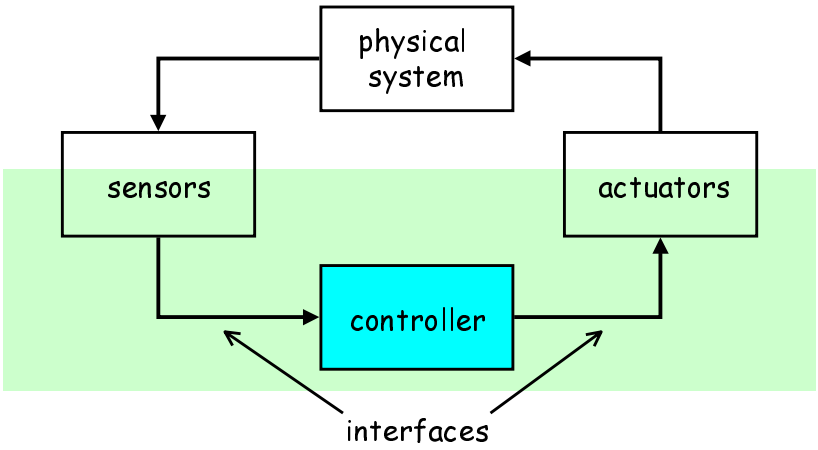
Controlling and reacting to the environment

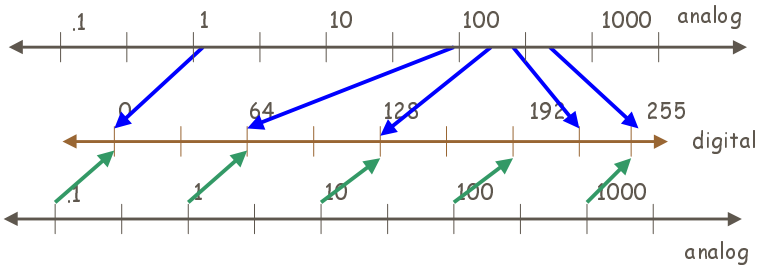
- To control or react to the environment we need to interface the microcontroller to peripheral devices
- Things we want to measure or control
  - light, temperature, sound, pressure, velocity, position
- Sensors and actuators
  - switches, photoresistors, photodiodes, phototransistors, compass, sonar
  - motors, relays, LEDs, sonar
- Software
- Microcontroller
  - executes software
  - may contain specialized interfaces to sensors and actuators

Typical control system



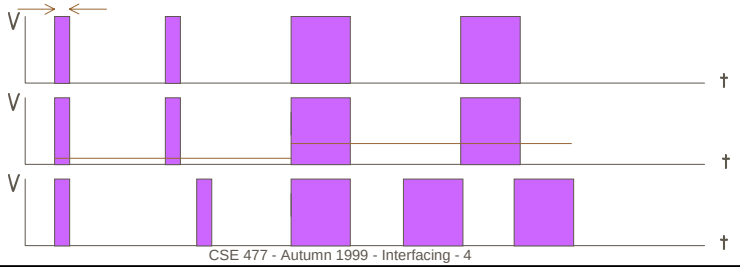
Analog to digital conversion

- Map analog inputs to a range of binary values
  - 8-bit A/D has outputs in range 0-255
- What if we need more information?
  - linear vs. logarithmic mappings
  - larger range of outputs (16-bit a/d)



Digital to analog conversion

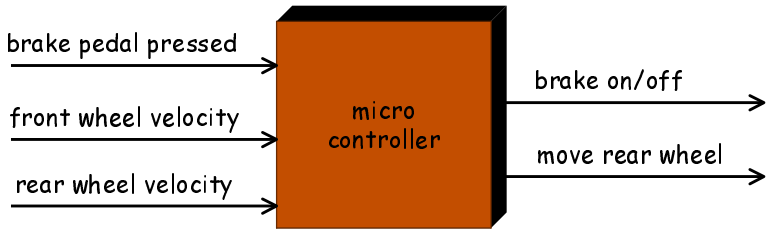
- Map binary values to analog outputs (voltages)
- Most devices have a digital interface
- Time-varying digital signals
  - pulse-code modulation (width of pulse is data)
  - pulse-width modulation (creates an average voltage)
  - frequency modulation (number of cycles is data)



