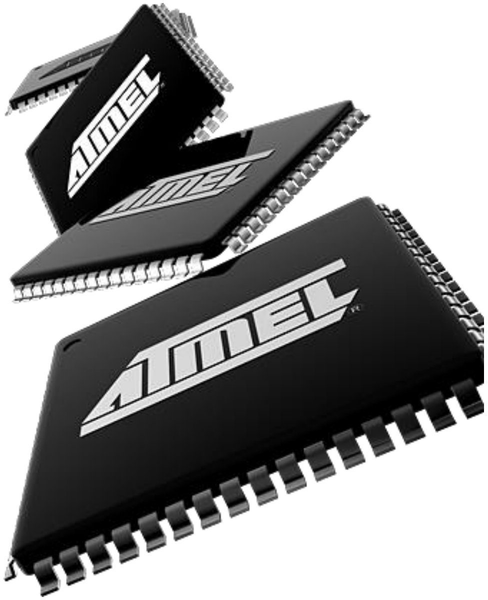
Principles and Applications of Microcontrollers

Yan-Fu Kuo
Dept. of Biomechatronics Engineering
National Taiwan University

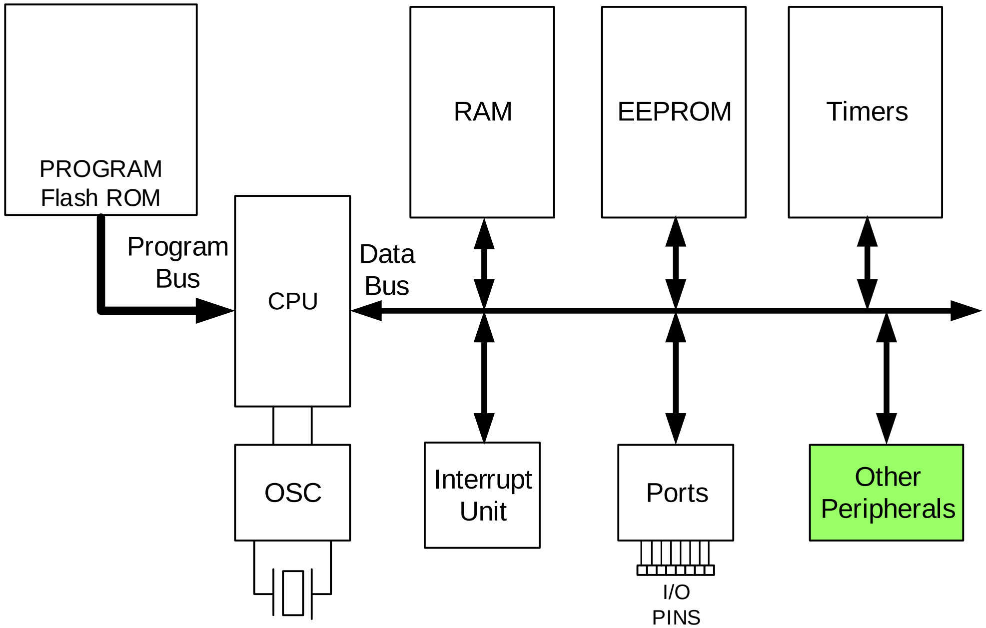
Today:

- Pulse width modulation (PWM)



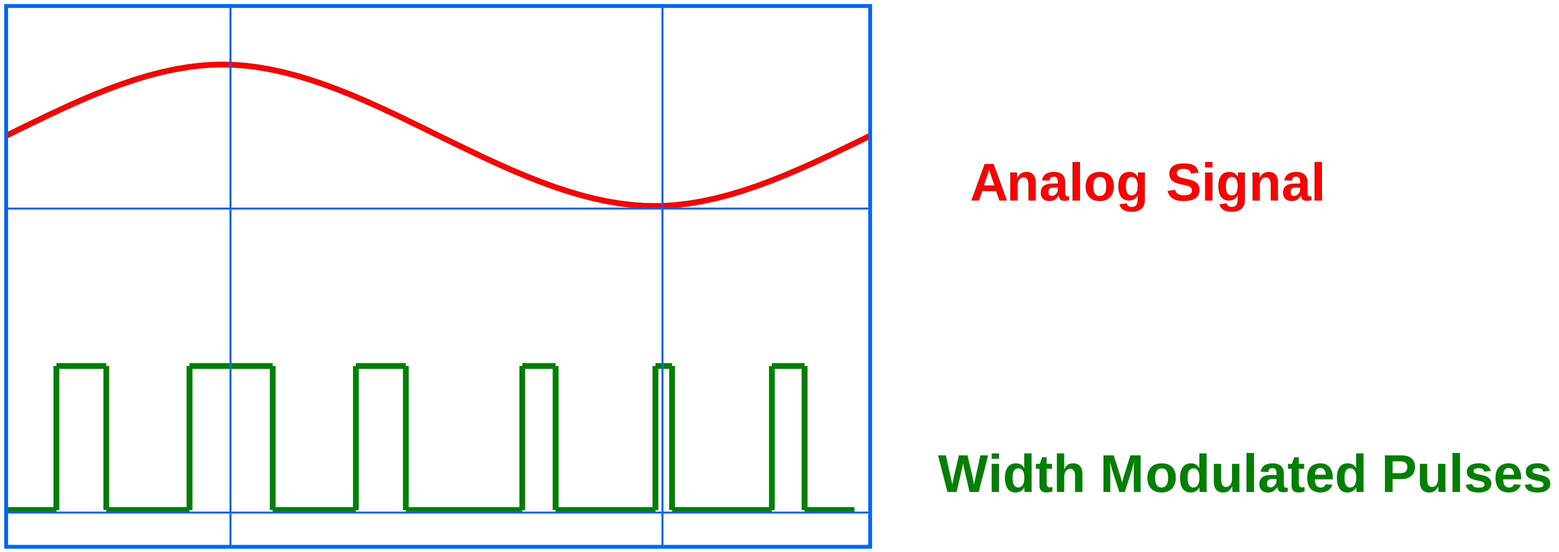
Outline

- Pulse width modulation (PWM)
  - What is PWM?
  - AVR PWM pinout
  - Fast PWM
  - Phase correct PWM
  - Square wave
  - Duty cycle and frequency
- Getting started



Pulse Width Modulation

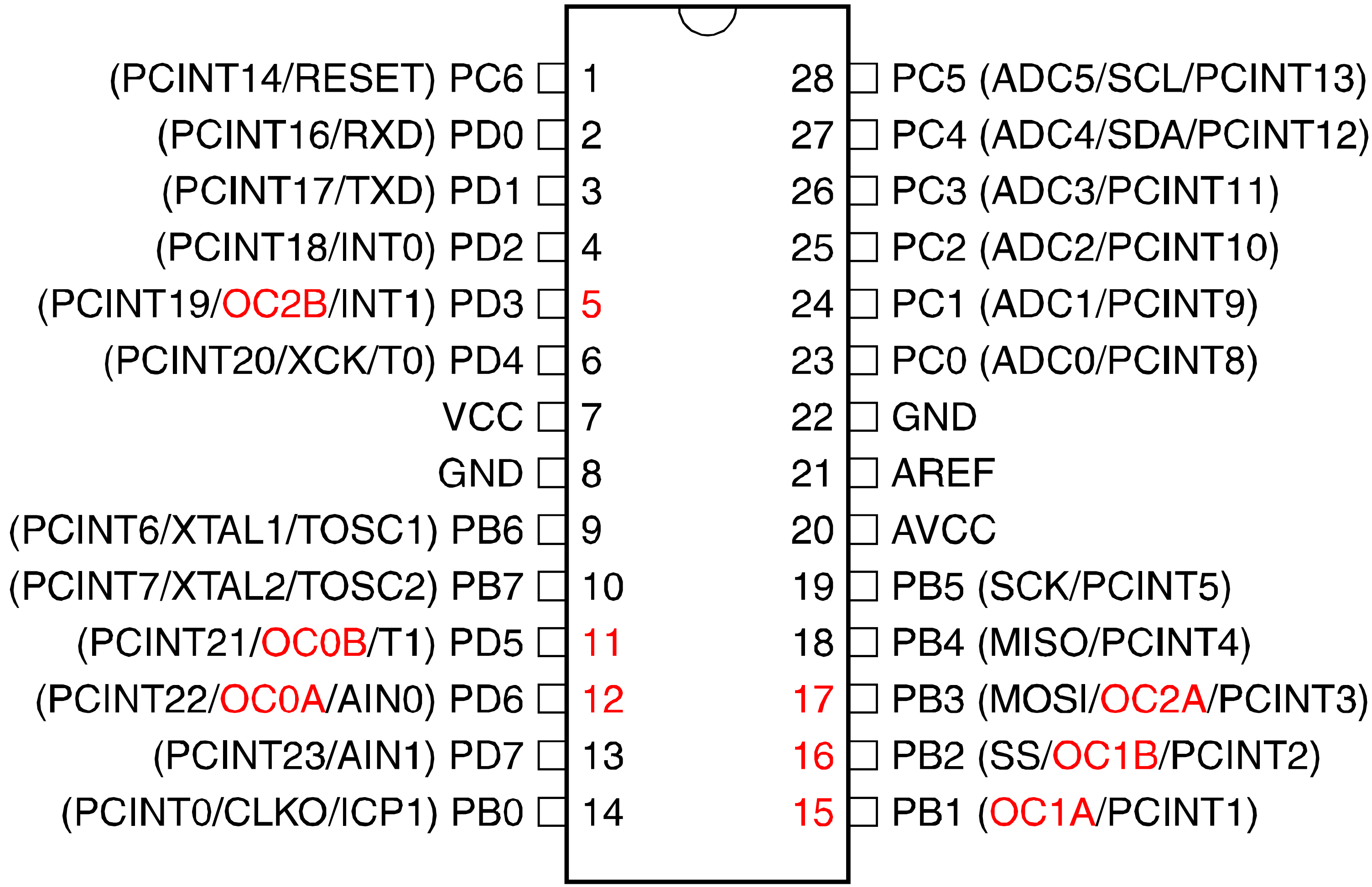
- An approach of approximating analog signal in the form of pulses (binary signal)



