ECE3411 – Spring 2019

Lecture 9

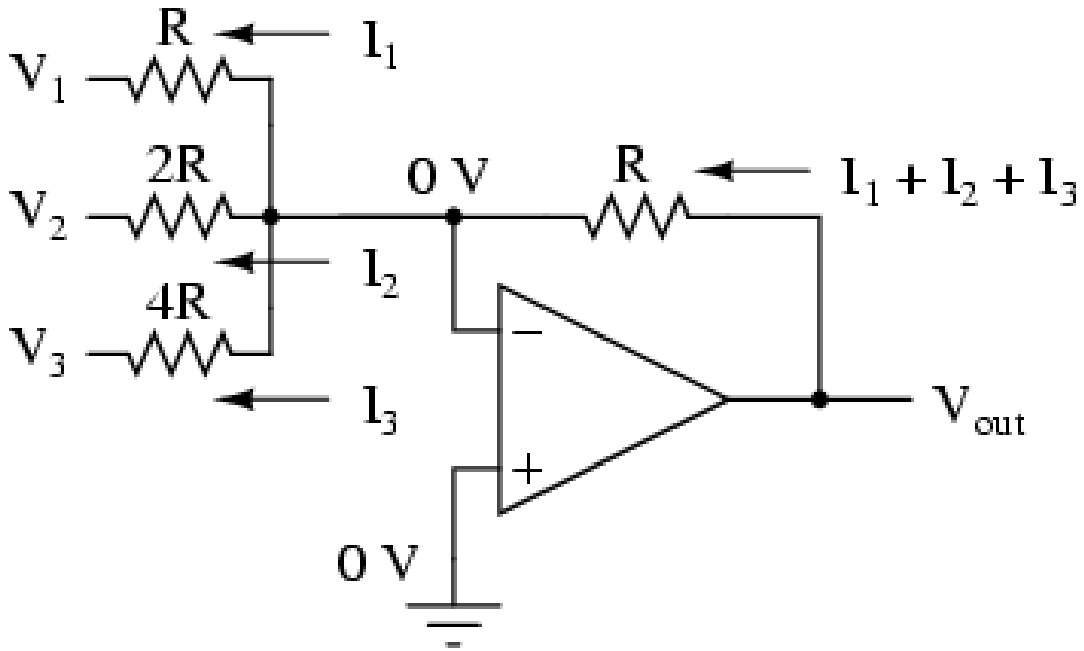
PWM: Pulse Width Modulation

Sung Yeul Park
Department of Electrical & Computer Engineering
University of Connecticut
Email: sung_yeul.park@uconn.edu

Copied from Lecture 3b, ECE3411 – Fall 2015, by
Marten van Dijk and Syed Kamran Haider

Based on the Atmega328PB datasheet

DAC: Binary weighted input



$$V_{out} = -\left(V_1 + \frac{V_2}{2} + \frac{V_3}{4}\right)$$

Microcontroller DAC

- Can not use analog techniques
- Use digital techniques to generate a sequence of pulses - pulse width modulation (PWM)
- Pass through a low-pass filter to generate the analog signal