Anti-lock brake system

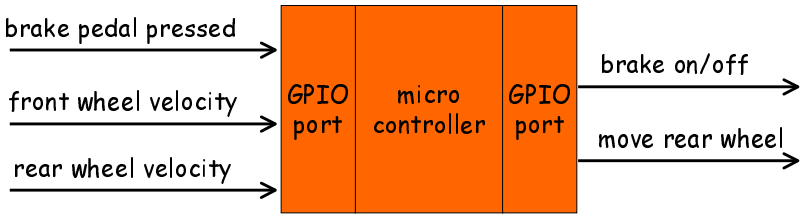
- Rear wheel controller/anti-lock brake system
  - normal operation
    - regulate velocity of rear wheel
  - brake pressed
    - gradually increase amount of breaking
    - if skidding (front wheel is moving much faster than rear wheel) then temporarily reduce amount of breaking
- Inputs
  - brake pedal
  - front wheel
  - rear wheel
- Outputs
  - pulse-width modulation rear wheel velocity
  - pulse-width modulation brake on/off

Rear wheel controller/anti-lock brake system



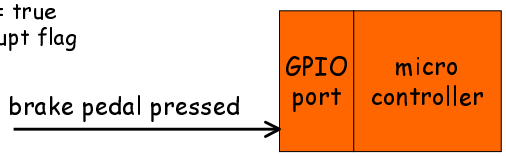
Basic I/O ports (brakes)

- Check if brake pedal pressed
  - brakePressed = read (brakePedalPort)
- Turn brake on/off
  - write (brakePort, onOff)
- Move rear wheel
  - write (rearWheel, onOff)



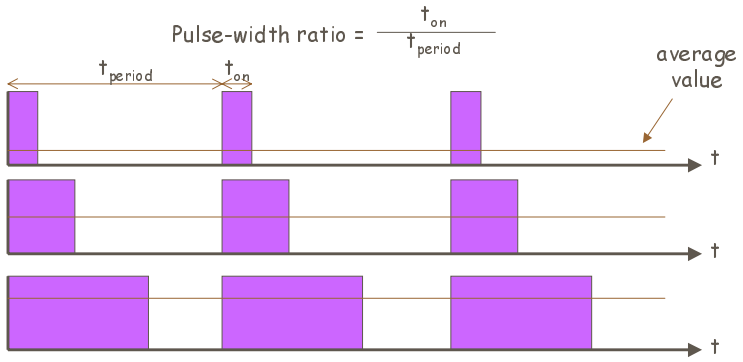
Polling vs. interrupts

- Software must repeatedly check
  - brake pedal port
  - how often?
  - need to make sure not to forget to do so (use timer)
- Use automatic detection capability of processor
  - connect brake pedal to GPIO input capture
  - interrupt on level change
  - register interrupt handler for brakePedalHandler
  - interrupt handler
    - if brakePressed = true
      then clear interrupt flag



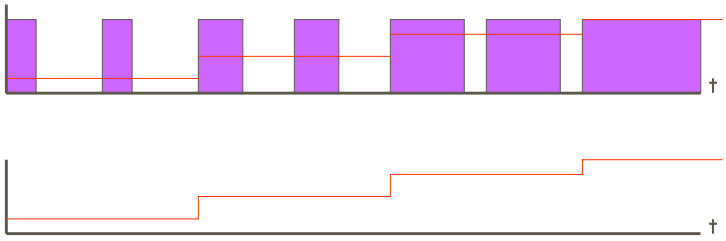
Pulse-width modulation

- Pulse a digital signal to get an average "analog" value
- The longer the pulse width, the higher the voltage