Advantages and Application

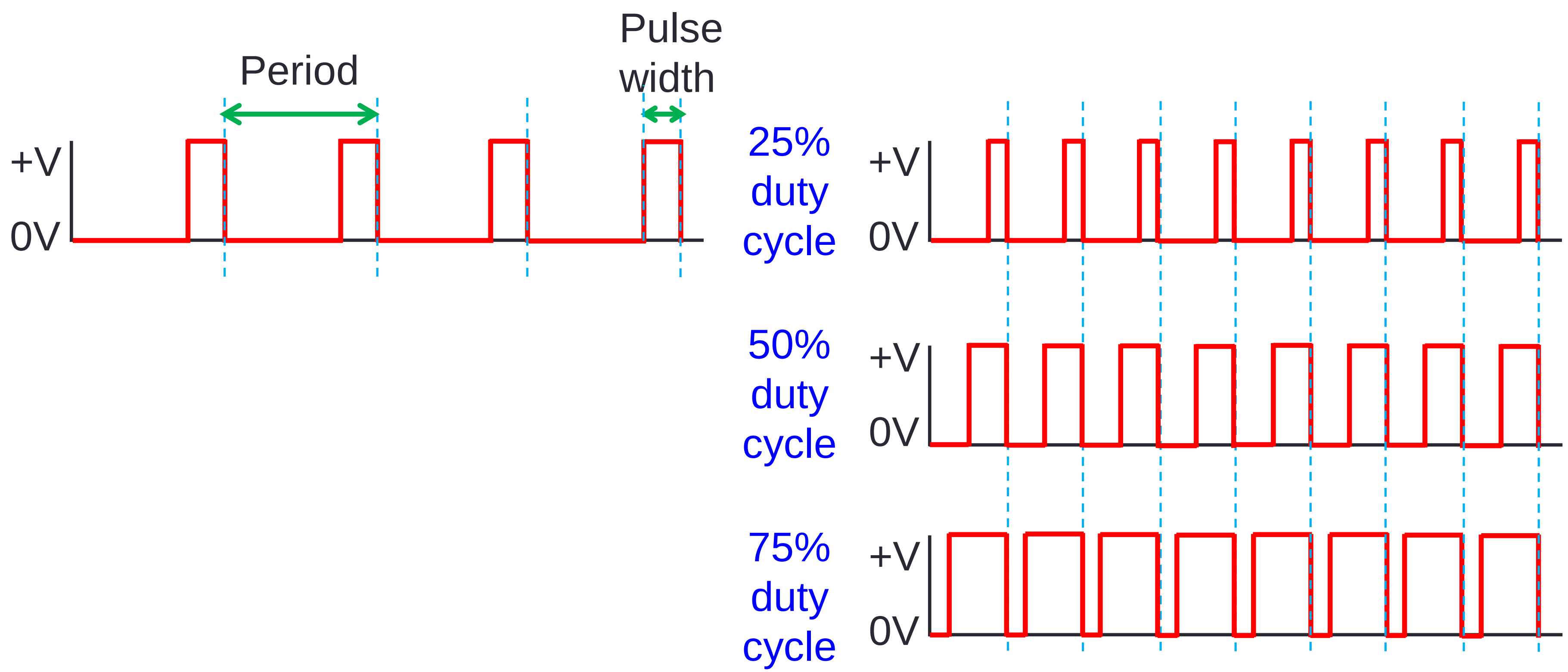
- Advantage:
  - Providing control on output power of a binary output device without changing hardware
  - Digital signal is resistant to noise
  - Less heat dissipated versus using resistors for intermediate voltage values
- Application:
  - Electric stove heater
  - Lamp dimmers
  - Voltage regulation – convert 12 volts to 5 volts by having a 41.7% duty cycle
  - Sound production

PWM Pinout


(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

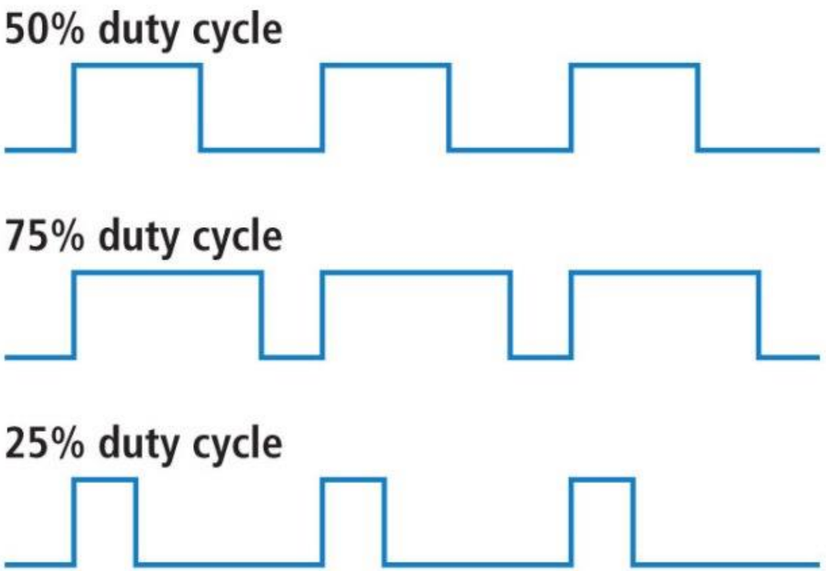
Factors to Consider

1. Duty cycle $= \frac{Pulse\ Width}{Period}$
2. Frequency



Outline (Cont'd)

- Pulse width modulation (PWM)
  - What is PWM?
  - AVR PWM pinout
  - Fast PWM
  - Phase correct PWM
  - Square wave
  - Duty cycle and frequency
- Getting started



PWM Mode Control: TCCR0A/B

TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
TCCR0B	FOC0A	FOC0B	-	-	WGM02	**CS02**	**CS01**	**CS00**

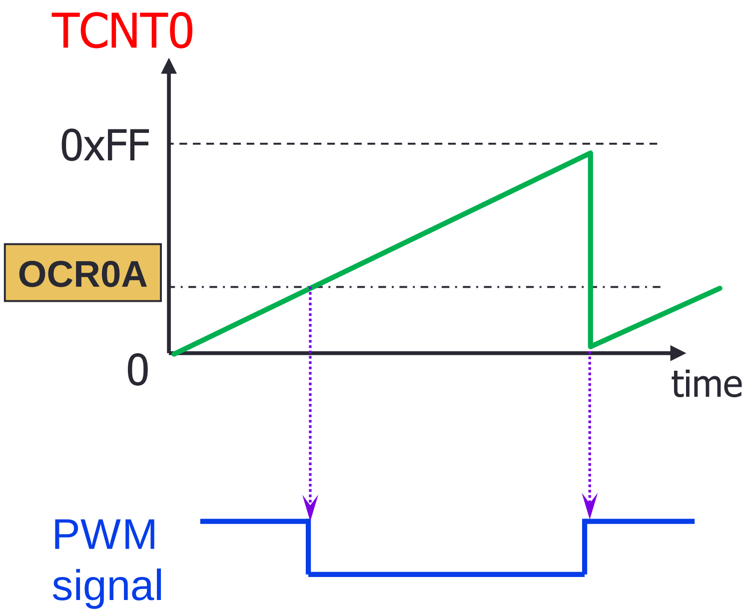
Timer Mode (WGM)

