EE 109 Unit E – Pulse Width Modulation

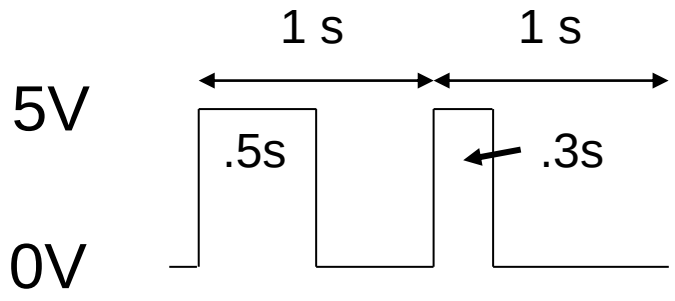
Power

- Recall (or learn) that Power is a measure of:
  - Energy per unit time
- In an electronic circuit, P = I * V
  - Power = Current & Voltage (each may be varying w/ time)
- A circuit that draws a constant 2 mA of current at a constant 5V would consume 10 mW
- Since voltage and current may change rapidly, it is often helpful to calculate the average power

$$P = \frac{1}{T}\int_0^T P(t)dt$$

- Just sum the total power and divide by the total time

I = 1A



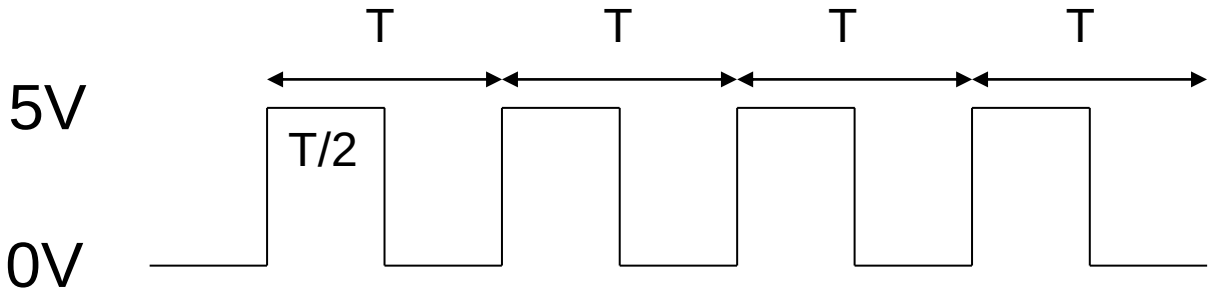
Average Power
= (1*5*.8)/2 = 2W

Duty Cycle

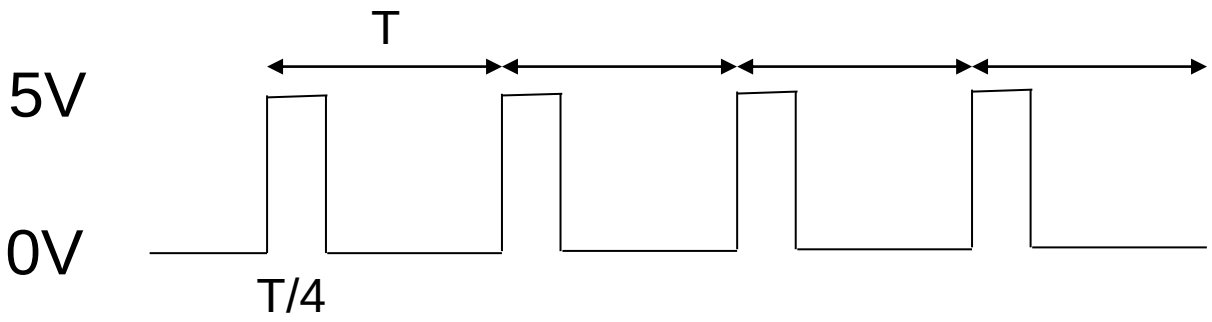
- A pulse is just a short window of time when a signal is 'on'
- We could repeat the pulse at some regular period, T
- We define the duty cycle as

