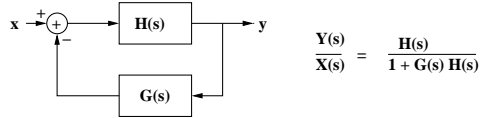


## 23.1 Controlling Position

- Servomechanisms are of this form:

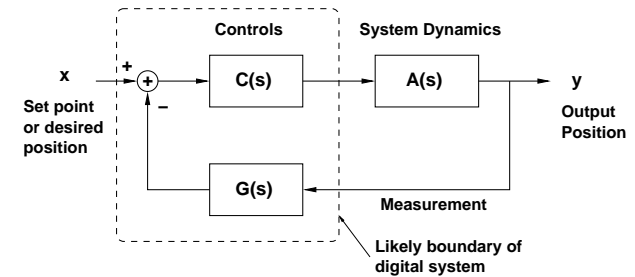


- Some General Features of Servos:

- They are feedback circuits.
- Natural frequencies are 'zeros' of  $1+G(s)H(s)$ .
- System is unstable if these zeros are in the right half plane.
- 'Negative' feedback becomes positive with 180 degree phase shift.
- Putting an integrator into  $H(s)$  drives steady error to zero.
- But high order systems are more likely to have RHP zeros.
- Time delay and high gain lead to RHP zeros.

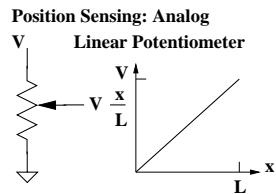
## 23.2 Digital Servos

- Major parts of the system are digital.
  - Digital systems are more flexible to design.
  - More repeatable (not subject to gain drift)
- Analog parts are important,
  - But in many cases can be avoided.



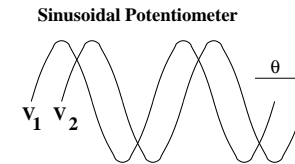
## 23.3 Position Measurement

- Voltage is proportional to position.
  - Linear or rotary potentiometer can be used.
  - Accuracy limited by that of potentiometer.
  - Accuracy limited by voltage source.



## 23.4 Measure Rotary Position

- Two sinusoidal potentiometers are used.
  - $V_1 = V_0 \cos(\theta)$
  - $V_2 = V_0 \sin(\theta)$
- This can be done magnetically too.
  - Requires complex analog detection.
  - Is called a Resolver.

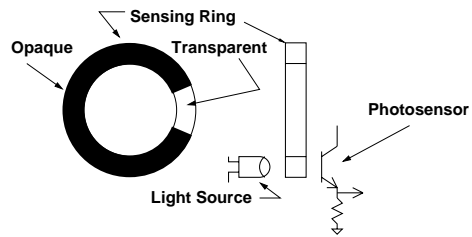


- These are still analog.
  - Accuracy limited
  - Subject to drift
  - Complex calculations

## 23.5 Digital Measurement of Position

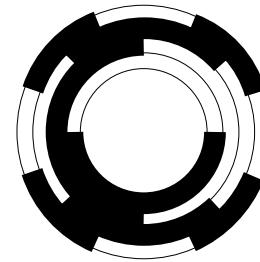
### □ Sense light transmission

- Typically through a transparent sector.
- Gives a reading over a range of positions.
- May need a lot of sensors.



## 23.6 Another Digital Method

### □ Low Resolution Absolute Sensor



Here is a 4-bit (22.5 degree) resolution wheel.  
One source per sensor bit.

Can make these wider.  
Resolution is

$$\frac{360^\circ}{2^N}$$

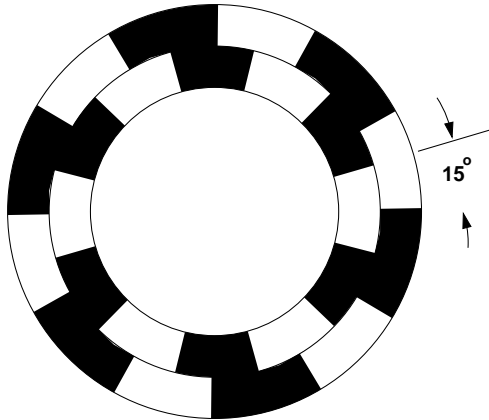
Use a Gray Code to eliminate chatter.

