

# 임베디드 시스템을 위한 SW 구조 설계

(file #5 / 10) ver0.2

Yongseok Chi

## 1. Develop an understanding of technologies

about the micro controller & processor systems using evaluation kit

## 2. Skill up a **design ability the micro controller application systems**

(1) technology of **the hardware and software** components

(2) **debugging** technology about the micro controller

(3) understanding of a **circuit design** skill

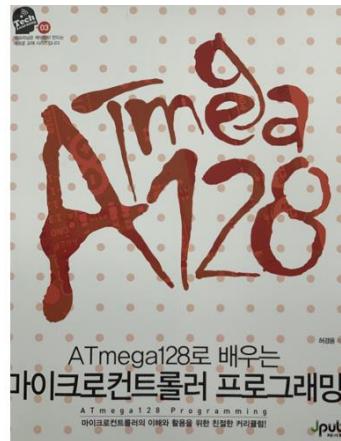
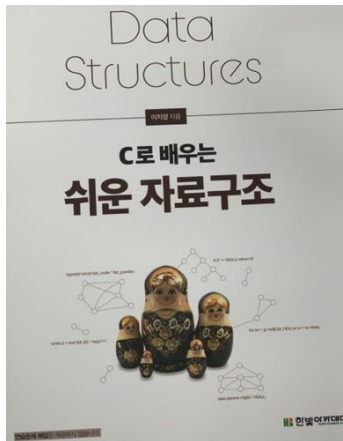
## 3. Develop an ability of design about the micro controller

## 1. Reference information

(1) Atmel datasheet, <http://atmel.com> (→ [microchip.com](http://microchip.com))

(2) ARM Architecture Reference Manual, <http://arm.com>

## 2. Books



## 3. 실험KIT (Evaluation board)

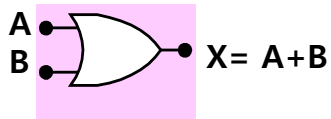
한백전자 IOT 실험 KIT



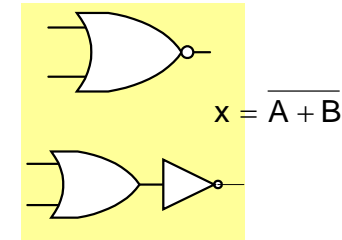
KUT-128\_Com 실험 키트

- 
1. Micro Processor 원리
  2. Atmel사의 8bit Micro-controller
  3. KUT0128 Evaluation Board 기능과 특징
  4. IO Port 제어
  5. External Interrupt 제어
  6. Timer / Counter 제어
  7. UART 제어
  8. AD Converter 제어
  9. Comparator 제어
  10. EEPROM 제어 (IIC, Parallel method)
  11. SPI 제어

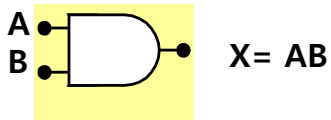
- Logic Gates : the basic elements of logic circuits

**OR**

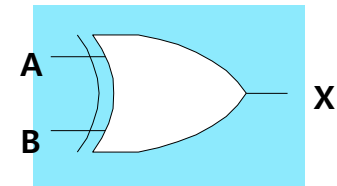
A	B	$x = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

**NOR**

A	B	$\overline{A+B}$	$A+B$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

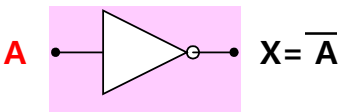
**AND**

A	B	$x = AB$
0	0	0
0	1	0
1	0	0
1	1	1

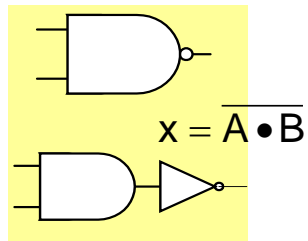
**EX-OR**

A	B	$x$
0	0	0
0	1	1
1	0	1
1	1	0

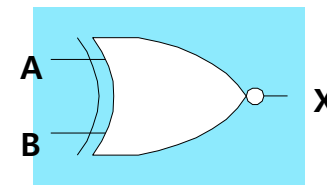
$$x = \overline{A}B + A\overline{B} = A \oplus B$$

**NOT**

A	$x = A'$
0	1
1	0

**NAND**

A	B	$\overline{AB}$	$AB$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

**EX-NOR**

A	B	$x$
0	0	1
0	1	0
1	0	0
1	1	1

$$x = AB + \overline{A}\overline{B} = \overline{A \oplus B}$$

## • Number

Bit

1000    0111    0110    0101    0011    0110    0011

**MSB/ LSB( Most/ Least Significant Bit )**

BCD(Binary-Coded-Decimal) Code

◆ 8	7	4	(Decimal)
1000	0111	0100	(BCD)

◆ Comparison of BCD and Binary

$137_{10} = 10001001_2$   
(Binary) - *require only 8 bits*

$137_{10} = 0001\ 0011\ 0111_{BCD}$   
(BCD) - *require 12 bits*

ASCII Code

◆ American Standard Code for Information Interchange(ASCII)

◆ 7 bit code,  $2^7 = 128$  possible codes

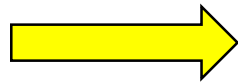
0	1	A	a	z	
30H	31H	41H	61H	7AH	28H

Hex	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

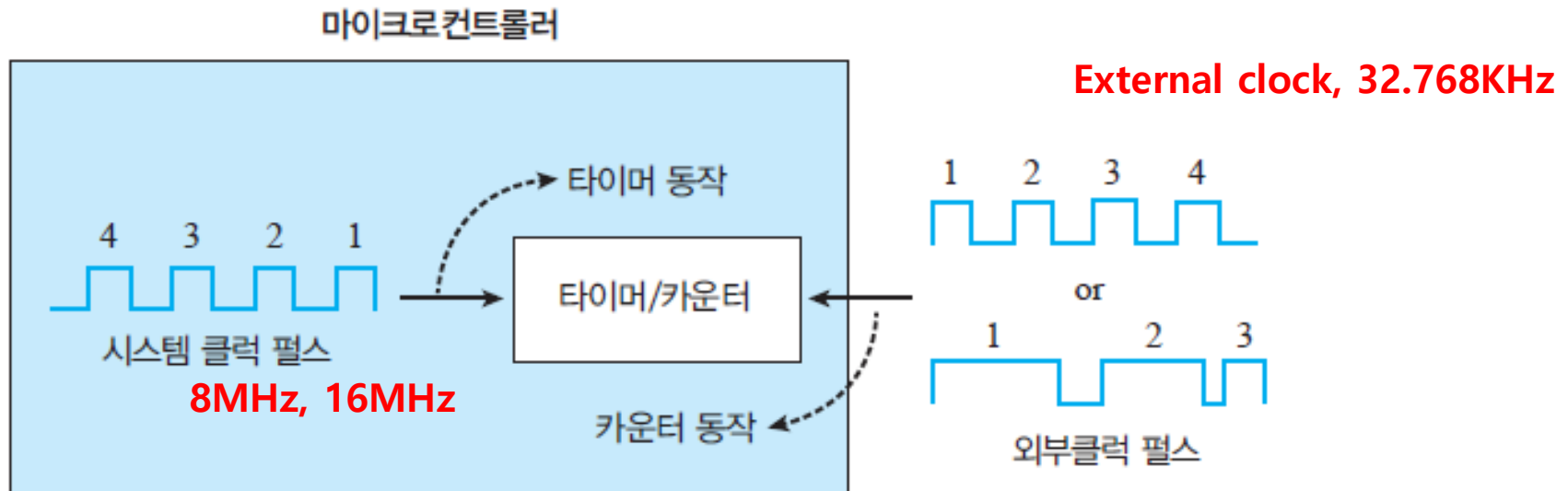
### Timer / Counter (숙제 : 교재 176page 읽기)

- 마이크로프로세서는 **타이머/카운터 기능으로 일정 시간 간격으로 디바이스 제어**
- **CPU 부담 없이**, 시간 경과를 알 필요가 있음
- **CPU 부담 없이**, 장치 제어를 위한 주기적인 펄스 출력이 필요함
- **CPU 부담 없이**, 외부에서 입력되는 펄스의 정확한 발생 시각을 알 필요가 있음

#include<delay.h>의 delay\_ms(ms), delay\_us(us)는



**CPU가 시간 지연 동작 외에는 다른 작업을 할 수 없음**



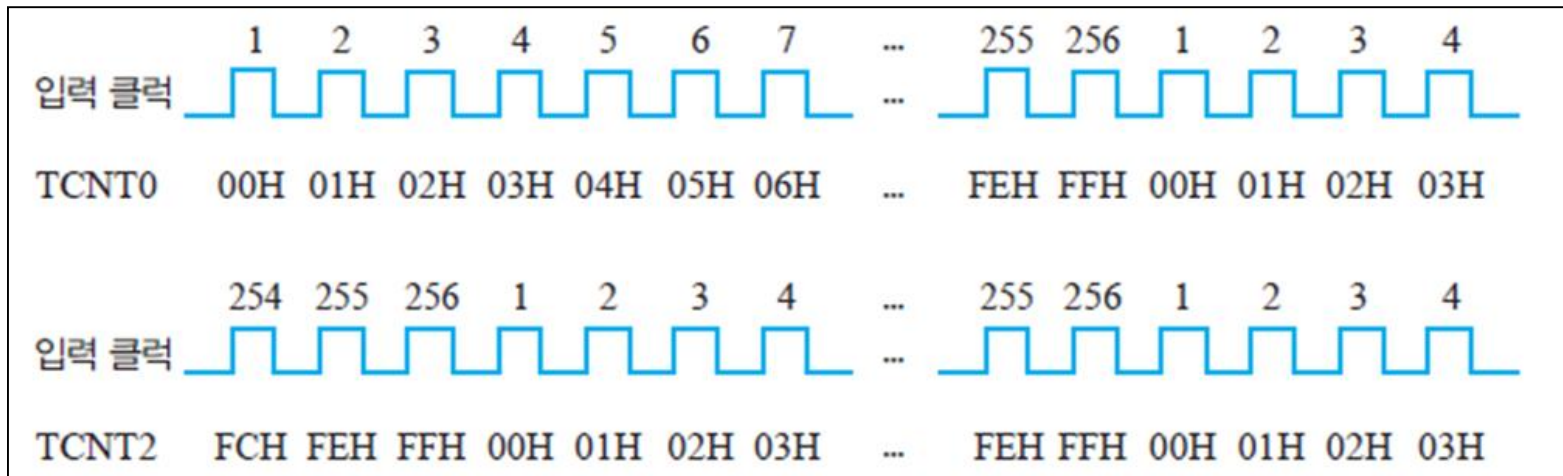
## 6. Timer / Counter

Strictly Private and Confidential

### Timer / Counter

- 마이크로프로세서는 카운트하는 입력 펄스 개수를 register TCNTn 에 기록
- Register TCNTn = 최대값이 8bits 이면, 8bits Timer / Counter 0,2
- Register TCNTn = 최대값이 16bits 이면, 16bits Timer / Counter 1,3

n : 0, 1, 2, 3



8bits Timer / Counter 0,2



### 8bits or 16bits Timer / Counter

- Timer / Counter 0, Timer / Counter 1 , Timer / Counter 2 , Timer / Counter 3

Timer / Counter	설명
Timer / Counter 0 Timer / Counter 2	<ul style="list-style-type: none"><li>• 8bits Timer / Counter</li><li>• 최대 <math>2^8</math> (=256, 00H~FFH)개의 입력 펄스를 카운트</li><li>• Counter 개수가 FF일 때 1개를 카운트하면, 00으로 반복해서 카운트</li></ul>
Timer / Counter 1 Timer / Counter 3	<ul style="list-style-type: none"><li>• 16bits Timer / Counter</li><li>• 최대 <math>2^{16}</math> (=65536, 0000H~FFFFH)개의 입력 펄스를 카운트</li><li>• Counter 개수가 FFFF일 때 1개를 카운트하면, 0000으로 반복해서 카운트</li></ul>

## 6. Timer / Counter

Strictly Private and Confidential

### 8bits or 16bits Timer / Counter



	타이머/카운터0	타이머/카운터2	타이머/카운터1	타이머/카운터3
구조	8비트	8비트	16비트	16비트
카운터 경우, 외부 clock 입력	TOSC1	T2	T1	T3
10bits Pre-scaler	1, 8, 32, 64, 128, 256, 1024	1, 8, 64, 256, 1024	1, 8, 64, 256, 1024	1, 8, 64, 256, 1024
Register	TCNT0, TCCR0, OCR0, ASSR, TIMSK, TIFR, SFIO	TCNT2, TCCR2, OCR2, TIMSK, TIFR, SFIO	TCNT1, TCCR1A, TCCR1B, TCCR1C, OCR1A, OCR1B, OCR1C, ICR1, TIMSK, ETIMSK, TIFR, ETIFR, SFIO	TCNT3, TCCR3A, TCCR3B, TCCR3C, OCR3A, OCR3B, OCR3C, ICR3, TIMSK, ETIMSK, TIFR, ETIFR, SFIO
Input pin	TOSC1, TOSC2	T2	T1, IC1	T3, IC3
Output pin	OC0	OC2	OC1A, OC1B, OC1C	OC3A, OC3B, OC3C
Interrupt	Overflow, 출력 비교 match	Overflow, 출력 비교 match	Overflow, 출력 비교 match A/B/C, 입력 capture	Overflow, 출력 비교 match A/B/C, 입력 capture
특징	RTC기능, 타이머/카운터 모두 Pre-scaler	—	Capture 기능	Capture 기능

### 8bits or 16bits Timer / Counter

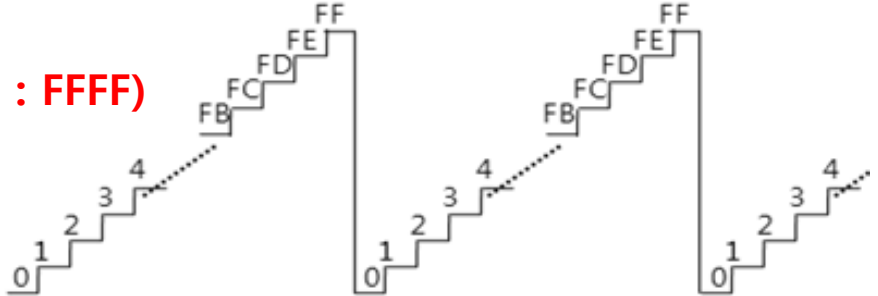
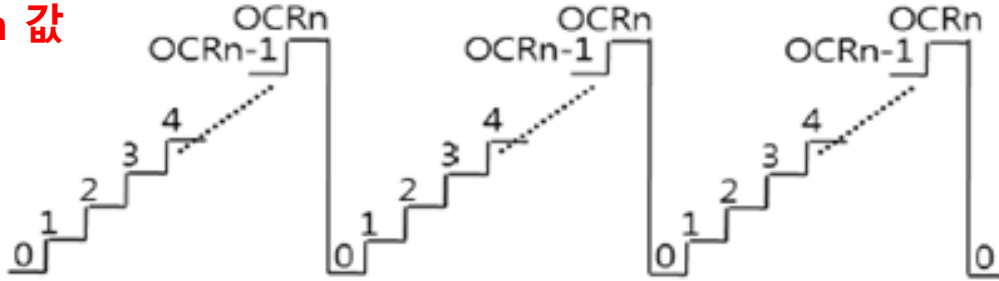
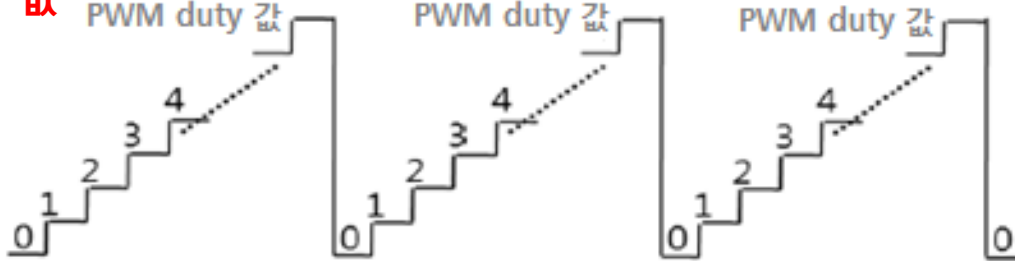


기본 동작	설명
Overflow	<ul style="list-style-type: none"><li>• 0부터 1씩 증가되어 8비트가 모두 1이 되는 0xFF에 도달된 후 Overflow</li><li>• 입력 펄스를 최대 개수까지 카운트하는 것을 반복하는 동작</li><li>• 8(16)비트 타이머/카운터는 카운트한 입력 clock이 FFH→00H (FFFFH→0000H)일 때 오버플로우 발생</li></ul>
Compare match	<ul style="list-style-type: none"><li>• 입력 펄스를 지정한 개수까지 카운트하는 것을 반복하는 동작</li><li>• 8(16)비트 타이머/카운터는 00H~FFH(0000H~FFFFH)내에서 비교 매치할 값을 지정</li></ul>
PWM	<ul style="list-style-type: none"><li>• PWM 파형을 ATmega128의 특정 핀으로 출력시킬 수 있음</li></ul>

# 6. Timer / Counter

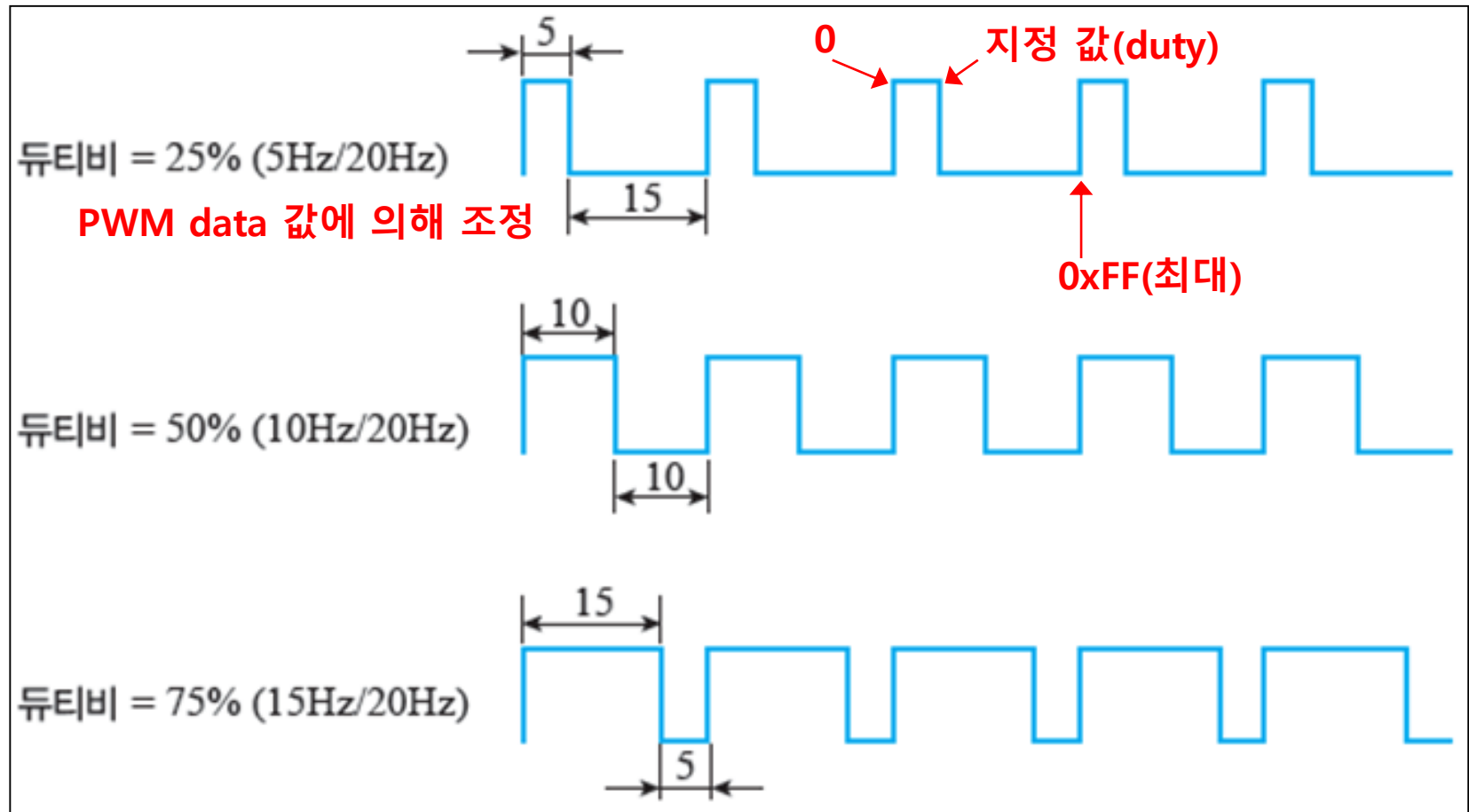
Strictly Private and Confidential

## 8bits or 16bits Timer / Counter

기본 동작	설명
<b>Overflow mode</b> : up counter : down counter	<b>최대값</b> (8bits : FF, 16bits : FFFF) 
<b>Compare match mode</b>	<b>Match 값</b> 
<b>PWM mode</b>	<b>PWM 값</b> 

### PWM (Pulse Width Modulation)

- 펄스 폭 변조
- 펄스 폭을 가변 시킴. PWM 파형의 duty ratio(듀티비)를 조절



## 6. Timer / Counter

Strictly Private and Confidential

### 8bits or 16bits Timer / Counter

- 카운트하는 입력 펄스 개수를 **TCNTn** 에 기록
- 8비트 타이머/카운터 경우에,

**TCNTn** =  $2^8(=255)$ 일 때 입력 펄스가 추가되면 **TCNTn** = 0 으로 기록

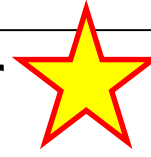


➡ **TCNT0, TCNT2 값**

## 6. Timer / Counter

Strictly Private and Confidential

8bits or 16bits Timer / Counter의 Pre-scaler



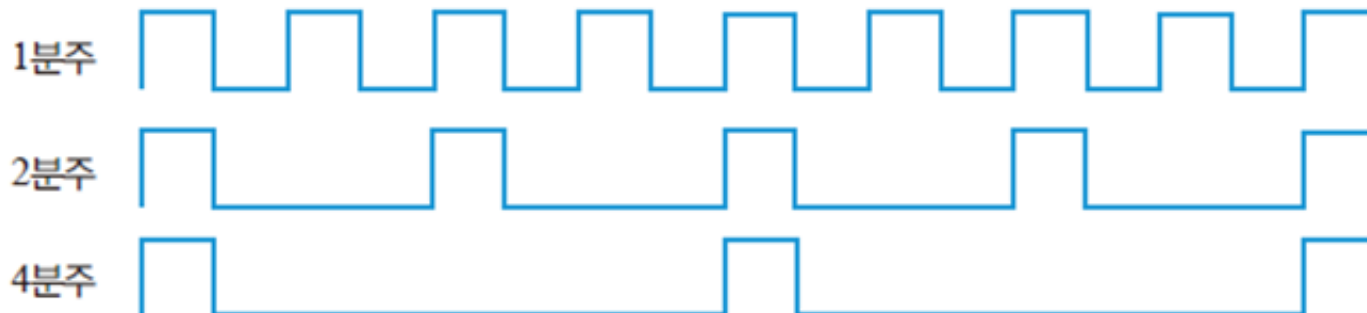
- 입력 clock의 분주(divide) 비
- ATmega128의 시스템 clock 16MHz (**1clock = 1/16MHz = 0.063μsec = 63nsec**)일 때,  
**Pre-scaler = 8**이면, system clock을 8분주한 clock을 카운트하게 됨  
즉 system clock보다 8배 느린 clock을 카운트 하게 됨 → 분주 비만큼 느려짐

주파수 = 16MHz

$$\frac{16 \times 10^6}{8} \text{ Hz} = 2 \times 10^6 \text{ Hz} \xRightarrow{\text{역수}} \frac{1}{2 \times 10^6} \text{ sec} = 0.5 \mu\text{sec} = 1\text{clock}$$

8 Pre-scaler, 8분주

500nsec

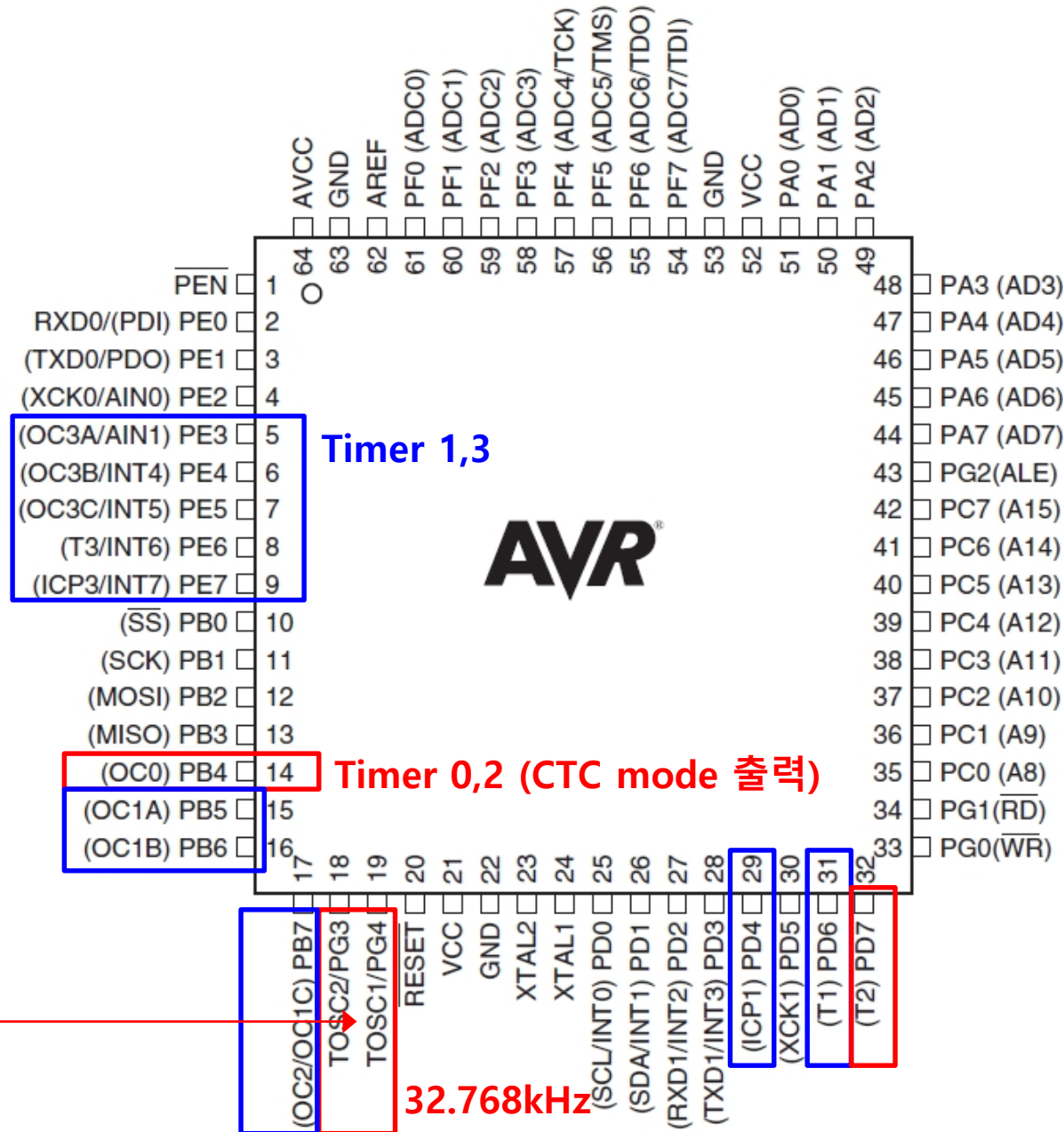


## 6. Timer / Counter

Strictly Private and Confidential

### 8bits Timer / Counter 0, 2

- 8비트 업/다운 카운터
- 10bits Pre-scaler 내장
- 비교 match에서  
timer clear(=0)(auto reload)
- 카운터로 동작할 때  
clock 입력 단자 : TOSC1
- 32.768KHz 시계 clock 접속  
: TOSC1, TOSC2
- 인터럽트 소스  
: Overflow  
: Output Compare match





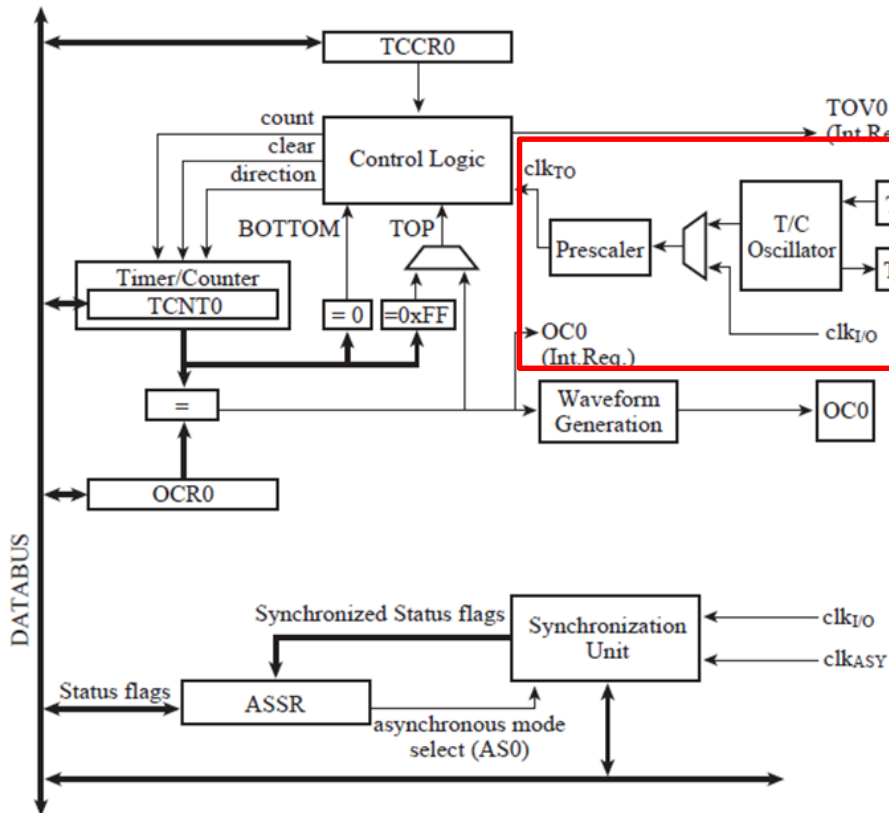
### 8bits Timer / Counter 0,2 Register

Register	기능
<b>TCNTn</b>	Timer/Counter n Register ( <b>TCNT0</b> , <b>TCNT2</b> )
<b>TCCRn</b>	Timer/Counter n <b>Control Register</b> ( <b>TCCR0</b> , <b>TCCR2</b> )
<b>OCRn</b>	<b>Output Compare Register</b> n ( <b>OCR0</b> , <b>OCR2</b> )
<b>TIMSK</b>	Timer/Counter <b>Interrupt Mask Register</b>
<b>TIFR</b>	Timer/Counter <b>Interrupt Flag Register</b>
<b>ASSR</b>	비동기 상태 레지스터(Asynchronous Status Register)
<b>SFIOR</b>	특수 기능 I/O 레지스터(Special Function I/O Register)

