 128 중 선정

10. 분주비에 따른 카운터 횟수 설정 : 256 - 250 = 6 (64), 256 - 125 = 131 (128)

6.1 LED 제어

[실습14] ATmega128의 Timer/Counter0를 이용하여 LED를 1초 간격으로 ON-OFF하는 프로그램을 작성.

Code Wizard를 실행하여**MCU** 메인클럭이 **16MHz**이고 **Timer/Counter 0**의 오버플로우 인터럽트가 **1ms**마다 발생하게 설정을 한다.



KYUNG HEE UNIVERSITY

6.1 LED 제어

[실습14] ATmega128의 Timer/Counter0를 이용하여 LED를 1초 간격으로 ON-OFF하는 프로그램을 작성.

설정을 마친 후 **generate code**를 클릭 하면 옆에 **Program Preview**에 코드가 생성됨, 이 코드 중 필요한 부분(생성한 부분)을 복사하여 사용하면 된다.

```
// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Reinitialize Timer 0 value
TCNT0=0x06;
// Place your code here

}
```

타이머 인터럽트 함수

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 250.000 kHz
// Mode: Normal top=0xFF
// OC0 output: Disconnected
// Timer Period: 1 ms
ASSR=0<<AS0;
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (1<<CS02) | (0<<CS01) | (0<<CS00);
TCNT0=0x06;
OCR0=0x00;
```

Timer/Counter 0 레지스터를 초기화

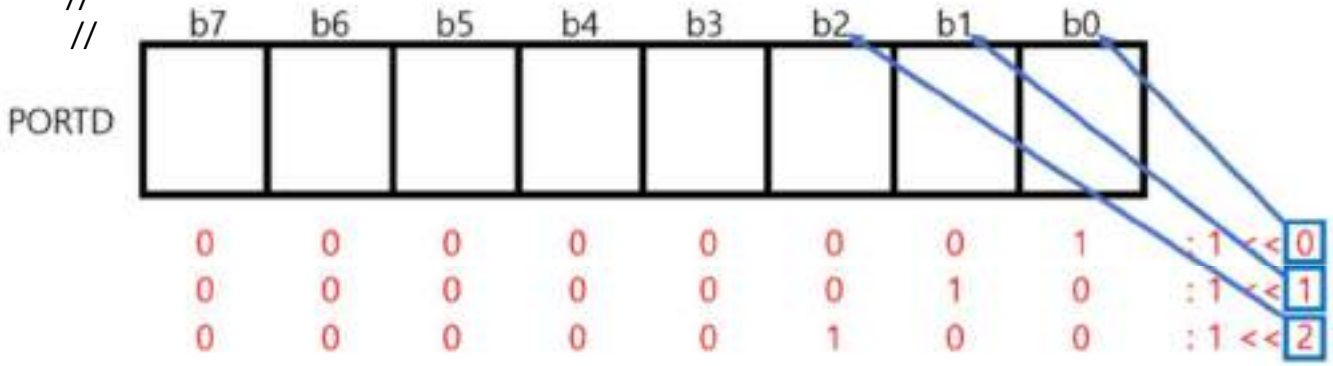
```
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (1<<TOIE0);
ETIMSK=(0<<TICIE3) | (0<<OCIE3A) | (0<<OCIE3B) | (0<<TOIE3) | (0<<OCIE3C) | (0<<OCIE1C);
```

```
// Globally enable interrupts
#asm("sei")
```

타이머 인터럽트 초기화 및 어셈블리어로 인터럽트 허용 명령

```
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (1<<TOIE0);
ETIMSK=(0<<TICIE3) | (0<<OCIE3A) | (0<<OCIE3B) | (0<<TOIE3) | (0<<OCIE3C) | (0<<OCIE1C);
```