- Re

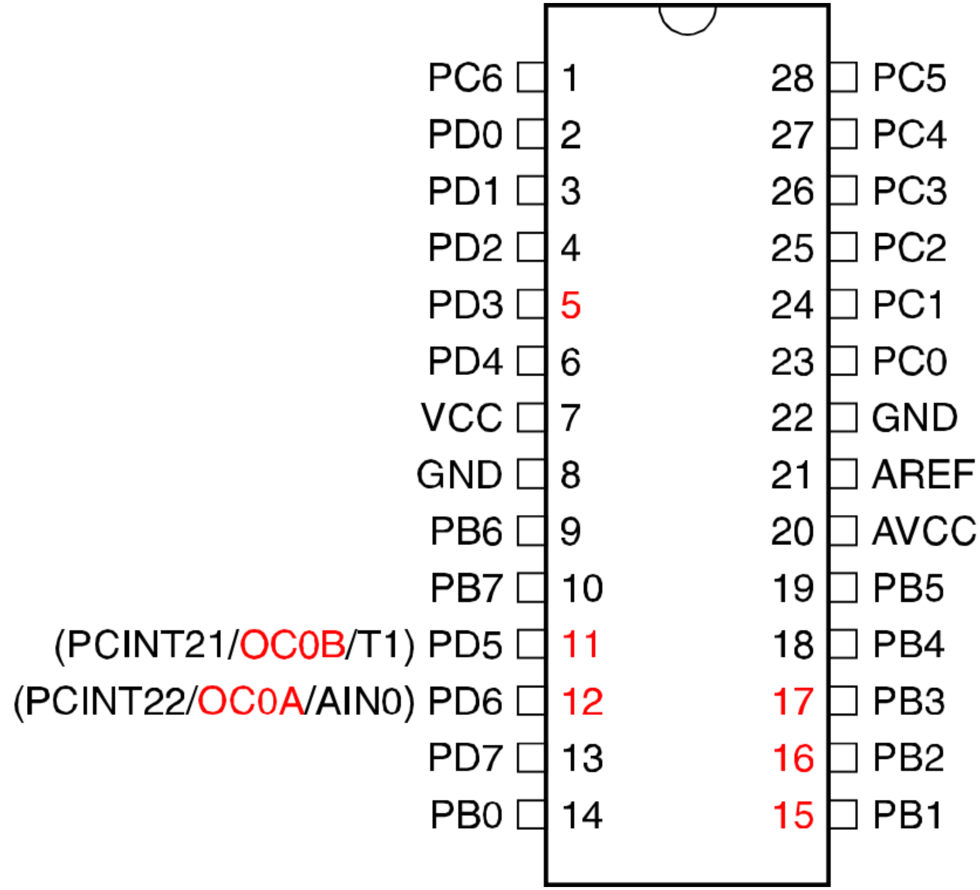
WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase Correct PWM, top 0xFF
0	1	0	CTC (Clear Timer on Compare Match)
0	1	1	Fast PWM, top 0xFF
1	0	0	Reserved
1	0	1	Phase Correct PWM, top OCR0A
1	1	0	Reserved
1	1	1	Fast PWM, top OCR0A

Fast PWM

- Timers are used to generate PWM signals
- OCR0A determines the duty cycle

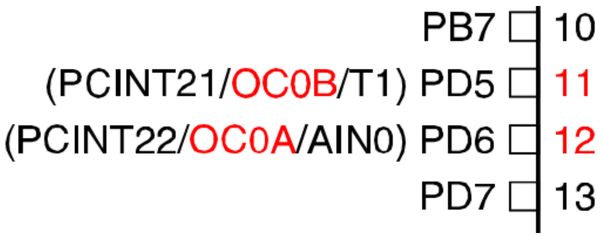


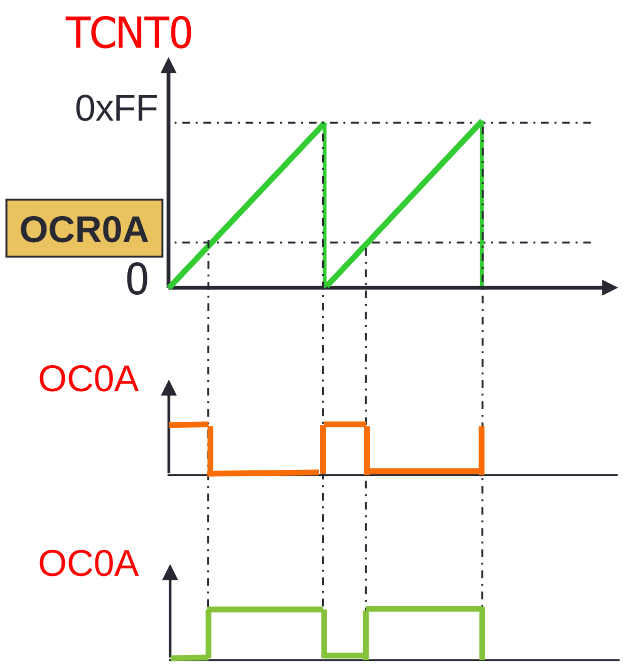
TCNT0: 0 → 0xFF



Fast PWM Duty Cycle (OC0A)

- Non-inverted vs. inverted





Non-inverted mode

$$\frac{Duty}{cycle} = \frac{OCR0A + 1}{256} \times 100$$

Inverted mode

$$\frac{Duty}{cycle} = \frac{255 - OCR0A}{256} \times 100$$

OC0A Invert Control: TCCR0A/B

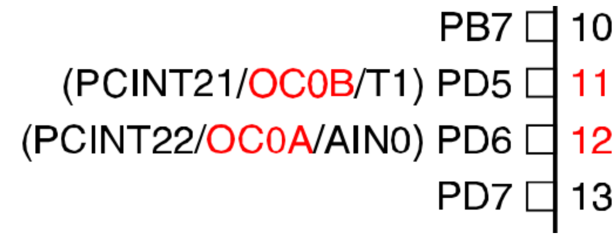
TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
TCCR0B	FOC0A	FOC0B	-	-	WGM02	**CS02**	**CS01**	**CS00**

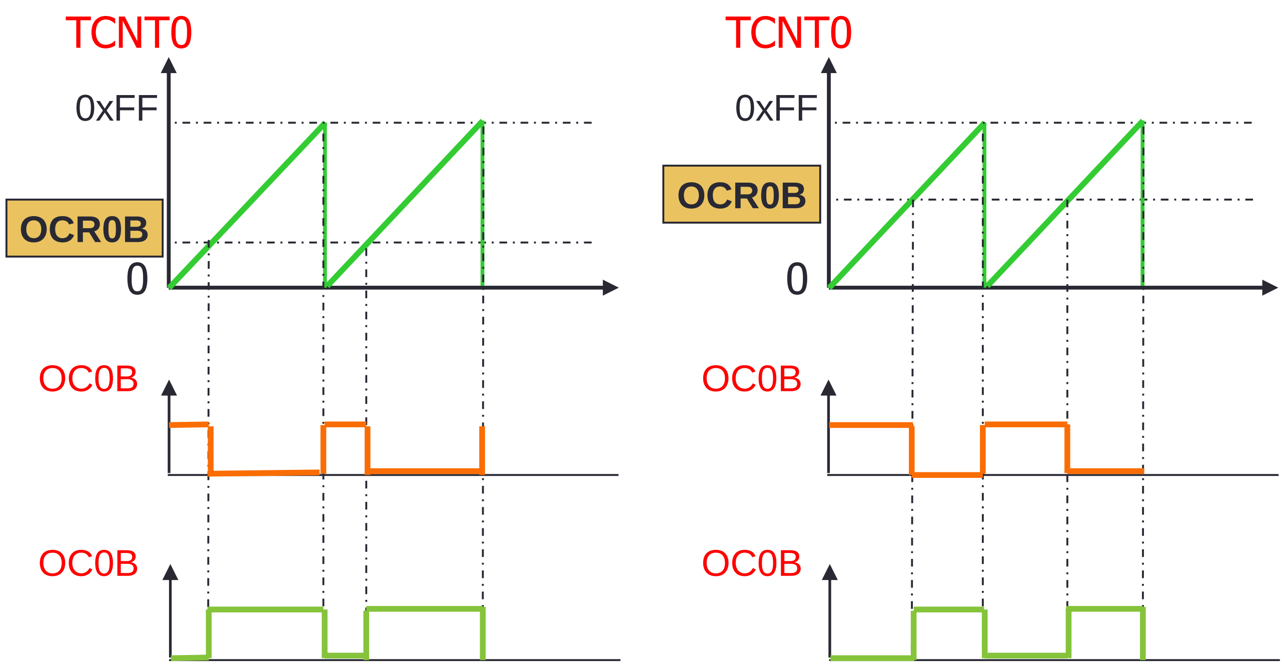
Compare Output Mode

	COM0A1	COM0A0	Comment
Fast PWM	0	0	PWM off
	0	1	WGM02 = 0: PWM off WGM02 = 1: Toggle OC0A on compare match
	1	0	PWM, non-inverted
	1	1	PWM, inverted
Phase Correct PWM	**COM0A1**	**COM0A0**	**Comment**
	0	0	PWM off
	0	1	WGM02 = 0: PWM off WGM02 = 1: Toggle OC0A on compare match
	1	0	PWM, non-inverted
	1	1	PWM, inverted

Fast PWM Duty Cycle (OC0B)

- Non-inverted vs. inverted





Non-inverted mode

$$\text{Duty cycle} = \frac{OCR0B + 1}{256} \times 100$$

Inverted mode

$$\text{Duty cycle} = \frac{255 - OCR0B}{256} \times 100$$

OC0B Invert Control: TCCR0A/B

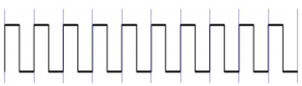
TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
TCCR0B	FOC0A	FOC0B			WGM02	**CS02**	**CS01**	**CS00**

