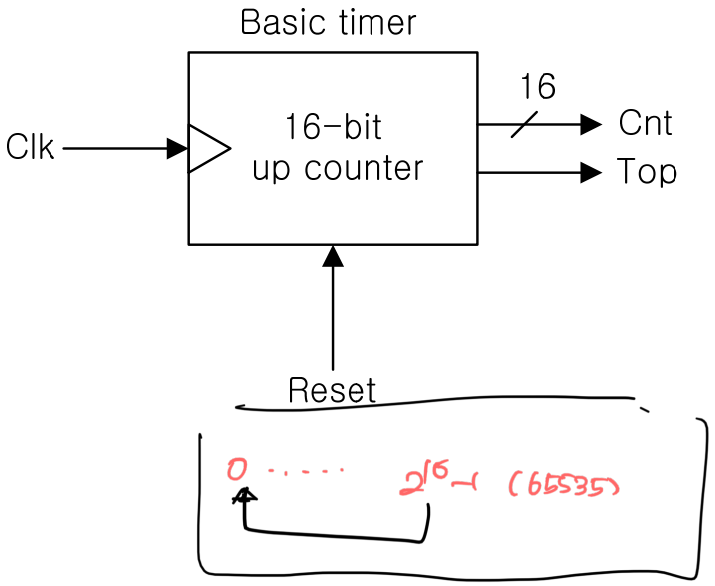
Timer/Counter

Prof. Lee, Y. S.

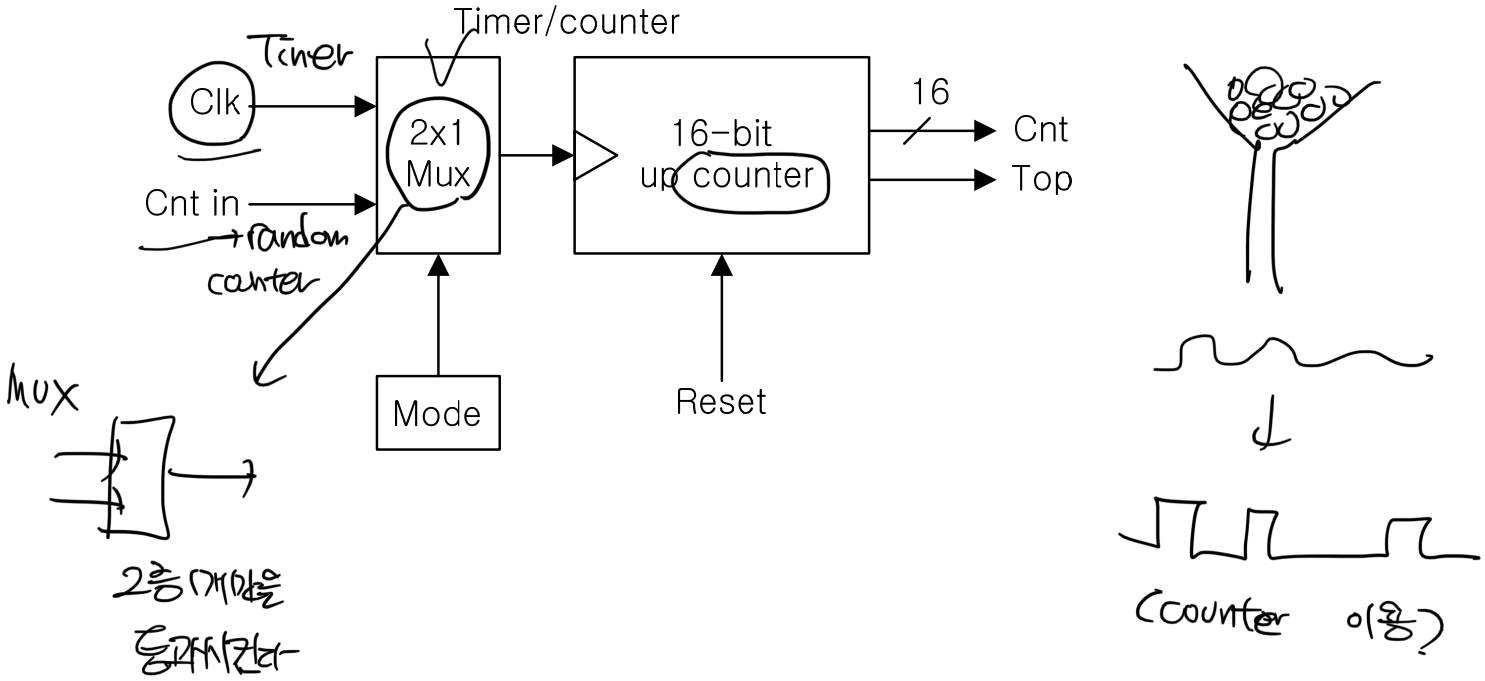
Timer/Counter

- **Timer : measures time intervals**
  - **To generate timed output events**
    (Ex) hold traffic light green for 10 s
  - **To measure input events**
    (Ex) measure a car's speed



