Lecture 16: Actuators 2

ME/AE 6705

Introduction to Mechatronics

Dr. Jonathan Rogers

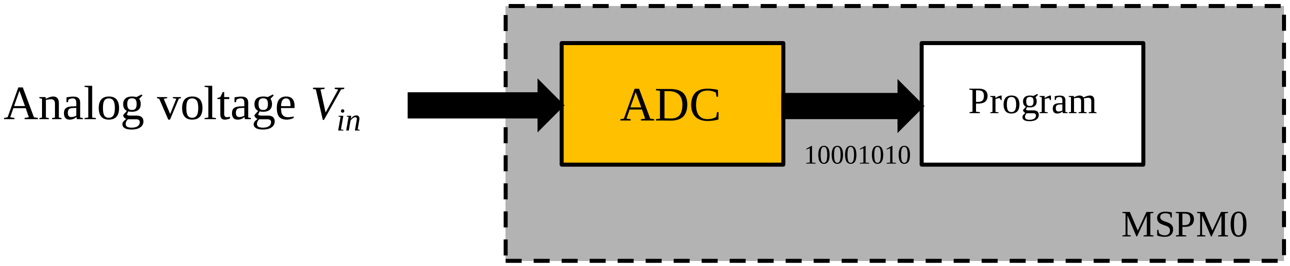
Lesson Objectives

- Be able to generate an analog signal from a digital output using pulse width modulation (PWM)
- Understand and be able to calculate duty cycle
- Be able to implement variable speed controller for DC motor using PWM signal on MSPM0
- Be able to interface with servos using PWM signal on MSP
- Be able to construct motor interface using H-bridge circuit

Georgia Tech

Digital to Analog Conversion

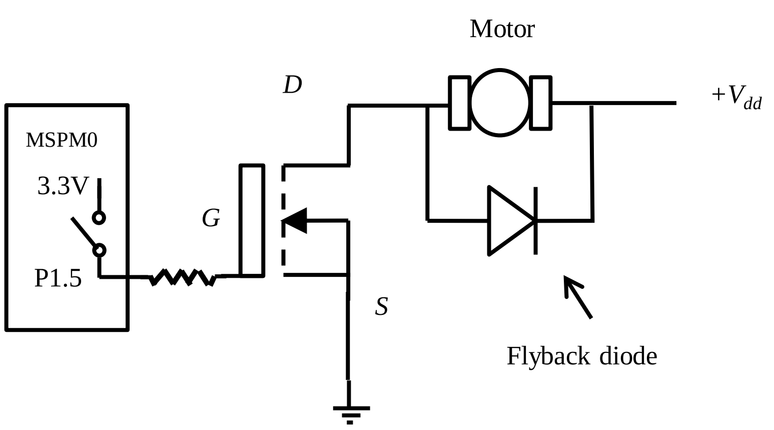
- During ADC lecture we learned how to turn an analog signal into a digital signal for use on MCU



  - This is very useful when you want to measure sensor data or other analog signals and use in your program
  - ADC used a lot when interfacing with sensors!

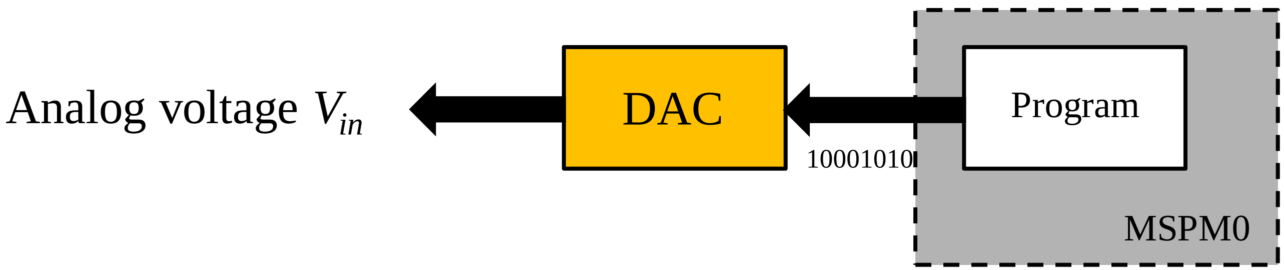
Digital to Analog Conversion

- What happens if we want to go the other way?
- Suppose we want to take a digital signal on MCU, and send it out as analog voltage?
  - For instance, what happens if we want to change speed of motor?
  - Currently, we can only know how to send out 0V or 3.3V.
  - This means motor will either turn at one speed, or not at all

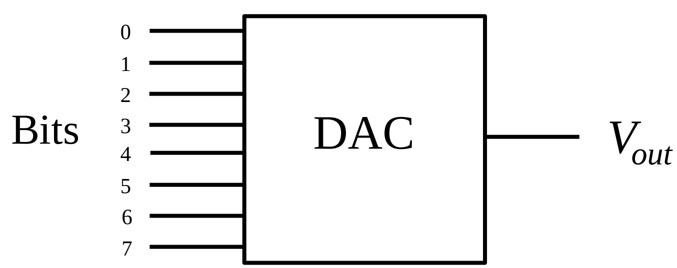


Digital to Analog Conversion

- A digital to analog converter (DAC) converts digital signals to analog ones



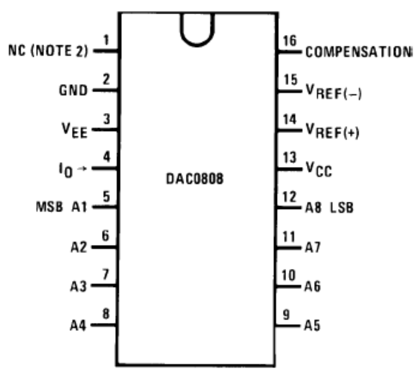
  - Special integrated circuits that generate a variable voltage output given a binary number input

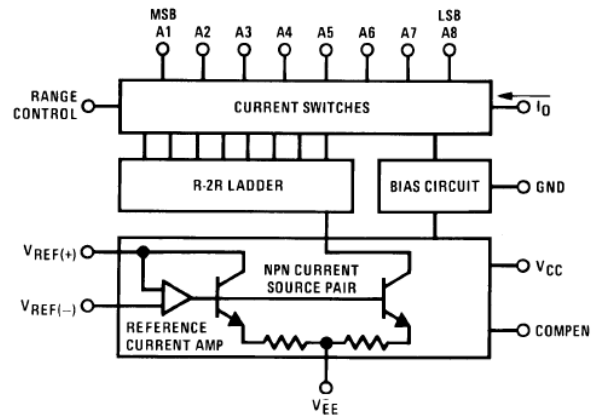


Digital to Analog Conversion

- Example: DAC0808 8-bit D/A Converter

Dual-In-Line Package



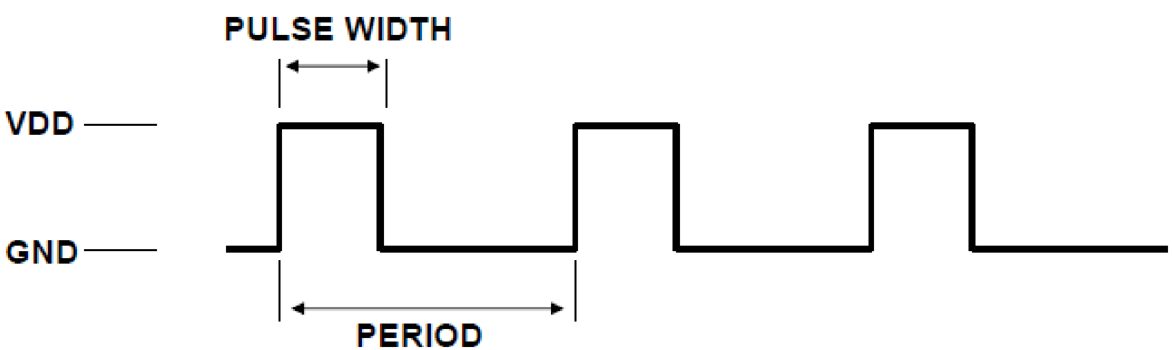


- Problem: D/A converters use a lot of MCU output pins.
- For instance, using 0808 chip we would need to use all pins from one entire port just to generate one analog voltage signal
- Most MCU’s do not have DAC’s built in