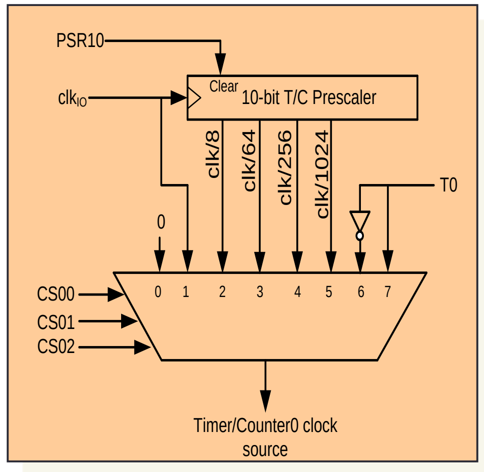
Compare Output Mode

	COM0A1	COM0A0	Comment
Fast PWM	0	0	PWM off
	0	1	WGM02 = 0: PWM off WGM02 = 1: Toggle OC0B on compare match
	1	0	PWM, non-inverted
	1	1	PWM, inverted
Phase Correct PWM	**COM0A1**	**COM0A0**	**Comment**
	0	0	PWM off
	0	1	WGM02 = 0: PWM off WGM02 = 1: Toggle OC0B on compare match
	1	0	PWM, non-inverted
	1	1	PWM, inverted

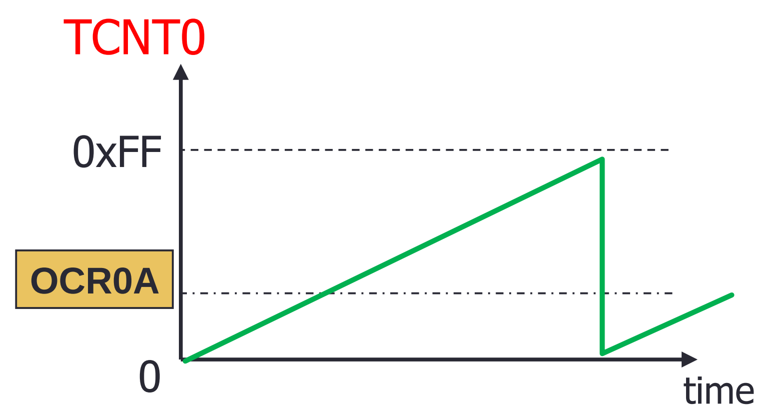
Fast PWM Frequency

$$f_{CLK}$$

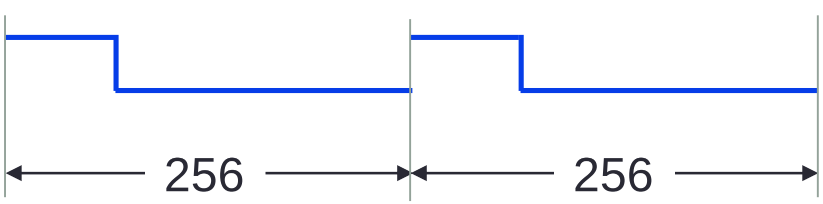




$$f_{CLK} \div p$$



$$f_{PWM} = f_{CLK} \div p \div 256$$



Review: Timer Prescaler Control (TCCR0A/B)

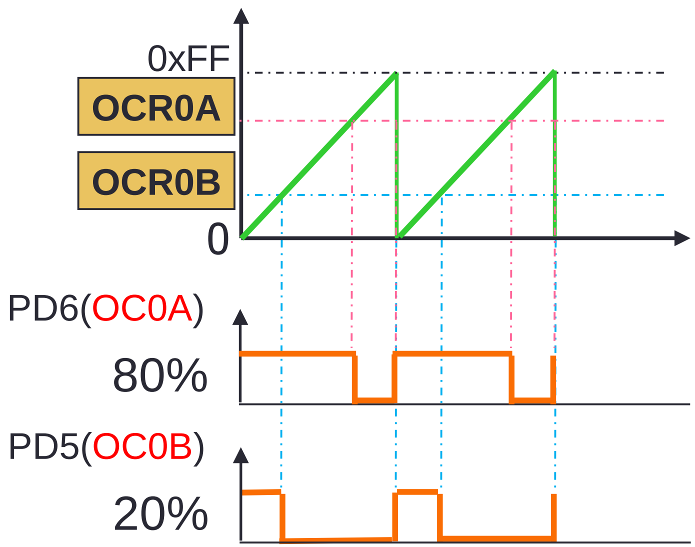
TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
TCCR0B	FOC0A	FOC0B	-	-	WGM02	**CS02**	**CS01**	**CS00**

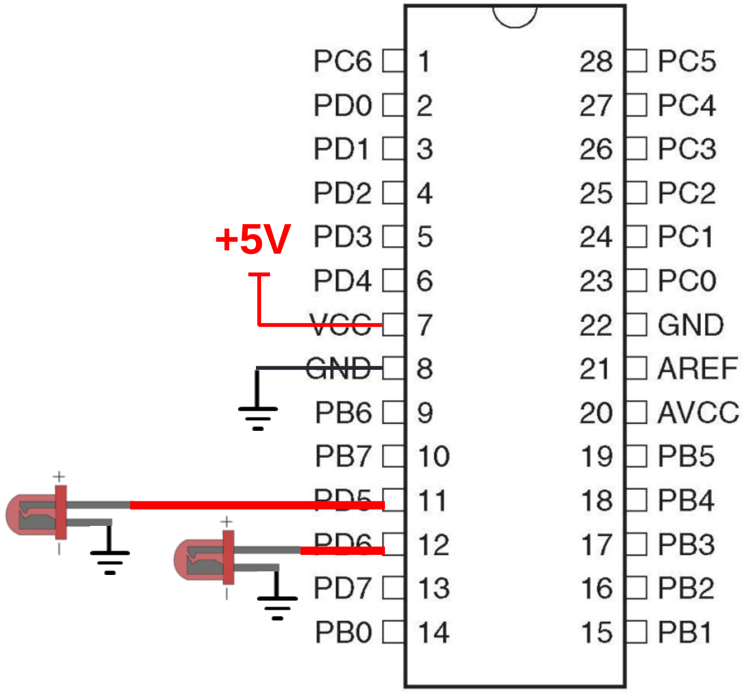
Clock Selector (CS)

CS02	CS01	CS00	Comment
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

Example: Fast PWM

- Suppose the MCU runs at 1Mhz
- Generate PWM waves using Timer0 (non-inverted)
- Output a wave of 80% duty cycle at PD6 (OC0A)
- Output a wave of 20% duty cycle at PD5 (OC0B)
- Prescaler factor $p = 1024$





Flowchart (Fast PWM)

- What value do we set to the controller registers?

$$f_{PWM} = 10^6 \div 1024 \div 256 \approx 3.8 \text{ Hz}$$

$$\frac{OCR0A + 1}{256} \times 100 = 80 \implies OCR0A = 203$$

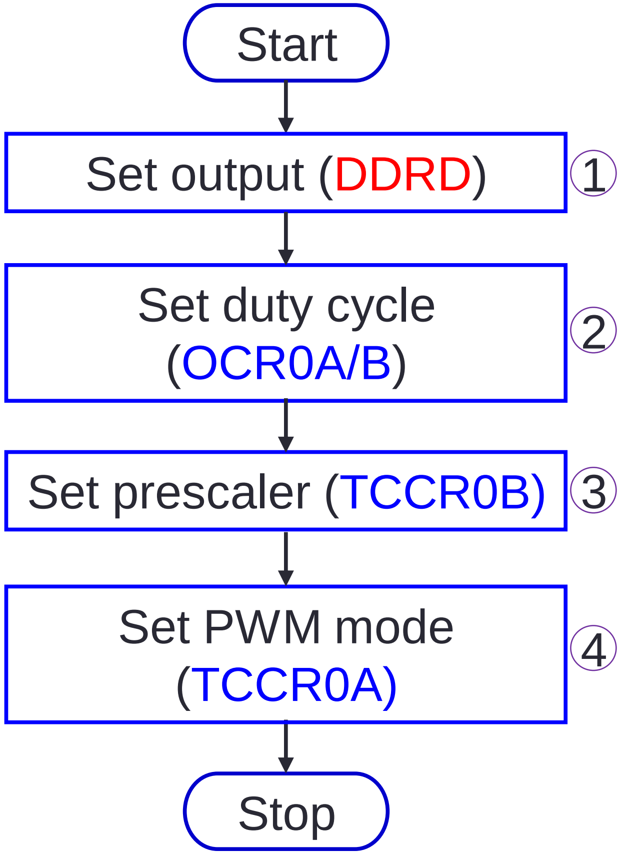
$$\frac{OCR0B + 1}{256} \times 100 = 20 \implies OCR0B = 50$$