```
/* ***** TIMER_COUNTER_1 ************** */
/* TIMSK - Timer/Counter Interrupt Mask Register */
#define    TOIE1          2       // Timer/Counter1 Overflow Interrupt Enable
#define    OCIE1B         3       // Timer/Counter1 Output CompareB Match Interrupt Enable
#define    OCIE1A         4       // Timer/Counter1 Output CompareA Match Interrupt Enable
#define    TICIE1         5       // Timer/Counter1 Input Capture Interrupt Enable
#define    TOIE2          6       //
#define    OCIE2          7       //
```



쉬프트 연산을 수행할 숫자는 비트의 번호와도 대응이 된다.

즉, 2번비트를 1로 만들고 싶으면 2만큼 쉬프트 연산을 수행해주면 되는것이다.

그러므로 코드를 짤 때, 1<<3 이라는것은 3번비트가 1인 값이라고도 해석 할 수 있어야 한다.

6.1 LED 제어

```
#include <mega128.h>

bit one_sec_flag;
int cnt;

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Reinitialize Timer 0 value
        TCNT0=0x06;
// Place your code here
        cnt++;
        if(cnt >=1000)
        {
                cnt = 0;
                one_sec_flag = 1;
         }
}

void main(void)
{
// Port A initialization
DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) | (1<<DDA1) | (1<<DDA0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);
```

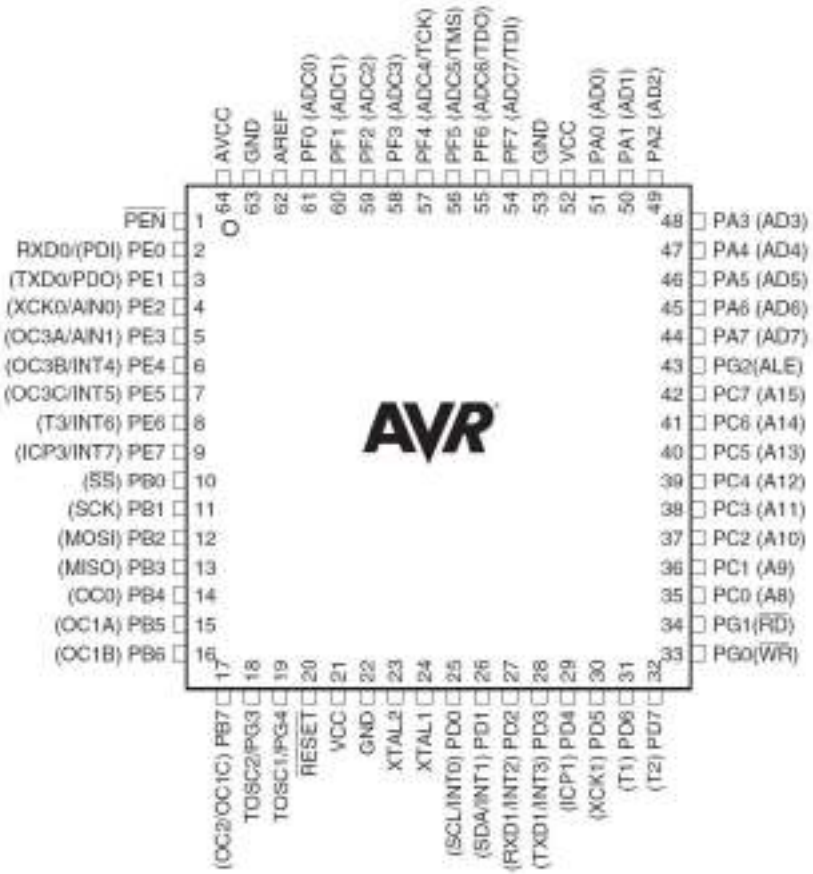
6.1 LED 제어

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 250.000 kHz
// Mode: Normal top=0xFF
// OC0 output: Disconnected
// Timer Period: 1 ms
ASSR=0<<AS0;
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (1<<CS02) | (0<<CS01) | (0<<CS00);

TCNT0=0x06;
OCR0=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (1<<TOIE0);
ETIMSK=(0<<TICIE3) | (0<<OCIE3A) | (0<<OCIE3B) | (0<<TOIE3) | (0<<OCIE3C) | (0<<OCIE1C);

// Globally enable interrupts
#asm("sei")

while (1)
{
    // Place your code here
      if(one_sec_flag ==1)
      {
            one_sec_flag = 0;
            PORTA = PORTA^0xFF;
      }
}
}
```

6.1 LED 제어

[실습15] ATmega128의 Timer/Counter0의 Fast PWM을 이용하여 LED의 밝기 조절하는 프로그램을 작성.