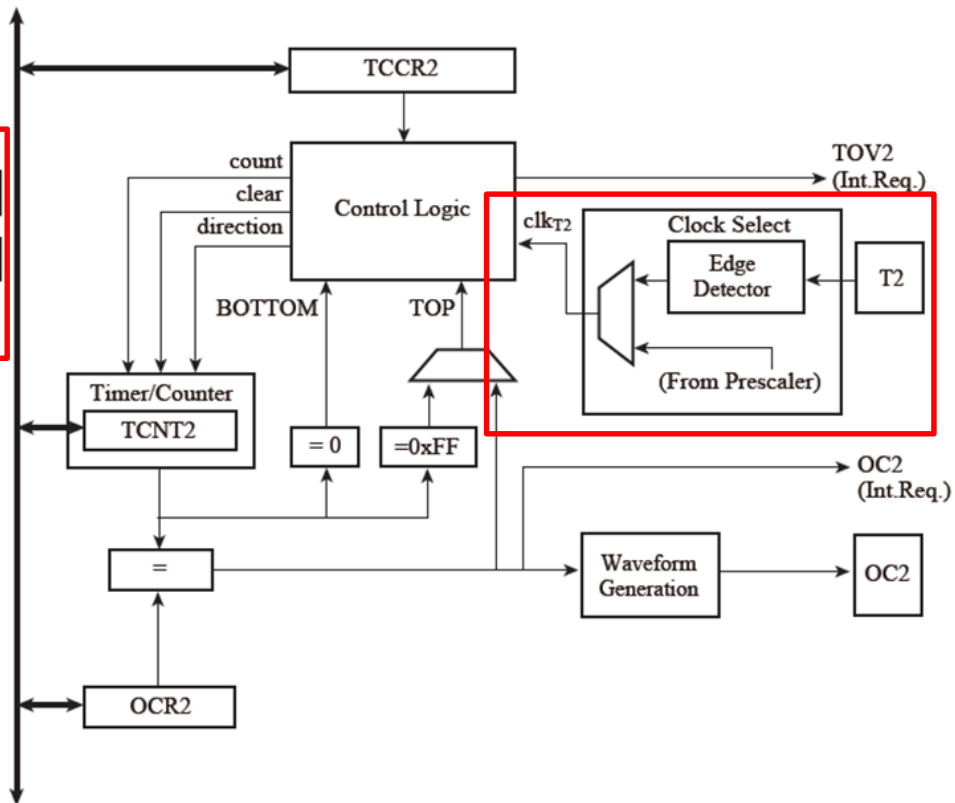
**n : 0, 2**

## 8bits Timer / Counter 0,2 구조

- $clk_{T0}$ ,  $clk_{T2}$  생성 방법을 제외하고는 8bits Timer / Counter 0,2 기능적으로 동일



[8bits Timer / Counter 0 구조]



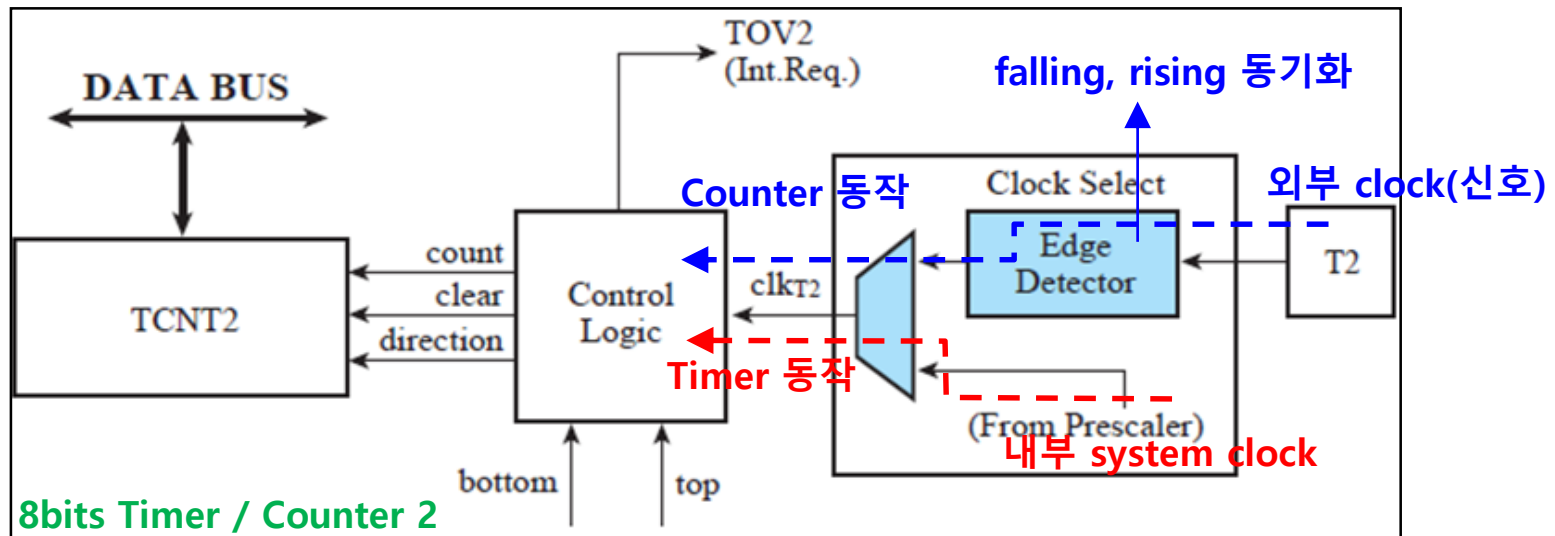
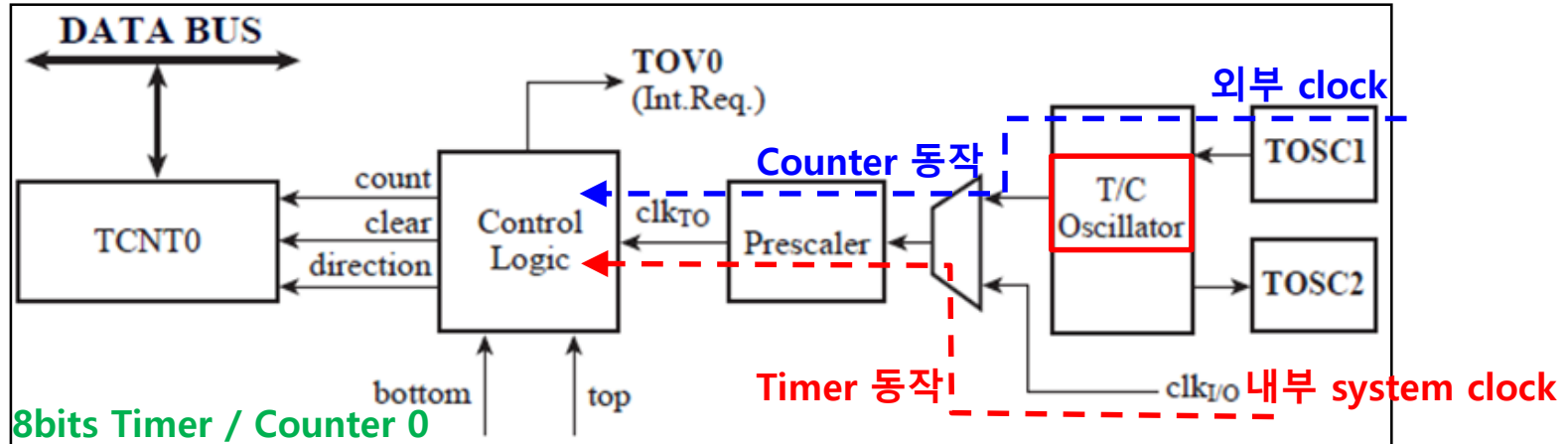
[8bits Timer / Counter 2 구조]

## 6. Timer / Counter


Strictly Private and Confidential

### 8bits Timer / Counter 0,2 구조 & 동작

- $clk_{T0}$ ,  $clk_{T2}$ 의 생성 방법

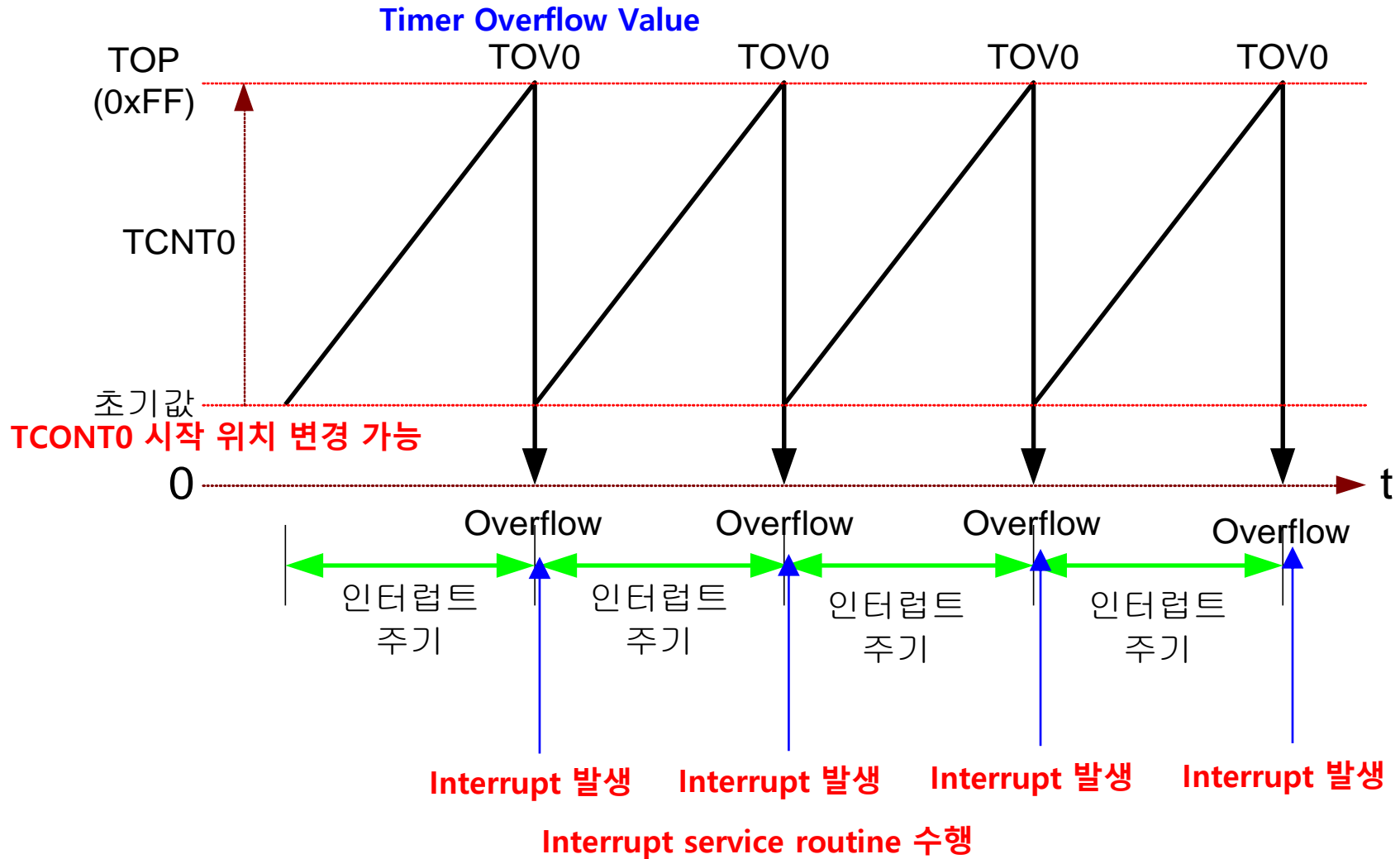


### 8bits Timer / Counter 0,2 동작

- Timer / Counter 0,2 는 4개의 동작 모드를 **TCCRn**의 **WGMn1, WGMn0**로 설정
  - : **Normal mode (overflow mode)**
  - : **CTC(Clear Timer on Compare Match) mode**
  - : **Fast PWM mode**
  - : **PC(Phase Correct) PWM mode**
- Normal mode
  - **Up counter**로만 동작
  - Timer / Counter 0,2 값이 8비트 최대값 0xFF(TOP=0xFF)가 되면, 0x00부터 다시 시작
  - Timer / Counter 0,2 의 **Overflow Interrupt Flag TOV0, TOV2 bit**는  
Register TIFR (Timer Interrupt Flag Register)의 TOV0 [0]bit, TOV2 [6]bit  
  
**TCNT0값이 0xFF에서 0x00으로 될 때 1이 되어 인터럽트 발생**
  - **상한 값(top)** 고정(0xFF), **하한 값(bottom)** TCNT0 초기값 ➔ 초기값으로 주기 조정

### 8bits Timer / Counter 0,2 동작

- Normal mode



## 6. Timer / Counter

Strictly Private and Confidential

엑셀에서 최대, 최소, 예제

### 8bits Timer / Counter 0,2 동작

- Normal mode

#### ➤ Overflow Interrupt

: TCNTn 값이 overflow(FF→00) 될 때, **TOVn=1** 발생

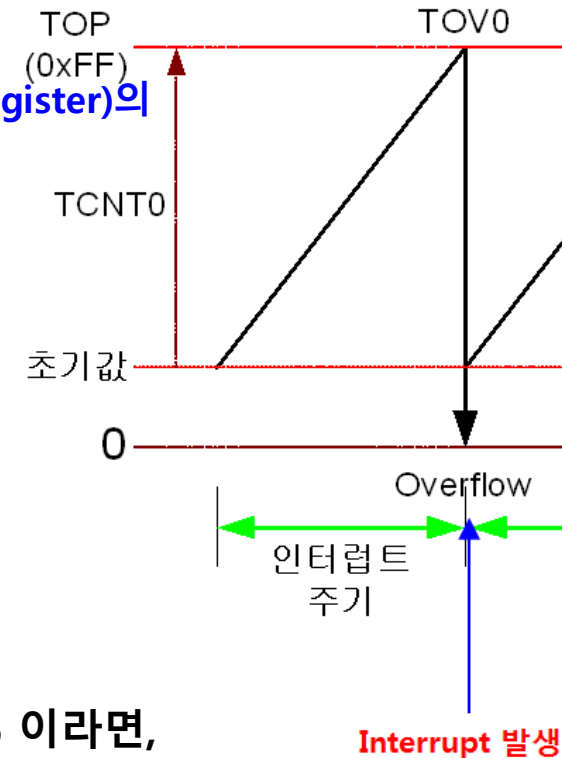
: **TOIE<sub>n</sub>=1** & **SREG의 I=1**로 설정되었으면,  
overflow interrupt service routine 실행

#### ➤ Overflow interrupt 주기

➔  $(1/16)\mu s \times \text{분주 비} \times (256 - \text{TCNT0 초기값})$

예, 시스템 16MHz, TCCR0=0x02(8분주), TCNT0=156 이라면,

인터럽트 주기는 50us마다 계속해서 Overflow interrupt 발생



TIFR (Timer Interrupt Flag Register)의  
TOV0 [0]bit, TOV2 [6]bit



TCCR0

$$\frac{16 \times 10^6}{8} \text{ Hz} = 2 \times 10^6 \text{ Hz} \xRightarrow{\text{역수}} \frac{1}{2 \times 10^6} \text{ sec} = 0.5 \mu\text{sec} = 1 \text{ clock}$$

$$1 \text{ clock} \times (256 - \text{TCNT0 초기값}) = 0.5 \mu\text{sec} \times (256 - 156) = 50 \mu\text{sec}$$

TCNT0 시작 위치 변경 가능

## 6. Timer / Counter

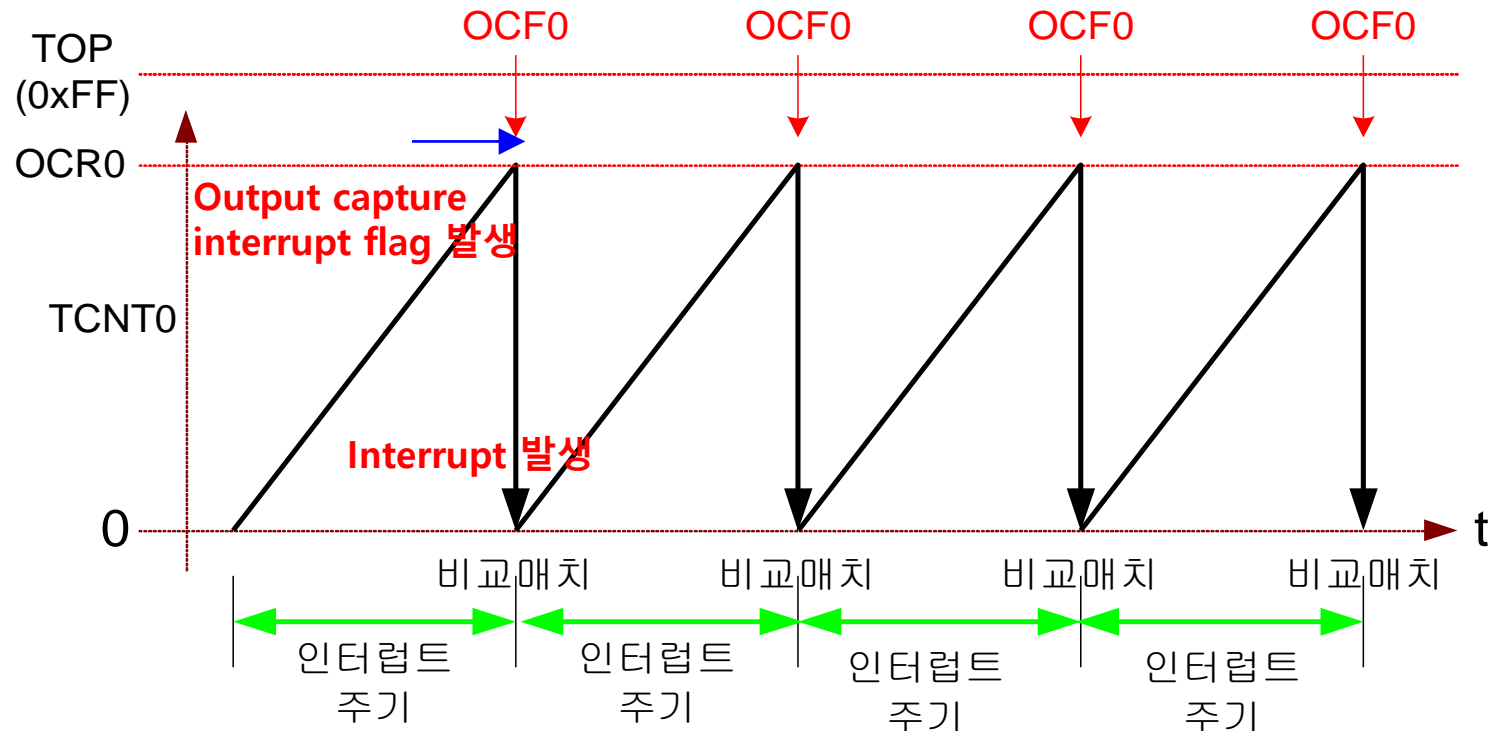
Strictly Private and Confidential

### 8bits Timer / Counter 0,2 동작



Output Compare Register n (OCR0, OCR2)

- CTC(Clear Timer **on Compare Match**) mode
  - TCNTn 이 입력 clock 개수에 따라 **증가하다가 OCRn 값과 일치하면, TCNTn은 0으로 되돌아 감. 즉 다음 clock에서 0으로 clear.**
  - 입력 clock을 **OCRn 값 만큼만 카운트, TCNTn 값은 00H~OCRn 값의 범위를 가짐**  
: TCNTn=OCRn 일 때 **OCn 출력 pin에 파형 출력 가능**



## 6. Timer / Counter

Strictly Private and Confidential

엑셀에서 최대, 최소, 예제

### 8bits Timer / Counter 0,2 동작

- CTC(Clear Timer on Compare Match) mode
- Timer / Counter n 의 **Output compare match interrupt**

:  $TCNTn = OCRn$  일 때,  $OCFn = 1$  이 되면서 발생

$OCIE_n = 1$  & SREG의 I=1로 설정되었으면

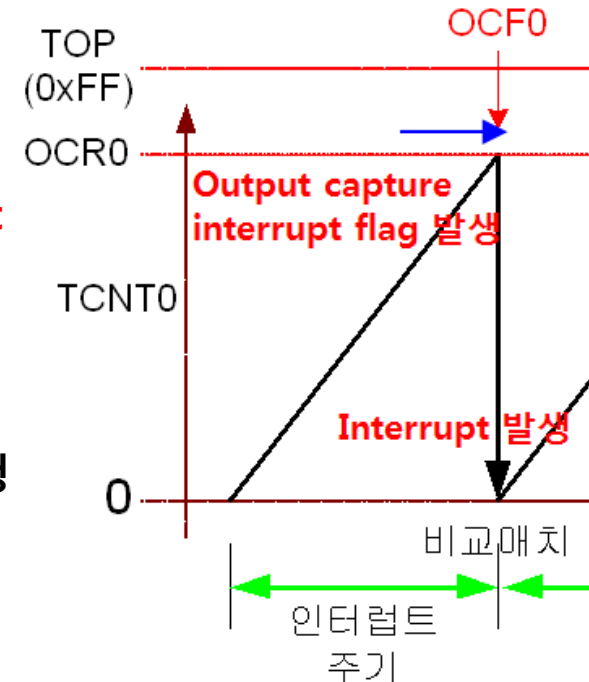
Output compare match interrupt service routine 실행

- **Output compare match interrupt 주기**

➔  $(1/16)\mu s * \text{분주 비} * (1 + OCRn \text{ 초기값})$

예, 시스템 16MHz,  $TCCR0 = 0x0A$  (CTC모드, 8분주),  $OCR0 = 99$  이라면,

인터럽트 주기는  $50\mu s$ 마다 계속해서 **Output compare match interrupt 발생**



★  $TCCR0$

$$\frac{16 \times 10^6}{8} \text{ Hz} = 2 \times 10^6 \text{ Hz} \xRightarrow{\text{역수}} \frac{1}{2 \times 10^6} \text{ sec} = 0.5\mu\text{sec} = 1\text{clock}$$

$$1\text{clock} \times (1 + OCRn \text{ 초기값}) = 0.5\mu\text{sec} \times (1 + 99) = 50\mu\text{sec}$$



### 8bits Timer / Counter 0,2 동작

- CTC(Clear Timer on Compare Match) mode

➤ OCn 출력 pin을 통해 파형 출력

: OCn 핀(PB4, PB7)은 출력으로 설정되어 있어야 출력됨

➔ 예, 시스템 16MHz, TCCR0=0x0A(CTC모드, 8분주), OCR0=99 이라면,

인터럽트 주기는 50us마다 계속해서 Output compare match interrupt 발생

$$\frac{16 \times 10^6}{8} \text{ Hz} = 2 \times 10^6 \text{ Hz} \xRightarrow{\text{역수}} \frac{1}{2 \times 10^6} \text{ sec} = 0.5 \mu\text{sec} = 1\text{clock}$$

$$1\text{clock} \times (1 + \text{OCRn 초기값}) = 0.5\mu\text{sec} \times (1+99) = 50\mu\text{sec}$$

$$\text{주파수} = 1/(50\mu\text{sec}) = 20\text{KHz} \text{ 파형 출력됨}$$

### 8bits Timer / Counter 0,2 동작

- Fast PWM(Fast Pulse Width Modulation) mode
- TCNT0 값은 0에서 0xFF까지 증가하다 다시 0부터 카운트하는 동작 반복

: 비교 출력모드 TCCRn에서 [COMn1 COMn0] = [1 0]에서

TCNT0값은 OCR0와 비교되어 일치하면 → OC0((PB4, PB7) 핀을 통해 0이 출력되고,

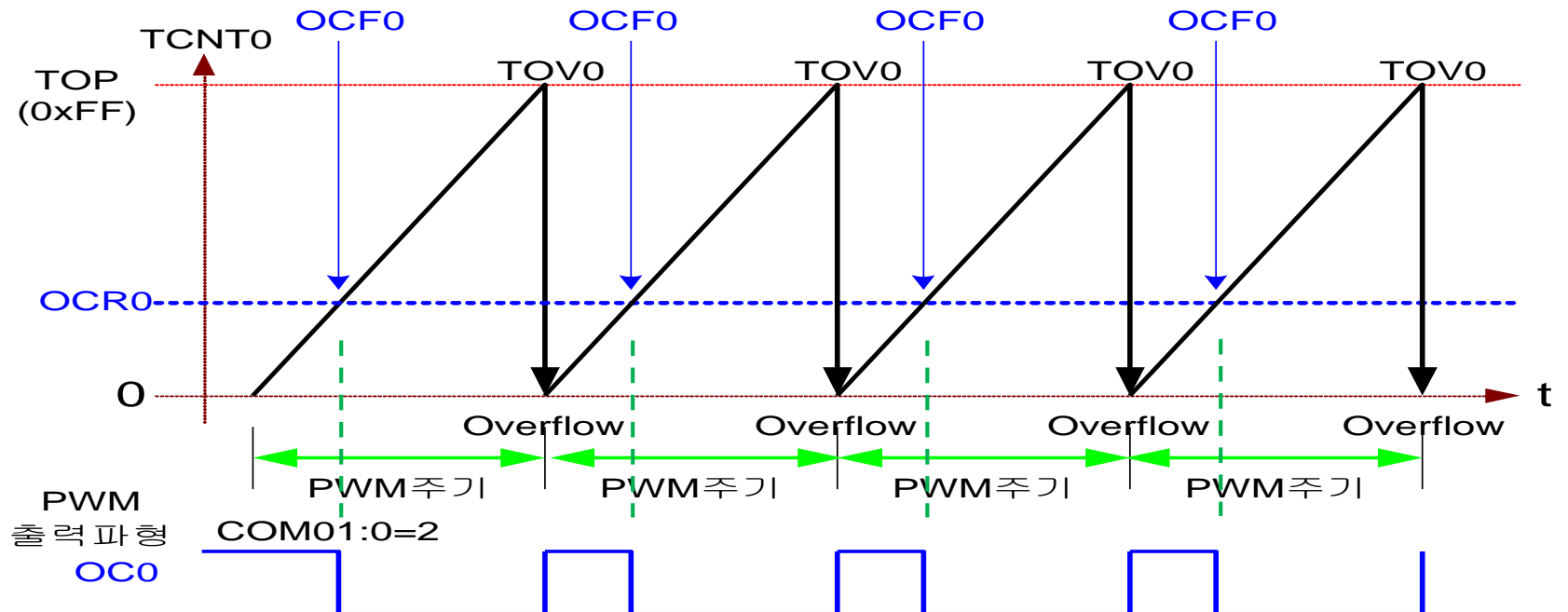
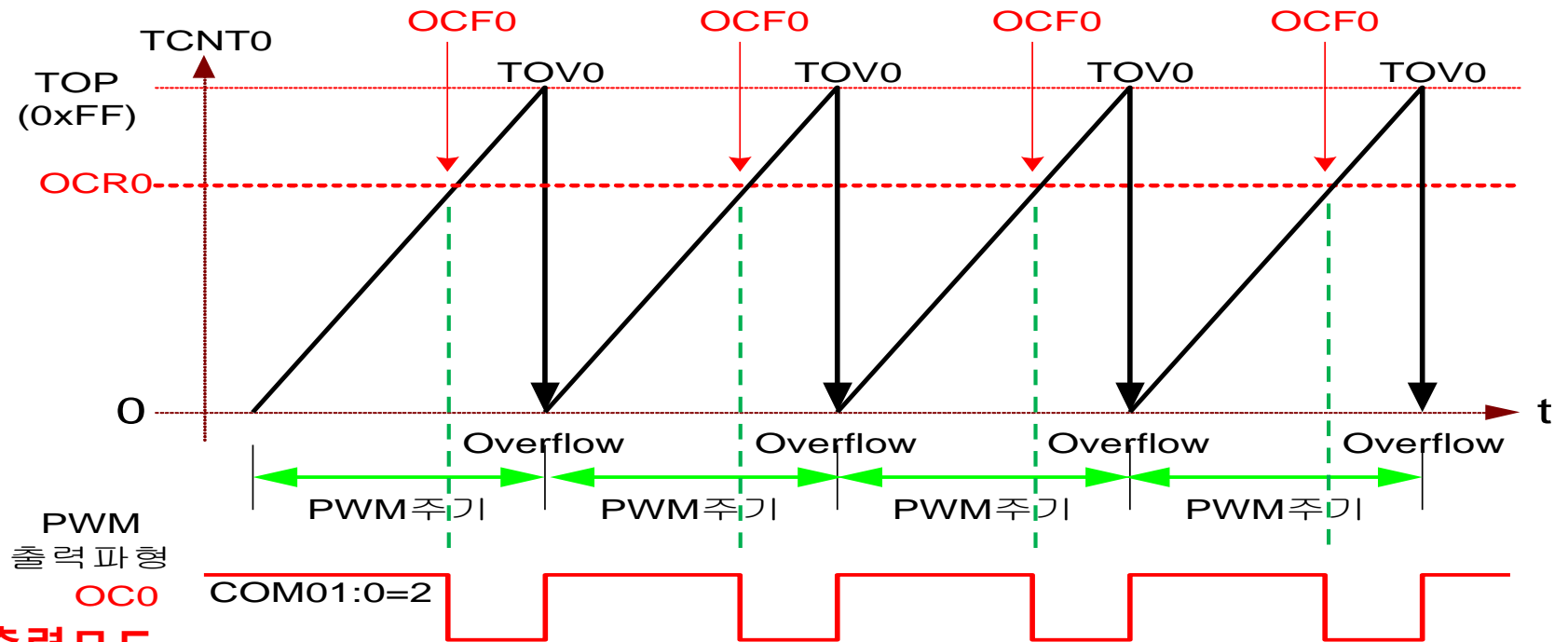
TCNT0값이 0이 되면 1이 출력