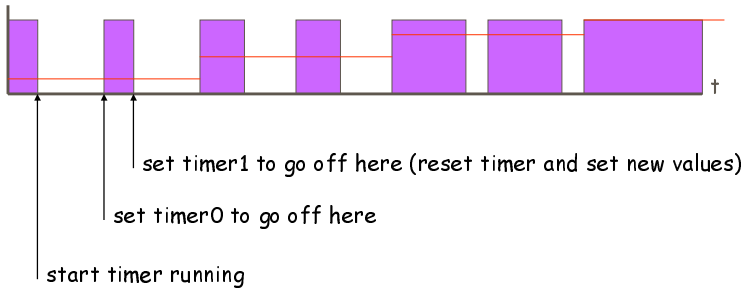
Pulse-width modulation for brakes

- To pump the brakes gradually increase the duty-cycle (t_{on}) until car stops



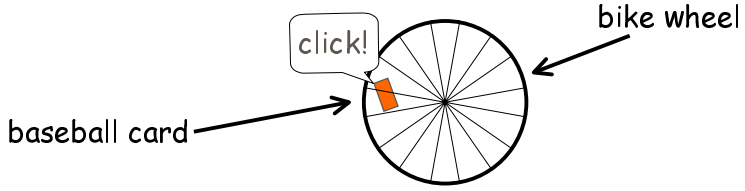
Brake pump setup

- Use two timer registers to turn brake on and off
  - timer0 applies break
  - timer1 releases break
  - how do we tell which interrupt is which?



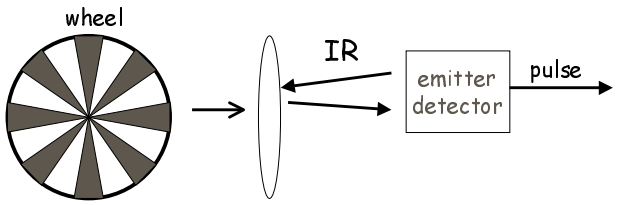
Shaft encoders

- Need to determine the rear wheel velocity
  - use sensor to detect wheel moving
- Determine speed of a bicycle
  - attach baseball card so it pokes through spokes
  - we know number of spokes
  - count clicks per unit time to get velocity
- Baseball card sensor is a shaft encoder



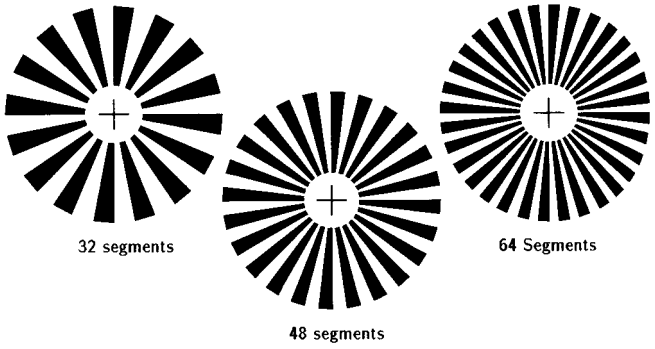
Shaft encoders

- Instead of spokes we'll use black and white segments
- Black segments absorb infrared light, white reflects
- Count pulses instead of clicks



IR reflective patterns

- How many segments should be used?
  - more segments give finer resolution
  - fewer segments require less processing



Interfacing shaft encoders

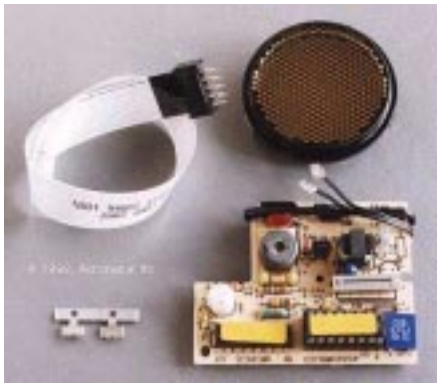
- Use interrupt on GPIO pin
  - every interrupt, increment counter
- Use timer to set period for counting
  - when timer interrupts, read GPIO pin counter
  - velocity = counter * distance per click / period
  - reset counter
- Pulse accumulator function
  - common function
  - some microcontrollers have this in a single peripheral device

Sonar range finder

- Uses ultra-sound (not audible) to measure distance
- Time echo return
- Sound travels at approximately 343m/sec
  - need 34.3kHz timer for cm resolution
- One simple echo not enough
  - many possible reflections
  - want to take multiple readings for high accuracy

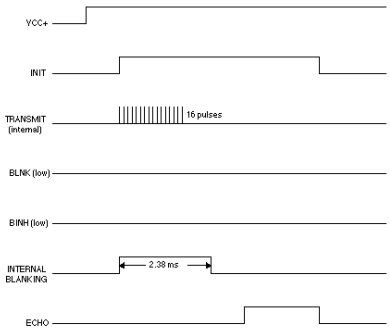
Polaroid 6500 sonar range finder

- Commonly found on old Polaroid cameras, now a frequently used part in mobile robots
- Transducer (gold disc)
  - charged up to high voltage and "snapped"
  - disc stays sentisized so it can detect echo (acts as microphone)
- Controller board
  - high-voltage circuitry to prepare disc for transmitting and then receiving (careful !!!)



