Position and Velocity Measurements

- Important in many embedded applications, especially those in which the microprocessor must interface to mechanical hardware.
- Question: How many examples can you name?
  -
  -
  -
- Many microcontrollers have special features that expedite such measurements.
- The MPC5553 has an enhanced Time Processing Unit (eTPU) that is essentially a special purpose microcomputer that operates simultaneously with the CPU. The eTPU performs special functions that otherwise would require CPU interrupt service.
  - Quadrature Decoding
  - many others...
- Analog vs. Digital Measurement Technology
  - Signal/Noise ratio (S/N)
  - Dynamic Range (largest vs. smallest measurements)
- Information about technology for position and velocity measurements can be found in many textbooks, including [1], [2], and [4]. Reference [1] was used primarily in developing this set of notes.

Analog Velocity Measurement

- Tachometer: produces a voltage proportional to velocity (usually rotational velocity)
- same operating principle as DC motor; in fact, it is possible to use a DC motor as a tachometer. (will describe when we discuss DC motors)
- tachometer performance is often limited by *noise* (e.g., brush noise)
- at low velocities S/N ratio is poor, and it is difficult to determine when velocity is zero (i.e., when the mechanical system is stopped)

Analog Position Measurement

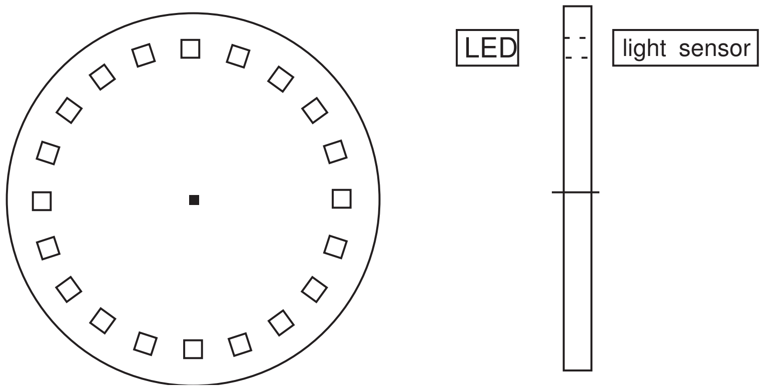
- Potentiometer: produces a voltage proportional to distance from a reference point
- Example: Measure distance of a cart from one end of a track
- Since there is a sliding mechanical contact, noise, dirt, humidity, and mechanical wear limit precision.
- The maximum range is also limited!

Optical Encoder

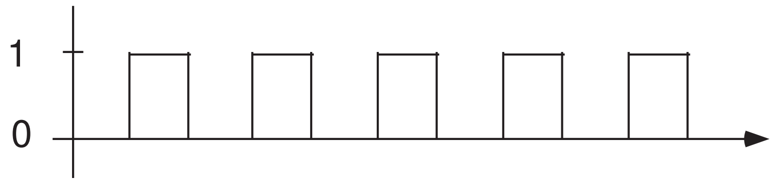
- Often used for digital position and velocity measurement
- Two types
  - absolute encoders: gives actual position
  - incremental encoders: gives change in position
- Usually encoders measure *angular displacement*
  - can be used to measure rectilinear position
  - computer mouse (2-dimensional position!)
- Will first discuss incremental encoder, because we will use in lab.

Incremental Encoder

- a wheel with little windows (front and side views):



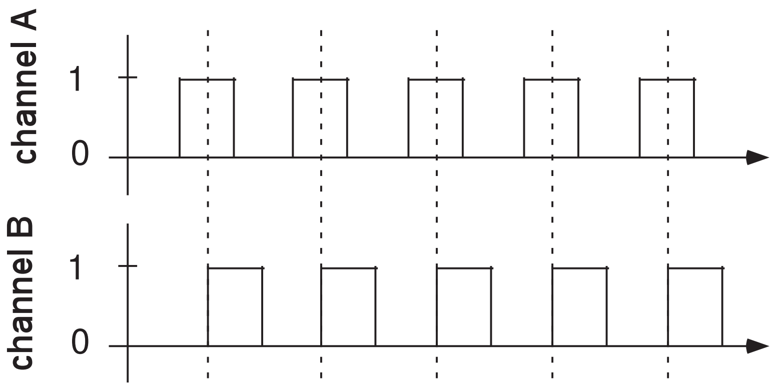
- As wheel rotates, the photocell generates a digital signal:



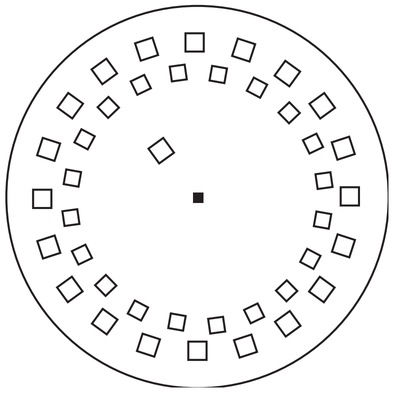
- If windows and dividers are of equal width, then the "on" and "off" times are of equal duration for constant rotation rate.

Quadrature Decoding

- Encoders actually generate a *quadrature* signal, consisting of two square waves that are 90° out of phase.

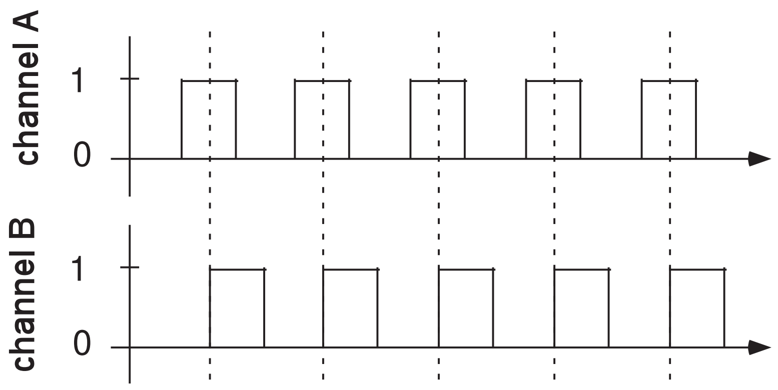


- This may be done using a one track encoder with two light sensors placed side by side.
- Or a two track encoder:

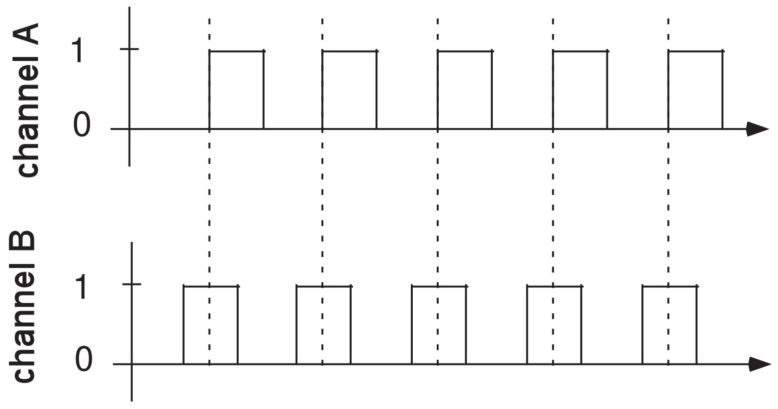


Direction of Rotation

- Direction of rotation is determined by the *phase difference* between the two signals
- For example, if Channel A leads Channel B, then the encoder is rotating (say) CCW