: 반전 비교 TCCRn에서 출력모드 [COMn1 COMn0] = [1 1]에서

TCNT0값은 OCR0와 비교되어 일치하면 → OC0((PB4, PB7) 핀을 통해 1이 출력되고,

TCNT0값이 0이 되면 0이 출력

## 비교 출력모드

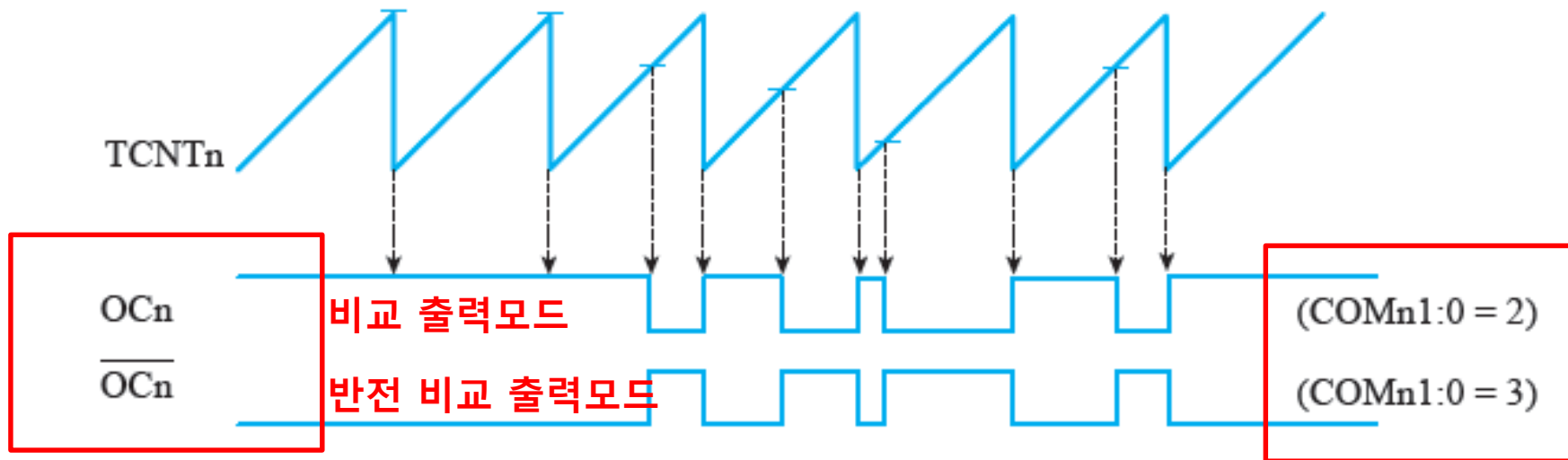


### 8bits Timer / Counter 0,2 동작

- Fast PWM(Fast Pulse Width Modulation) mode

: 비교 출력모드 TCCRn에서  $[COMn1\ COMn0] = [1\ 0]$ 에서

: 반전 비교 TCCRn에서 출력모드  $[COMn1\ COMn0] = [1\ 1]$ 에서



### 8bits Timer / Counter 0,2 동작

- Fast PWM(Fast Pulse Width Modulation) mode

#### ➤ PWM 분해능

: OCRn 값에 따라 PWM 파형의 "H" 펄스 폭을 가변 시킴

: PWM 파형의 분해능은 OCRn이 8비트 레지스터이므로 256을 가짐

: OCRn 값을 00H~FFH 범위 내에서 지정함에 따라,

**PWM 파형의 "H" 폭을 1/256 ~ 256/256 등분으로 가변 (듀티비)시킴**

#### ➤ 인터럽트

: 타이머/카운터n 출력 비교 매치 인터럽트

: TNCTn=OCRn일 때 **OCFn=1이 되면서 발생**

: OCIE<sub>n</sub>=1 & SREG의 I=1로 설정되었다면,

**타이머/카운터n 출력 비교 매치 인터럽트 서비스 루틴이 실행됨**

### 8bits Timer / Counter 0,2 동작

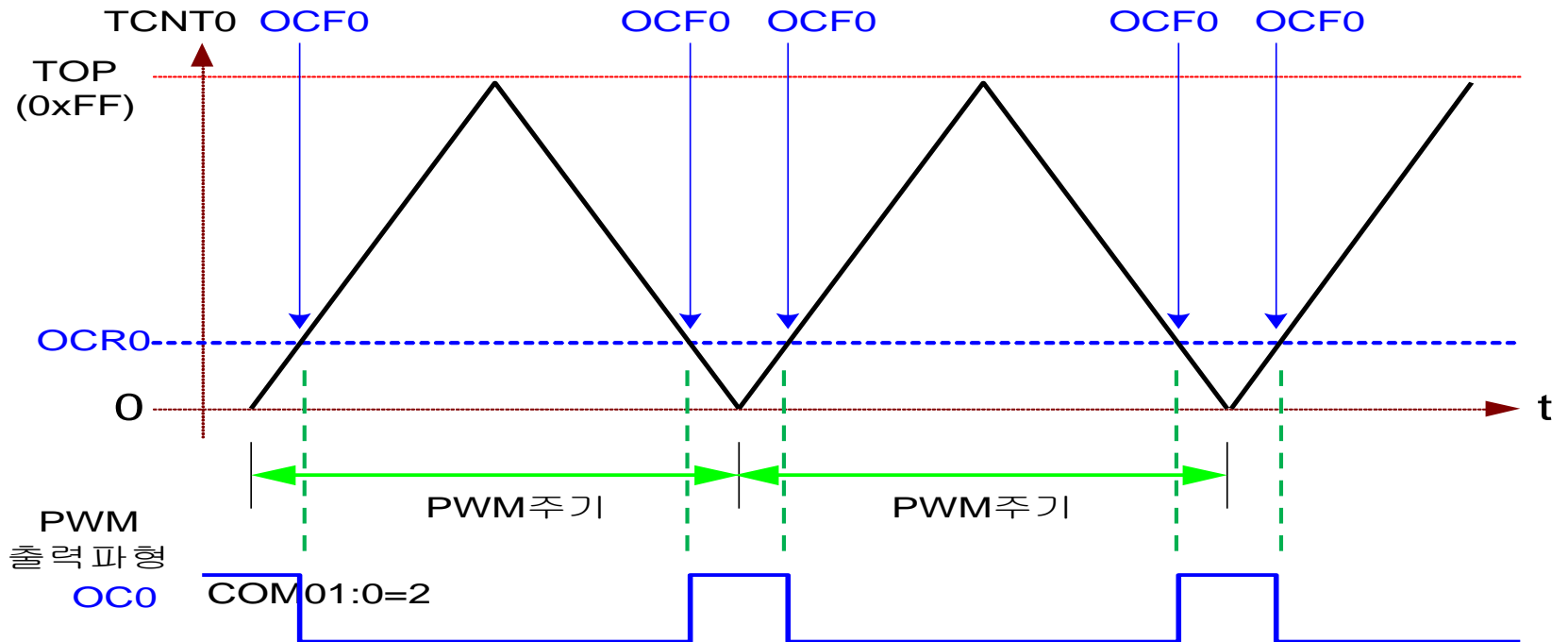
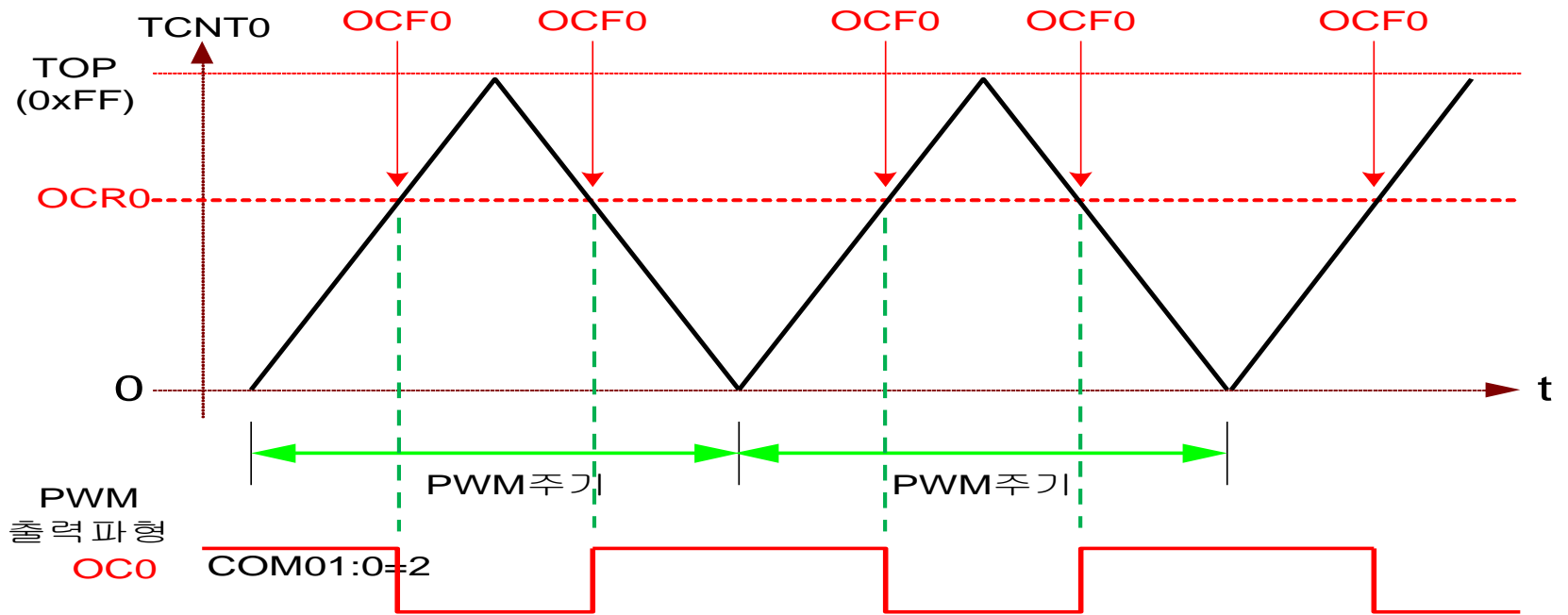
- Fast PWM(Fast Pulse Width Modulation) mode
- OCn 출력 파형 주기  
: OCn 핀(PB4, PB7)은 **출력으로 설정 되어 있어야 출력됨**

$$f_{\text{OCnFPWM}} = \frac{f_{\text{clkI/O}}}{N \cdot (1 + 255)} = \frac{f_{\text{clkTn}}}{256}$$

N : 분주 비, fclkI/O : 시스템 clock 주파수(16MHz)

### 8bits Timer / Counter 0,2 동작

- **PC(Phase Correct) PWM mode**
  - 높은 분해능의 PWM 파형을 발생하는데 유용
  - TCNT0 값은 0에서 255까지 **증가하였다가 0으로 감소하는 동작 반복**  
: **Up count, Down count**
  - **Overflow Interrupt**  
: TCNT0값이 0x00이 될 때마다 발생,  
: TCNT0과 OCR0값이 일치하면 다음 clock에서 Output compare match interrupt 발생
  - **PWM 분해능**  
: OCn로 출력되는 파형은 **Fast PWM 모드에 비해 2배의 분해 능을 가지는 반면에,**  
**주파수는 1/2이 됨**





### 8bits Timer / Counter 0,2 동작

- PC(Phase Correct) PWM mode

➤ OCn 출력 파형 주기

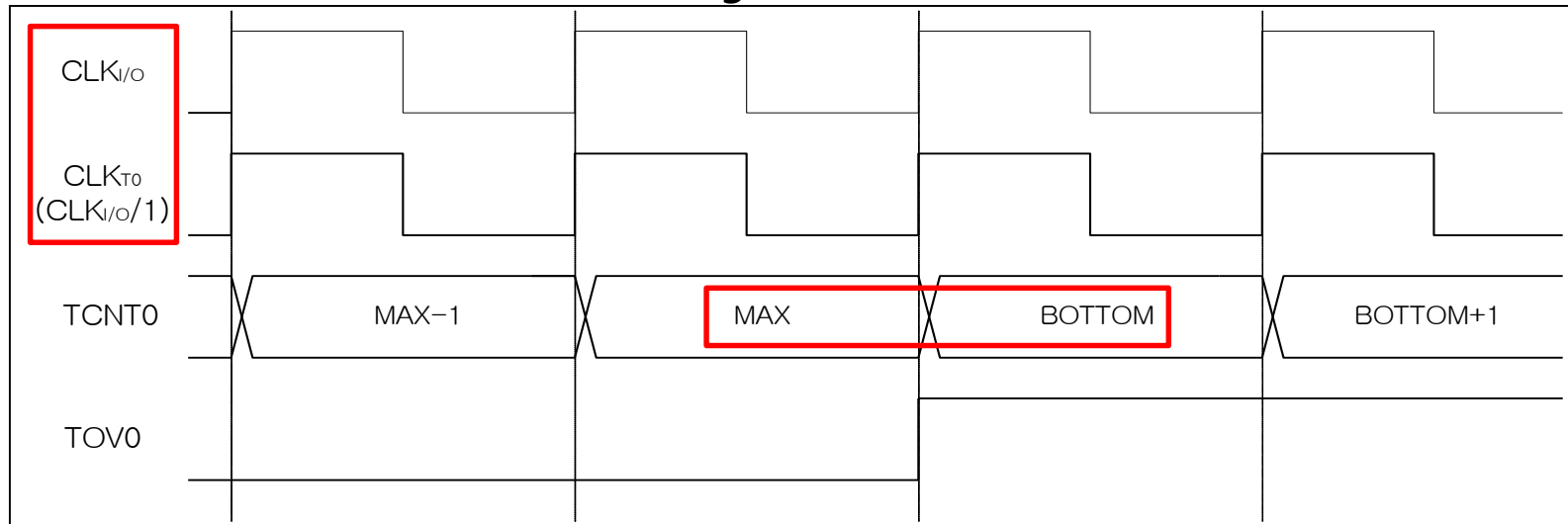
: OCn 핀(PB4, PB7)은 **출력으로 설정 되어 있어야 출력됨**

$$f_{OCnPCPWM} = \frac{f_{clk_{I/O}}}{2 \cdot N \cdot 255} = \frac{f_{clk_{Tn}}}{510}$$

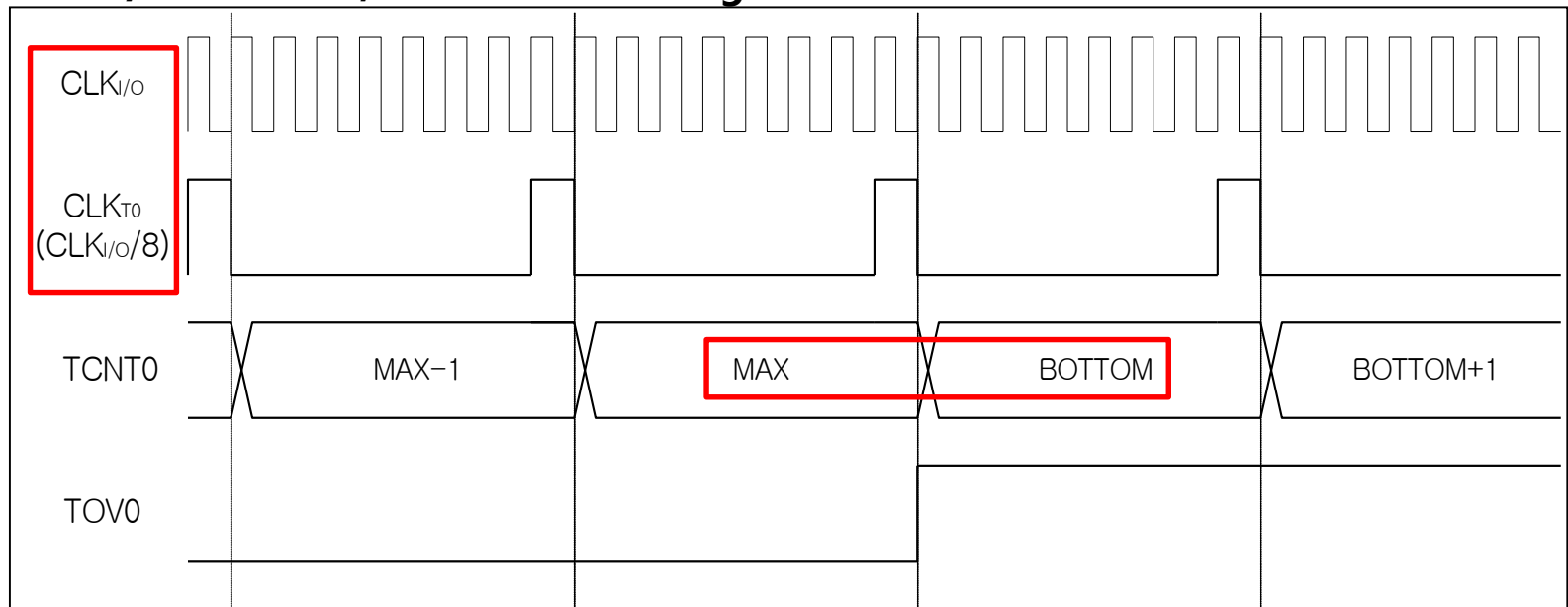
N : 분주 비, fclk<sub>I/O</sub> : 시스템 clock 주파수(16MHz)

## 8bits Timer / Counter 0,2 Timing Chart

- Timer / Counter 0,2 Overflow Timing : **Pre-scaler = 1**

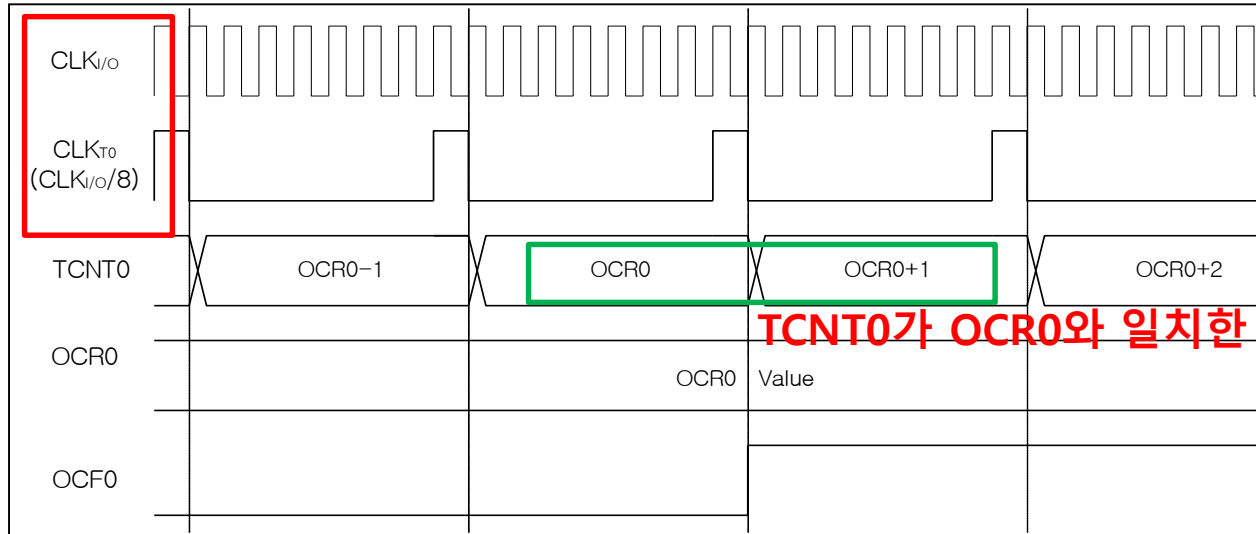


- Timer / Counter 0,2 Overflow Timing : **Pre-scaler = 8**



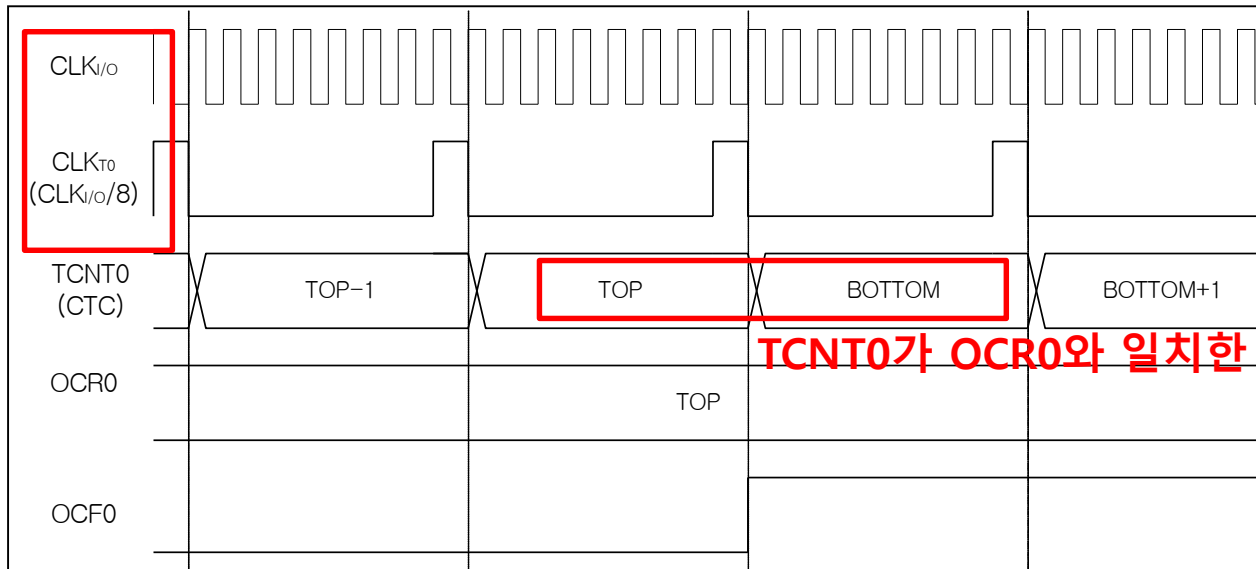
## 8bits Timer / Counter 0,2 Timing Chart

- Timer / Counter 0,2 Output Compare Match Timing : Pre-scaler = 8



TCNT0가 OCR0와 일치한 후에도 계속 증가

- Timer / Counter 0,2 Output Compare Match Timing : Pre-scaler = 8



TCNT0가 OCR0와 일치한 후에 0으로 리셋

## 8bits Timer / Counter register

- 타이머/카운터0 제어 레지스터(TCCR0, TCCR2 : Timer/Counter0 Control Register)

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### Bit 7

: FOC0(Force Output Compare)

: 이 비트는 PWM모드가 아닌 경우에만 유효

: 1이면 비교 출력 OC0를 통해 COM01 : COM00 비트에 의해 설정된 값 즉시 출력

### Bit 6, Bit 3

: **WGM00, WGM01(Waveform Generation Mode)**

: 타이머/카운터0의 동작모드 설정

모드	WGM01[3]	WGM00[6]	동작 모드	TOP	OCR0 업데이트시점	TOV0 Flag set 시점
0	0	0	Normal	0xFF	설정 즉시	MAX
1	0	1	Phase Correct PWM	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	설정 즉시	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

## 8bits Timer / Counter register

- 타이머/카운터0 제어 레지스터(TCCR0, TCCR2 : Timer/Counter0 Control Register)

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 5, Bit 4

: COM01, COM00 (**Compare Match Output Mode**)

: 타이머/카운터0의 **출력단자 OC0핀의 동작 제어**

: OC0에 해당하는 데이터 방향 레지스터 DDRB의 4비트는 **출력이 되도록 설정**

COM01	COM00	OC0핀의 출력기능
0	0	일반 I/O 포트 동작 ( <b>OC0 차단</b> )
0	1	<b>비교 매치에서 OC0 출력</b>
1	0	비교 매치에서 OC0 clear(=0) <b>비교 출력 (PWM)</b>
1	1	비교 매치에서 OC0 set(=1) <b>반전 출력 (PWM)</b>

### 8bits Timer / Counter register

- 타이머/카운터0 제어 레지스터(TCCR0, TCCR2 : Timer/Counter0 Control Register)

COM01	COM00	OC0핀의 출력기능
0	0	일반 I/O 포트 동작(OC0 차단)
0	1	-
1	0	업 카운트 중의 비교매치에서는 OC0 clear(=0), 다운 카운트 중에 비교매치에서는 set(=1)
1	1	업 카운트 중의 비교 매치에서는 OC0 set(=1), 다운 카운트 중에 비교매치에서는 clear(=0)

#### Phase Correct PWM 모드인 경우의 출력기능

COM01	COM00	OC0핀의 출력기능
0	0	일반 I/O 포트 동작(OC0 차단)
0	1	-
1	0	비교 매치에서 OC0 clear(=0), TOP에서 set(=1) 비교 출력 (PWM)
1	1	비교 매치에서 OC0 set(=1), TOP에서 clear(=0) 반전 출력 (PWM)

#### Fast PWM 모드인 경우의 출력기능

## 8bits Timer / Counter register

- 타이머/카운터0 제어 레지스터(TCCR0, TCCR2 : Timer/Counter0 Control Register)

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit [2] ~ [0]

: CS02 : CS00 (clock 선택)

: 타이머/카운터0의 분주 비(Pre-scaler) 를 결정

CS02	CS01	CS00	기 능
0	0	0	clock 입력 차단(타이머/카운터 정지)
0	0	1	$clk_{TOS} / 1$
0	1	0	$clk_{TOS} / 8$
0	1	1	$clk_{TOS} / 32$
1	0	0	$clk_{TOS} / 64$
1	0	1	$clk_{TOS} / 128$
1	1	0	$clk_{TOS} / 256$
1	1	1	$clk_{TOS} / 1024$

### 8bits Timer / Counter register

- 타이머/카운터0 레지스터(TCNT0, 2 Timer/Counter0 Register)

: 타이머/카운터0, 2가 카운트하고 있는 입력 펄스 개수를 기록하는 8비트 레지스터

bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
초기값	0	0	0	0	0	0	0	0	

bit	7	6	5	4	3	2	1	0	
	TCNT2[7:0]								TCNT2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
초기값	0	0	0	0	0	0	0	0	

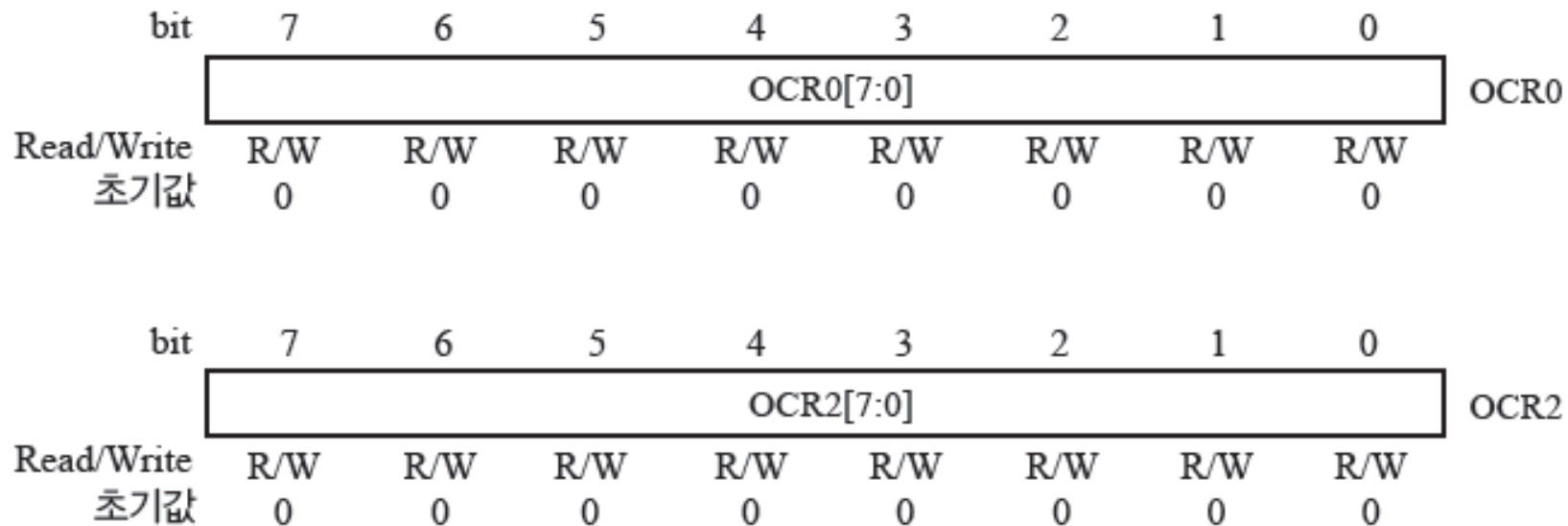


### 8bits Timer / Counter register

- OCR0, 2(Output Compare Register 0, 2)

: TCNT0, 2의 값과 비교하는 8비트 출력 비교 레지스터

: OCRn 과 TCNTn 값이 같을 때 OCn 핀 (PB4, PB7) 으로 파형 출력



## 8bits Timer / Counter register

- TIMSK (Timer/Counter Interrupt Mask Register)

: 타이머/카운터의 인터럽트의 enable / disable 설정

bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
초기값	0	0	0	0	0	0	0	0	

: OCIE<sub>n</sub> (Timer/Counter 0 Output Compare Match Interrupt Enable) : n=0, 2

- 타이머/카운터<sub>n</sub>의 출력 비교 매치 enable / disable 비트

- **TCNT<sub>n</sub>=OCR<sub>n</sub> 일 때 TIFR의 OCF<sub>n</sub>=1** 이 되고,

타이머/카운터<sub>n</sub> 출력비교 매치 인터럽트 발생

- **OCIE<sub>n</sub>=1 & SREG의 I=1이면**, 타이머/카운터<sub>n</sub>의 출력 비교 매치 인터럽트가 처리

: TOIE<sub>n</sub> (Timer/Counter n Overflow Interrupt Enable) : n=0, 2

- 타이머/카운터<sub>n</sub>의 Overflow Interrupt Enable

- 타이머/카운터<sub>n</sub>에서 Overflow (0xFF→0x00)가 발생하면,

TIFR의 TOV<sub>n</sub>=1이 되고 타이머/카운터<sub>n</sub> Overflow Interrupt 발생

- **TOIE<sub>n</sub>=1 & SREG의 I=1이면**, 타이머/카운터<sub>n</sub>의 Overflow Interrupt 처리

## 8bits Timer / Counter register

- TIFR (Timer/Counter Interrupt Flag Register)

: 타이머/카운터의 **인터럽트 발생 여부를 기록**

bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
초기값	0	0	0	0	0	0	0	0	

: OCFn (Output Compare **Flag** n)

- 타이머/카운터n의 Output Compare Match Flag Bit
- TCNTn 값과 OCRn 값이 일치하면 **OCFn=1**이 됨
- 타이머/카운터n Output Compare Match 인터럽트 **서비스 루틴이 실행되면**,

자동적으로 **OCFn=0**이 됨

: TOVn (Timer/Counter n Overflow **Flag**)

- 타이머/카운터n의 Overflow 플래그 비트
- 타이머/카운터n이 Overflow (FFH→00H)될 때 TOVn=1이 됨
- 타이머/카운터n Overflow 인터럽트 **서비스 루틴이 실행되면**,

자동적으로 **TOVn=0**이 됨

## 8bits Timer / Counter register

- ASSR (Asynchronous Status Register)

: 타이머/카운터0를 카운터 동작으로 설정할 때 사용

bit	7	6	5	4	3	2	1	0	
	-	-	-	-	AS0	TCN0UB	OCR0UB	TCR0UB	ASSR
Read/Write 초기값	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	

: AS0 (Asynchronous Timer/Counter 0)

- 타이머/카운터0의 clock source 선택 bit
- **AS0=0** → 내부 clock **clkI/O**가 선택되어 타이머(동기 모드)로 동작
- **AS0=1** → TOSC1 으로 입력되는 신호가 선택되어 카운터(비동기 모드)로 동작
- 이 비트가 변경되면 **TCNT0, OCR0, TCCR0** 값이 바뀔 수 있기 때문에 주의를 요함

: TCN0UB (Timer/Counter 0 **Update Busy**)

- 타이머/카운터0가 카운터로 동작할 때 **TCNT0**에 새로운 값으로 업데이트 되면,  
**TCN0UB=1**이 되고  
**TCNT0**에 **write**가 완료되면 자동적으로 **TCN0UB=0**이 됨

## 8bits Timer / Counter register

- ASSR (Asynchronous Status Register)

: 타이머/카운터0를 카운터 동작으로 설정할 때 사용

bit	7	6	5	4	3	2	1	0	
	-	-	-	-	AS0	TCN0UB	OCR0UB	TCR0UB	ASSR
Read/Write 초기값	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	

: OCR0UB (Output Compare Register 0 Update Busy)

- 타이머/카운터0가 카운터로 동작할 때 OCR0에 새로운 값이 write되면

이 비트는 set(=1) 되고

OCR0에 write 가 완료되면 자동적으로 clear(=0)

: TCR0UB (Timer/Counter Control Register 0 Update Busy)

- 타이머/카운터0가 카운터로 동작할 때 TCCR0에 새로운 값이 write되면

이 비트는 set(=1) 되고

TCCR0에 write가 완료되면 자동적으로 clear(=0)

- 
1. Micro Processor 원리
  2. Atmel사의 8bit Micro-controller
  3. KUT0128 Evaluation Board 기능과 특징
  4. IO Port 제어
  5. External Interrupt 제어
  6. Timer / Counter 제어
  7. UART 제어
  8. AD Converter 제어
  9. Comparator 제어
  10. EEPROM 제어 (IIC, Parallel method)
  11. SPI 제어

### 16bits Timer / Counter

- Timer / Counter 0, Timer / Counter 1 , Timer / Counter 2 , Timer / Counter 3

Timer / Counter	설명
Timer / Counter 0 Timer / Counter 2	<ul style="list-style-type: none"><li>• 8bits Timer / Counter</li><li>• 최대 <math>2^8(=256, 00H\sim FFH)</math>개의 입력 펄스를 카운트</li><li>• Counter 개수가 FF일 때 1개를 카운트하면, 00으로 반복해서 카운트</li></ul>
<hr/> <b>16bits TIMER 의미는?</b> <hr/>	
Timer / Counter 1 Timer / Counter 3	<ul style="list-style-type: none"><li>• 16bits Timer / Counter</li><li>• 최대 <math>2^{16}(=65536, 0000H\sim FFFFH)</math>개의 입력 펄스를 카운트</li><li>• Counter 개수가 FFFF일 때 1개를 카운트하면, 0000으로 반복해서 카운트</li></ul>

## 6. Timer / Counter

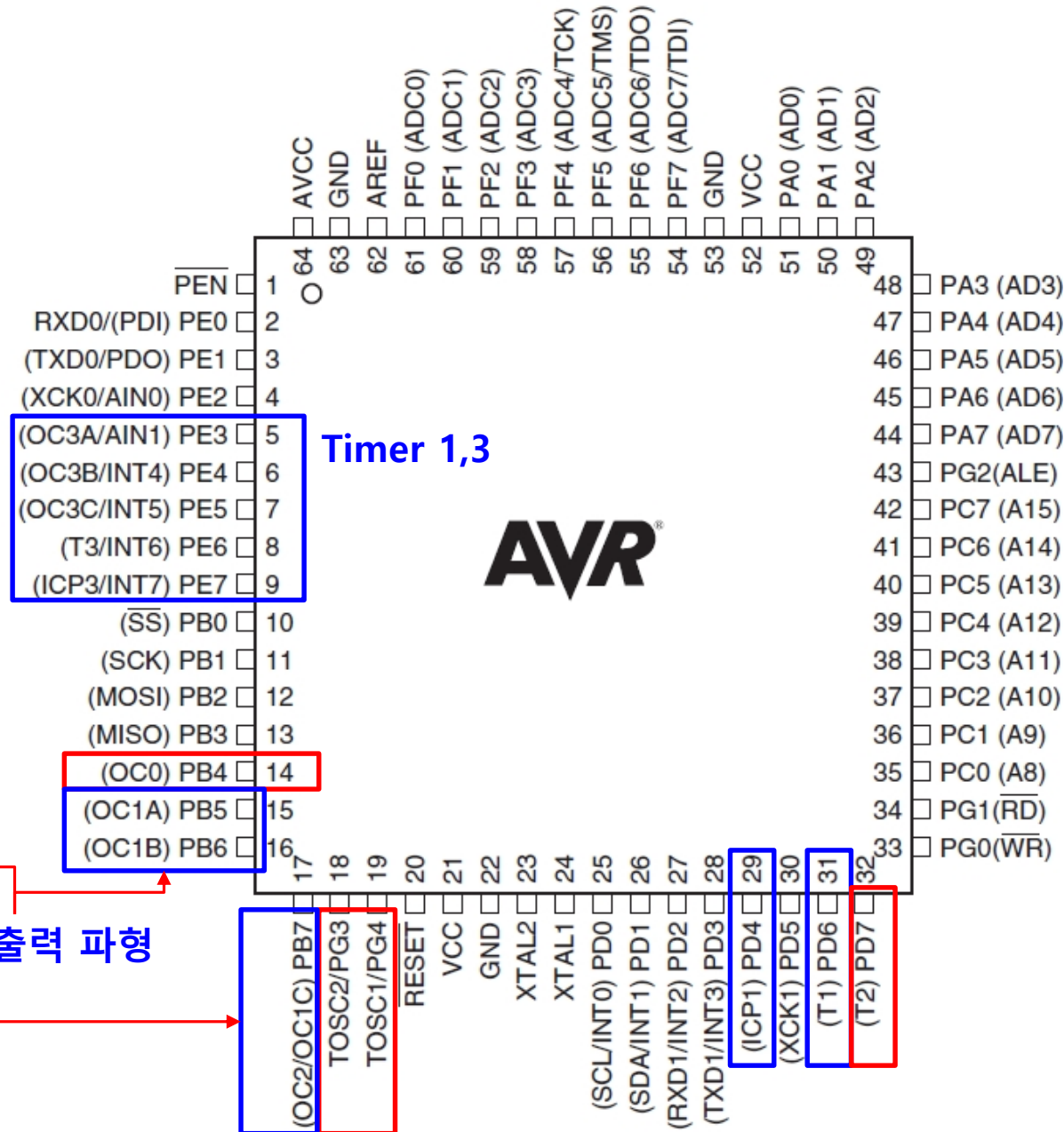
Strictly Private and Confidential

### 16bits Timer / Counter 1, 3

- 16비트 업/다운 카운터
- 10bits Pre-scaler 내장
- 비교 match에서  
timer clear(=0)(auto reload)