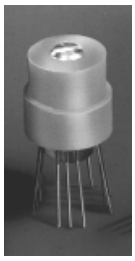
Polaroid 6500 sonar range finder (cont'd)

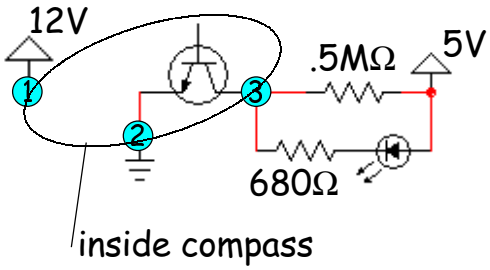
- We'll use it in its simplest mode
- Only need to connect two pins to microcontroller
  - INIT - start transmitting
  - ECHO - return signal
- Some important tips
  - INIT requires large current (greater than microcontroller can provide)
  - ECHO requires a pull-up resistor

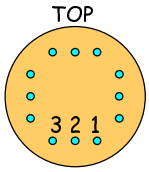


Digital compass

- Four compass directions (each has three pins)
- One-hot/two-hot encoding
  - one-hot for N, E, S, W
  - two-hot for NE, SE, SW, NW

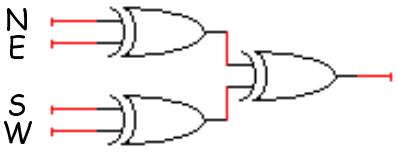






Digital compass (cont'd)

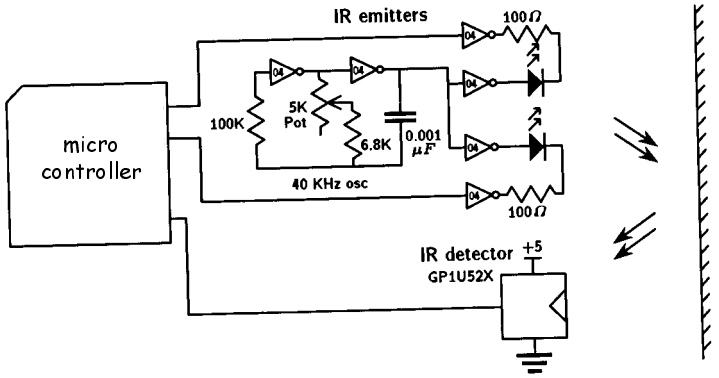
- Detecting a change in compass direction
  - 4 bits change from 0001 to 0011 to 0010 to 0110 to 0100 …
  - always alternating between one bit on and two bits on
- Parity tree can detect difference between one and two bits being asserted
  - XOR tree of four bits (one TTL SSI package)
  - output must change at least once for every change in orientation



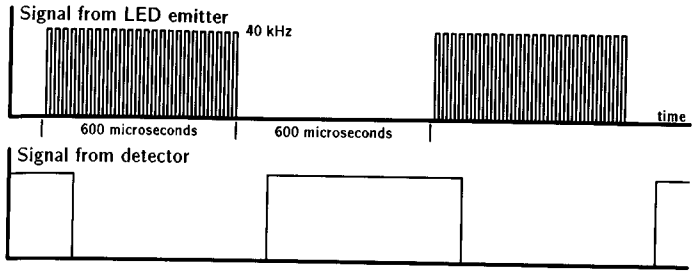
e.g., NE -> E -> SE
1100 -> 0100 -> 0110
0 -> 1 -> 0