Pulse Width Modulation

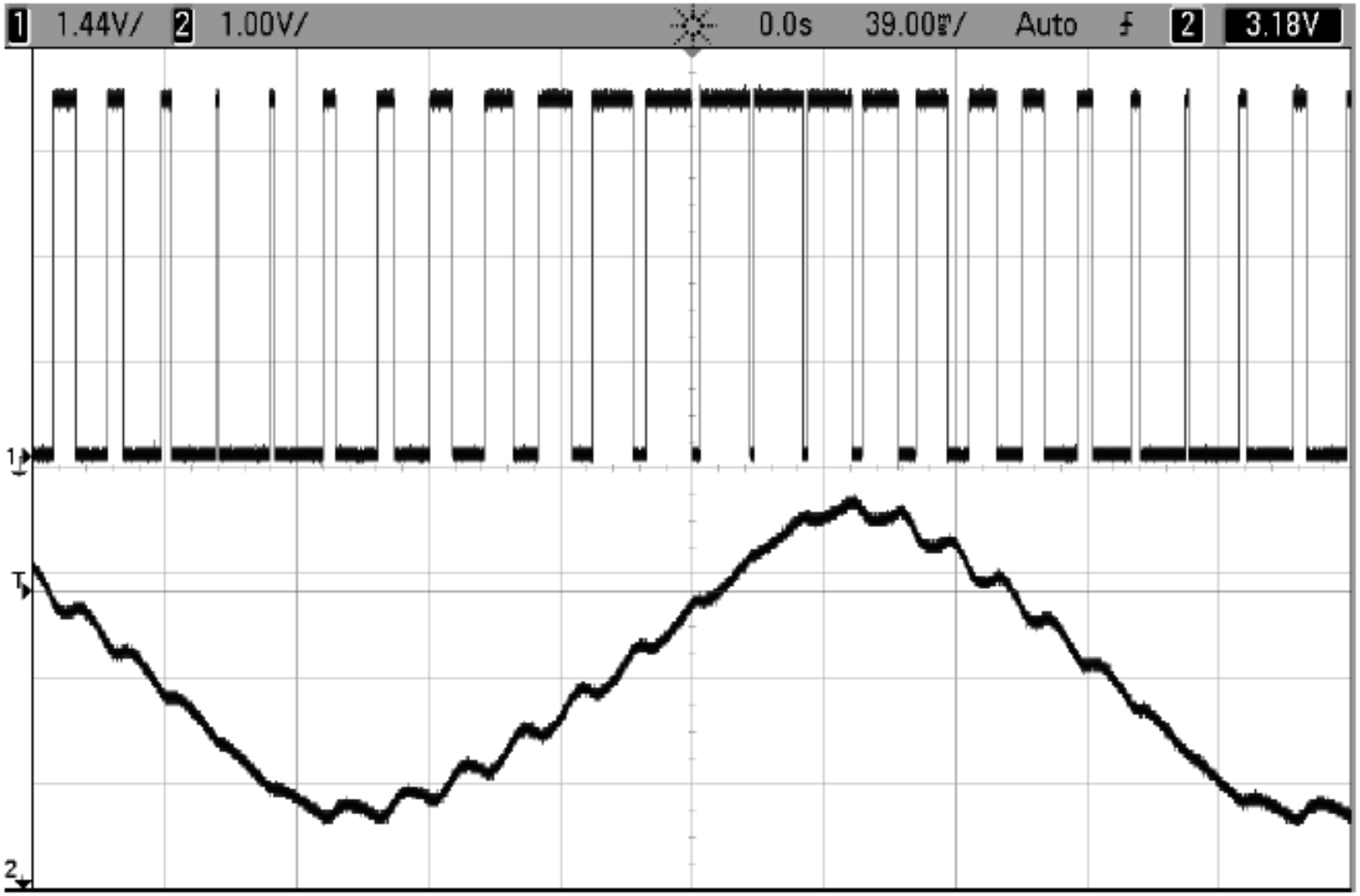
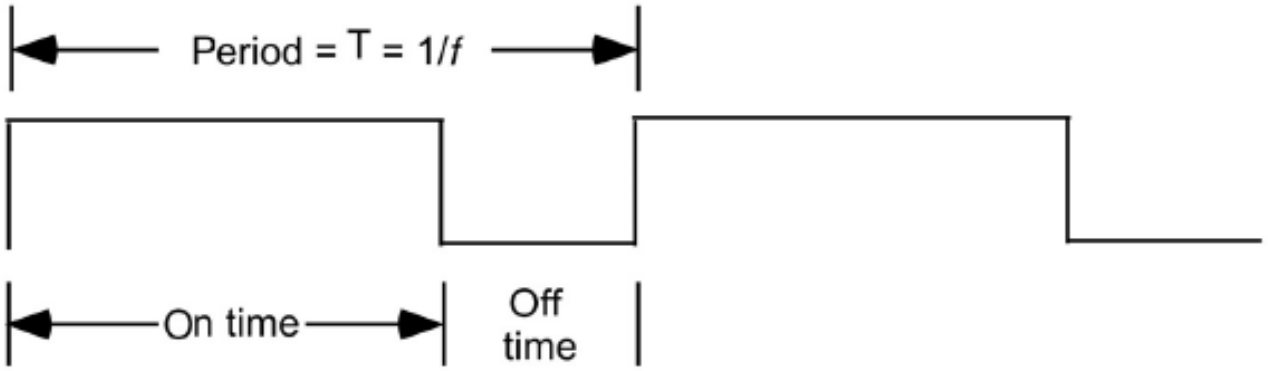