TCCR0A

1	0	1	0	0	0	1	1
COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

TCCR0B

FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00
				0	1	0	1



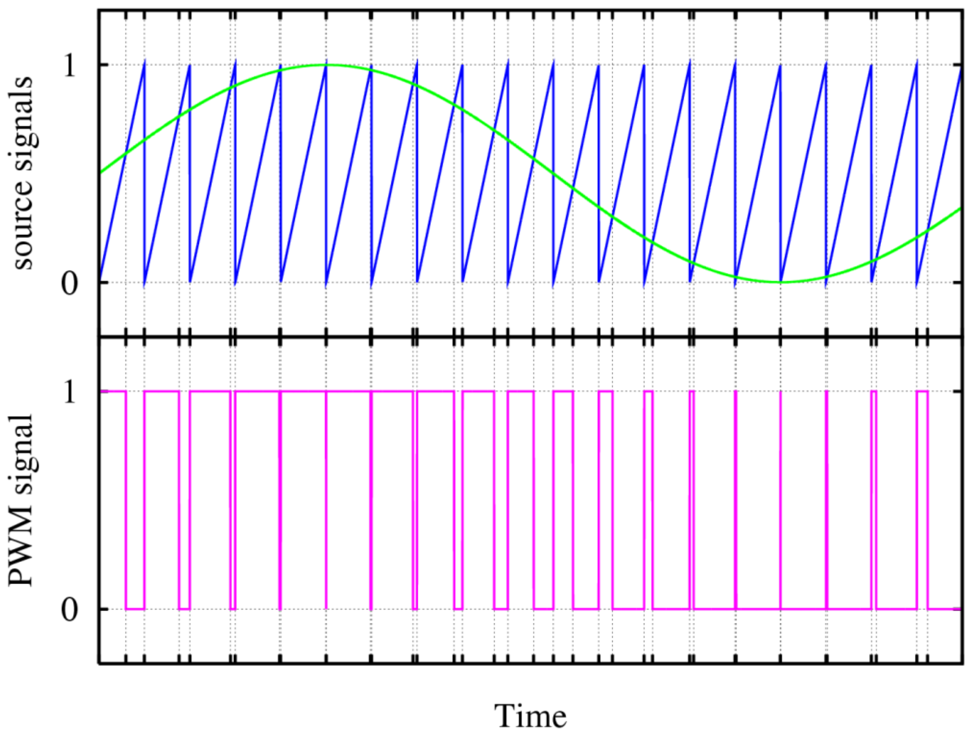
Fast PWM at 3.8 Hz

```
#include <avr/io.h>

int main(void)
{
    CLKPR=(1<<CLKPCE);
    CLKPR=0b00000011;                // set clk to 1Mhz
 1  DDRD=0xFF;                       // PORTD as output
 2  OCR0A=203;                       // 80% duty cycle
    OCR0B=50;                        // 20% duty cycle
 3  TCCR0A=0b10100011;               // fast PWM, non-inverted
 4  TCCR0B=0b00000101;               // timer prescaler
}
```

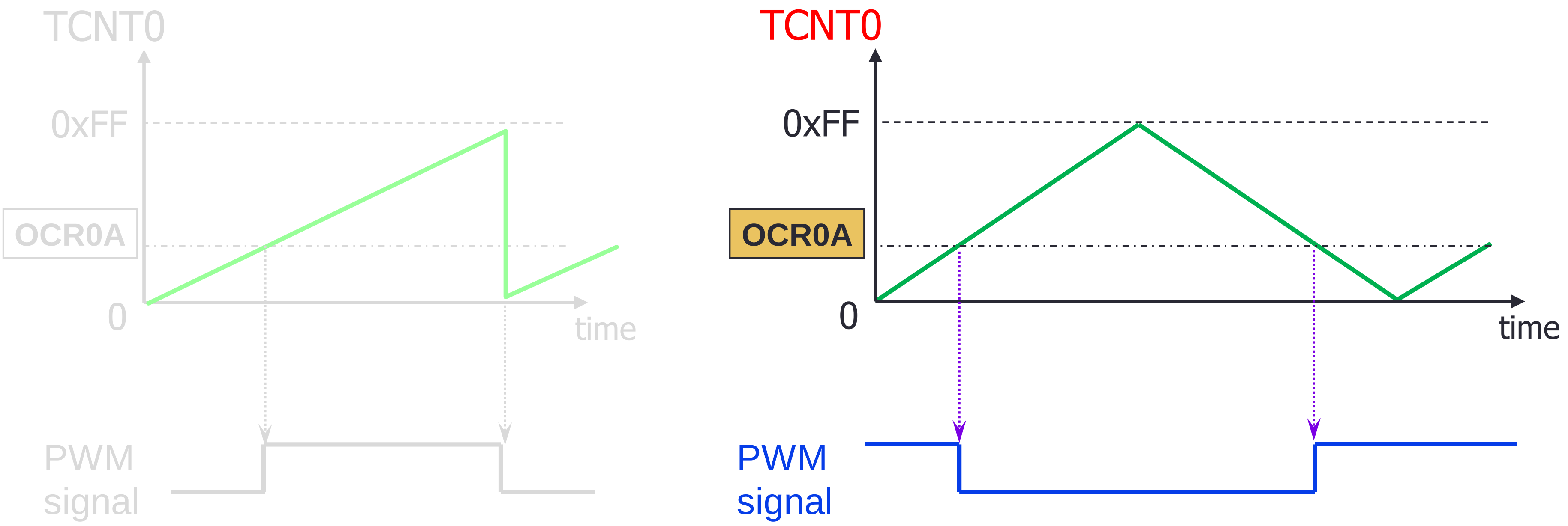
Outline (Cont'd)

- Pulse width modulation (PWM)
  - What is PWM?
  - AVR PWM pinout
  - Fast PWM
  - Phase correct PWM
  - Square wave
  - Duty cycle and frequency
- Getting started



Phase Correct PWM

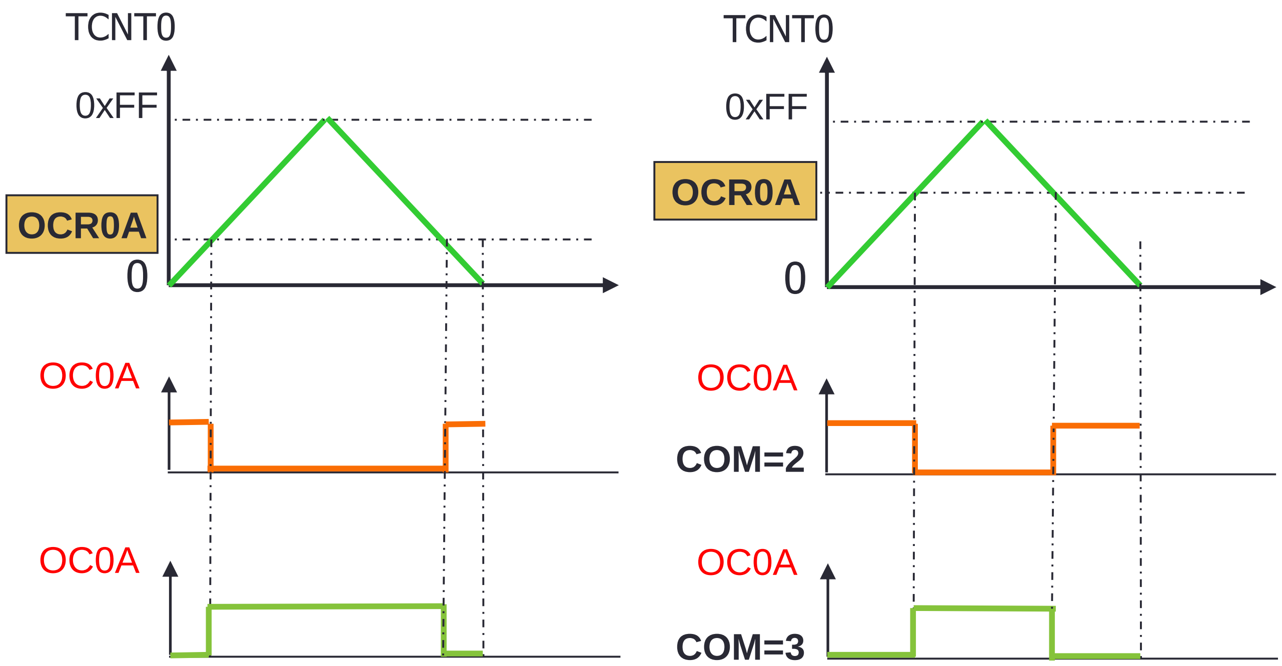
- Timers are used to generate PWM signals



TCNT0: 0 → 0xFF → 0

Phase Correct PWM Frequency and Duty Cycle

- $f_{PWM} = f_{CLK} \div p \div 510$ (p: scalar factor of the timer)
- Non-inverted vs. inverted



Non-inverted mode

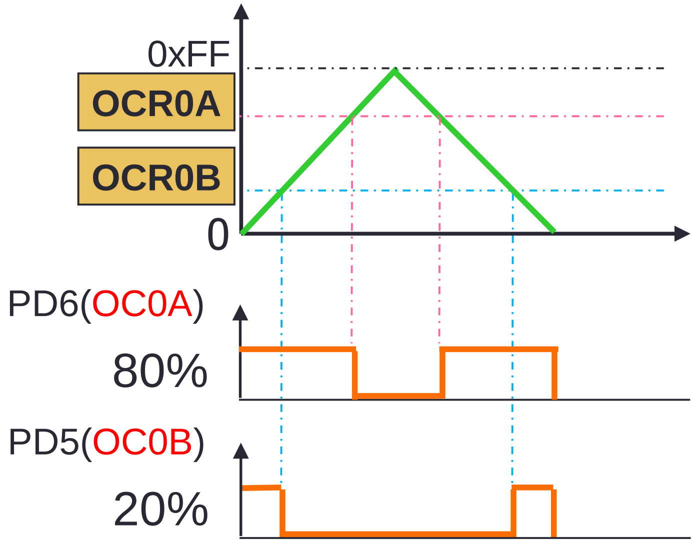
$$\begin{matrix} Duty \\ cycle \end{matrix} = \frac{OCR0A + 1}{255} \times 100$$