Pulse Width Modulation

- Pulse width modulation (PWM) is very common method of generating a variable power signal using single output pin
  - – Used often in motor and servo control
- Three quantities define PWM signal:

- Pulse width
- Period
- Voltage

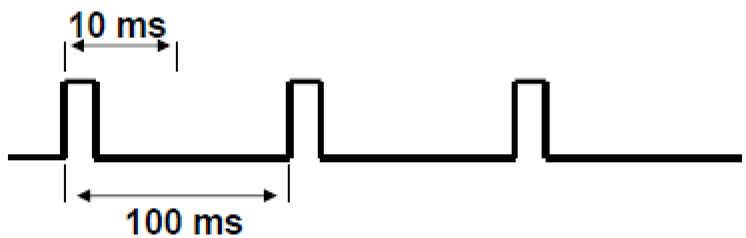


$$\text{Duty Cycle} = \frac{\text{Pulse Width}}{\text{Period}} \times 100\%$$

Duty Cycle

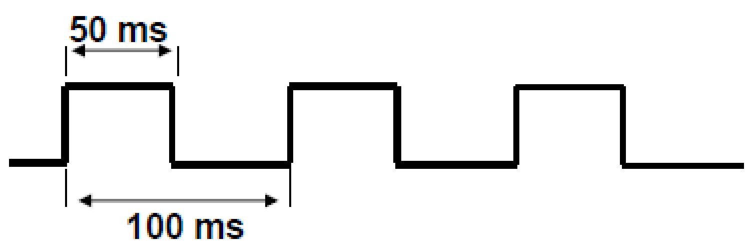
- Consider following 3 cases:

A)



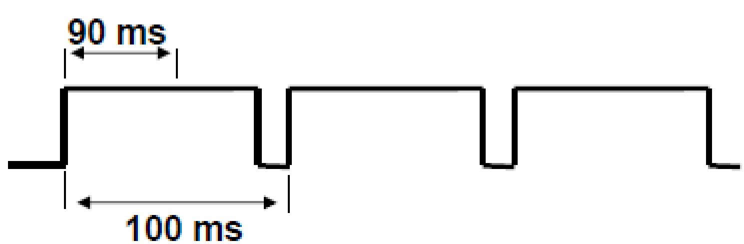
Duty Cycle = 10%

B)



Duty Cycle = 50%

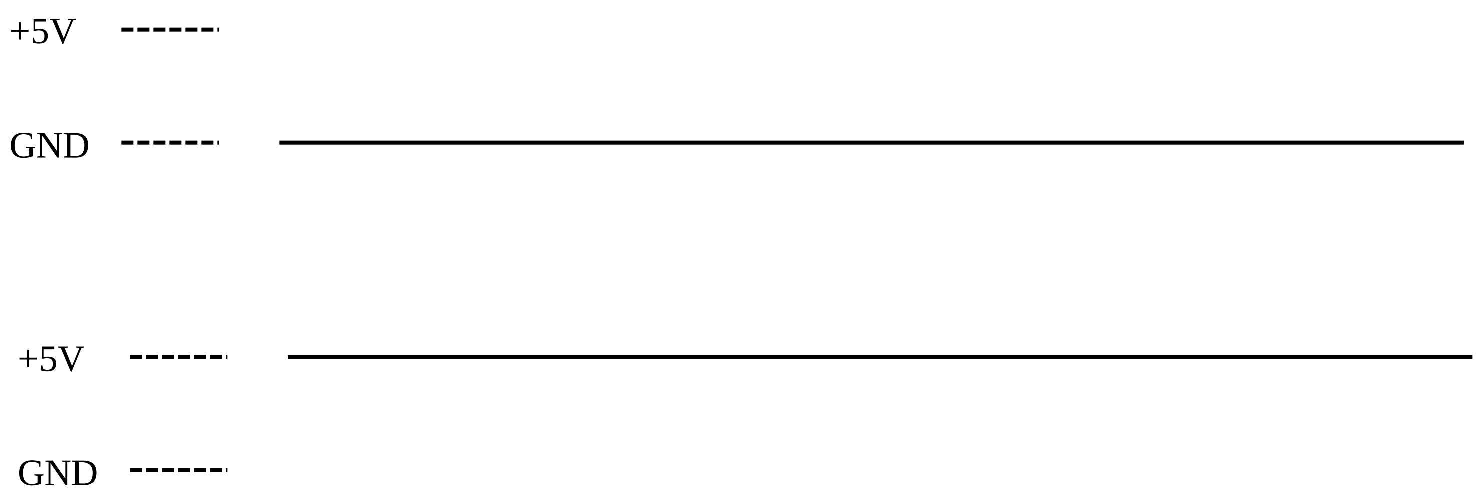
C)



Duty Cycle = 90%

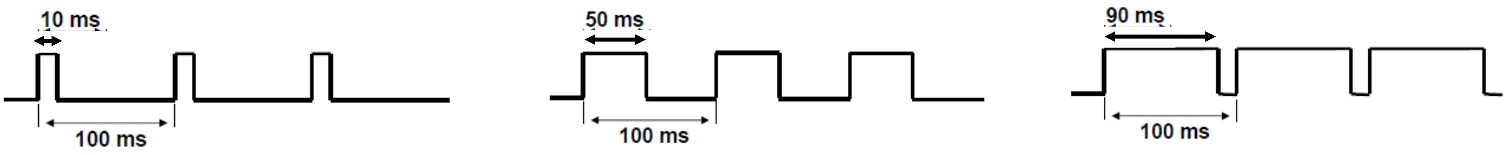
Duty Cycle

- Question: What is the duty cycle of the below two signals?



PWM Signal

- Notes:
  - PWM signal has a <u>fixed frequency</u> that is independent of the duty cycle

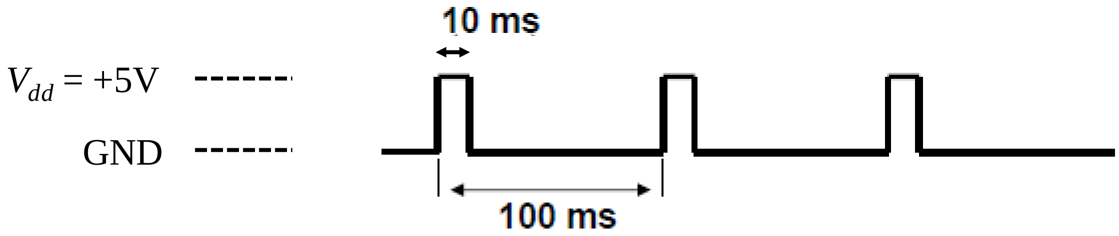


All these signals have a frequency of 1/100 ms = 10 Hz

  - PWM signal can be generated using a digital output pin by rapidly setting pin high/low (square wave signal)

PWM Signal as Variable Voltage Signal

- What is the voltage of the PWM signal averaged over 1 cycle?