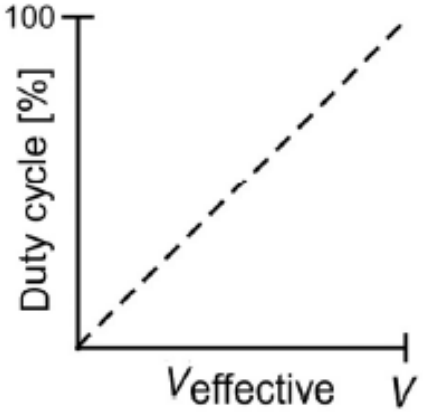
Figure 10-1. *PWM oscilloscope traces*

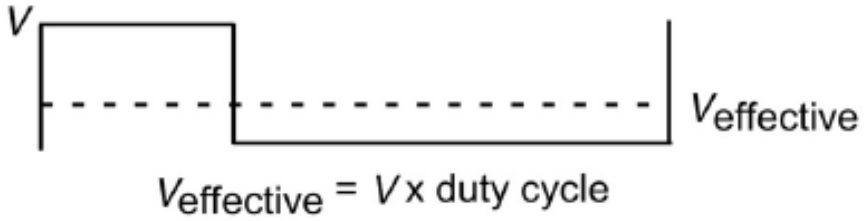
Pulse Width Modulation



Duty cycle = (On time/period) x 100%

(a) Signal parameters

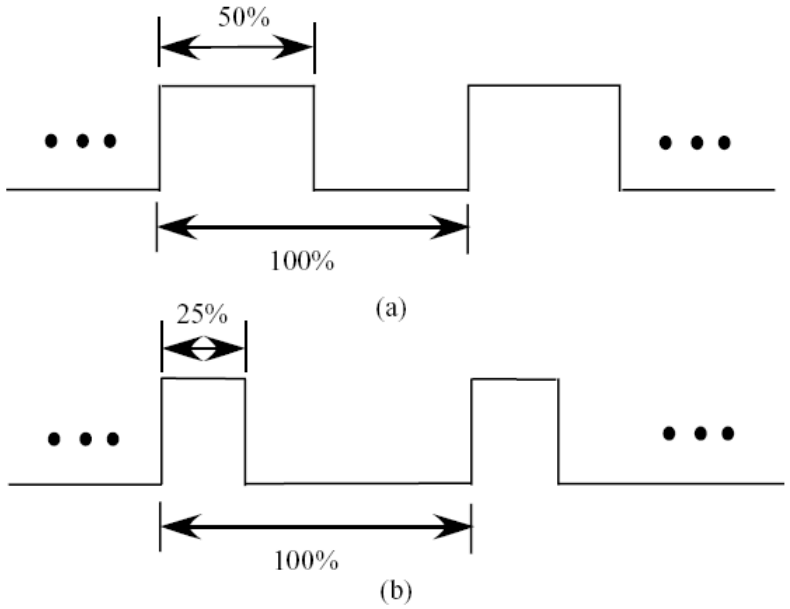


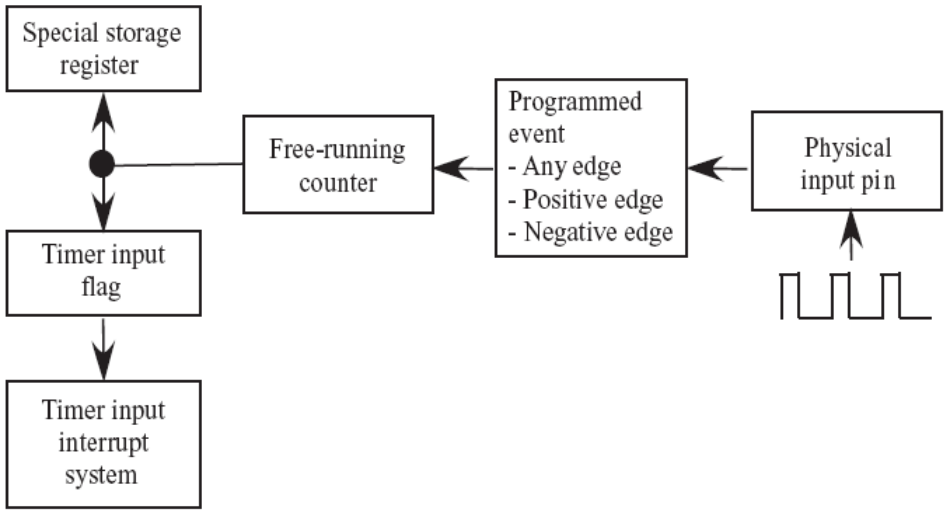


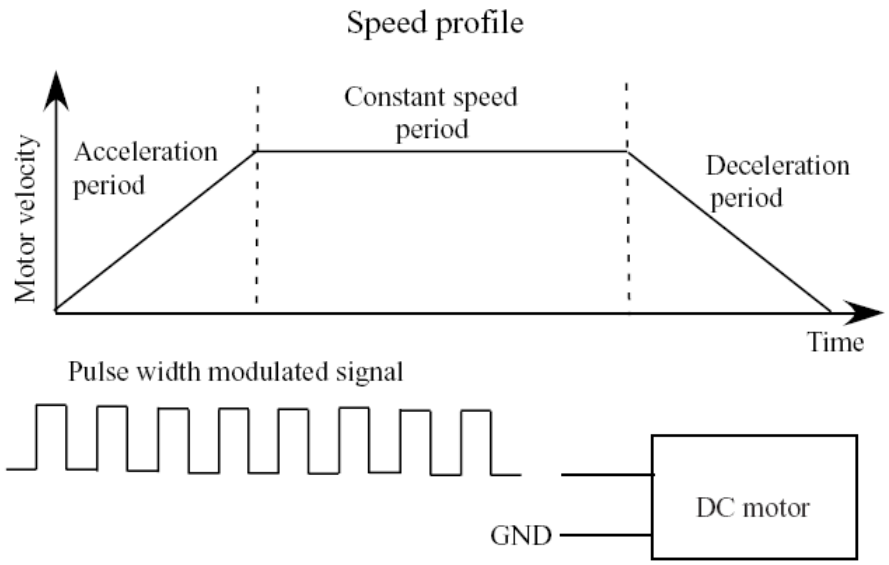
(b) Pulse width modulation (PWM) concepts

