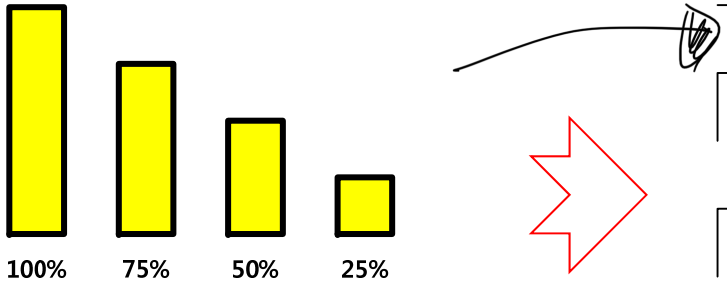
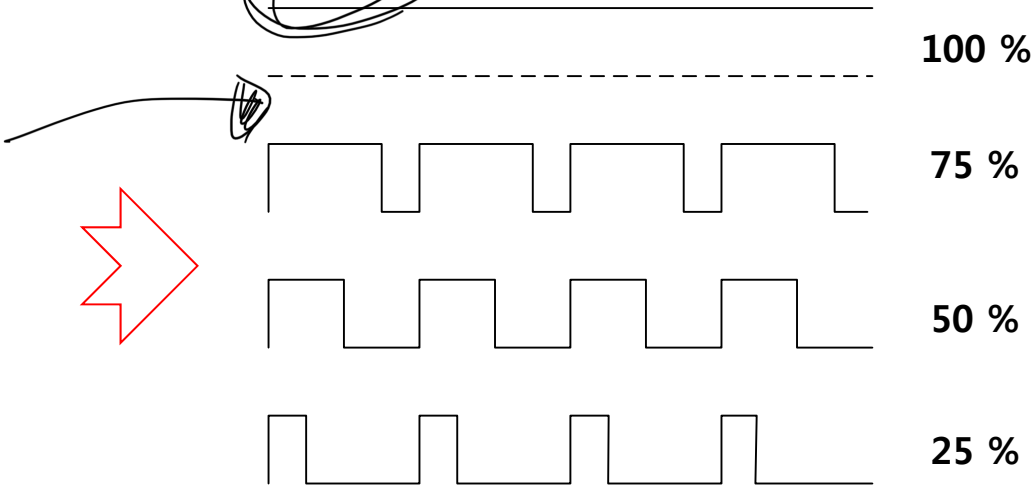
PWM

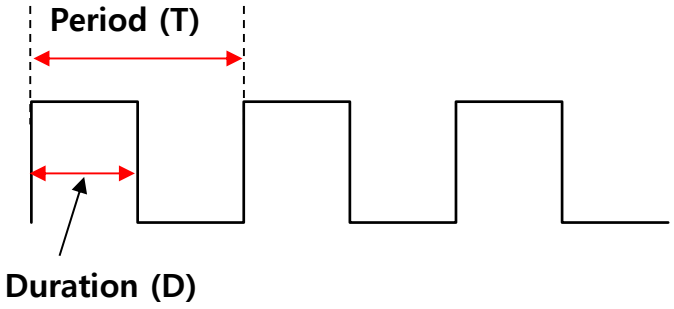
Prof. Lee, Y. S.

PWM

- PWM (Pulse Width Modulation)
  - Information is represented by the width of the pulse
  - Duty ratio = Duration(D)/Period(T)
  - PWM frequency = 1/T
  - PWM is used for switching transistors in a motor drive or Class D amplifiers.
  - When connected to LPF, it acts as a DAC