IR proximity detector

- Oscillator must be set to match detector



IR frequency modulation

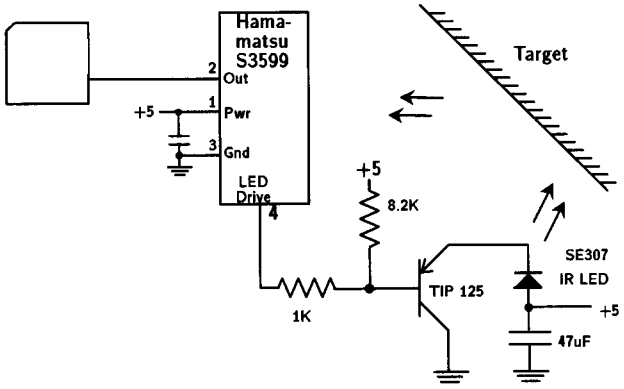


Proximity code

```
void ir_detect() {
       turn on emitter
       sleep for 600us                  // wait at 1667Hz
       val_on = read detector           // emitter is on
       turn off emitter
       sleep for 600us                  // wait at 1667Hz
       val_off = read detector          // while emitter off
       return( val_on & ~val_off );
}
```


Another proximity detector

- Detector drives LED (guaranteed to match frequency)

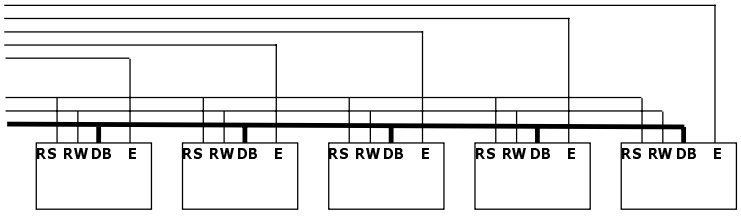


I/O ports

- The are never enough I/O ports
- Techniques for creating more ports
  - port sharing with simple glue logic
  - decoders/multiplexors
  - memory-mapped I/O
  - port expansion units
- Direction of ports is important
  - single direction port easier to implement
  - timing important for bidirectional ports

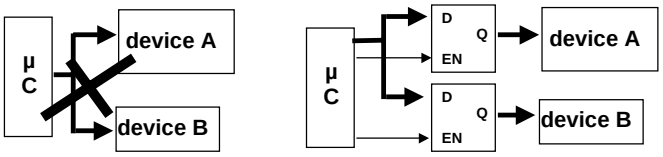
Port sharing

- If signals all in same direction and have a separate guard signal, then able to share without glue logic
- Example: connect 5 LCD displays to microcontroller
  - can share connections to RS, RW, and DB but not E
  - changes on E affect display – must guarantee only one is activated



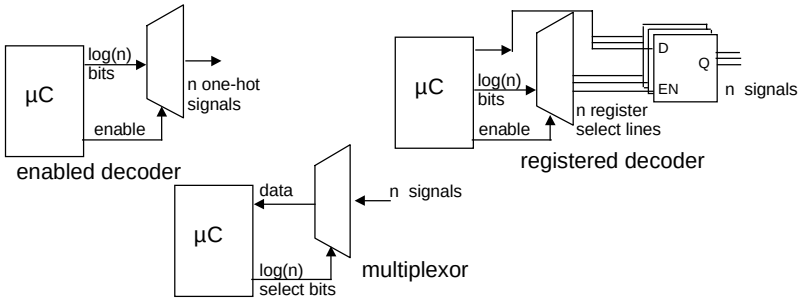
Forced sharing

- Conflict on device signals (e.g., one signal can affect both)
  - solution is to insert intervening registers that keep signals stable
  - registers require enable signals which now need ports as well



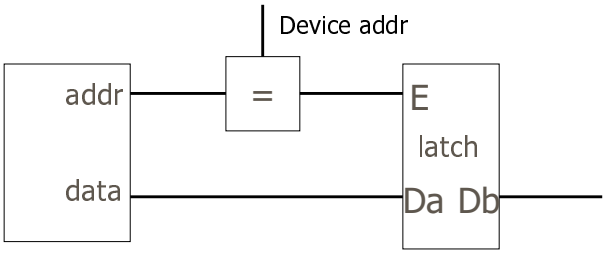
Decoders and multiplexors

- Encode n single-bit device ports using log n bits of a controller port
  - enabled decoder: one-hot, input-only device ports
  - registered decoder: input-only (but not one-hot) device ports
  - multiplexor: output-only device ports