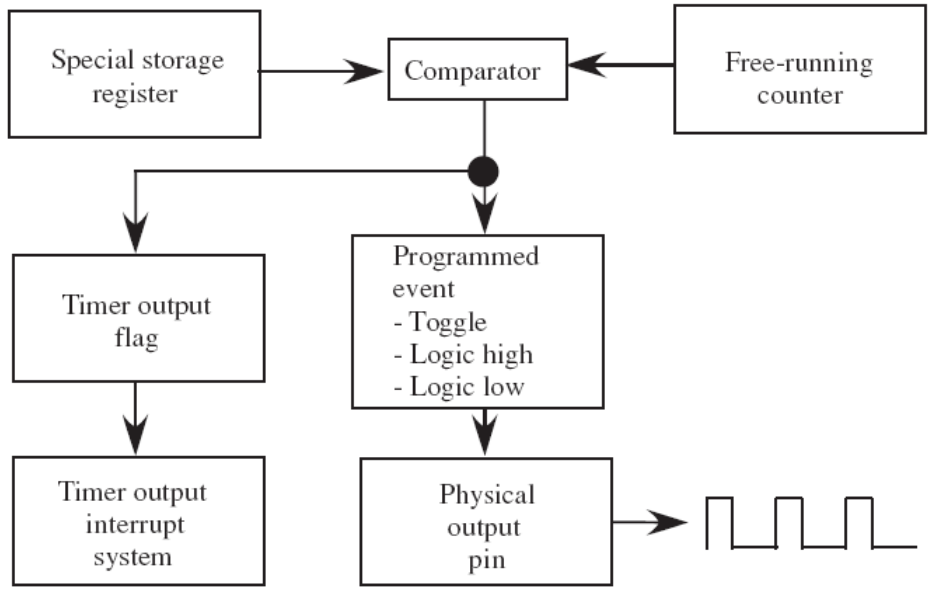
- A pulse width modulated or PWM signal is characterized by a fixed frequency and a varying duty cycle.
- A duty cycle is defined as the percentage of time a periodic signal is logic high over the signal period.