Timer 1,3 출력 파형

Timer 1,3 출력 파형





### 16bits Timer / Counter 1,3 Register

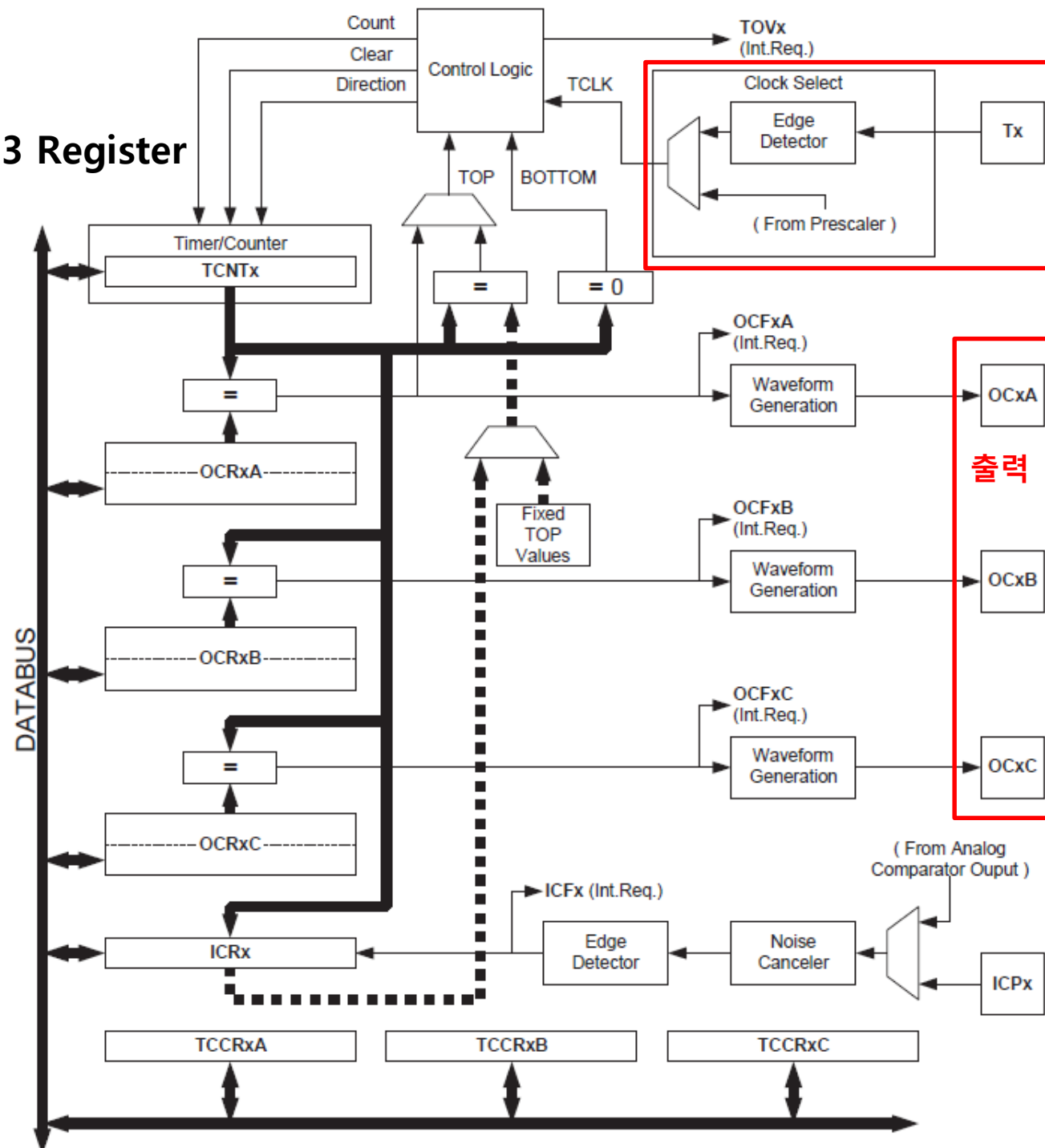
Register	기능
TCNTn	Timer/Counter n Register (TCNT1, TCNT3)
TCCRn	Timer/Counter n Control Register (TCCR1A, TCCR1B, TCCR1C, TCCR3A, TCCR3B, TCCR3C)
OCRn	Output Compare Register n (OCR1A, OCR1B, OCR1C, OCR3A, OCR3B, OCR3C)
ICRn	Input Capture Register
TIMSK	Timer/Counter Interrupt Mask Register
TIFR	Timer/Counter Interrupt Flag Register
SFIOR	특수 기능 I/O 레지스터(Special Function I/O Register)

n : 1, 3

## 6. Timer / Counter

### 16bits Timer / Counter 1,3 Register

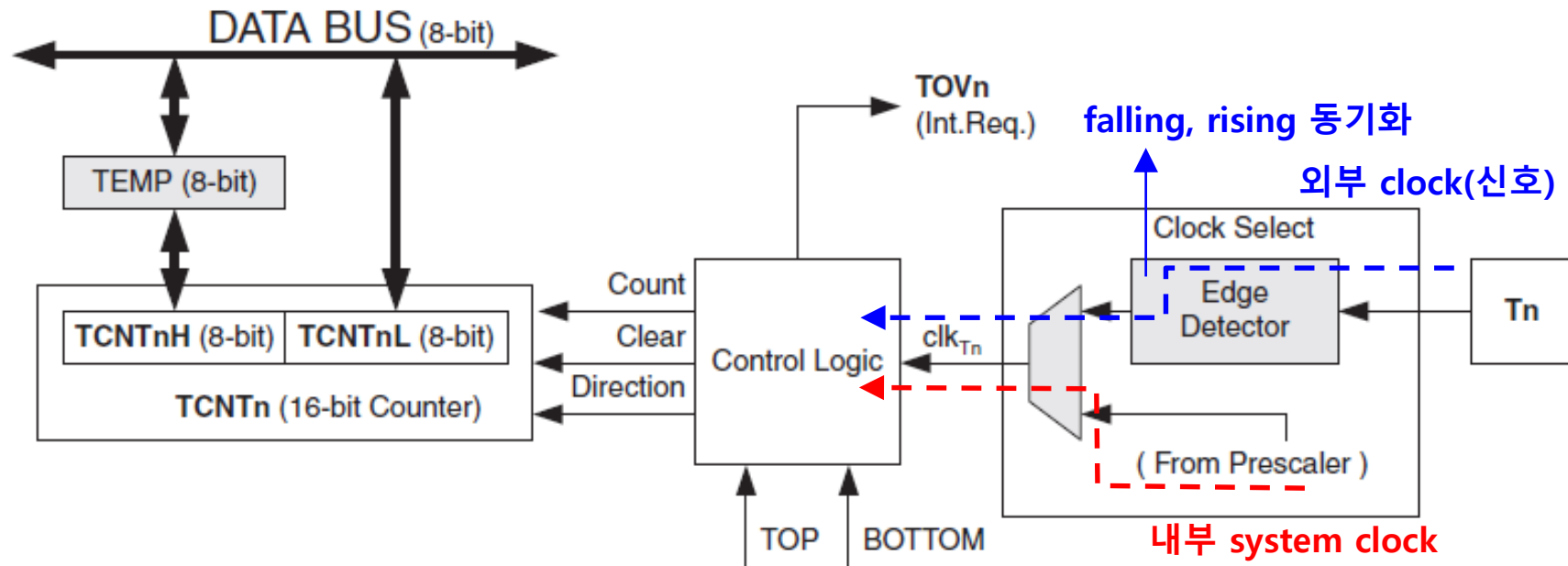
- Clock은 pre-scaler를  
내부 또는  
 $T_n(n=1,3)$  핀을 통해  
입력된 외부 clock 사용
- 출력파형은  
 $OC_nA, OC_nB, OC_nC$ 를  
통해 출력되며,
- $ICP_n$ 은  
타이머/카운터1,3의  
 $TCNT_n$ 값을  $ICR_n$ 에  
저장할 때 사용되는  
trigger 입력



## 6. Timer / Counter

Strictly Private and Confidential

### 16bits Timer / Counter 1,3 구조 & 동작



- 타이머/카운터 제어 레지스터(TCCR1A, TCCR3A : Timer/Counter0 Control Register)

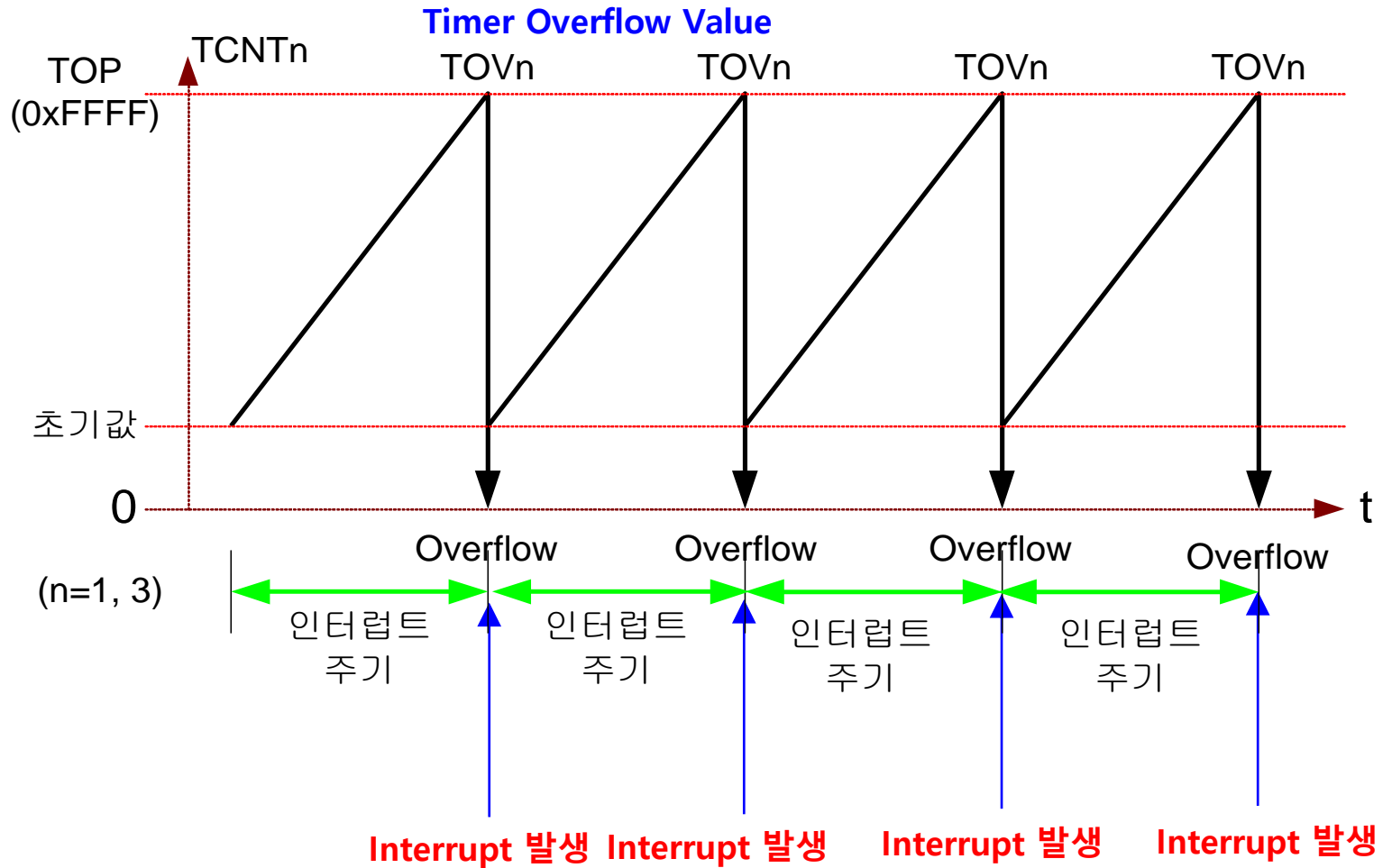
TCCRNb			TCCRNbA		Timer/Counter Mode of Operation <sup>(1)</sup>	TOP	Update of OCRnX at	TOVn Flag Set on
Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)				
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM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

### 16bits Timer / Counter 1,3 동작

- Timer / Counter 1,3 는 4개의 동작 모드를 **TCCR1n, TCCR3n** 의 [1],[0]bit로 설정
  - : Normal mode
  - : CTC(Clear Timer on Compare Match) mode
  - : Fast PWM mode
  - : PC(Phase Correct) PWM mode
- Normal mode
  - TIFR (Timer Interrupt Flag Register)의 TOV1 [2]bit,  
ETIFR (Extended Timer Interrupt Flag Register)의 TOV3 [2]bit
  - Up counter로만 동작
  - Timer / Counter 1,3 값이 16비트 최대값 TOP=0xFFFF가 되면, 0x0000부터 다시 시작
  - Timer / Counter 1,3 의 **Overflow Interrupt Flag TOV1, TOV3**는  
TCNT0값이 0xFFFF에서 0x0000으로 될 때 1이 되어 인터럽트 발생
  - 상한 값 고정(0xFFFF), 하한 값 TCNT0 초기값 ➔ 초기값으로 주기 조정

### 16bits Timer / Counter 1,3 동작

- Normal mode



## 6. Timer / Counter

Strictly Private and Confidential

### 16bits Timer / Counter 1,3 동작

- Normal mode

#### ➤ Overflow Interrupt

: TCNTn 값이 overflow 발생할 때 TOVn=1 발생

: TOIE n=1 & SREG의 I=1로 설정되었으면,

overflow interrupt service routine 실행

#### ➤ Overflow interrupt 주기

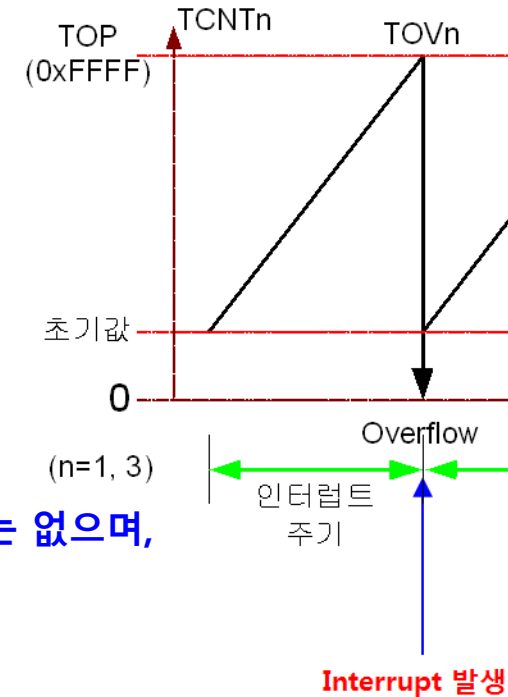
➔  $(1/16)\mu s \times \text{분주 비} \times (65536 - \text{TCNTn 초기값})$

예, 시스템 16MHz, TCCRnA=0x00, TCCRnB=0x02(8분주), TCNTn=55536 이라면,

인터럽트 주기는 5ms마다 계속해서 Overflow interrupt 발생

$$\frac{16 \times 10^6}{8} \text{ Hz} = 2 \times 10^6 \text{ Hz} \quad \overset{\text{역수}}{\leftrightarrow} \quad \frac{1}{2 \times 10^6} \text{ sec} = 0.5 \mu\text{sec}$$

$$\begin{aligned} &1\text{clock} \times (65536 - \text{TCNTn 초기값}) \\ &= 0.5\mu\text{sec} \times (65536 - 55536) = 5\text{msec} \end{aligned}$$



Interrupt 발생

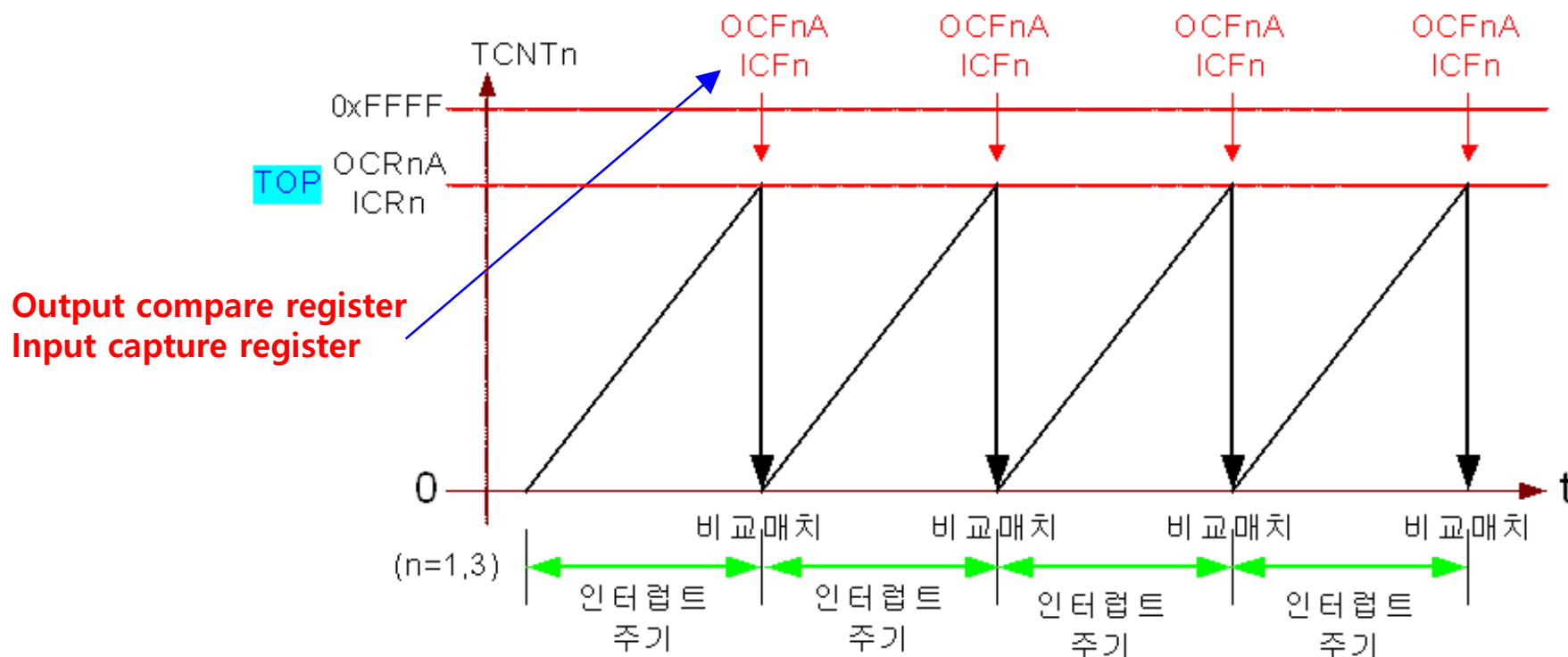
초기값이 0일 필요는 없으며,



인터럽트  
주기

### 16bits Timer / Counter 1,3 동작

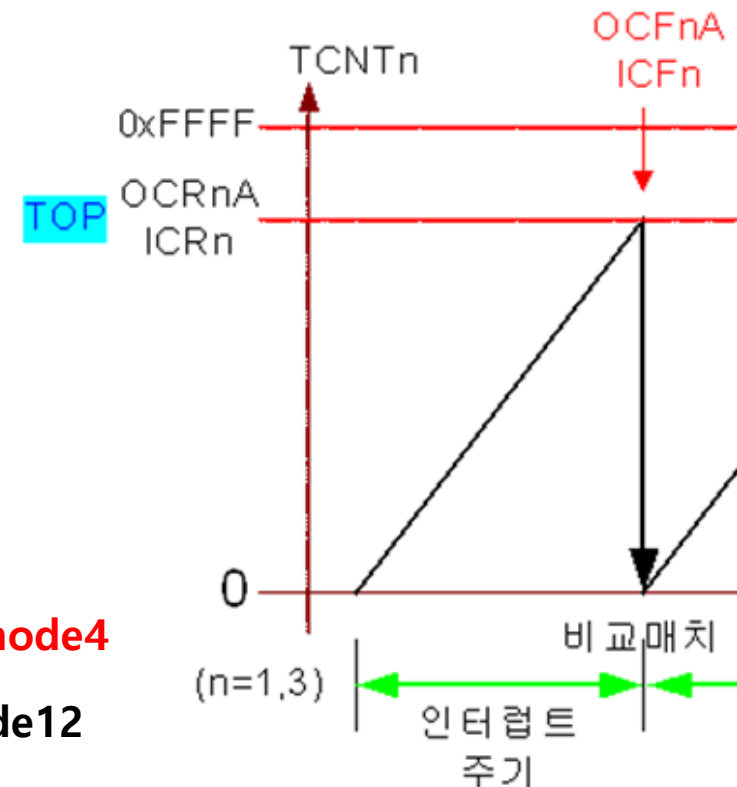
- CTC(Clear Timer on Compare Match) mode
  - TCNTn 이 입력 clock 개수에 따라 증가하다가  
OCRnA(mode4), ICRn(mode12)값과 일치하면, TCNTn은 다음 clock에서 0으로 clear.
  - TCNTn 값은 00H(하한 값은 0으로 고정) ~ OCRnA (또는 ICRn) 값의 범위를 가짐





### 16bits Timer / Counter 1,3 동작

- CTC(Clear Timer on Compare Match) mode
    - [Mode4] **Output compare match interrupt**  
:  $TCNTn = OCRnA$  일 때,  $OCFnA = 1$  이 되면서 발생
    - [Mode12] **Input capture interrupt**  
:  $TCNTn = ICRn$  일 때,  $ICFn = 1$  이 되면서 발생
    - **Interrupt 주기**
      - ➔  $(1/16)\mu s * \text{분주 비} * (1 + OCRnA \text{ 초기값})$  : **mode4**
      - ➔  $(1/16)\mu s * \text{분주 비} * (1 + ICRn \text{ 초기값})$  : **mode12**
- 시스템 16MHz,  $OCnA$  핀을 통해 주파수 출력.



### 16bits Timer / Counter 1,3 동작

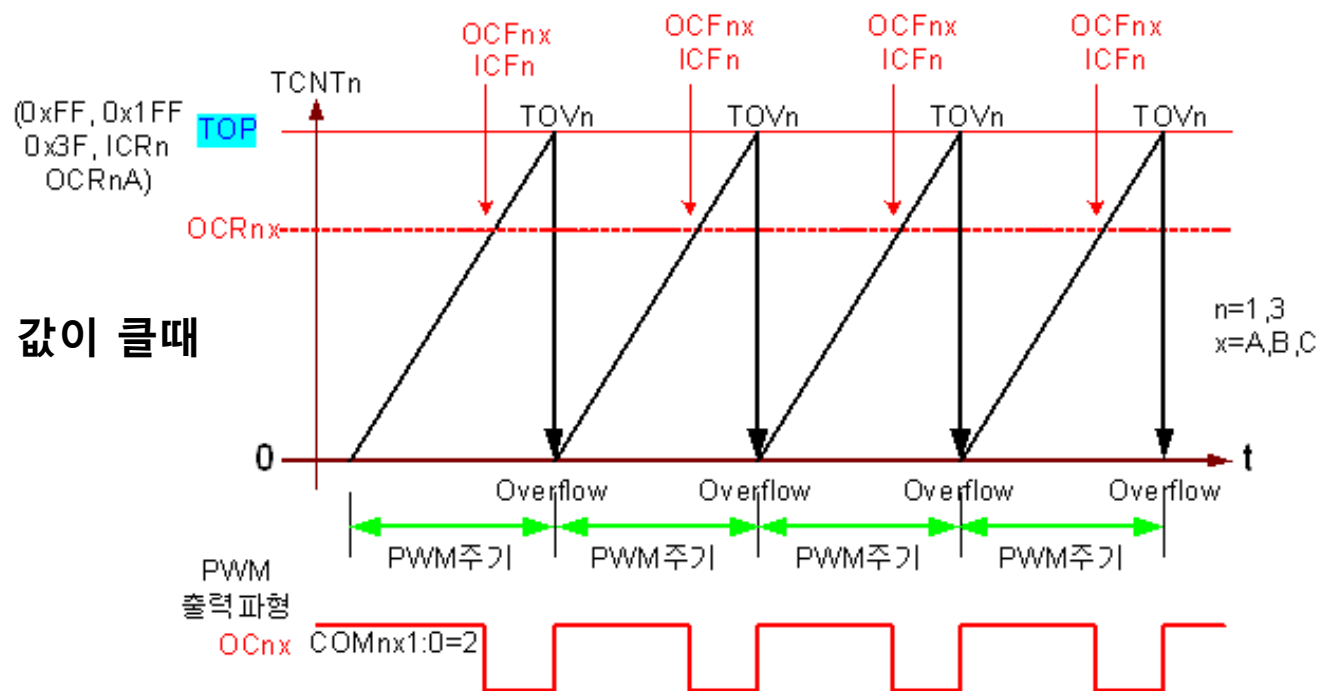
- Fast PWM(Fast Pulse Width Modulation) mode
  - Mode 5,6,7,14,15 동작 mode
  - TCNTn 값은 0에서 top까지 증가하다 다시 0부터 카운트하는 동작 반복
    - : top 값은 0x00FF(8bits), 0x01FF(9bits), 0x03FF(10bits)로 ICRn 값과 OCRnA 지정
    - : 비교 출력모드 TCCRn에서 [COMnx 1:0 = 2]에서  
TCNTn값은 OCRnx와 비교되어 일치하면  
OCnx(OC3A,OC3B,OC3C, OC1A, OC1B,OC1C) 핀을 통해 0이 출력되고,  
TCNTn값이 0이 되면 1이 출력
    - : 반전 비교 TCCRn에서 출력모드 [COMnx 1:0 = 3]에서  
TCNTn값은 OCRnx와 비교되어 일치하면  
OCnx(OC3A,OC3B,OC3C, OC1A, OC1B,OC1C) 핀을 통해 1이 출력되고,  
TCNTn값이 1이 되면 0이 출력

# 타이머/카운터n의

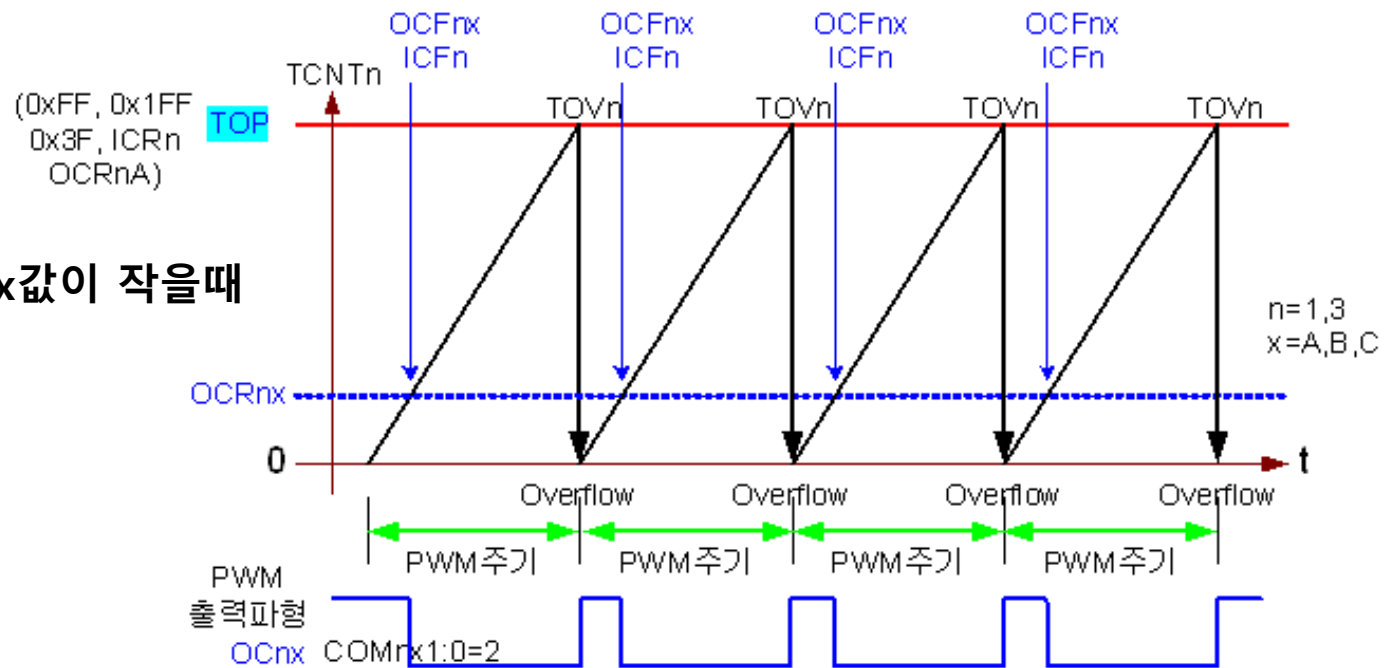
## Fast PWM모드

(n=1,3 X=A,B,C)