- **Code Wizard**를 실행하여 **MCU** 메인클럭이 **16MHz**이고 **Timer/Counter 0**의 **Fast PWM** 설정을 한다.

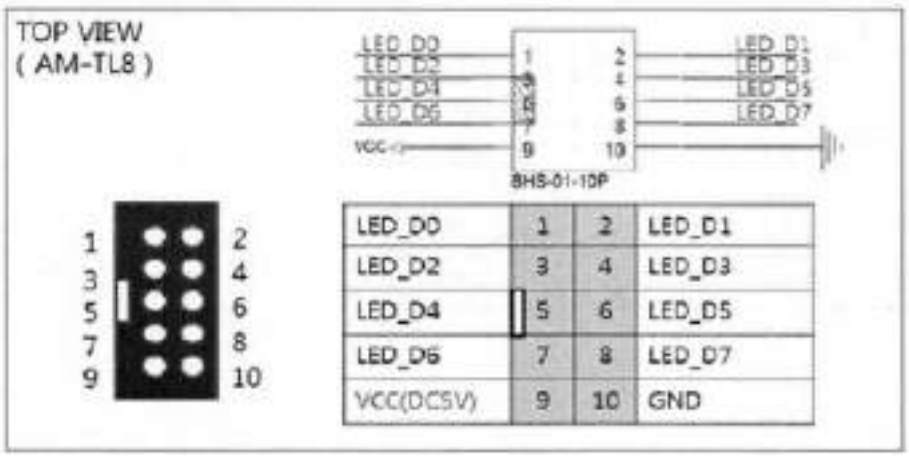


KYUNG HEE UNIVERSITY

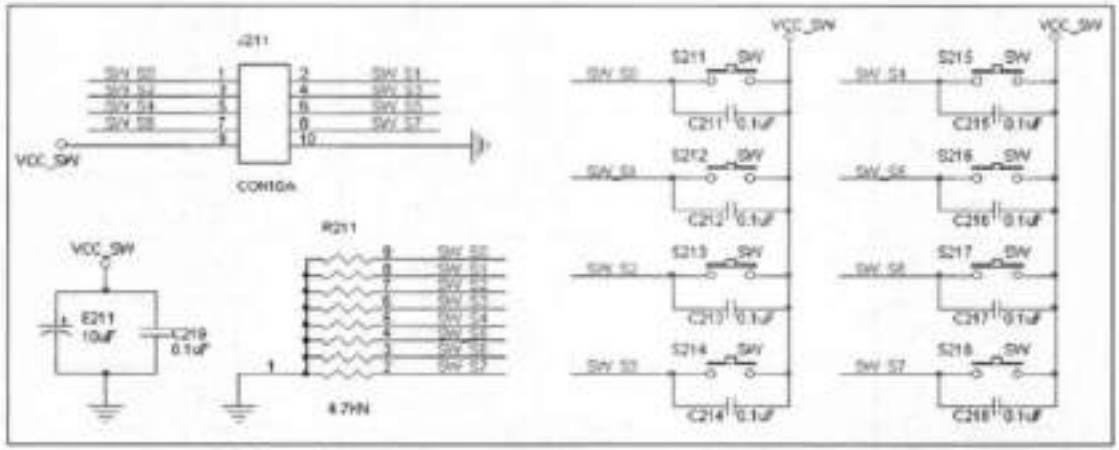
6.1 LED 제어

[실습15] ATmega128의 Timer/Counter0의 Fast PWM 을 이용하여 LED의 밝기 조절하는 프로그램을 작성.

- 밝기 조절 스위치는 **PORTA**에 연결
- **LED**는 **PORTB**에 연결 → 왜 일까요?