$$\text{Duty Cycle \%} = (\text{ON Time} / T) * 100$$

Duty Cycle = 50%



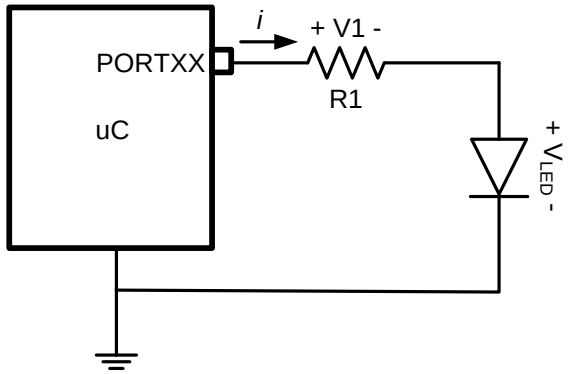
Duty Cycle = 25%

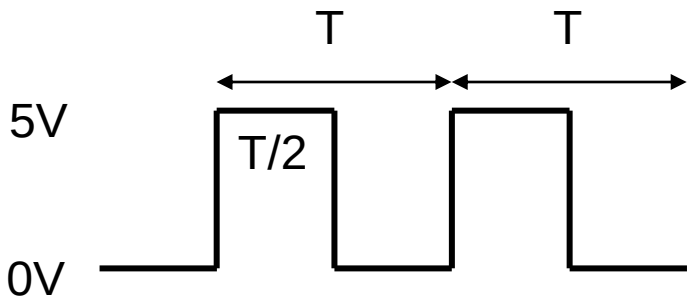


Power & Duty Cycle

- When we light up an LED we often just turn a PORTxx output 'on' and leave it 'on'
  - This supplies the maximum power possible to the LED
- We could pulse the output at some duty cycle (say 50%) at a fast rate
  - Fast so that the human eye can't detect it flashing
  - Average power would be ½ the original always 'on' power
  - Result would be a 'dimmer' LED glow

PORTxx 'on' constantly

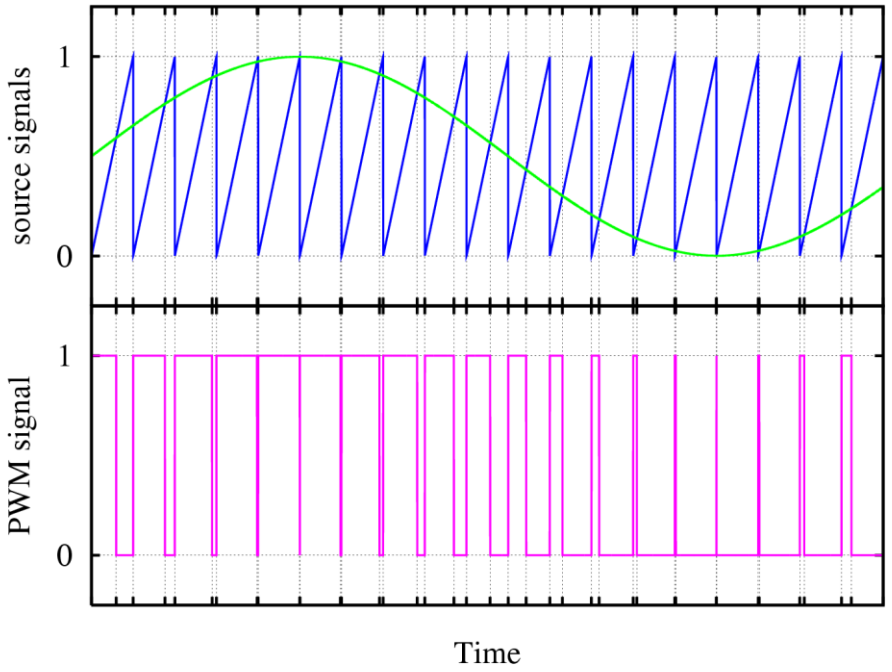




PORTxx 'on' 50% of time

PWM

- Modulation refers to changing a value based on some signal (i.e. changing one signal based on another)
- Pulse width modulation refers to modifying the width of a pulse based on another signal
- It can be used to transform one signal into another
  - Example below of sine wave represented as pulses w/ different widths
- Or it can just be used to alter average power as in the last activity