OCRnx 값이 클때



OCRnx 값이 작을때



### 16bits Timer / Counter 1,3 동작

- Fast PWM(Fast Pulse Width Modulation) mode

- PWM 분해능

: OCRnx 값에 따라 PWM 파형의 “H” 펄스 폭을 가변 시킴

- OCn 출력 파형 주기

: OCnx(OC3A,OC3B,OC3C, OC1A, OC1B,OC1C) 핀을 출력으로 설정 되어 있어야 출력

$$f_{OC_{nx}PWM} = \frac{f_{clk(I/O)}}{N \cdot (1 + TOP)}$$

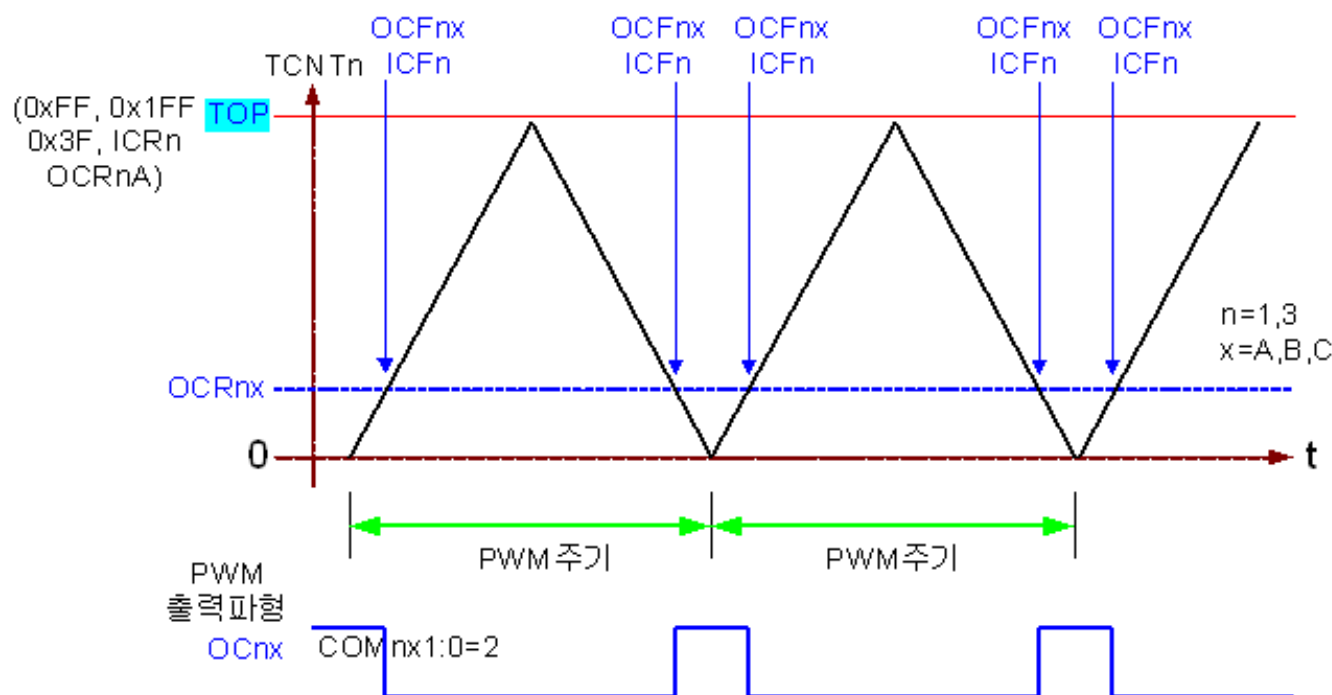
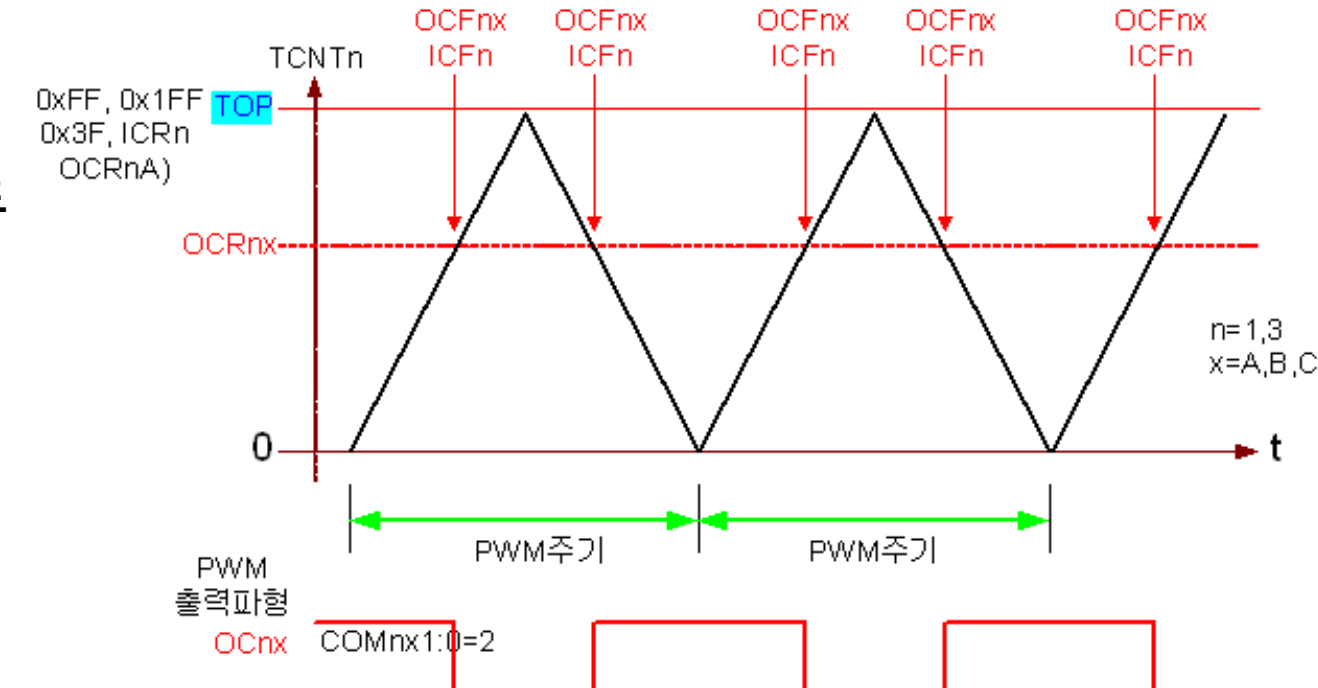
N : 분주 비, fclkI/O : 시스템 clock 주파수(16MHz)

### 16bits Timer / Counter 1,3 동작

- PC(Phase Correct) PWM mode
  - 높은 분해능의 PWM 파형을 발생하는데 유용
  - TCNTn 값은 0에서 top까지 **증가하였다가 0으로 감소하는 동작 반복**  
: top 값은 0x00FF(8bits), 0x01FF(9bits), 0x03FF(10bits)로 ICRn 값과 OCRnA 지정
  - 비반전 비교 출력모드에서 업카운트 중에  
TCNTn과 출력 비교 레지스터 OCRnx의 값이 일치하면  
OCnx핀을 통해 0이 출력되고, 다운 카운트 중에 일치하면 1이 출력
  - OCRnx 설정값에 따라 High(또는 Low) 부분의 펄스폭이 가변되어 PWM 파형 출력
  - Fast PWM 모드에 비해 2배의 분해 능을 가지는 반면에, 주파수는 1/2이 됨
  - Ocnx핀을 통해 출력되는 파형 주파수는

$$f_{OC_{nx}PWM} = \frac{f_{clk(I/O)}}{2 \cdot N \cdot TOP}$$

# 타이머/카운터n의 Phase Correct PWM모드 (n=1,3 x=A,B,C)



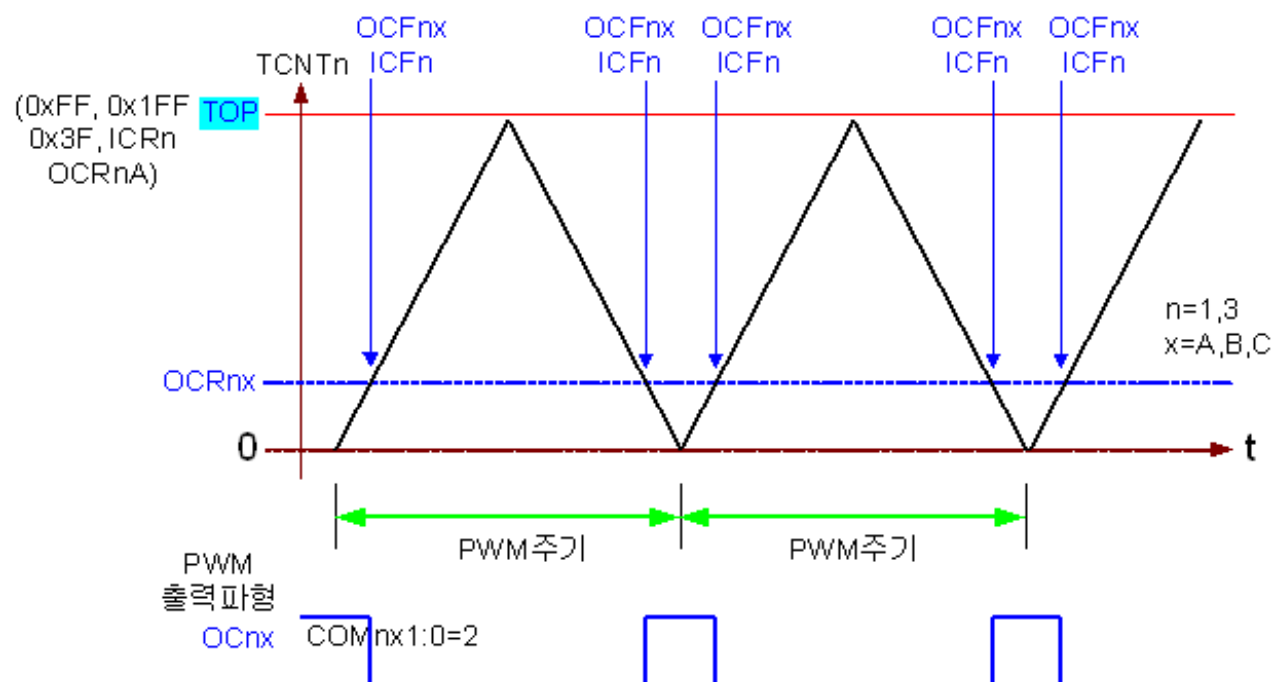
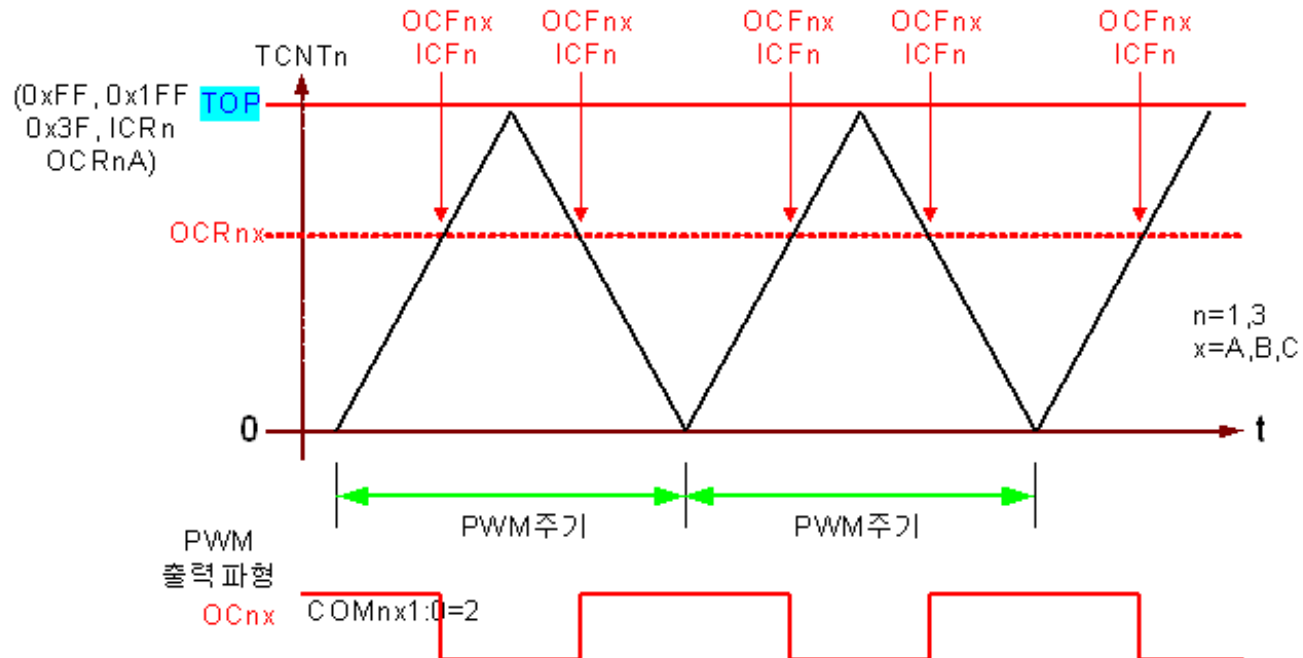
## 16bits Timer / Counter 1,3 동작

- PFC(Phase and Frequency Correct) PWM mode
- 높은 분해능의 PWM 파형을 발생하는데 유용
- PC PWM모드와 동작은 같지만

출력 비교 레지스터 OCRnx의 값이 변하는 시점이 다르다

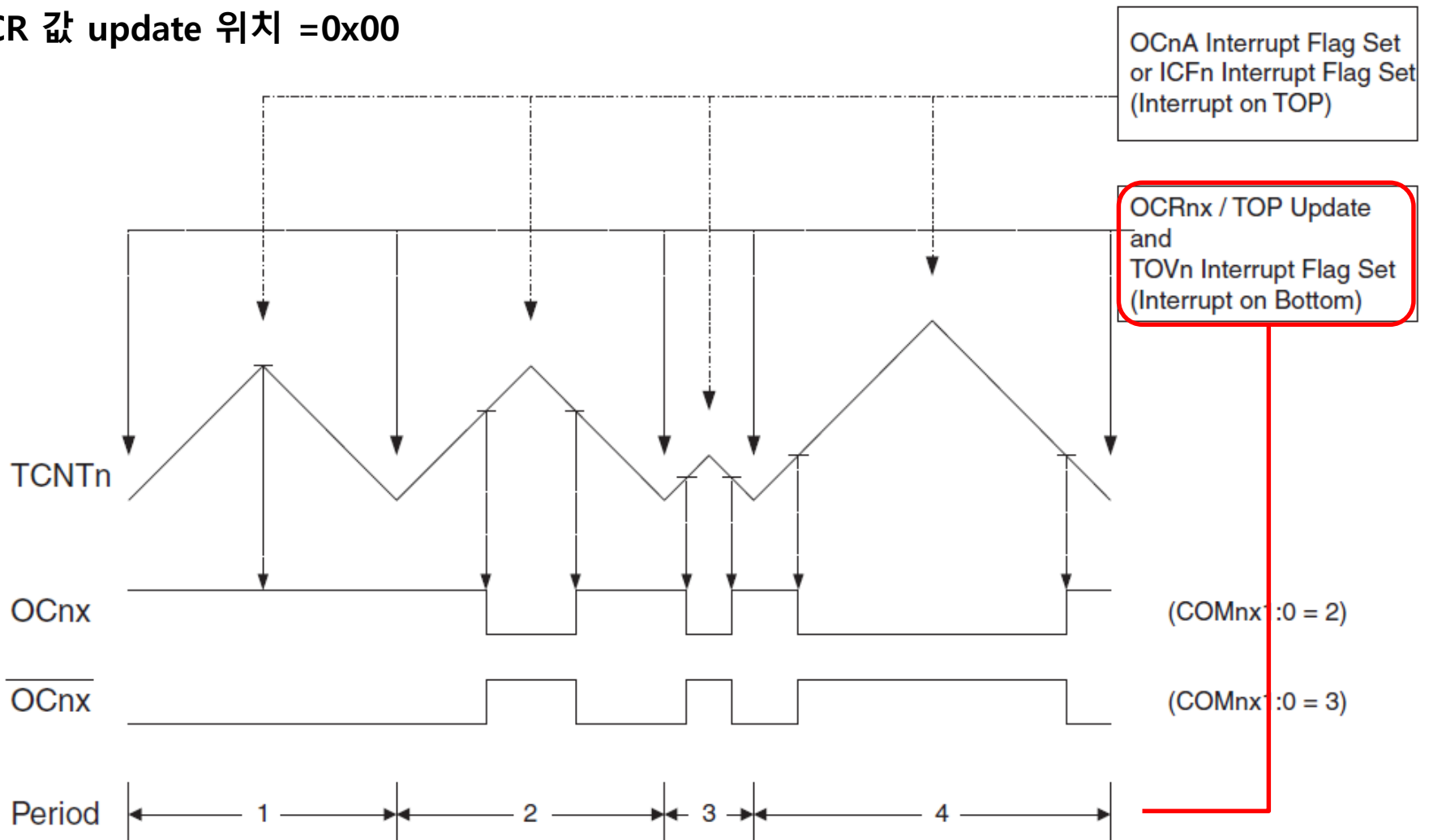
모드	WGM n3	WGM n2	WGM n1	WGM n0	동작모드	TOP값	OCRnx의 업데이트 시점	TOVn 플래그set 시점
1	0	0	0	1	Phase Correct PWM(8비트)	0x00FF	TOP	0x0000
2	0	0	1	0	Phase Correct PWM(9비트)	0x01FF	TOP	0x0000
3	0	0	1	1	Phase Correct PWM(10비트)	0x03FF	TOP	0x0000
10	1	0	1	0	Phase Correct PWM	ICRn	TOP	0x0000
11	1	0	1	1	Phase Correct PWM	OCRnA	TOP	0x0000
8	1	0	0	0	Phase and Frequency Correct PWM	ICRn	0x00	0x0000
9	1	0	0	1	Phase and Frequency Correct PWM	OCRnA	0x00	0x0000

# 타이머/카운터n의 Phase and Frequency Correct PWM모드 (n=1,3 x=A,B,C)





## OCR 값 update 위치 = 0x00



### 16bits Timer / Counter register

- 타이머/카운터 레지스터(TCNT1, 3 Timer/Counter Register)

: 타이머/카운터1, 3가 카운트하고 있는 입력 펄스 개수를 기록하는 16비트 레지스터

: TCNT1H, TCNT1L, TCNT3H, TCNT3L

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	TCNT3[15:8]								TCNT3H
	TCNT3[7:0]								TCNT3L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### 16bits Timer / Counter register

- 타이머/카운터 레지스터(TCNT1, 3 Timer/Counter Register)

TCNT1이나 TCNT3에 값을 쓰거나 읽을 때  
다음과 같은 주의 사항 필요

- (1) 어셈블리어로 프로그램을 작성할 때는  
8비트 단위로 2회에 걸쳐

읽기 동작에서는 하위 바이트부터 먼저 읽고,  
쓸 때는 상위 바이트부터 쓰기 동작 실행

- (2) C로 프로그램을 작성 할 때는  
16비트 레지스터명 TCNT1(TCNT3)을 이용하여  
한 스텝으로 프로그램 작성

위와 같은 규칙은  
OCRnA, OCRnB, OCRnC, ICRn에  
대해서도 동일하게 적용

```
; Set TCNTn to 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNTnH,r17
out TCNTnL,r16
; Read TCNTn into r17:r16
in r16,TCNTnL
in r17,TCNTnH
```

```
unsigned int i;

...

/* Set TCNTn to 0x01FF */
TCNTn = 0x1FF;

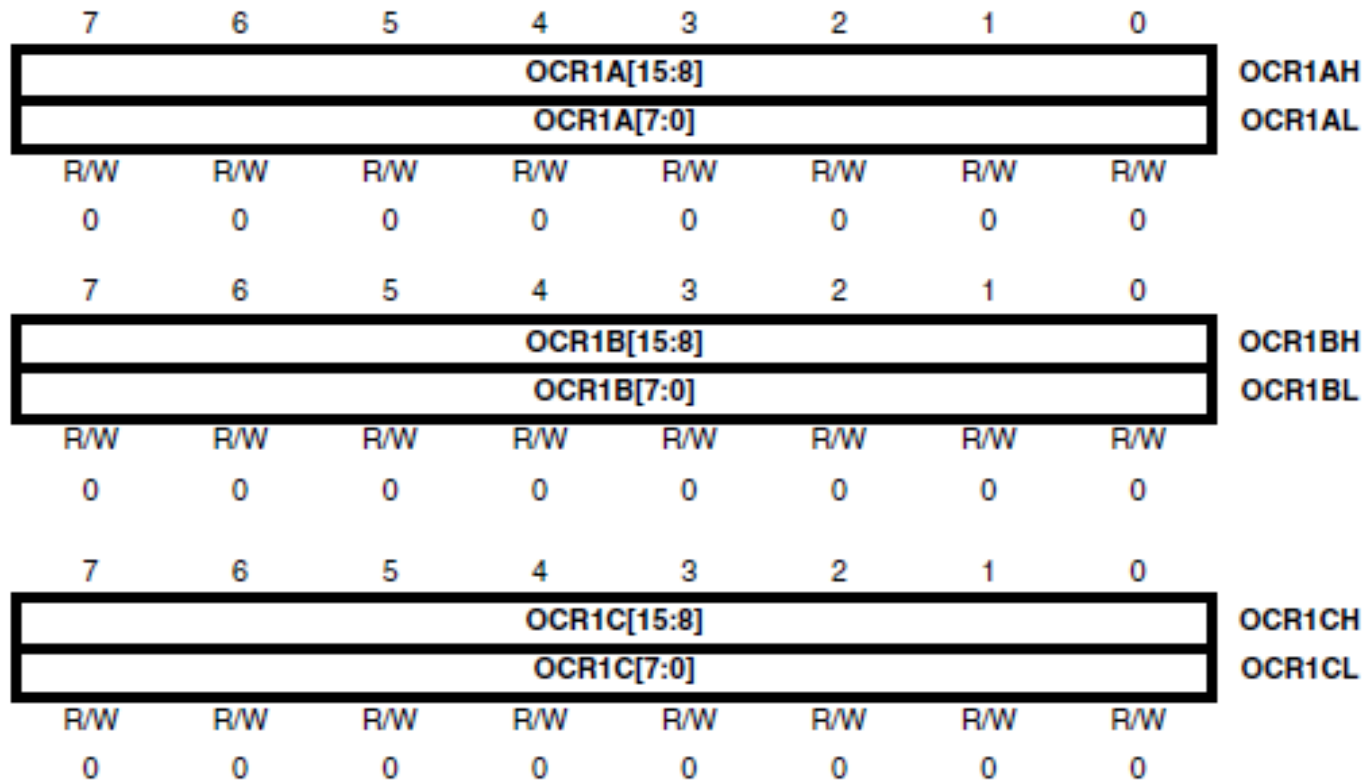
/* Read TCNTn into i */
i = TCNTn;
```

### 16bits Timer / Counter register

- OCR1x, OCR3x (Output Compare Register 1, 3)

: TCNTn의 값과 비교하는 16비트 출력 비교 레지스터

: OCRnA, OCRnB, OCRnC와 TCNTn이 같을 때 OCnA, OCnB, OCnC 핀으로 파형 출력



## 16bits Timer / Counter register

- OCR3x (Output Compare Register 3)

7	6	5	4	3	2	1	0	
OCR3A[15:8]								OCR3AH
OCR3A[7:0]								OCR3AL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	
7	6	5	4	3	2	1	0	
OCR3B[15:8]								OCR3BH
OCR3B[7:0]								OCR3BL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	
7	6	5	4	3	2	1	0	
OCR3C[15:8]								OCR3CH
OCR3C[7:0]								OCR3CL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

### 16bits Timer / Counter register

- ICR1, ICR3 (Input Capture Register 1, 3)

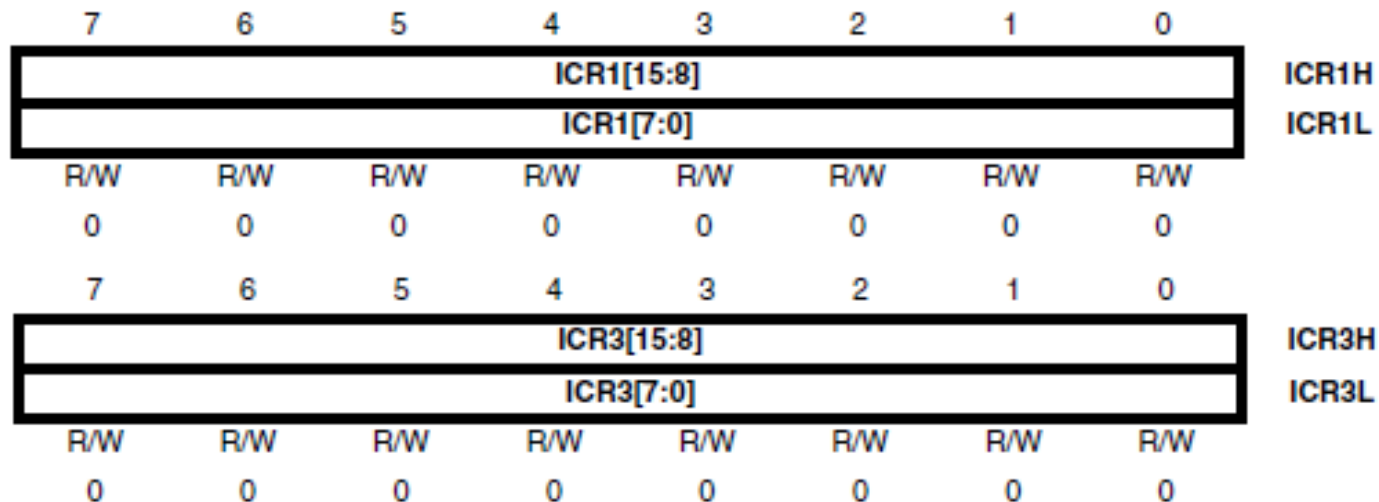
: 입력 capture 핀 ICP1, ICP3에 이벤트가 발생할 때마다

TCNTn값으로 갱신되는 16비트 레지스터

: 타이머/카운터 1과 3의 동작모드에 따라 TOP 값을 정의하는데도 이용

8비트 레지스터 ICRnH와 ICRnL로 각각 구성

: 레지스터들에 값을 쓰거나 읽는 프로그램을 할 때 TCNTn의 경우와 같이 주의 필요



### 16bits Timer / Counter register

- 타이머/카운터 제어 레지스터(TCCR1A, TCCR3A : Timer/Counter Control Register A)

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	TCCR3A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 1, Bit 0

: Waveform Generation Mode(WGM11, WGM10, WGM31, WGM30)

: 타이머/카운터의 동작모드 설정

- 타이머/카운터 제어 레지스터(TCCR1A, TCCR3A : Timer/Counter0 Control Register)

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



## 16bits Timer / Counter register

- 타이머/카운터 제어 레지스터(TCCR1A, TCCR3A : Timer/Counter Control Register A)

7	6	5	4	3	2	1	0	
COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	
7	6	5	4	3	2	1	0	
COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	TCCR3A
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Bit 7 ~ Bit 1

: OCnx 핀의 출력 모드 선택 비트(n=1,3 x=A,B,C)

: 타이머/카운터 1과 3에서 비교 매치 동작에 따른 출력 핀의 동작 결정

COMnx1	COMnx0	Normal, CTC 동작모드에서 OCnx핀의 출력 동작 (n=1,3 x=A,B,C)
0	0	일반 I/O 포트로 사용 (OCnx핀 차단)
0	1	비교 매치에서 OCnx 출력 (CTC mode)
1	0	OCnx 출력 clear (low 출력) <b>비교 출력</b>
1	1	OCnx 출력 set (high 출력) <b>반전 출력</b>

## 16bits Timer / Counter register

- 타이머/카운터 제어 레지스터(TCCR1A, TCCR3A : Timer/Counter Control Register A)

COMnx1	COMnx0	Fast PWM 동작모드에서 OCnx핀의 출력동작
0	0	일반 I/O 포트 사용 (OCnx핀 차단)
0	1	모드 15인 경우에만 비교 매치에서 <b>OCnA출력</b> , <b>OCnB핀</b> 은 차단된다. 그밖에 모드에서는 일반 I/O포트로 사용 (OCnA/OCnB 출력 차단)
1	0	비교매치에서 OCnx 출력 clear, TOP에서 OCnx 출력 set(=1) <b>비교 출력</b>
1	1	비교매치에서 OCnx 출력 set(=1), TOP에서 OCnx 출력 clear <b>반전 출력</b>

COMnx1	COMnx0	Phase( and Frequency) Correct PWM 동작모드에서 OCnx핀의 출력동작
0	0	일반 I/O 포트 사용(OCnx핀 차단)
0	1	모드9,11인 경우에만 비교 매치에서 <b>OCnA출력</b> , <b>OCnB핀</b> 은 차단된다. 그 밖의 모드에서는 일반 I/O포트로 사용(OCnx 출력 차단)
1	0	업 카운트 중의 비교매치에서 OCnX 출력 clear 다운 카운트 중의 비교매치에서 OCnx 출력 set(=1) <b>비교 출력</b>
1	1	업 카운트 중의 비교매치에서 OCnx 출력 set(=1) 다운 카운트 중의 비교매치에서 OCnx 출력 clear <b>반전 출력</b>

## 16bits Timer / Counter register

- 타이머/카운터 제어 레지스터(TCCR1B, TCCR3B : Timer/Counter Control Register B)

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### Bit 7

: ICNCn (**Input Capture Noise Canceler**) 입력 캡처 n 노이즈 제거기

- 이 비트가 1이면, 입력 capture noise 제거기가 동작하여

ICn (n=1, 3)핀으로 입력되는 신호 필터링 처리

- 이 필터에는 출력이 변화하는데 있어

같은 입력 신호 값이 4clock 동안 지속되는 것이 요구되며,

이에 따라 입력 capture는 4clock 만큼 지연되어 동작

### 16bits Timer / Counter register

- 타이머/카운터 제어 레지스터(TCCR1B, TCCR3B : Timer/Counter Control Register B)

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

#### Bit 6

: ICESn (Input Capture Edge Select) 입력 캡처 에지 선택

- 이 비트가 **0이면**, Input Capture 핀 ICn에 입력되는 신호의 **falling edge**에서  
타이머/카운터n의 값이 입력캡처 레지스터 ICRn에 전달
- 이 비트가 **1이면**, ICn에 입력되는 신호의 **rising edge**에서  
타이머/카운터n의 값이 입력캡처 레지스터 ICRn에 전달
- ICRn에 값이 저장되면, **Input Capture Interrupt Flag ICFn = 1** 되어 인터럽트 발생
- 레지스터 ICRn이 타이머/카운터1의 최대값(TOP)을 저장하는 레지스터로 사용하는 동작모드에서는 ICn핀은 차단되고 Input Capture 기능은 사용할 수 없음

## 16bits Timer / Counter register

- 타이머/카운터 제어 레지스터(TCCR1B, TCCR3B : Timer/Counter Control Register B)

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### Bit 4~3

: TCCRnA 레지스터의 WGMn1, WGMn0비트와 함께 동작모드 설정

### Bit 2~0

: clock source

CSn2	CSn1	CSn0	Clock 선택 ( $clk_{I/O}$ : system clock)
0	0	0	타이머/카운터n 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	Tn핀에 입력되는 외부 clock(falling edge 동작)
1	1	1	Tn핀에 입력되는 외부 clock(rising edge 동작)

### 16bits Timer / Counter register

- 타이머/카운터 제어 레지스터(TCCR1C, TCCR3C : Timer/Counter Control Register C)

Bit	7	6	5	4	3	2	1	0	
	FOC1A	FOC1B	FOC1C	—	—	—	—	—	TCCR1C
Read/Write	W	W	W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	FOC3A	FOC3B	FOC3C	—	—	—	—	—	TCCR3C
Read/Write	W	W	W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

#### Bit 7~5

: FOC<sub>n</sub>x (Force Output Compare) (n=1,3 x=A, B, C)

- PWM 모드가 아닌 경우에만 유효하며,

이 비트가 1이면 비교 출력단자 O<sub>cn</sub>x를 통해 COM<sub>n</sub>x1, COM<sub>n</sub>x0 비트에 의해 설정된 값이 즉시 출력

## 16bits Timer / Counter register

- TIMSK (Timer/Counter Interrupt Mask Register)

: 타이머/카운터의 인터럽트의 enable / disable 설정

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

: [Bit 5] TICIE1 : 타이머/카운터1 Input Capture 인터럽트 enable 비트

- 이 비트와 SREG의 I비트=1이면 타이머/카운터1 Input Capture 인터럽트 enable
- IC1 핀에서 capture 트리거(trigger) 이벤트가 발생하면 인터럽트 실행

: [Bit 4] OCIE1A : 타이머/카운터1 Output Compare A Match 인터럽트 enable 비트

- 이 비트와 SREG의 I비트=1이면 타이머/카운터1의 Compare A Match 인터럽트 enable
- 타이머/카운터1의 비교A 매치가 발생하면 인터럽트 실행

: [Bit 3] OCIE1B : 타이머/카운터1 Output Compare B Match 인터럽트 enable 비트

: [Bit 2] TOIE1 : 타이머/카운터1 Overflow 인터럽트 enable 비트

- 이 비트와 SREG의 I비트=1이면 타이머/카운터1의 Overflow 인터럽트 enable
- 타이머/카운터 Overflow 가 발생하면 인터럽트 실행

# 16bits Timer / Counter register

- ETIMSK (Extended Timer/Counter Interrupt Mask Register)

: 확장 타이머/카운터의 인터럽트의 enable / disable 설정

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

: [Bit 5] TICIE3 : 타이머/카운터3 입력 Capture 인터럽트 enable 비트

- 이 비트와 SREG의 I비트가 1이면 타이머/카운터3 입력 Capture 인터럽트 enable
- IC3 핀에서 Capture 트리거(trigger) 이벤트가 발생하면 인터럽트 실행

: [Bit 4] OCIE3A : 타이머/카운터3 출력 비교 A 매치 인터럽트 enable 비트

- 이 비트와 SREG의 I비트가 1이면 타이머/카운터3의 비교 A매치 인터럽트 enable
- 타이머/카운터3의 비교A 매치가 발생하면 인터럽트 실행