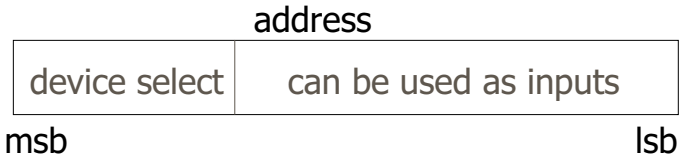
Memory-mapped I/O

- Address bus selects device
- Data bus contains data



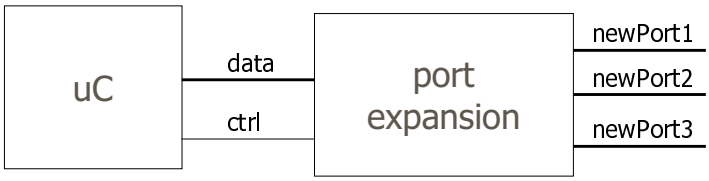
Memory-mapped I/O

- Partition the address space
- Assign memory-mapped locations
- Software
  - loads read from the device
  - stores write to the device
- Can exploit unused bits for device input-only ports



Port expansion units

- Problem of port shortage so common port expansion chips exist
- Easily connect to the microprocessor
- Timing on ports may be slightly different
- May not support interrupts

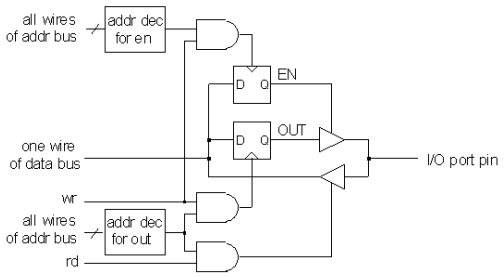


Automatically connecting peripherals

- Exploit specialized functions (e.g., UART, timers)
- Attempt to connect directly to a device port without adding interface hardware (e.g., registers), try to share registers if possible but beware of unwanted interactions if a signal goes to more than one device
- If out of ports, must force sharing by adding hardware to make a dedicated port sharable (e.g., adding registers and enable signals for the registers)
- If still run out of ports, then most encode signals to increase bandwidth (e.g., use decoders)
- If all else fails, then backup position is memory-mapped I/O, i.e., what we would have done if we had a bare microprocessor

64-bit I/O port

- Suppose we wanted a 64-bit I/O port
- If EN is true, then we have an output pin
- If EN is false, then we have an input pin

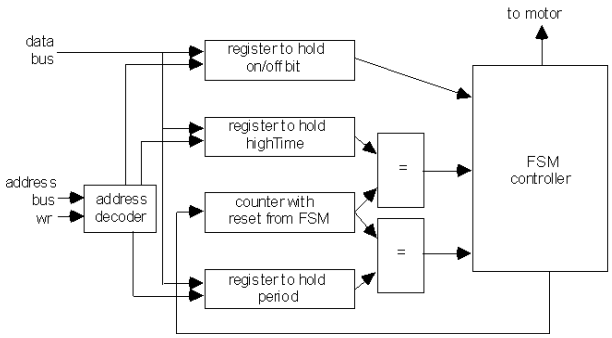


64-bit I/O port software

- We need 8 8-bit registers to store/write the 64 bits
  - Select the EN addresses to be $...000 to $...007
  - Select OUT addresses to be $...010 to $...017
- Read 15th bit
  - load value at address $...011 (2nd set of OUT regs)
  - logical AND with 0x80
  - bit position 7 of result is 15th bit
- Write the 47th bit
  - read OUT register at $...015
  - set bit position 7 to desired value (or with 0x80)
  - store in $...015
  - load EN register at $...005
  - set bit to output
  - store value back to $...005

External PWM Unit

- Design a system to control many digital motors
- Solution: design a PWM unit



External PWM FSM Controller

```
if (onOff == OFF)
    nextState = MotorLow
    reset counter
else if (period NOT Expired)
    nextState = MotorLow
else if (period Expired)
    nextState = MotorHigh
    reset counter
```

```
if (onOff == OFF)
    nextState = MotorLow
else if (highTime Expired)
    nextState = MotorLow
else if (highTime NOT Expired)
    nextState = MotorHigh
```