PWM Duty Ratio and its Usage

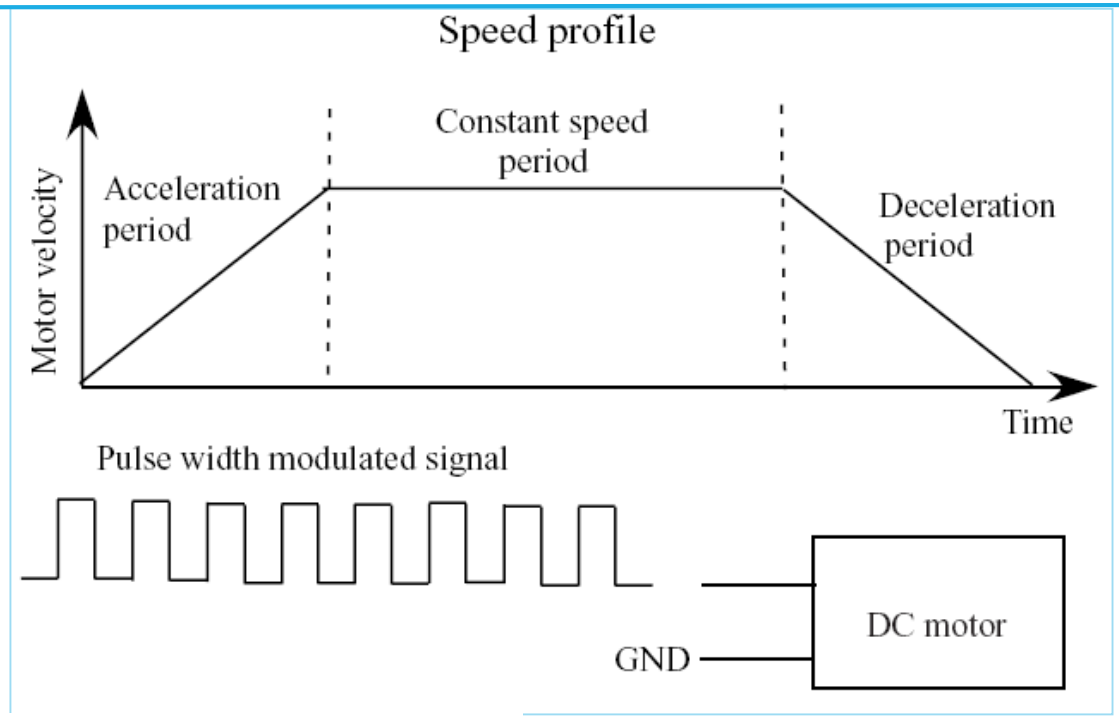


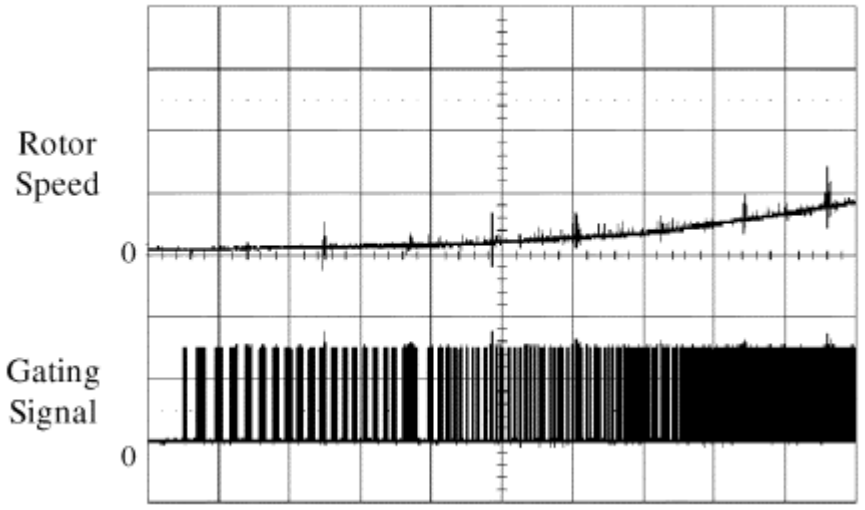


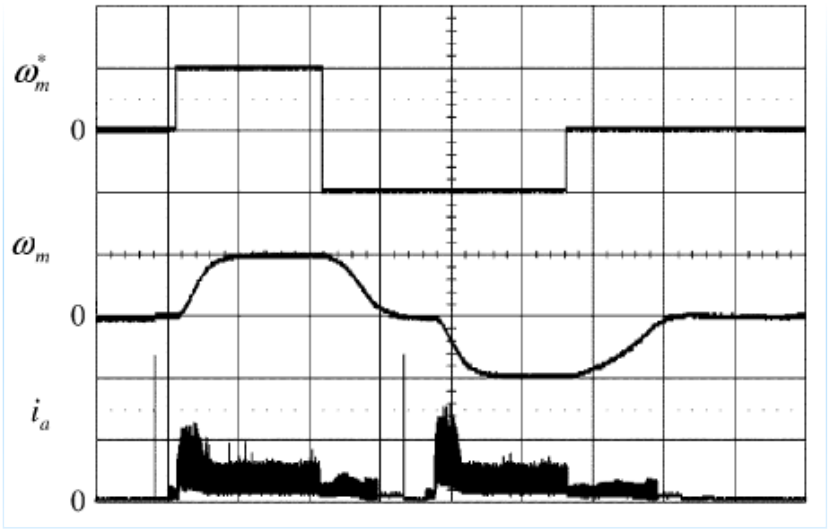




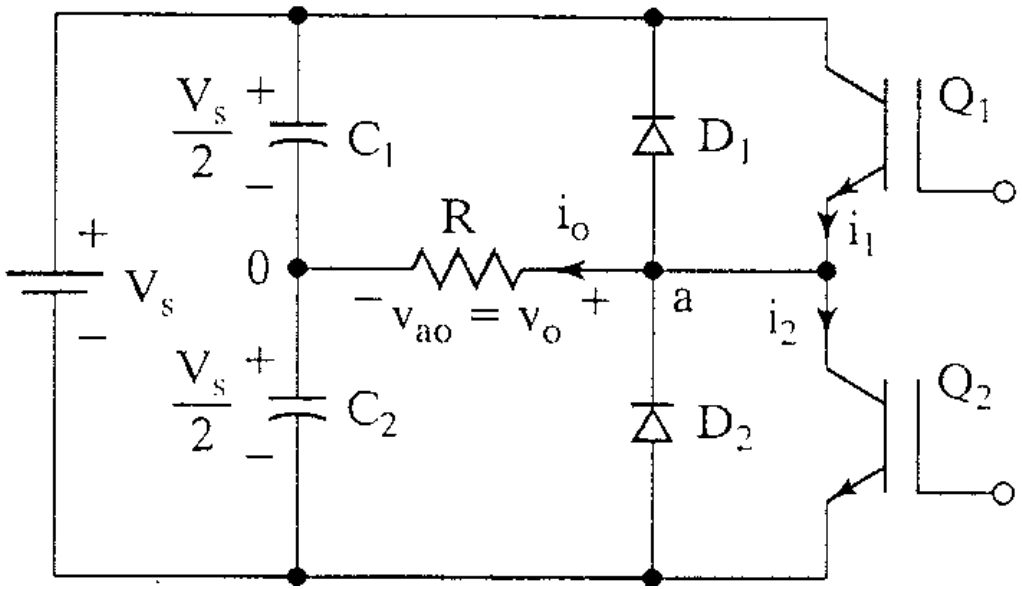
PWM Applications(1) : Motor Control

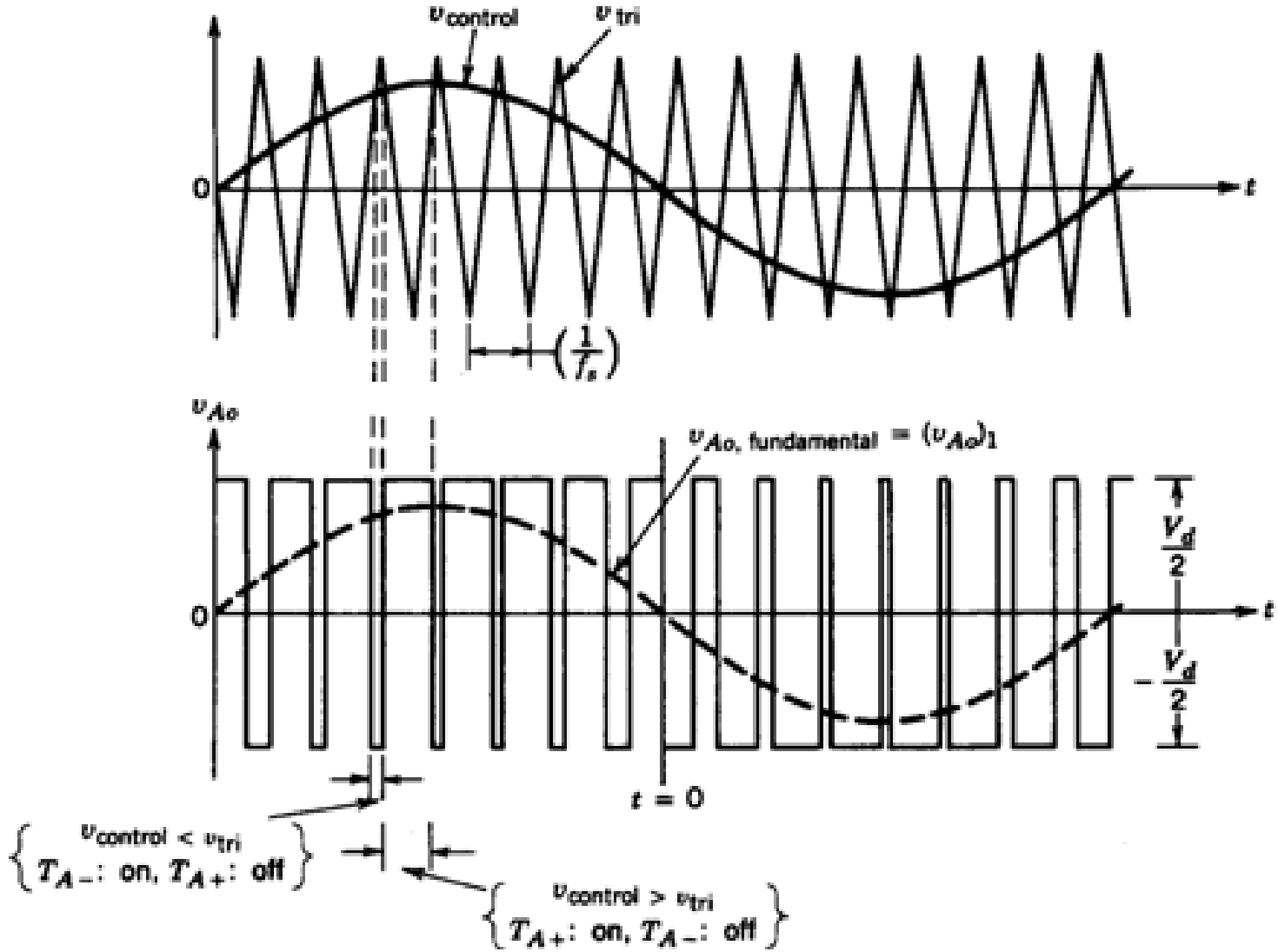




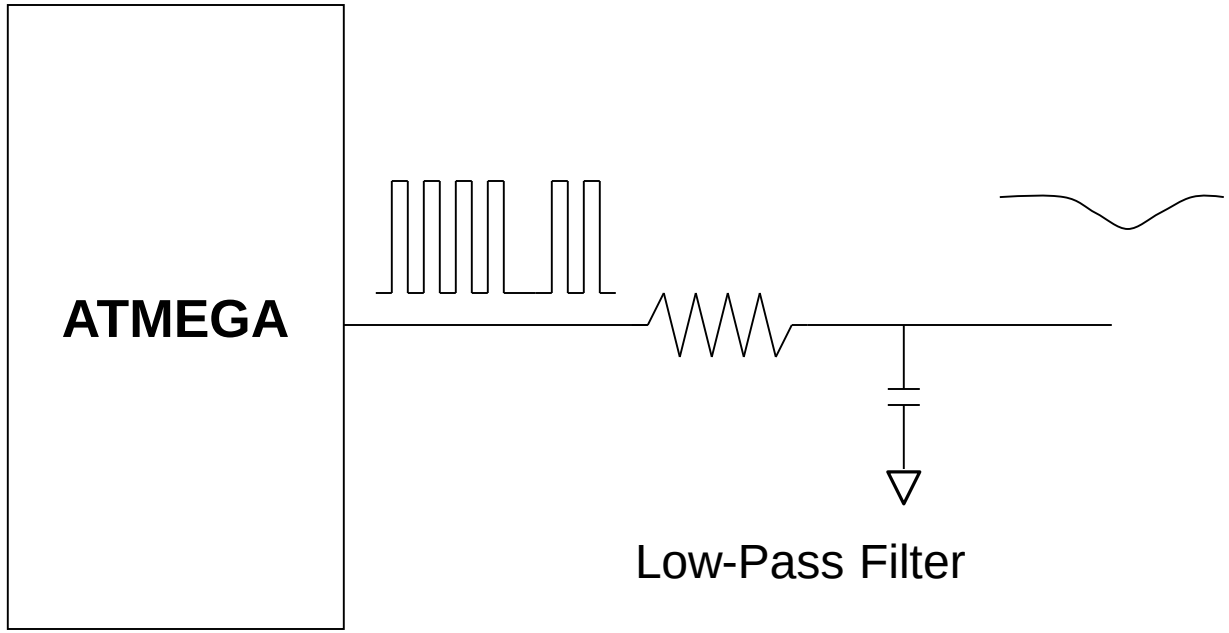


PWM Applications(2) : Power Converter





Pulse Width Modulation



- 16-bit 44KHz compact disk(CD) sampling would require a 2.9GHz clock

Pulse Width Modulation

- If duty cycle is d, the width of each pulse is Td, where T is the length of each period. T is the inverse of the switching frequency.