: [Bit 3] OCIE3B : 타이머/카운터3 출력 비교 B 매치 인터럽트 enable 비트

- 이 비트와 SREG의 I비트가 1이면 타이머/카운터3의 비교 B매치 인터럽트 enable
- 타이머/카운터3의 비교B 매치가 발생하면 인터럽트 실행



## 16bits Timer / Counter register

- ETIMSK (Extended Timer/Counter Interrupt Mask Register)

: 확장 타이머/카운터의 인터럽트의 enable / disable 설정

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

: [Bit 2] TOIE3 : 타이머/카운터3 Overflow 인터럽트 enable 비트

- 이 비트와 SREG의 I비트가 1이면 **타이머/카운터3**의 Overflow 인터럽트 enable
- 타이머/카운터3의 Overflow 가 발생하면 인터럽트 실행

: [Bit 1] OCIE3C : 타이머/카운터3 출력 비교 C 매치 인터럽트 enable 비트

- 이 비트와 SREG의 I비트가 1이면 **타이머/카운터3**의 비교**C** match 인터럽트 enable
- 타이머/카운터3의 비교C match 가 발생하면 인터럽트 실행

: [Bit 0] OCIE1C : 타이머/카운터1 출력 비교 C 매치 인터럽트 enable 비트

- 이 비트와 SREG의 I비트가 1이면 **타이머/카운터1**의 비교**C** match 인터럽트 enable
- 타이머/카운터1의 비교C match 가 발생하면 인터럽트 실행

## 16bits Timer / Counter register

- TIFR (Timer/Counter Interrupt Flag Register)

: 타이머/카운터의 **인터럽트 발생 여부를 기록**

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

: [Bit 5] ICF1 : 타이머/카운터1 **Input Capture Flag** Bit

- IC1핀에 **Capture 이벤트가 발생했을 때 set(=1) 되며, Input Capture 인터럽트 발생**
- ICR1 레지스터가 TOP 값으로 사용되는 동작모드에서는  
TCNT1의 값이 TOP이 될 때 set (=1)
- **SREG의 I, TIMSK의 TICIE1과 ICF1 비트가 set** 되면 Input Capture 인터럽트 실행
- 이 비트는 대응되는 인터럽트 처리 루틴이 실행되면 자동적으로 clear (=0)

: [Bit 4] OCF1A : Output Compare Flag Bit 1A

- **TCNT1과 출력 비교 레지스터 1A인 OCR1A가 일치했을 경우 set (=1)**
- **SREG의 I, TIMSK의 OCIE1A과 OCF1A 비트가 set** 되면 비교 A매치 인터럽트 실행
- OCF1A는 대응되는 인터럽트 처리 루틴을 실행할 경우에 하드웨어적으로 자동 clear 되거나 플래그에 1을 쓸 경우 clear

# 16bits Timer / Counter register

- TIFR (Timer/Counter Interrupt Flag Register)

: 타이머/카운터의 **인터럽트 발생 여부를 기록**

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

: [Bit 3] OCF1B : Output Compare Flag Bit 1B

- TCNT1과 출력 비교 레지스터 1B인 OCR1B가 일치했을 경우 **set(=1)**
- SREG의 I, TIMSK의 OCIE1B와 OCF1B 비트가 **set(=1)** 되면

타이머/카운터1 비교 B 매치 인터럽트 실행

- OCF1B는 대응되는 인터럽트 처리 루틴을 실행할 경우에 하드웨어적으로 자동 clear 되거나 플래그에 1을 쓸 경우 clear

: [Bit 2] TOV1 : 타이머/카운터1 Overflow 플래그 비트

- 타이머/카운터1에서 Overflow 가 발생했을 경우 **set(=1)**
- SREG의 I, TIMSK의 TOIE1과 TOV1 비트가 **set(=1)** 되면

타이머/카운터1 Overflow 인터럽트 실행

- TOV1은 대응되는 인터럽트 처리 루틴을 실행할 경우에 하드웨어적으로 자동 clear 되거나, TOV1에 1을 쓰면 clear
- Phase Correct PWM모드에서 이 비트는 타이머/카운터1의 카운터 값이 0x0000에서 카운터 방향이 바뀔 때 set

## 16bits Timer / Counter register

- ETIFR (Extended Timer/Counter Interrupt Flag Register)

: 타이머/카운터의 **인터럽트 발생 여부를 기록**

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

: [Bit 5] ICF3 : 타이머/카운터3 입력 캡처 플래그 비트

- IC3핀에 캡처 이벤트가 발생했을 때 set 되며, 입력 캡처 인터럽트 발생
- ICR3 레지스터가 TOP값으로 사용되는 동작모드에서는  
TCNT3의 값이 TOP값이 될 때 set
- SREG의 I, ETIMSK의 TICIE3와 ICF3 비트가 set 되면  
타이머/카운터3 캡처 인터럽트 실행
- 이 비트는 대응되는 인터럽트 처리 루틴이 실행되면 자동적으로 clear

: [Bit 4] OCF3A : 출력 비교 플래그 비트 3A

- TCNT3과 출력 비교 레지스터 3A인 OCR3A가 일치했을 경우 set
- SREG의 I비트, ETIMSK의 OCIE비트와 OCF3A가 set 되면  
타이머/카운터3 비교 A 매치 인터럽트 실행
- 이 비트는 대응되는 인터럽트가 실행되면  
하드웨어적으로 자동 clear 되거나 플래그에 1을 쓰면 clear

# 16bits Timer / Counter register

- ETIFR (Extended Timer/Counter Interrupt Flag Register)

: 타이머/카운터의 **인터럽트 발생 여부를 기록**

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

: [Bit 3] OCF3B : 출력 비교 플래그 비트 3B

- TCNT3과 출력 비교 레지스터 3B인 OCR3B가 일치했을 경우 set
- SREG의 I비트, ETIMSK의 OCIE3B비트와 OCF3B가 set 되면

타이머/카운터3 비교 B매치 인터럽트 실행

- 이 비트는 대응되는 인터럽트 처리 루틴을 실행할 경우에  
하드웨어적으로 자동 clear 되거나 플래그에 1을 쓸 경우 clear

: [Bit 2] TOV3 : 타이머/카운터3 오버플로우 플래그 비트

- 타이머/카운터3에서 오버플로우가 발생했을 경우 set
- SREG의 , ETIMSK의 TOIE3과 TOV3비트가 set 되면

타이머/카운터3 오버플로우 인터럽트 실행

- TOV3은 대응되는 인터럽트 처리 루틴을 실행할 경우에  
하드웨어적으로 자동 clear 되거나, TOV3에 1을 쓰면 clear
- Phase Correct PWM모드에서 이 비트는  
타이머/카운터3의 카운터 값이 0x0000에서 카운터 방향이 바뀔 때 set

## 16bits Timer / Counter register

- ETIFR (Extended Timer/Counter Interrupt Flag Register)

: 타이머/카운터의 **인터럽트 발생 여부를 기록**

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

: [Bit 1] OCF3C : 출력 비교 플래그 3C 비트

- TCNT3과 출력 비교 레지스터 3C인 OCR3C가 일치했을 경우 set
- SREG의 I, ETIMSK의 OCIE3C와 OCF3C 비트가 set 되었을 때  
타이머/카운터3 비교 C 매치 인터럽트 실행
- OCF3C는 대응되는 인터럽트 처리 루틴을 실행할 경우에  
하드웨어적으로 자동 clear 되거나 플래그에 1을 쓸 경우 clear

: [Bit 0] OCF1C : 출력 비교 플래그 1C 비트

- TCNT1과 출력 비교 레지스터 1C인 OCR1C가 일치했을 경우 set
- SREG의 I, ETIMSK의 OCIE1C와 OCF1C 비트가 set 되었을 때  
타이머/카운터1 비교 C 매치 인터럽트 실행
- OCF1C는 대응되는 인터럽트 처리 루틴을 실행할 경우에  
하드웨어적으로 자동 clear 되거나 플래그에 1을 쓸 경우 clear

# 16bits Timer / Counter register

- SFIOR (Special Function IO Register)

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

: [Bit 7] TSM(Timer/Counter **Synchronization Mode**) : 타이머/카운터 동기 모드

- 모든 타이머/카운터를 동기화시키는 기능
- 1로 설정하면 PSR0 및 PSR321 비트 값이 그대로 유지
- PSR0과 PSR321이 모두 1인 경우  
대응하는 프리스케일러를 계속해서 reset 상태를 유지하게 해주어  
모든 타이머/카운터를 정지시켜 주는 역할을 함
- 타이머/카운터가 정지된 상태에서 타이머/카운터에 같은 값을 설정
- 이 비트를 0으로 하면 PSR0과 PSR321비트는 하드웨어적으로 clear 되어  
타이머/카운터는 동시에 카운팅을 시작하게 되어 동기를 맞출 수 있음

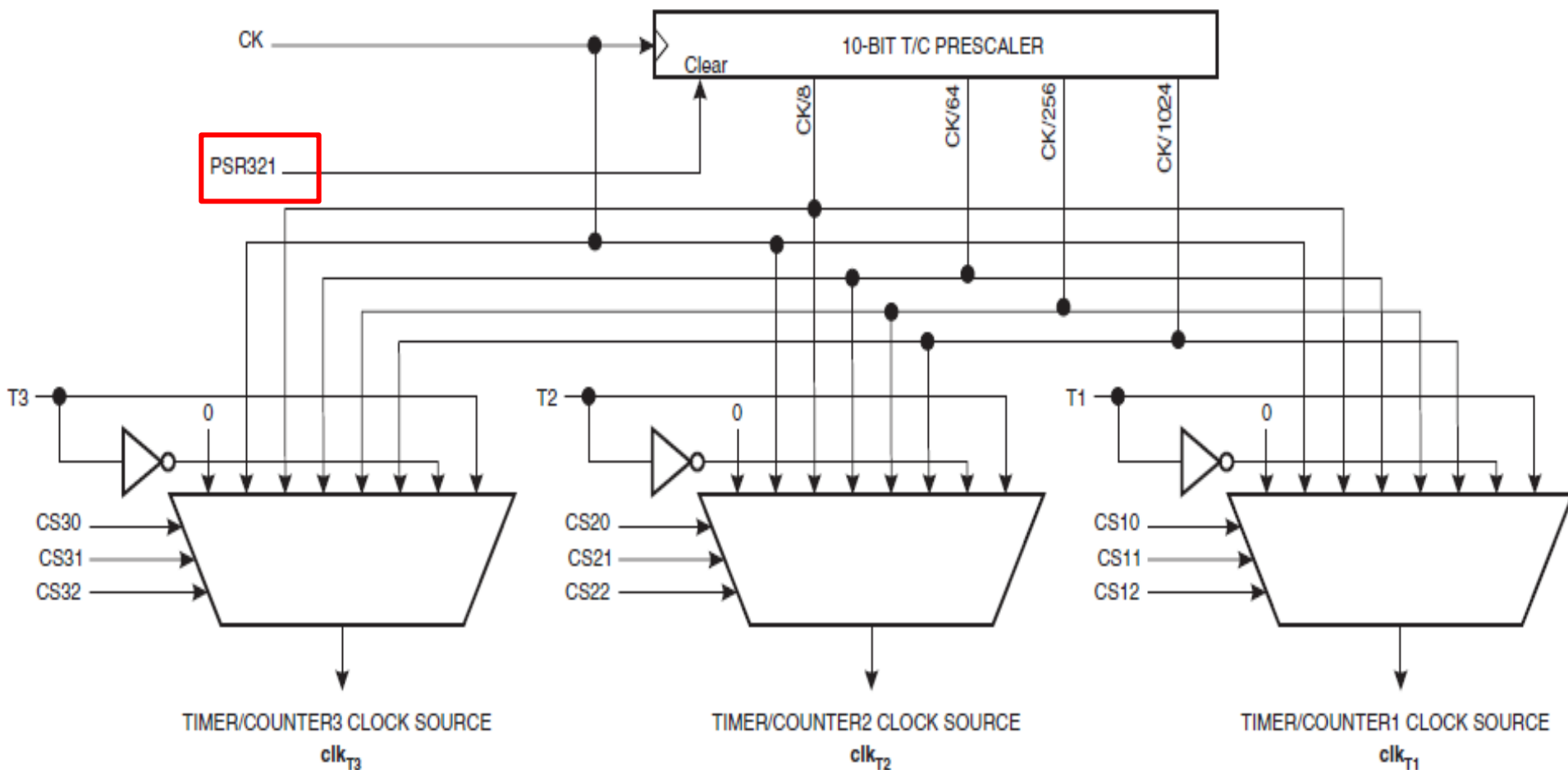
: [Bit 0] PSR321 : 타이머/카운터 1, 2, 3의 **pre-scaler reset**

- 1로 설정하면 타이머/카운터 1~3이 공통으로 사용하고 있는  
pre-scaler 를 reset시키며,  
TSM비트가 0이면 이 비트는 자동적으로 clear

# 16bits Timer / Counter register

- SFIOR (Special Function IO Register)

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





## [인터럽트 기능]

### 1. 타이머/카운터 인터럽트 마스크 레지스터 set

사용 레지스터 : **TIMSK(Timer / Counter 0, Timer / Counter 2, Timer / Counter 1)**  
**ETIMSK(Timer / Counter 1, Timer / Counter 3)**

### 2. 타이머/카운터 동작모드 및 분주비 설정

사용 레지스터 : **TCCR0, ASSR(Timer / Counter 0), TCCR2(Timer / Counter 2)**  
**TCCR1A/B/C(Timer / Counter 1), TCCR3A/B/C(Timer / Counter 3)**

### 3. 타이머/카운터 레지스터, 출력비교 레지스터 초기값 설정

사용 레지스터 : **TCNT0, TCNT1, TCNT2, TCNT3**  
**OCR0, OCR1A/B/C, OCR2, OCR3A/B/C, ICR1, ICR3**

➔ 동작모드에 따라 설정하는 레지스터가 다르므로 동작 모드 표 참조

### 4. 전역 인터럽트 인에이블 비트 I set

사용 레지스터 : **SREG(최상위 비트)**

➔ 가능하면 모든 초기 설정이 끝난 후 제일 마지막에 I 비트를 set 한다.

### 5. 인터럽트 함수 작성

## [PWM 파형 출력 기능]

1. PWM 출력 핀의 포트를 출력으로 설정

2. 타이머/카운터 동작모드 및 분주비 작성

사용 레지스터 : **TCCR0, ASSR(Timer / Counter 0), TCCR2(Timer / Counter 2)**  
**TCCR1A/B/C(Timer / Counter 1), TCCR3A/B/C(Timer / Counter 3)**

3. 타이머/카운터 레지스터, 출력비교 레지스터 초기값 설정

사용 레지스터 : **TCNT0, TCNT1, TCNT2, TCNT3**  
**OCR0, OCR1/A/B/C, OCR2, OCR3A/B/C,**  
**ICR1, ICR3**

➔ 동작모드에 따라 설정하는 레지스터가 다르므로 동작 모드 표 참조

<경우에 따라 타이머/카운터의 값을 reset하기 위해 인터럽트 기능을 사용할 때도 있음>

```
interrupt [인터럽트 소스명] void 함수명(void)
{
    인터럽트 서브루틴
}
```

[예] 타이머/카운터0 비교 매치

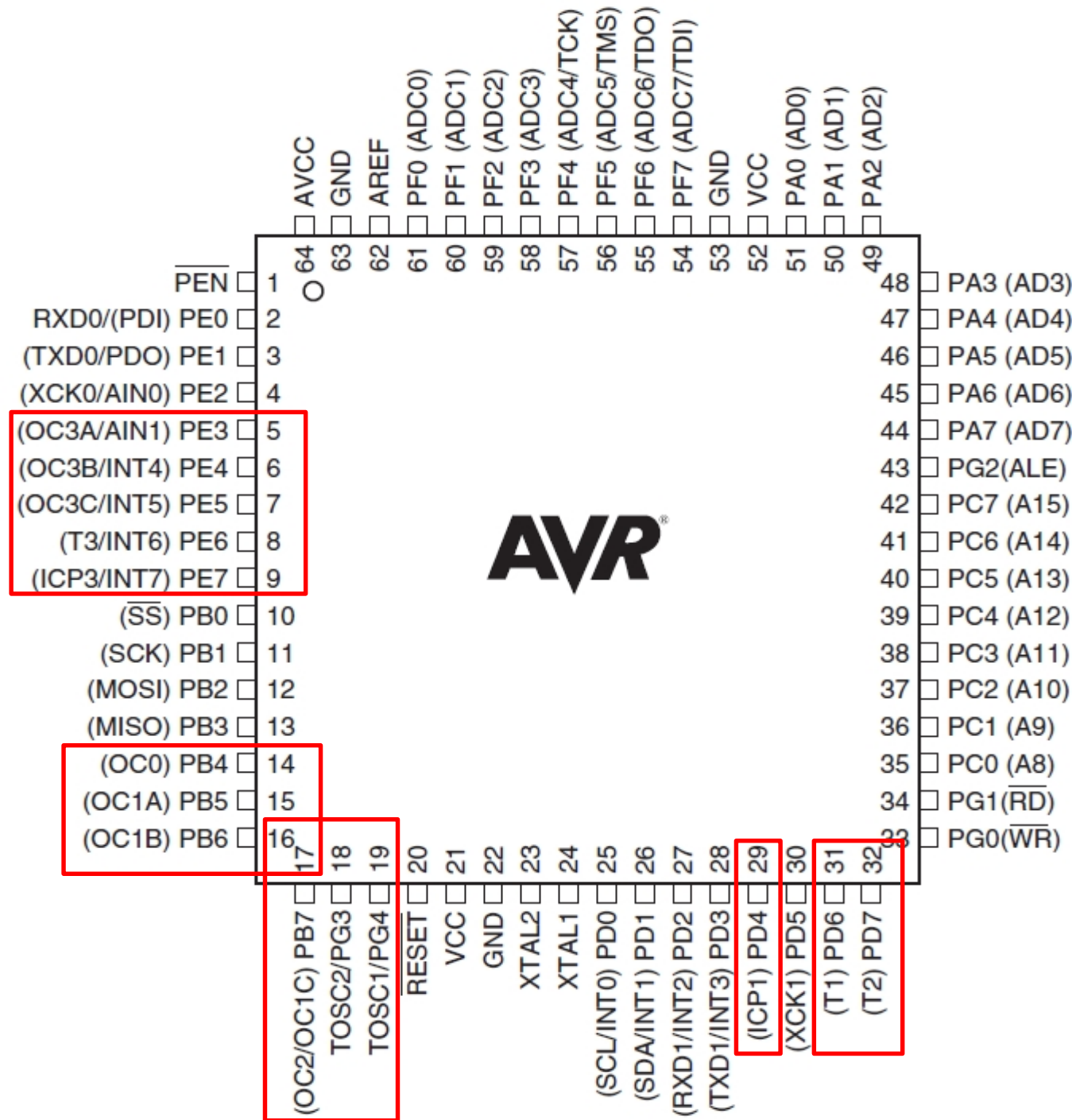
```
interrupt [TIM0_COMP] void tim0_comp(void)
{
    . . . .
}
```

[예] 타이머/카운터1 오버플로우 인터럽트

```
interrupt [TIM1_OVF] void tim1_ovf(void)
{
    . . . .
}
```

- 
1. Micro Processor 원리
  2. Atmel사의 8bit Micro-controller
  3. KUT0128 Evaluation Board 기능과 특징
  4. IO Port 제어
  5. External Interrupt 제어
  6. Timer / Counter 제어
  7. UART 제어
  8. AD Converter 제어
  9. Comparator 제어
  10. EEPROM 제어 (IIC, Parallel method)
  11. SPI 제어

## 6. Timer / Counter



## 6. Timer / Counter

Strictly Private and Confidential

	타이머/카운터0	타이머/카운터2	타이머/카운터1	타이머/카운터3
구조	8비트	8비트	16비트	16비트
카운터 경우, 외부 clock 입력	TOSC1	T2	T1	T3
Pre-scaler	1, 8, 32, 64, 128, 256, 1024	1, 8, 64, 256, 1024	1, 8, 64, 256, 1024	1, 8, 64, 256, 1024
Register	TCNT0, TCCR0, OCR0, ASSR, TIMSK, TIFR, SFior	TCNT2, TCCR2, OCR2, TIMSK, TIFR, SFior	TCNT1, TCCR1A, TCCR1B, TCCR1C, OCR1A, OCR1B, OCR1C, ICR1, TIMSK, ETIMSK, TIFR, ETIFR, SFior	TCNT3, TCCR3A, TCCR3B, TCCR3C, OCR3A, OCR3B, OCR3C, ICR3, TIMSK, ETIMSK, TIFR, ETIFR, SFior
Input pin	TOSC1, TOSC2	T2	T1, IC1	T3, IC3
Output pin	OC0	OC2	OC1A, OC1B, OC1C	OC3A, OC3B, OC3C
Interrupt	Overflow, 출력 비교 match	Overflow, 출력 비교 match	Overflow, 출력 비교 match A/B/C, 입력 capture	Overflow, 출력 비교 match A/B/C, 입력 capture
특징	RTC기능, 타이머/카운터 모두 Pre-scaler	—	Capture 기능	Capture 기능

## [인터럽트 기능]

### 1. 타이머/카운터 인터럽트 마스크 레지스터 set

사용 레지스터 : **TIMSK(Timer / Counter 0, Timer / Counter 2, Timer / Counter 1)**  
**ETIMSK(Timer / Counter 1, Timer / Counter 3)**

### 2. 타이머/카운터 동작모드 및 분주비 설정

사용 레지스터 : **TCCR0, ASSR(Timer / Counter 0), TCCR2(Timer / Counter 2)**  
**TCCR1A/B/C(Timer / Counter 1), TCCR3A/B/C(Timer / Counter 3)**

### 3. 타이머/카운터 레지스터, 출력비교 레지스터 초기값 설정

사용 레지스터 : **TCNT0, TCNT1, TCNT2, TCNT3**  
**OCR0, OCR1A/B/C, OCR2, OCR3A/B/C, ICR1, ICR3**

➔ 동작모드에 따라 설정하는 레지스터가 다르므로 동작 모드 표 참조

### 4. 전역 인터럽트 인에이블 비트 I set

사용 레지스터 : **SREG(최상위 비트)**

➔ 가능하면 모든 초기 설정이 끝난 후 제일 마지막에 I 비트를 set 한다.

### 5. 인터럽트 함수 작성

## [PWM 파형 출력 기능]

1. PWM 출력 핀의 포트를 출력으로 설정

2. 타이머/카운터 동작모드 및 분주비 작성

사용 레지스터 : **TCCR0, ASSR(Timer / Counter 0), TCCR2(Timer / Counter 2)**  
**TCCR1A/B/C(Timer / Counter 1), TCCR3A/B/C(Timer / Counter 3)**

3. 타이머/카운터 레지스터, 출력비교 레지스터 초기값 설정

사용 레지스터 : **TCNT0, TCNT1, TCNT2, TCNT3**  
**OCR0, OCR1A/B/C, OCR2, OCR3A/B/C,**  
**ICR1, ICR3**

➔ 동작모드에 따라 설정하는 레지스터가 다르므로 동작 모드 표 참조

<경우에 따라 타이머/카운터의 값을 reset하기 위해 인터럽트 기능을 사용할 때도 있음>



```
interrupt [인터럽트 소스명] void 함수명(void)
{
    인터럽트 서브루틴
}
```

[예] 타이머/카운터0 비교 매치

```
interrupt [TIM0_COMP] void tim0_comp(void)
{
    . . . .
}
```

[예] 타이머/카운터1 오버플로우 인터럽트

```
interrupt [TIM1_OVF] void tim1_ovf(void)
{
    . . . .
}
```

- CodeVisionAVR에서 제공하는 헤더 파일 <mega128.h>에서 정의된 소스명을 사용

```
#define TIM2_COMP      10
#define TIM2_OVF       11
#define TIM1_CAPT      12
#define TIM1_COMPA     13
#define TIM1_COMPB     14
#define TIM1_OVF       15
#define TIM0_COMP      16
#define TIM0_OVF       17
```

```
#define TIM1_COMPC     25
#define TIM3_CAPT      26
#define TIM3_COMPA     27
#define TIM3_COMPB     28
#define TIM3_COMPC     29
#define TIM3_OVF       30
```

- 함수명은 사용자가 임의로 지정

- Project 생성시,  
C-Compiler에서  
clock 8MHz를 16MHz로



Configure Project test4\_6.prj

Files C Compiler Before Build After Build

Code Generation Libraries Messages Globally #define Paths

Active Build Configuration: Debug

Chip: ATmega128A

Clock: 16 MHz

Memory Model: Small

Optimize for: Size

Optimization Level: Maximal

Program Type: Application

(s)printf Features: int\_width

(s)scanf Features: int\_width

RAM

Data Stack Size: 1024 bytes

Heap Size: 0 bytes

Internal RAM Size: 4096 bytes

External RAM Size: 0 bytes

☐ External RAM Wait State

Code Generation

Bit Variables Size: 16

☒ Promote char to int ☒ char is unsigned

☒ 8 bit enums Enh. Param: Mode 2

☒ Smart Register Allocation

☒ Automatic Global Register Allocation

☐ Store Global Constants in FLASH Memory

☐ Use an External Startup Initialization File

☒ Clear Global Variables at Program Startup

☐ Stack End Markers

File Output Formats: COF ROM HEXEEP

Preprocessor

☐ Create Preprocessor Output Files

☒ Include I/O Registers Bits Definitions

OK Cancel Help

## [예제 6-1] 타이머/카운터0을 이용한 LED제어 실습[일반 모드]

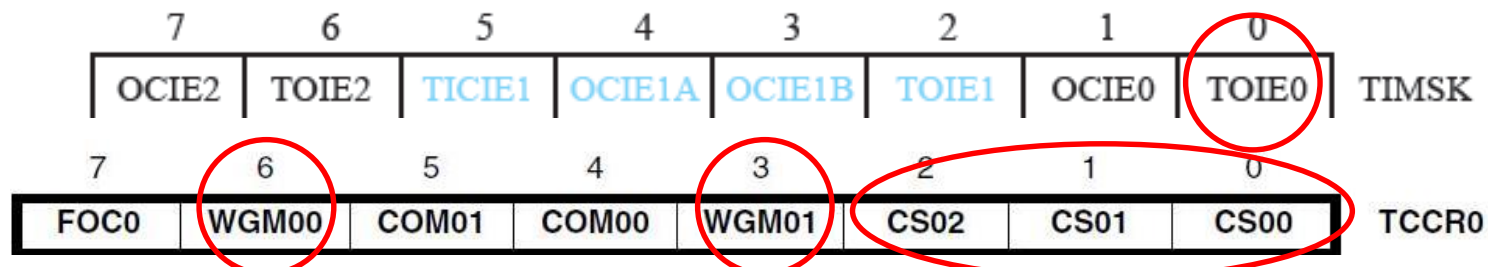
타이머/카운터0을 일반 모드로 사용하여

**Overflow 인터럽트가 발생할 때마다** 포트C에 연결된 LED를 1비트씩 쉬프트 하면서 ON시키는 프로그램을 작성하라.

타이머/카운터의 **인터럽트 주기는 최대가 되도록 TCNT0을 0으로** 하고  
Pre-scaler는 1024 분주로 한다.

Timer / Counter 0 : 일반 모드, **Overflow 인터럽트가 발생 (TIMSK, TOIE0 bit)**

1. Interrupt enable : 포트C에 연결된 LED를 1비트씩 쉬프트 하면서 ON
2. Timer control : 인터럽트 주기는 최대가 되도록 **TCNT0=0**
3. Data (timer 관련) : Pre-scaler (**TCCR0**) 는 **1024** 분주  
:  $1\text{clock} = 1/(16\text{MHz}/1024) = 64\mu\text{sec}$   
: 주기 =  $64\mu\text{sec} \times (256 - 0) = 16.384\text{ms}$



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

```
unsigned char led = 0xFE;
```

```
void main(void)
```

```
{
```

```
    DDRC = 0xFF;          // 포트 C 출력으로 설정
```

```
    PORTC = led;          // 포트 C에 초기값 출력
```

```
    TIMSK = 0x01;         // TOIE0 = 1(overflow interrupt enable)
```

```
    TCCR0 = 0x07;         // 일반모드, 1024분주
```

```
    TCNT0 = 0x00;         // 타이머/카운터0 레지스터 초기값
```

```
    SREG |= 0x80;         // 전역 인터럽트 enable 비트 I set
```

```
    while(1);
```

```
}
```

```
interrupt [TIM0_OVF] void timer_int0(void)
```

```
{
```

```
    TCNT0 = 0x0;
```

```
    led = led << 1;
```

```
    led = led | 0x01;
```

```
    if(led == 0xFF) led = 0xFE;
```

```
    PORTC = led;
```

```
}
```

교재의 REGISTER 확인하며 프로그램  
(179page ~)

TCNT0=0x9C : 156

주기 =  $64\mu\text{sec} \times (256 - 156) = 6.4\text{ms}$

주기 =  $64\mu\text{sec} \times (256 - 0) = 16.384\text{ms}$



// 초기값 재설정(0인 경우 생략가능)

// 1비트 쉬프트

// 최하위 비트 set

// 모두 off이면 초기값 재설정

// 포트 출력

## 8bits Timer / Counter register

- 타이머/카운터0 제어 레지스터(TCCR0, TCCR2 : Timer/Counter0 Control Register)

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

### Bit 7

: FOC0(Force Output Compare)

: 이 비트는 PWM모드가 아닌 경우에만 유효

: 1이면 비교 출력 OC0를 통해 COM01 : COM00 비트에 의해 설정된 값 즉시 출력

### Bit 6, Bit 3

: **WGM00, WGM01(Waveform Generation Mode)**

: 타이머/카운터0의 동작모드 설정

모드	WGM01[3]	WGM00[6]	동작 모드	TOP	OCR0 업데이트시점	TOV0 Flag set 시점
0	0	0	Normal	0xFF	설정 즉시	MAX
1	0	1	Phase Correct PWM	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	설정 즉시	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

## 8bits Timer / Counter register

- 타이머/카운터0 제어 레지스터(TCCR0, TCCR2 : Timer/Counter0 Control Register)

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 5, Bit 4

: COM01, COM00 (**Compare Match Output Mode**)

: 타이머/카운터0의 **출력단자 OC0핀의 동작 제어**

: OC0에 해당하는 데이터 방향 레지스터 DDRB의 4비트는 **출력이 되도록 설정**

COM01	COM00	OC0핀의 출력기능
0	0	일반 I/O 포트 동작 ( <b>OC0 차단</b> )
0	1	<b>비교 매치에서 OC0 출력</b>
1	0	비교 매치에서 OC0 clear(=0) <b>비교 출력 (PWM)</b>
1	1	비교 매치에서 OC0 set(=1) <b>반전 출력 (PWM)</b>

## 8bits Timer / Counter register

- 타이머/카운터0 제어 레지스터(TCCR0, TCCR2 : Timer/Counter0 Control Register)

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit [2] ~ [0]

: CS02 : CS00 (clock 선택)

: 타이머/카운터0의 분주 비(Pre-scaler) 를 결정

CS02	CS01	CS00	기 능
0	0	0	clock 입력 차단(타이머/카운터 정지)
0	0	1	$clk_{TOS} / 1$
0	1	0	$clk_{TOS} / 8$
0	1	1	$clk_{TOS} / 32$
1	0	0	$clk_{TOS} / 64$
1	0	1	$clk_{TOS} / 128$
1	1	0	$clk_{TOS} / 256$
1	1	1	$clk_{TOS} / 1024$



## [예제 6-2] 타이머/카운터0을 이용한 LED 제어 실험 (출력비교매치 인터럽트)

타이머/카운터0을 CTC모드로 이용하여 10ms마다 출력비교 매치 인터럽트를 발생시키고, 인터럽트가 발생할 때마다 포트 C에 연결된 LED를 1비트씩 쉬프트 하면서 ON시키는 프로그램을 작성하라.