Motor Low State

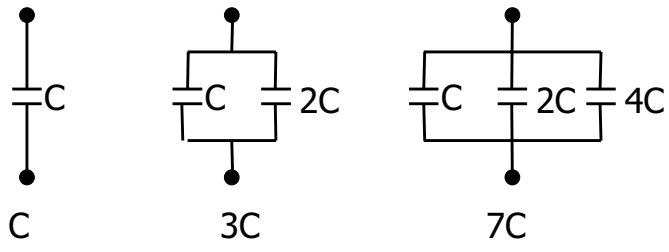
Motor High State

External PWM software

```
// in initialization code
Write off to onOff register

// do some stuff

// set up PWM
Repeat for each motor
    Write highTime and period registers

// turn motors on
Repeat for each motor
    Write on to the onOFF register

// more stuff
```


Analog to digital conversion

- Use charge-redistribution technique
  - no sample and hold circuitry needed
  - even with perfect circuits quantization error occurs
- Basic capacitors
  - sum parallel capacitance



Analog to digital conversion

- Two reference voltage
  - mark bottom and top end of range of analog values that can be converted (V_L and V_H)
  - voltage to convert must be within these bounds (V_X)
- Successive approximation
  - most approaches to A/D conversion are based on this
  - 8 to 16 bits of accuracy
- Approach
  - sample value
  - hold it so it doesn't change
  - successively approximate
  - report closest match

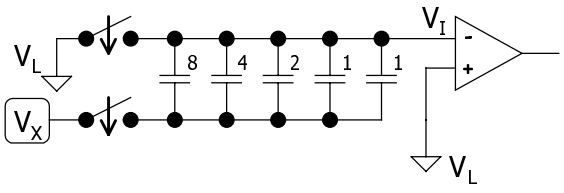
V_H

V_X

V_L

A/D - sample

- During the sample time the top plate of all caps switched to reference low V_L
- Bottom plate set to unknown analog input V_X
- $Q = CV$
- $Q_S = 16 (V_X - V_L)$



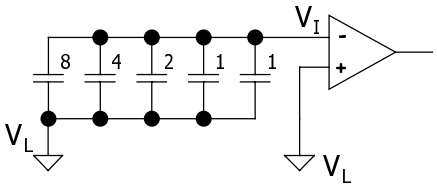
V_H

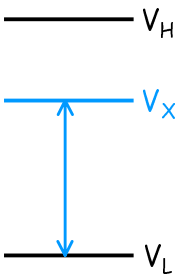
V_X

V_L

A/D - hold

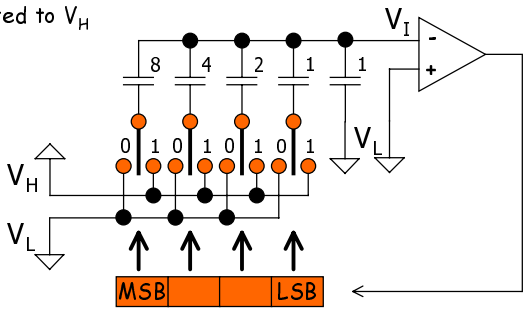
- Hold state using logically controlled analog switches
  - Top plates disconnected from V_L
  - Bottom plates switched from V_X to V_L
- $Q_H = 16\ (V_L - V_I)$
  - conservation of charge $Q_S = Q_H$
  - $16\ (V_X - V_L) = 16\ (V_L - V_I)$
  - $V_X - V_L = V_L - V_I$ (output of op-amp)





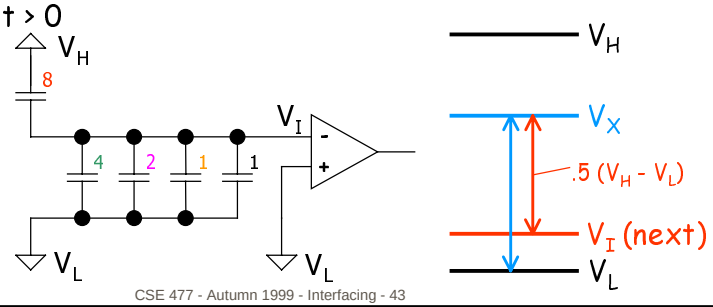
A/D - successive approximation

- Each capacitor successively switched from V_L to V_H
  - Largest capacitor corresponds to MSB
- Output of comparator determines bottom plate voltage of cap
  - > 0 : remain connected to V_H
  - < 0 : return to V_L