- Resolution of each pulse is determined by the f_{CLK}
- If you want 8-bit resolution, that means you need 256 possible pulse widths. Thus, the smallest T you can have is $\frac{256}{f_{CLK}}$.
- More generally, the highest possible switching frequency is $\frac{f_{CLK}}{2^n}$ for n–bit resolution
- 16-bit 44KHz CD sampling would require a 2.9GHz clock

TCCR0A

TC0 Control Register A

Name: TCCR0A

Bit	7	6	5	4	3	2	1	0
	COM0A[1:0]		COM0B [1:0]				WGM0[1:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Mode	WGM0[2]	WGM0[1]	WGM0[0]	Timer/Counter Mode of Operation	TOP	Update of OCR0x at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCR0A	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	-	-	-
5	1	0	1	PWM, Phase Correct	OCR0A	TOP	BOTTOM
6	1	1	0	Reserved	-	-	-
7	1	1	1	Fast PWM	OCR0A	BOTTOM	TOP

Timer 1

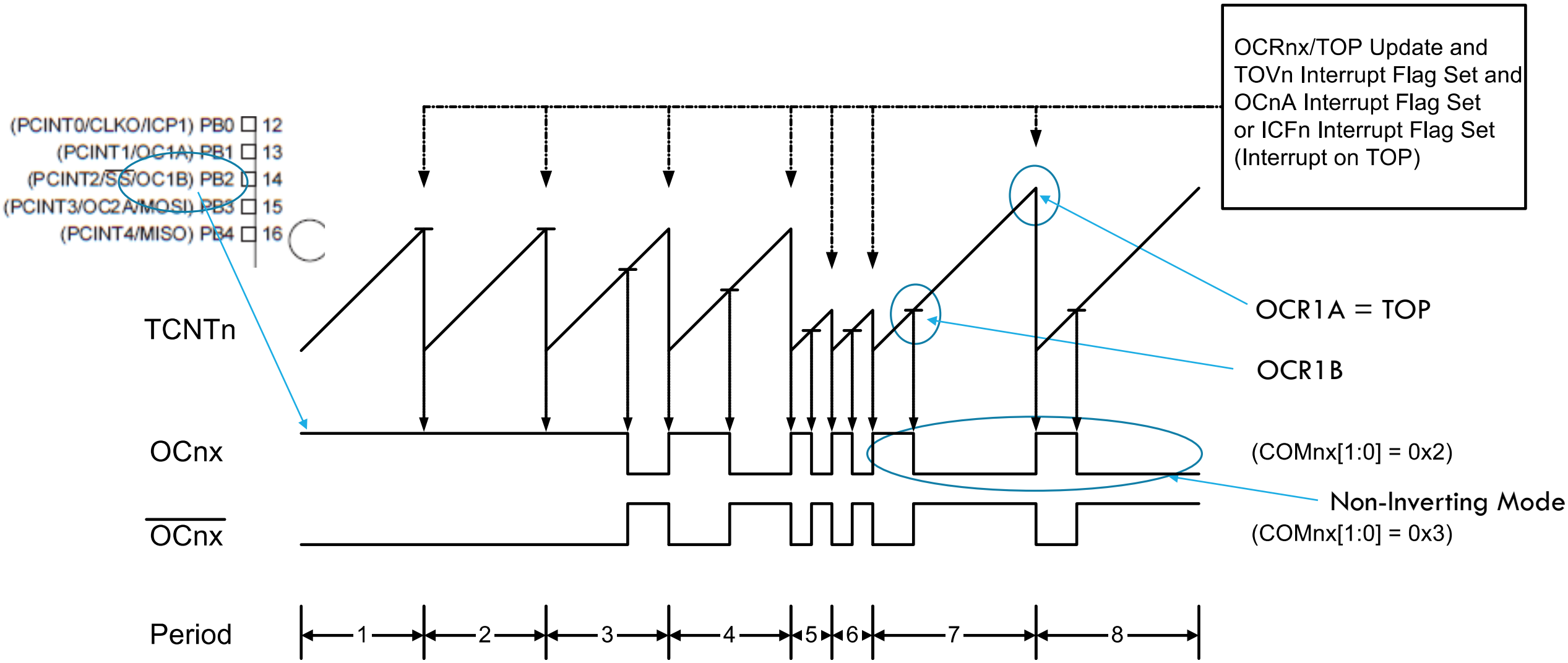
Mode	WGM1[3]	WGM1[2] (CTC1)(1)	WGM1[1] (PWM1[1])(1)	WGM1[0] (PWM1[0])(1)	Timer/ Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	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	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	-	-	-
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Compare Output Mode