LED_D0	1	2	LED_D1
LED_D2	3	4	LED_D3
LED_D4	5	6	LED_D5
LED_D6	7	8	LED_D7
VCC(DC5V)	9	10	GND

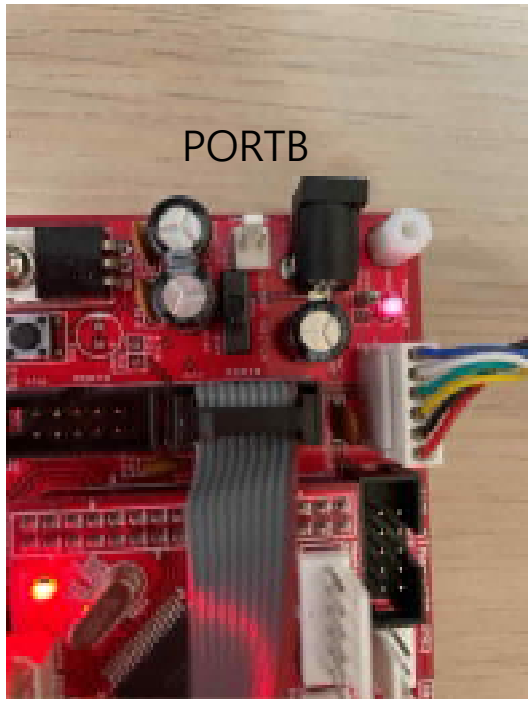


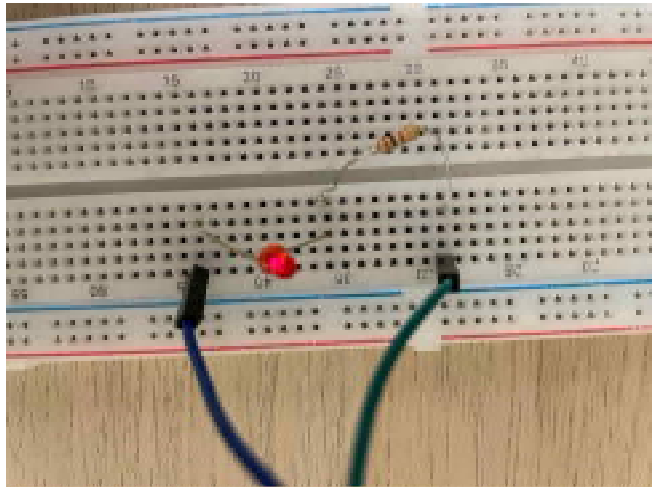
#1	#3	#5	#7	E9
S/W 1	S/W 3	S/W 5	S/W 8	전원 5V
#2	#4	#6	#8	#10
S/W 2	S/W 4	S/W 6	S/W 7	전원 GND

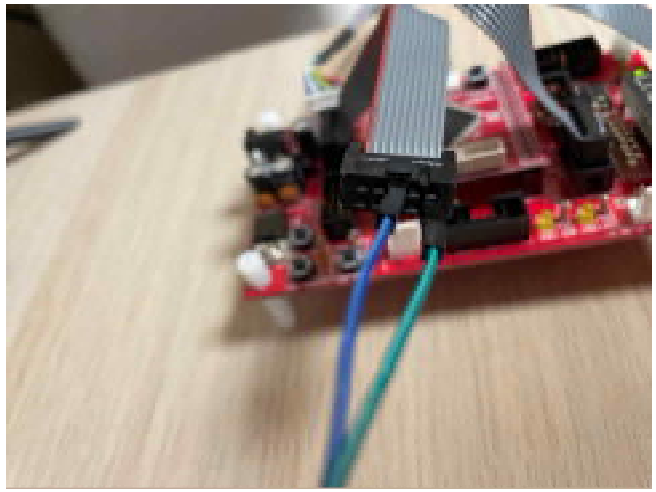
KYUNG HEE UNIVERSITY

6.1 LED 제어

[실습15] ATmega128의 Timer/Counter0의 Fast PWM 을 이용하여 LED의 밝기 조절하는 프로그램을 작성.