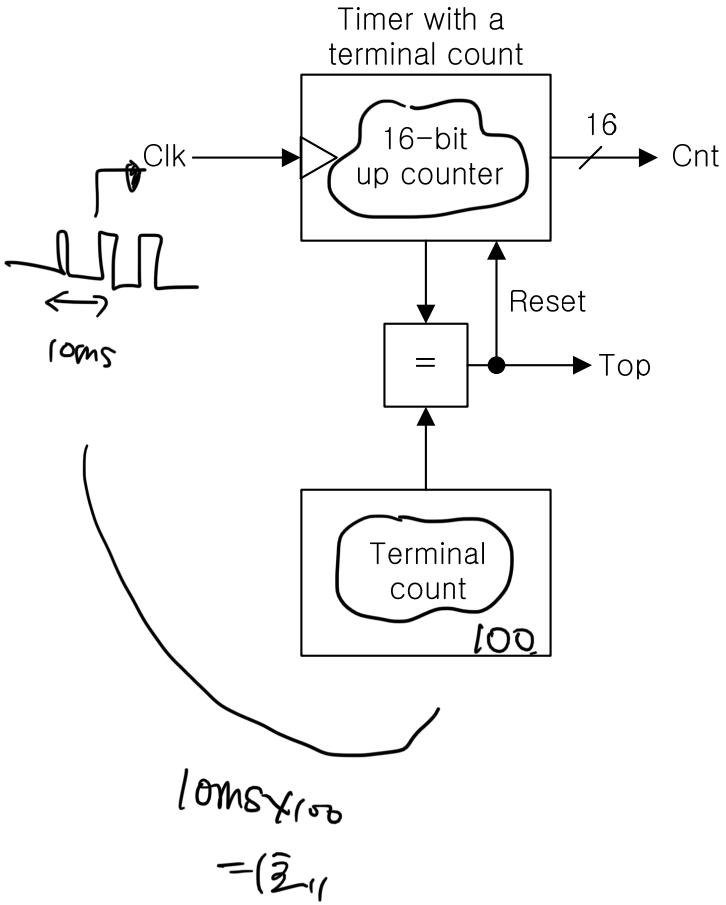
- **Based on counting clock pulses**
  - **E.g., let Clk period be 10 ns**
  - **And we count 20,000 Clk pulses**
  - **Then 200 microseconds have passed**
  - **16-bit counter would count up to 65,535×10 ns**
    = 655.35 micro seconds.,
    resolution = 10 ns
  - **Top: indicates top count reached, wrap-around**

- **Counter** : like a timer, but counts pulses on a general input signal rather than periodic clock pulse
  - e.g., count cars passing over a sensor
  - Can often configure device as either a timer or counter



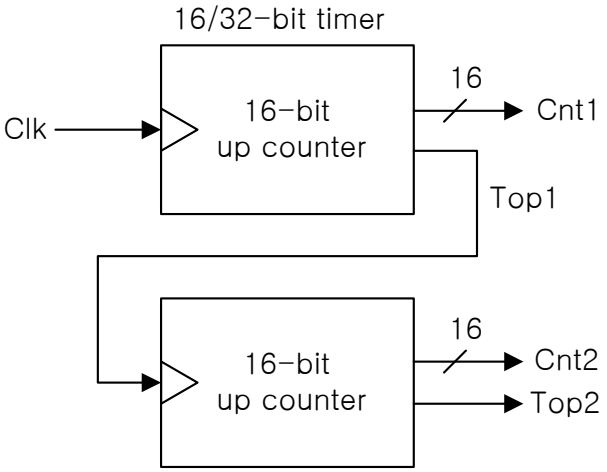
- **Interval timer**
  - **Indicates when desired time interval has passed**
  - **We set terminal count to desired interval**

    Number of clock cycles = Desired time interval / Clock period

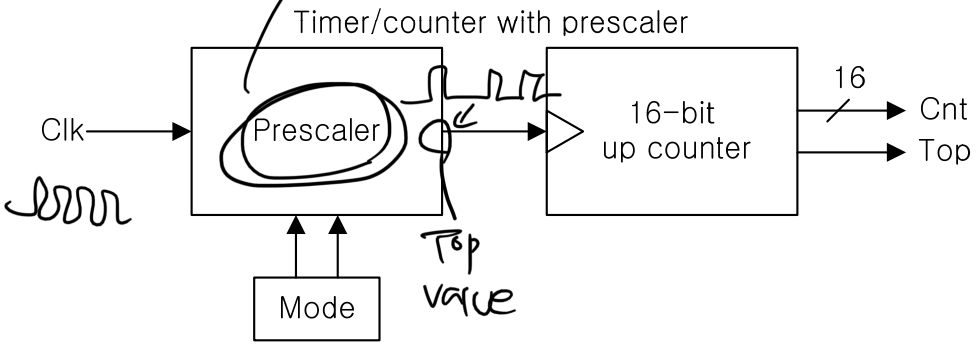


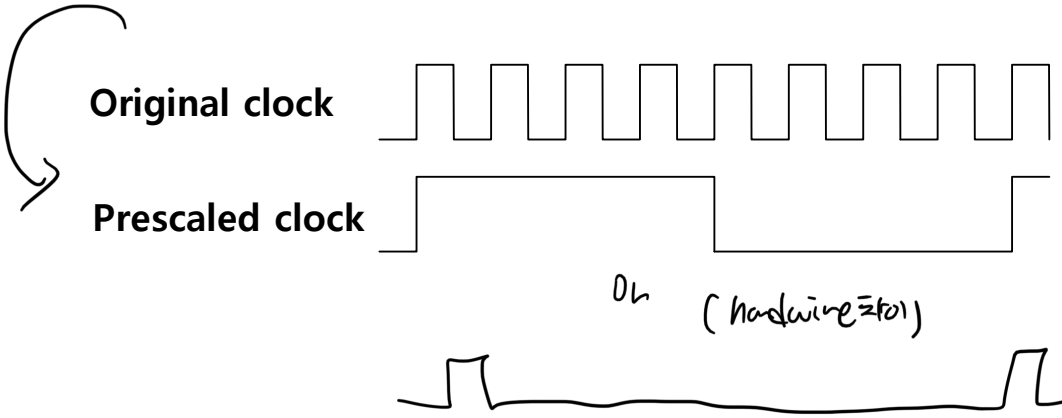
Can be used to configure the period of timer interrupt service routine