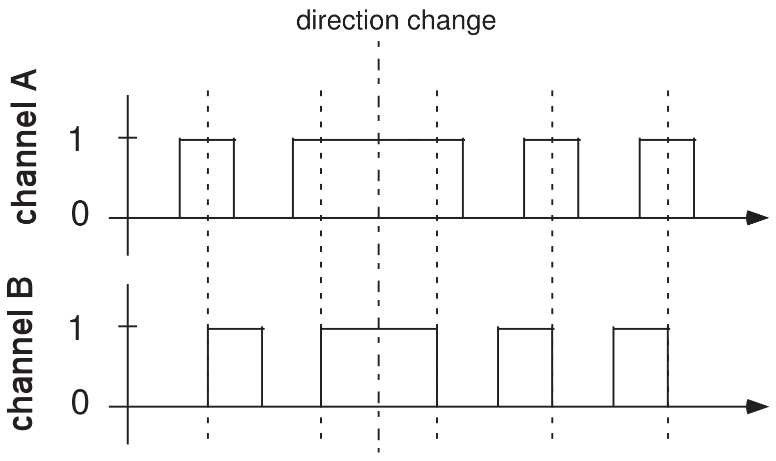
- If Channel B leads Channel A, then the direction of rotation is CW



- Most encoders have an extra track, with just one window, that may be used for alignment to determine absolute position at startup.

Direction Changes

- Two consecutive transitions on the same channel indicate a change in direction:



Encoder Operation Modes

- The idea of quadrature decoding is to generate a pulse train that may be used to increment or decrement a counter. The counter is incremented if the phase difference between channels indicates CCW movement, and decremented otherwise.
- Several types of decoding
  - $1X$: the counter is incremented or decremented only according to leading edges of Channel A. (Lowest resolution)
  - $2X$: the counter is incremented or decremented according to both leading and trailing edges of Channel A. (Higher resolution)
  - $4X$: the counter is incremented or decremented according to leading and trailing edges of Channels A and B. (Highest resolution)
  - The QD eTPU function on the MPC5553 has three modes: slow, normal, and fast [3].
    - slow: the same as $4X$ mode
    - normal: the same as $4X$ mode except that direction is not determined
    - fast: the same as $1X$ mode, except that direction is not determined

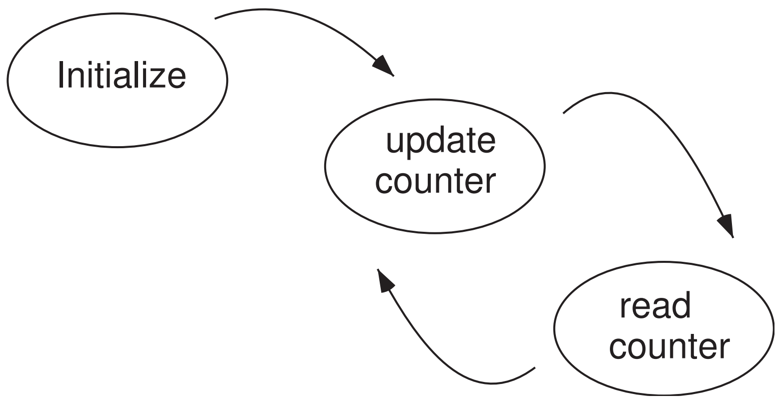
Resolution of Encoder

Question: How to determine resolution in degrees of rotation/pulse?

- Consider $1X$ mode.
  - Suppose there are N windows/track.
  - Then there are N pulses/revolution.
  - Each pulse corresponds to a displacement of $360^\circ/N$.
- Consider $2X$ mode.
  - Suppose there are N windows/track.
  - Then there are $2N$ pulses/revolution.
  - Each pulse corresponds to a displacement of $360^\circ/2N$.
- $4X$ mode?
- Example: $N = 6$
  - $1X$ mode: resolution $= 60^\circ$/pulse
  - $2X$ mode: resolution $= 30^\circ$/pulse
- You will read the documentation to determine the resolution of the encoder used in the EECS 461 lab.
  - How to account for the gear ratio?

Issues

- If the encoder is moving too fast, then the sharp edges of the square wave may be so distorted that a transition escapes detection.
  - a problem with the physical time constants
- Noise may cause spurious transitions
  - ignore short duration pulses
  - ignore pulses that result in invalid transitions
  - a spurious pulse that occurs on only *one* channel will cancel itself out
  - spurious pulses that occur on both channels can result in errors
- Must insure that when the CPU reads the counter, it does not obtain a spurious value due to reading it while it is being updated.
  - update counter state vs. read counter state



  - Even in "read counter" state, no pulses can be missed!