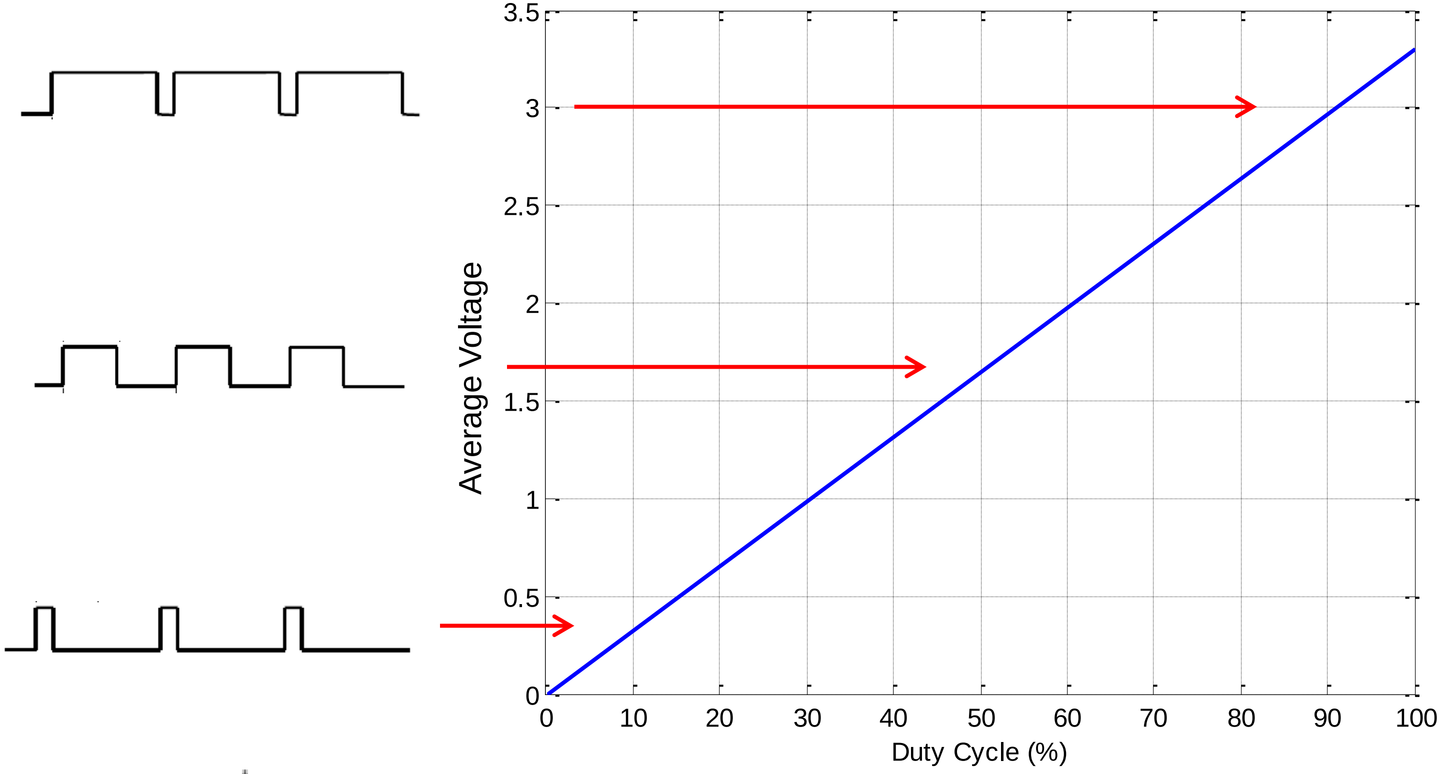


$$\bar{V} = \frac{(5\,\mathrm{V})(10\,\mathrm{ms}) + (0\,\mathrm{V})(90\,\mathrm{ms})}{100\,\mathrm{ms}} = 0.5\,\mathrm{V}$$

$\Rightarrow$ $\boxed{\text{Average voltage = (Duty cycle) x } (V_{dd})}$

PWM Signal as a Variable Voltage Signal

- Suppose V_{dd} = 3.3 V



PWM Signal as Variable Power Signal

- Let's assume that a certain circuit draws current *I* at voltage V_{dd}
  - What is average power delivered by PWM signal over one cycle?

$$\bar{P} = \frac{V_{dd} \times I \times T_{high} + 0 \times I \times T_{low}}{T_{high} + T_{low}} = V_{dd} \times I \times (\text{Duty Cycle})$$

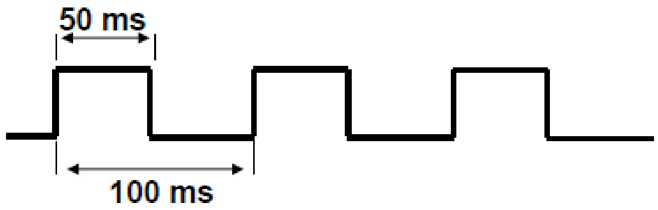
T_{high} = time signal is high

T_{low} = time signal is low

→ PWM also provides us with a method to generate a variable power signal

PWM Data Transmission

- PWM also offers a way to transmit digital data
  - Duty cycle (or pulse width) carries information
- Consider transmission of 8-bit number (0-255)
  - Achieved by using a PWM signal with period of 255 ms
  - Each 1 ms of pulse width signifies a bit

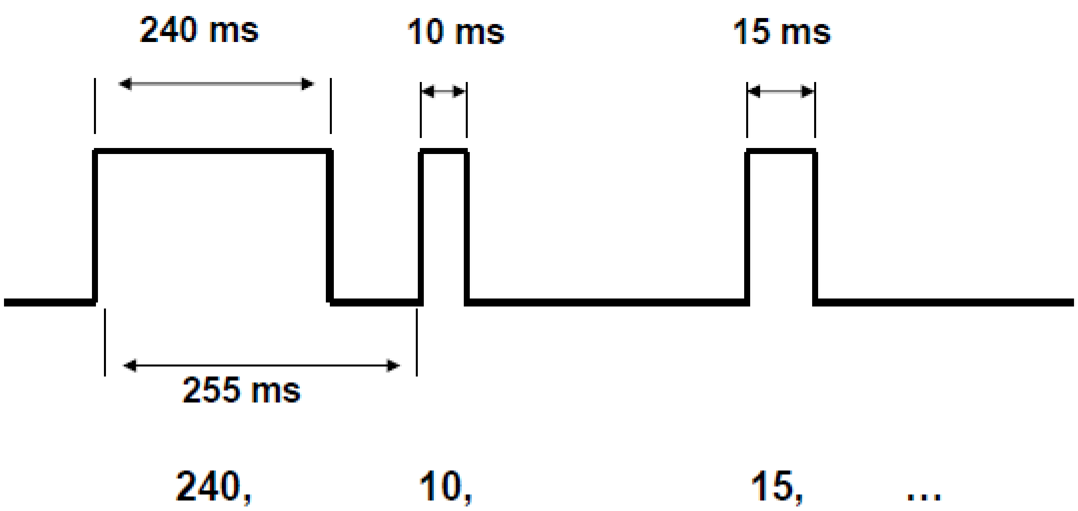


In this case, signal is transmitting number 128 (repeatedly).

- This is how servos work...see upcoming slides.