- **Cascaded counters**
- **Prescaler**
  - **Divides clock**
  - **Increases range, decreases resolution**

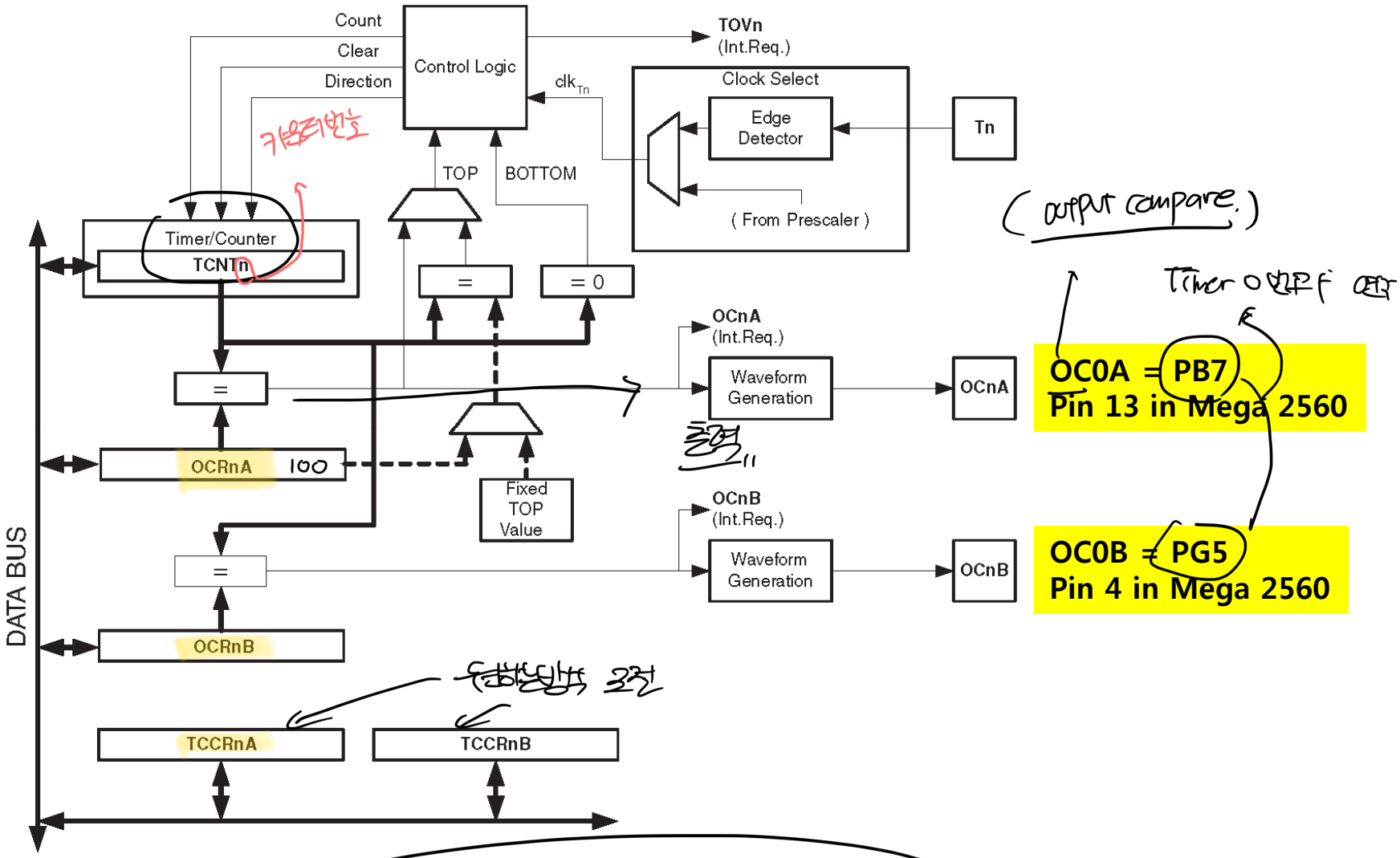


Cascaded counter





8-bit Timer/Counter Block (0,2)