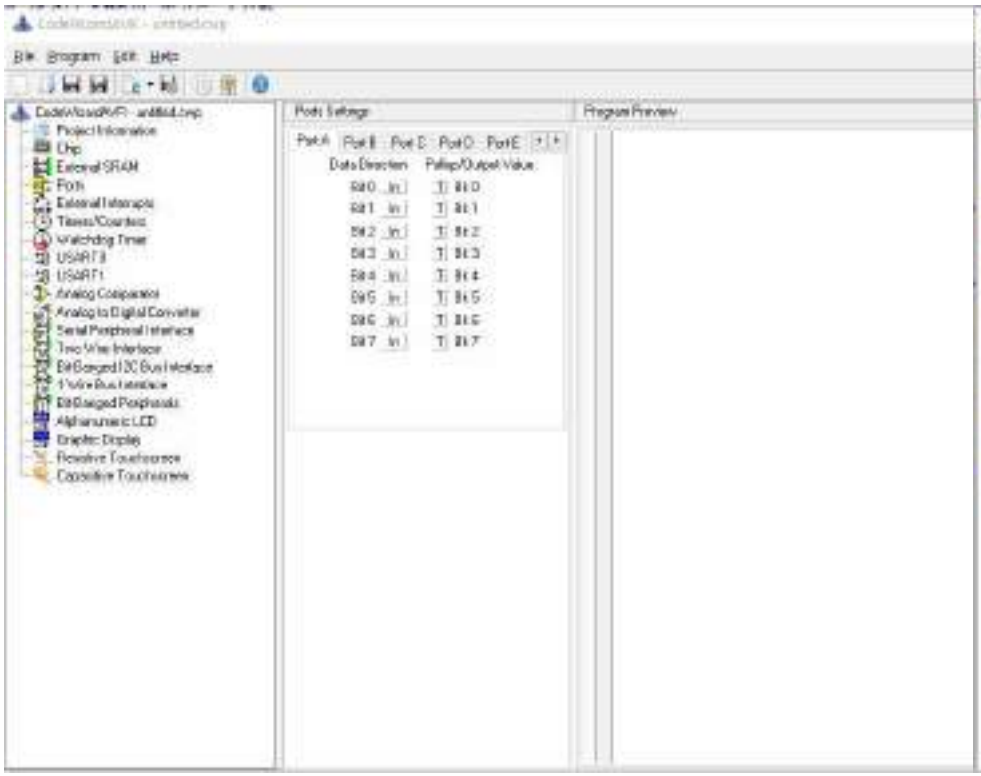


KYUNG HEE UNIVERSITY

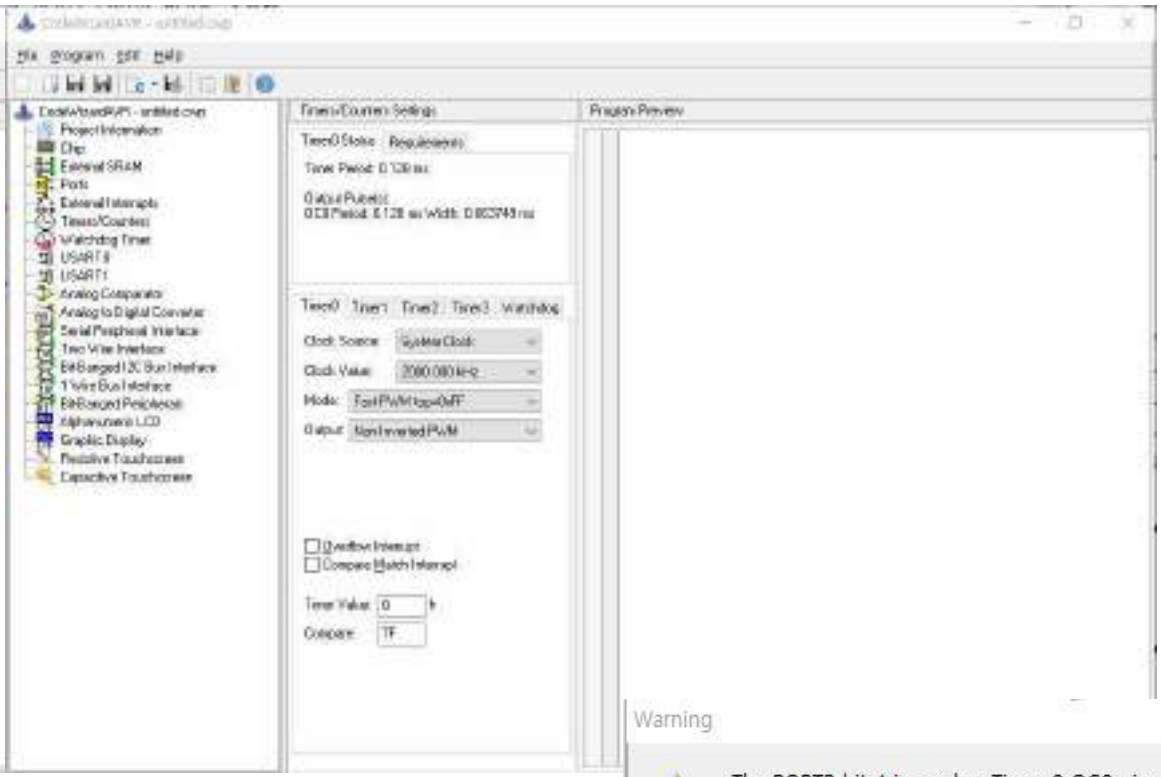
6.1 LED 제어

[실습15] ATmega128의 Timer/Counter0의 Fast PWM 을 이용하여 LED의 밝기 조절하는 프로그램을 작성.

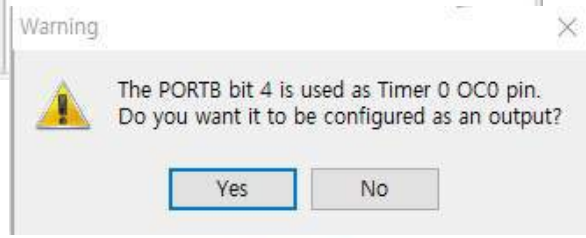
설정을 마친 후 **generate code**를 클릭 하면 옆에 **Program Preview**에 코드가 생성됨, 이 코드 중 필요한 부분(생성한 부분)을 복사하여 사용하면 된다.



PORTA 설정 – 스위치 연결



Timer/Counter 0 레지스터를 초기 설정



[실습15] ATmega128의 Timer/Counter0의 Fast PWM을 이용하여 LED의 밝기 조절하는 프로그램을 작성.

설정을 마친 후 **generate code**를 클릭 하면 옆에 **Program Preview**에 코드가 생성됨, 이 코드 중 필요한 부분(생성한 부분)을 복사하여 사용하면 된다.

```
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1) | (0<<DDA0);
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (1<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (0<<DDB0);
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 2000.000 kHz
// Mode: Fast PWM top=0xFF
// OC0 output: Non-Inverted PWM
// Timer Period: 0.128 ms
// Output Pulse(s):
// OC0 Period: 0.128 ms Width: 0.063749 ms
ASSR=0<<AS0;
TCCR0=(1<<WGM00) | (1<<COM01) | (0<<COM00) | (1<<WGM01) | (0<<CS02) | (1<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0=0x7F;
```

6.1 LED 제어

```
#include <mega128.h>

void main(void)
{
   DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1) | (0<<DDA0);
   PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

   DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (1<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (0<<DDB0);
   PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

   // Timer/Counter 0 initialization
   // Clock source: System Clock
   // Clock value: 2000.000 kHz
   // Mode: Fast PWM top=0xFF
   // OC0 output: Non-Inverted PWM
   // Timer Period: 0.128 ms
   // Output Pulse(s):
   // OC0 Period: 0.128 ms Width: 0.063749 ms
   ASSR=0<<AS0;
   TCCR0=(1<<WGM00) | (1<<COM01) | (0<<COM00) | (1<<WGM01) | (0<<CS02) | (1<<CS01) | (0<<CS00);
   TCNT0=0x00;
   OCR0=0x7F;
```

6.1 LED 제어

```
while (1)
  {
    if (PINA.0 == 1)
    {
      OCR0 = 0x00;
    }
    else if (PINA.1 == 1)
    {
      OCR0 = 0x0f;
    }
    else if (PINA.2 == 1)
    {
      OCR0 = 0x8f;
    }
    else if (PINA.3 == 1)
    {
      OCR0 = 0xff;
    }
    else OCR0 = 0x00;
  }
}
```