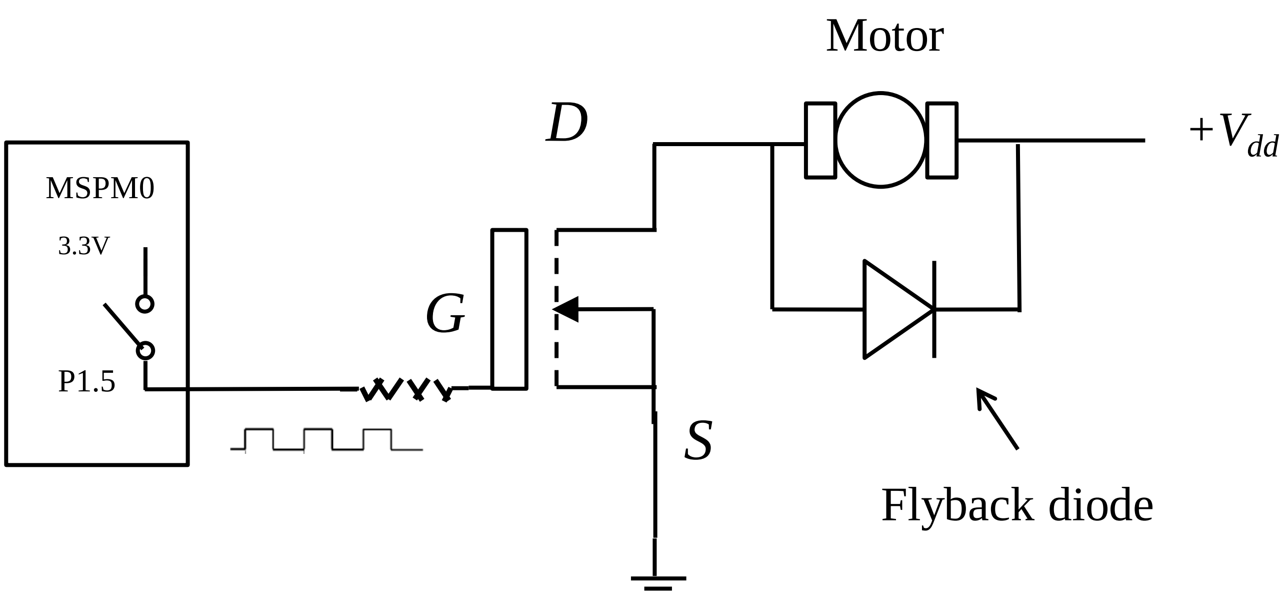
PWM Data Transmission

- What is PWM signal used to transmit sequence: {240, 10, 15}?
- Solution: Using period of 255 ms, we send a 240 ms pulse, 10 ms pulse, and 15 ms pulse



PWM Control of Mechanical Devices

- Suppose I drive a motor by activating the transistor gate with a PWM signal
  – What is the average voltage the motor sees?



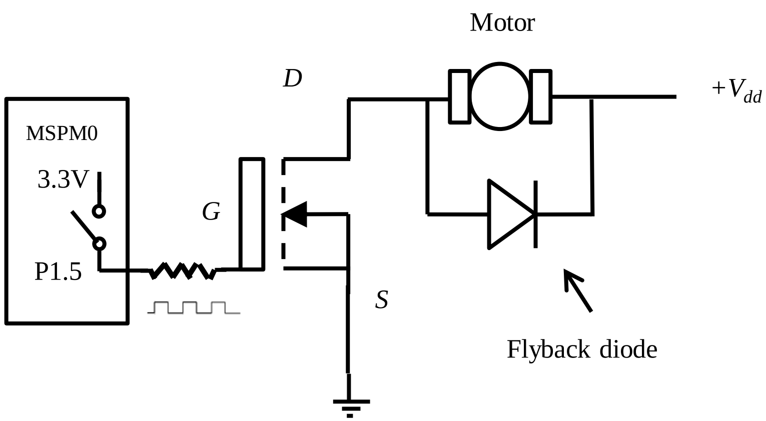
PWM Control of Mechanical Devices

- Suppose I drive a motor by activating the transistor gate with a PWM signal
  - What is the <u>average voltage the motor sees?</u>

$$\bar{V} = \frac{V_{dd} \times T_{high} + 0 \times T_{low}}{T_{high} + T_{low}}$$

$$= V_{dd} \times (\text{Duty Cycle})$$

Thus, if V_{dd} = 12 V, we can produce a signal with any average voltage in 0-12 V range by adjusting duty cycle between 0-100%.



Note: Switching times of transistors are in the ~10 ns range. PWM signal periods are in the 10's of ms range, so MOSFET switching time is negligible.

Georgia Tech

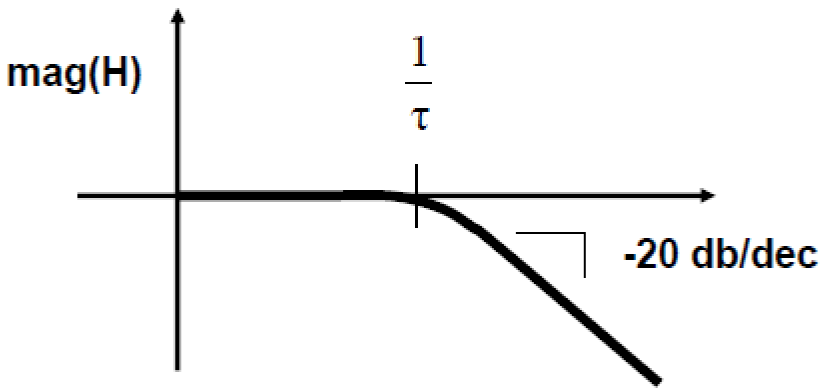
PWM Control of Mechanical Devices

- Question: When you give a PWM input voltage signal to a motor, why doesn't it respond to each square wave peak and valley?
  - i.e., Why doesn't it slow down and speed up with every pulse?
  - Explain in terms of frequency response.

PWM Control of Mechanical Devices

- Question: When you give a PWM input voltage signal to a motor, why doesn't it respond to each square wave peak and valley?
- Answer: PWM frequency is well beyond the cutoff frequency for a motor

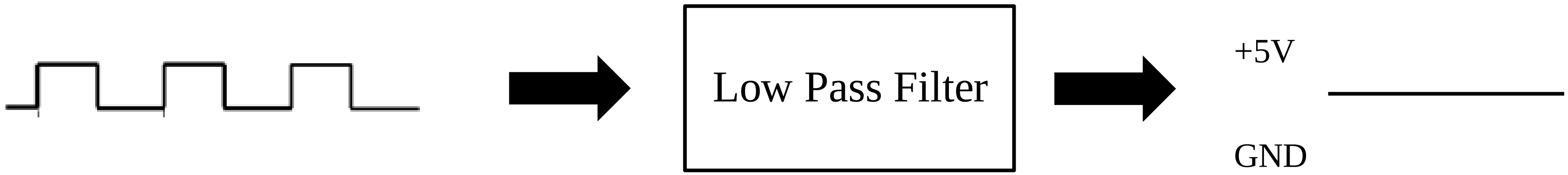
Bode Plot for RL Circuit



- RL circuit such as a motor has can be modeled as a first-order system with time constant τ.
- Cutoff frequency for such a system is $1/\tau$.
- Inputs with frequency higher than cutoff frequency get attenuated – meaning system does not respond to periodic input.

PWM Control of Mechanical Devices

- Like motors, most mechanical devices act as a low-pass filter
- If you low pass filter a PWM signal, the periodic content is lost and just the mean value of the signal is left
- Thus, PWM + low pass filter = analog voltage signal



Servo Control

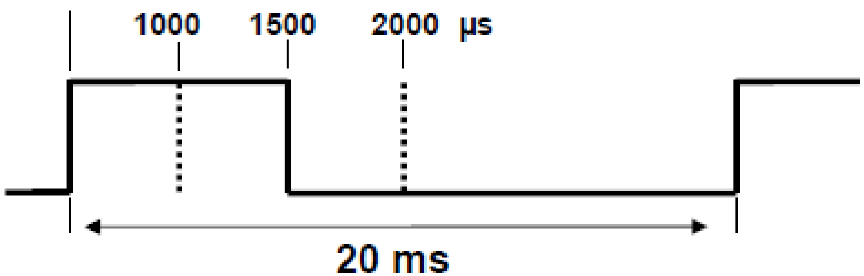
- Servos are used in many industrial and robotics applications
  - Provides precise positioning
  - Used often in control systems to generate system inputs
  - For instance, steer a R/C car
- Servos are complete system used for positioning incorporating:
  - Motor
  - Gears
  - Potentiometer for feedback

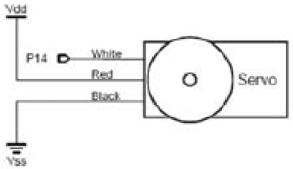
Parallax 900 Servo

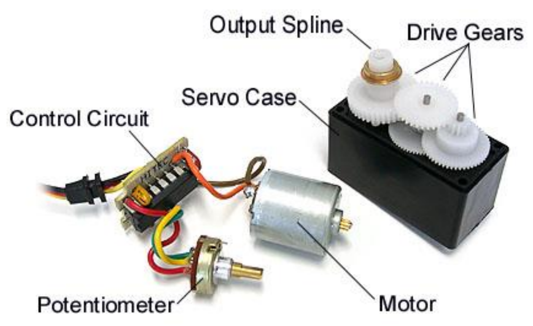
Servo Control

- Control input to servo is a PWM signal with 20 ms period
  - Valid duty cycle is 5%-10% (T_{high} = 1000 μs to 2000 μs)
- Servo wires:
  - Red = Power (5V)
  - Black = GND
  - White = PWM signal