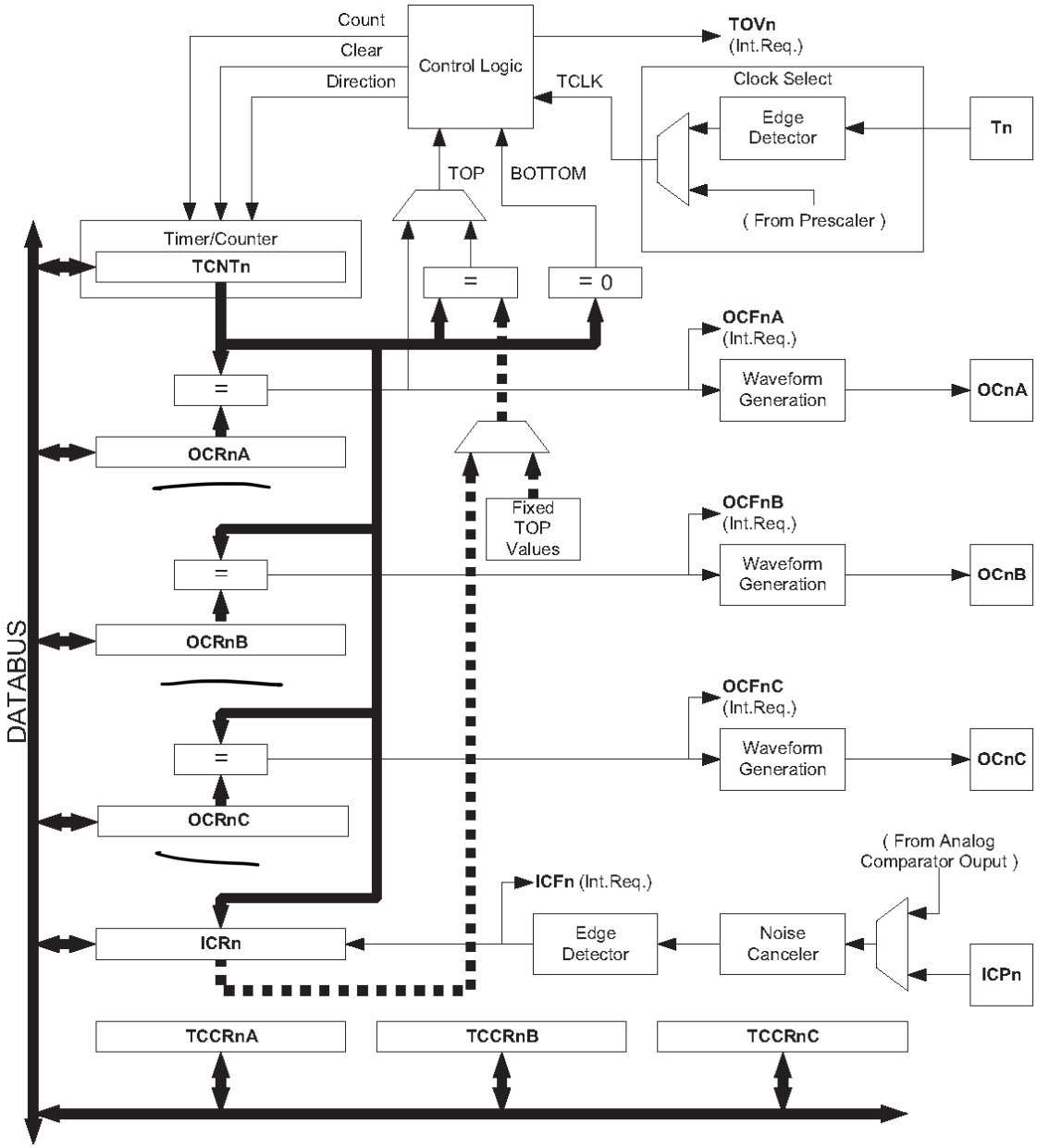


OC0A = PB7
Pin 13 in Mega 2560

OC0B = PG5
Pin 4 in Mega 2560

OCR : Output compare register
TCCR : Timer counter control register

16-bit Timer/Counter Block (1,3,4,5)



OC1A = PB5 = Pin 11 in Mega 2560

OC1B = PB6 = Pin 12 in Mega 2560

OC1C = PB7 = Pin 13 in Mega 2560

OC3A = PE3 = Pin 5 in Mega 2560

OC3B = PE4 = Pin 2 in Mega 2560

OC3C = PE5 = Pin 3 in Mega 2560

Which Counter/Which Mode?

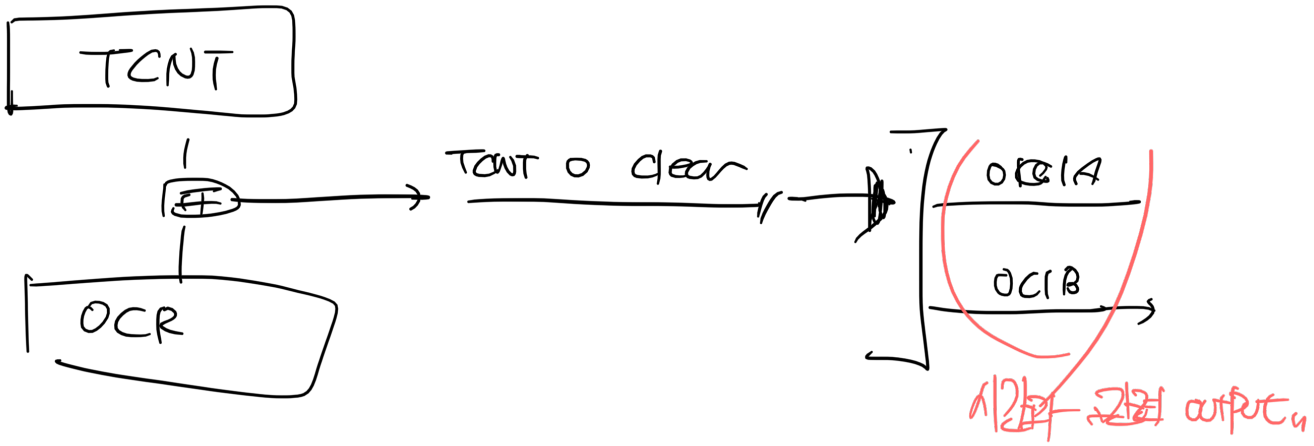
- **Timer/Counter Modes**
  - **Normal mode**
  - **CTC mode (Clear Timer on Compare Match)**
  - **Fast PWM mode**
  - **Phase Correct PWM mode (PC PWM)**
  - **Phase and Frequency Correct PWM mode (PFC PWM)**

[Remark] CTC mode can be used for timer interrupt.