0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

## 23.7 Two-Phase Encoder

### □ Two Source-Sensor Sets

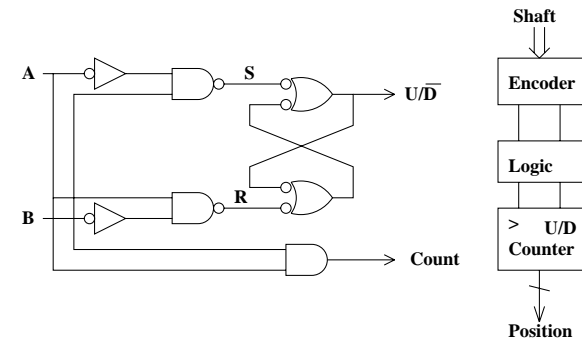
- Offset by half sector width.
- This example has 30 degree sectors.
- And 15 degree resolution.



## 23.8 Use of Encoder

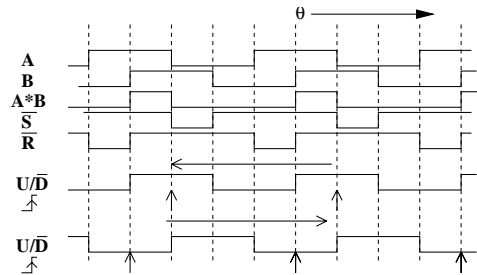
### □ This circuit generates:

- An Up/Down signal (CW or CCW)
- A Count Signal (Edge Encountered)



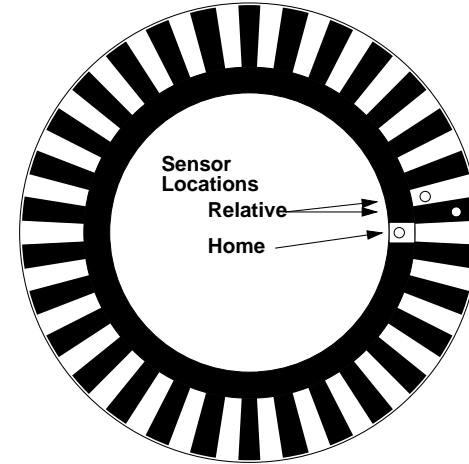
### 23.9 Waveforms

- A and B are sensor signals.
- Rotating one way, Count Edge is when U/D is high.
- Rotating other way, Count Edge is when U/D is low.



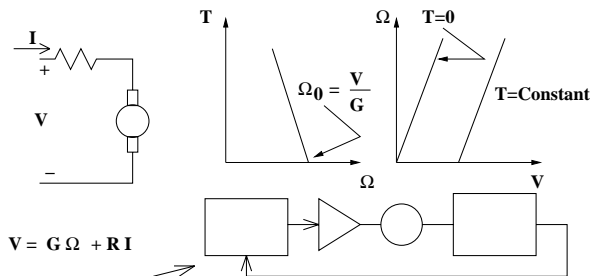
### 23.10 Another way of doing an encoder

Displace Sensors by 1/2 band.  
Add a 'Home' row to sense an absolute position.



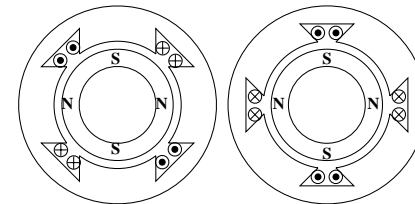
### 23.11 Motors

- Simple servomechanisms are made with DC motors.
  - Servo Strategy:
    - Command torque by setting current.
    - Measure speed.
- DC Motor Model is simple:
  - Resistance in series with a voltage source.
  - Motor produces torque.
  - Mechanical system (controlled system) determines speed as influenced by torque.
- Permanent Magnet DC Motors are very commonly used:
  - 'Back Voltage' proportional to speed.
  - Torque proportional to current.
- Running open loop:
  - There is a 'zero torque' speed and torque proportional to difference from that speed.