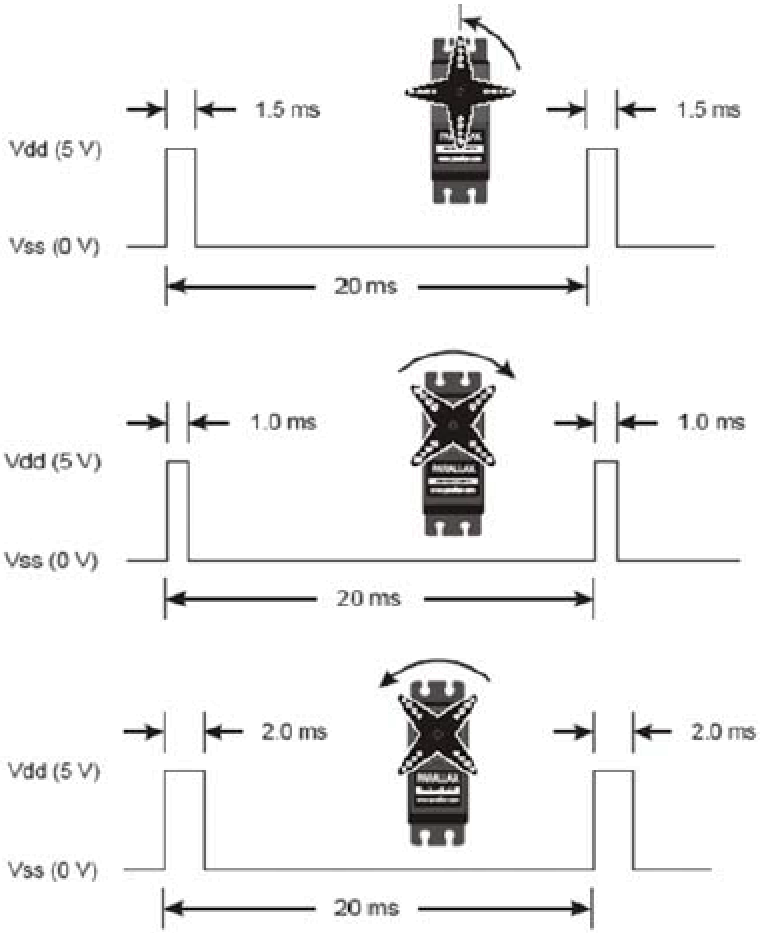
(Diagram not to scale)







Servo Control



Pulse width of 1500 μs (7.5% duty cycle) is neutral position.

Pulse width of 1000 μs (5% duty cycle) is 45 deg clockwise.

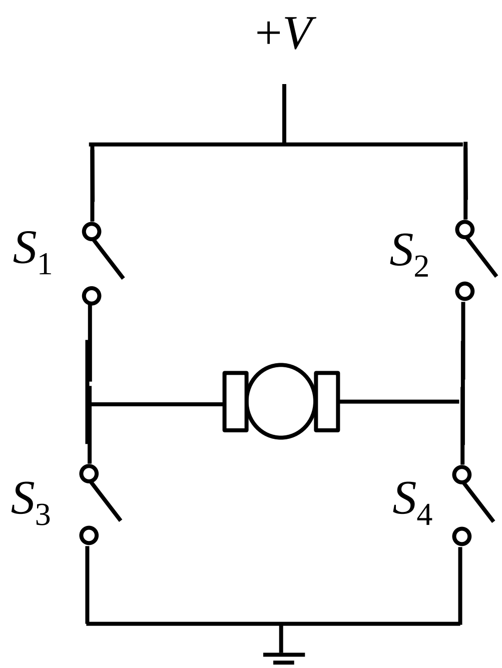
Stops in hardware usually limit rotation to 45 deg.

Pulse width of 2000 μs (10% duty cycle) is 45 deg counterclockwise.

Note: You must maintain PWM signal to servos continually during operation.

Motor Interfacing – H-Bridge

- Recall our discussion of H-bridges from Lecture 2
  - H-bridges are commonly used to drive DC motors since they can easily reverse direction
  - Most motors are controlled through H-bridges



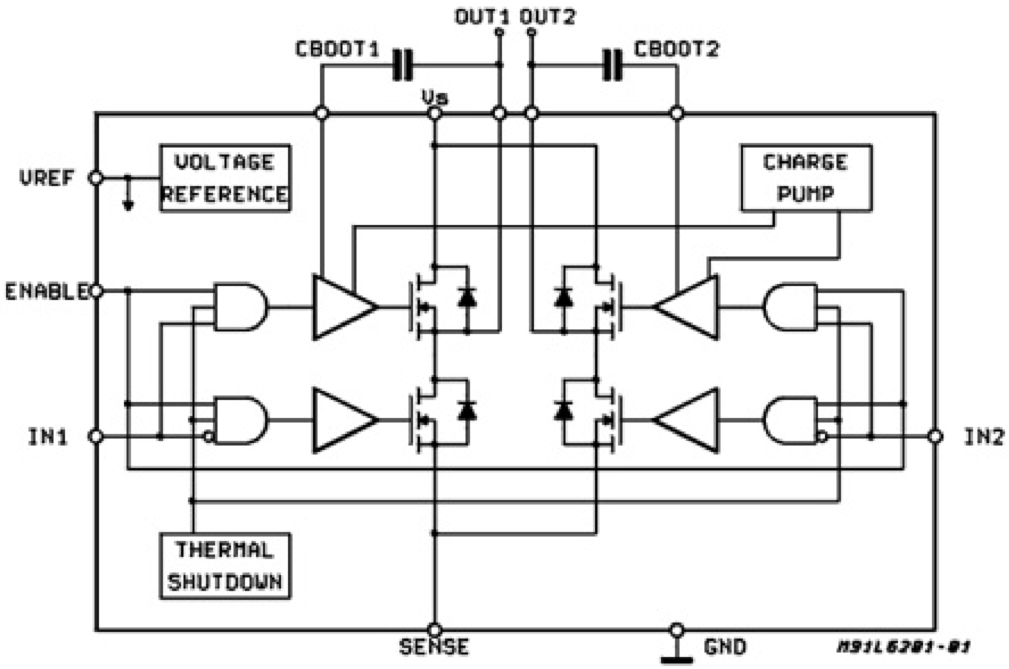
Simple H-Bridge Schematic

- When switches S_1 and S_3 are open and S_2 and S_4 are closed, motor turns one direction
- When switches S_1 and S_3 are closed and S_2 and S_4 are open, motor turns other direction

Georgia Tech

H-Bridges

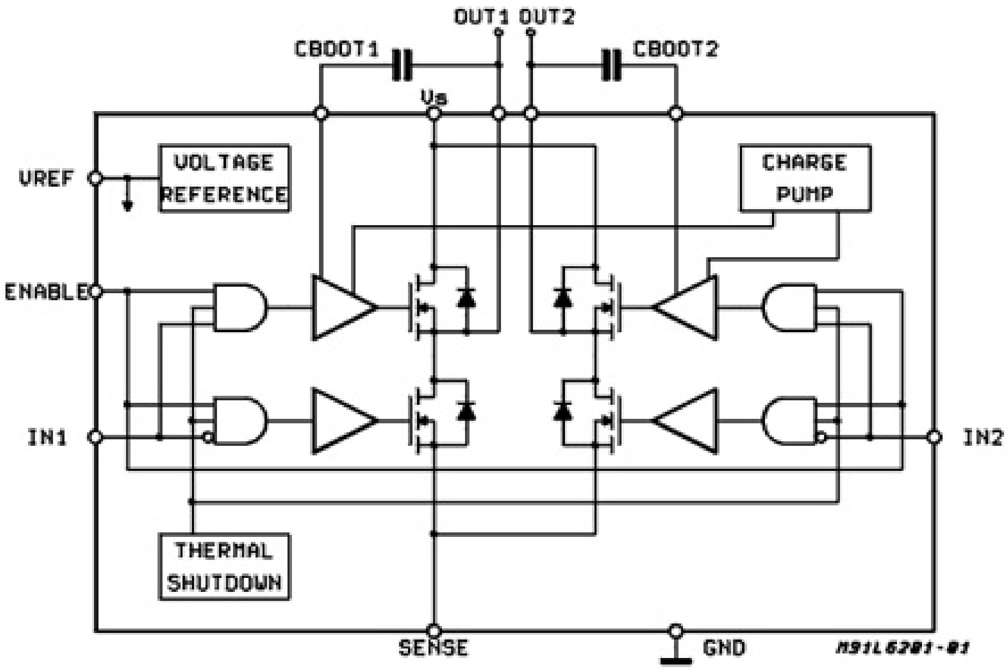
- Actual H-bridge IC's are constructed of transistors rather than mechanical switches
- Example: L6203 H-bridge