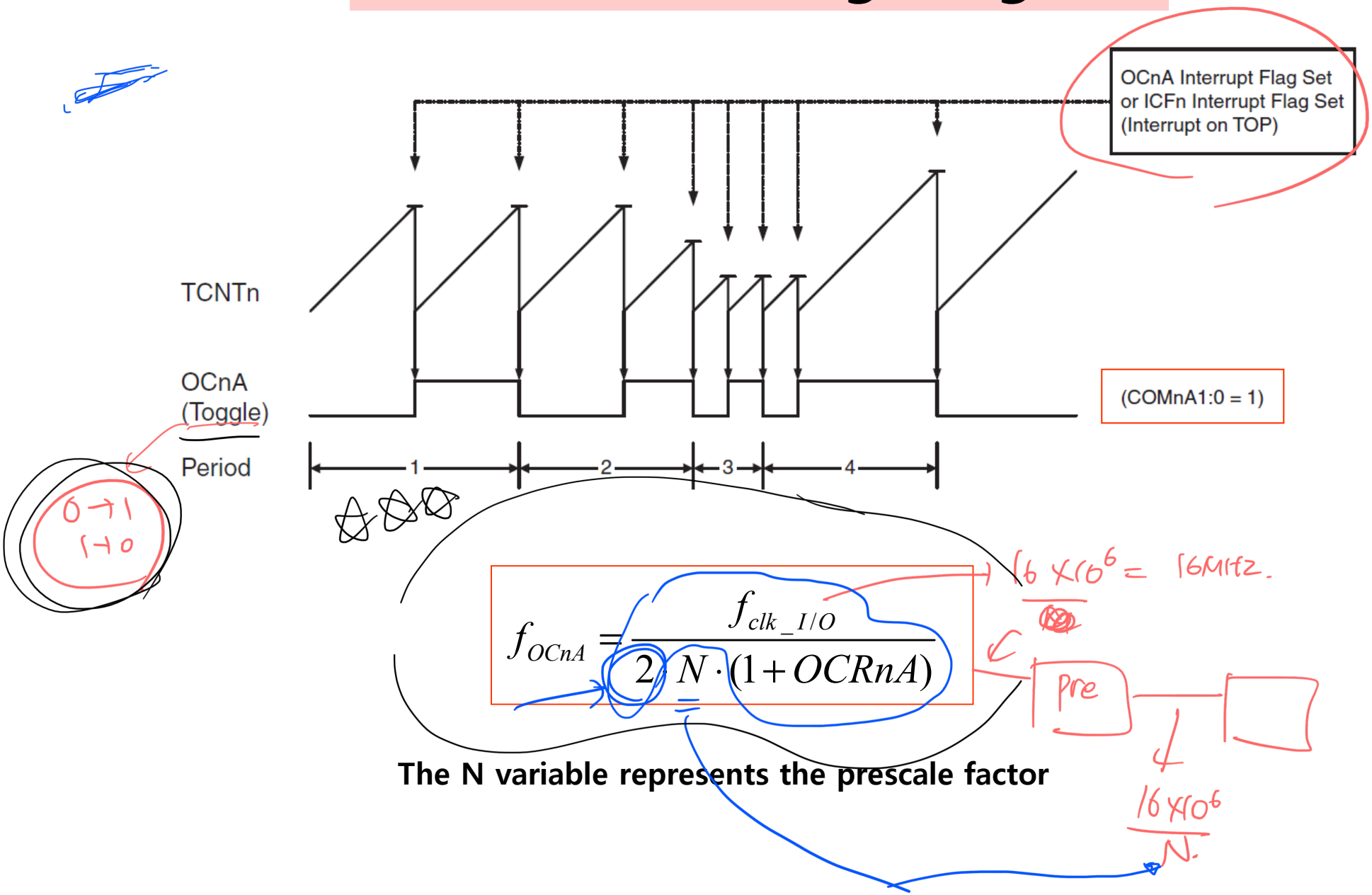
Normal Mode, CTC Mode

- **Normal mode**
  - **The counting direction is always up, and no counter clear is performed. The counter simply overruns when it passes its maximum and then restarts from the BOTTOM.**

- **CTC Mode : Clear Timer on Compare Match**
  - **The counter is cleared to zero when the counter value (TCNTn) matches either OCRnA or the ICRn.**
  - **The OCRnA or ICRn define the top value for the counter, hence also its resolution.**