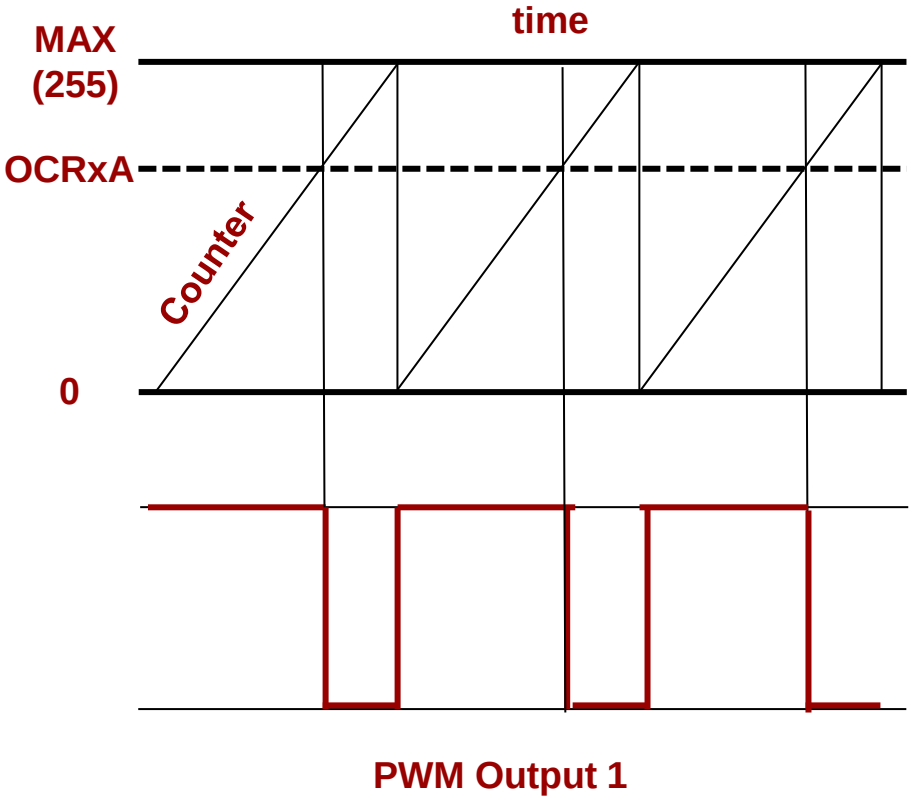


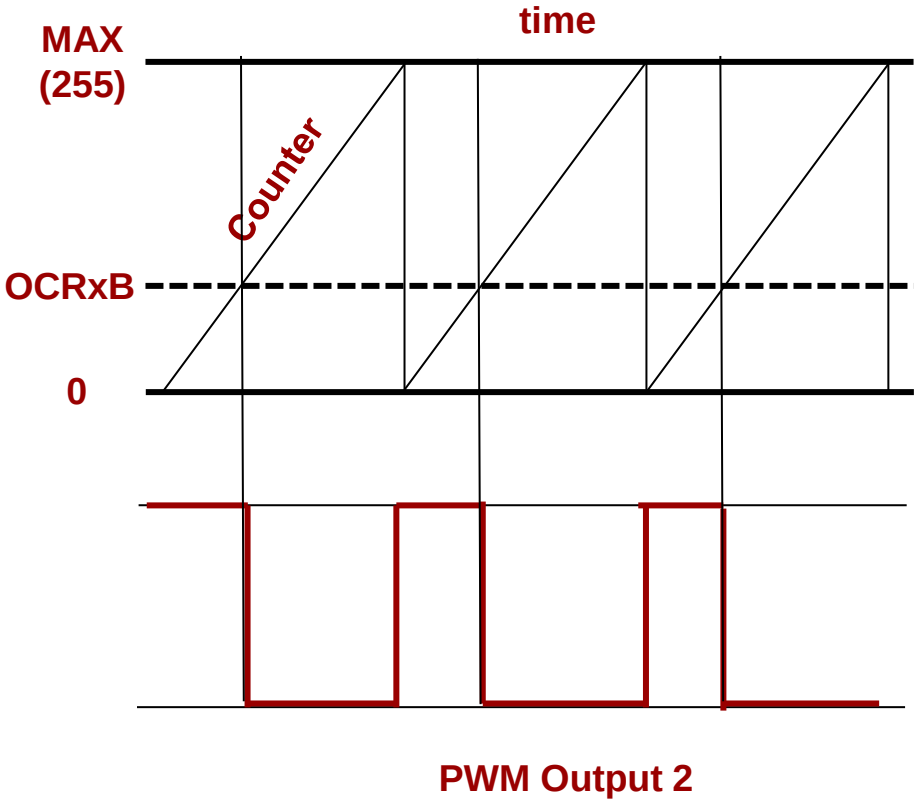
Implementing PWM

- Can use delays or timers to make your own pulse signals
- Most microcontrollers have hardware to automatically generate PWM signals based on the contents of some control registers
- Many microcontrollers use the Timers to also serve as PWM signals
  - Recall the timer module gave us a counter that would increment until it hit some 'modulus' (MAX) count which would cause it to restart and also generate an interrupt

Using Timers for PWM

- For PWM we can use that counter to just count 0 to some MAX count making the:
  - PWM output = '1' while the count < threshold (OCRxx) and
  - PWM output = '0' when the count >= threshold (OCRxx)





PWM Control Registers

- The Arduino has 3 timers that can be used for PWM:
  - **Two 8-bit times: Timer/Counter0 and Timer/Counter2**
  - **One 16-bit time: Timer/Counter1**
- Refer to the timer register definitions on the next slides
  - Set WGM0[2:0] bits for Fast PWM mode as opposed to CTC
- The 3 timers can control the following output pin behavior using the control register bits and thresholds in the given registers