Encoder Overflow

- If not read sufficiently often, the encoder may overflow.
- Two approaches
  1. The counter may be set to zero each time it is read.
     - Each reading is an increment from the previous, and may be added to previous reading (in any word size) on the CPU
  2. The counter is free running, overflowing or underflowing depending on direction
     - Use 2's complement arithmetic to subtract previous from current readings to obtain position increment.
- In either case, counter must be read before ambiguity occurs

Two's Complement Arithmetic, I

- A way to represent signed decimal numbers in n-bit binary form.
- Case 1: Nonnegative Number
  - (i). convert unsigned portion into $(n-1)$-bit binary
  - (ii). set leading bit to 0
- Case 2: Negative Number
  - (i). convert unsigned portion into $(n-1)$-bit binary
  - (ii). take 1's complement (change all 0's to 1, all 1's to 0)
  - (iii). add 1
  - (iv). set leading bit to 1
- Example: $n = 8$. Represent 6 in 2's complement
  - (i). $6 \rightarrow 0000110$
  - (ii). set leading bit to 0: $0000110 \rightarrow 00000110$
- Example: $n = 8$. Represent -6 in 2's complement
  - (i). $6 \rightarrow 0000110$
  - (ii). take 1's complement $0000110 \rightarrow 1111001$
  - (iii). add 1: $1111001 + 0000001 = 1111010$
  - (iv). set leading bit to 1: $1111010 \rightarrow 11111010$

Two's Complement Arithmetic, II

- 2^{n-1} nonnegative, and 2^{n-1} negative numbers
- Example: $n = 3$

decimal	binary
-4	100
-3	101
-2	110
-1	111
0	000
1	001
2	010
3	011

- Note that
  - positive numbers always have "0" as the most significant bit
  - negative numbers always have "1" as the most significant bit

Sign Extension

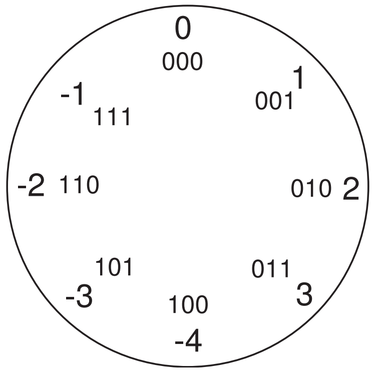
- Sometimes it is necessary to convert an n-bit two's complement number into a $2n$-bit number
- It is necessary that the leading bit of the n-bit number be "extended" so the the $2n$-bit number has the correct sign
- Example: $n = 2 \rightarrow n = 4$

decimal	2-bit	decimal	4-bit
		-8	1000
		-7	1001
		-6	1010
		-5	1011
		-4	1100
		-3	1101
-2	10	-2	1110
-1	11	-1	1111
0	00	0	0000
1	01	1	0001
		2	0010
		3	0011
		4	0100
		5	0101
		6	0110
		7	0111

- Note: simply converting the 2-bit number to a 4-bit number by adding leading zeros will yield the wrong answer!

Encoder Overflow

- How to calculate position change?
  - Consider a 3-bit encoder (CCW = forward)



- Example 1
  - Suppose previous reading is 101
  - Current reading is 000
  - Subtract: $000 - 101 = 011 \Rightarrow 3$ forward steps

- Example 2
  - Suppose previous reading is 101
  - Current reading is 011
  - Subtract: $011 - 101 = 110 \Rightarrow 2$ backward steps

- Necessary to read counter often enough that directional information isn't lost (at most 3 forward or 4 backward steps)

- Example 3
  - Suppose previous reading is 101
  - and current reading is 001
  - Subtract: $001 - 101 = 100 \Rightarrow 4$ backward steps
  - Ambiguity: It could also be 4 forward steps....

  $\Rightarrow$ Recall our discussion of aliasing and the rotating wheel!