COM1A[1]/ COM1B[1]	COM1A[0]/ COM1B[0]	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM1[3:0] = 14 or 15: Toggle OC1A on compare match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at BOTTOM (Non-inverting mode)
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at BOTTOM (Inverting mode)

Pulse Width Modulation using Timer 1

Figure 19-6. Fast PWM Mode, Timing Diagram



Frequency & Duty Cycle

- $F_{\{OC1B\}} = \frac{F_{\{CPU\}}}{prescalar * (1+OCR1A)}$
- $DutyCycle = \frac{1+OCR1B}{1+OCR1A}$

For example, frequency OCB1 at 523 Hz with F_CPU = 16 MHz and no prescalar gives OCR1A = 30592

Interrupts

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	

1 1

Allows one to play music!

- Modulate duty cycle in ISR (change OCR1B)
- Modulate frequency in ISR (change OCR1A)

Interrupt Vectors

- **Bit 2 – OCIE1B: Timer/Counter1, Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector (see “Interrupts” on page 57) is executed when the OCF1B Flag, located in TIFR1, is set.

- **Bit 1 – OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector (see “Interrupts” on page 57) is executed when the OCF1A Flag, located in TIFR1, is set.

Easier Solution for 50% Duty Cycle

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
	0	1					0	0	

Bit	7	6	5	4	3	2	1	0	
(0x81)	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	
				0	1	0	0	1	

- Mode 4 in Table 15.4: CTC Mode for Waveform Generation
- No prescalar
- Toggle OC1A on Compare Match

Table 19-3. Compare Output Mode, Non-PWM

COM1A[1]/ COM1B[1]	COM1A[0]/ COM1B[0]	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on compare match.
1	0	Clear OC1A/OC1B on compare match (Set output to low level).
1	1	Set OC1A/OC1B on compare match (Set output to high level).

Updating Frequency

- When changing frequencies in ISR(TIMER1_COMPA_vect) by modifying OCR1A:
- Be careful not to modify OCR1A to a value <TCNT1
- Otherwise, TCNT1 cycles through to $2^{16} - 1$
- This may create a glitch!

Example Calculations

- Assume a clock frequency of F_CPU=20 MHz.
- In non-inverting fast PWM mode, configure a PWM output signal on pin OC1B so that it is active high for 1/3 of a 24ms period:
- Which prescalar for the timer do you use? Given the prescalar of your choice, at what values should you set OCRnA and OCRnB?

Example Calculations

- Prescalar of 1
- 1/20MHz = 50 ns / TCNT1 tick.
- 24ms = 24ms/50s TCNT1 ticks, i.e., 480,000 TCNT1 ticks.
- Too big for OCR1A, so prescalar of 1 does not work

Example Calculations

- Prescalar of 8
- 8/20MHz = 0.4μs/ TCNT1 tick
- 24ms = 24ms/0.4us TCNT1 ticks, i.e., 60,000 TCNT1 ticks.
- By setting OCR1A = 59999 we get a PWM of period 24 ms
- By setting OCR1B = 19999 we get a duty cycle of (OCR1B+1)/(OCR1A+1) = 1/3.

Example Calculations

- Prescalar of 64
  - 64/20MHz = 3.2μs / TCNT1 tick
  - 24ms = 24ms/3.2us TCNT1 ticks, i.e., 7,500 TCNT1 ticks
  - By setting OCR1A = 7499 we get a PWM of period 24 ms
  - By setting OCR1B = 2499 we get a duty cycle of (OCR1B+1)/(OCR1A+1) = 1/3.
- Prescalar of 256
  - 256/20MHz = 12.8μs / TCNT1 tick
  - 24ms = 24ms/12.8us TCNT1 ticks, i.e., 1,875 TCNT1 ticks
  - By setting OCR1A = 1874 we get a PWM of period 24 ms
  - By setting OCR1B = 624 we get a duty cycle of (OCR1B+1)/(OCR1A+1) = 1/3.

Example Calculations

- Prescalar of 1024
- 256/20MHz = 51.2μs / TCNT1 tick
- 24ms = 24ms/51.2us TCNT1 ticks, i.e., 468.75 TCNT1 ticks
- By setting OCR1A = 468 we get a PWM of period ~24 ms (but not exact)
- By setting OCR1B = 155 we get a duty cycle of (OCR1B+1)/(OCR1A+1) approximately 1/3.

Example Calculations

- Besides registers OCRnA and OCRnB which other registers *need* to be set in order to have pin B2 output the PWM signal?
- Solution

```
DDRB = 0x04; // pin PB2 = OCB1 is an output
TCCR1A = (1<<COM1B1)|(1<<WGM11)|(1<<WGM10);
// non-inverting, fast PWM with TOP = OCR1A
TCCR1B = (1<<WGM13)|(1<<WGM12)|(1<<CS11);
// fast PWM with TOP = OCR1A, prescalar = 8
```