Timer	Output Pin	Threshold	Control Reg Bits
0	PD6	OCR0A (8-bit)	COM0A[1:0]
0	PD5	OCR0B (8-bit)	COM0B[1:0]
1	PB1	OCR1A (16-bit)	COM1A[1:0]
1	PB2	OCR1B (16-bit)	COM1B[1:0]
2	PB3	OCR2A (8-bit)	COM2A[1:0]
2	PD3	OCR2B (8-bit)	COM2B[1:0]

See datasheet, textbook or other documentation for further explanation

8-bit PWM Control Registers

- Set WGM bits for PWM mode [usually Fast PWM mode] as opposed to CTC
- Pick COMxy[1:0] for desired waveform
  - X = 0 or 2 (for timer 0 or 2)
  - Y = A or B
- Still need to pick a prescaler to slow down the clock
- Set OCRxA or OCRxB to the desired threshold which will effectively control the duty cycle of the PWM output

COM0 A1	COM0 A0	COM0 B1	COM0 B0	-	-	WGM 01	WGM 00

TCCR0A Reg. (TCCR2A)
Timer/Counter0 Control Register

FOC 0A	FOC 0B	-	-	WGM 02	CS02	CS01	CS00

TCCR0B Reg. (TCCR2B)
Timer/Counter0 Control Register

CSx [2:0]	Prescaler
010	Clk / 8
011	Clk / 64
100	Clk / 256
101	Clk / 1024

Timer 0 Prescalars (Note: Timer 2 has different prescalars; refer to datasheet section 18.11.2)

COMxy[1], COMxy[0]	Output Compare pin (assume WGMx2=0)
00	Don't use Pin
01	Don't use Pin
10	Set Pin on CTR=0x00, Clear pin on match=OCR?
11	Clear Pin on CTR=0x00, Set pin on match=OCR?