Timer / Counter 0 : CTC 모드, Output compare match 인터럽트 (TIMSK, OCIE0 bit)

: 포트C에 연결된 LED를 1비트씩 쉬프트 하면서 ON

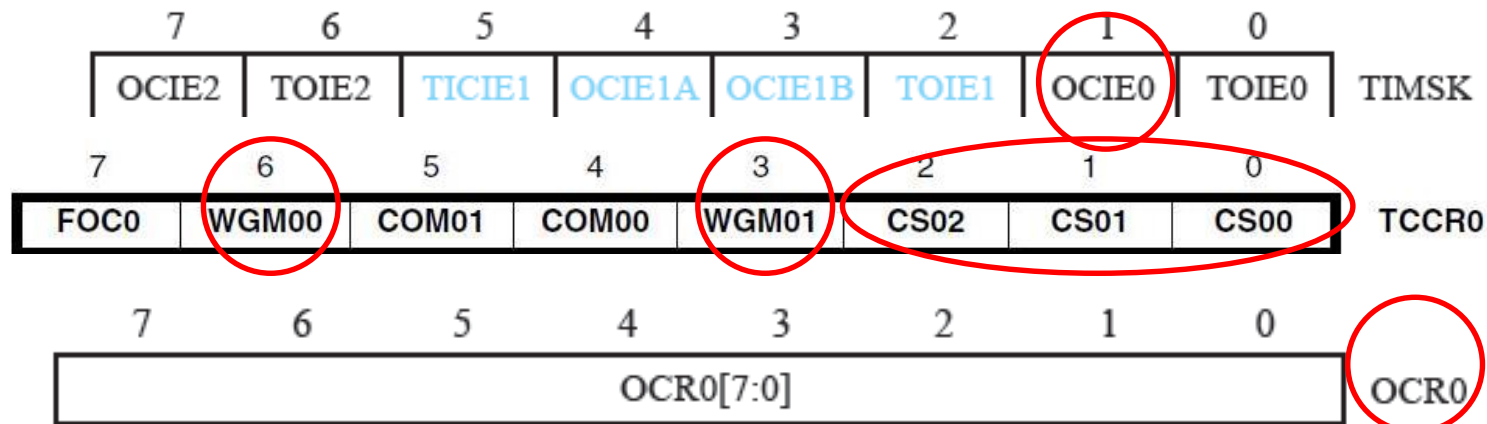
: 인터럽트 주기는  $TCNT0 = 0, OCR0 = 155$

1. Interrupt enable
2. Timer control
3. Data (timer 관련)

: Pre-scaler (TCCR0) 는 1024 분주

:  $1\text{clock} = 1/(16\text{MHz}/1024) = 64\text{ usec}$

: 주기 =  $64\text{ usec} \times (1+155) = 9.984\text{ms}$



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

```
unsigned char led = 0xFE;
```

```
void main(void)
```

```
{
```

```
    DDRC = 0xFF;      // 포트 C 출력으로 설정  
    PORTC = led;      // 포트 C에 초기값 출력
```

```
    TIMSK = 0x02;      // OCIE0 = 1(출력 비교 인터럽트 인에이블)
```

```
    TCCR0 = 0x0F;      // WGM01=1, CTC 모드, 1024분주
```

```
    OCR0 = 155;        // 출력 비교 레지스터값(9.98ms 주기)
```

```
    TCNT0 = 0x0;       // 타이머/카운터0 레지스터 초기값
```

```
    SREG = 0x80;       // 전역 인터럽트 인에이블 비트 I set
```

```
    while(1);
```

```
}
```

```
// 타이머/카운터0 출력비교(TCNT0 = OCR0 일때) 인터럽트 서비스 루틴
```

```
// 인터럽트 발생 주기  $1/16\mu s * 1024\text{분주} * (1 + 155) = 9.98\text{ms}$ 
```

```
interrupt [TIM0_COMP] void timer_comp0(void)
```

```
{
```

```
    led = led << 1;      // 1비트 쉬프트
```

```
    led = led | 0x01;    // 최하위 비트 set
```

```
    if(led == 0xFF) led = 0xFE; // 모두 off이면 초기값 재설정
```

```
    PORTC = led;        // 포트 출력
```

```
}
```

```

#include <mega128.h>

unsigned char led = 0xFE;

void main(void)
{
    DDRC = 0xFF;    // 포트 C 출력으로 설정
    PORTC = led;    // 포트 C에 초기값 출력

    TIMSK = 0x42;    // timer0 compare, timer2 overflow interrupt

    TCCR0 = 0x0F;    // WGM01=1, CTC 모드, 1024분주
    OCR0 = 155;      // 출력 비교 레지스터값(9.98ms 주기)
    TCNT0 = 0x0;     // 타이머/카운터0 레지스터 초기값

    TCCR2 = 0x05;    // overflow mode, 1024분주
    TCNT2 = 0x0;     // 타이머/카운터0 레지스터 초기값

    SREG = 0x80;    // 전역 인터럽트 인에이블 비트 I set

    while(1);
}

// 타이머/카운터0 출력비교(TCNT0 = OCR0 일때) 인터럽트 서비스 루틴
// 인터럽트 발생 주기 1/16us * 1024분주 * (1 + 155) = 9.98ms

interrupt [TIM0_COMP] void timer_comp0(void)
{
    led = led << 1;    // 1비트 쉬프트
    led = led | 0x01;   // 최하위 비트 set
    if(led == 0xFF) led = 0xFE; // 모두 off이면 초기값 재설정
    PORTC = led;        // 포트 출력
}

interrupt [TIM2_OVF] void timer_ovf2(void)
{
}

```

## [예제 6-3]타이머/카운터0을 이용 LED 제어 실험

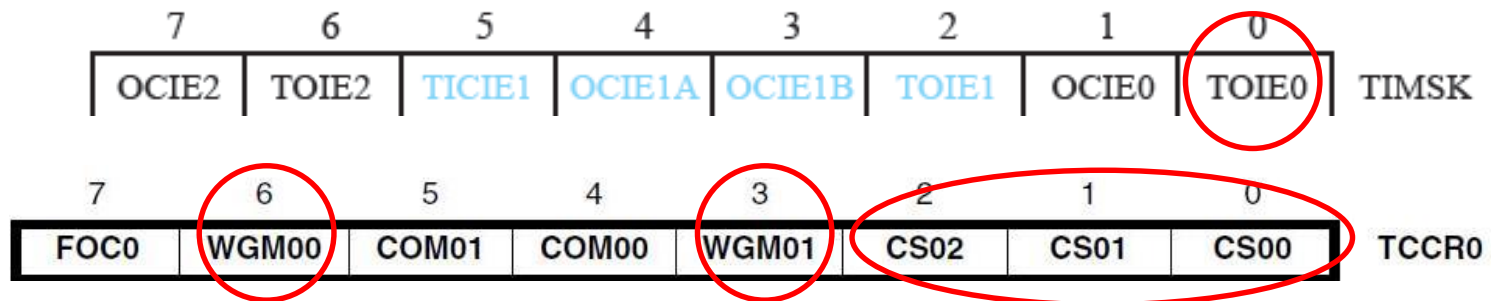
[일반모드 0.5초 간격]

타이머/카운터0의 **overflow 인터럽트**를 이용하여 **0.5초**마다 모든 포트 C에 연결된 LED를 ON/OFF 시키는 프로그램을 작성하라. X-TAL clock의 **1024분주**한 clock을 타이머/카운터의 클럭으로 이용한다.

(힌트 : 16MHz의 클럭을 사용하는 KUT-128\_COM보드에서 타이머/카운터0의 **가장 긴 overflow 인터럽트 주기**는  $1/16\mu s * 1024\text{분주} * (256-0) = 16.384\text{ms}$  가 된다.

인터럽트가 **31회** 걸리면 **508ms**되어 약 0.5초의 시간 주기를 얻을 수 있다.)

1. Interrupt enable
2. Timer control
3. Data  
(timer 관련)



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

```
unsigned char led = 0xFF; // LED 출력 초기값
```

```
unsigned char cnt = 0; // 타이머/카운터0 인터럽트 발생 횟수
```

```
void main(void)
```

```
{
```

```
    DDRC = 0xFF; // 포트 C 출력으로 설정
```

```
    PORTC = led; // 포트 C에 초기값 출력
```

```
    TIMSK = 0x01; // TOIE0 = 1
```

```
    TCCR0 = 0x07; // 프리스케일 = CK/1024
```

```
    TCNT0 = 0x00; // 타이머/카운터0 레지스터 초기값
```

```
    SREG = 0x80; // 전역 인터럽트 인에이블 비트 I set
```

```
    while(1);
```

```
}
```

```
// 1/16us x 1024 x 256 = 16.384ms
```

```
interrupt [TIM0_OVF] void timer_int0(void)
```

```
{
```

```
    TCNT0 = 0x0; // 초기값 재설정(0인 경우 생략가능)
```

```
    cnt++; // 인터럽트 횟수 +1
```

```
    if(cnt == 31){ // 16.384ms x 31 = 0.5sec
```

```
        led = led ^ 0xFF; // led값 반전
```

```
        PORTC = led; // 포트 출력
```

```
        cnt = 0; // 인터럽트 카운터 리셋
```

```
    }
```

```
}
```

# [예제 6-4] 타이머/카운터1을 이용한 LED제어 실험

## (모드 0: 일반 모드)



타이머/카운터1 레지스터의 값을 0x8000(32768)으로 초기 설정하고, 256분주 된 클럭을 타이머/카운터1의 클럭으로 이용하여 overflow 인터럽트가 발생할 때 마다 포트C에 연결된 LED를 쉬프트하면서 ON시키는 프로그램을 작성하라

1. Interrupt enable
2. Timer control
3. Data (timer 관련)

7	6	5	4	3	2	1	0	
OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
7	6	5	4	3	2	1	0	
COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
7	6	5	4	3	2	1	0	
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
7	6	5	4	3	2	1	0	
FOC1A	FOC1B	FOC1C	-	-	-	-	-	TCCR1C

1clock  
 = 1/(16MHz/256)  
 = 16usec

주기  
 = 16usec x (65536-32768)  
 = 524.288ms

TIMSK = 0x04 : TOIE1 = 1, overflow 인터럽트 enable  
 TCCR1A = 0x00 : 모드 0(일반모드)  
 TCCR1B = 0x04 : 프리스케일 = CK/256  
 TCCR1C = 0x0  
 TCNT1 = time : 타이머/카운터1 레지스터 초기값  
 SREG = 0x80 : 전역 인터럽트 enable 비트 I set

• 타이머/카운터 제어 레지스터(TCCR1A, TCCR3A : Timer/Counter0 Control Register)

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

```

#include <mega128.h>
unsigned char led = 0xFE;
unsigned int time = 0x8000;    // 32768

void main(void)
{
    DDRC = 0xFF;                // 포트 C 출력으로 설정
    PORTC = led;                // 포트 C에 초기값 출력

    TIMSK = 0x04;               // TOIE1 = 1, 오버플로우 인터럽트 인에이블

    TCCR1A = 0x00;              // 모드 0(일반모드)
    TCCR1B = 0x04;              // 프리스케일 = CK/256
    TCCR1C = 0x0;

    TCNT1 = time;               // 타이머/카운터1 레지스터 초기값 : 16 비트
    SREG = 0x80;                // 전역 인터럽트 인에이블 비트 I set

    while(1);
}

// (1/16)us * 256분주 * (65536 - 32768) = 524ms
interrupt [TIM1_OVF] void timer_int1(void)
{
    TCNT1 = time;
    led = led << 1;              // 1비트 쉬프트
    led = led | 0x01;            // 최하위 비트 set
    if(led == 0xFF) led = 0xFE;  // 모두 off이면 초기값 재설정
    PORTC = led;                // 포트 출력
}

```





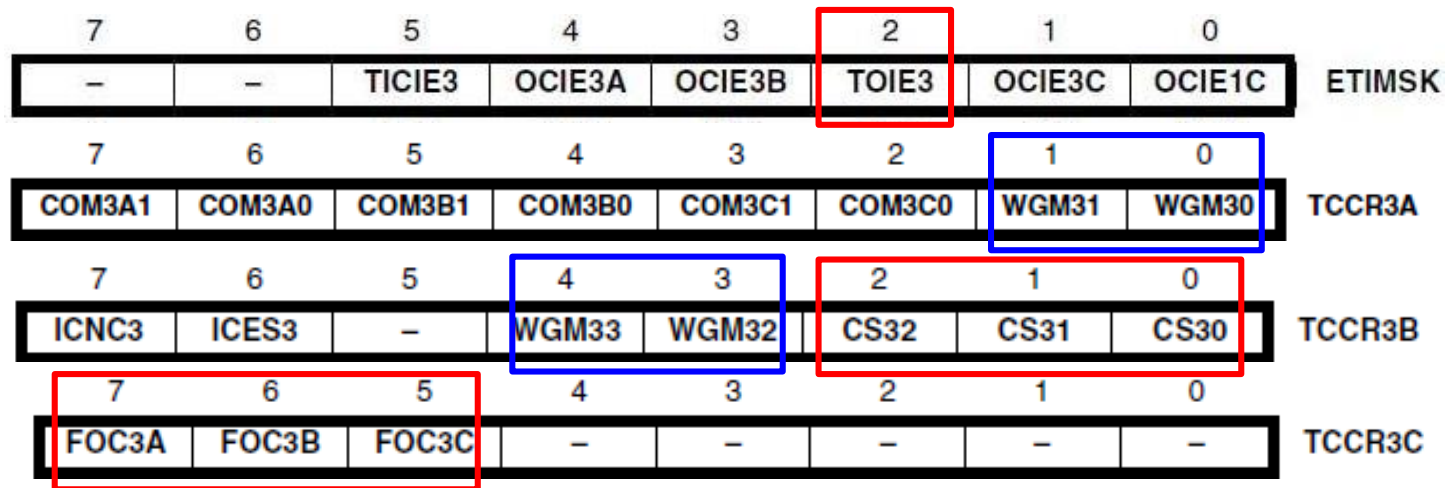
## [예제 6-5] 타이머/카운터3을 이용한 카운터 모드 실험 [T3핀 clock 입력]

T3에 연결된 스위치 SW3을 누를 때마다, 포트 C에 연결된 LED를 한 비트씩 쉬프트 하면서 ON시키는 프로그램을 작성하라.

Falling edge

즉, 타이머/카운터3을 카운터로 이용하여 SW3이 눌러질 때마다 타이머/카운터3에 Overflow 인터럽트가 발생하도록 하라. clock 소스를 외부 신호 T3(SW3) 이용

1. Interrupt enable
2. Timer control
3. Data (timer 관련)



ETIMSK = 0b00000100;

TCCR3A = 0x0;

TCCR3B = 0x06;

TCCR3C = 0x0;

TCNT3H = 0xFF;

TCNT3L = 0xFF;

// TOIE3 = 1

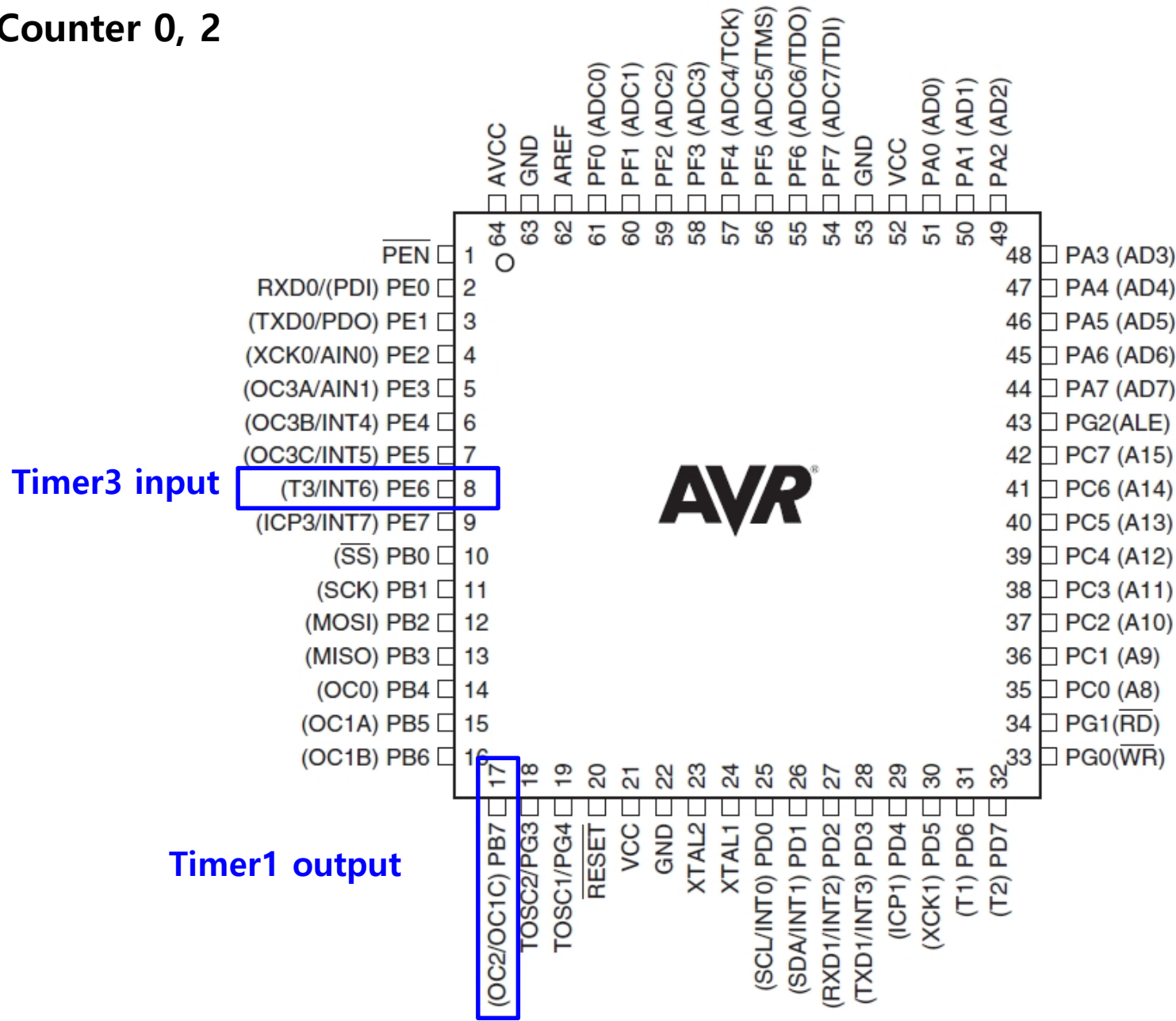
// 일반 MODE

// T3 falling edge에서 카운팅

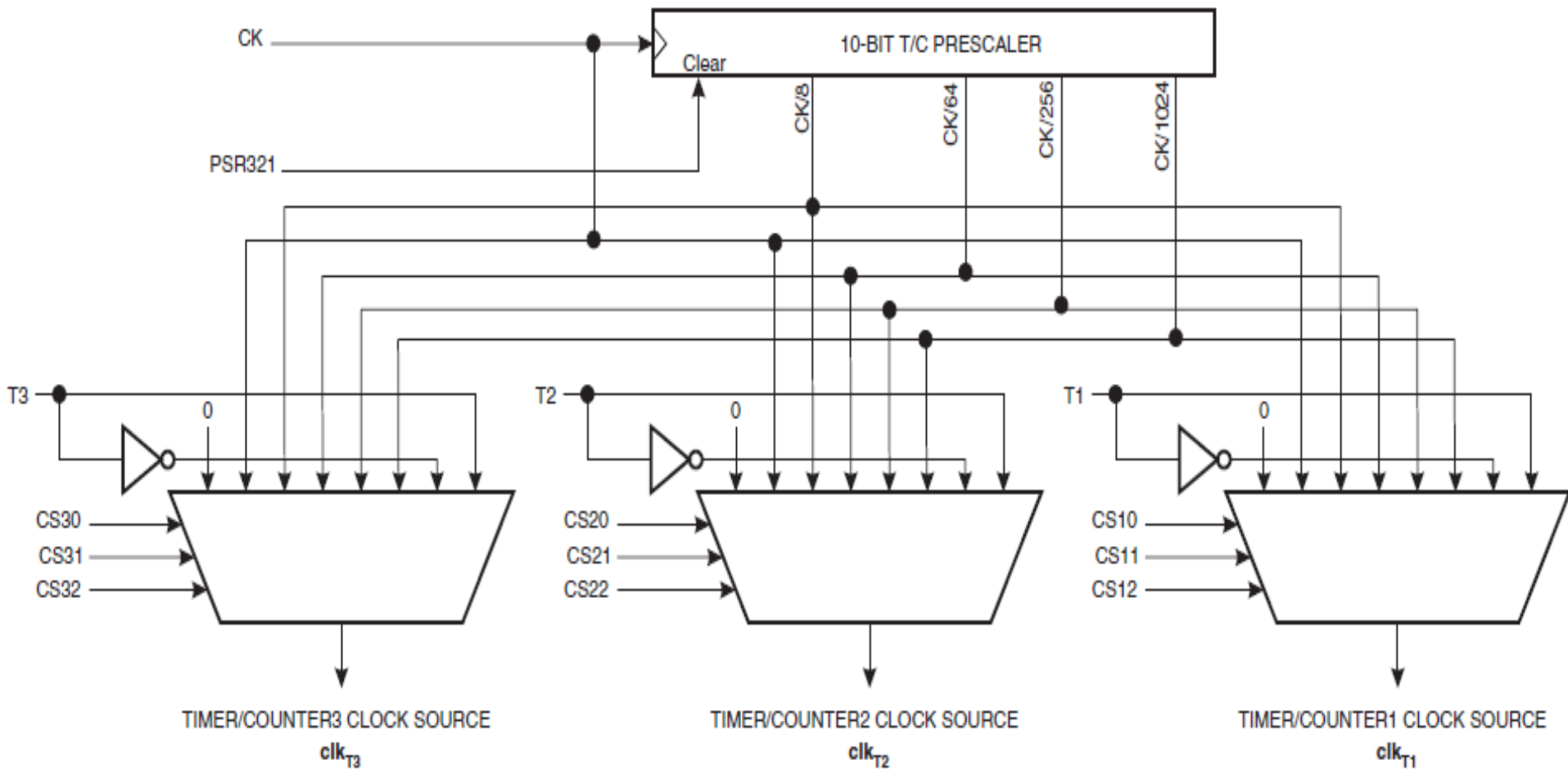
// TCNT3 초기값 설정 : 외부에서 클럭 1개

// 입력되면서 0이 되면서 인터럽트 발생

## 8bits Timer / Counter 0, 2



# [예제 6-5] 타이머/카운터3을 이용한 카운터 모드 실험 [T3핀 clock 입력]



Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

CSn2	CSn1	CSn0	Clock 선택 ( $clk_{I/O}$ : system clock)
0	0	0	타이머/카운터n 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	Tn핀에 입력되는 외부 clock(falling edge 동작)
1	1	1	Tn핀에 입력되는 외부 clock(rising edge 동작)

```

#include <mega128.h>

unsigned char led = 0xFE; // LED 출력 초기값

void main(void)
{
    DDRC = 0xFF;           // 포트 C 출력으로 설정
    PORTC = led;           // 포트 C에 초기값 출력

    ETIMSK = 0b00000100;   // TOIE3 = 1
    TCCR3A = 0x0;          // 일반 MODE
    TCCR3B = 0x06;         // T3 하강 에지에서 카운팅
    TCCR3C = 0x0;

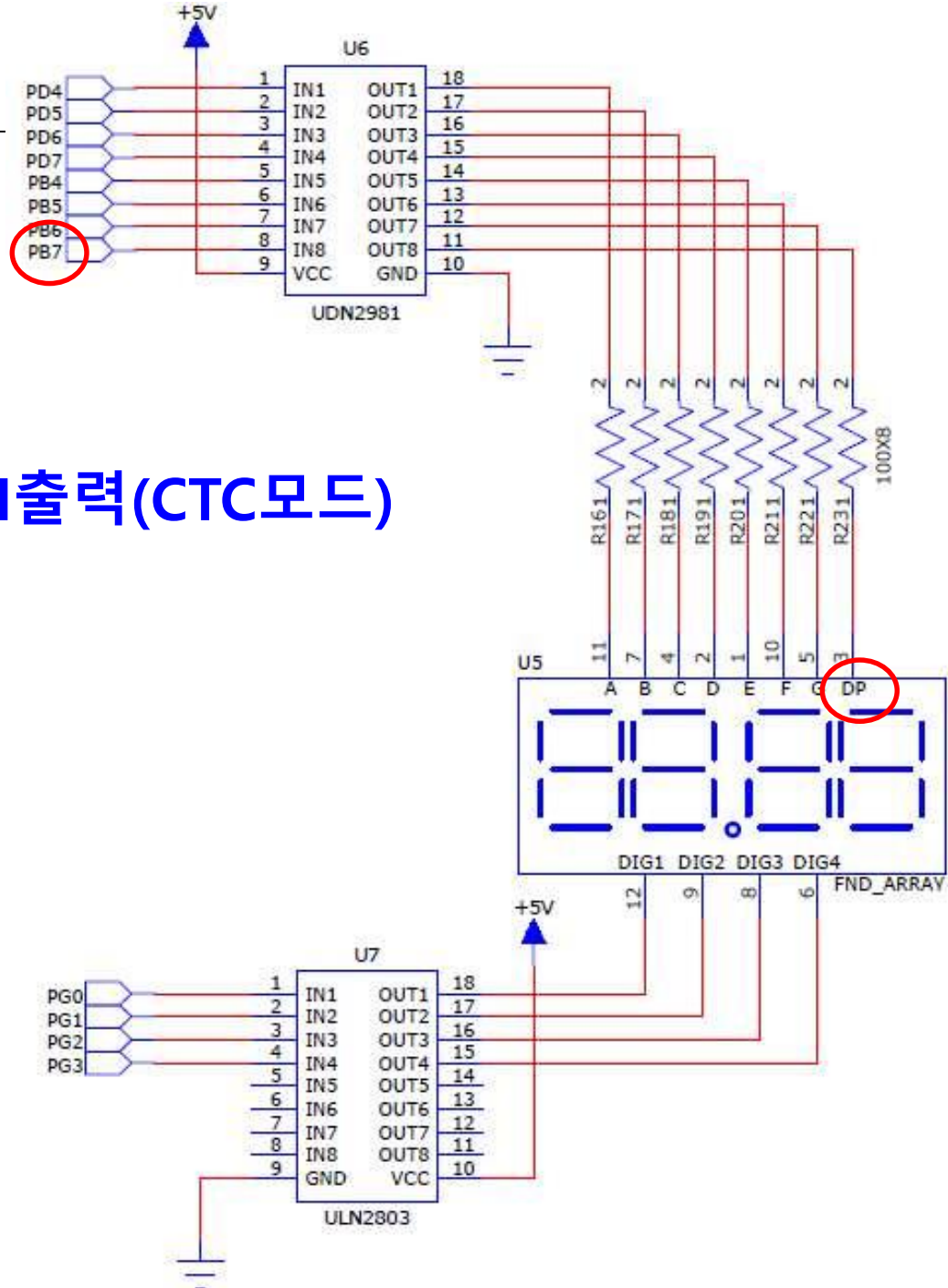
    TCNT3H = 0xFF;         // TCNT3 초기값 설정 : 외부에서 클럭 1개
    TCNT3L = 0xFF;         // 입력되면서 0이 되면서 인터럽트 발생
    SREG = 0x80;           // 전역 인터럽트 인에이블 비트 1 셋

    while(1);
}

interrupt [TIM3_OVF] void timer_int3(void)
{
    TCNT3H = 0xFF;         // 초기값 재설정
    TCNT3L = 0xFF;

    led = led << 1;        // 1비트 쉬프트
    led = led | 0x01;       // 최하위 비트 셋
    if(led == 0xFF) led = 0xFE; // 모두 off이면 초기값 재설정
    PORTC = led;           // 포트 출력
}

```



[예제 6-6] 타이머/카운터1을  
이용한 PWM출력(CTC모드)

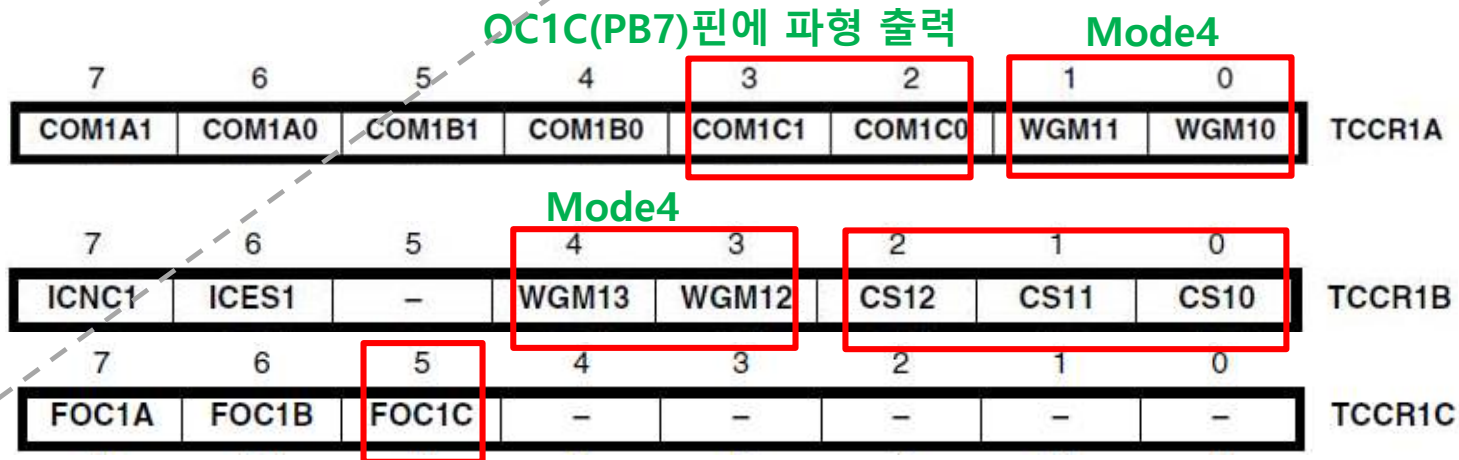
## [예제 6-6] 타이머/카운터1을 이용한 PWM출력(CTC모드)

타이머/카운터1을 모드4의 CTC모드로 동작시키고, OCR1C 레지스터와 비교매치에서 파형을 출력 OC1C(PB7)핀으로 자동 출력되는 프로그램을 작성하라

: 출력되는 파형의 주기는 1초와 500ms를 각각 6번, 12번 반복 출력.

☞ CTC모드에서 파형의 주기

$$\text{출력파형 주기} = 1/16\mu\text{s} * \text{분주비} * (1 + \text{OCR1A})$$



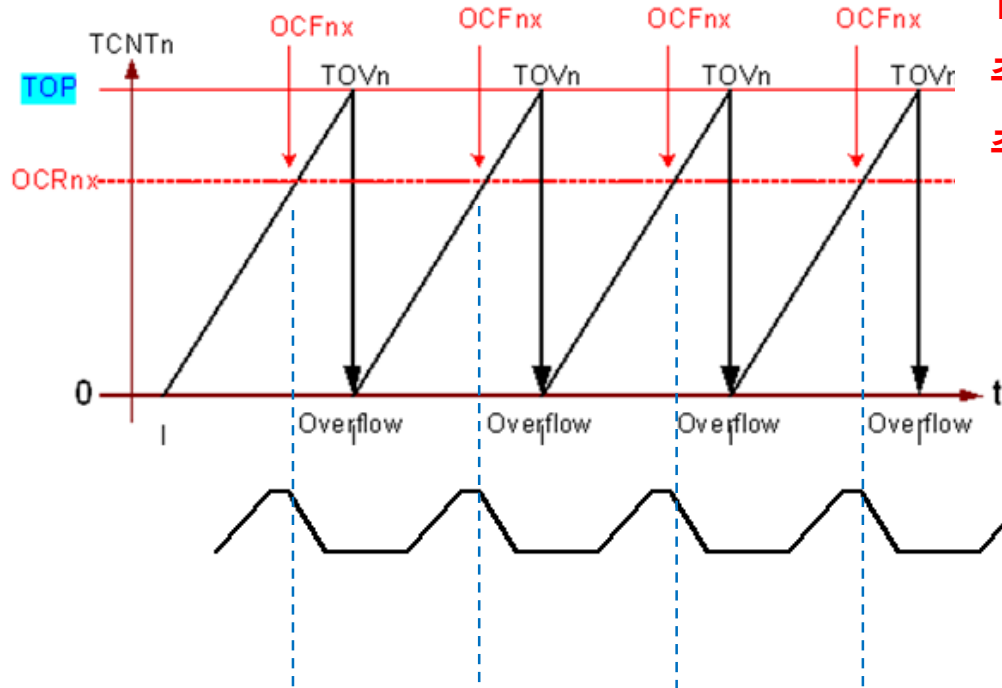
## [예제 6-6] 타이머/카운터1을 이용한 PWM출력(CTC모드)

: 출력되는 파형의 주기는 1초와 500ms를 각각 6초 동안 출력하는 것으로 한다.