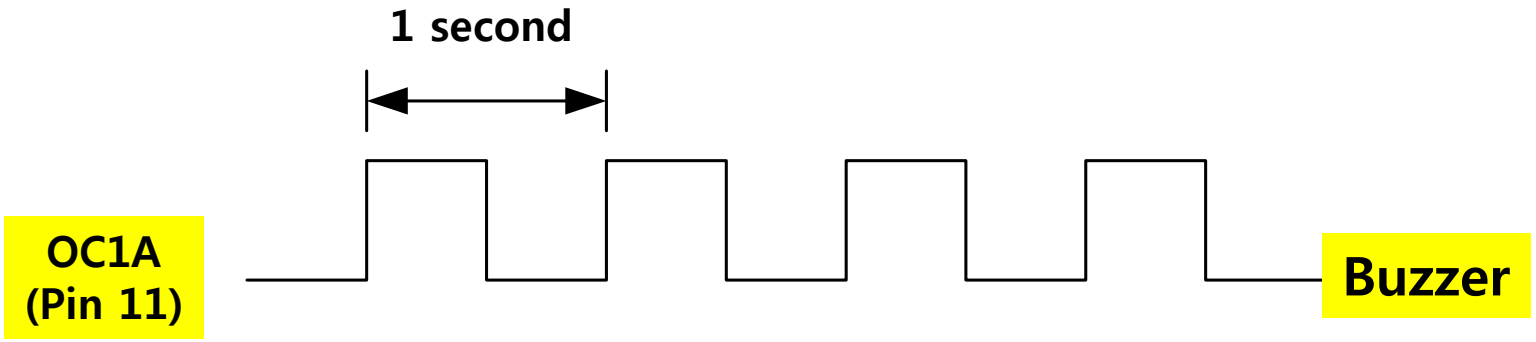
CTC Mode Timing Diagram



The N variable represents the prescale factor

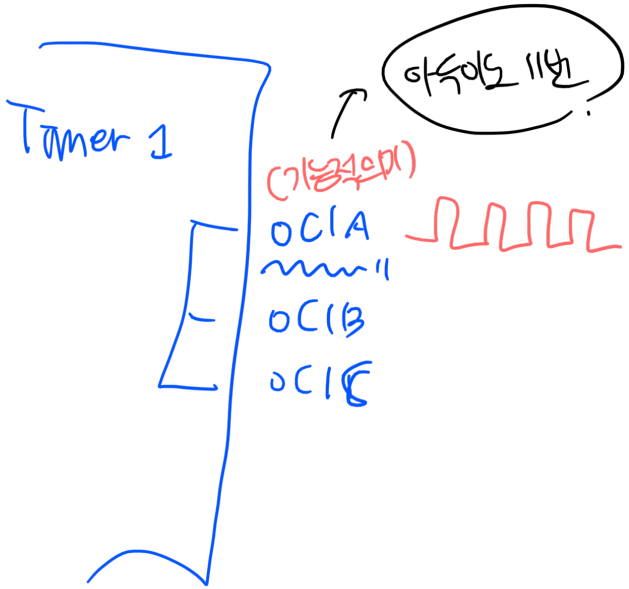
Buzzer Control using CTC Mode

Problem: We want to generate a beep sound every 1 second using a buzzer



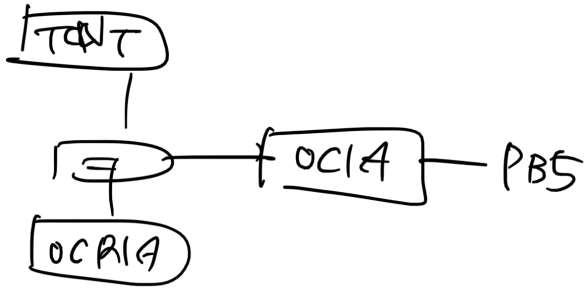
Wire Connection

Arduino Pin #	Peripheral Board : IO_J3 (LED)
11 (PB5, OC1A)	Buzzer (LED8)



What to do

- Configure Timer/Counter1 as CTC mode (TCCR1A,TCCR1B)
- Enable OC1A for buzzer control (TCCR1A)
- Set the prescaler value (TCCR1B)
- Set the top value (OCR1A)
- Clear Timer/Counter1 (TCNT1)

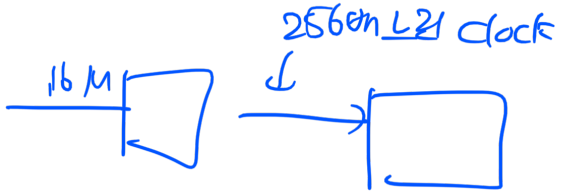


$$f_{OC1A} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCR1A)}$$

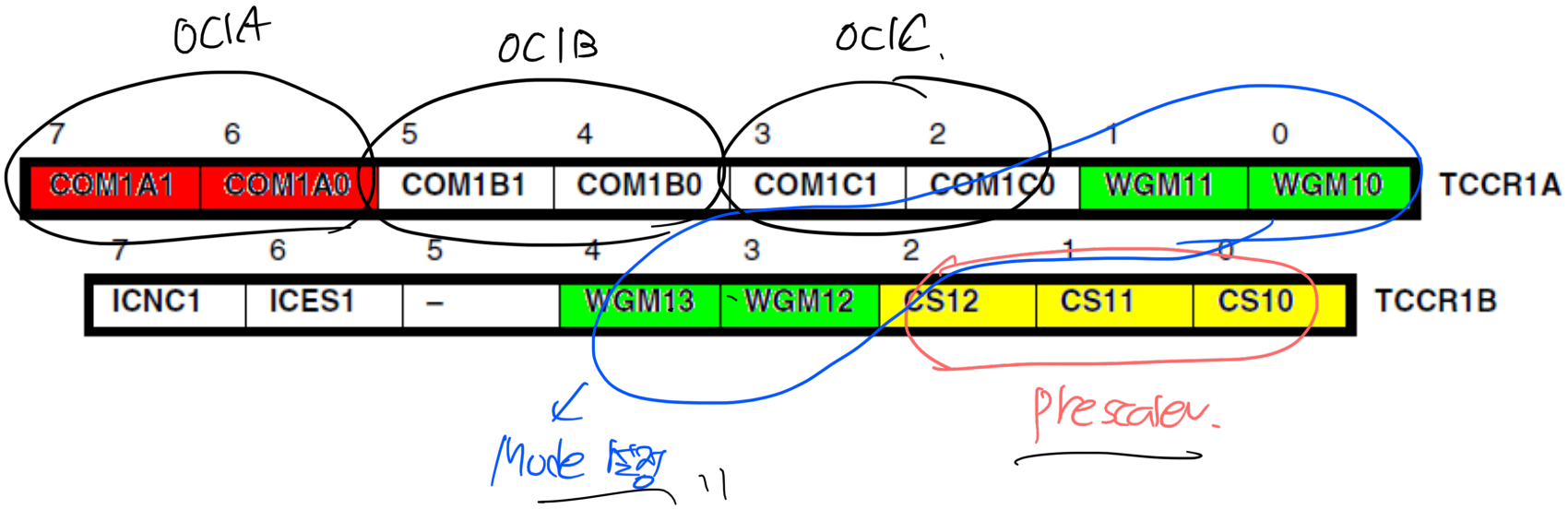
$$f_{clk\_I/O} = 16 \times 10^6$$

$$N = 256,\ OCR1A = 31249$$

Prescaler=256, OCR1A=31249



Related Registers



Bits	Functions
WGM13,WGM12,WGM11,WGM10	Timer/Counter mode selection
COM1A1,COM1A0	Compare output mode selection
CS12,CS11,CS10	Prescaler