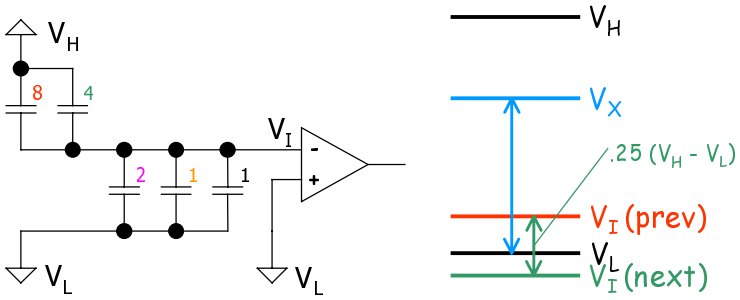
A/D example - MSB

- Suppose $V_X = 21/32\ (V_H - V_L)$ and already sampled
- Compare after shifting half of capacitance to V_H
  - V_I goes up by $+ 8/16\ (V_H - V_I) - 8/16\ (V_L - V_I) = + 8/16\ (V_H - V_L)$
  - original $V_L - V_I$ goes down and becomes
  - $V_L - (\ V_I + .5\ (V_H - V_L)\ ) = V_L - V_I - .5\ (V_H - V_L)$
- Output > 0



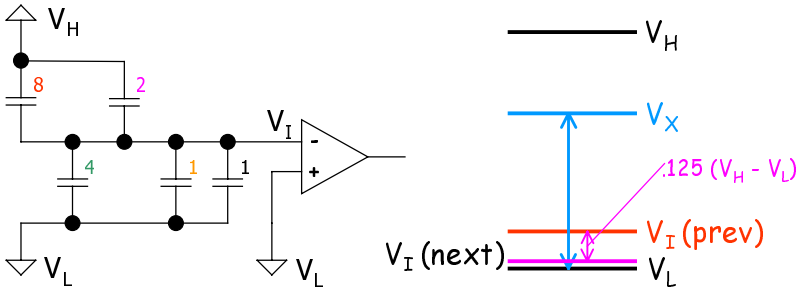
A/D example - (MSB-1)

- Compare after shifting another part of cap. to V_H
  - V_I goes up by $+ 4/16\ (V_H - V_I) - 4/16\ (V_L - V_I) = + 4/16\ (V_H - V_L)$
  - original $V_L - V_I$ goes down and becomes
  - $V_L - (\ V_I + .25\ (V_H - V_L)\ ) = V_L - V_I - .25\ (V_H - V_L)$
- Output < 0 (went too far)



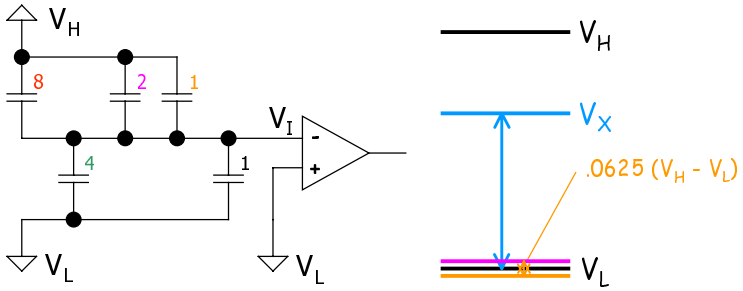
A/D example - (MSB-2)

- Compare after shifting another part of cap. to V_H
  - V_I goes up by + 2/16 (V_H-V_I) - 2/16 (V_L-V_I) = + 2/16 (V_H - V_L)
  - original V_L - V_I goes down and becomes
  - V_L - (V_I + .125 (V_H - V_L)) = V_L - V_I - .125 (V_H - V_L)
- Output > 0



A/D example - LSB

- Compare after shifting another part of cap. to V_H
  - V_I goes up by + 1/16 (V_H-V_I) - 1/16 (V_L-V_I) = + 1/16 (V_H - V_L)
  - original V_L - V_I goes down and becomes
  - V_L - (V_I + .0625 (V_H - V_L)) = V_L - V_I - .0625 (V_H - V_L)
- Output < 0 (went too far again)



A/D example final result

- Input sample of 21/32
- Gives result of 1010 or 10/16 = 20/32
- 3% error