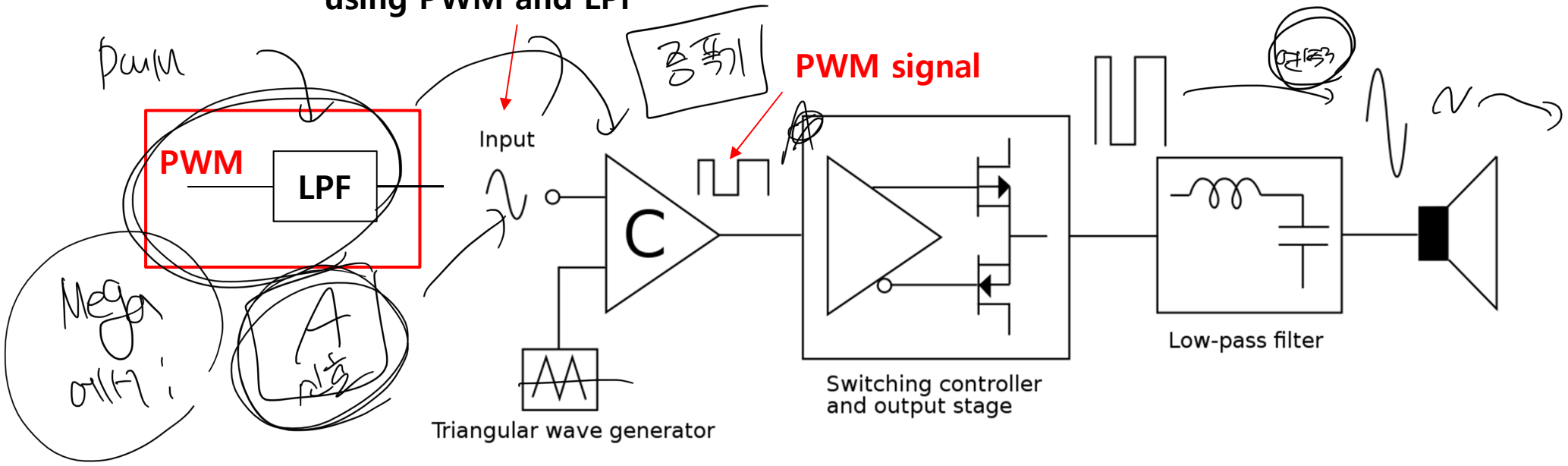


- How can we set the period ?
- How can we change the duty ratio ?

Class D Amplifier

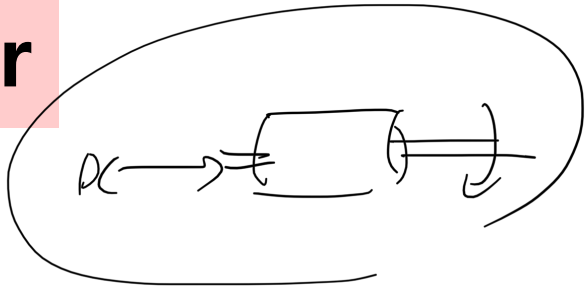
- **Class D Amplifier**
  - **Switching amplifier**
  - **High efficiency (over 90%)**