- Can supply up to 1 A at up to 48 V
- Three input signals: ENABLE, IN1, and IN2
- Motor power and ground leads connected to OUT1 and OUT2
- Note – flyback diodes already incorporated!

H-Bridge Operation

- L6203 H-bridge operation

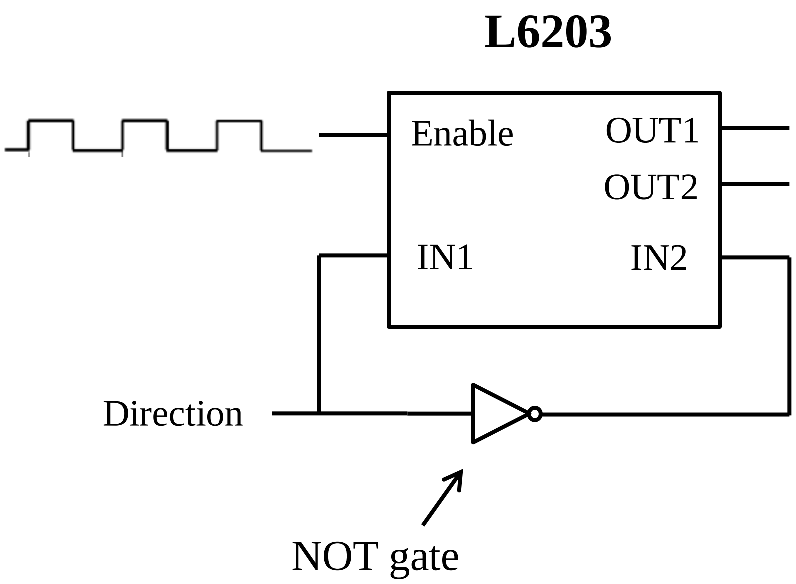


- ENABLE input must be high for H-bridge to operate
- V_{ref} is HIGH reference voltage

Input		Output	
IN1	**IN2**	**OUT1**	**OUT2**
High	Low	V_s	GND
Low	High	GND	V_s

Using PWM With H-Bridges

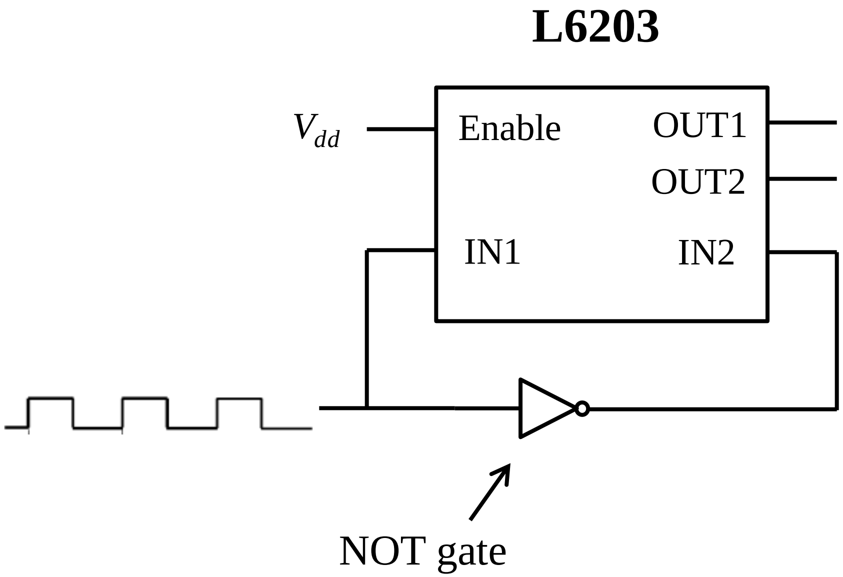
- PWM signals can be interfaced with H-bridges using two possible methods:
  - Sign magnitude method
  - Locked anti-phase method
- Sign magnitude method:



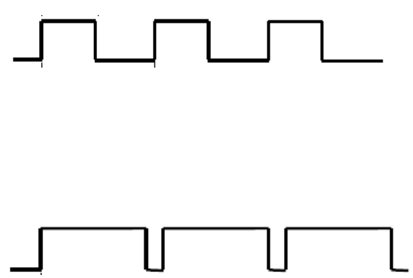
- MCU generates two signals: PWM and Direction (low or high)
- As PWM duty cycle increases, average voltage across OUT1 and OUT2 increases
- Direction (low or high) used to switch direction of motor

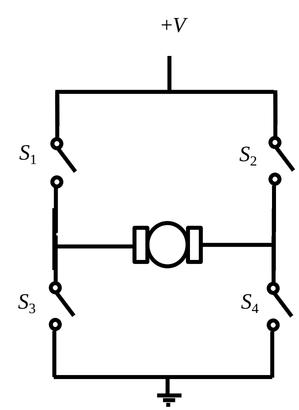
Using PWM With H-Bridges

- Locked anti-phase method:



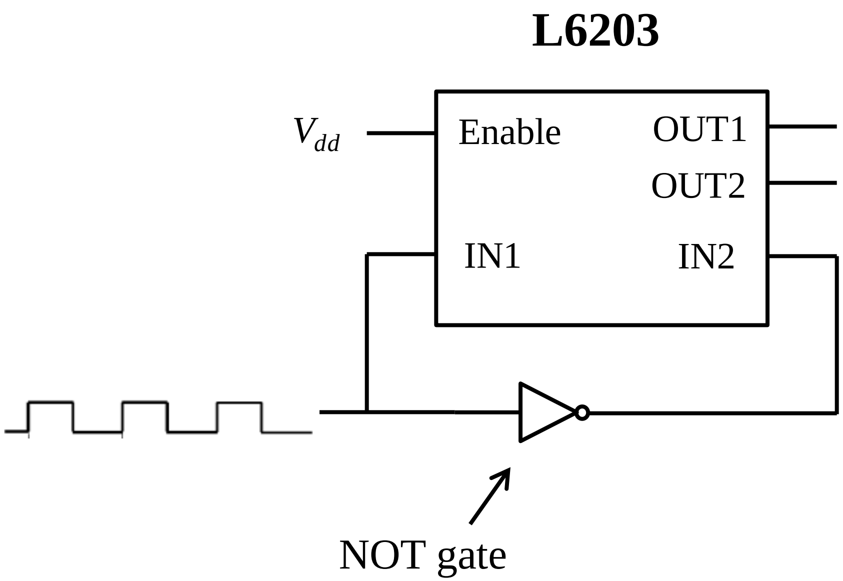
- MCU now generates only one signal: PWM
- PWM used to control both voltage delivered to load and direction of rotation
- PWM signal causes transistor pairs (S_1 and S_3, S_2 and S_4) to switch every PWM period
- If duty cycle is 50%, opposing pairs of switches close for same duration over cycle – no net voltage → no movement
- If duty cycle is 80%, a net 60% (80%-20%) voltage is applied during each cycle