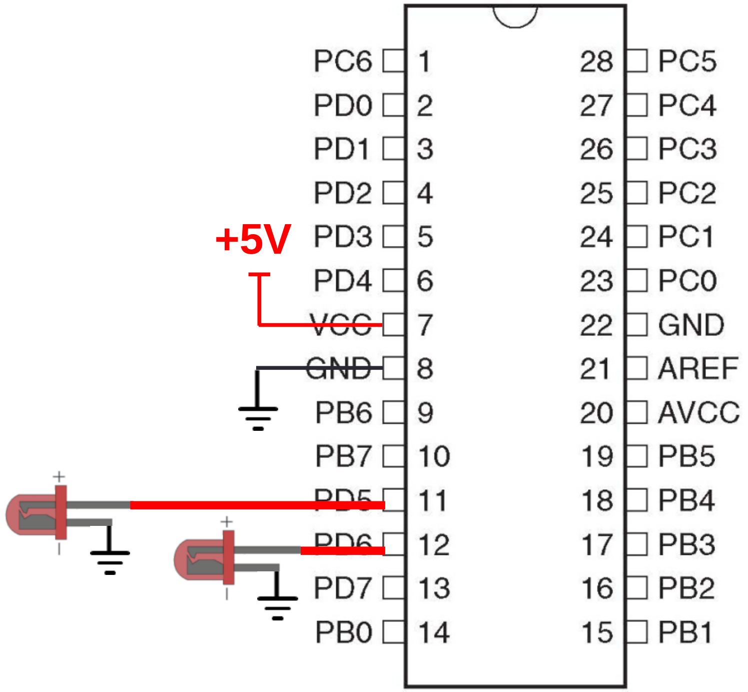
Inverted mode

$$\begin{matrix} Duty \\ cycle \end{matrix} = \frac{255 - OCR0A}{255} \times 100$$

Example: Phase Correct PWM

- Suppose the MCU runs at 1Mhz
- Generate PWM waves using Timer0 (non-inverted)
- Output a wave of 80% duty cycle at PD6 (OC0A)
- Output a wave of 20% duty cycle at PD5 (OC0B)
- Prescaler factor $p = 1024$





Flowchart (Phase Correct PWM)

- What value do we set the controller registers?

$$f_{PWM} = 10^6 \div 1024 \div 510 \approx 1.9 \text{ Hz}$$

$$\frac{OCR0A + 1}{255} \times 100 = 80 \implies OCR0A = 203$$

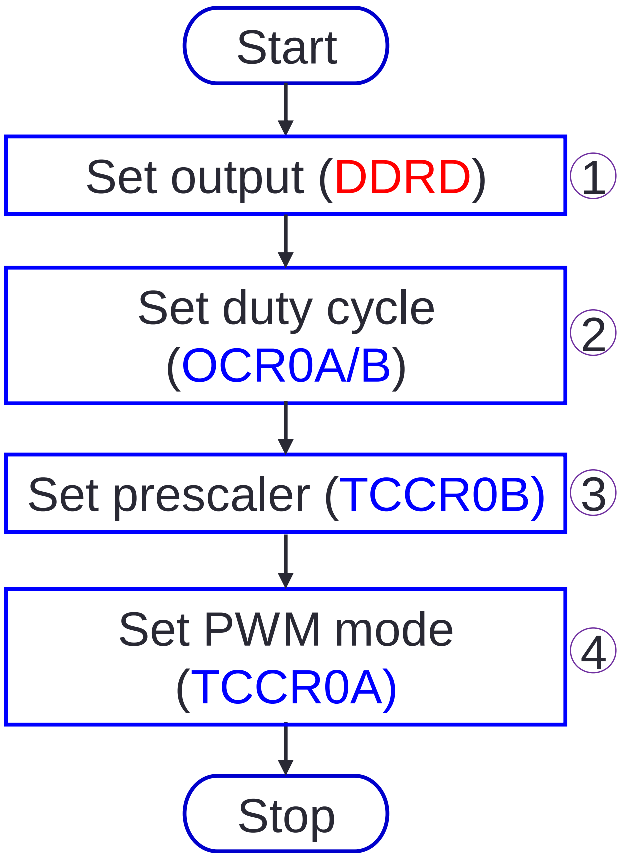
$$\frac{OCR0B + 1}{255} \times 100 = 20 \implies OCR0B = 50$$

TCCR0A

1	0	1	0	0	0	0	1
COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00
				0	1	0	1

TCCR0B



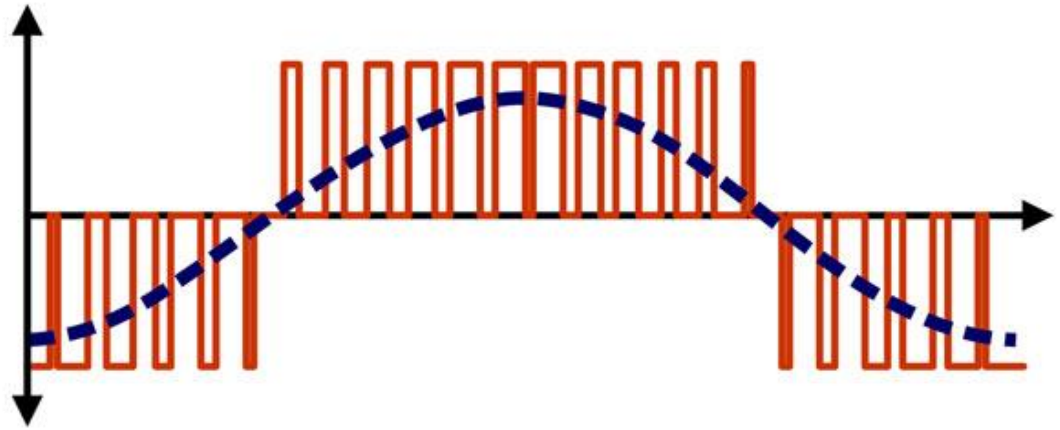
Phase Correct PWM at 1.9Hz

```
#include <avr/io.h>

int main(void)
{
    CLKPR=(1<<CLKPCE);
    CLKPR=0b00000011;                  // set clk to 1Mhz
 ① DDRD=0xFF;                          // PORTD as output
 ② OCR0A=203;                          // 80% duty cycle
    OCR0B=50;                          // 20% duty cycle
 ③ TCCR0A=0b10100001;                  // phase correct PWM
 ④ TCCR0B=0b00000101;                  // timer prescaler
}
```

Outline (Cont'd)

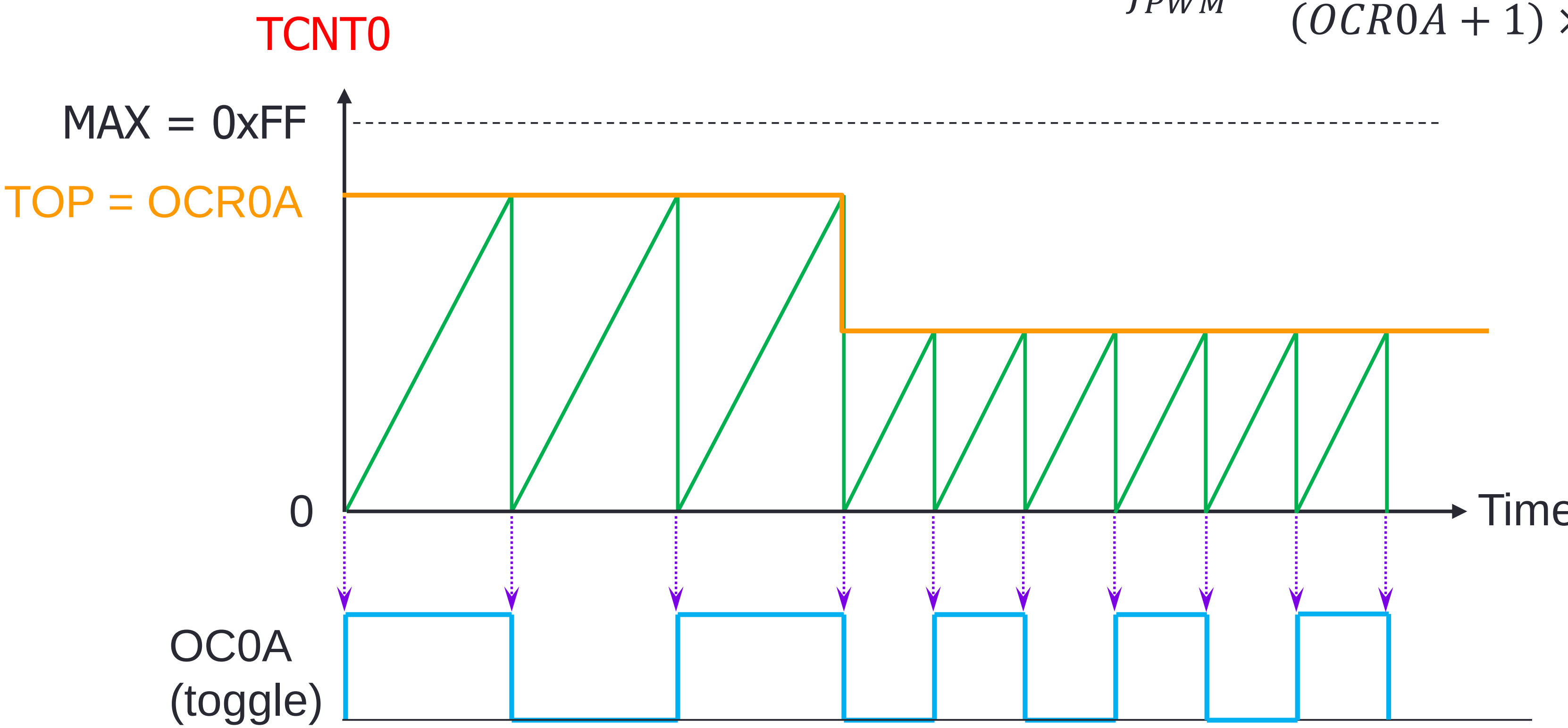
- Pulse width modulation (PWM)
  - What is PWM?
  - AVR PWM pinout
  - Fast PWM
  - Phase correct PWM
  - Square wave
  - Duty cycle and frequency
- Getting started