We will generate this analog signal using PWM and LPF

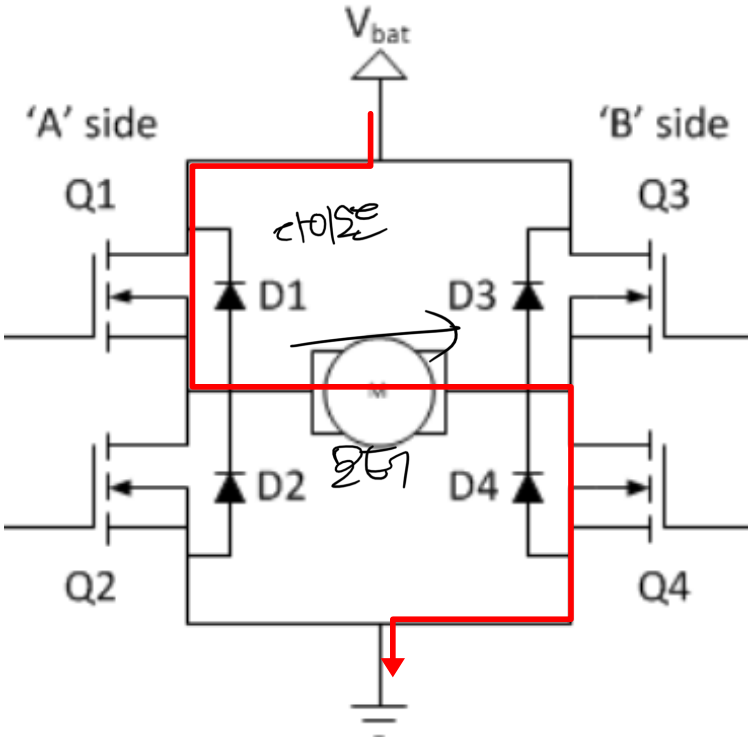


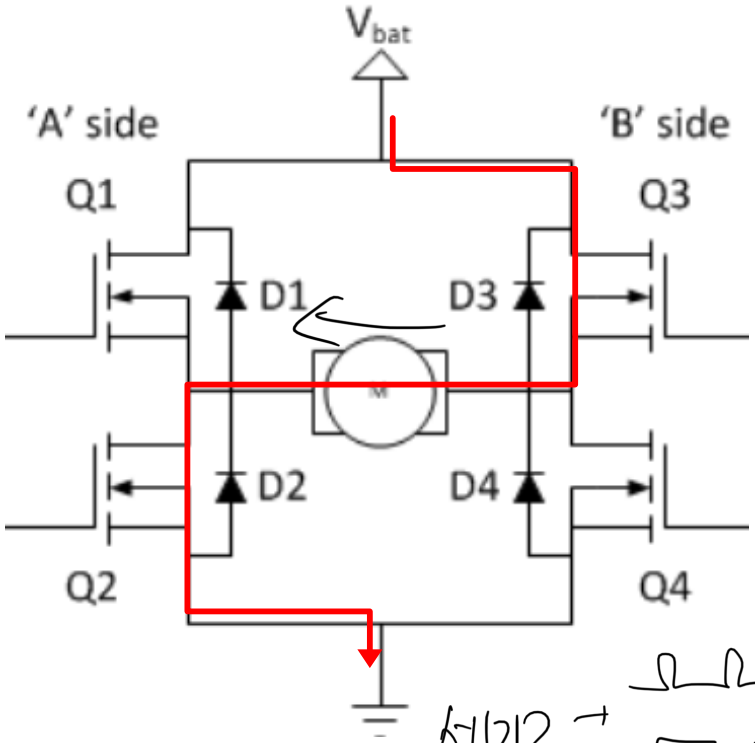
<Class D amplifier>

DC Motor Driver



- **H-bridge**
  - **It consists of 4 FETs and 4 free-wheeling diodes**
  - **PWM is normally used for switching FETs**