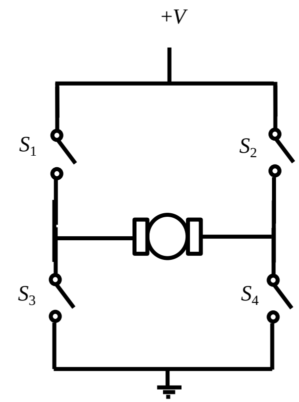


Using PWM With H-Bridges

- Locked anti-phase method:



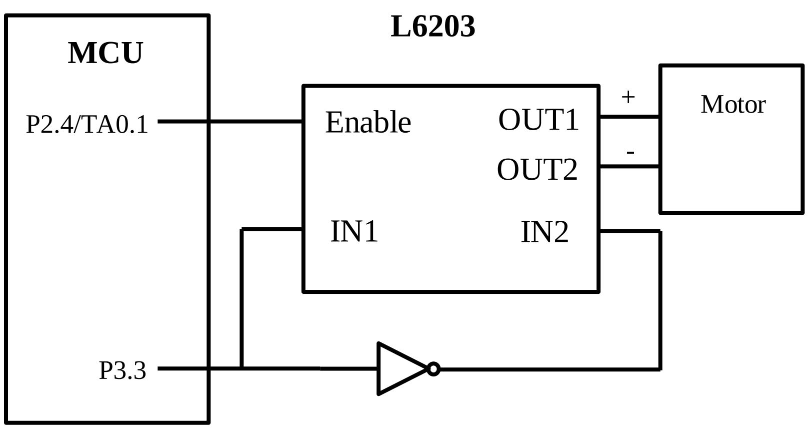
- Thus, when duty cycle is less than 50%, motor will spin in one direction
- When duty cycle is greater than 50%, motor will spin in other direction
- Maximum voltage applied to motor at both 0% and 100% duty cycles (in opposite directions)
- Advantage of this method: Single control signal used to control both direction and magnitude
- Disadvantage: Results in ripple currents through motor (generates heat + dissipates power)



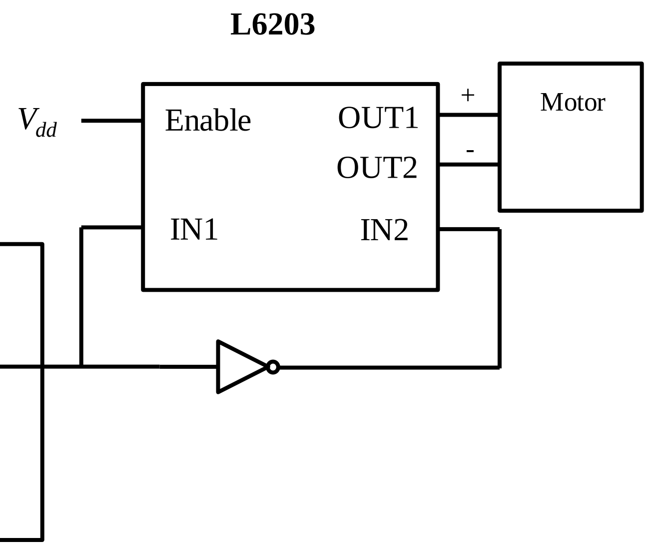
Using PWM with H-Bridges

- So interfacing motor and MCU:

<u>Sign Magnitude Method</u>



<u>Locked Anti-Phase Method</u>

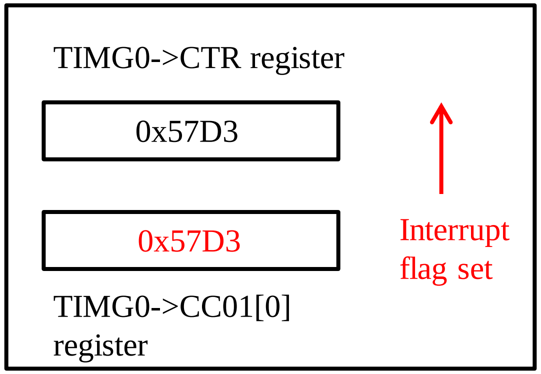


Generating PWM on MSPM0

- PWM signals are generated on MSPM0 using General Purpose Timer (TIMG) device
- TIMG has built-in capability to generate PWM signals through proper configuration
- Recall “compare” functionality of timers

Compare

Every time counter register reaches value in compare register, an interrupt flag is triggered.



Timer A capture / compare interrupt triggered

Generating PWM on MSPM0

- When TIMG operates in PWM mode, it switches signal high/low every time compare operation is true
  - How it does so can be configured

Down-Counting Timer

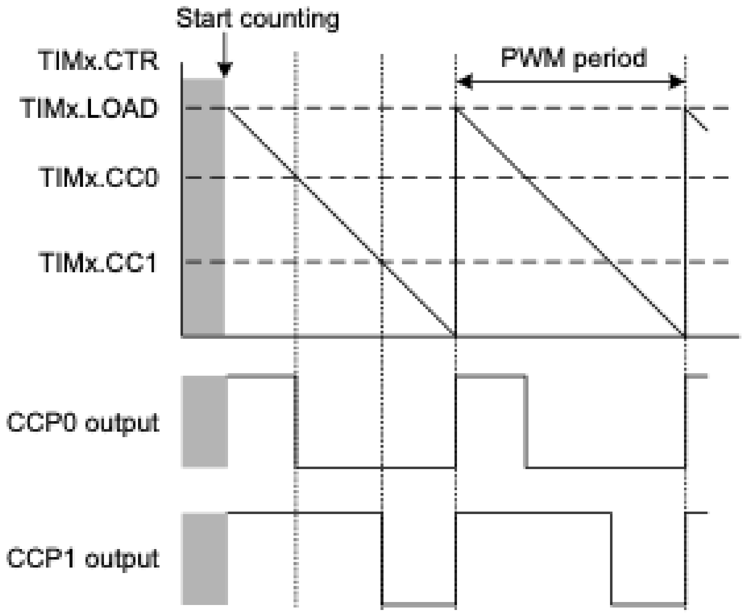


Figure 25-28. Edge-Aligned PWM Signals in Down-Counting Mode

Up-Counting Timer

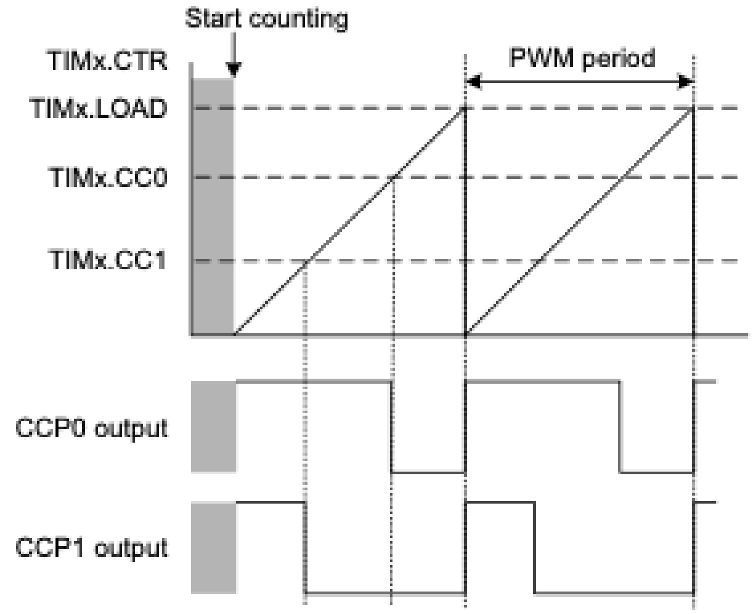


Figure 25-29. Edge-Aligned PWM Signals in Up-Counting Mode

Generating PWM on MSPM0

- Suppose we want to generate a PWM signal with:
  - Period of 16 ms
  - Duty cycle of 25%
  - TIMG has clock rate of 32 MHz

→ Timer is incremented once every 1 / 32,000,000 sec = 0.03125 μs

→ Set clock period (register TIMx.LOAD) to 512,000, since 512000 x 0.03125 μs = 16 ms

→ Put timer in up-counting mode. Set compare register TIMx.CC0 to 128000.