WGMx1, WGMx0	WGMx2=0	WGMx2=1
00	Normal (Counter)	Unused
01	Phase Correct PWM	Phase Correct PWM (Top=OCR0A)
10	CTC (Timer)	Unused
11	**Fast PWM (Top=255, Thresh=OCRx)**	Fast PWM (Top=OCR0A, Thresh = OCRB)

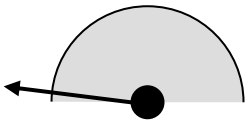
Servo Motors

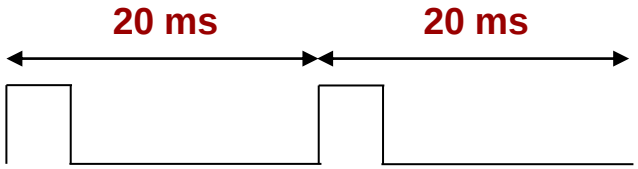
- Many embedded systems use servo motors to move or rotate mechanical devices
- Most servo motors use some form of pulse width modulation to control the direction and speed of their rotation
- 2 Kinds
  - Standard servo motors: can only rotate through a certain arc (usually 180 degrees)
  - Continuous: can keep spinning round and round while pulses are provided

Standard Servo Motor

- Pulse width determines angle (position) of servo motor
- Must continue to give pulses for the duration of time it takes to rotate to the desired position
- No pulses = stay put

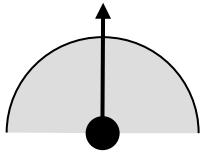
Full left

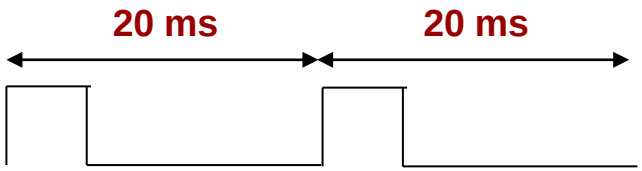




Pulse width = 750us

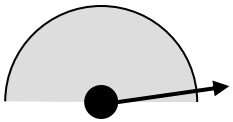
Centered

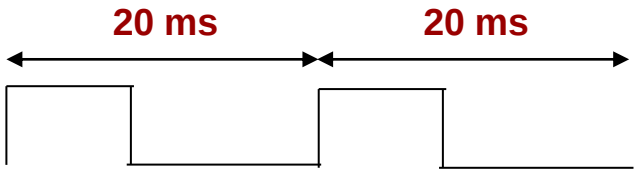




Pulse width = 1500us

Full right



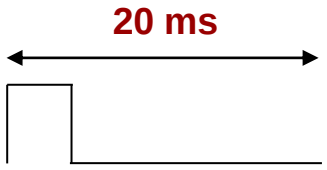


Pulse width = 2250us

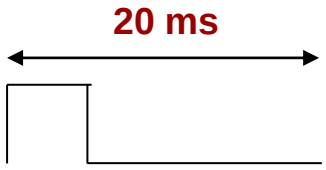
Do an Internet search for Standard Servo Motors & try to find the appropriate pulse width for each position

Continuous Servo Motors

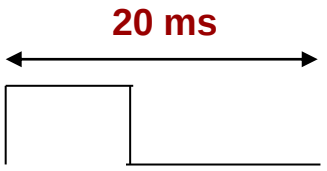
- Pulse width determines speed & direction of rotation
- Controlled via PWM (Pulse Width Modulation)
  - Short pulse = Rotate one direction
  - Medium pulse = Stop
  - Long pulse = Rotate other direction