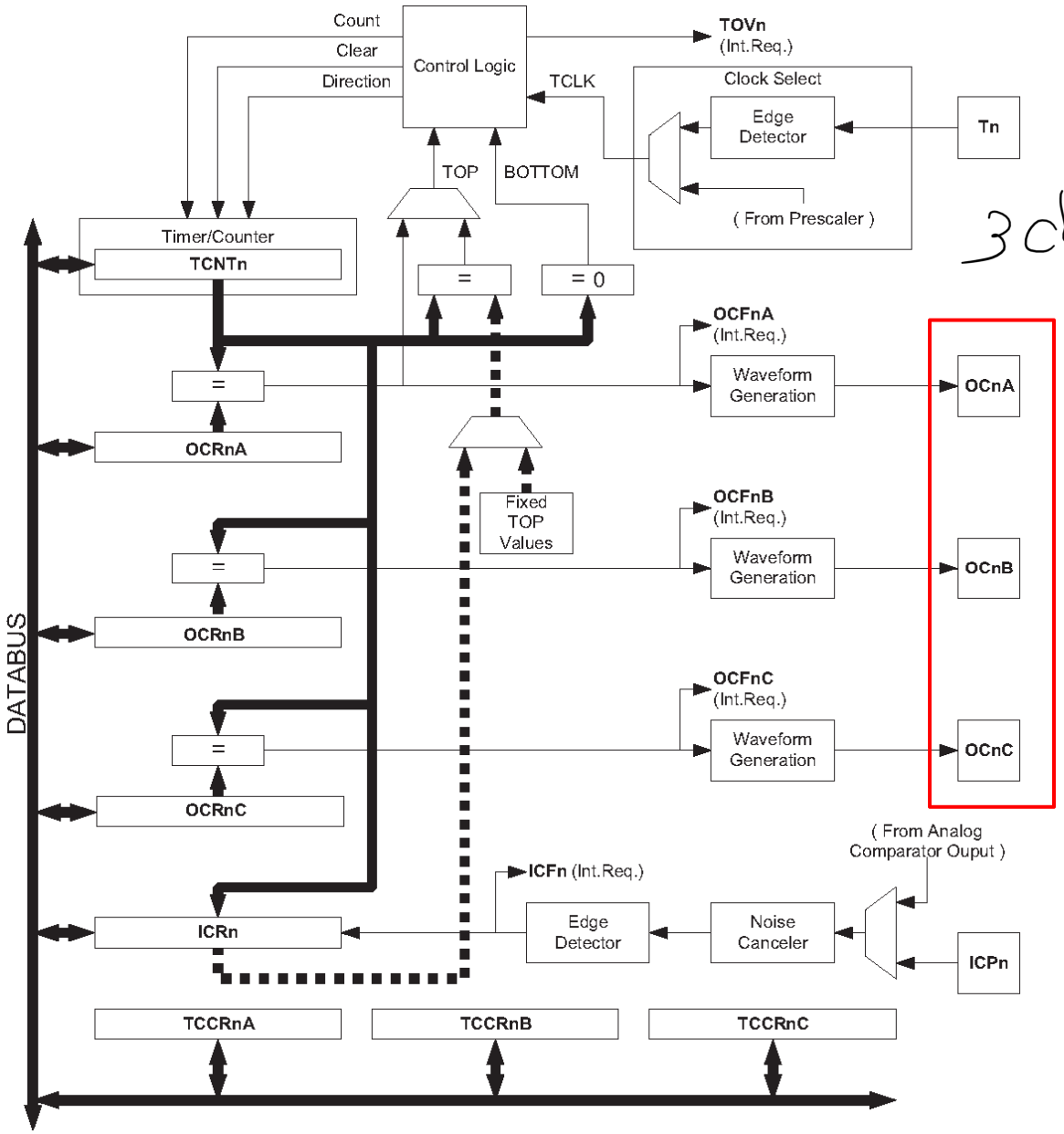




Q_1,Q_4=On, Q_2, Q_3=Off → Forward rotation

Q_1,Q_4=Off, Q_2, Q_3=On → Backward rotation

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



3 ch

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

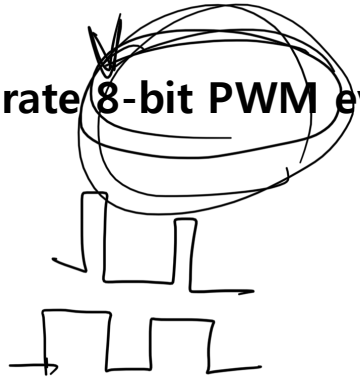