☞ CTC모드에서 파형의 주기

$$\text{출력파형 주기} = 1/16\mu\text{s} * \text{분주비} * (1 + \text{OCR1A})$$

on, off 이므로  
=> 절반으로 on



$$1\text{clock} = 1/(16\text{MHz}/256) = 16\mu\text{sec}$$

$$\text{주기} = 16\mu\text{sec} \times (1 + 31249) = 500\text{ms}$$

$$\text{주기} = 16\mu\text{sec} \times (1 + 15624) = 250\text{ms}$$



## 16bits Timer / Counter register

- 타이머/카운터 제어 레지스터(TCCR1A, TCCR3A : Timer/Counter Control Register A)

7	6	5	4	3	2	1	0	
COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	
7	6	5	4	3	2	1	0	
COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	TCCR3A
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

### Bit 7 ~ Bit 1

: OCnx 핀의 출력 모드 선택 비트(n=1,3 x=A,B,C)

: 타이머/카운터 1과 3에서 비교 매치 동작에 따른 출력 핀의 동작 결정

COMnx1	COMnx0	Normal, CTC 동작모드에서 OCnx핀의 출력 동작 (n=1,3 x=A,B,C)
0	0	일반 I/O 포트로 사용 (OCnx핀 차단)
0	1	비교 매치에서 OCnx 출력 (CTC mode)
1	0	OCnx 출력 clear (low 출력) 비교 출력
1	1	OCnx 출력 set (high 출력) 반전 출력

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

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

```
void main(void)
```

```
{
```

```
    DDRB = 0xFF;          // OC1C(PB7) 핀 출력방향설정  
    DDRG = 0xFF;          // 7-세그먼트 ON/OFF 제어 포트  
    PORTG = 0x0F;         // 7-세그먼트 모두 ON
```

```
    // 타이머/카운터1 초기화
```

```
    TCCR1A = 0b00000100;  // OC1C(PB7)핀에 파형 출력, mode4
```

```
    TCCR1B = 0b00001100;  // mode4(CTC 모드), 256분주
```

```
    TCCR1C = 0x20;        // 영향 없음
```

```
    TCNT1 = 0x0;          // 타이머/카운터1 레지스터 초기화
```

```
    OCR1A = 31249;        // 출력 파형 주기 초기값 : 1sec
```

```
    // OCR1CH= 0x0;       // 출력 비교 레지스터 C
```

```
    // OCR1CL = 0xFF;     // 토글되는 타/카1 비교값
```

```
    while(1){
```

```
        OCR1A = 31249;    // 주기 1sec
```

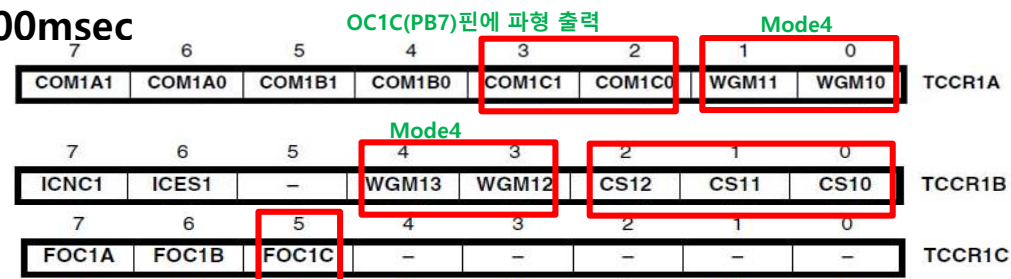
```
        delay_ms(6000);
```

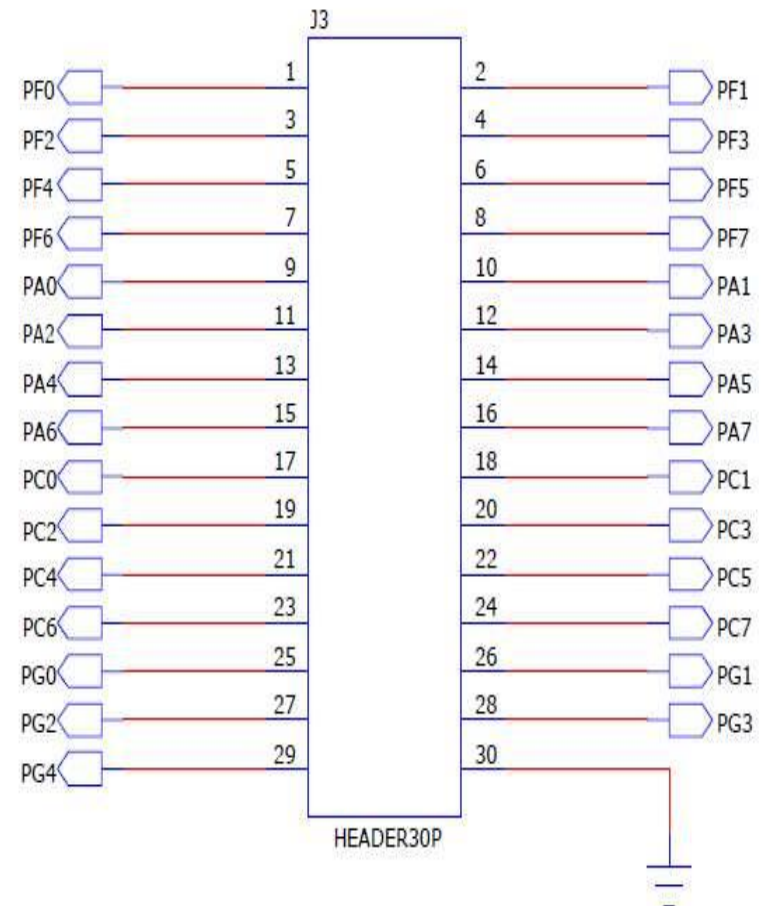
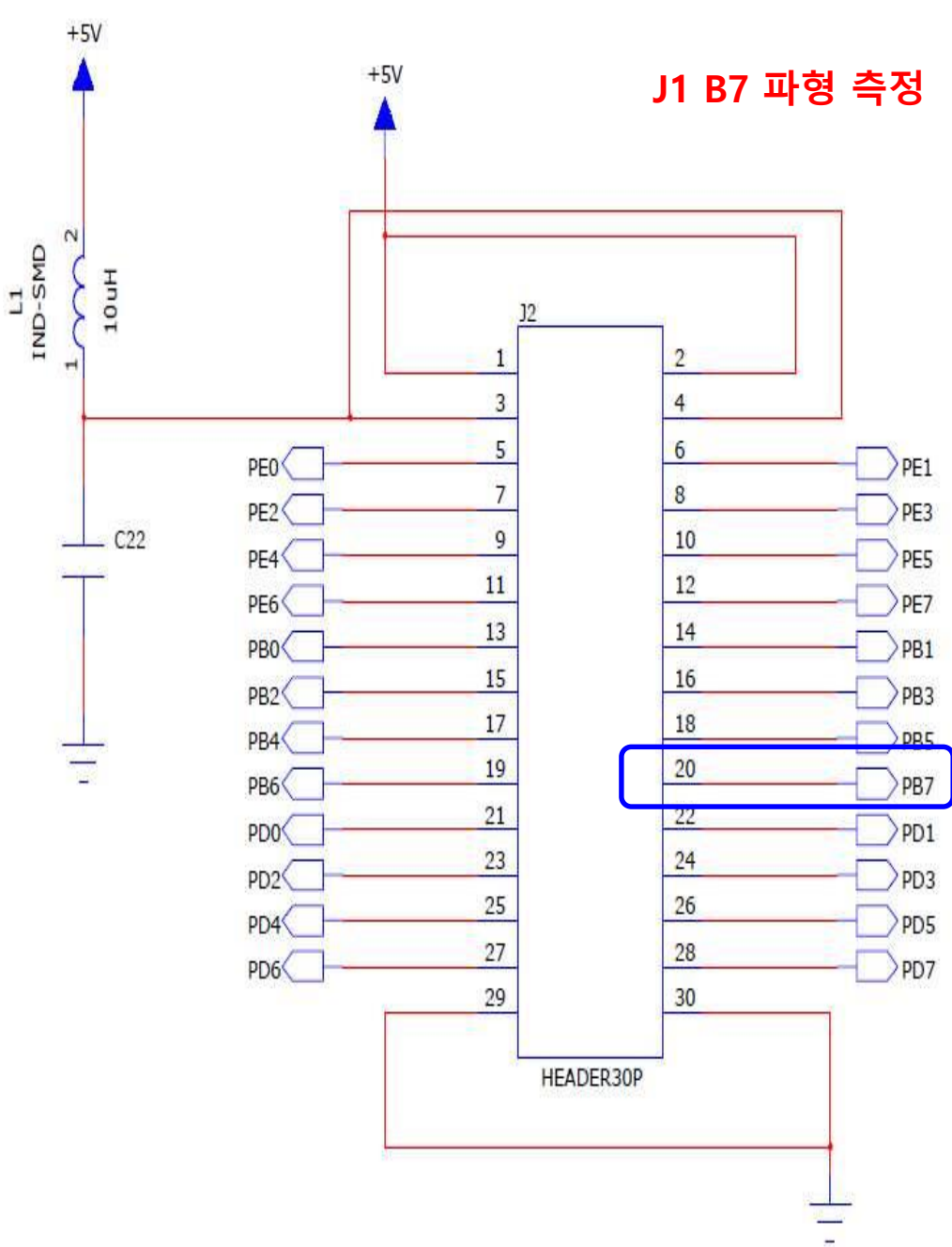
```
        OCR1A = 15624;    // 주기 500msec
```

```
        delay_ms(6000);
```

```
    }
```

```
}
```





# [PWM]

---

PWM 제어를 위해서는 기본적으로 ioport 가 출력으로 설정되어야 합니다

예를 들어, TIMER0 PWM PIN은 PB4 이며, DDRB의 4PIN 출력설정

TIMER2 PWM PIN은 PB7이며, DDRB의 7PIN 출력설정

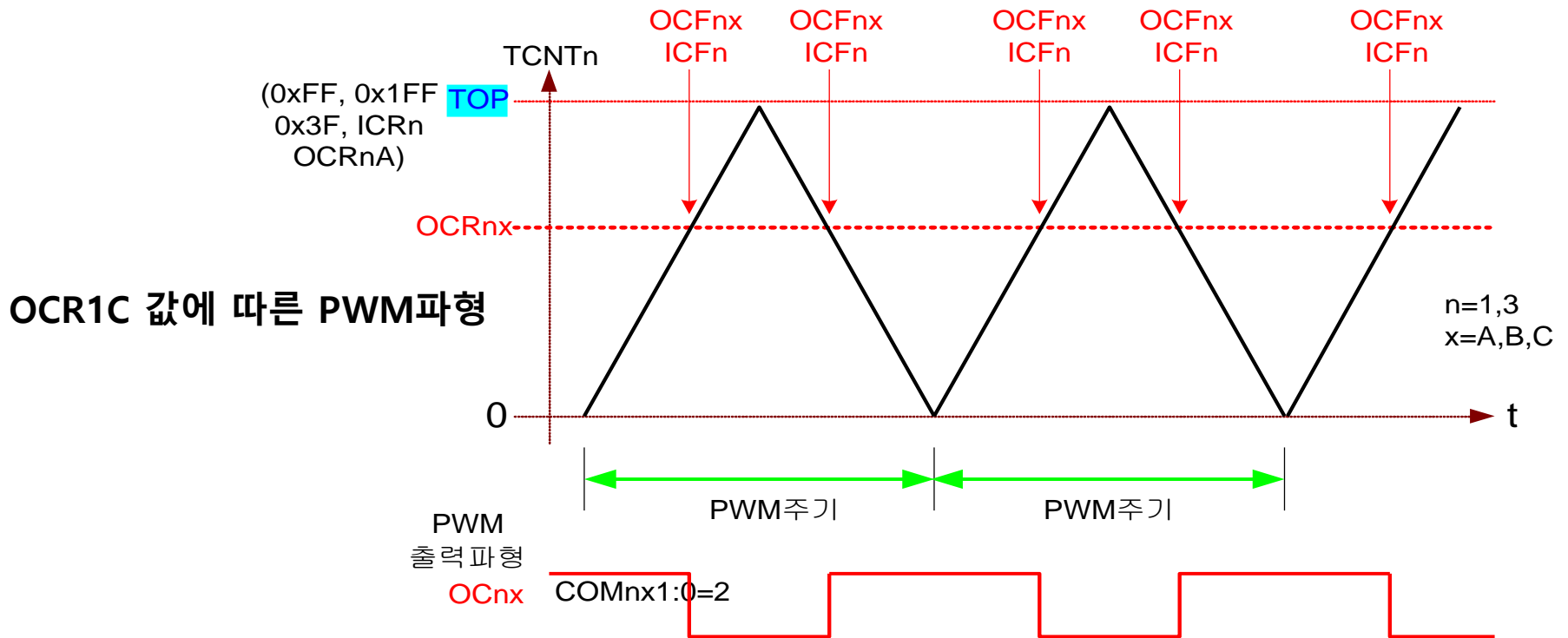
TIMER1 PWM PIN은 PB5,PB6,PB7 이며, DDRB 5,6,7 출력설정

TIMER3 PWM PIN은 PE3,PE4,PE5 이며, DDRE 3,4,5 출력설정해야 PWM 파형

# [예제 6-7] 타이머/카운터1를 이용한 PWM 제어 실험

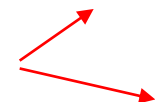
## (PC PWM : 모드3)

타이머/카운터1의 PWM모드에 대한 실험으로서, 모드3의 PC PWM 모드를 이용하여 SW1이 눌러지면 OC1C(PB7)핀에 연결되어 있는 7-세그먼트 DP LED의 ON시간이 길어지게 하고, SW2가 눌러지면 LED의 ON 시간이 짧아지도록 프로그램을 작성하라



## [힌트]

- 타이머/카운터1의 모드 3(10bit=1024)에서 TOP값은 0x03FF
- OC1C핀을 통해 출력되는 파형의 1또는 0이 되는 시간 폭은 출력비교 레지스터 OCR1C에 의해 결정
- 출력파형의 주기는 일정하며, OCR1C의 값을 가변시키면, 듀티비가 다른 PWM의 파형을 얻을 수 있게 된다.
- OC1C핀에 출력되는 파형은 레지스터 TCCR1A의 COM1C1:COM1C0에 의해 설정

TCCR1A = 0b00001011;  207 page

- COM1C1:COM1C0을 1:0로 설정하면 OC1C는 업 카운트의 비교매치에서 0, 다운 카운트의 비교매치에서는 1이 출력
- 출력비교 C 레지스터 OCR1C의 값을 증가시키면 0출력부분이 짧아지게 되며, OCR1C의 값을 감소시키면 0출력부분이 길어짐
- KUT-128\_COM보드에서 7-세그먼트 DP는 1을 출력할 때 ON이 되도록 설계
  - SW1이 눌러지면 OCR1C의 값을 증가시켜 1 출력부분이 길어지게 함
  - SW2가 눌러지면 OCR1C의 값을 감소시켜 1 출력부분이 짧아지게 함
  - 파형 주기는  $(1/16)\mu s \times 1024(TOP) \times 1024\text{분주} \times 2 = 131ms$   
(1024 분주비를 64분주로 바꿔서 실행해보도록 한다)

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

```
unsigned int pwm = 0x0200; // 현재의 출력비교 레지스터 값 저장
```

```
void main(void)
```

```
{
```

```
    DDRB = 0xFF; // OC1C(PB7) 핀 출력방향설정  
    DDRG = 0xFF; // 7-세그먼트 ON/OFF 제어 포트  
    PORTG = 0x0F; // 7-세그먼트 모두 ON
```

```
    EIMSK = 0b00110000; // 외부INT4 = 1, INT5 = 1
```

```
    EICRB = 0b00001010; // 외부 인터럽트 falling edge
```

```
    TCCR1A = 0b00001011; // 모드 3-10bits Phase Correct PWM
```

```
    TCCR1B = 0b00000100; // 타이머/카운터1 프리스케일러 = CK/256
```

```
    TCCR1C = 0x0;
```

```
    TCNT1 = 0x0; // 타이머/카운터1 레지스터 초기값 설정
```

```
    OCR1CH = (pwm & 0xFF00) >> 8; // OCR1C 초기값 설정 0x02
```

```
    OCR1CL = pwm & 0x00FF; // 0x00
```

```
    SREG = 0x80; // 전역 인터럽트 인에이블 비트 1 셋
```

```
    while(1);
```

```
}
```

임의 값

비교출력

// 외부 인터럽트 요구 4 서비스 루틴(SW1 처리)

**interrupt [EXT\_INT4] void external\_int4(void)**

```
{  
    if(pwm < 0x03B0) pwm += 0x0040; // 0x03B0보다 작으면 증가  
    OCR1CH = (pwm & 0xFF00) >> 8; // OCR1C 값 갱신  
    OCR1CL = pwm & 0x00FF;  
}
```

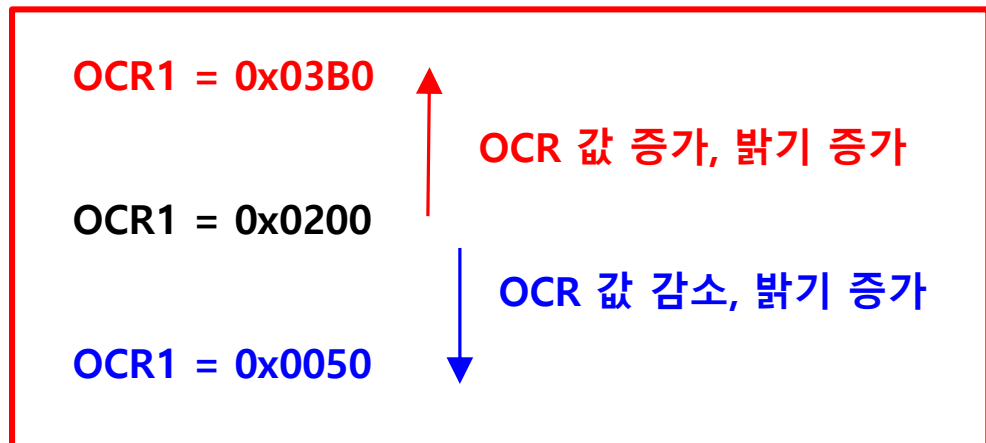
3FF 기준으로 임의 값

// 외부 인터럽트 요구 5 서비스 루틴(SW2 처리)

**interrupt [EXT\_INT5] void external\_int5(void)**

```
{  
    if(pwm > 0x0050) pwm -= 0x0040; // 0x0050보다 크면 감소  
    OCR1CH = (pwm & 0xFF00) >> 8; // OCR1C 값 갱신  
    OCR1CL = pwm & 0x00FF;  
}
```

파형 측정 : B7 오실로스코프





## [예제 6-8] 타이머/카운터0을 이용한 시계 제작

타이머/카운터0의 일반모드를 이용하여 시계를 제작하는 프로그램을 작성하라. 시간 조정은 SW1(INT4)과 SW2(INT5)의 외부 인터럽트를 이용하여 INT5는 입력할 자리 선택, INT4는 선택된 자리의 값을 +1 하도록 한다.

- 초기값 0, 분주 비 1024일 때 인터럽트 주기  
 $(256-0) \times 1024\text{분주} \times 1/16\mu\text{s} = 16.385\text{ms}$
- 1sec가 되기 위해서 16.384ms 인터럽트가  
요구될 횟수  $16.384\text{ms} \times 61\text{회} = 999.424\text{ms}$

# [예제 6-8] 타이머/카운터0을 이용한 시계 제작

## • 시계 조정

시계 자리	SW 누르기	SW 누름 위치	의미	표현 가능 숫자
	안 누름	POS = 0	1단위 분	0 - 9
	1번 누름	POS = 1	10단위 분	0 - 5
	2번 누름	POS = 2	1단위 시간	만일 POS=3의 값이 : 0,1경우, 0 - 9 가능 : 2경우, 0 - 3 가능
	3번 누름	POS = 3	10단위 시간	만일 POS=2의 값이 : 0,1,2,3 경우, 0, 1, 2 가능 : 4, - ,9 경우, 0, 1 가능

```

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

typedef unsigned char u_char;

flash u_char seg_pat[10]= {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f};

u_char N1 = 0, N10 = 0, N100 = 0, N1000 = 0;
u_char pos = 0;      // 0 : 1자리, 1 : 10자리 , 2 : 100자리 , 3 : 1000자리
u_char hour = 12, min = 0, sec = 0;
u_char cnt = 0;

void Time_out(void);      // 시간 출력 함수

void main(void)
{
    DDRB = 0xF0;          // 포트 B 상위 4비트 출력 설정
    DDRD = 0xF0;          // 포트 D 상위 4비트 출력 설정
    DDRG = 0x0F;          // 포트 G 하위 4비트 출력 설정

    EIMSK = 0b00110000;   // 외부 인터럽트 4,5 enable
    EICRB = 0b00001010;   // 외부 인터럽트 4,5 : falling edge
    TIMSK = 0x01;         // TOIE0 = 1(타이머/카운터0 오버플로우 인터럽트 enable
    TCCR0 = 0x07;         // 일반모드, 1024 분주
    TCNT0 = 0x00;         // 타이머/카운터0 레지스터 초기값
    SREG = 0x80;          // 전역 인터럽트 enable 비트 I 셋

```

```

while(1) {
    Time_out();          // 시간 출력

    if(cnt >= 61){        // 시간 값 갱신
        cnt = 0;
        sec = sec + 1;
        if(sec == 60) {
            sec = 0;
            min = min + 1;
            if(min == 60) {
                min = 0;
                hour = (hour + 1) % 24;
            }
        }
    }
}

```

**// 시간 출력 함수**

**void Time\_out(void)**

```
{  
    PORTG = 0b00001000;      // 7-Seg DIG4 ON(PG3=1), 분 1자리 표시  
    PORTD = ((seg_pat[min % 10] & 0x0F) << 4) | (PORTD & 0x0F);  
    PORTB = (seg_pat[min % 10] & 0x70 ) | (PORTB & 0x0F);  
    delay_ms(5);  
  
    PORTG = 0b00000100;      // 7-Seg DIG3 ON(PG2=1), 분 10자리 표시  
    PORTD = ((seg_pat[min / 10] & 0x0F) << 4) | (PORTD & 0x0F);  
    PORTB = (seg_pat[min / 10] & 0x70 ) | (PORTB & 0x0F);  
    delay_ms(5);  
  
    PORTG = 0b00000010;      // 7-Seg DIG2 ON(PG1=1), 시간 1자리 표시  
    PORTD = ((seg_pat[hour % 10] & 0x0F) << 4) | (PORTD & 0x0F);  
    PORTB = (seg_pat[hour % 10] & 0x70 ) | (PORTB & 0x0F);  
    delay_ms(5);  
  
    PORTG = 0b00000001;      // 7-Seg DIG1 ON(PG0=1), 시간 10자리 표시  
    PORTD = ((seg_pat[hour / 10] & 0x0F) << 4) | (PORTD & 0x0F);  
    PORTB = (seg_pat[hour / 10] & 0x70 ) | (PORTB & 0x0F);  
    delay_ms(5);  
}
```



**// 외부 인터럽트 요구 5 서비스 루틴(SW2 처리)**

```
interrupt [EXT_INT5] void external_int5(void)  
{  
    pos = (pos + 1) % 4;          // 입력 자리 이동  
}
```

**// 타이머/카운터0 오버플로우 인터럽트 처리, 주기 =  $1/16 * 1024 * 256 = 16.384\text{ms}$**

```
interrupt [TIM0_OVF] void timer0_int(void)  
{  
    TCNT0 = 0x0;          // 타이머/카운터0 초기값 재설정(생략가능)  
    cnt++;                // 인터럽트 회수 +1  
}
```

• Stabilization of SW - MISRA (Motor Industry Software Reliability Association)

: MISRA

“Guidelines for the use of the C language in vehicle based software” – 1998, 영국산업신뢰성협회

S.No	MISRA Rule#	Description
1	MISRA rule 5	Only those characters and escape sequences which are defined in the ISO C standard shall be used.
2	MISRA rule 7	Trigraphs shall not be used.
3	MISRA rule 8	Multibyte characters and wide string literals shall not be used.
4	MISRA rule 9	Comments shall not be nested.
5	MISRA rule 10	Sections of code should not be 'commented out'
6	MISRA rule 13	The basic types of char, int, long, float and double should not be used, but specific length equivalents should be typedefed for the specific compiler (and microprocessor).
7	MISRA rule 14	The type char shall always be declared as unsigned char or signed char.
8	MISRA rule 16	The underlying bit representations of floating point numbers shall not be used in any way by the programmer.
9	MISRA rule 17	typedef names shall not be reused.
10	MISRA rule 19	Octal constants (other than zero) shall not be used.
11	MISRA rule 20	All object and function identifiers shall be declared before use.
12	MISRA rule 21	Identifiers in an inner scope shall not use the same name as an identifier in an outer scope and therefore hide that identifier.
13	MISRA rule 22	Declarations of objects should be at function scope unless a wider scope is necessary.
14	MISRA rule 23	All declarations at file scope should be static where possible.
15	MISRA rule 25	An identifier with external linkage shall have exactly one external definition.
16	MISRA rule 26	If objects or functions are declared more than once they shall have compatible declarations
17	MISRA rule 27	External objects should not be declared in more than one file.
18	MISRA rule 29	The use of a tag shall agree with ist declaration.
19	MISRA rule 30	All automatic variables shall have been assigned a value before being used.
20	MISRA rule 31	Braces shall be used to indicate and match the structure in the non-zero initialisation of arrays and structures.
21	MISRA rule 32	In an enumerator list, the "=" construct shall not be used to explicitly initialise members other than the first, unless all items are explicitly initialised.
22	MISRA rule 33	The right hand operand of a && or    operator shall not contain side effects.
23	MISRA rule 34	The operands of a logical && or    shall be primary expressions
24	MISRA rule 35	Assignment operators shall not be used in expressions which return Boolean values.
25	MISRA rule 37	Bitwise operations shall not be performed on signed integer types.
26	MISRA rule 38	The right hand operand of a shift operator shall lie between zero and one less than the width in bits of the lefthand operand (inclusive).
27	MISRA rule 39	The unary minus operator shall not be applied to an unsigned expression.
28	MISRA rule 40	The sizeof operator should not be used on expressions that contain side effects.
29	MISRA rule 43	Implicit conversions which may result in a loss of information shall not be used.
30	MISRA rule 45	Type casting from any type to or from pointers shall not be used.
31	MISRA rule 46	The value of an expression shall be the same under any order of evaluation that the standard permits.
32	MISRA rule 48	Mixed precision arithmetic should use explicit casting to generate the desired result.
33	MISRA rule 50	Floating point variables shall not be tested for exact equality or inequality.
34	MISRA rule 52	There shall be no unreachable code.
35	MISRA rule 56	The goto statement shall not be used.
36	MISRA rule 59	The statements forming the body of an if, else if, else, while, do... While or for statement shall always be enclosed in braces.
37	MISRA rule 61	Every non-empty case clause in a switch statement shall be terminated with a break statement.
38	MISRA rule 62	All switch statements should contain a final default clause.
39	MISRA rule 64	Every switch statement shall have at least one case.
40	MISRA rule 65	Floating point variables shall not be used as loop counters.
41	MISRA rule 68	Functions shall always be declared at file scope.
42	MISRA rule 69	Functions with variable numbers of arguments shall not be used.
43	MISRA rule 70	Functions shall not call themselves, either directly or indirectly.
44	MISRA rule 71	Functions shall always have prototype declarations and the prototype shall be visible at both the function definition and call.
45	MISRA rule 75	Every function shall have an explicit return type.
46	MISRA rule 76	Functions with no parameters shall be declared with parameter type void.
47	MISRA rule 78	The number of parameters passed to a function shall match the function prototype.
48	MISRA rule 79	The values returned by void functions shall not be used.
49	MISRA rule 80	Void expressions shall not be passed as function parameters.
50	MISRA rule 81	const qualification should be used on function parameters which are passed by reference, where it is intended that the function will not modify the parameter.
51	MISRA rule 83	For functions with non-void return type
52		i, there shall be one return statement for every exit branch (including the end of program)
53		ii, each return shall have an expression
54		iii, the return expression shall match the declared return type.
55	MISRA rule 84	For functions with void return type, return statements shall not have an expression.
56	MISRA rule 85	Functions called with no parameters should have empty parentheses.
57	MISRA rule 87	#include statements in a file shall only be preceded by other preprocessor directives or comments.
58	MISRA rule 88	Non-standard characters shall not occur in header file names in #include directives.
59	MISRA rule 89	The #include directive shall be followed by either a <filename> or "filename" sequence.
60	MISRA rule 91	Macros shall not be #defined and #undefed within a block.
61	MISRA rule 94	A function-like macro shall not be 'called' without all of its arguments.
62	MISRA rule 95	Arguments to a function-like macro shall not contain tokens that look like preprocessing directives.
63	MISRA rule 96	In the definition of a function-like macro the whole definition, and each instance of a parameter, shall be enclosed in parentheses.
64	MISRA rule 98	There shall be at most one occurrence of the # or ## pre-processor operators in a single macro definition.
65	MISRA rule 99	All uses of the #pragma directive shall be documented and explained.
66	MISRA rule 100	The defined pre-processor operator shall only be used in one of the two standard forms.
67	MISRA rule 102	No more than 2 levels of pointer indirection should be used.
68	MISRA rule 103	Relational operators shall not be applied to pointer types except where both operands are of the same type and point to the same array, structure or union.
69	MISRA rule 104	Non-constant pointers to functions shall not be used.
70	MISRA rule 105	All the functions pointed to by a single pointer to function shall be identical in the number and type of parameters and the return type.
71	MISRA rule 106	The address of an object with automatic storage shall not be assigned to and object which may persist after the object has ceased to exist.
72	MISRA rule 107	The null pointer shall not be de-referenced.

Abbreviation	description	Criteria
N1	Total Number of operator	
N2	Total number of operands	
STMT	Number of statement	< =50
VG	Cyclomatic Number (VG)	< =20
AVGS	Average size of statement	< =9
GOTO	Number of GOTO statement	0
RETU	Number of Return Statement	1
NBCALLING	Number of caller	< =10
PATH	Number of path	< =1000
PARA	Number of function parameter	< =5
ap_cg_cycle	call Graph recursions	0
	dead code	
DRCT_CALLS	Number of calls direct	
LEVL	Number of Level	





1. while(1)문 시작 전에 7segments에 자신의 학번(8자리)을 쓰고 눈으로 확인할 수 있도록 디스플레이 하기. 이때 학번 8자리 반드시 표시하고(표시방법은 자유롭게 프로그램) 그리고 while(1)문에서 계속 학번 뒤 4자리를 계속 디스플레이 하기
2. while(1)이전에서 External INT 4, 5, 6, 7를 활성화(Enable 시킬 것)시키고, nesting은 허용하지 않도록 함. While(1)에서는 아래의 이벤트를 수행한다
  - (1) External INT 4(rising edge) 이벤트가 발생하면 8bit timer2 overflow mode로 학번 뒤 (2자리 x msec) 주기를 만들고, (예를 들어 20211234라면 34msec) 8bit timer2 overflow Interrupt를 발생시키고, 이를 이용하여 4500msec 될 때마다 ADC를 통해 온도(free running mode) 측정하여 7-segment에 디스플레이 할 것. 이때 학번 표시는 멈추고, 온도를 디스플레이 한다. 온도는 소수점 1의 자리까지, 소수점 표시 => 예 31.2 : 7-segments 4개 모두 사용) 그리고 ADC를 통해 온도검출후 온도표시는 10회 동안만 동작시킨다. 그이후 다시 학번을 정상적으로 표시한다.
  - (2) External INT 5(falling edge) 발생하면 발생하면 16bit timer1 compare match(CTC)로 학번 뒤 (2자리 x 100msec) 주기를 만들고, (예를 들어 20211234라면 3400msec) timer1 CTC interrupt를 발생시키고 Interrupt 발생하면 ADC를 통해 전압(single mode)을 측정하고 측정된 값을 UART를 동작시켜 PC화면에 전압 값을 디스플레이 시킬 것
  - (3) External INT 6(rising edge) 발생하면 timer3의 90bit PWM mode6를 동작시키고, OC3A(PE3) pin으로 DUTY비-학번\_뒤자리\_2개 (예를 들어 20211234라면 34% duty)를 PWM를 출력시킨다
  - (4) External INT 7(falling edge) 발생하면 문제3의 timer3의 9bit PWM mode 출력을 중지시키고 timer3의 9bit Phase correct PWM mode2를 동작시키고, OC3A(PE3) pin으로 DUTY비-학번\_뒤자리\_2개 (예를 들어 20211234라면 34% duty)를 PWM를 출력시킨다
3. interrupt service routine은 짧게 프로그래밍할 것 / 변수는 typedef 문을 사용하여 선언할 것 / 프로그램 맨 위에는 주석 작성

## Project 2-1

- : Main 함수
- : while(1)
- : External interrupt
- : TIMER 설정

감사합니다