Pulse Width = 750 us = Full Speed Clockwise



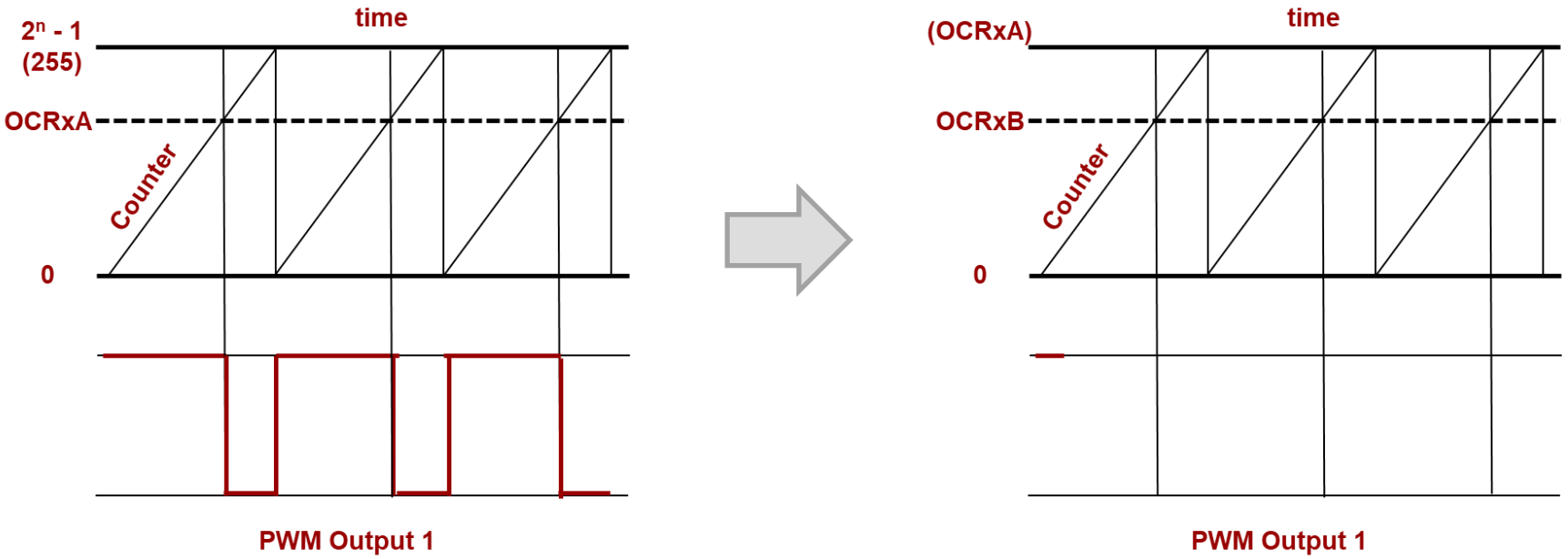
Pulse Width = 1500 us = Stopped



Pulse Width = 2250 us = Full Speed Counter-Clockwise

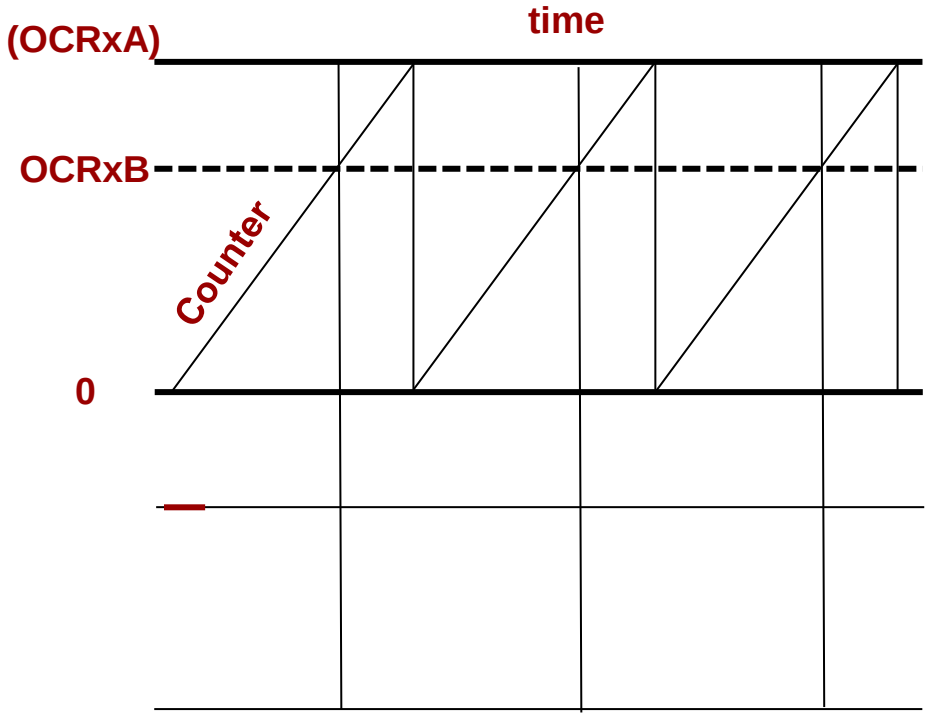
A Different Maximum

- Currently, the counter would count to the full scale maximum value $(2^n - 1)$ which would yield a particular PERIOD (not duty cycle) based on the prescaler.
  - The period didn't matter, as we mainly cared about duty cycle
- But what if we need a specific period that is not based on $(2^n - 1)$, an alternate mode allows OCRxA to determine the period and OCRxB determines the duty cycle