OC3B = PE4 = Pin 2 in Mega 2560

OC3C = PE5 = Pin 3 in Mega 2560

Which Counter/Which Mode?

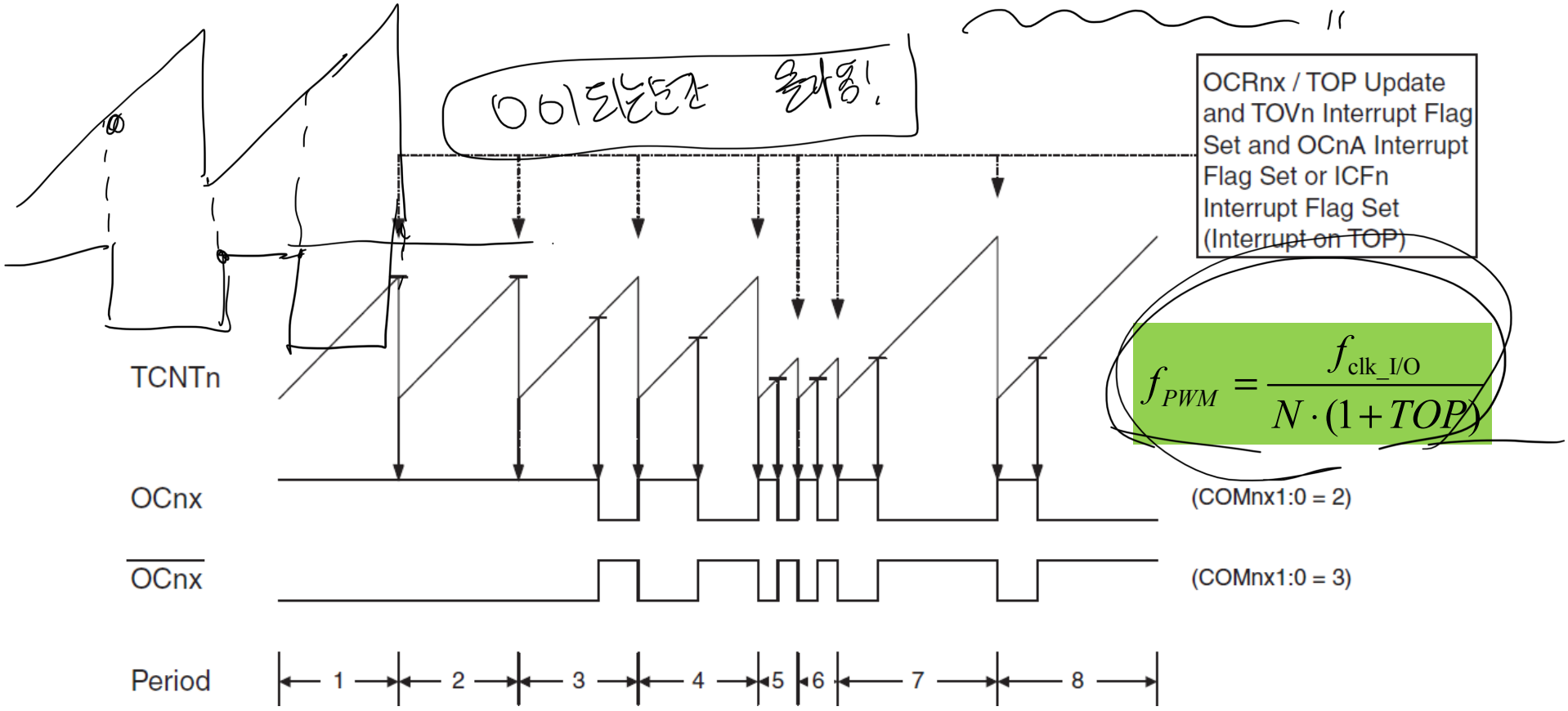
- **Timer/Counter Modes**
  - Normal mode
  - CTC mode (Clear Timer on Compare Match)
  - Fast PWM mode
  - Phase Correct PWM mode (PC PWM)
  - Phase and Frequency Correct PWM mode (PFC PWM)