Timer/counter mode selection

CTC → Top output(OCR)
↳ Input (ICR)

Table 14-5. Waveform Generation Mode Bit Description (1)

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

0100 : CTC mode with OCRnA as a top value register

1100 : CTC mode with ICRn as a top value register

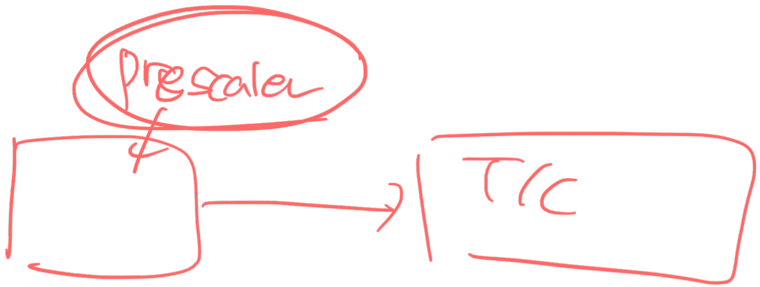
Compare output mode selection

Table 14-2. Compare Output Mode, non-PWM

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

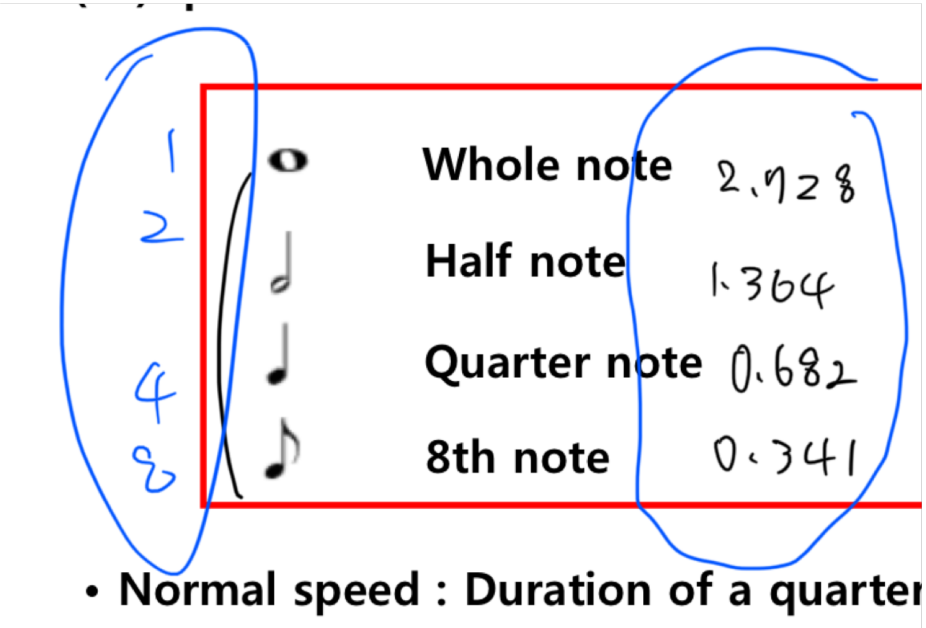
Table 14-6. Clock Select Bit Description