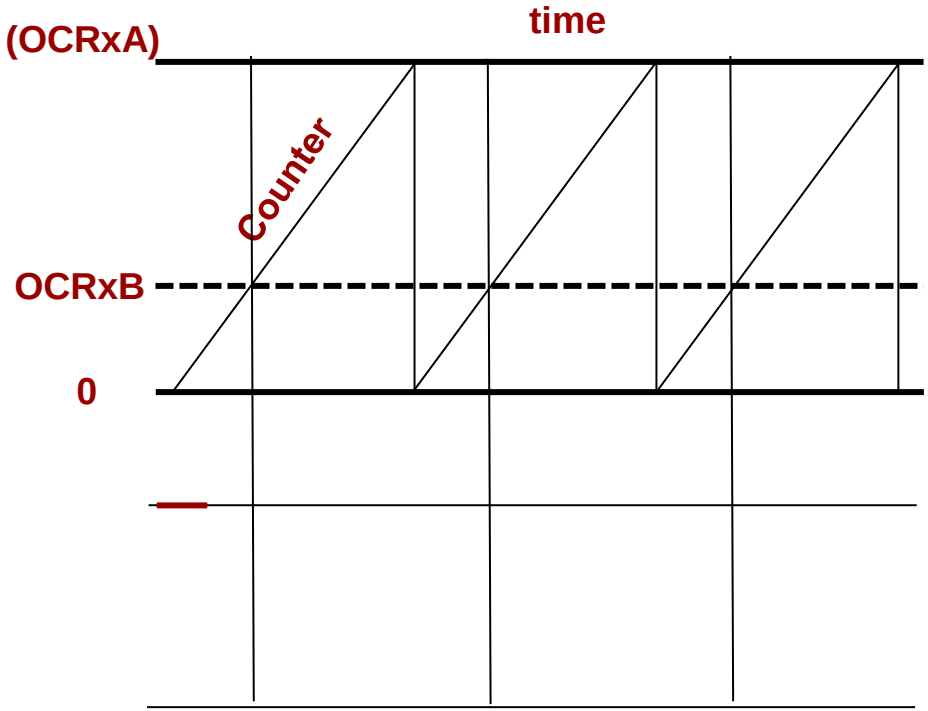


Using Timers for PWM

- In the examples we've show, the MAX is 2^n-1 (where n=8 or 16) based on which timer you use
- In a different mode (WGMx2 = 1) we can have a user-specified MAX by placing any number in OCRA



PWM Output 1



PWM Output 2

16-bit PWM Control Registers

- Set WGM bits for PWM mode [usually Fast PWM mode] as opposed to CTC
- Pick COM0?[1:0] for desired waveform
- Still need to pick a prescaler to slow down the clock
- Set OCRA or OCRB to the desired threshold which will effectively control the duty cycle of the PWM output

COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10

TCCR1A Reg.
Timer/Counter0 Control Register

FOC1A	FOC1B	-	WGM13	WGM12	CS12	CS11	CS10

TCCR1B Reg.
Timer/Counter0 Control Register

CS1[2:0]	Prescaler
010	Clk / 8
011	Clk / 64
100	Clk / 256
101	Clk / 1024

COM1?1, COM1?0	Output Compare pin (assume WGM02=0)
00	Don't use Pin
01	Don't use Pin
10	Set Pin on CTR=0x00, Clear pin on match=OCR?
11	Clear Pin on CTR=0x00, Set pin on match=OCR?

WGM11, WGM10	WGM12=0	WGM12=1
00	Normal (Counter)	Unused
01	Phase Correct PWM	Phase Correct PWM (Top=OCRA)
10	CTC (Timer)	Unused
11	**Fast PWM (Top=2^{16}-1, Thresh=OCRx)**	Fast PWM (Top=OCR1A, Thresh=OCR1B)