- **Timer1**
  - We will configure Timer1 to generate 8-bit PWM even though it is a 16-bit timer/counter.



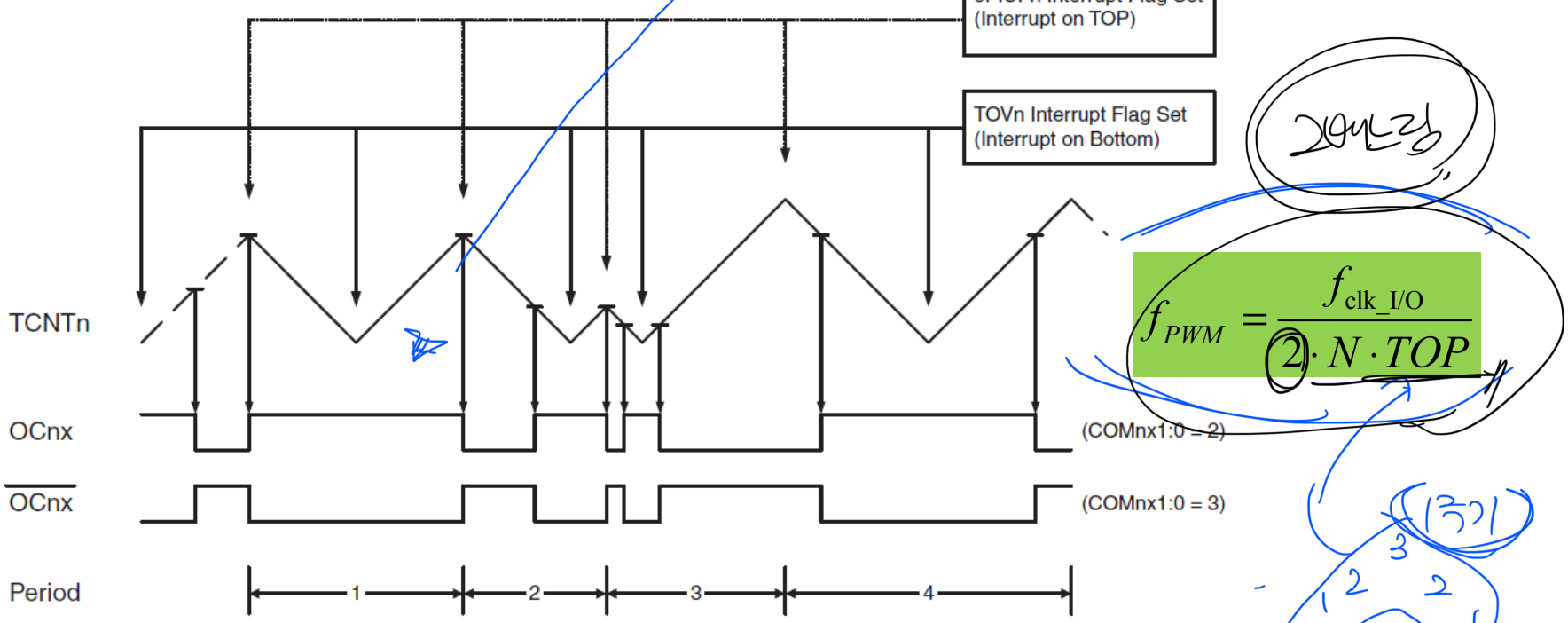
Fast PWM Mode

- **Fast PWM mode**
  - **It provides a high frequency PWM**
  - **It differs from the other PWM options by its single-slope operation**



Phase Correct PWM Mode

- **Phase Correct PWM Mode**
  - **It provides high resolution PWM**
  - **It is based on a dual slope operation**

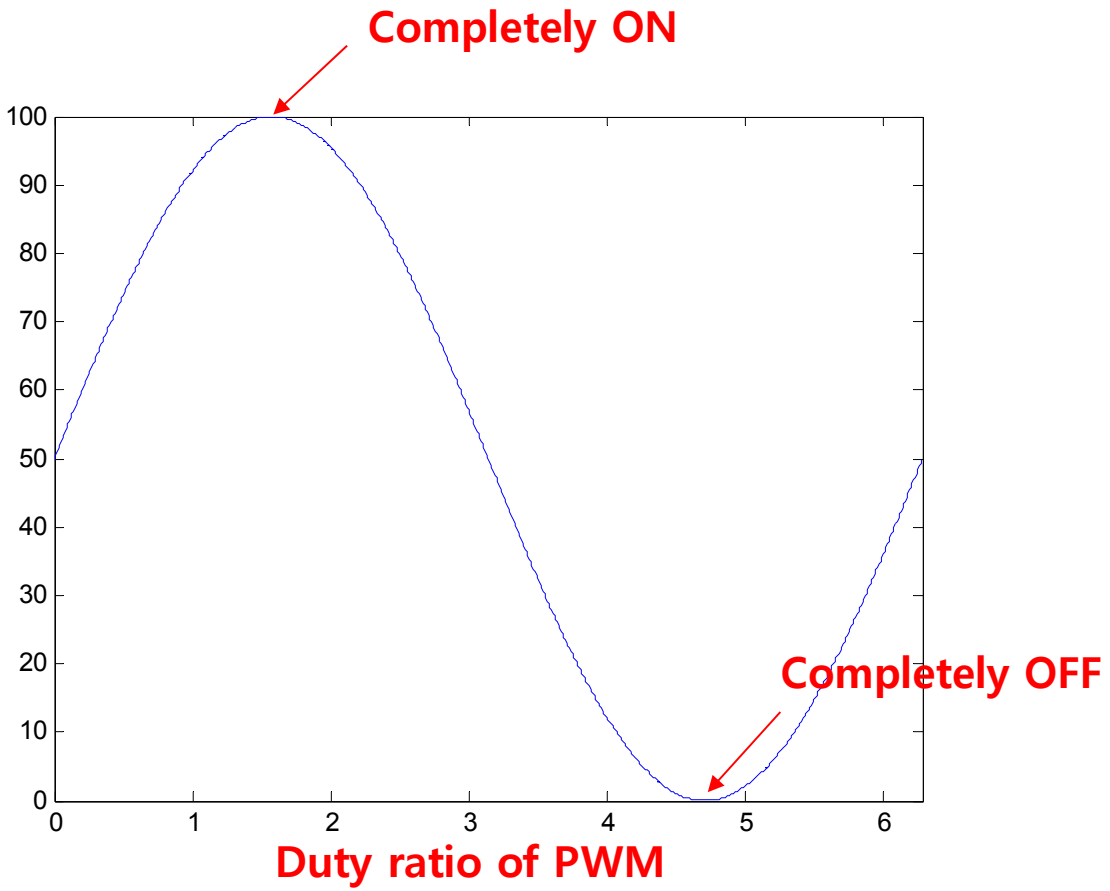


$$f_{PWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

[Note] Phase and frequency correct PWM mode is also based on dual slope operation.