Square Wave with Frequency Control

- Generate square waves with various frequencies

$$f_{PWM} = \frac{f_{CLK}}{(OCR0A + 1) \times p}$$



Note: duty cycle cannot be controlled in this mode!

Square Wave Frequency Control: Mode Setup

TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
TCCR0B	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00

Timer Mode (WGM)

WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	Phase Correct PWM, top 0xFF
0	1	0	CTC (Clear Timer on Compare Match)
0	1	1	Fast PWM, top 0xFF
1	0	0	Reserved
1	0	1	Phase Correct PWM, top OCR0A
1	1	0	Reserved
1	1	1	Fast PWM, top OCR0A

Square Wave Frequency Control: Toggle Setup

TCCR0A							
COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

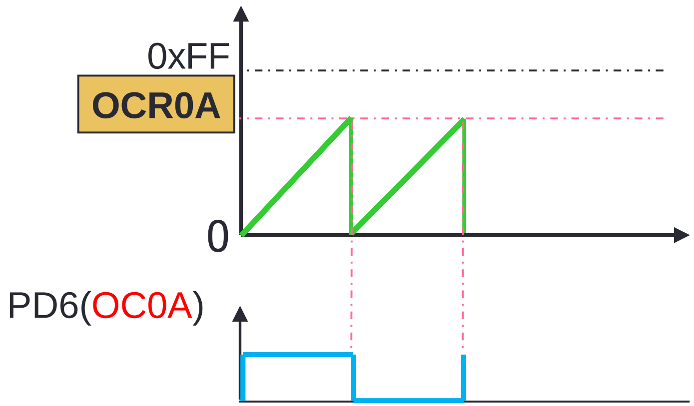
Compare Output Mode

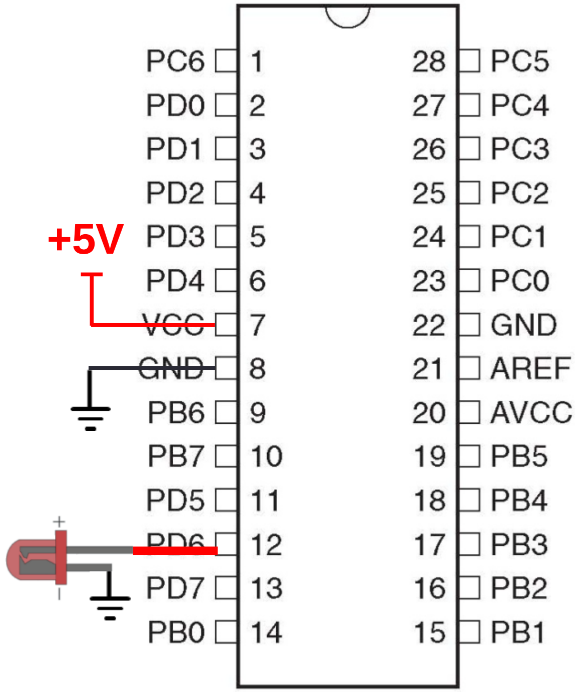
	COM0A1	COM0A0	Comment
Fast PWM	0	0	PWM off
	0	1	WGM02 = 0: PWM off WGM02 = 1: Toggle OC0A on compare match
	1	0	PWM, non-inverted
	1	1	PWM, inverted

	COM0A1	COM0A0	Comment
Phase Correct PWM	0	0	PWM off
	0	1	WGM02 = 0: PWM off WGM02 = 1: Toggle OC0A on compare match
	1	0	PWM, non-inverted
	1	1	PWM, inverted

Example: Varying LED Flash Frequency

- Suppose the MCU runs at 1Mhz
- Generate square waves continuously using PWM
- Vary the OCR0A from 30 to 250 with an increment of 55
- Output the wave at PD6 (OC0A)
- Prescaler factor $p = 1024$





Flowchart (LED Flash)

$$f_{PWM} = \frac{10^6}{(30+1)\times 1024} \approx 31.5 \text{ Hz}$$

$$f_{PWM} = \frac{10^6}{(250+1)\times 1024} \approx 3.9 \text{ Hz}$$

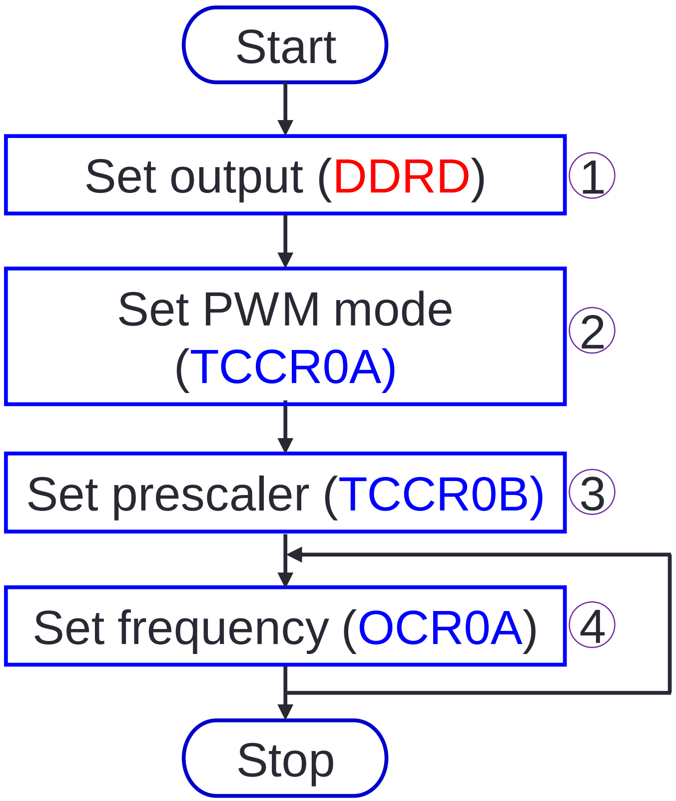
- What value do we set the controller registers?

TCCR0A

0	1	0	0	0	0	1	1
COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00
				1	1	0	1