### 23.12 Stepper Motors

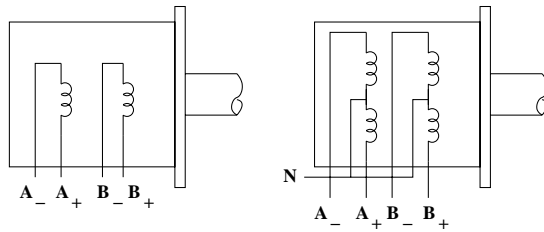
- Digital motors
- Two "stacks" (phases)
- Usually biased by permanent magnets
- Move a discrete distance per 'step'
- This is an axial view cut through both of two sections.



### 23.13 Stepper Motor Windings

□ Two distinct 'phases'

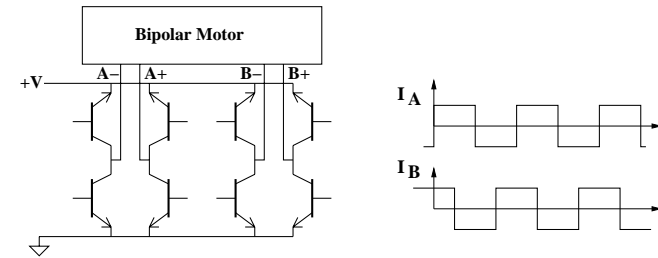
- May be driven as distinct windings,
- Or may be driven as 'bifilar' windings.
- Bifilar is easier but less efficient.



### 23.14 Bipolar Winding

□ Driven by 'H-Bridges' of transistors.

- Can put current through windings in either direction.
- But note upper transistor gating is tricky.
- Uses all of the winding.

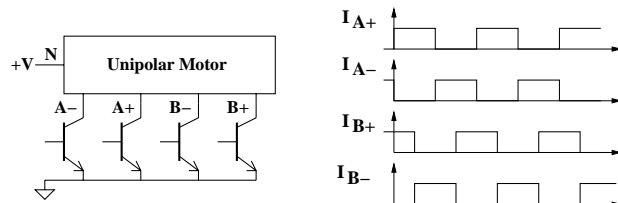


### 23.15 Bifilar Winding

□ Driven by four transistors to ground.

□ Note the center of winding is held 'high',

- Transistors are between winding and ground.
- NPN bipolars work well.
- Transistor gates are easily handled.



### 23.16 Motors Run in Either Direction.

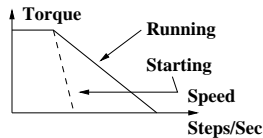
□ Current drive strategy

- Bipolar winding on left
- Bifilar winding on right

Step	$I_A$	$I_B$		Step	$I_{A+}$	$I_{A-}$	$I_{B+}$	$I_{B-}$	
1	+	+	Clockwise ↑ Counter-Clockwise	1	1	0	1	0	Clockwise ↑ Counter-Clockwise
2	+	-		2	1	0	0	1	
3	-	-		3	0	1	0	1	
4	-	+		4	0	1	1	0	

## 23.17 Dynamics are Important.

- Stepper can hold a certain torque.
- Stepper can carry more torque at low speed.
- At high speed, torque must be de-rated.
- Motors draw CURRENT!
  - Need to make sure devices can handle it.



Must sometimes 'ramp up' speed.  
Holding torque is limited by heating and by saturation.

## 23.18 Quiz 2 Review

- Friday, April 13, 2001
- 12:00 to 1:00 (class time)
- Room 50-340 (Walker)
- 2 handwritten 'crib sheets' allowed.