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

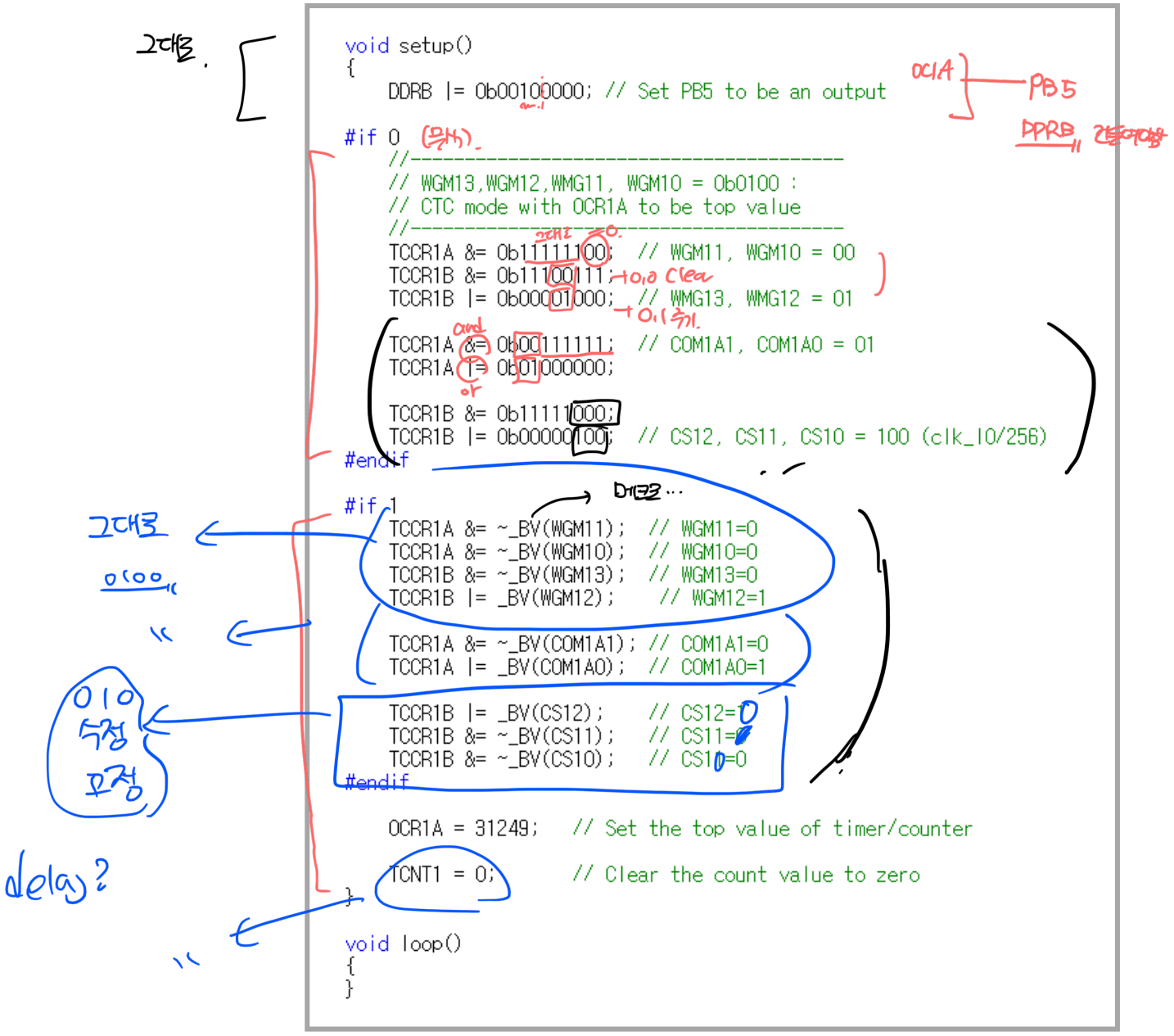


BuzzerByTimer1.ino

```
//-------------------------------------------------------------------------
// File name : BuzzerByTimer1.ino
// <Abstract>
// 16-bit Timer/Counter1를 이용해서 Buzzer를 1초마다 한번씩 beep sound를
// 내도록 한다. Timer/Counter의 CTC mode를 사용한다.
//
// <Connection Info>
// ---------------------------------------
//  Arduino Pin      Peripheral Board
// ---------------------------------------
//      11 (PB5, OC1A)   Buzzer (LED8)
// ---------------------------------------
//-------------------------------------------------------------------------
```



BuzzerByTimer1.ino



```
void setup()
{
    DDRB |= 0b00100000; // Set PB5 to be an output

#if 0
    //-----------------------------------------
    // WGM13,WGM12,WMG11, WGM10 = 0b0100 :
    // CTC mode with OCR1A to be top value
    //-----------------------------------------
    TCCR1A &= 0b11111100;  // WGM11, WGM10 = 00
    TCCR1B &= 0b11100111;
    TCCR1B |= 0b00001000;  // WMG13, WMG12 = 01

    TCCR1A &= 0b00111111;  // COM1A1, COM1A0 = 01
    TCCR1A |= 0b01000000;

    TCCR1B &= 0b11111000;
    TCCR1B |= 0b00000100;  // CS12, CS11, CS10 = 100 (clk_IO/256)
#endif

#if 1
    TCCR1A &= ~_BV(WGM11);  // WGM11=0
    TCCR1A &= ~_BV(WGM10);  // WGM10=0
    TCCR1B &= ~_BV(WGM13);  // WGM13=0
    TCCR1B |= _BV(WGM12);    // WGM12=1

    TCCR1A &= ~_BV(COM1A1); // COM1A1=0
    TCCR1A |= _BV(COM1A0);  // COM1A0=1

    TCCR1B |= _BV(CS12);    // CS12=1
    TCCR1B &= ~_BV(CS11);   // CS11=0
    TCCR1B &= ~_BV(CS10);   // CS10=0
#endif

    OCR1A = 31249;   // Set the top value of timer/counter

    TCNT1 = 0;       // Clear the count value to zero
}

void loop()
{
}
```

Related header file

<sfr_defs.h>

```
#define __SFR_OFFSET 0x20
#define _SFR_IO8(io_addr) ((io_addr) + __SFR_OFFSET)
```

<iomxx0_1.h>

```
#define TCCR0A  _SFR_IO8(0x24)
#define COM0A1  7
#define COM0A0  6
#define COM0B1  5
#define COM0B0  4
#define WGM01   1
#define WGM00   0

#define TCCR0B  _SFR_IO8(0x25)
#define FOC0A   7
#define FOC0B   6
#define WGM02   3
#define CS02    2
#define CS01    1
#define CS00    0
```

Directory for the above header file :
→ c:₩arduino-1.8.0₩hardware₩tools₩avr₩avr₩include₩avr₩

My Arduino IDE version is 1.8.0. Yours are different from mine. Then the directory info should be based on your Arduino version.

The BV macro

- In the C language one assigns and test bits using bit operators, the assign operators, and the concept of bit masks:

```
PORTC |= 0x01;     // set bit 0 only
PORTC &= ~0x01; // clear bit 0 only
PORTC ^= 0x01;   // Toggle bit 0 only
PORTC |= 0x80;    // Set bit 7 only
```

- Using macros make this easier to read. The _BV macro in avr-libc takes a number as the argument and converts it to the appropriate bit mask. (BV = Bit Value)

```
#define _BV(x)  (1<<x)
```

```
PORTC |= _BV(0);       // set bit 0 only
PORTC &=~(_BV(1)); // clear bit 0 only
PORTC |= _BV(7);       // Set bit 7 only
```

$$f_{OC1A} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCR1A)}$$

$f_{clk\_I/O} = 16 \times 10^6$

$N = 256$

What if OCR1A varies from 1 to 1562 ?

$$OCR1A = 1 \quad \rightarrow f_{OC1A} = 15,625\ Hz$$

$$OCR1A = 1561 \rightarrow f_{OC1A} = 20.006\ Hz$$