Velocity from a Position Encoder

- Using counter
  - If counter is read (sampled) at regular intervals, then position increment is proportional to average velocity over the period between samples.
  - Problem: Low velocity may read inaccurately or even zero.

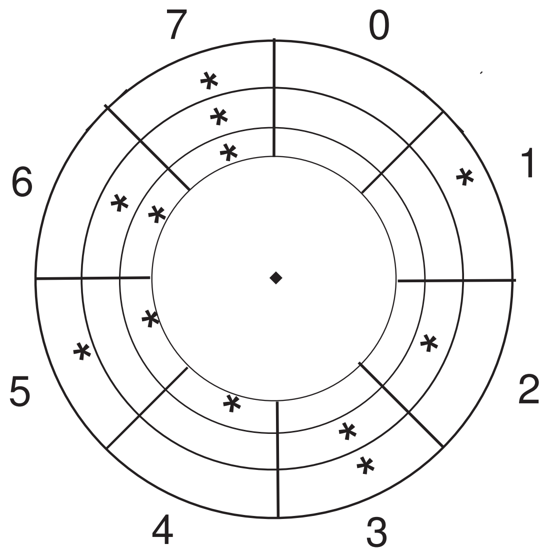
- Using frequency of pulses
  - Count *frequency* of pulses: the number of pulses per time interval
  - Same problem at low velocity

- Using period of pulses
  - Count *period* of pulses: the time between consecutive pulses
  - Problem: at high velocity, too little time between pulses

- Problems of period counting and frequency counting are complementary

Absolute Encoders

- Gives absolute (not relative!) position directly in digital form
- Uses several tracks: n tracks for an n-bit absolute encoder yields 2^n sectors, with resolution $(360/2^n)^\circ$.



- Problem: multi-bit changes
  - If position is recorded while in transition, then it may be wildly wrong!
  - Since it is impossible to know when the wheel is going to move, can never guarantee that it won't move while being read
  - Sensors for each track must be carefully aligned, otherwise, some could be reading one sector, and others could be reading adjacent sector
  - Resolution: use Gray code instead of natural binary

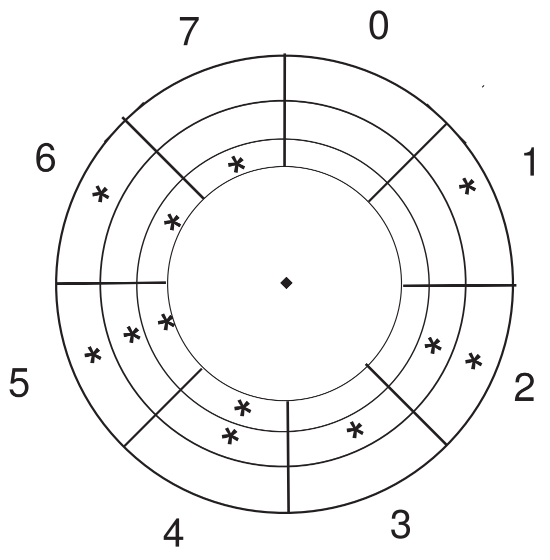
Gray Code

- Idea: Represent consecutive decimal numbers using binary numbers that differ in only one digit
- Example: 3-bit Gray code

decimal	natural binary	Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Absolute Encoders with Gray Code

- Consecutive sectors of encoder differ only in one bit.



- must translate back into natural binary....

References

[1] D. Auslander and C. J. Kempf. *Mechatronics: Mechanical Systems Interfacing*. Prentice-Hall, 1996.

[2] W. Bolton. *Mechatronics: Electronic Control Systems in Mechanical and Elecrical Engineering, 2nd ed.* Longman, 1999.

[3] M. Brejl, M. Princ, and A. Butok. *Using the Quadrature Decoder (QD) $eTPU$ Function*. Freescale Semiconductor, Application Note AN2842, April 2005.

[4] C. W. deSilva. *Control Sensors and Actuators*. Prentice Hall, 1989.