### General Topics

- Finite State Machines
- PALs and CPLDs
- VHDL
  - Entity and Architecture declarations
  - Process as a wrapper around sequential statements
  - Assignments and logical statements
  - if-then, elsif-then, else, endif
  - case-when-end case
  - How to code FSMs

## 23.19 Quiz 2 Review

### General Topics (continued)

- SRAM (e.g., 6264)
  - Read/Write functions
  - Chip Enable
  - Write Pulse
- ROM (e.g., 28F256A Flash Memory)
- Microcoded Control Units (MCUs)
  - Use of a counter (e.g., '163) as a sequencer
  - 2-instruction machine
    - ▶ Use of MSB to decode
    - ▶ IF Condition Jump Address
    - ▶ Assert (or Do)
- Chips (concepts) to be familiar with:
  - MSI stuff - AND, OR, NAND, NOR, XOR, ...
  - Clocked D, T, J-K flip-flops and registers
  - '163, '169, and '393 counters
  - MUXs and Decoders
  - Shift Registers
  - Comparators
  - SRAMs and ROMs
  - PALs and 374i CPLD

## 23.20 Design Rules

- Use Modularity.
  - Small subsystems are easier to design.
  - Subsystem definition is important.
- Design for testability.
  - Design subsystems so they will run alone.
  - Design in break-points.
- Use Don't Cares to simplify combinational logic.
  - Avoid trap states (check use of Don't Cares).
  - Decade counters and other FSMs may have strange sequences for illegal states.
- Do your logic design carefully, and first.
  - Make up block (functional) and circuit diagrams (with pin numbers).
  - Use logic symbols appropriate to algebraic representation.
  - Use names appropriate to assertion level.

## 23.21 Design Rules (continued)

- Avoid problems from 'glitches'.
  - Gate delays can (and do) cause 'glitches'.
  - Glitch-free combinational logic requires (but is not guaranteed by) single input changes.
  - CLK, G, /PR, /CL inputs must NOT have glitches.
  - Carry Output from counter (e.g., '163) can have glitches.
  - Be sure that combinational output is stable before it is sampled by CLK.
  
- Use proper timing.
  - Clock period < MAX(FF delay, Input changes) + CL delay + Setup.
  - Obey FF timing restrictions - setup and hold times, clock width.
  - Don't derive asynchronous clear from flip-flops to be cleared.
  - Operate all edge-triggered flip-flops from the SAME clock edge.
  - Beware of clock skew. Use a tree structure to expand the clock.
  - Change inputs only (just) after the clock edge.
  
- Be careful of multi-ended wiring paths.
  - Avoid tri-state bus contention.
  - Account for turn-off delays.
  - Don't overload outputs (observe fanouts).

## 23.22 Design Rules (continued)

- Be careful about asynchronous events.
  - Synchronize all external inputs.
  - Asynchronous event should only change one flip-flop.
  - Consider pulse width carefully. Does your application need a narrow pulse or a sustained level?
  - Beware of bouncing switches.
  - Use monostables sparingly (if at all).
  
- Use memory properly.
  - Avoid high-Z address to SRAM when CE is asserted.
  - Avoid address changes when write pulse is true.
  - Make sure your write pulse is 'clean' (i.e., no glitches).
  
- Wire properly.
  - Keep wires short.
  - Wire all inputs (even unused ones).
  - Use bypass (decoupling) capacitors. (They are already on your kit.)
  - Alternate grounds with signals in flat cables.
  - Use twisted pairs (different colors!) between kits.
  - Don't overload your power supply.
  - Use a separate power supply for motors.

## 23.23 Design Rules (continued)

- Use debugging strategy.
  - Write short test programs.
  - Debug modules systematically.
  - Check for power supply and ground on every chip.
  - Is every pin wired? Why not?
  - Thermal debugging may help detect bus contention.
  - Use a 'scope! Check valid logic levels and power supply.
  - sssse your logic analyzer for checking sequencing.
  
- Gating the clock
  - Don't!
  
- If you must, do it carefully.
  - Account for clock skew. This increases effective setup and hold times.

## 23.24 How to Make Your Project Work

- Read all of the handout.
  - It is 'old' but all of it is good advice.
  
- Sections that are particularly relevant are:
  - Wiring Errors
  - Care and Feeding of the Power Supply
  - Unused Inputs
  - Behavior of Ungrounded Parts
  - Tri-State Logic Signals
  - Handling CMOS Parts
  - Wire Routing
  - Clock Distribution
  - Gating the Clock
  - RAM Write Pulses
  - Synchronizer Errors
  - Testing Strategies
  - Driving High Current Devices