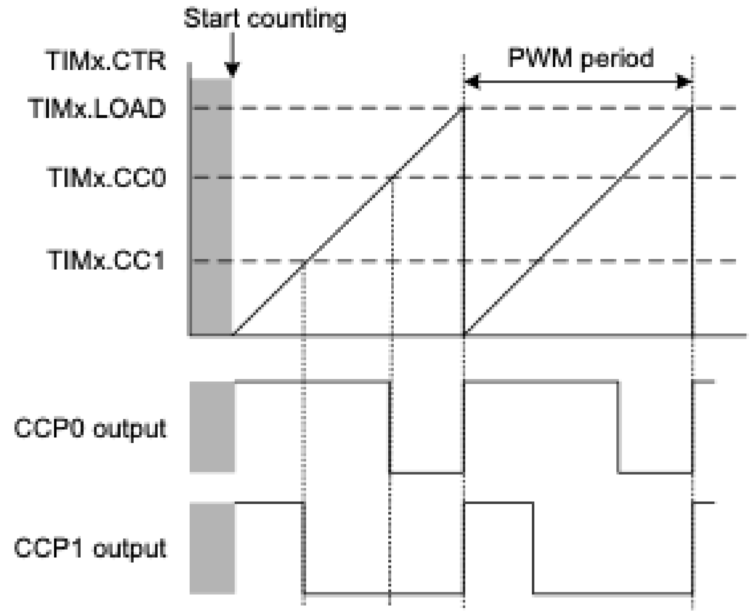


Figure 25-29. Edge-Aligned PWM Signals in Up-Counting Mode

TIMG Setup for PWM

- PWM outputs are available on pins labeled TIMGx_Cy
  - x is the timer module (x = 7, 12)
  - y = 0, 1 is the capture-compare output
- PWM setup
  - Set pin functionality
  - Set TIMG clock configuration
  - Set TIMG PWM configuration (up/down mode, period, etc.)
  - Set up capture mode
  - Set duty cycle
  - Set Capture/Compare mode to output (for PWM, rather than capture functionality)
  - Start TIMG counter

TIMG Setup for PWM

- Set pin functionality
  – Recall our table of pin functionality

30	PB13		UART3_RX *[2]* / TIMA0_C3 *[3]* / TIMG12_C0 *[4]* / TIMA0_C1N *[5]*	1	–	–	–	Standard

```
// PWM functionality on pin PB13 (TIMG12_C0)
DL_GPIO_initPeripheralOutputFunction(IOMUX_PINCM30,4);

// Enable output
DL_GPIO_enableOutput(GPIOB, DL_GPIO_PIN_13);
```

TIMG Setup for PWM

- Set TIMG clock configuration

Configuration Struct:

```
DL_TimerG_ClockConfig gPWM_0ClockConfig = {
    .clockSel = DL_TIMER_CLOCK_BUSCLK,
    .divideRatio = DL_TIMER_CLOCK_DIVIDE_1,
    .prescale = 0U
};
```

Setting clock configuration:

```
// Configure TIMG clock
DL_TimerG_setClockConfig(TIMG12, &gPWM_0ClockConfig);
```

TIMG Setup for PWM

- Set PWM configuration

Configuration Struct:

```
DL_TimerG_PWMConfig gPWM_0Config = {
   .pwmMode = DL_TIMER_PWM_MODE_EDGE_ALIGN_UP,
   .period = 512000,
   .startTimer = DL_TIMER_STOP,
};
```

Setting clock configuration:

```
// Configure PWM
DL_TimerG_initPWMMode(TIMG12, &gPWM_0Config);
```

TIMG Setup for PWM

- Set capture mode
  - Sets a bunch of options associated with PWM generation

```
void DL_Timer_setCaptureCompareOutCtl(GPTIMER_Regs * gptimer,
uint32_t ccpIV, uint32_t ccpOInv, uint32_t ccpO, DL_TIMER_CC_INDEX ccIndex)
```

- Set duty cycle

```
void DL_Timer_setCaptureCompareValue(GPTIMER_Regs *gptimer, uint32_t value, DL_TIMER_CC_INDEX ccIndex)
```

  - Places value in capture/compare register
  - Can be called at any time during program to change duty cycle

TIMG Setup for PWM

- Enable clock

```
void DL_Timer_enableClock (GPTIMER_Regs * gptimer )
```

- Set capture/compare direction

```
void DL_Timer_setCCPDirection(GPTIMER_Regs * gptimer, uint32_t ccpConfig)
```

  - Set to OUTPUT to output PWM on pin, set to INPUT to input digital signal to capture
  - e.g., DL_TIMER_CC0_OUTPUT sets capture/compare channel 0 to output

TIMG Setup for PWM

- Start timer counter

```
void DL_Timer_startCounter (GPTIMER_Regs *gptimer)
```

Example Code

- Let’s look at an example where we setup MSPM0 to generate a PWM output
  - Output pin is PB13, listed as TIMG12_C0 on datasheet
    - *Thus, x* = 12 *and y* = 0
  - Clock rate will be 32 MHz, sourced from BUSCLK
  - PWM period will be 16 ms, duty cycle will be 10%
- What should we use for:
  - Rollover value (TIMG12->COUNTERREGS.LOAD)?
  - Compare value (TIMG12->COUNTERREGS.CC_01[0]), if we use up counting mode?

Example Code

- Let's look at an example where we setup MSPM0 to generate a PWM output
  - Output pin is PB13, listed as TIMG12_C0 on datasheet
    - *Thus, x* = 12 *and y* = 0
  - Clock rate will be 32 MHz, sourced from BUSCLK
  - PWM period will be 16 ms, duty cycle will be 10%
- What should we use for:
  - Rollover value (TIMG12->COUNTERREGS.LOAD)?

$1/32{,}000{,}000 = 0.3125 \times 10^{-7}$ sec
0.3125×10^{-7} sec $\times\ 512000 = 16$ ms

TIMG12->COUNTERREGS.LOAD = 3000

  - Compare value (TIMG12->COUNTERREGS.CC_01[0]), if we use up counting mode?

$512000 \times 0.10 = 51200$

TIMG12->COUNTERREGS.CC_01[0] = 51200

```
// PWM functionality on pin PB13 (TIMG12_C0)
DL_GPIO_initPeripheralOutputFunction(IOMUX_PINCM30,4);
DL_GPIO_enableOutput(GPIOB, DL_GPIO_PIN_13);


// Configure TIMG
DL_TimerG_setClockConfig(TIMG12, &gPWM_0ClockConfig);

DL_TimerG_initPWMMode(TIMG12, &gPWM_0Config);


DL_TimerG_setCaptureCompareOutCtl(TIMG12, DL_TIMER_CC_OCTL_INIT_VAL_LOW, DL_TIMER_CC_OCTL_INV_OUT_DISABLED,
DL_TIMER_CC_OCTL_SRC_FUNCVAL, DL_TIMER_CC_0_INDEX);

// Set initial duty cycle
DL_TimerG_setCaptureCompareValue(TIMG12, duty_cycle, DL_TIMER_CC_0_INDEX);


// Enable timer clock
DL_TimerG_enableClock(TIMG12);


// Set CCP direction
DL_TimerG_setCCPDirection(TIMG12 , DL_TIMER_CC0_OUTPUT) ;


// Start counter
DL_TimerG_startCounter(TIMG12);
```

Changing Duty Cycle

- At any time in our code, we can change the value in our compare register (TIMG12->COUNTERREGS.CCxx), which changes the duty cycle automatically
  - No additional steps required.
- To change value in this register just call:

```
void DL_Timer_setCaptureCompareValue(GPTIMER_Regs *gptimer, uint32_t value, DL_TIMER_CC_INDEX ccIndex)
```