TCCR0B



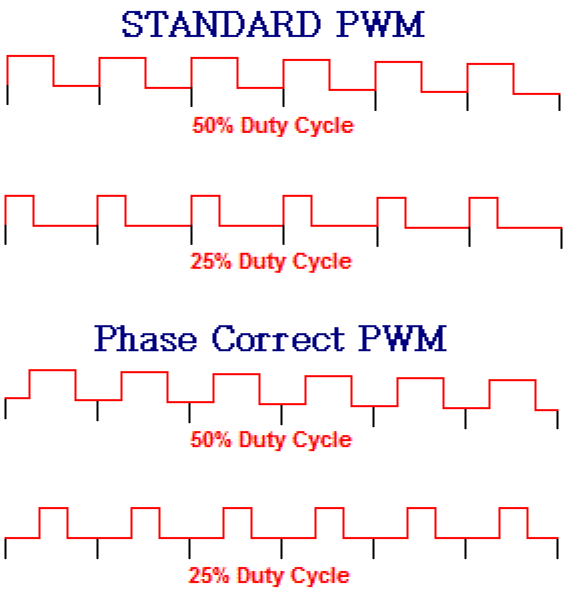
Varying LED Flash Frequency

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    CLKPR=(1<<CLKPCE);
    CLKPR=0b00000011;                // set clk to 1Mhz
 1  DDRD=0xFF;                       // PORTD as output
 2  TCCR0A=0b01000011;               // TOP OCR0A, toggle at TOP
 3  TCCR0B=0b00001101;               // timer prescaler
    while (1){
        for (int i=30; i<=250; i=i+55){
 4          OCR0A=i;
            _delay_ms(2000);
        }
    }
}
```

Outline (Cont'd)

- Pulse width modulation (PWM)
  - What is PWM?
  - AVR PWM pinout
  - Fast PWM
  - Phase correct PWM
  - Square wave
  - Duty cycle and frequency control
- Getting started



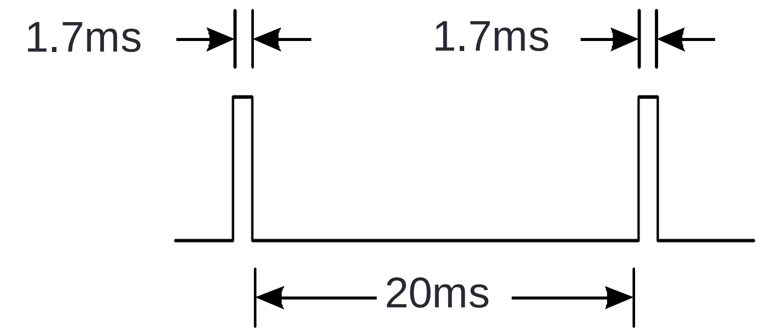
PWM Modes of Timer/Counter1

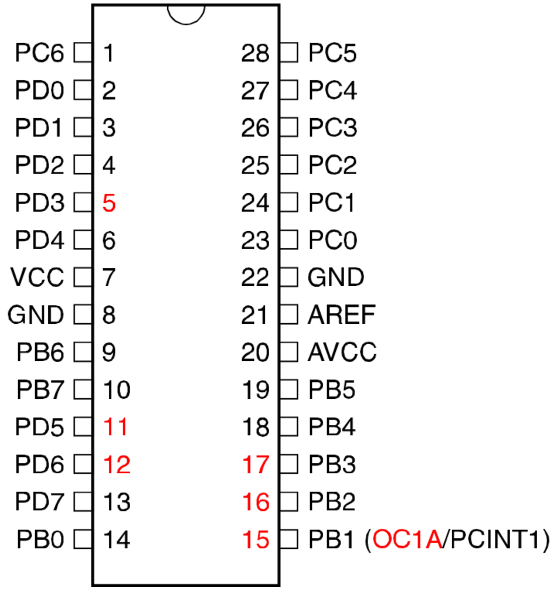
Register								
TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10
TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	**CS12**	**CS11**	**CS10**
TCCR1C	FOC1A	FOC1B	-			-		

Mode	WGM 13	WGM 12	WGM 11	WGM 10	Timer/counter mode	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

Example: 46Hz with 8% Duty Cycle

- Suppose the MCU runs at 1MHz
- Generate a wave as the figure (46Hz and 8% duty cycle)
- Use PWM of Timer1
- Prescaler factor $p = 1$





Example: 46Hz with 8% Duty Cycle (Cont'd)

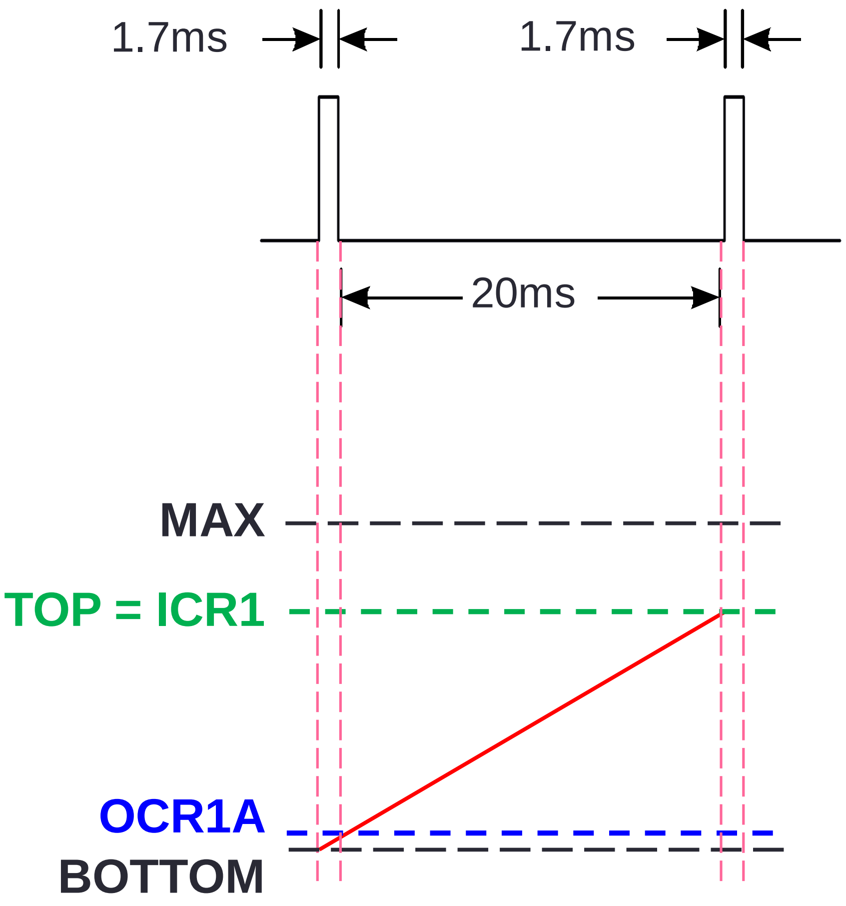
- Strategy:
  1. Mode 14 fast PWM
  2. Set frequency using **ICR1**

     $$f = \frac{10^6}{\mathbf{ICR1}} = 46$$

     $\Rightarrow$ **ICR1** = 21701
  3. Set duty cycle using **OCR1A**

     $$DC = \frac{1.7}{20+1.7} = \frac{\mathbf{OCR1A}}{\mathbf{ICR1}}$$

     $\Rightarrow$ **OCR1A** = 1700



OC1A Toggle Mode: TCCR0A

TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10
TCCR1B	IC C1	IC S1	-	WGM13	WGM12	**CS12**	**CS11**	**CS10**
TCCR1C	FO 1A	FO 1B	-	-	-			

Compare Output Mode

	COM0A1	COM0A0	Comment
Fast PWM	1	0	Clear OC1A on compare match, Set OC1A at BOTTOM
	1	1	Set OC1A on compare match, Clear OC1A at BOTTOM

	COM0A1	COM0A0	Comment
Phase Correct PWM	1	0	Clear OC1A on compare match, Set OC1A at BOTTOM
	1	1	Set OC1A on compare match, Clear OC1A at BOTTOM

Note: the setup for OC1B is the same

46Hz with 8% Duty Cycle

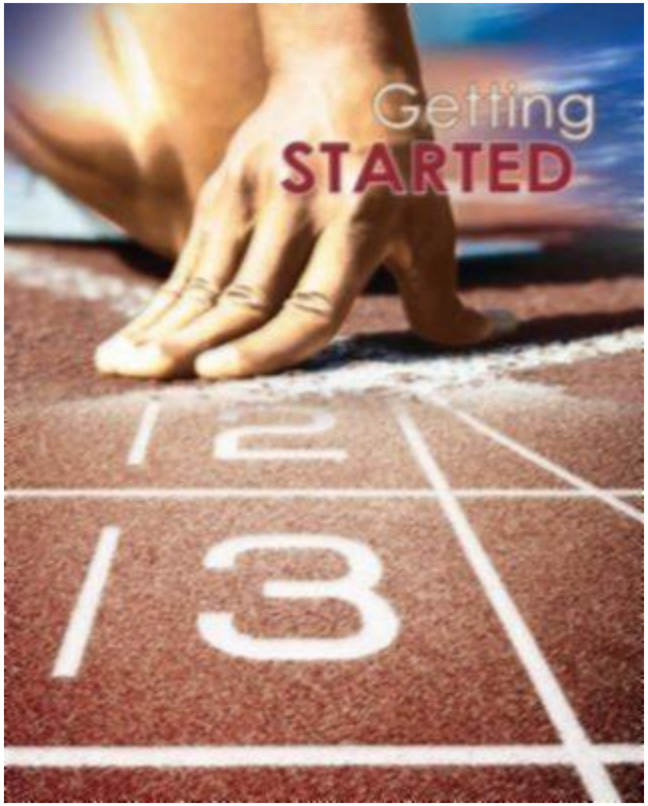
	1	0	0	0	0	0	1	0
TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10
	0	0	0	1	1	1	0	1
TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

```
#include <avr/io.h>

int main(void)
{
    CLKPR=(1<<CLKPCE);
    CLKPR=0b00000011;                    // set clk to 1Mhz
    DDRB=0xFF;                           // PORTB as output
    ICR1=21701;
    OCR1A=1700;
    TCCR1A=0b10000010;
    TCCR1B=0b00011001;
}
```

Outline (Cont'd)

- Pulse width modulation (PWM)
  - What is PWM?
  - AVR PWM pinout
  - PWM modes
- Getting started



Reference

- ATmega328P data sheet
- AVR 8-bit instruction set
- AVR131: Using the AVR's High-speed PWM
- M. A. Mazidi, S. Naimi, and S. Naimi, *The AVR Microcontroller and Embedded Systems: Using Assembly and C*, Prentice Hall, 2010