LED Brightness Control using PWM

Problem: We want to change the brightness of an LED using PWM whose duty ratio is given below



Wire Connection

Arduino Pin #	Peripheral Board : IO_J3 (LED)
11 (PB5, OC1A)	LED1

Timer/counter mode selection

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

0101 : Fast PWM, 8-bit

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).

Clock Select Bit

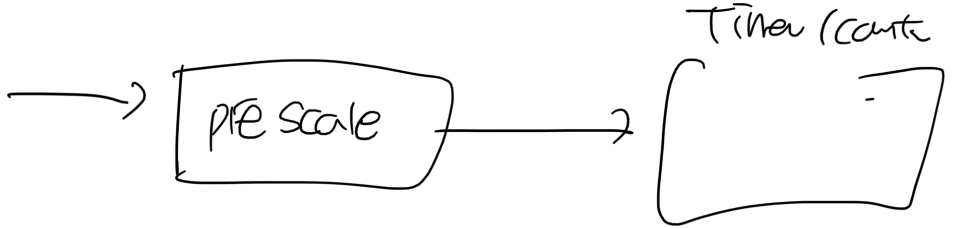


Table 14-6. Clock Select Bit Description

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

$$f_{PWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)} = \frac{16 \times 10^6}{64 \cdot (1 + 255)} = 976.5625\,[\text{Hz}]$$

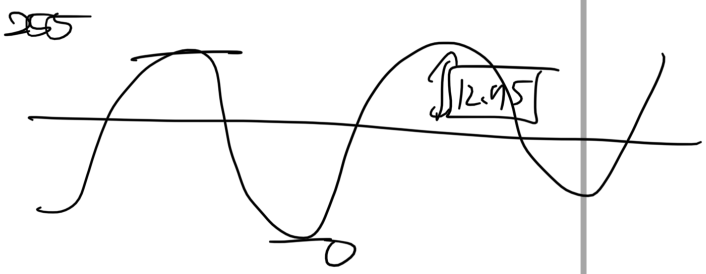
PwmLed.ino

PwmLed.ino (1)

```
//-----------------------------------------------------------------------
// File name : PwmLed.ino
// <Abstract>
// We control the brightness of an LED using PWM generated by Timer/Counter 1
//
// <Connection Info>
// ---------------------------------
//  Arduino Pin      Peripheral Board
// ---------------------------------
//      11 (PB5, OC1A)   LED1
// ---------------------------------
//
// Programmed by Prof. Lee on 18 Oct 2018.
//-----------------------------------------------------------------------
float sample_time;
float sim_time = 0.0;  // Simulation time

uint32_t start_time;         // time when the new iteration should begin
uint32_t MicrosSampleTime;   // Integer variable for the sampling time in microseconds.
uint16_t duty_value;         // Integer variable for stroing the data to be written on OCR1A
```

PwmLed.ino (2)

```
void setup()
{
    Serial.begin(115200);
    sample_time = 0.02;  // sampling time
    MicrosSampleTime = (uint32_t)(sample_time*1e6); // sampling time in microseconds

    DDRB |= 0b00100000;    // Set PB5 to be an output

    // Fast PWM Mode, 8-bit  WMG13,WGM12,WGM11,WGM10 = 0101
    TCCR1B &= ~_BV(WGM13);  // WGM13=0
    TCCR1B |= _BV(WGM12);   // WGM12=1
    TCCR1A &= ~_BV(WGM11);  // WGM11=0
    TCCR1A |= _BV(WGM10);   // WGM10=1

    // Clear OC1A on compare match
    TCCR1A |= _BV(COM1A1);   // COM1A1=1
    TCCR1A &= ~_BV(COM1A0);  // COM1A0=0

    // Prescaler : clk_io/64
    TCCR1B &= ~_BV(CS12);    // CS12=0
    TCCR1B |= _BV(CS11);     // CS11=1
    TCCR1B |= _BV(CS10);     // CS10=1

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

    start_time = micros() +  MicrosSampleTime; // time when the new iteration should begin
}

void loop()
{
    sim_time += sample_time;  // update of the simulation time
    duty_value = (uint16_t)(127.5*sin(2*sim_time) + 127.5);  // computation of duty value using sin function
    OCR1A = duty_value;              // writing the duty value on OCR1A register to change the duty ratio
    Serial.println(duty_value);// Sending duty_value to the serial plotter

    while(!((start_time-micros()) & 0x80000000)); // Check whether the sample time has elapsed.
    start_time += MicrosSampleTime;               // Update of the start_time
}
```



Serial Plotter Graph